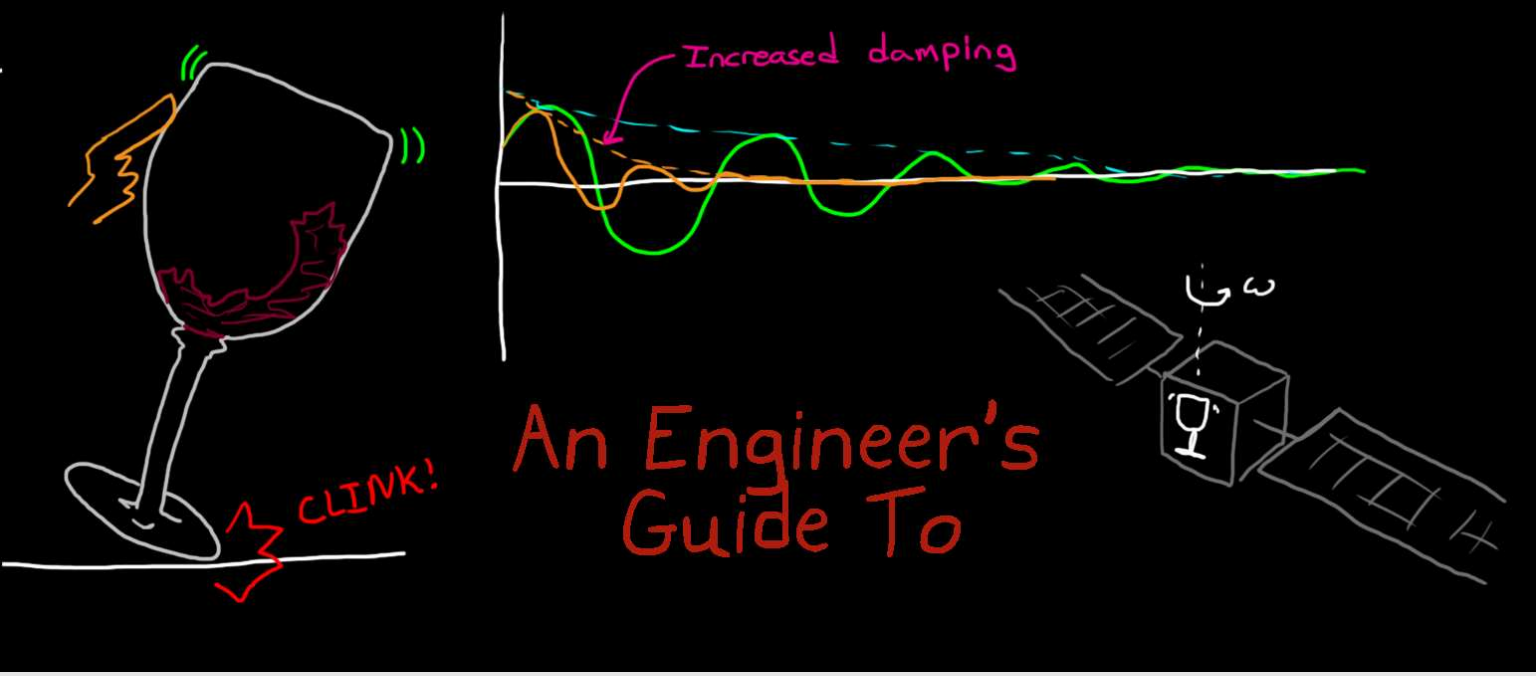
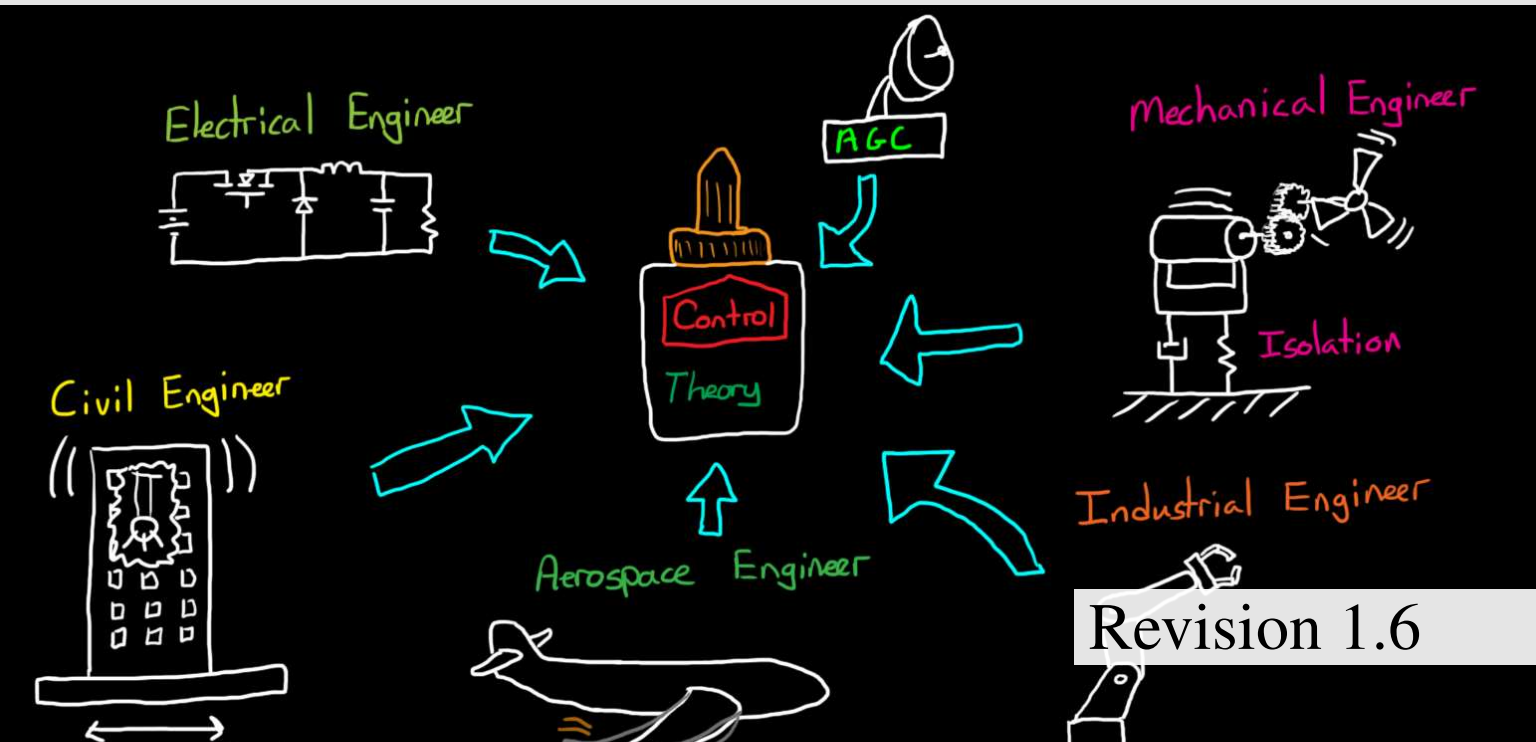


The Fundamentals of Control Theory

An Intuitive Approach from the Creator of *Control System Lectures* on YouTube

Brian Douglas



Revision 1.6

Revision, 1.6

Printing Date, July 11, 2019

Contents

PREFACE

Welcome to the Fundamentals of Control Theory! This book is the direct result of my online video lectures on control system theory and the overwhelming positive feedback and encouragement I've received from my viewers to write a book. I started my YouTube channel (https://youtube.com/ControlLectures) because I was frustrated by the lack of straightforward and easy to understand videos on the topic and felt that what some students needed was a more practical and intuitive approach to understanding the material. This book is an extension of that idea.

I'm releasing this book one section at a time. This allows me to get the early chapters out there helping people right away while being able to have your reactions and responses influence the later chapters.

So with that in mind, as I write this book there are four goals I hope to accomplish that I think will make it a valuable resource for any aspiring controls engineer.

1. **Provide an intuitive understanding -** I want to start by saying there already exist several fantastic control system textbooks. Therefore, I don't think I would be able to write a useful book in this crowded field by presenting the same information in the same formal format. So I'm not going to try to duplicate them, instead I'm creating a book that is a bit different. The language is a little less formal - it's written as though we're having a conversation - and the mathematical proofs are a little more casual. However, I claim that what you'll learn from this book is just as useful as the existing textbooks because you'll gain an overall understanding of the problem and how to approach it.
2. **Update the book frequently -** One of the luxuries of making an eBook is that I can distribute updates quickly and cheaply. I am approaching this book more like a distribution of software, where bug fixes and minor layout changes can be updated and distributed as a point release

(minor update) rather than a major new edition. With this model I can fix bugs and add content on a regular basis so that readers will always have the most up to date revision.[1]

3. **Allow the readers to participate in improving the book -** How many times have you come across an error in a textbook or a really confusing explanation and wished you had a way of providing feedback easily to the author? I want to hear that feedback! It is your feedback that will drive the quick point releases and help me create the most useful textbook possible. I ask that you please email me at controlsystemlectures@gmail.com any time you come across errors in calculations, vague or confusing explanations, and missing content in the book so that I can fix it and it won't confuse the next round of readers. Plus, if you want, I'll add your name to the list of contributors at the end of each chapter that you help fix!
4. **Make the book as inexpensive as possible -** Lastly, college textbooks are expensive and if I want this book to really help students all over the world then it needs to be affordable. I understand that students don't have much money and so having to buy several $180 books each semester is not high on your list of fun activities. That is why I'm releasing this book under the Creative Commons License and giving the book out for free. If you want to support me and my efforts I only ask that you provide feedback on the book, watch my videos, and share them and the book with as many people as possible.

Engineering problems are inherently multi-disciplinary and so you have your choice of learning any number of specialized fields that will allow you to contribute to a project. But I think the best reason to learn control theory is that it is the glue that combines all other engineering fields and by understanding the fundamentals of control theory it opens the door for you to understand all of those other fields at a more basic level. It is actually a fascinating subject and through this book I hope to infect you with the same enthusiasm for the subject that I have.

Chapter 1 describes the control problem. This chapter sets the stage for what we're trying to accomplish as control system engineers and defines the terms that we use throughout this book.

[1] This hasn't really panned out, unfortunately.

Chapter 2 introduces a way of describing a system mathematically using transfer functions. This chapter builds up the fundamental concepts behind transfer functions and sets the foundation that we will build on going forward.

Chapter 3 explains the properties of creating and manipulating block diagrams. We will use block diagrams to simplify concepts and systems throughout this book and having a firm grasp of how to use them to help you solve problems is critical for any control engineer.

Once written, the rest of the book will cover transfer functions, how we represent systems with block diagrams, and concepts like system stability, time, frequency, discrete domains, and system identification. We'll then cover how we use specialized plotting tools like Root Locus, Nyquist plots, and Bode plots to analyze and understand our system. Later chapters will describe compensation techniques like lead and lag, loop shaping, and PID.

By the end of this book I hope you realize that control system theory is so much more than *just* tuning a PID controller or getting an inverted pendulum to stand upright. It's building models of your system and simulating it to make predictions, it's understanding the dynamics and how they interact with the rest of the system, it's filtering out noise and rejecting outside disturbances, it's designing or selecting proper sensors and actuators, and it's testing your system to ensure it'll perform as expected in an unexpected environment.

Now before you proceed any further I want to thank you for reading this book[2]. I hope you gain a better intuition into control theory and ultimately you become a more well-rounded engineer.

Brian Douglas

https://www.engineeringmediallc.com

[2] and the preface! Who reads the preface anyway?

PART 1

The Big Picture

1 The Control Problem

In this chapter, we'll get an overview of the big picture problem that we're trying to solve as control system engineers. This will give context to everything we'll cover in later chapters and in doing so I think will help you understand why you are learning the topics presented in this book.

1.1 What is a system?

To begin we describe exactly what a system is. The concept is really straight forward but since the term is so generic we tend to apply the word to describe just about everything. This can get confusing to someone new to the field when we refer to something called the control *system* which is then used to control the actual *system* and when put together the two parts make yet another larger *system*. As someone learning control theory the question becomes *what system am I working on?* To answer this let's start with the definition and work from there.

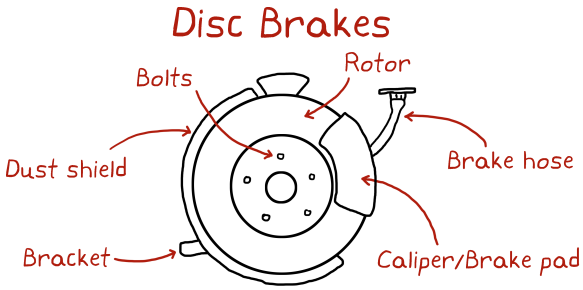
A **system** is a collection of interconnected parts that form a larger more complex whole.

Engineering projects are typically complex. Dividing complex projects into smaller pieces, or systems simplifies the problem because it allows people to specialize in their functional area and not have to be a generalist in all areas. Therefore, as a specialist, you might be working on just one of the interconnected parts that form the entire system. However, there might be many layers of complexity such that the small part you are working on is actually a complex system in its own right!

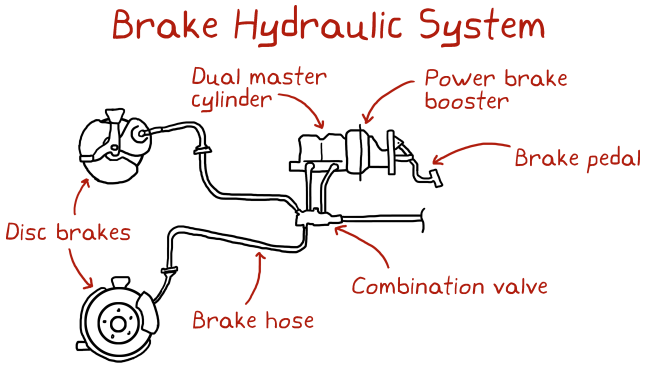
The same is true for specialists in control theory. As a control engineer, your goal is to create something that meets the functional or performance requirements you set for the project. In general, we refer to the collection of the interconnected parts that are created specifically to meet these requirements as the control system. For any project other than the very simplest ones, however, the control system again might be a collection of interconnected parts that require specialists like sensor experts, actuators experts, digital signal processing experts, or state estimation experts.

To illustrate this, let's imagine that you have accepted a job at an automotive company and you will be working on the braking system. At first glance, you might suspect that you will be involved in all parts related to slowing the vehicle. However, there are many parts to the braking system on your car and it takes many different specialists to design the complete product.

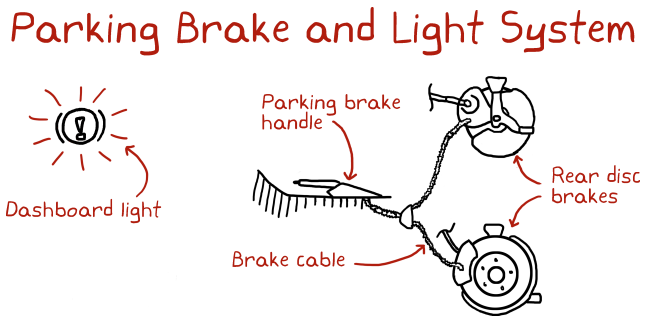
The most obvious component is the disc brake assembly in each wheel. This is the part that is actually converting the car's kinetic energy into heat energy and reducing the vehicle's speed. Yet the disc brakes are small systems on their own because they are made up of rotors, calipers, brackets, shielding, fasteners, and hoses which allow the disc brakes to function correctly.



Engaging the brakes requires the brake hydraulic system which is responsible for transferring the pressure applied by your foot at the brake pedal through the power booster, dual master cylinder, and the combination valve and finally to the brake calipers at each of the four wheels.



There is the mechanical parking brake system that bypasses the hydraulic system with a secondary cable path to the brakes. The brake light system is responsible for lighting the tail lights and that annoying dashboard light that tells you the parking brake is engaged.



Finally, there are any number of electronic brake control systems that override the human input to keep the vehicle from skidding on slick surfaces or a distracted driver from crashing into the car in front of them.

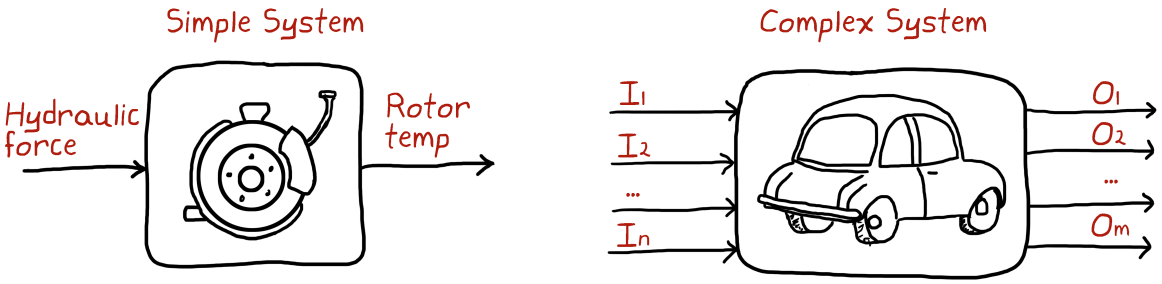
All of these smaller systems - the brakes, hydraulics, parking brake, lighting, and electronic controls - are the interconnected parts that form the larger and complete braking system. Furthermore, the braking system is just one of many interconnected parts that create the car itself.

As a control specialist in the brake department you might be responsible for writing and testing the algorithm for the electronic brake control system but have very little impact on, say, the cable routing for the parking brake.

Defining different systems allows complex projects to exist but it does create this potential confusion of everything being called a system. To mitigate this, depending on the field you work in, there is usually a term for each of the different hierarchical levels of complexity in a project. For example, a couple of parts creates a component which in turn creates a subsystem which then finally creates a system. I'm not going to try to define where the boundaries are between each of those because every industry and company do it differently. However, it is important that you recognize this and are clear on what someone is specifically referring to when they ask you to design a controller for some system. In this book, I will try to be explicit when I say system because what I'm referring to will change based on the context of the problem.

In general, we will represent any system graphically as a box. Arrows going into the box represent external inputs acting on the system. The system then responds over time to these inputs to produce an output - which are arrows leaving the box.

Typically we define the system in the box with a mathematical model that describes its equations of motion. At the moment we don't need to worry about the math, the most important thing is that we understand what this box means physically. For example the *system* could be really simple like a single disc brake with the inputs being the force of the hydraulic fluid and the output being the temperature of the rotor. Or the *system* could be complex like the entire car and have hundreds of inputs and thousands of outputs.



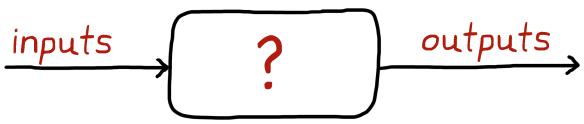
In both cases, our graphical representation would look similar; a box with arrows going into it and arrows coming out of it. Later we will string several systems (boxes) together to create complex block diagrams. These block diagrams will contain the relevant interconnected parts of an even larger system. For the next section, however, this single box representation will give us some insight into three different types of problems that we'll face throughout this book and as practicing engineers.

1.2 The three different problems

You will notice that there are three parts to our simple block diagram; there is the box itself which represents a system, the inputs that are driving the system, and the outputs that the system generates. At any given time one of the three parts are unknown to you, and whichever part you don't know defines the problem that you are trying to solve.

1.2.1 The system identification problem

As a student, you are usually given a mathematical model of your system at the start of your problem and then asked to perform some analysis of it or asked to design a control system around it. However, as a practicing engineer you won't always be given a model of your system[1], you'll need to determine that yourself. Determining the mathematical model is done through a process called system identification.

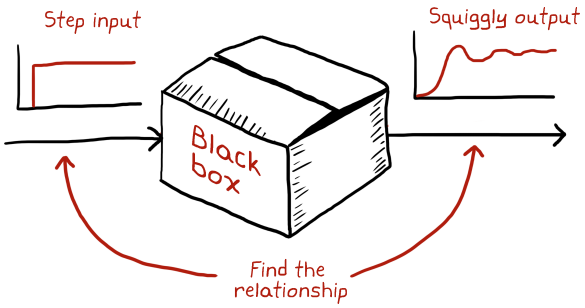


You might be doing system identification if you find yourself asking the following questions:

[1] In fact, you'll almost never be given a model since what you're working on is very likely the first of its kind

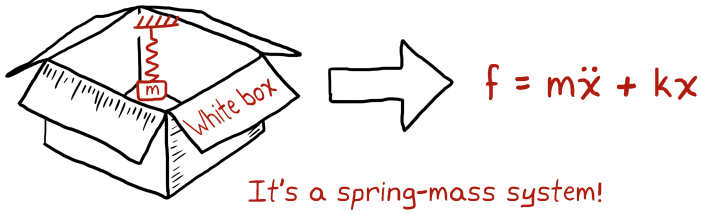
- How can I model the system that I'm trying to control?
- What are the relevant dynamics for my system (what should I model)?
- What is the mathematical equation that will convert my known inputs into my measured outputs?

There are at least two ways that we can answer these questions. The first is referred to as the black box method. Imagine you are given a box that you can not open but you are asked to make a model of what is inside. You could subject what is in the box to various known inputs, measure the resulting outputs and then infer what's inside the box based on the relationship between the two.



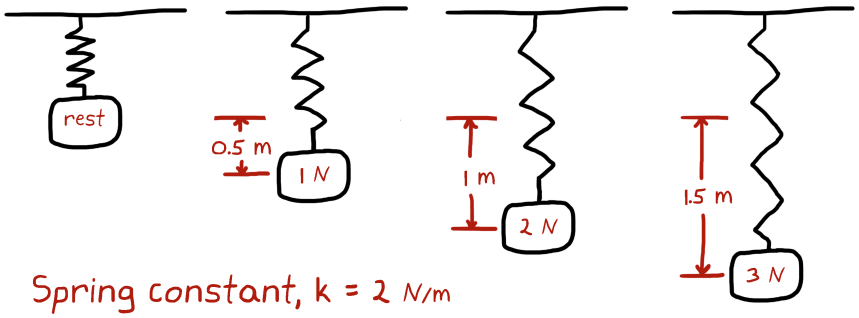
The second way to perform system identification is referred to as the white box method. Imagine now you are able to see exactly what is inside the box - all of the electronics, mechanisms, and software. Knowing the components of the system you can write the mathematical equations of the dynamics directly. This is exactly what you're doing when you use Newton's equa-

tions of motion or you are determining the equations of motion based on the energy in the system.



You might argue that the white box method is simpler than the black box method because you don't need to run a test to write out the equations of motion, but that's not always true. Even with this white box method, you may need to apply known inputs to the system and measure the outputs so you can calculate any unique parameters for your system. For example, you might need to model a linear spring - the equation of motion is well known - but you will have to perform a stretch test to determine the exact spring constant for it[2].

[2] I know you might be thinking, 'but I've been given the spring constant from the manufacturer so I don't have to perform that test!' And to that I have two responses, 1) if the parameter is really important to you, are you really going to trust the manufacturer's datasheet? And 2) if you are fine with the accuracy stated in the datasheet then this is a case where you *were* given enough information to write a model of your system without a test.



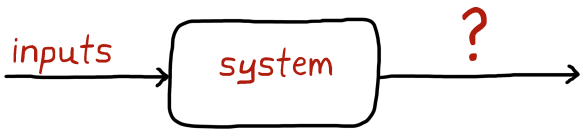
To test for the spring constant, you could hang a known mass hang from the spring and measure how much much the spring stretches from its rest length. The input into the system is the force that the mass is exerting on the spring and the output is the stretched length of the spring. We can tell from the relationship between the input forces and the output lengths that the spring constant is $2\ \frac{N}{m}$.

Often, you'll find that when you are trying to write out a model of your system, you'll use both the white box method (writing equations directly) and the black box method (determining unique aspects of your system through testing). System identification is an important aspect of designing a control system, and therefore, this book will devote an entire part to it.

1.2.2 The simulation problem

The simulation problem is predicting how your outputs change given a known set of inputs and the mathematical model of the system. This prob-

lem is interesting because you'll likely spend a lot of your design time in this stage. The trick here is figuring out the set of meaningful inputs and their ranges so that you have a complete idea of how your system behaves.



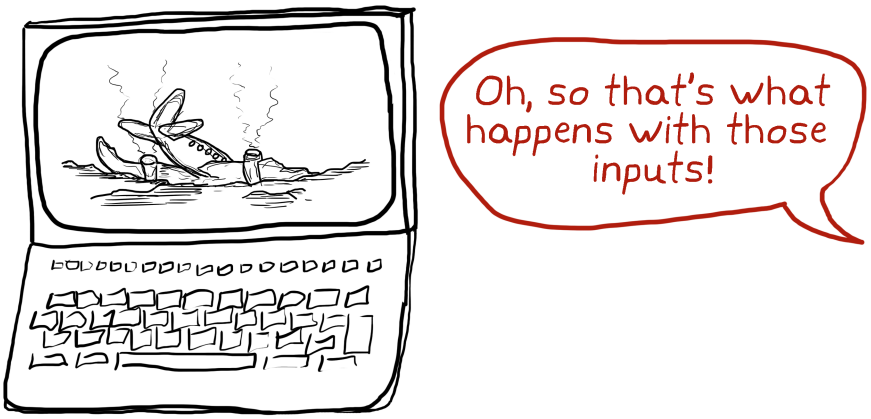
You might need to run a simulation if you find yourself asking the following questions:

- Does my system model match my test data?
- Will my system work in all operating environments?
- How does my system behave if I drive it with potentially destructive commands?

To see how simulation is important, imagine you have a very good model of a passenger airplane and you're designing a pitch control system for it. You want to know how your system behaves across the entire envelope the aircraft will operate in and you're weighing the different ways you can test this. You could use a test airplane and fly it in every operating condition it could expect to see during its life and directly observe its behavior. The problem with this approach is that the operating envelope is huge[3] and flight test campaigns are expensive and so minimizing the amount of flight time

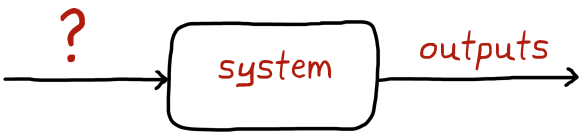
[3] You need to make sure that your pitch control system will function across all operating altitudes, wing angles of attack, air speeds, center of gravity locations, flap settings, and weather conditions.

could save your project a lot of money. More importantly, flight tests can be dangerous to perform if you are trying to push the limits of your system. Rather than risk project budget, and possibly human life, it makes more sense to simulate your system in these extreme situations.



1.2.3 The control problem

Lastly, if we know the model of the system and we know how we want the system outputs to behave then we can determine the appropriate inputs through various control methods. This is the control problem - how do we generate the appropriate system input that will produce the desired output? Control theory gives you the tools needed to answer that question. Without control theory, the designer is relegated to choosing a control system through trial and error.



You might need to design a control system if you find yourself asking the following questions:

- How can I get my system to meet my performance requirements?
- How can I automate a process that currently involves humans in the loop?
- How can my system operate in a dynamic and noisy environment?

This book covers the fundamentals needed to design a control system, but that doesn't mean that it focuses solely on the control problem. In order to design a control system, you usually have to solve the system identification problem as well as the simulation problem. Often, all three have to be solved more than once as you refine and modify your design. Here is one example of how you could use all three in the course of building a controller.

1. You are given a plant and asked to meet some performance goals.
2. You first build a mathematical model of your plant using SYSTEM IDENTIFICATION.
3. After you get a model, you SIMULATE a few simple, known inputs and compare the outputs to the real system. This is a model verification step to make sure it represents the physical plant well enough.

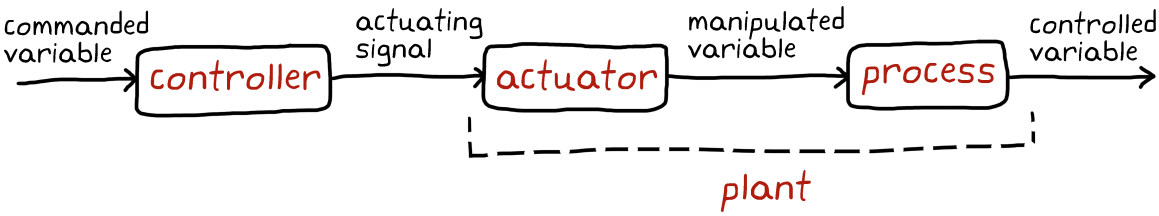
4. You analyze the performance of your plant. After you determine the performance does not meet the desired goals, you design a CONTROL system that will give you the necessary performance.
5. You now SIMULATE a larger number of inputs to cover the operational envelope of your system. You verify the control system design by comparing the simulated results to real test data for a subset of the simulated test inputs.
6. You make adjustments to your CONTROL system based on the simulation results - making the system more robust to a wider range of conditions and environments.
7. Having completed the preliminary control system design, you add more fidelity to your model with better representations of the plant, sensors, and actuators using SYSTEM IDENTIFICATION.
8. You continue cycling through steps each of these steps until you have confidence in your model and your control system.

This book will lay out the fundamental tools needed to perform each of the above steps and in doing so I think you'll find that control theory can be challenging but a lot of fun and very intuitive. Before we move on to learning these tools let's take a step back and describe in more detail why we need control systems in the first place.

1.3 Why do we need a feedback control system?

Let's start with our simple system block diagram, but now the box isn't just any system, it's specifically a system that we want to *control*. From now on we'll refer to the system that is being controlled as the process[4]. The inputs into the process are variables that we have access to and can change based on whichever control scheme we choose and so we'll refer to them as the manipulated variables.

These variables are manipulated by an actuator. An actuator is a generic term that refers to a device or motor that is responsible for controlling a system[5]. Since the actuator is a physical device and is usually embedded within the process itself it can be useful to refer to the collection of both process and actuators as a single system that we'll call the plant[6].



[4]Or if you lack creativity you could just call it the *controlled system*

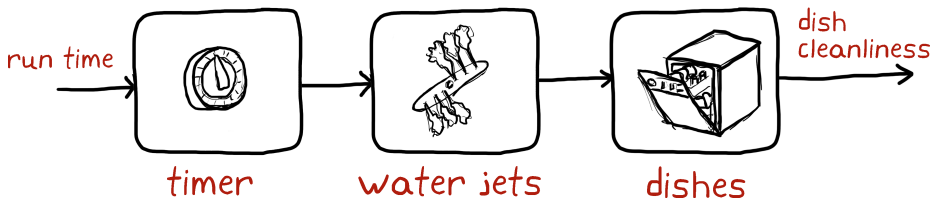
[5]A car's engine and drive train are obvious actuators because they generate the force that manipulates the speed of the car (the process), but actuators can be less obvious as well like a voltage regulator in an electrical circuit

[6]Other textbooks and online resources might use plant and process interchangeably so be aware of that when referring to them. In this book, however, I will stick to this definition.

The actuators are driven by an actuating signal that is generated by the controller. The controller is designed specifically to convert a commanded variable - which comes from someone operating this device or from a higher level control system - into appropriate actuating signals. At this point, we have our first, and simplest, control system. As the operators, we could now select a set of pre-determined commands that we play through our controller. This will generate the resulting actuator commands which in turn affect the manipulated variable which then affects the process in a way that we desire. This type of control system is referred to as open-loop since the inputs into the controller are not fed back from the output of the process.

Open-loop control systems are typically reserved for simple processes that have well-defined input to output behaviors. A common household example of an open-loop control system is a dishwasher. This is an open-loop system because once the user sets the wash timer the dishwasher will run for that set time. This is true regardless of whether the dishes are actually clean or not when it finishes running - that is, the cleanliness of the dishes are not fed back into the dishwasher timer to increase or decrease the wash time. If the dishes were clean to begin with the dishwasher would still run for the prescribed time and if you filled the dishwasher full of pots with baked on grime then the set time might not be enough to fully clean them and would have to be run again. We accept this inefficiency in the run time of

the dishwasher because the starting process (the dirty plates that we want to be cleaned) is generally well known and therefore the time it takes to clean them is pretty consistent.



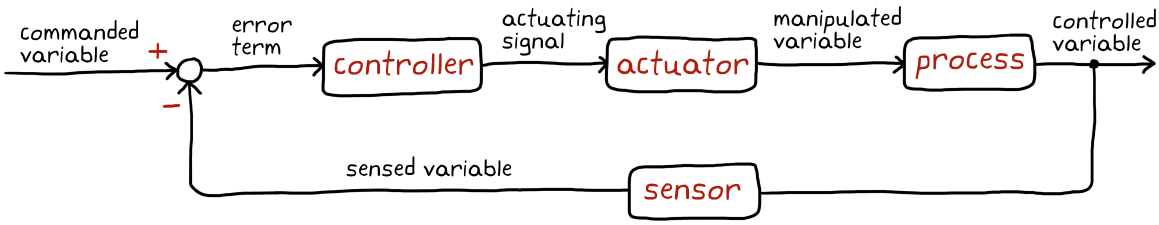
The manufacturers understand though that sometimes you will fill the dishwasher with grimy pots and want it to run for a longer period. Rather than build a complicated closed-loop system they have addressed this problem by adding additional pre-determined commands; that is they add multiple types of cleaning cycles that run for different times and at different temperatures. Then it is up to the user to select the correct set of commands to get the desired effect.

For any arbitrary process, though, an open-loop control system is typically not sufficient. This is because there are disturbances that affect your system that are random by nature and beyond your control. Additionally, the process itself might have variations that you don't expect or prepare for[7]. Process variation and external disturbances will alter the behavior of your

[7]One example of process variation is the change in electrical resistance over temperature. An open-loop control system that works at room temperature might not work when your process is extremely cold or hot due to the variation in resistance throughout your electronics

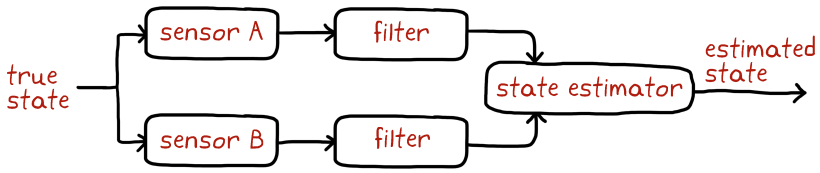
system - typically negatively - and an open-loop system will not be able to respond to them since it has no knowledge of the variation in the process output.

So what can we do about this? We add feedback to our system! We accept the fact that disturbances and process variations are going to influence the controlled variable. However, instead of living with the resulting error, we add a sensor that will measure the controlled variable and pass it along to our controller. Now we can compare the sensed variable with the commanded variable and generate an error term. The error term is a measure of how far off the process is from where you want it to be and the controller can use this to produce a suitable actuating signal which then produces a suitable manipulated variable which finally affects the process in such a way that it reduces the error term. The beauty of the feedback control system - or a closed-loop control system[8] is that it is able to react to changes to the controlled variable automatically by constantly driving the error term to zero.



[8]The term closed-loop comes from the resulting loop that is formed in the block diagram when you feed back the controlled variable.

The feedback structure is very powerful and robust which makes it indispensable as a control tool. Unfortunately, with the addition of the feedback structure comes new problems that we now have to address. We need to think about the accuracy of the controlled variable at steady state, the speed with which the system can respond to changes and reject disturbances, and the stability of the system as a whole. Also, we've added sensors which have noise and other inaccuracies that get injected into our loop and affect the performance. To counter this last problem we can add redundant sensors that measure different state variables, we filter them to reduce the noise, and then we blend them together to create a more accurate estimate of the true state. These are some of the tools we can employ as system designers and part of what we'll cover in the rest of this book.



1.4 What is a control system?

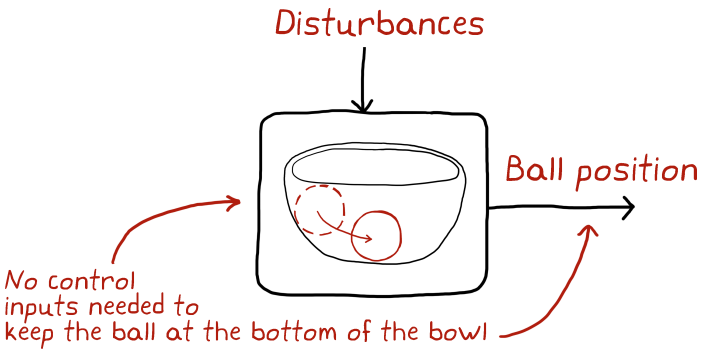
From the first few sections you probably already have a vague understanding of what a control system is. You might be thinking that it is something that makes a system behave in an automated fashion or is something that allows a system to operate without human intervention. This is true, to some

degree, but the actual definition is broader than you might think.

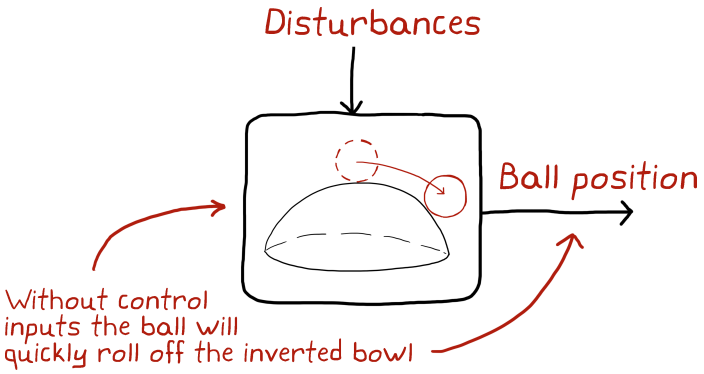
A control system is a mechanism that alters the behavior (or the future state) of a system.

Sounds like almost anything can be considered a control system, right? Well, one of the defining characteristics of a control system is that the future behavior of the system must tend towards a state that is desired. That means that, as the designer, you have to know what you want your system to do and then design your control system to generate that desired outcome.

In some very rare cases, the system naturally behaves the way you want it to and doesn't require any special input from the designer. For example, if you want a system that keeps a ball at the bottom of a bowl there wouldn't be a need for you to design a control system because the system performs that way naturally. When the ball is disturbed it will always roll back toward the bottom of the bowl on its own.

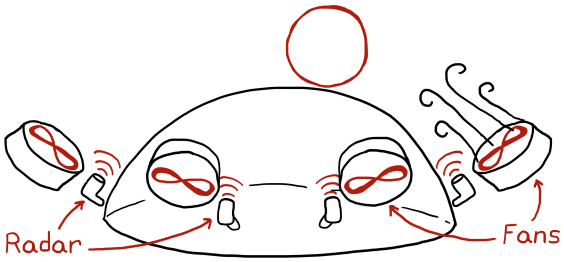


However, if you wanted a system that keeps a ball at the top of an *inverted* bowl, then you would need to design a control system to accomplish that. This is because when the ball is disturbed it will not roll back to the center naturally but instead continue rolling off the side of the inverted bowl.

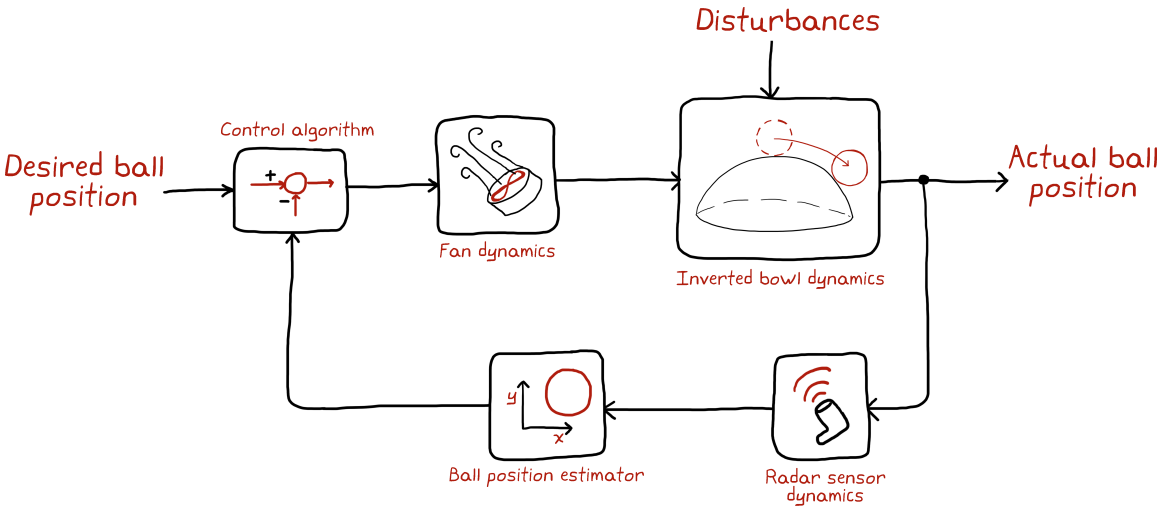


There are a number of different control systems that would work here. There is no *right answer* but let me propose a possible solution - even if it is quite fanciful and not very practical. Imagine a set of position detecting radar guns and wind generating fans that went around the rim of the bowl. As the

ball deviated from the top of the bowl the fan closest to it would turn on and blow the ball back up to the top.



The set of fans, radar guns, position estimators and control algorithms would count as a control system because together they are altering the behavior of the ball and inverted bowl dynamics. More importantly, however, they're driving the ball toward the desired state - the top of the inverted bowl. If we considered each interconnected part as its own little system, then the block diagram of the entire feedback system would look something like this:



This is a very natural way to split up the project as well because it allows multiple control specialists to work together toward a common goal. Instead of everyone trying to develop all parts of a complicated control system, each person would be responsible for designing and testing their part. Someone would be responsible for selecting the appropriate radar gun and developing the ball position estimator algorithm. Another person would be responsible for building the fans and the electronics to run them. Finally, a third person would be responsible for developing the control algorithm and setting the system requirements for the fan and radar specialists. Together these three ball balancing engineers would make up the control system team.

1.5 The first feedback control system

Feedback control systems exist in almost every technology in modern times, but there was a first feedback control system. Its story[9] takes place in the 3rd century BC in Alexandria, Egypt - 2000 years before the Industrial Revolution, at a time when Euclid was laying out the principles of geometry and Archimedes was yelling "Eureka!" over discovering how the volume of displaced water could lead to the density of the object. Our protagonist is the Greek mathematician Ctesibius, inventor of the organ and widely regarded

[9] If you're not interested in a story you can skip this section without loss of continuity in the book ... but c'mon, it's only a few pages and it's an interesting tale of the ancient ingenuity that led to modern day control theory

as the *father of pneumatics* due to his published work on the elasticity of air. We can overhear the conversation Ctesibius is having with one of his students, Heron, about an invention he has just completed - an invention that you will soon see is directly related to the Fundamentals of Control theory.

CTESIBIUS: I have done it!

HERON: What is that Master Ctesibius?

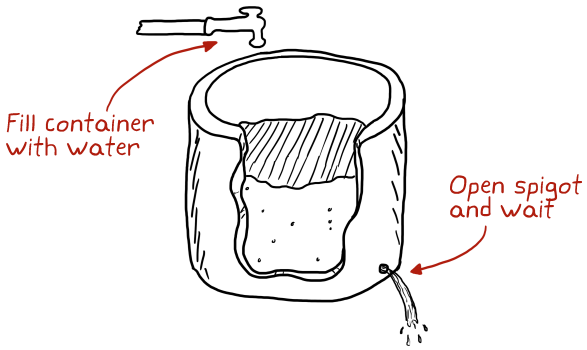
CTESIBIUS: I have invented a mechanism - something rather ingenious I might add - that will allow *anyone* to know the time of day.

HERON: But Master, we already have a number of ways of knowing the time. As you well know I can glance over at this sundial and see from the casted shadow that it is just past 11 in the morning.

CTESIBIUS: That is true. The sundial is wonderfully simple and humans have been using it to tell time for at least 3000 years. But it has its problems. How can you tell the time at night, or if it is a cloudy day, or if you are inside a building?

HERON: That's easy! At those times we use our water clocks. We take a container that has a small spigot at the bottom and

fill it with water. We let the water drain slowly through the spigot until it is empty. Since we know how long it takes to empty the container we then know how much time has passed.

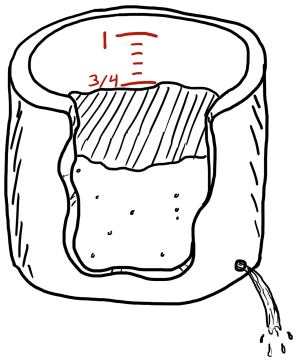


CTESIBIUS: Hmm, that is once again true, but what you have described is a *timer* and not a *clock*. There is a difference and it is very important. A timer is a device for measuring how much time has elapsed over some interval, whereas a clock's measurement is related back to the time of day. The sundial is a clock because we can see that it is 11 am, but using your water container we only know that perhaps one hour has passed since we opened the spigot.

HERON: Well, if we started the spigot at 11 am then when it completes we know that it is noon. Then we just start the process over again to always know the time.

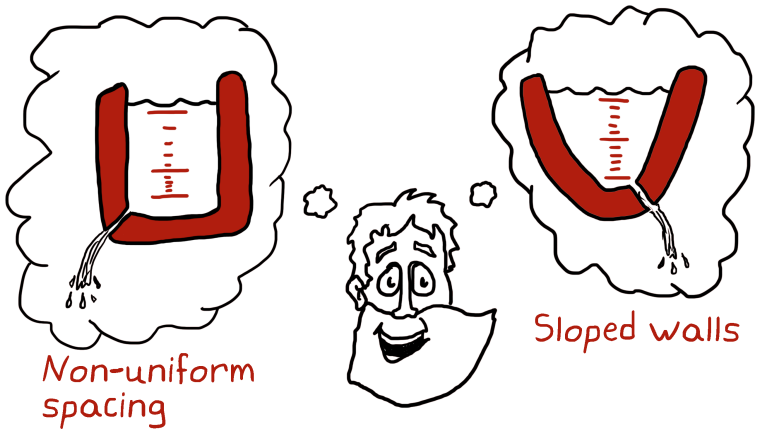
CTESIBIUS: Exactly, that would make it a clock! But what if you wanted to know the time when the container was not yet empty?

HERON: I guess we could see how full the container was by indicating the height of the water on the inside wall. Then if the container was three-quarters full we would say that one-quarter of the time has elapsed.



CTESIBIUS: There is a problem with this method though. Can you see it? The water will drain from the spigot much faster when the water level is high and the flow will gradually slow down as it empties. Therefore, a three-quarter-filled container means that *less* than a quarter of the time has actually elapsed.

I can tell that you are not convinced that this is a problem, right? You are going to tell me that the markings on the inside of the container don't need to be uniformly spaced, or that the walls of the container could slope inward so that water level will drop at a uniform pace.



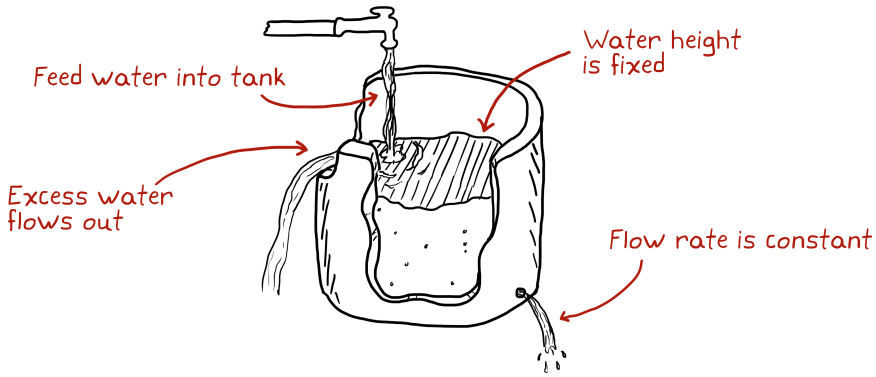
But this is hiding the real issue. By accepting that the flow rate through the spigot will change over time we need to have some extra knowledge - namely how to shape the inside of the container or where to mark the non-uniform indications on the inside wall. Once we create those markings or that container then we have no control over it. Any errors in manufacturing will generate errors in our results.

What we really need is a spigot that will have a steady and continuous flow rate. That way we don't rely on the con-

tainer at all, we just need to know how much water comes out in an hour. From there, deriving *any* other time is as simple as measuring how much water has been released and comparing it to the amount released in an hour. Do you have any ideas on how to accomplish a steady flow rate from the spigot?

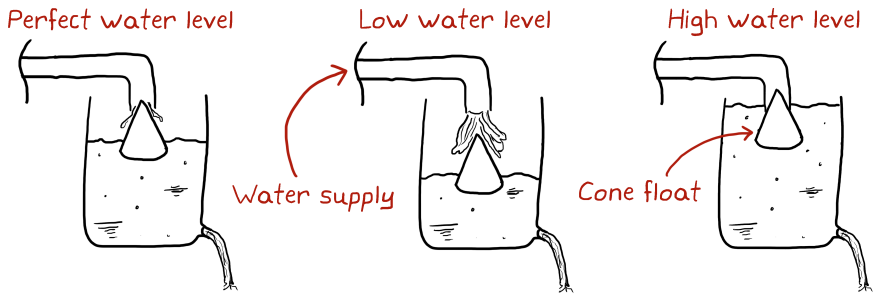
HERON: Let me think about it. The flow rate decreases because the water level drops in the container and the pressure exerted at the spigot is lower. So by keeping the water level in the container constant, the flow rate will be constant. Ah, but how do we keep the water level constant?

I got it! I'll modify the tank. It will have two spigots, a large one at the top and a smaller at the bottom. We'll feed water into this container at a much faster rate than the smaller spigot can handle which will fill the container up. Once the water level reaches the larger spigot, it will begin to overflow, which will keep the water fixed at that height. Is this your invention Master Ctesibius?



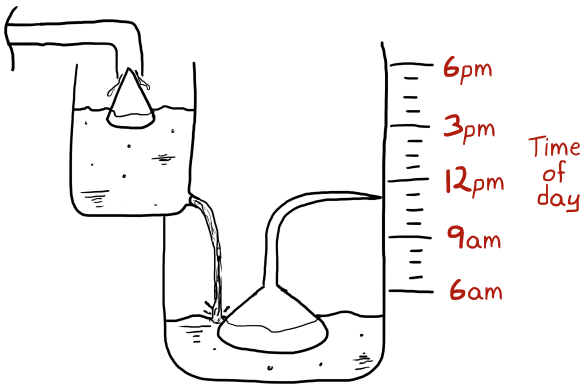
CTESIBIUS: No, no, no! The overflow tank will accomplish a steady flow rate for sure but in solving our problem you've just introduced another one. Your clock will use and waste a lot of water. This will cause you to need a larger water source reservoir and that will make your clock less mobile. There is an elegant solution to this problem, and that is what I've just discovered.

We still have the same container from before, but it is fed by a continuous water supply with a valve with a circular opening. We place a float in the shape of an inverted cone in the water container just below the water source inlet.



When the water level is high the float is pushed up into the valve such that it shuts off the water supply. As the container water level decreases, the float drops, allowing the supply water to flow. The water level will reach a steady state once the float is low enough that the water flowing in is exactly equal to the water leaving the spigot. The beauty of this is that it is self-correcting! If for some unforeseen reason the water level drops in the container, the float drops with it causing more water to be supplied to the container than it is losing through the spigot. This has the benefit of filling the container back up. It's brilliant!

Now we can add a larger container to catch the falling water and place a second float in there to mark the hour of the day. We only need as much water as it takes to fill the two containers in order to know the time all day. No wasted water!



HERON: That is brilliant! I bet there are many different applications where we could use this float regulator.

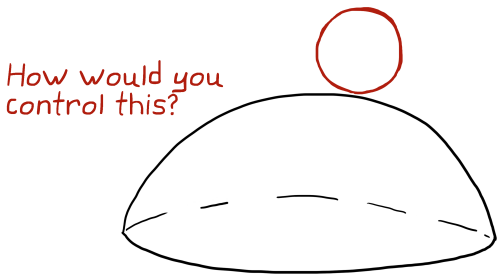
CTESIBIUS: Absolutely. Oh, is it really almost 12 pm? I have to run, I'm needed at the Museum of Alexandria. Think of those other uses for this invention. I'd love to hear what you come up with.

With 2300 years of technological advancements to base your opinion on, this cone shaped float might not seem like much but the float regulator would go on to be used for many applications - in fact, it's the same technology that is still used in modern day toilets. The real benefit of feedback control systems is that the system is self-correcting. Now that systems could automatically adjust to the changing environment it removed the need for people to be part of the machine operation. This is similar to cruise control on your car removing *you* as the controller of your speed. This is the goal

of a control system engineer - through ingenious design and application develop systems that can regulate an output automatically. However, as you will see this opens a series of challenging and exciting problems. It will take mathematics to expose these problems and so now that we have the context of the problem we're trying to solve let's move onto the next chapter and discuss how we represent the problem mathematically.

1.6 Try This!

1. Come up with another possible control system for the inverted bowl problem.



a) What does your system use for sensors? Actuators?

b) Explain some of the benefits and drawbacks of your design? Consider things like power usage, sound volume, ease of assembly, the ability to scale to larger systems, and coolness factor.

c) Make a sketch of your control system and point out how you would split up the project so that multiple people could work on it.

2. What additional applications are there for the float regulator? These can be problems that *have* been solved by a float regulator and those that *could* be solved by them.

3. How do humans act as the control system and how do they 'close the loop' with machines? As an example think about what a person is doing when they're driving a car.

4. Find examples of control systems in your everyday life (closed loop, open loop). Control systems don't always need to be machines, think about how feedback control is part of your relationships, your approach to studying for an exam, or the cycle of being hungry and eating.

5. Describe each of the following control systems. Determine the inputs, outputs, and process. Describe whether it is open-loop or closed-loop.

a) Your home air conditioning system

b) Sprinkler system for your lawn

c) The lighting in a room

d) The fly-by-wire system on an aircraft

e) The population of a colony in an area with limited resources

Chapter Credits

Meg Douglas	Kirkland, Washington	Benjamin Martin	Dubai, UAE
Federico Pistono	1 AU	Wong, C.J.	Zootopia

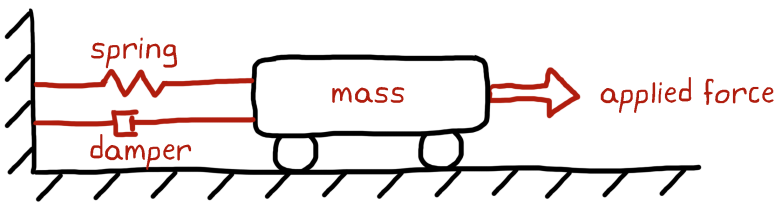
Thanks for making this chapter awesome!

2 Transfer Functions

As an engineer, it is crucial that you are able to describe your system in an efficient and useful manner. What is efficient and useful, however, changes depending on the context of your problem. For example, the first chapter explained systems conceptually so they were depicted with drawings illustrating the collection of parts that make them up. The drawings were also complemented with descriptive words to help you understand what the drawings represented.

If, however, your problem is to build a controller that alters the behavior of your system then words and pictures are no longer the most efficient and useful way to represent your system. For example, imagine you were given the following description of a restrained cart and asked to describe how the cart moves if a force of 1 Newton is applied for 1 second.

> There's a wheeled cart that can roll without friction along the floor. The cart is attached to a wall with a spring and damper. A user can apply a force to the cart and set it in motion.



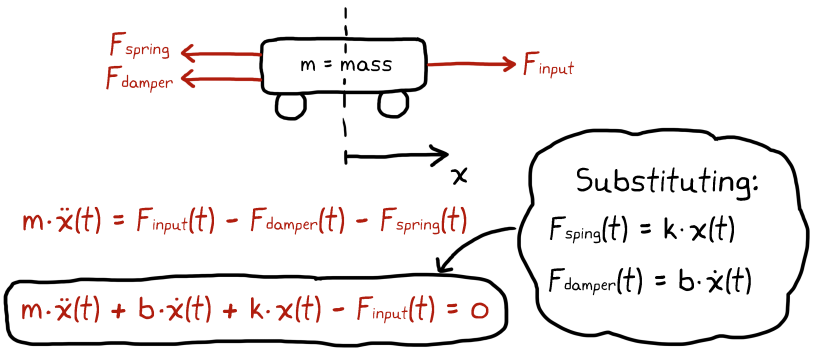
You could probably reason through this problem by imagining that while the force is applied the cart will begin to accelerate in the direction of the force

- faster at first but slowing as the restorative force from the spring grows. Once the applied force is removed the cart will spring back toward its resting state. The damper will remove energy from the system, and depending on its relative strength, will either cause the cart to oscillate with a shrinking amplitude or will cause the cart to asymptotically return to its starting position.

Not a bad description - but if you were tasked with developing a system that automatically adjusted the applied force so that the cart followed the desired profile, then this description of the system would not be sufficient.

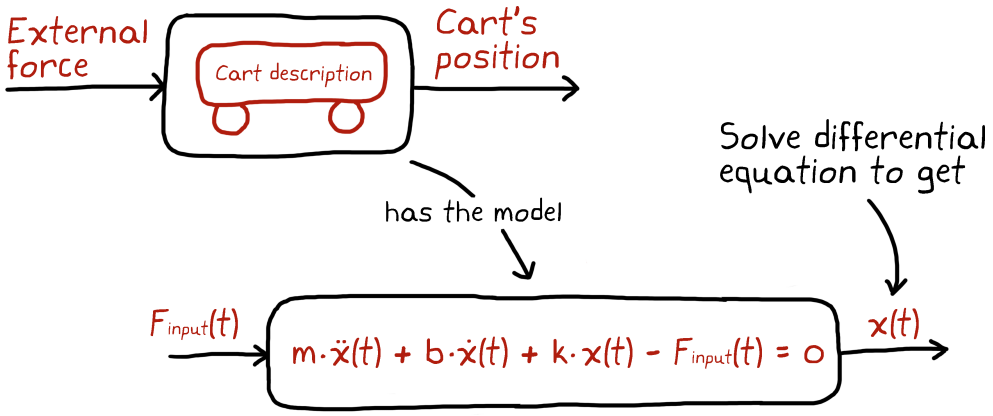
So what can we do?

We describe the system mathematically. We do this by writing the equations of motion in the form of differential equations. From the description of the problem, the equations of motion can be written directly using a free-body diagram. The mass, spring constant, or damping coefficient were not specified so we can make them variables, m, k, and b, respectively[1].



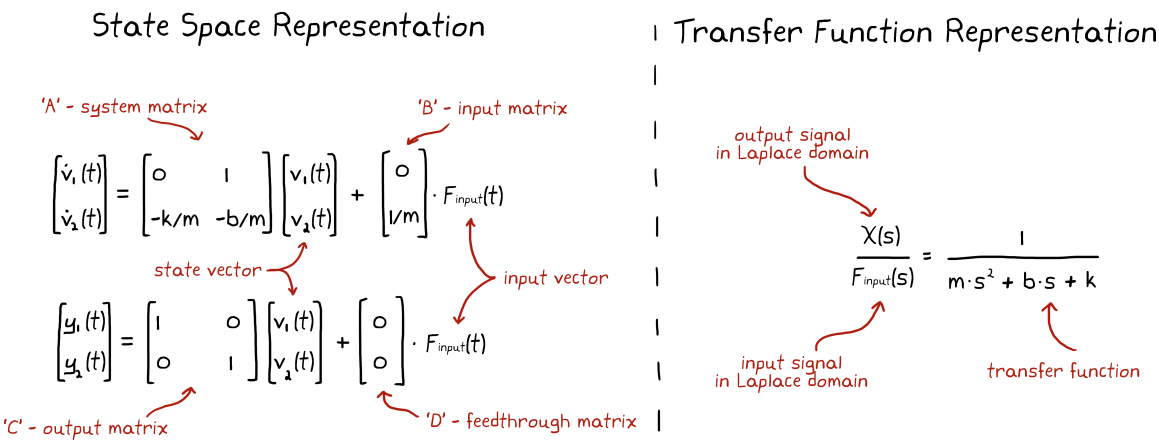
[1]It's important to remember that most of the time when you are trying to solve the control problem you also have to solve the system identification problem and the simulation problem as well. Writing out the equations of motion is part of system identification and is the white box method that was introduced in the first chapter. Beyond this, however, finding the appropriate m, k, and b requires testing of your system and this is part of the black box method.

At this point we are left with a single 2nd order ordinary differential equation, $m\ddot{x}(t) + b\dot{x}(t) + kx(t) - F_{input}(t) = 0$. We call this the mathematical model of the system. The external input, or the excitation into our model, is the force applied to the cart, F_{input}, and when we solve the differential equation we get the position of the cart over time, $x(t)$.



A differential equation is great for solving for the response of a system but it doesn't lend itself very well to analysis and manipulation, and this is something that we absolutely want to do as control engineers. The whole point is to analyze how the system naturally behaves and then manipulate it so it behaves the way we want it to. Luckily there are other ways to represent a mathematical model which makes the control engineer's job easier.

The two most popular representations, and the two that we'll cover in depth in this book, are *state space representation* and *transfer functions*. Loosely speaking, transfer functions are a Laplace domain representation of your system and they are commonly associated with the era of control techniques labeled *classical control theory*. State space is a time domain representation, packaged in matrix form, and they are commonly associated with the era labeled *modern control theory*.



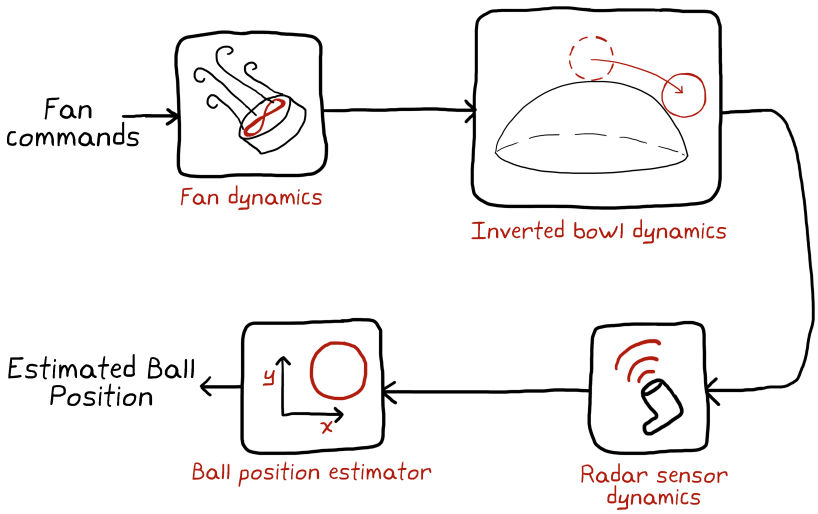
Each representation has its own set of benefits and drawbacks and as a control engineer, you will need to be very familiar with both. Don't get too hung up on the labels *classical* and *modern*. In this book, we won't separate classical control and modern control into two separate sections. Rather, we will derive techniques and solve problems using whichever method is most appropriate. This way you will become comfortable switching between representations, and therefore switching between the set of tools that you can use as the problem requires.

In this chapter, we will focus on transfer functions and so to begin let's start with the formal definition.

A **transfer function** is the Laplace transform of the impulse response of a linear, time-invariant system with a single input and single output when you set the initial conditions to zero. They allow us to connect several systems in series by performing convolution through simple multiplication.

Yikes, that was a lot to take in! Don't worry if that didn't make any sense. Transfer functions are too important in control theory to gloss over quickly so we'll walk through each of those terms very carefully and explicitly. That way not only will you have a good idea of *how* to use transfer functions but you'll learn *why* and *when* to use them as well.

Before we jump into explaining the definition of a transfer function let's set up an example where representing our system as transfer functions make the control engineer's job easier. Let's take the inverted bowl control problem from the first chapter. Before a control system can be designed that will keep the ball on top of the inverted bowl, we first need to understand the behavior of the entire system. That means when we apply a command to the fan we want to know how that affects the estimated ball position.



Let's say we apply a step command to the fan - for example, to go from off to half of its max speed[2]. There is a delay as the fan has to accelerate over some time period to get to speed and after which there is some variation

[2]You do this whenever you turn on a house fan to medium

in fan speed due to physical disturbances in the system. The output of the fan system is air velocity which is subsequently the input into the inverted bowl dynamics. The bowl dynamics system calculates the force on the ball from the air and uses that force to determine how the ball moves. The true ball position is then sent to the radar sensor model which produces a relative distance to the radar gun. Just like the fan, the radar system introduces more delay and it also adds errors in the measurement. The relative ball position is then used to estimate where the ball is in the bowl frame which is the final output of our system.

We could write out the differential equation for the entire end-to-end system which relates the fan command to the estimated ball position, but this would be difficult to do because of how complex each of the parts are individually. Also, recall that we separated the system into smaller, more manageable parts so that we could have several engineers working simultaneously on the problem. Therefore, what would be ideal is a way to represent each part separately and then combine them into a full system later. This would allow each engineer to write out a simpler mathematical model of their own part and supply it to the person responsible for pulling the model together and designing the control system for it.

So with this example in mind let's walk through the explanation of transfer functions and see why they are the perfect representation for our problem.

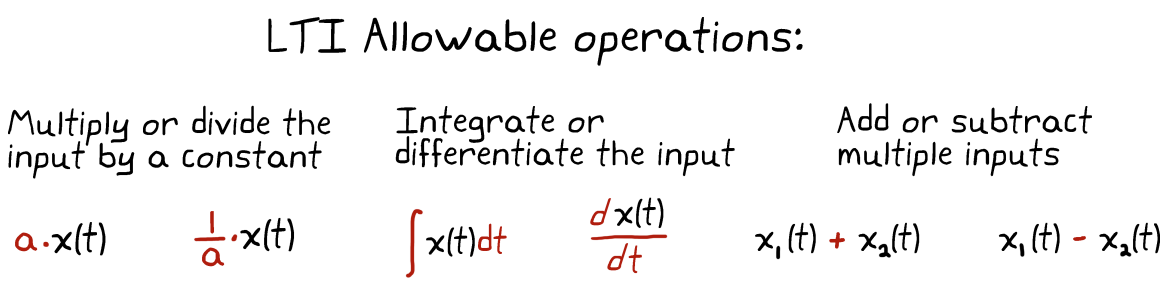
2.1 LTI Systems

According to our formal definition, transfer functions require that you have a linear and time-invariant (LTI) system. To understand why this is the case

we need to learn what an LTI system is and review a small amount of linear theory.

When you model your system you get to choose the set of mathematical operations that map the system inputs to the outputs. This is a fancy way of saying that you get to derive the differential equations that represent the behavior of your system and since you are in charge of your model you can decide what to represent and how complex your representation is.

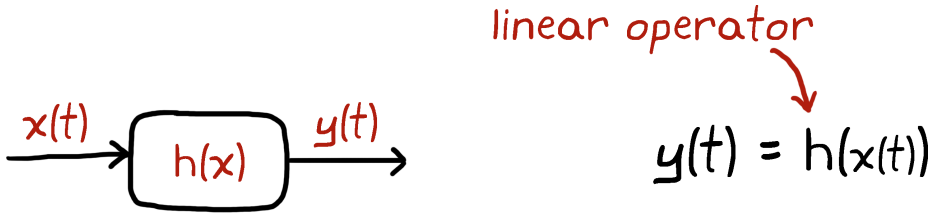
You have dozens of mathematical operations at your disposal, however, I'm going to make a case for representing the system as linear, time-invariant, in which case you actually can choose only from a few mathematical operations. There are so few in fact that we can easily list every one of them.



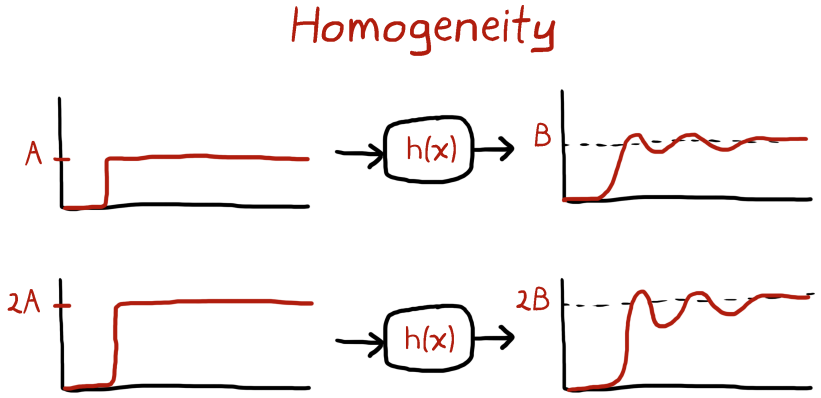
If you can model your system with some combination of these six operations, and only these six operations, then you can make assertions about how the output of the system will change based on changes to the input. This is because all LTI systems have the following properties; homogeneity, superposition, and time-invariance. These properties are what cause the system to behave in predictable ways. To understand what these properties mean let's walk through them one at a time.

For these examples, I'm going to represent a collection of linear operations with the linear operator, h. With this notation, $y(t) = h(x(t))$, can be read as

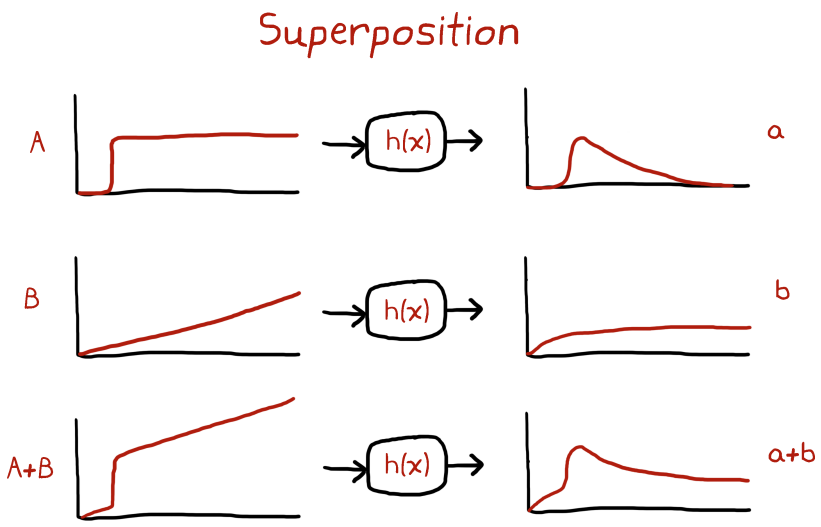
the operator, h, provides a linear mapping between the vector $x(t)$ and the vector $y(t)$.



Homogeneity means that if you scale the input, $x(t)$, by factor, a, then the output, $y(t)$, will also be scaled by a. So in the example below, a step input of height A produces an oscillating step to height B. Since $h(x)$ is a linear system then a step input that is doubled to $2A$ will produce an output that is exactly doubled as well.



Superposition, or you might also hear it called additivity, means that if you sum two separate inputs together, the response through a linear system will be the summed outputs of each individual input. In the example below the step input, A, produces output, a, and the ramp input, B, produces output, b. Superposition states that if we sum inputs $A+B$ then the resulting output is the sum $a+b$.



We can describe these properties formally as:

Homogeneity

$h(ax) = a \cdot h(x)$

Superposition

$h(x_1) + h(x_2) = h(x_1 + x_2)$

Or more generally, and if you make the substitution that $y(t) = h(x(t))$, we can combine homogeneity and superposition into one large definition.

$$ay_1(t) + by_2(t) = h(ax_1(t) + bx_2(t))$$

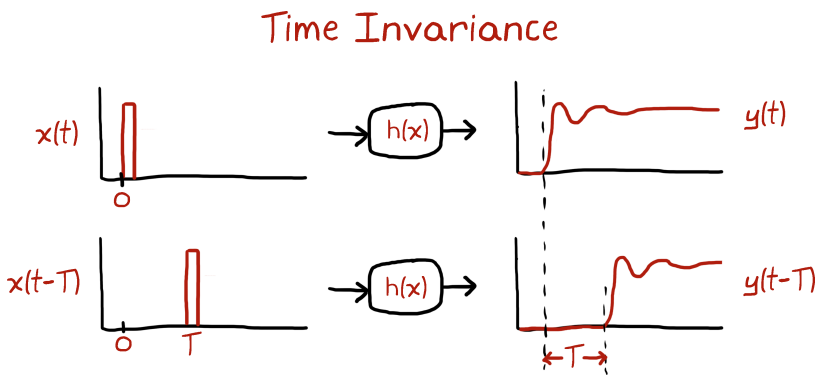
A system is defined as a linear system if it has the two properties homogeneity and superposition.

Side note: Don't mistake a linear system and a linear equation. They are two separate things. We've just described a linear system, it is a mapping

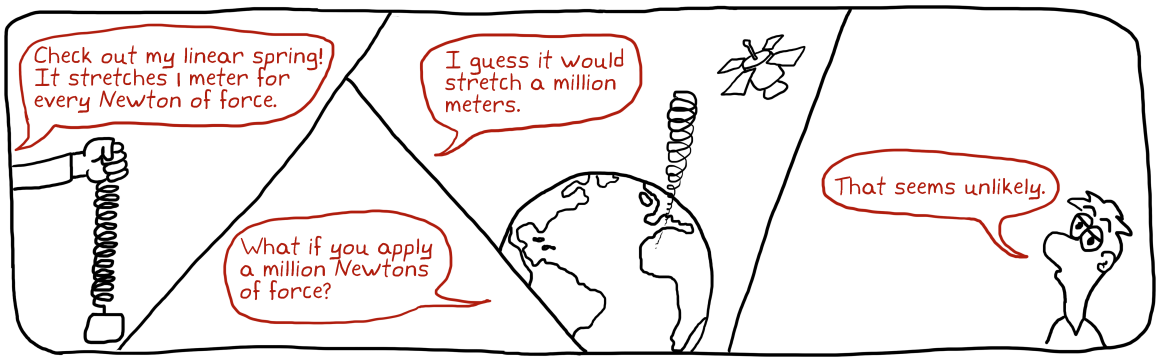
between two vector spaces that obeys the properties of homogeneity and superposition. A linear equation, on the other hand, is an algebraic expression where every term is a single, first-power variable multiplied by a constant.

A well-known example of a linear equation is the equation of a line in a two-dimensional plane, $y = mx + b$. You can verify that this equation is not homogenous because, for example, the output is not doubled when you double the input, $2(mx+b) \neq m(2x)+b$.

Linearity is only the first part of an LTI system. The second part, time invariance, refers to a system behaving the same regardless of when in time the action takes place. Given $y(t) = h(x(t))$, if we shift the input, $x(t)$, by a fixed time, T, then the output, $y(t)$, is also shifted by that fixed time. We can write this as $y(t-T) = h(x(t-T))$. Sometimes this is also referred to as translation invariance which covers translation through space as well as time. Here's an example of how shifting the input results in a shifted output in a time-invariant system.



The restrictions required by an LTI system are severe[3], so severe in fact that no real physical system meets them. There is always some aspect of non-linearity or variation over time in the real world.



So you might be thinking, "great, we know how we can scale, shift, and sum inputs into an LTI system, but if they don't represent real systems then why are they so important and how does it help us understand transfer functions?"

To answer your first question I think theoretical physicist, Richard Feynman, said it best when he said *"Linear systems are important because we can solve them."* We have an entire arsenal of mathematical tools that are capable of solving LTI systems and, alternatively, we can only solve very simple and contrived non-LTI systems. Even though no real system is LTI there are, however, a wide range of real problems can be *approximated* very accurately with an LTI model. As long as your system behaves linearly over some region of operation, then you can treat it as LTI over the region. You can create a linear model from a non-linear equation through a process called linearization which is a skill that we'll cover in a later chapter.

[3]We only get to choose from 6 operations?! That's like restricting Van Gogh to just the color brown!

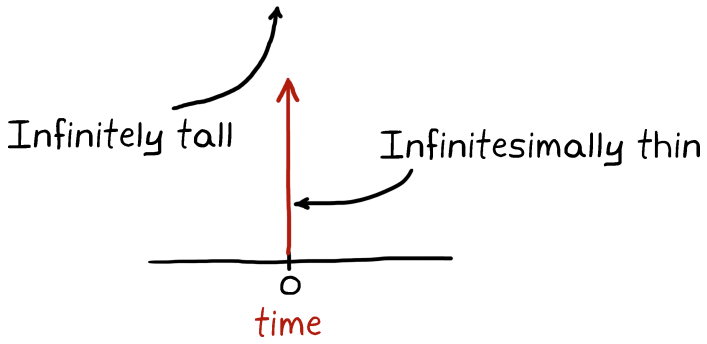
To answer your second question, we know that the definition of transfer functions requires that the system is LTI, but we haven't gotten to why. To answer that question we need to talk about the impulse function.

2.2 Impulse Function

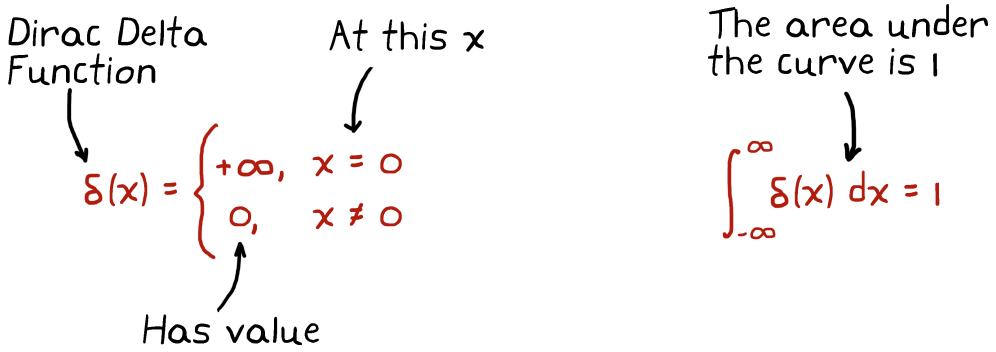
An LTI system can be fully characterized by knowing how the system behaves when an impulse is applied to it. The resulting output of a system that is subjected to an impulse function is called the *impulse response* of the system.

You probably already have a general idea of what an impulse is[4], but just to be clear let's define it here. The impulse function is a signal that is infinitesimally short in time but has infinite magnitude. It is also referred to as the Dirac Delta function, which is named for Paul Dirac, the theoretical physicist who first introduced the concept. I'll use the terms *impulse function* and *Dirac Delta function* interchangeably throughout this book. Since it is impossible to draw a line that is infinitesimally thin and infinitely tall we represent a Dirac Delta function as an arrow pointing up at the time the impulse is applied.

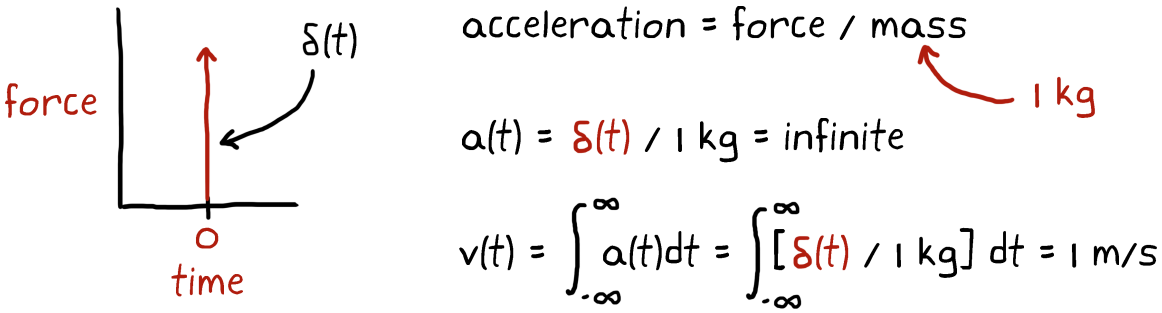
[4] An impulse is that sudden and strong urge to buy a snack in the checkout line at the grocery store



You can also represent this mathematically by stating that the function returns a value of positive infinity at time zero and returns a value of zero at all other times. Additionally, the impulse function is defined such that the integral of the function is one. Or to put it differently, even though the function is infinitesimally thin, and therefore has no thickness and so no area, when we perform an integration on the function we say that it actually does have area and define that area to be one unit.

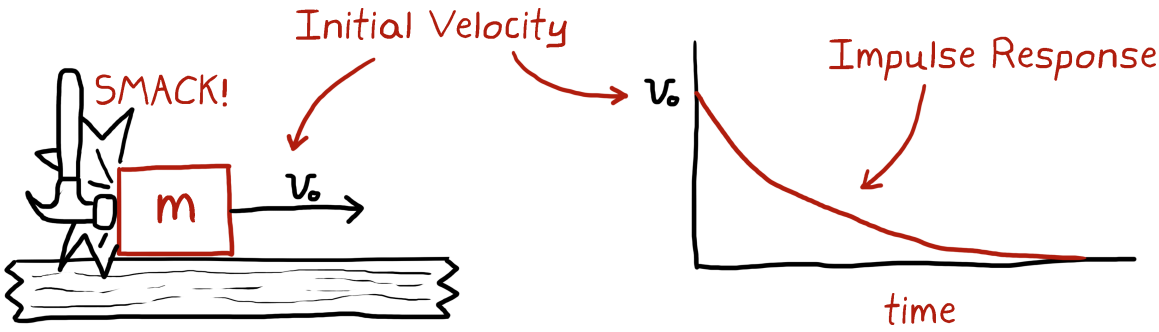


Being able to integrate the Dirac Delta function is crucial because we can assign it to physical properties and perform mathematical operations on those properties like we would with real finite functions. For example, the impulse could represent the force you exert on a 1kg mass.

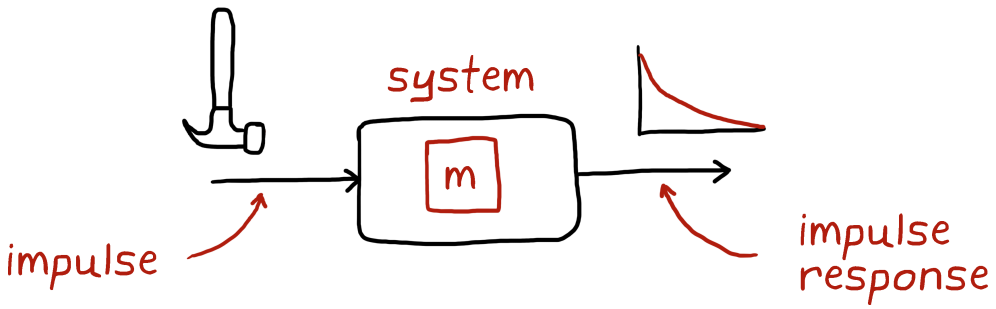


We know that acceleration is force divided by mass. Therefore, if the force was infinite we'd find the object would experience infinite acceleration - not a useful result. However, if we integrate the acceleration over time we get the object's velocity. Using the definition of the integral of the impulse function we find that the mass accelerates instantly to 1 m/s - and this *is* a useful result. We can use the impulse function to change the state of the system in zero time, or in this case, we were able to mathematically give the mass an initial velocity of 1 m/s.

Let's walk through a thought exercise using the impulse function and see if we can start to tie this back to LTI systems and eventually back to transfer functions. Imagine you have a block sitting on a table and you hit it with a hammer. This would be very close to an impulse because the hammer would apply a very large force to the block over a very short period of time. This would give the block an instantaneous initial velocity and start it sliding across the table. The block would then slow down over some amount of time due to friction and would eventually stop. The resulting change in velocity over time is the impulse response of the system.



Here is our system drawn out in block diagram form so you get a better idea of the input to output relationship.

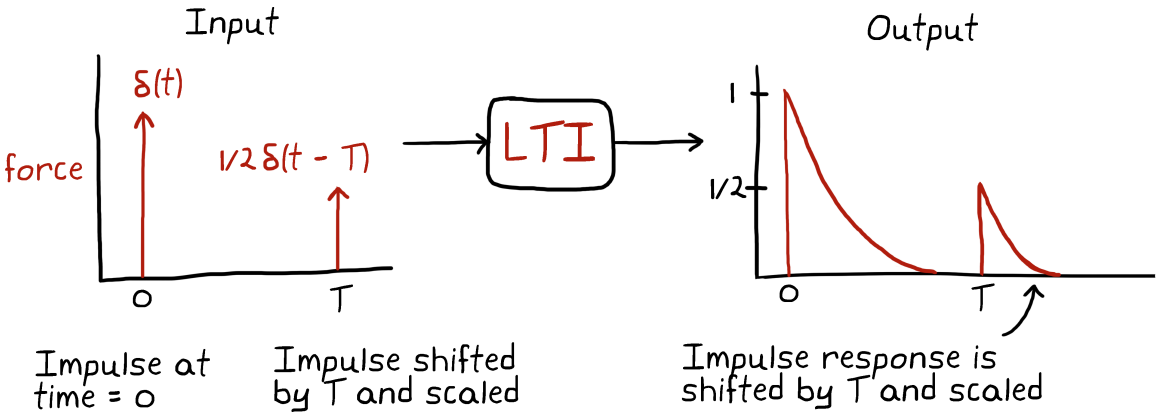


We still call it the impulse response regardless of whether we treat this system as LTI or not[5], however, for the sake of this thought exercise, we'll say that our system behaves like an LTI system so that we can take advantage of the LTI properties of homogeneity, superposition, and time invariance.

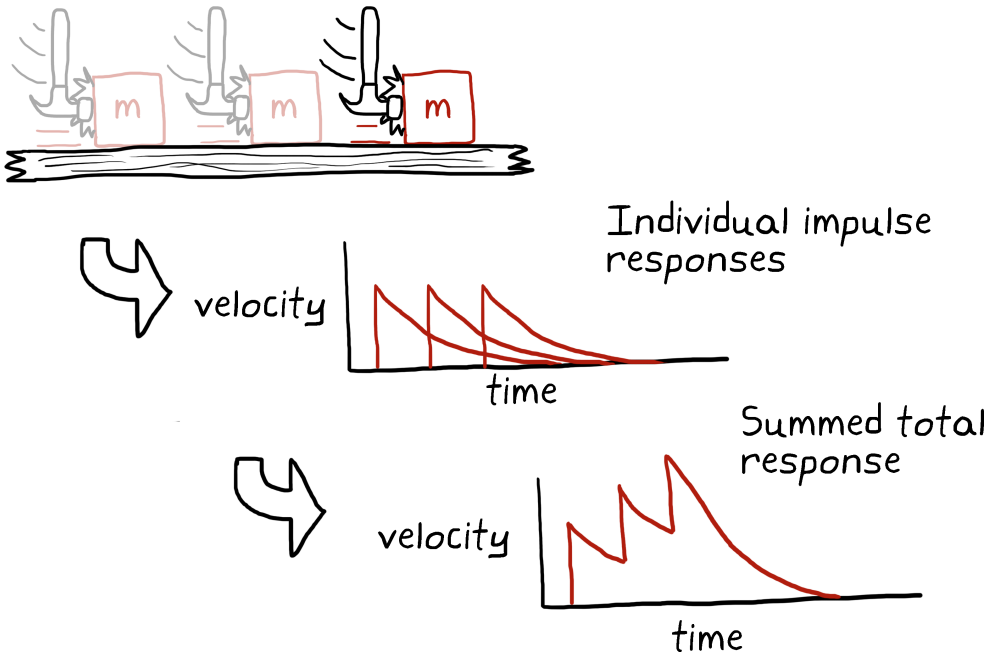
To continue, let's say that after the mass came to rest we hit it again with the hammer but this time half as hard. Since this system is time invariant we know that if we shift the impulse by time T then the impulse response is also shifted by T. Additionally, because the system obeys homogeneity,

[5]Non-LTI systems still respond to impulses, we just can't infer as much about the system from the impulse response as we can with LTI systems.

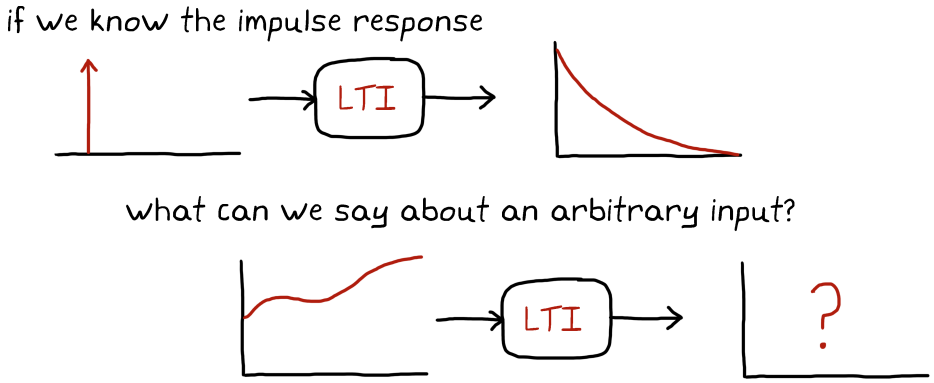
if we scale the input by one-half then the output will also be scaled by one-half. Finally, due to superposition, we know the motion of the block is the summation of the first impulse response and the second impulse response.



We can see the power of summing the impulse responses of an LTI system by striking the block multiple times in quick succession.



If the input into our system is a series of impulse functions then we know how to sum the individual responses to create the total response of the system. This brings up the question: what if our input isn't an impulse function? Is there something we can claim about the response to any arbitrary input if we know the impulse response?

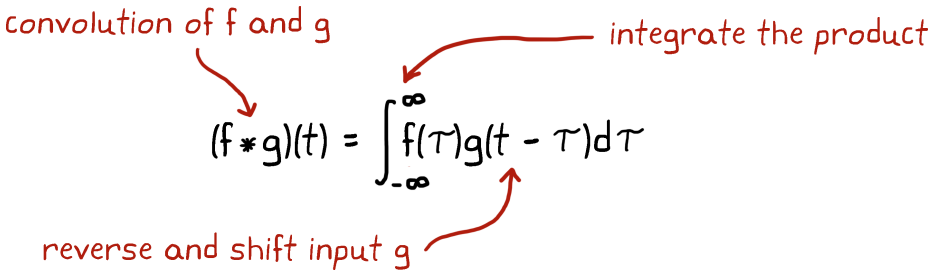


Well, this is a completely acceptable question because there is no such thing as an ideal impulse in real applications. Infinitely high and infinitesimally thin are concepts that can't physically happen. Therefore, let's see how we extend our summing technique to real continuous inputs using the convolution integral.

2.3 Convolution Integral

The convolution integral is a mathematical operation that you perform on two functions - we'll call them $f(t)$ and $g(t)$ - and it is written in shorthand notation as an asterisk or a star. Since the convolution integral resides in the mathematical domain a lot of effort is spent solving the integral for a number of unique and interesting input functions. However, being able to solve the

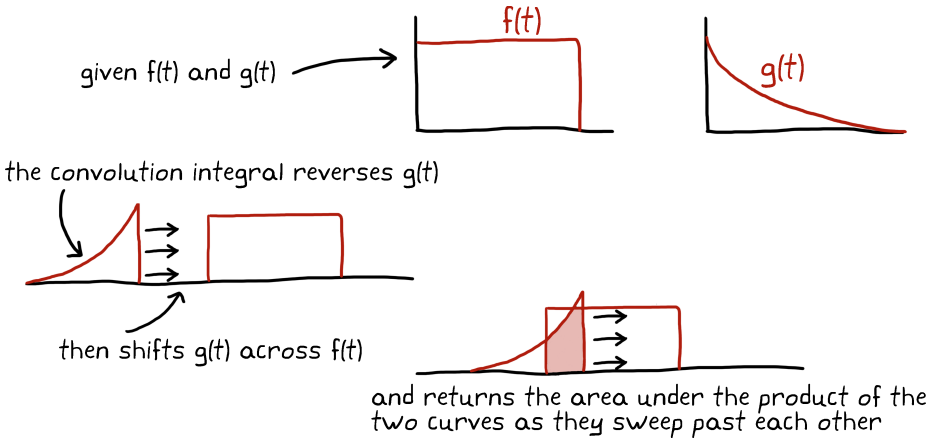
integral usually does not help the student understand what the convolution integral is doing or why it works in the first place. Hopefully, by walking through how it relates to 'playing' arbitrary inputs through an LTI system this chapter can make convolution a little less convoluted[6].



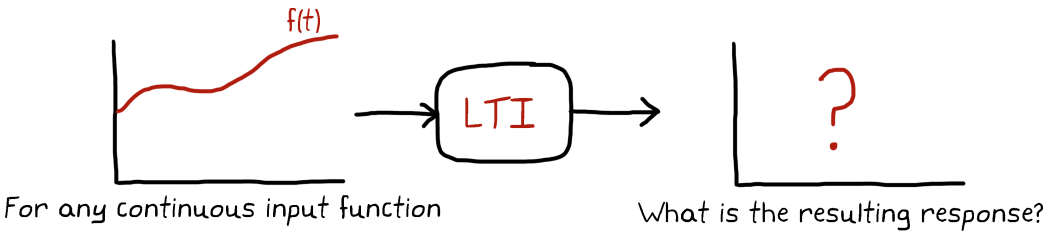
The convolution integral might look daunting at first, but there are really only three parts to the equation; (1) you reverse the input function $g(t)$ and shift through all of time, (2) you multiply the reversed and shifted $g(t)$ by $f(t)$, and (3) you sum the product over all of time.

You can picture this graphically by taking a time history of $g(t)$, drawing it backward, and sliding it across $f(t)$. The value of the output function is the area under the two input functions when they overlap.

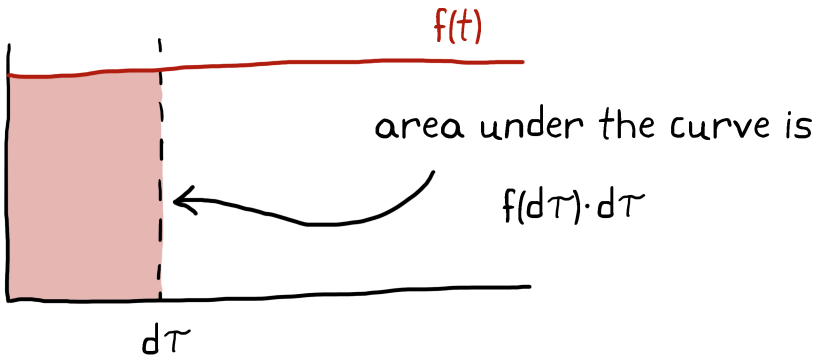
[6] See what I did there?



This isn't a bad visual explanation of the convolution integral, but it still doesn't explain why this produces a function that means anything. So let's answer that by deriving the integral from scratch. We'll accomplish that by solving the problem we started with: how to play arbitrary inputs, $f(t)$, into an LTI system and determine the response.

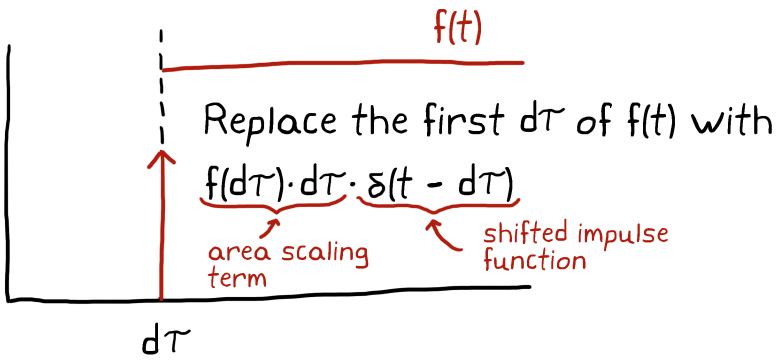


Let's magnify $f(t)$ and just look at the very beginning of the function. In fact, we'll just look at the first infinitesimally thin slice of the input function, $d\tau$. The area under the curve for this brief portion can be approximated as $f(d\tau) \cdot d\tau$, or in words, the height of the function times the width of the slice.



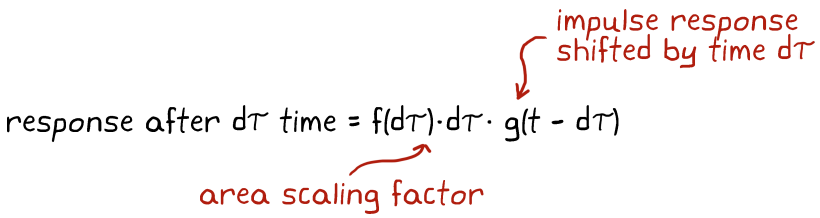
This assumes that $f(t)$ is constant for the entire slice of $d\tau$, hence it's an approximation. However, if we take the limit as $d\tau \to 0$ then this becomes the exact area under that instant of the curve and if we sum each $d\tau$ over all time then we get the area under the entire curve[7]. More about taking the limit later, for now, we'll just assume $d\tau$ has a thickness and this process is just an approximation.

Since we have the area under this slice we can replace just this small section of $f(t)$ with a single scaled impulse function. The impulse function has an area of 1 so if we multiply it by the area under our slice we've essentially scaled the impulse function to have a similar area.

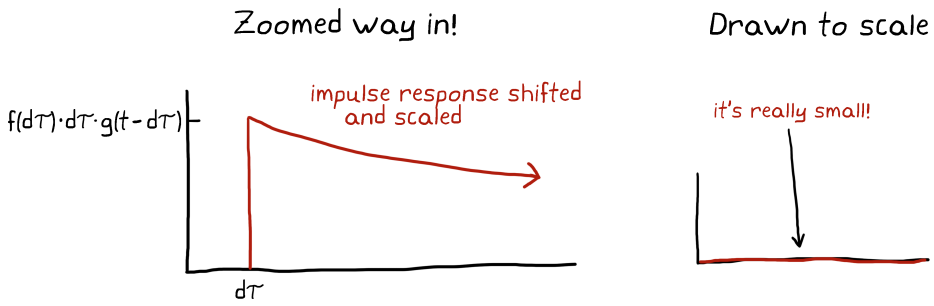


[7] You'll notice we just described standard integration of a function

We've converted the input, for just this small slice, $d\tau$, into a scaled impulse function, $f(d\tau) \cdot d\tau \cdot \delta(t - d\tau)$. Since this is an LTI system, we can assert what the system response will be to this input. That is, we know the response to this time-shifted impulse function, it's just the time-shifted impulse response of the system, $g(t - d\tau)$[8]. Through the homogeneity property of LTI systems, scaling the input function by $f(d\tau) \cdot d\tau$ scales the output by the same amount. Therefore, after $d\tau$ time, we are left with the following scaled impulse response.

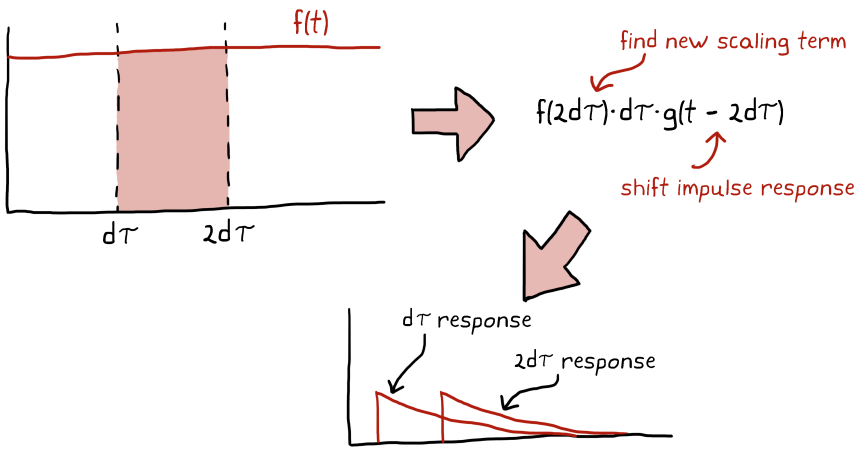


If we plot the system's response to just this first scaled impulse then it would look like the graph below on the left. Notice, however, if $d\tau$ is extremely small then the impulse response will be scaled way down and you wouldn't even notice its impact.

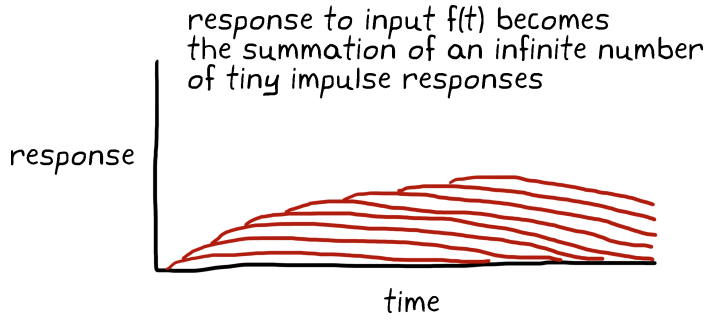


But don't worry! We will move onto the second $d\tau$ slice and you'll start to see a pattern building.

[8] That makes sense!

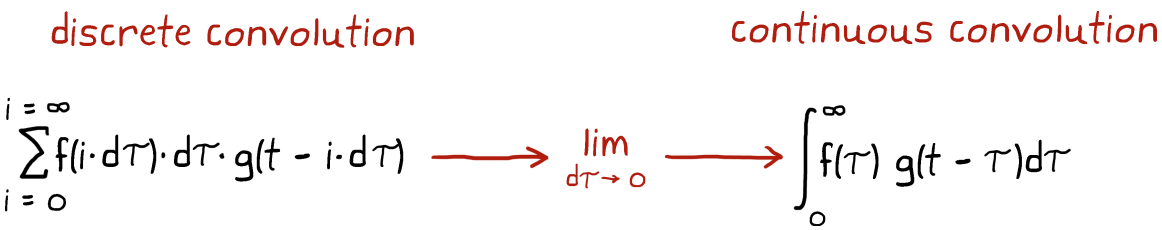


Each time you move to the next $d\tau$ you replace it with a scaled impulse function. This produces a scaled impulse response that is shifted in time to correspond to when the impulse function occurred. This shift in time is permitted because, again, our system is LTI and therefore time invariant. Also, each individual impulse response - which has been scaled down to almost nothing - is summed together using the property of superposition. As you move along the input function you are creating layer upon layer of infinitesimally small impulse responses that build on each other.



We can proceed through each of these discrete $d\tau$ steps until we get through the entire input function. We can write this in a compact form using the

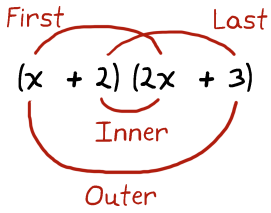
summation operation and stepping through an infinite number of i steps. This is discrete convolution and it is how computers perform the operation. Remember this is an approximation - we have to take the limit as $d\tau \rightarrow 0$ for it to be exact. When we do this, each discrete step is replaced with a continuous operation and the summation operator becomes the integral operator.



This is the continuous time convolution integral function that we started with - with one small difference; the integration limits go from zero to infinity in our example and negative infinity to infinity in the original function. This is one of the differences between a purely mathematical function and one that is used in practical applications. In engineering, often our signals start at time zero and since there is no signal between negative infinity and zero we don't bother performing the integration over that region. However, you'll end up with the same answer regardless of whether the lower bound is negative infinity or zero.

An interesting observation: Multiplying two polynomials together can be accomplished by performing the discrete convolution of the polynomial coefficients. To understand this let's look at how you would ap-

proach multiplying two, two-term polynomials, using the popular, but ultimately limiting FOIL method.

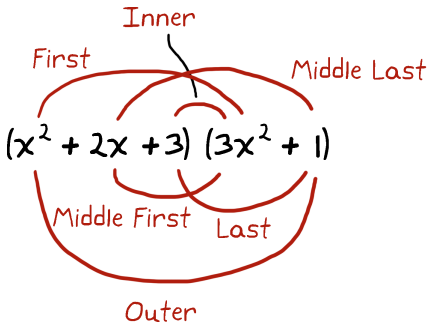


F O I L

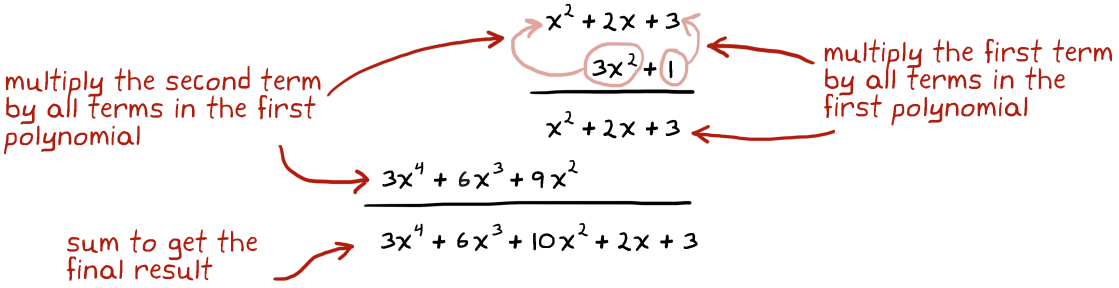
$(x \cdot 2x) + (x \cdot 3) + (2 \cdot 2x) + (2 \cdot 3)$

$= 2x^2 + 7x + 6$

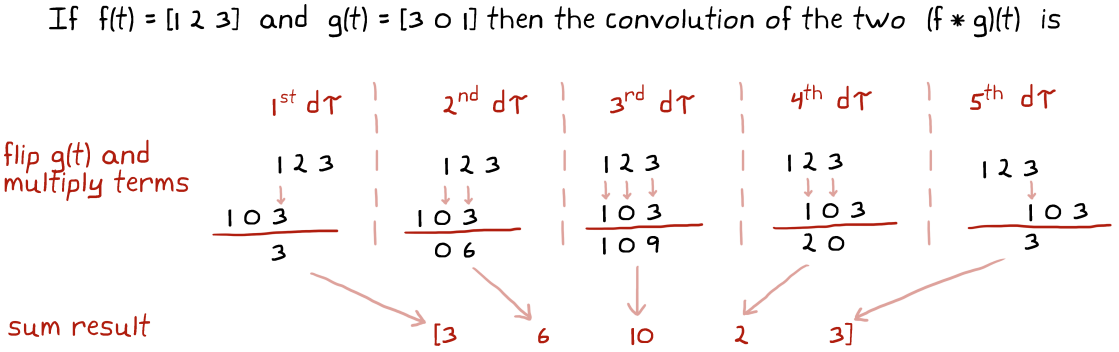
I say this is limiting because this rule of thumb breaks down for multiplication involving a polynomial with more than two terms.



Therefore, if you realize that polynomial multiplication is really no different than regular multiplication you'll notice that all you are doing is multiplying every term with every other term and summing like units. This example multiplies $x^2 + 2x + 3$ and $3x^2 + 1$.



This is exactly what discrete convolution is doing. We can define the first polynomial as the vector $[1,2,3]$ and the second polynomial as the vector $[3,0,1]$. To perform discrete convolution we reverse the order of one of the vectors, sweep it across the other vector, and sum the product of the two.



In fact, with the program MATLAB, you can perform polynomial multiplication using `conv`, the command for discrete convolution.

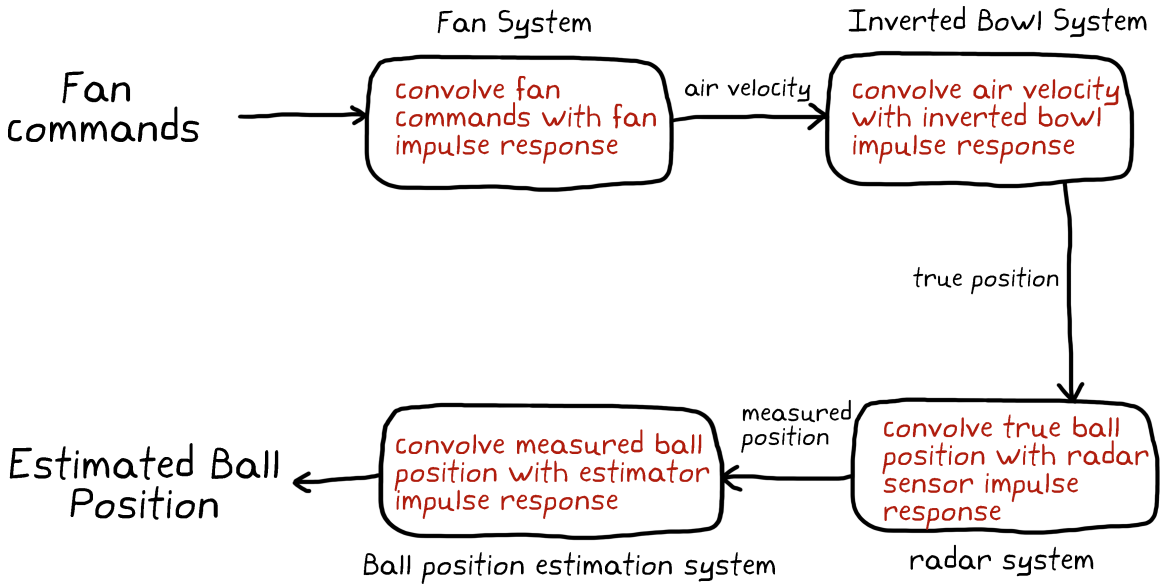
```
f = [1 2 3];
g = [3 0 1];
w = conv(f,g)


w = 3 6 10 2 3
```

This corresponds to the polynomial $3x^4 + 6x^3 + 10x^2 + 2x + 3$.

Convolution gives us the ability to determine an LTI system's response to any arbitrary input as long as we know the impulse response of the system. Going back to our inverted bowl problem we now have a way of stepping our fan commands through each separate system in order to determine how the total system behaves. We would first convolve[9] the fan commands with the impulse response of the fan. The output would be the air velocity which we would then convolve with the impulse response of the inverted bowl system. That output would be the true ball position that we would convolve with the radar sensor impulse response to generate the measured ball's position. Finally, we'd convolve that output with the estimator impulse response to give us the estimated position that results from the initial fan commands.

[9] Yes, the verb is convolve. It's not convolute, convolt, or convolutionize.



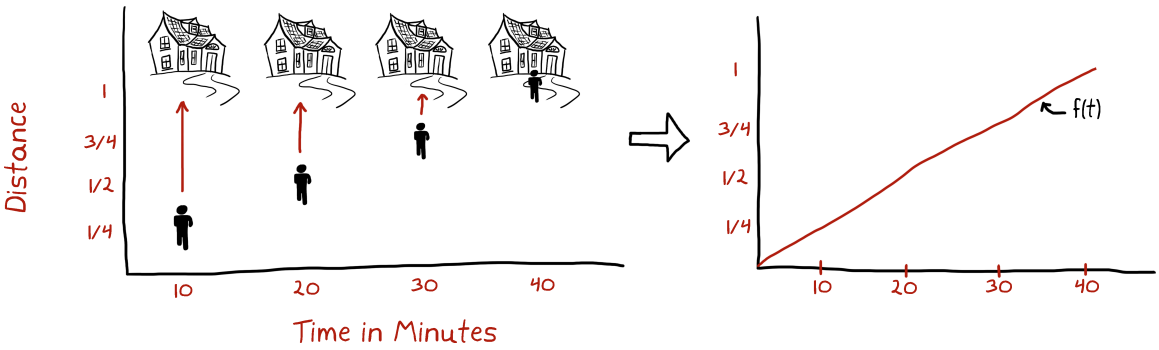
We've successfully played our commands through the entire system but perhaps you see a problem with this method. Namely, the convolution integral seems pretty messy and preforming that integration for arbitrary inputs would be overly cumbersome. You are correct, it is no fun! Not only is it difficult to perform the integration but convolution doesn't allow us to easily combine several systems into a single large system. For example, if we wanted to produce the differential equations that relate the fan commands directly to the estimated ball position without having to go through each step along the way then convolution isn't going to help us.

What will help us? You guessed it, transfer functions. We can perform convolution with transfer functions as well, but the good news is that we do that using multiplication rather than integration. In order to continue on our journey to the transfer function, we need to leave the comfort of the time domain and venture out into the frequency domain.

2.4 The Frequency Domain and the Fourier Transform

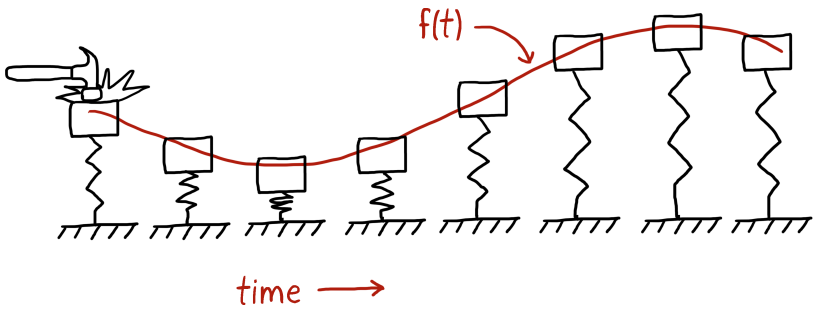
In this section, we will cover a brief introduction to the frequency domain and the Fourier Transform. I say it's brief because a full treatment of the material would be a whole book on its own. The goal of this section, however, is not to fully understand the math involved in getting to and from the frequency domain but rather to provide just enough information to grasp its importance to transfer functions and to understand why the frequency domain makes our lives easier as control engineers. At first glance going to the frequency domain will seem like an inconvenient step but as you will learn, and practice throughout this book, it will be well worth the effort.

It's easy to understand the physical meaning of time domain equations because we experience life in the time domain. You have probably worked with equations that had, for example, parameters that changed as a function of time. $Distance = velocity * time$ is a well known kinematic equation that describes an object's motion while moving at a constant velocity. Think back to when you were traveling somewhere - for example walking to your friend's house. Perhaps after 10 minutes you were a fourth of the way there so you do some quick mental math and predict that it will take you about 30 more minutes to get to their house. Plotting the journey through time on one axis and the journey through space on another axis produces your motion as a function of time, $f(t)$.



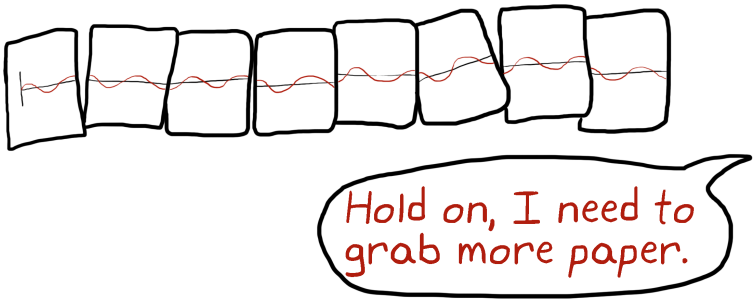
It makes sense to think of this type of equation in the time domain and it's a little comforting to be able to relate the result back to a physical experience.

However, let's move past the walking example and imagine a mass sitting on top of a spring. If we applied an impulse force to the mass it would start to bounce up and down like a jack-in-the-box. If there was no damping in the system or no loss of energy in any way[10] then the mass would continue to bounce up and down forever.

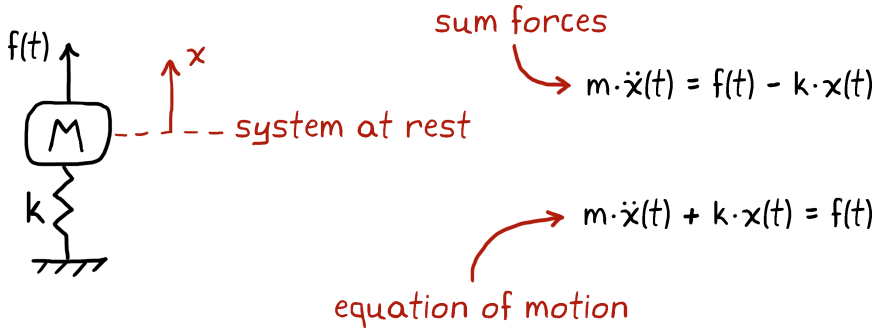


Forever is pretty hard to graph in the time domain, but more importantly, it can be difficult in some situations to observe meaningful behavior when you only see a system's time response.

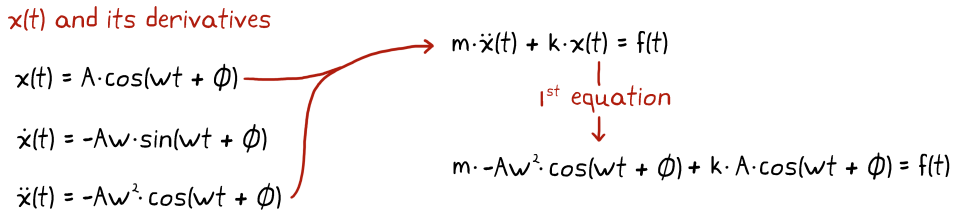
[10] We can write an equation for a system that doesn't lose energy but just like a linear system, this can't physically happen. In real life, there is always a loss of energy - usually from friction which generates heat which then leaves the system through convection, conduction, and radiation.



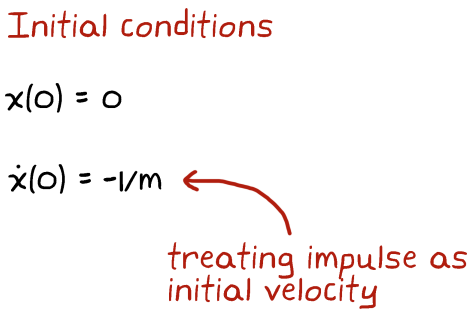
For this particular system we can fully characterize the response by defining just three parameters; the frequency of the bouncing, the amplitude of the bouncing, and the phase shift corresponding to the starting position of the mass. We can see this clearly by deriving the time domain equations of motion for the system.



From here we can solve the differential equation, $m\ddot{x}(t) + kx(t) = f(t)$, by assuming the solution is in the form $x(t) = Acos(\omega t + \phi)$ and then solving for the three unknown coefficients, A, ω, and ϕ. Since there are three unknowns we need three equations to solve for them. For the first equation we can calculate the 2^{nd} derivative of $x(t)$ and then with $x(t)$ plug them into our equation of motion.



The last two equations come from the two known initial conditions. We know the initial position is at rest resulting in $x(0) = 0$. We also know from earlier that the impulse force generates instantaneous velocity equal to 1 divided by the mass of the object which gives us $\dot{x}(0) = -1/m$. It's negative since the force is applied in the negative direction.



Since we're accounting for the input force as an initial velocity we set the force to zero in the first equation. At this point a little algebra gives us the frequency of the bouncing, $\omega = \sqrt{\frac{k}{m}}$, the initial starting point for the bouncing, $\phi = \frac{\pi}{2}$, and the amplitude of the bouncing, $A = \frac{1}{\sqrt{km}}$.

$-Aw^2 \cdot \cos(wt + \phi) \cdot m + A \cdot \cos(wt + \phi) \cdot k = 0$

$A \cdot \cos(\phi) = 0$

$-Aw \cdot \sin(\phi) = -1/m$

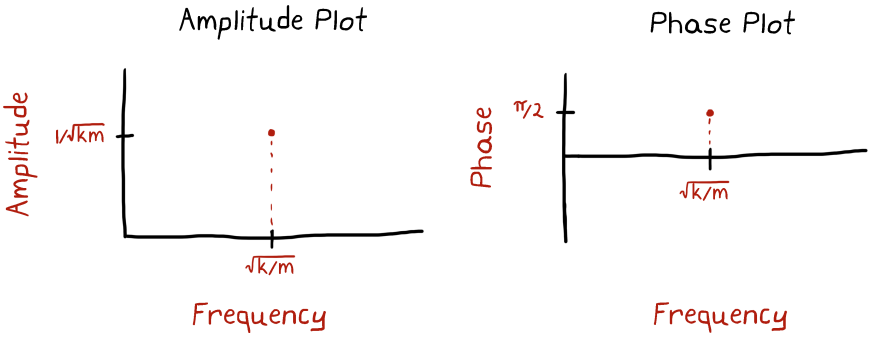
Solve these three equations to get

$w = \sqrt{k/m}$

$\phi = \pi/2$

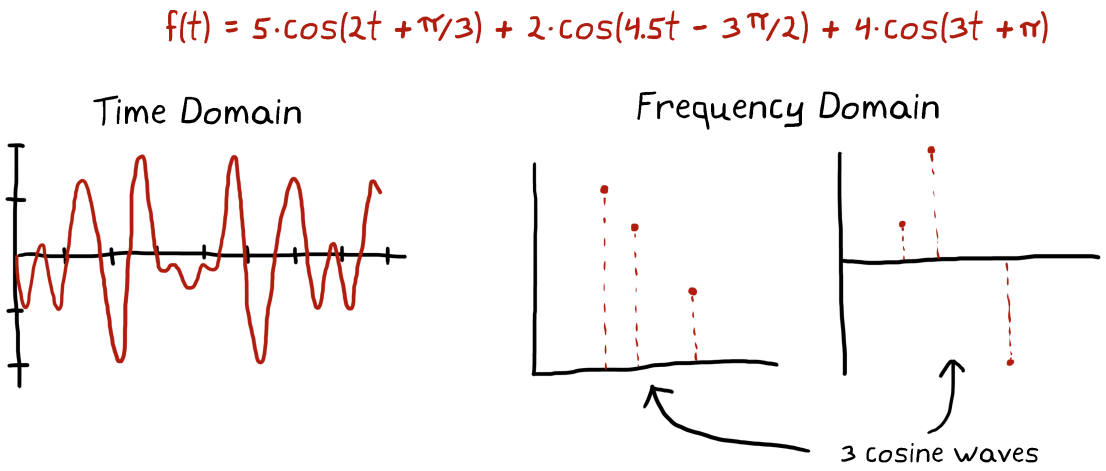
$A = 1/\sqrt{k \cdot m}$

We've just shown that the motion of the block is described in the time domain by a simple cosine wave, $x(t) = \frac{1}{\sqrt{km}} cos(\sqrt{\frac{k}{m}}t + \frac{\pi}{2})$, and if we wanted to plot this in the time domain then we'd need an infinite amount of paper. However, to recreate a cosine wave we only need to know its frequency, amplitude, and phase and we can plot that information easily using two separate graphs; one for amplitude and one for phase. When we start thinking about a signal in terms of frequency, amplitude, and phase we've moved out of the time domain and into the frequency domain. That is we are thinking of a signal in terms of the characteristics of the frequencies that make it up rather than how it changes over time.



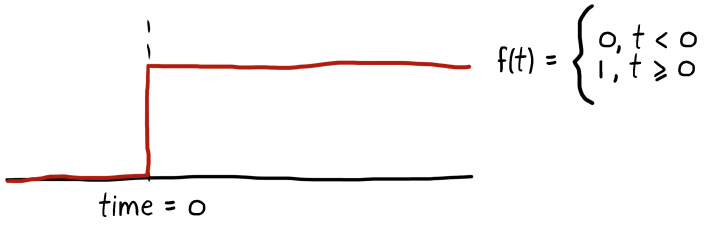
You can really see the benefit of viewing a signal in the frequency domain when your solution is made up of more than one cosine wave. In the time do-

main the signal could look random and chaotic, but in the frequency domain, it is represented very cleanly.

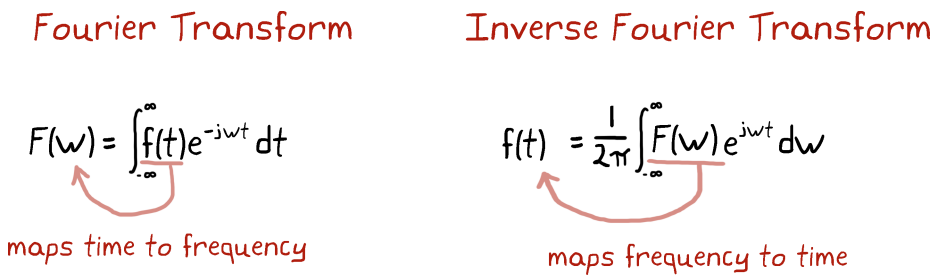


It's important to realize that we aren't losing any information when we represent a signal in the frequency domain, we're just relating the exact same information in a different format.

If our time domain equation is just a series of cosine waveforms, as the previous example was, then it's easy to see how you could transform that equation to the frequency domain - just pick out the three parameters for each cosine and plot them. However, it is not always the case that a time domain signal is written as the sum of a set of cosine waves. In fact, it is more often not the case. For example an extremely simple, non-cosine, time domain function is 0 for $t < 0$ and 1 for $t \geq 0$.



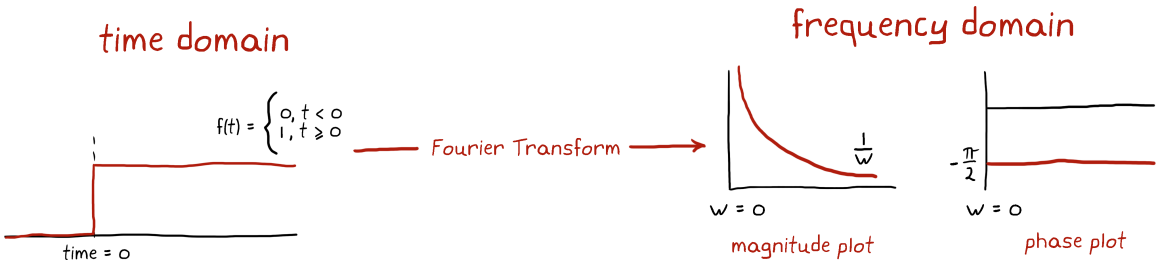
Even though this looks decidedly nothing like a cosine wave we can still represent this step function in the frequency domain - that is we can convert it into an infinite number of cosines, each at a different frequency and with a different amplitude and phase. This conversion is done using a very powerful tool called the Fourier transform. A transform is a mapping between two sets of data, or domains, and the Fourier transform maps a continuous signal in the time domain to its continuous frequency domain representation. We can use a similar transform, the inverse Fourier transform, to map from the frequency domain back to the time domain[11].



The Fourier transform looks complicated but I assure you that it makes sense as a whole if you spend a little time deciphering each of its parts. Having said that, we're not going to spend that time in this chapter! Instead, I'm asking you to believe that it really does map a signal from the time domain to the frequency domain and back again[12]. That means that if you have an equation as a function of time and perform the Fourier transform integral the result will be a signal as a function of frequency and the values will be related to amplitudes and phases.

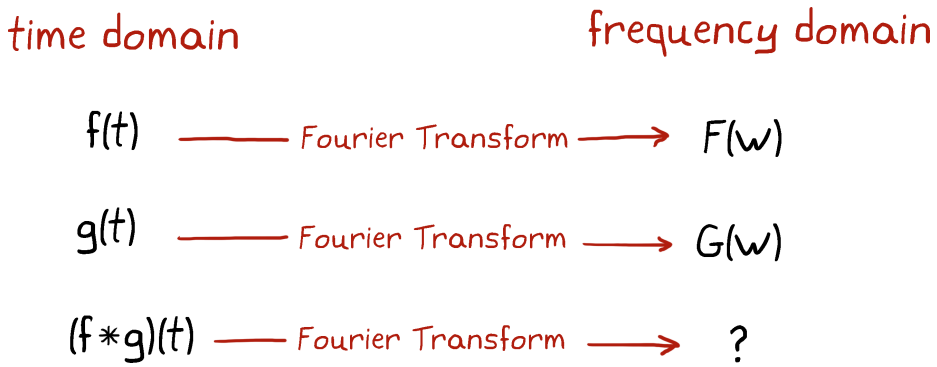
[11] Great, more integrals!

[12] If you are interested in a deeper understanding of the transform there ~~is~~ will be a short explanation in Appendix A



At this point, you might be thinking 'the step function above is represented cleanly in the time domain and since it's made up of an infinite number of cosine waves it's much more complicated in the frequency domain.' That's true, but remember the problem we're trying to simplify in this chapter is how to represent a system in a way that allows us to easily manipulate it and analyze it, not necessarily how to simplify plotting it. We got to the concept of convolution in the last section but got stuck because we realized that it is a difficult integral for any generic signal. Also, convolution doesn't provide a simple way of combining several systems together to create a single equation for the larger system.

To see how the frequency domain helps us simplify our problem let's consider the following chart.



We know that the Fourier transform maps functions $f(t)$ and $g(t)$ to $F(\omega)$ and $G(\omega)$, respectively, where $f(t)$ might represent an input signal and $g(t)$ might represent the system's impulse response. In the time domain we can convolve the two to get the system's response to input $f(t)$, but how can we manipulate $F(\omega)$ and $G(\omega)$ in the frequency domain to produce a similar result? Or another way of putting it, what is the Fourier transform of $(f * g)(t)$? I think you'll find the simplicity of it quite amazing.

2.5 Convolution versus Multiplication

In this section, we are going to prove the convolution theorem which states that the Fourier transform of convolution is just the multiplication of the individual Fourier transforms. To prove this we are going to walk through the Fourier transform of the convolution integral[13]. You'll probably never have to prove this outside of a homework assignment or exam question, however, walking through it at least once is important because it forces us to dedicate several pages and a little bit of time to this topic and hopefully it will help you to remember the concept. Every single time you multiply two transfer functions you are taking advantage of the convolution theorem and remembering that will give you a better intuition as to what multiplication is actually accomplishing.

To start the convolution theorem proof, let's remind ourselves of the convolution integral and the Fourier transform.

[13] Warning! This section is going to be math heavy.

Convolution Integral

$$(f*g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

Fourier Transform

$$F(w) = \int_{-\infty}^{\infty} f(t)e^{-jwt}\,dt$$

To take the Fourier transform of the convolution integral we just replace $f(t)$ with $(f*g)(t)$, which of course is the convolution integral itself. The fancy $\mathscr{F}$ just denotes that we are taking the Fourier transform of what's inside the parentheses.

This fancy F means take the Fourier transform

$$\mathcal{F}\left((f*g)(t)\right) = \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau\right] e^{-jwt}\,dt$$

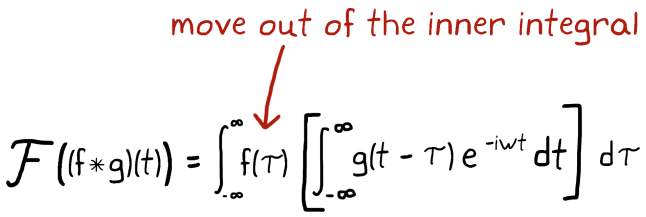
This looks rather complicated but let's begin to pick away at it and see how it can be simplified. The first thing we do is rearrange the order of the integral. Right now we perform the summation of the inner integral with respect to τ and then do the outer integral with respect to t. A double integral can be integrated in either order as long as you are careful to transform the limits of integration appropriately. Luckily, our limits are both from $-\infty$ to ∞ so rearranging the integrals is just a matter of pulling $e^{-j\omega t}dt$ in and pulling $d\tau$ out.

Now this is summation of dτ

and this is summation of dt

$$\mathcal{F}\left((f*g)(t)\right) = \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,e^{-iwt}\,dt\right] d\tau$$

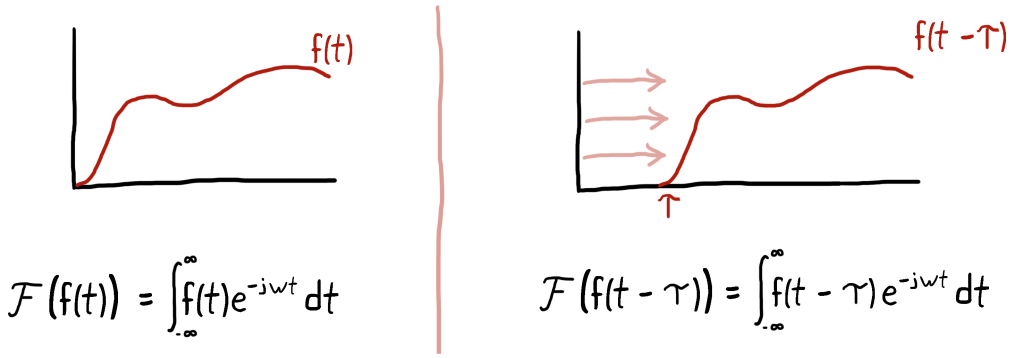
At this point, we can move $f(\tau)$ out of the inner integral because it is just a function of τ and therefore a constant when integrated with respect to t.



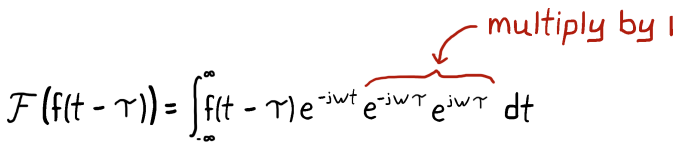
There is something special about the equation inside of the square brackets - it can be replaced with $e^{-j\omega\tau}G(\omega)$. To prove this we need to take a step back and talk about the Fourier transform shift theorem first.

The Fourier transform shift theorem

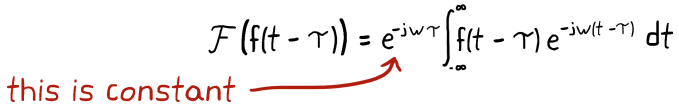
The image on the left shows the Fourier transform of an arbitrary function, $f(t)$. The image on the right shows that same function shifted or delayed by time, τ. The question we want answered is what is the Fourier transform of that shifted function? We start by replacing $f(t)$ with $f(t-\tau)$.



Then to get the time aligned to the same frame within the integral we can multiply the function by 1 (so we don't change anything) but represent 1 as $e^{-j\omega\tau}e^{j\omega\tau}$. Since these exponentials are a function of τ, and therefore constant with respect to t, we can put them inside of the integral.



We can pull $e^{-j\omega\tau}$ out of the integral[a] and combine the two remaining exponentials into $e^{-j\omega(t-\tau)}$.



This little bit of mathematical trickery has resulted in both functions inside of the integral to be functions of the same time frame, that is they are both functions of $t-\tau$. We can replace our time frame with T and adjust the integral limits accordingly. However, since we're integrating over all time, a function that is shifted by a finite value has no impact on the integration limits. You'll notice that what we are left with is just the standard Fourier transform.

substituting $\mathcal{T} = (t - \tau)$

$$\mathcal{F}\left(f(t-\tau)\right) = e^{-jw\tau}\int_{-\infty}^{\infty} f(\mathcal{T})\, e^{-jw\mathcal{T}}\, d\mathcal{T}$$

this is just the Fourier transform

So this is very interesting, the Fourier transform of a shifted function is just the Fourier transform of the original function multiplied by a constant related to the length of time of the shift, $e^{-j\omega\tau}$.

$$\mathcal{F}\left(f(t-\tau)\right) = e^{-jw\tau}\,\mathcal{F}\left(f(t)\right) = e^{-jw\tau}\,F(w)$$

[a] We didn't really need to put it in there in the first place but I find the extra step makes more sense

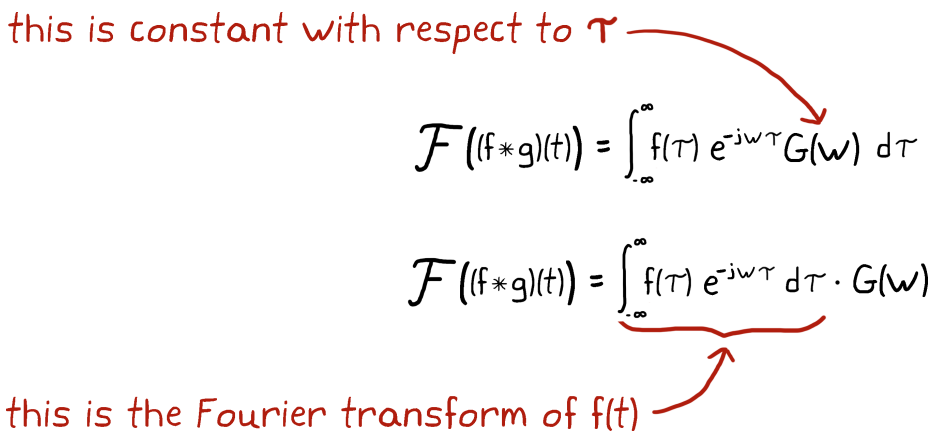
Using our new found knowledge of the Fourier shift theorem we can plainly see that the function inside of the square brackets is really just the Fourier transform of $g(t-\tau)$, or the delay constant $e^{-j\omega\tau}$ times $G(\omega)$.

this is the Fourier transform of $g(t-\tau)$

$$\mathcal{F}\left((f*g)(t)\right) = \int_{-\infty}^{\infty} f(\tau)\left[\int_{-\infty}^{\infty} g(t-\tau)\, e^{-jwt}\, dt\right] d\tau$$

and we can replace it with $e^{-jw\tau}G(w)$ using the shift theorem

Since $G(\omega)$ is constant with respect to τ we can pull it out of the integral, and what we are left with is the Fourier transform of $f(t)$.



So after all of that, we can safely conclude that the Fourier transform of the convolution integral really is just the multiplication of the individual Fourier transforms.

$$\mathcal{F}\{(f*g)(t)\} = F(w) \cdot G(w)$$

We're not at the definition of the transfer function just yet, but keep in mind what we've just shown here. When you're working in the frequency domain and you multiply two functions you are really accomplishing the same result as convolution in the time domain. So if you have a frequency domain representation of your system's impulse response and arbitrary input signal then you can calculate the system's response to that input by multiplying the two.

Transfer functions are not represented entirely in the frequency domain, however. They are in a higher order domain called the *s* domain where one dimension is in fact frequency, but the second dimension is exponential growth and decay. I know, this sounds crazy, but just like everything

we've covered so far, it is quite interesting and intuitive when you really understand it. So let's continue on!

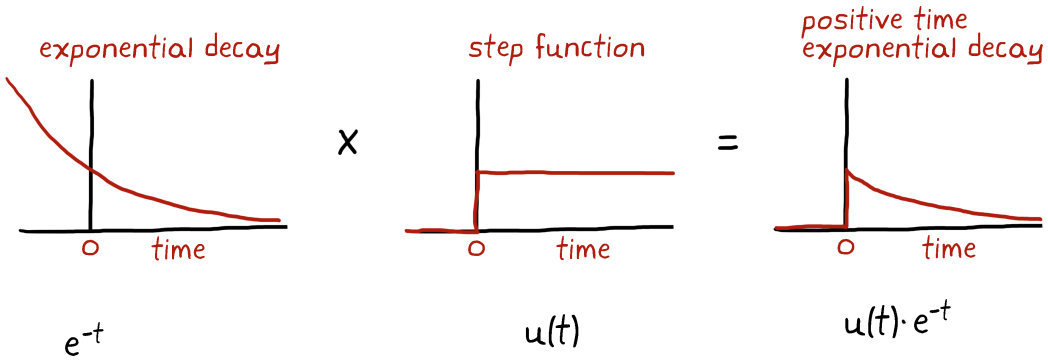
2.6 The *s* domain and the Laplace Transform

With each section of this chapter, we're drawing ever closer to understanding the transfer function, but this section is the most important yet. If up to this point you've just been quickly glossing through the chapter - hoping to absorb the information as fast as possible so you can go back to doing something fun - I encourage you to slow down now and really try to understand this section. There are many system analysis and control techniques that you will be exposed to that use transfer functions as the method for representing the system. Therefore, you will do yourself a huge favor by spending some time to fully grasp the Laplace transform and the *s* domain. Once you understand these two concepts, everything else involving transfer functions in the future will be much easier to learn and apply.

2.6.1 Remember the Fourier Transform!

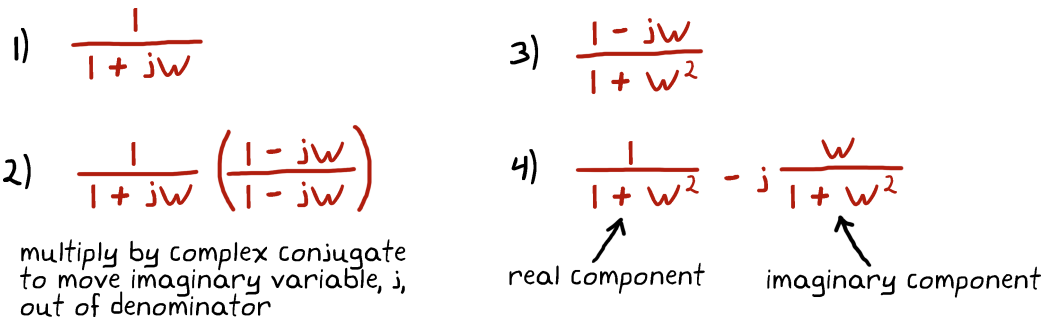
In the last section we took for granted that the Fourier transform maps a signal in the time domain to the frequency domain and I alluded to the fact that the signal in the frequency domain has two parts; one part that is related to the amplitude of the resulting cosine waves and another part that is related to their phase. We can plot each of those parts on their own separate graph where the x-axis is the frequency of the cosine waves and the y-axis is the magnitude and phase, respectively.

Let's show a quick example starting from a time domain function and ending with the two plots, magnitude, and phase, in the frequency domain. The time domain function we'll use is a simple exponential decay, e^{-t}. However, we want the signal to be zero for all values of $t < 0$ so we'll multiply it by a step function, $u(t)$. This produces a function whose value is zero for negative time and whose value is 1 starting at $t = 0$ and then decays exponentially with positive time.

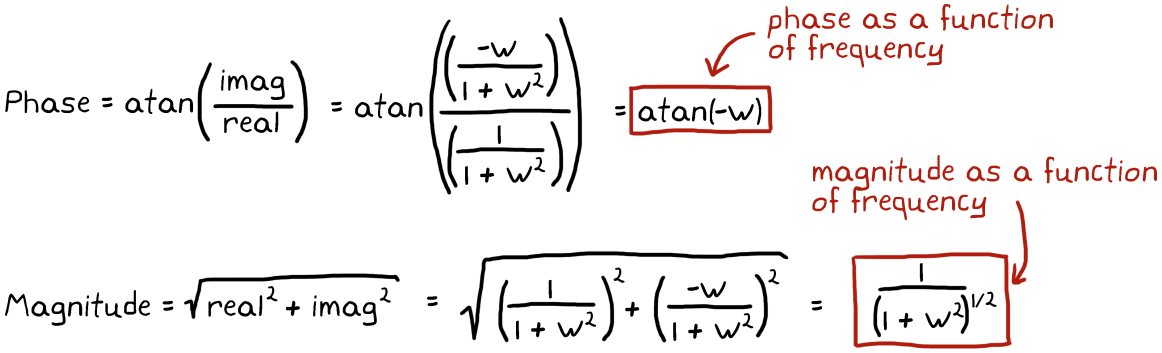


We can solve the Fourier transform for $f(t) = u(t)e^{-t}$, however, since this is a common time domain function we can simply look up its frequency domain representation in a Fourier transform table[14]. From any Fourier transform pair table online you can find it to be $\frac{1}{1+j\omega}$. Since this is a complex function it is made up of two-dimensional numbers that have real and imaginary components. We can rewrite this function to separate out those two parts.

[14] You are more than welcome to solve the Fourier transform integration to prove this to yourself - it's a good exercise - but for the purpose of writing transfer functions you'll find that for both Fourier transforms and Laplace transforms you'll more often than not just memorize the common ones or look up the conversion in a table. I'm not necessarily condoning this laziness, I'm just stating this is usually the case from my experience in engineering.

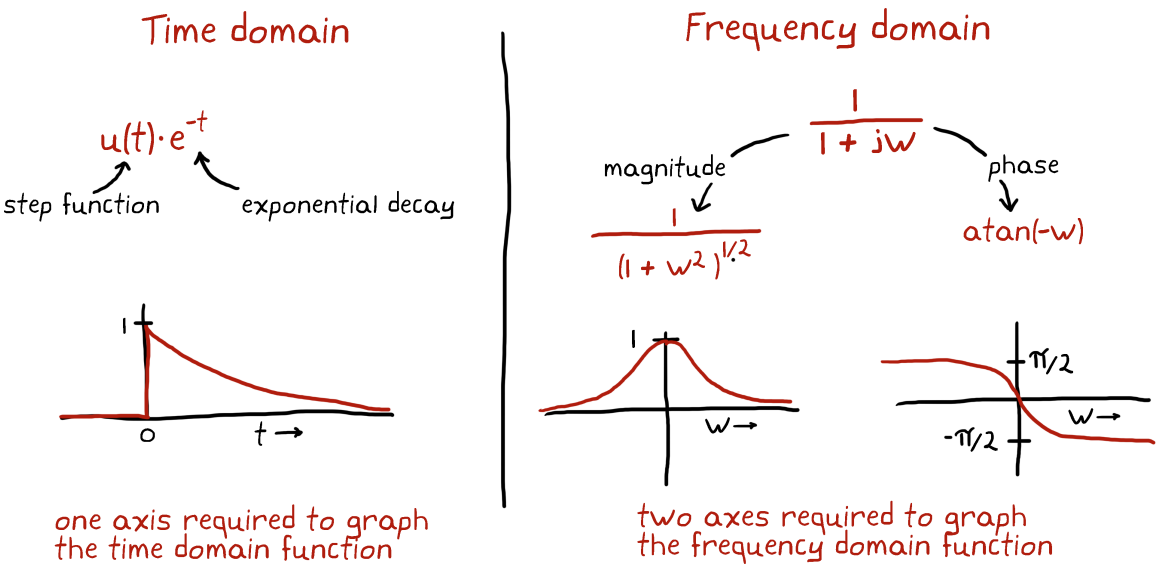


At this point, calculating the magnitude and phase is a matter of converting the rectangular coordinate representation, which are the real and imaginary parts, to polar coordinate representation[15].

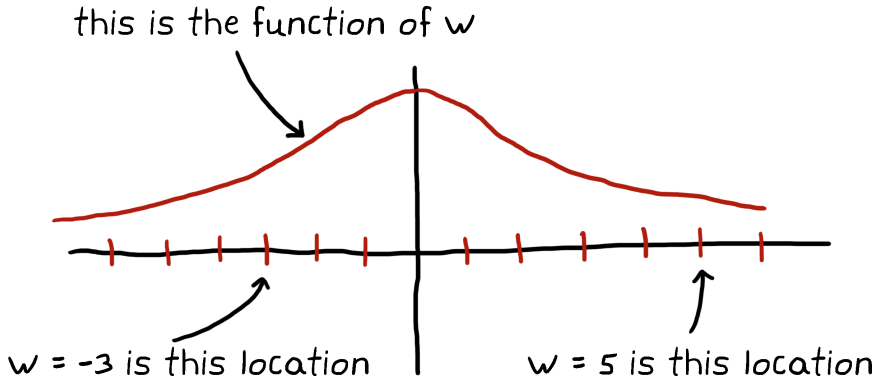


What we did is take a one-dimensional time domain function, $u(t)e^{-t}$, and turn it into a two-dimensional frequency domain function. In the time domain, the single dimension is the value of the function across all time. In the frequency domain, the two dimensions are the real and imaginary components which, through some additional algebra, are the magnitude and phase of the cosine waves that make up the original time domain function.

[15]This might be confusing for this particular revision of the book because I haven't written the appendix covering the mechanics of the Fourier transform yet. However, a better explanation will be provided in that section in a future release of the book.



The two graphs that are created in the frequency domain are a function of ω. In other words, the value of ω tells us *where* on the frequency line we are. The idea that the value of ω is a location on the frequency line seems like a really simple concept for me to state in this section, but keep it in mind as we move on to the s plane.



2.6.2 The s Plane

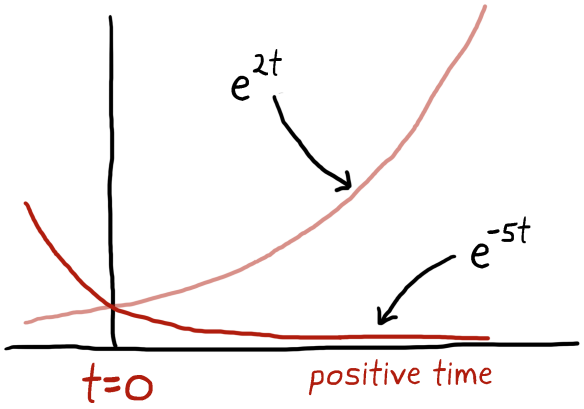
The Laplace transform takes the idea of the Fourier transform one step further. Instead of just cosine waves, the Laplace transform decomposes a time domain signal into both cosines *and* exponential functions. So you can imagine that for the Laplace transform we need a symbol that represents more than just frequency, ω, it needs to also account for the exponential aspect of the signal. This is where the variable s comes in.

s is a complex number, which means that it contains values for two dimensions; one dimension that describes the frequency of a cosine wave and the second dimension that describes the exponential term. It is defined as $s = \sigma + j\omega$. Let's step back a bit and explain this in more detail.

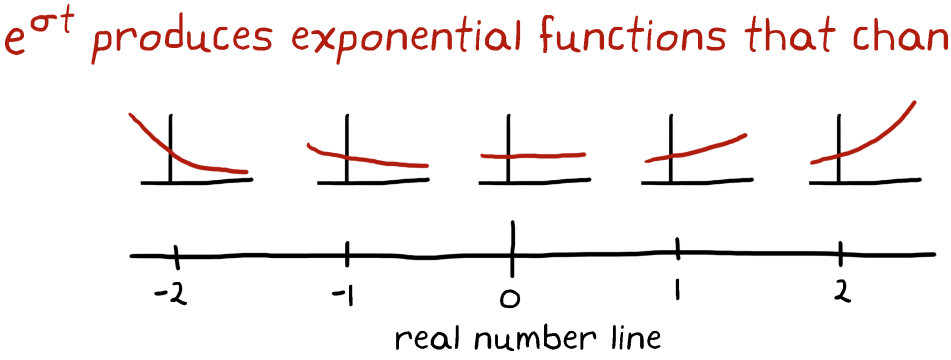
Exponential functions that have imaginary exponents, such as e^{j2t}, produce two-dimensional sinusoids through Euler's formula[16], $e^{j\omega t} = cos(\omega t) + j\,sin(\omega t)$. We've already seen how the variable ω is the frequency of the sine and cosine waves as well as describing the location on the ω line.

For exponential functions that have real numbers for exponents, negative real numbers give us exponentially decaying signals and positive real numbers give us exponentially growing signals. Two examples are e^{2t}, which grows exponentially for all positive time, and e^{-5t}, which decays exponentially for all positive time.

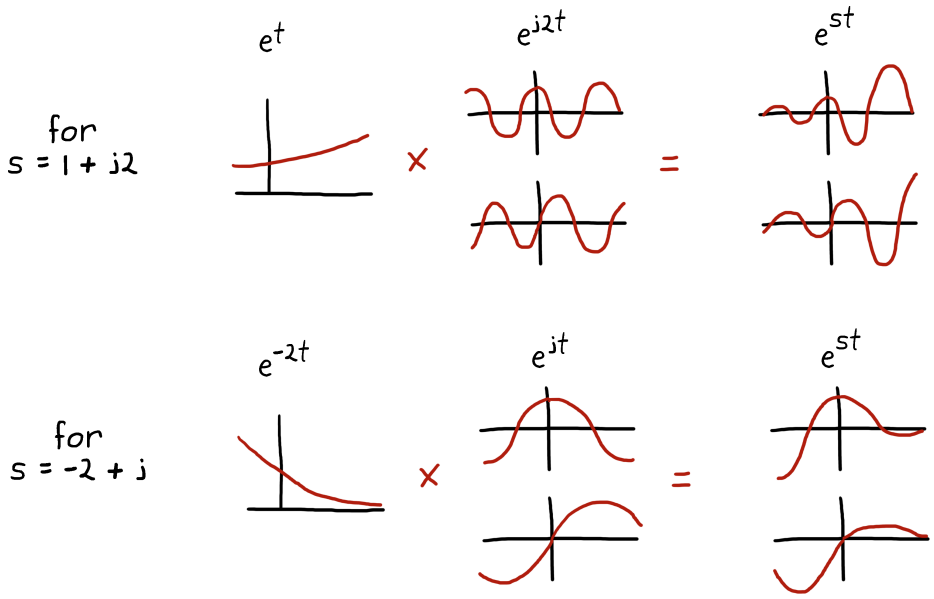
[16]Once again sorry for any confusion right now. The appendix on the Fourier transform will cover Euler's formula in more detail. For now, the important thing to know is that raising an exponent to an imaginary number produces cosine-like waves rather than a function that grows or decays exponentially.



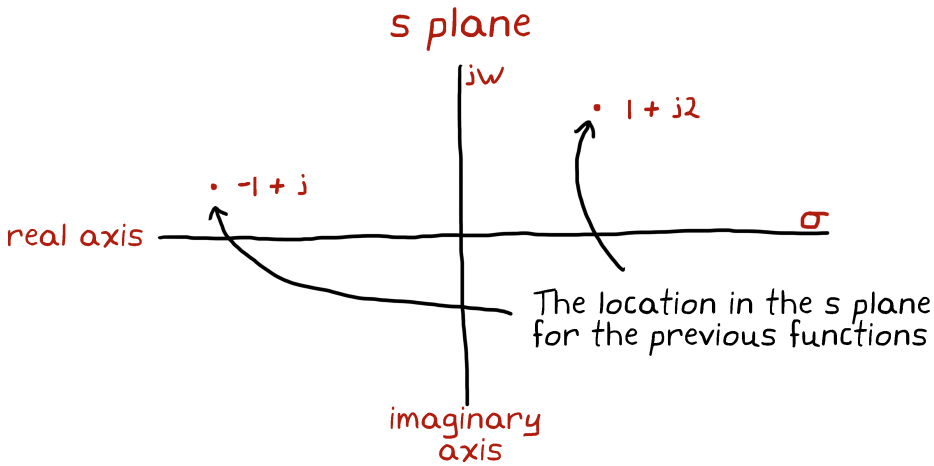
We can replace the real number in the exponent with the variable σ to give us $f(\sigma) = e^{\sigma t}$. Just like with ω, σ gives us a way to define a location on a real number line that corresponds to a particular exponential function. As you move away from the origin, the absolute value of the real number becomes larger and thus the signal decays or grows at a faster rate.



Now let's think about our new variable s which has both a real and imaginary component. Therefore, the equation e^{st} is really just an exponential function multiplied by a sinusoid, $e^{st} = e^{(\sigma+j\omega)t} = e^{\sigma t}e^{j\omega t}$.

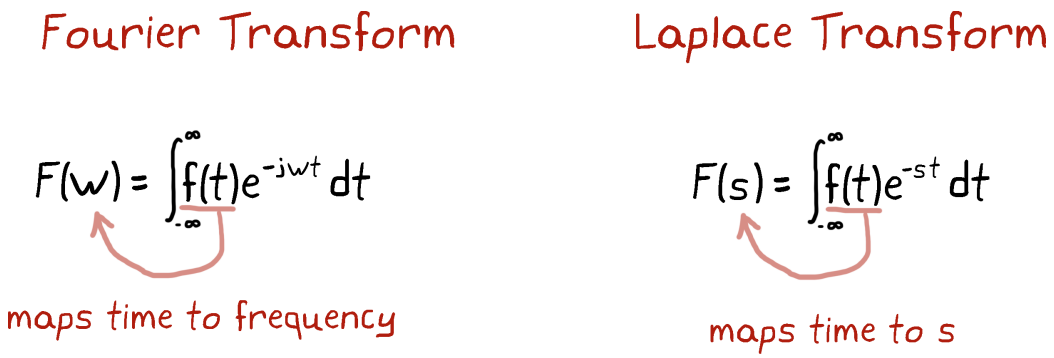


It would be cumbersome to have two separate number lines to describe s; one for frequency and one for the exponential rate. Therefore, instead we combine them into a two-dimensional plane where the real axis is the exponential line and the imaginary axis is the frequency line. The value of s provides a location in this plane and describes the resulting signal, e^{st}, as function of the selected ω and σ.



2.6.3 The Laplace Transform

With our new found knowledge of the variable s and how the function e^{st} produces a signal that has both an exponential and sinusoidal component, we can now move on to describing the Laplace transform. An intuitive way to understand the Laplace transform is by contrasting it with the Fourier transform. Mathematically the two are exceedingly similar and this can lead you to believe that we use their result in the same way. You'll soon see that this is not the case.



Obviously, the difference is that we've replaced jw with s. However, since $s = \sigma + j\omega$, rather than replacing jw what we are actually doing is adding the real component, σ, to the equation.

If we expand s in the Laplace transform and rearrange the equation something interesting emerges. We find that our original time domain function $f(t)$ is first multiplied by an exponential term $e^{-\sigma t}$ and then we end up taking the Fourier transform of the product $f(t)e^{-\sigma t}$.

$$F(s) = \int_{-\infty}^{\infty} f(t)e^{-st}\,dt$$

$$F(s) = \int_{-\infty}^{\infty} f(t)e^{-(\sigma + j\omega)t}\,dt$$

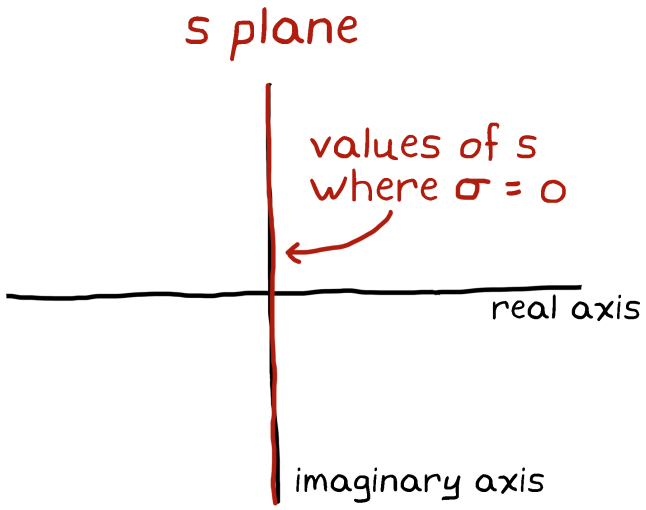
$$F(s) = \int_{-\infty}^{\infty} [f(t)e^{-\sigma t}]e^{-j\omega t}\,dt$$

Multiply f(t) by exponential

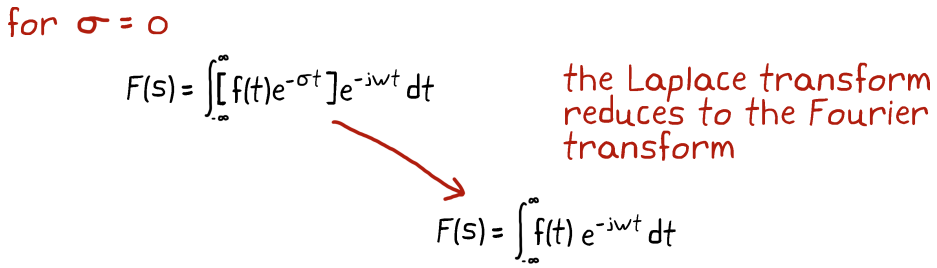
then take Fourier transform of result

What does that mean and why is it interesting? Well, this gives us a way to interpret the Laplace transform graphically.

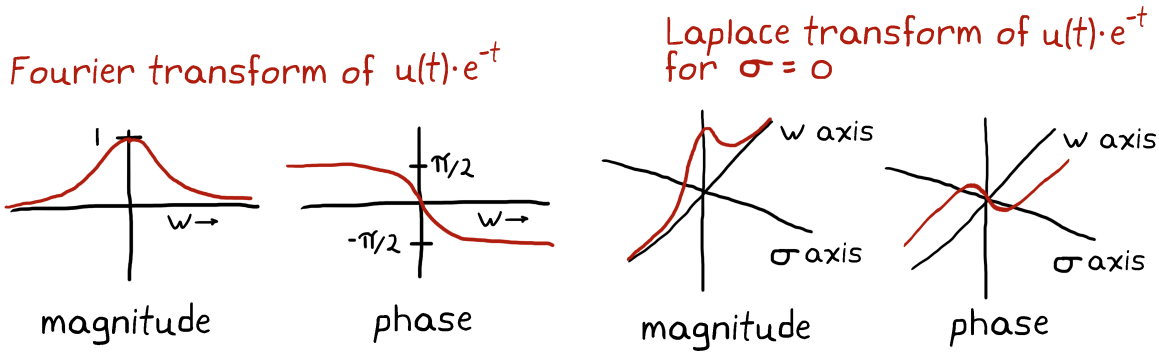
Let's refer back to the s plane and look at the imaginary axis. This is the line where $\sigma = 0$.



Since $\sigma = 0$, the Laplace transform for values of s along this line is exactly equal to the Fourier transform.



Remember that the results of the Fourier transform are a set of two-dimensional numbers that represent magnitude and phase for a given frequency. The results of the Laplace transform are still the same two-dimensional numbers, but now we plot them on a 3-dimensional s plane rather than just the along the frequency line.

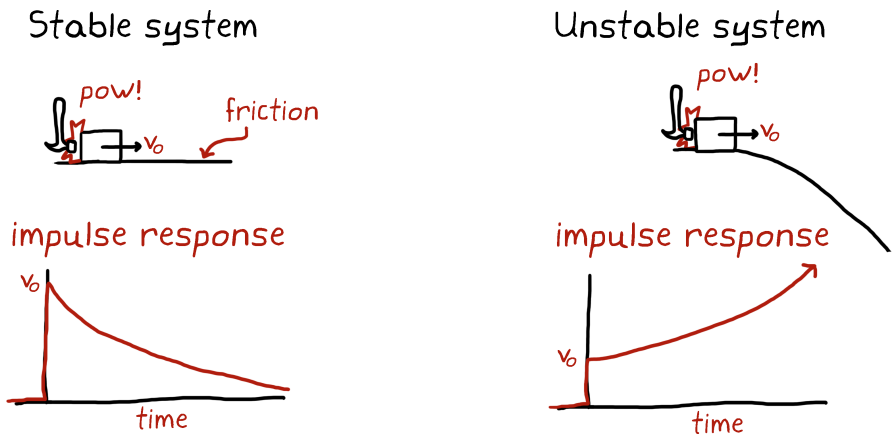


The Region of Convergence

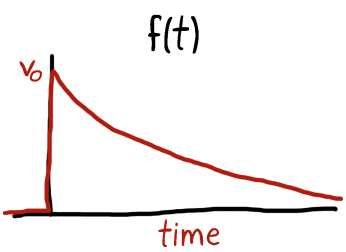
I will accidentally mislead you if I don't clarify a statement I made. I stated that the Laplace transform is *exactly* equal to the Fourier transform

for the case when $\sigma = 0$, but this is only true when the $\sigma = 0$ line is within something called the Region of Convergence, or RoC. As you can probably gather from the name, the RoC is the area in the s plane where the Laplace transform is absolutely integrable - or another way of putting it is that the integral converges - yet another way of putting it is if you sum up the area under the absolute value of a signal you're left with a finite value.

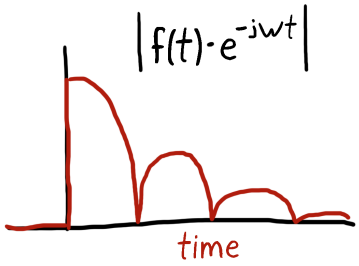
To understand this, let's look at two signals; the first is the impulse response for a stable system - notice the impulse the system response dies out over time. The second is the impulse response for an unstable system - notice the system response continues to grow over time.



We can take the Fourier transform of the stable response because after you multiply it by $e^{-j\omega t}$, essentially multiplying it by $cos(\omega t) + jsin(\omega t)$, the signal continues to decay and therefore the integral of the absolute value produces a finite sum.

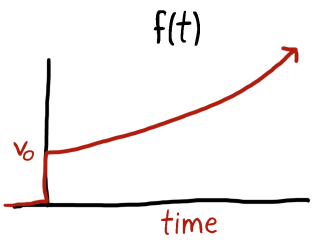


impulse response
is stable

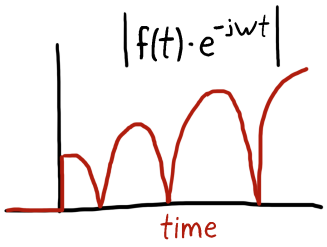


area under the curve is finite
so the integral converges

However, we can't take the Fourier transform of the unstable response because after you multiply it by $e^{-j\omega t}$ the signal continues to grow. If you integrate this signal you get an infinite value and so it lies outside of the RoC.



impulse response
is unstable

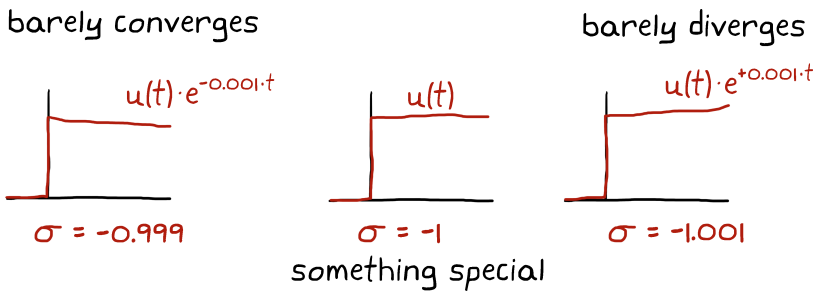


area under the curve is infinite
so the integral does not converge

We *can* take the Laplace transform of an unstable impulse response, however, because there are other areas of the s plane that are within the RoC.

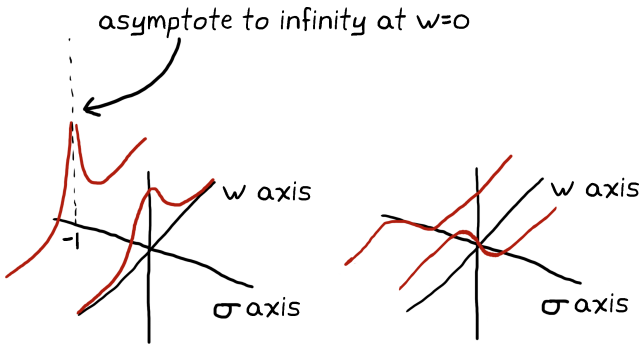
So far we've just filled out a single sliver of our s plane, the $\sigma = 0$ line. We can fill out another sliver, say the $\sigma = -1$ line, by taking our time domain signal, $u(t)e^{-t}$, multiplying it by the exponential $e^{-\sigma t}$ when $\sigma = -1$, and then taking the Fourier transform of the result. In this case, the two exponential functions cancel out and we're left with just the step function, $u(t)$.

This is a trick question though because the the Fourier transform of $u(t)$ does not converge. It's outside of the region of convergence, however, just barely. Imagine we chose a σ that produced a near step function that is slowly decaying. With this, the Fourier transform will converge. Conversely, imagine we chose a σ that produced a near step function that is slowly growing. In this case, the Fourier transform is even further from converging than just the simple step function.



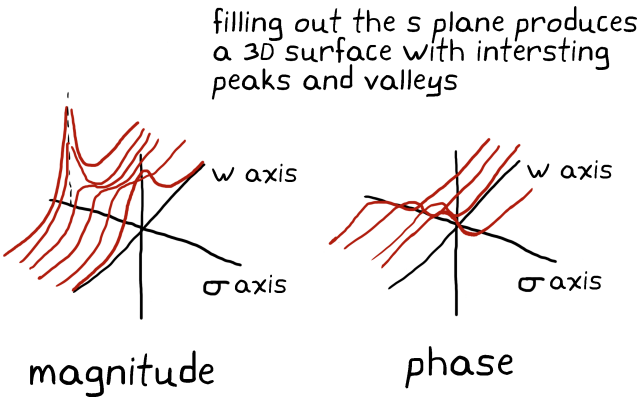
Therefore, there was something special about the location in the s plane that produced an integral that existed right on the cusp of converging and diverging. We could graph this new line on our plot at $\sigma = -1$. You'll see

from the graph, and from the result of the Fourier transform if you do the math, that there is a point right at $s = -1$ that goes to infinity.



If you've been very keen while reading this section you'll have realized that the impulse response of our system, $u(t)e^{-t}$, produced an interesting point in the s plane at $s = -1$, which, for e^{st}, equals e^{-t}. It's no coincidence that both our impulse response function and the interesting point in the s plane both contain e^{-t}. In fact, that is exactly what we're doing with the Laplace transform; we're probing the time domain function with e^{-st} across the entire s plane to see what it's made of. We are basically breaking it down into its base frequencies and exponential properties.

So far we've found one point in the s plane that produced an interesting result, but are there others? We could continue to fill out this graph manually one at a time by choosing a σ, pre-multiplying our signal by it, and taking the Fourier transform.



I think you can see the problem with this method of filling out the plane. It'll take an infinite number of Fourier transforms, one for each of the infinite σ values, to completely fill in 3D map in the s plane. Of course the actual Laplace transform doesn't work like this. When you solve the integral you are performing the infinite number of steps all at once, and rather than graph the resulting surface, we solve for the interesting points algebraically using the s domain function.

$$f(t) = e^{-t} \quad \text{for this impulse response}$$

$$F(s) = \int_{-\infty}^{\infty} f(t)e^{-st}\,dt = \int_{-\infty}^{\infty} e^{-t}e^{-st}\,dt$$

$$F(s) = \int_{-\infty}^{\infty} e^{-(s+1)t}\,dt$$

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

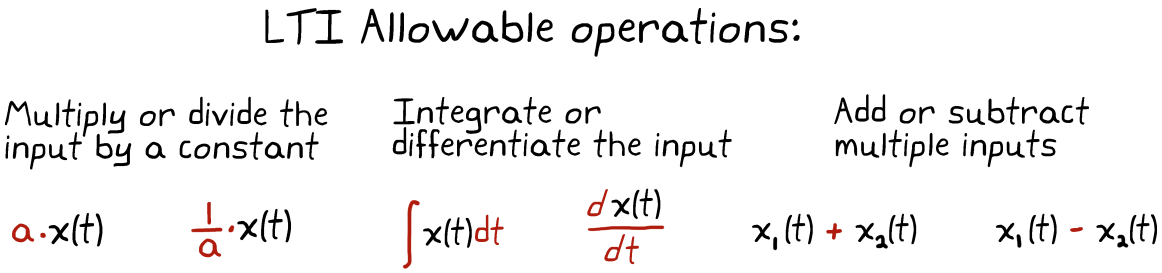
the interesting point is when s = -1 because this function blows up to infinity!

That is pretty awesome, right?

But wait a minute you cry! The Fourier transform decomposes a function into sinusoids. Then the Laplace transform decomposes a function into both

sinusoids and exponentials. So the question is, when does the madness end? You might expect that in the next section I'll introduce a transform that decomposes a signal into sinusoids, exponentials, *and* square waves.

Well, we actually end right here. And there's a good reason why. Remember we are talking about physical systems that can be modeled or approximated as linear and time invariant, and these types of systems can only be modeled using the following six operations.



In the real world, many physical parameters are related to each other through differential equations, and for LTI systems those become ordinary differential equations (ODE)[17]. The important thing to note is that the solution of ordinary differential equations can only consist of sinusoids and exponentials. That's because they are the only wave forms that don't change shape when subjected to any combination of the six legal operations. Think about it. If you take the derivative of a sinusoid it's still a sinusoid. If you take the integral of an exponential it's still an exponential. So you can see how those two wave forms[18] would be the solution to equations that have the form $\ddot{x}+\dot{x}+x=0$. However, if you take the integral of a square wave, for example, you get a sloping step pattern and not another square wave.

[17] More of this to come in the chapter on system identification methods

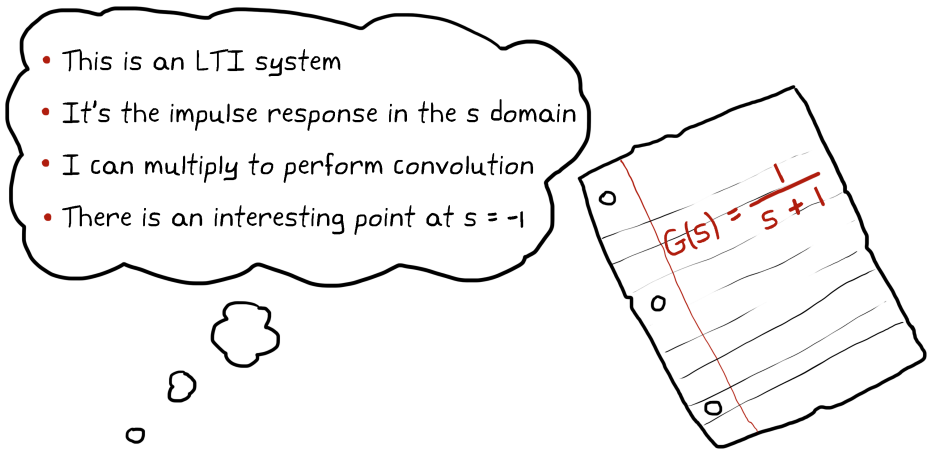
[18] Really it's just the one function, e^{st}, that generates both waveforms.

So it makes sense that we are defining a system's impulse response in terms of these wave forms and only these waveforms. The ubiquity of these types of physical relationships is why the Laplace transform is one of the most important techniques you'll learn for system analysis.

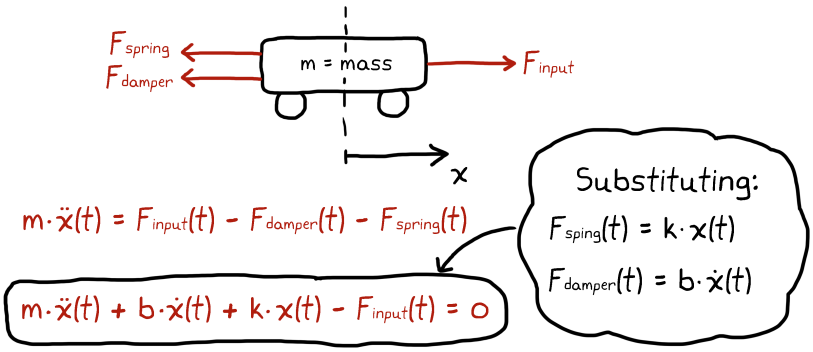
We've now set up all of the background information you'll need to really understand what makes transfer functions so powerful and exactly why they work the way they do. So let's finally put it all together in the next section.

2.7 Putting this all Together: Transfer Functions

The topics we've covered so far in this chapter are valuable in their own right, but you might not realize yet how to use them as an engineer working with transfer functions. In this section, let's see if we can fix that! When you are working with a transfer function we want to get to the point where you aren't just seeing a polynomial fraction that is a function of s, but you are intuitively comprehending what that transfer function represents and how it helps you simplify complex systems.



Recall the rolling cart problem from page 37. This is an exceedingly simple system for sure; probably simpler than anything you'll work on as a control engineer. However, using this simple example we can demonstrate how you can harness the power of transfer functions on a system of any complexity[19]. We began the rolling cart problem by using a free-body diagram to identify and sum the forces acting on the system. From this, we derived the equations of motion; basically, we came up with a 2^{nd} order differential equation that, given some input force, $F_{input}(t)$, we could solve for the position of the cart over time, $x(t)$.

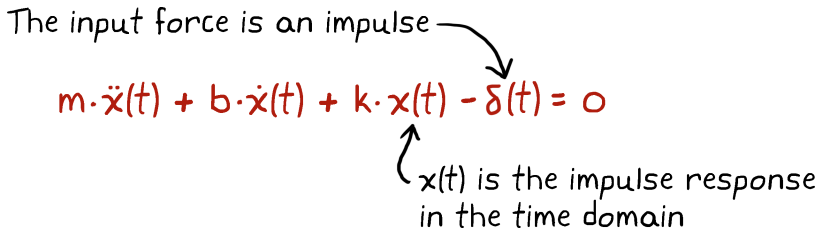


I want to keep reminding you that the process of generating a model of your system, like the one we just derived, is called system identification. This book will devote an entire part to system identification which is why, at this moment, we're not spending a lot of time describing the process. As an engineer, if you're not given a model of your system, and you want to apply the analytical techniques that the majority of this book covers, then you will have to create that model yourself. System identification happens more often than you might think in industry, and therefore is a great skill

[19]Of course, only if the system can be modeled by a linear, time invariant differential equation with a single-input and single-out. Picky, picky! But unfortunately, those are the rules!

to have. For now, let's accept the quick derivation of the rolling cart model and proceed back to discussing transfer functions.

This system model is a perfect candidate to convert into a transfer function; it's a linear, time-invariant system and there is a single input, F_{input}, and a single output, x. Per the definition of a transfer function, we need to take the Laplace transform of the impulse response of our system. How can we get the impulse response from our equation? Well, we set the single input to an impulse function, $F_{input}(t) = \delta(t)$, and solve for the response, $x(t)$.



That seems easy enough right? Possibly not! Solving linear, ordinary differential equations in the time domain can be time consuming. We can make the task easier by taking the Laplace transform of the entire differential equation, one term at a time, and solve for the impulse response in the s domain directly.

A great thing about working with linear systems[20] is that usually we don't have to perform the Laplace transform integration by hand. The Laplace transform for most of the terms that you'll come across in these types of models have been solved many times before and collected into tables. For example, the Laplace transform of an impulse function, $\delta(t)$, is 1. I've collected the other terms in our model into the following small table, but this is by no means a complete list.

[20] and being an engineer rather than a mathematician!

$f(t)$	$F(s)$
$\delta(t)$	1
$x(t)$	$X(s)$
$\dot{x}(t)$	$s \cdot X(s) - x(0)$
$\ddot{x}(t)$	$s^2 \cdot X(s) - s \cdot x(0) - \dot{x}(0)$

From here it's a straightforward operation to transform our time domain model into the s domain; simply take each term and replace with the corresponding s domain equivalent. Note that the initial position, $x(0)$, and the initial velocity, $\dot{x}(0)$, are both zero since the system starts at rest. We can now solve for the system impulse response, $X(s)$, algebraically.

time domain: $m \cdot \ddot{x}(t) + b \cdot \dot{x}(t) + k \cdot x(t) - \delta(t) = 0$

↓ take the Laplace Transform

s domain: $(m \cdot s^2 + b \cdot s + k) \cdot X(s) - 1 = 0$

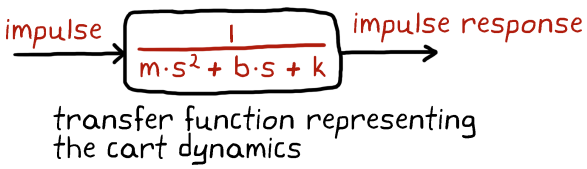
↓ solve for $X(s)$

$X(s)$ is the impulse response in the s domain: $X(s) = \dfrac{1}{m \cdot s^2 + b \cdot s + k}$

We now have the Laplace transform of the impulse response for this system, $X(s)$, which, by definition, is the transfer function!

Let's dwell on something for just a bit. We know that when we multiply in the s domain it is the same as convolution in the time domain. Therefore, when we did the division to solve for $X(s)$ we were really doing a deconvolution operation in the time domain. This is the power of manipulating system models in the s domain. Every time you perform a simple multiplication or division with a transfer function just imagine the complicated mathematics that would have had to occur in the time domain to produce the same result. With that in mind, if our goal was to solve for the impulse response in the time domain, then using the s domain is still the simplest way. To go back to the time domain we could preform the inverse Laplace transform on $X(s)$, or again just look it up in a table if it exists.

We are going to keep the impulse response in the s domain, however, because this is the transfer function that we've been working to create. The transfer function represents the dynamics of the cart system and we can use this representation to predict how the cart will move when subjected to arbitrary forces. For now, though, let's just consider how the cart will move when subjected to an impulse function[21]. In block diagram form, we can set up the problem conceptually in this way.



We can 'apply' an impulse to our transfer function by representing the input in the s domain and multiplying it with the transfer function. From our Laplace table we know that the impulse function in the s domain is 1, and

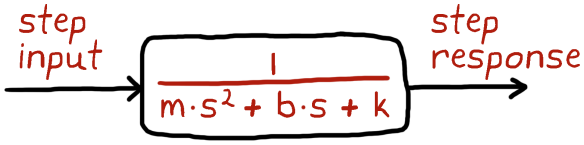
[21] I'm betting you've already figured out that it's just the impulse response.

therefore, the resulting output from our cart dynamics is 1 times the transfer function, or

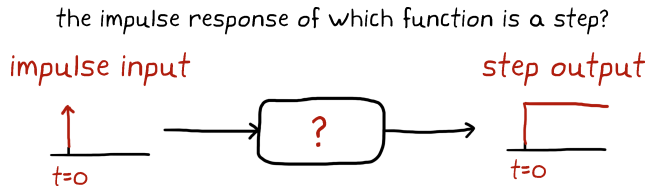
$$\frac{1}{m \cdot s^2 + b \cdot s + k}$$

There should be no surprise that the impulse response is equal to the transfer function because they are indeed one and the same!

Using this idea of representing an input in the s domain and multiplying it with the transfer function, we can easily apply different input forces to our system and see how the response changes. For example, what would the motion of the cart look like if the forcing function was a step rather than an impulse?



To solve this problem, we need to represent the step function in the s domain. We are basically looking for the transfer function that produces a step output, or a system whose impulse response is a step function.



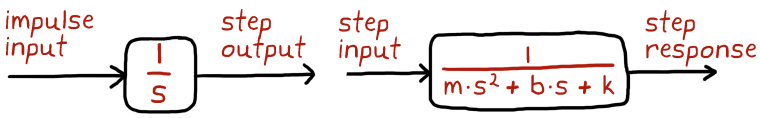
A step function can be written in equation form as 0 for $t < 0$ and 1 for $t \geq 0$. Once again, to get the transfer function we can solve the Laplace transform

of this step function equation or we can look up the result in a table. Either method you choose should result with the s domain equation,

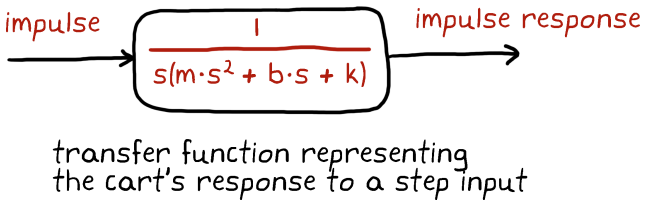
$$\frac{1}{s}$$

There's something kind of interesting about this relationship between impulse and step. Remember that even though the width of an impulse is infinitesimally thin, the area under the curve of an impulse function is 1. So if we integrate the impulse function with respect to time the result jumps to 1 at time zero and then stays 1 for the rest of time. In other words, integrating an impulse function produces a step function, and by extension, the transfer function $\frac{1}{s}$ is therefore the s domain representation of an integrator.

We can expand our block diagram to show the progression of the impulse input to the step response of the cart dynamics. The impulse input produces a step output through the $\frac{1}{s}$ transfer function, which is then applied to the cart dynamics to produce the step response of the cart system.



Finding the total transfer function for our system's response to the step input is as simple as multiplying the two transfer functions together. Notice that we've still retained the definition of a transfer function. That is, it is the s domain representation of the impulse response of a system. Impulse goes in and impulse response comes out.



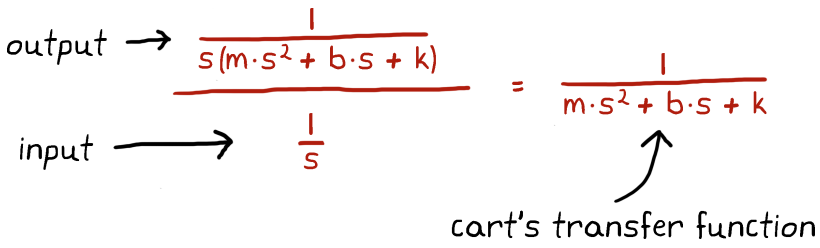
You may have heard that you can define a transfer function as the output of a system divided by its input in the s domain. It is true that this produces the same transfer function as the Laplace transform of the impulse response, however, it is misleading to think that it is the *definition* of a transfer function. This is because it hides what really makes a transfer function work; namely, that it's performing the convolution of the input and the impulse response. So I would encourage you to solve for transfer functions by dividing the output by the input[22], but while you're doing it remember the underlying mechanism that is really creating the transfer function for you.

Let's walk through an example that shows conceptually why dividing the output by the input works. Let's assume we have our cart system and we're trying to determine its transfer function. We decide to perform a test on the cart by applying a step function to it and measuring the resulting motion. We are then able to determine the s domain equation that produces a similar motion to be

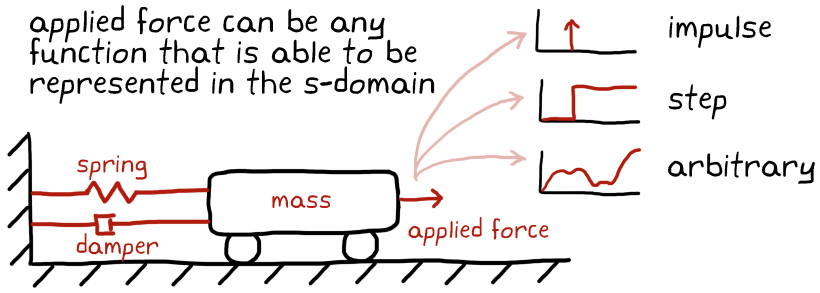
$$\frac{1}{s(m \cdot s^2 + b \cdot s + k)}$$

We know this is not the transfer function of the cart itself because it's not the impulse response; it's the step response. Therefore, we divide the output by the input to get the cart's transfer function.

[22] It is pretty easy after all!

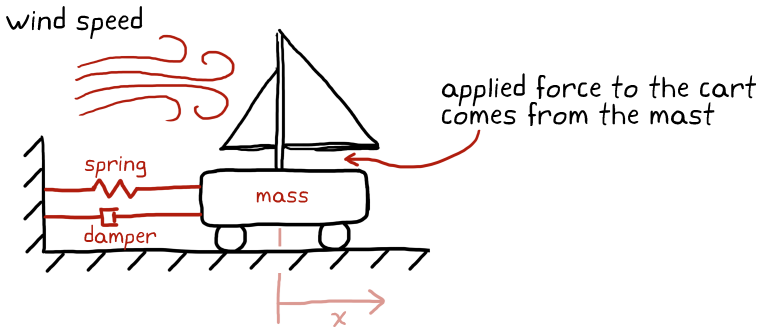


Using the cart's transfer function, and as long as that input can be represented in the s domain, we can apply any forcing functions we want to create a new model for the behavior of the system.

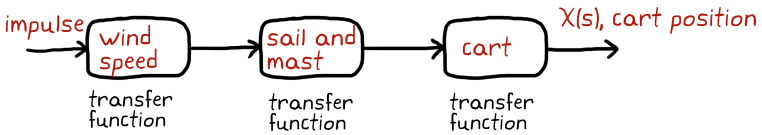


The real power with transfer functions goes beyond using them to apply different forcing functions to a system. It is that we can use them to represent very complex systems by building them out of smaller, less complex, systems.

For example, we can make our cart system a little more complex by saying that we don't apply the input force directly to the cart. Instead, it has a mast and sail that catches the wind and provides a resulting force.



The cart still has the same transfer function[23], but now the input force is driven by the output of the sail and mast system. The sail transfer function takes wind speed as an input and produces cart force as an output. Lastly, the wind speed can be an arbitrary function that would be represented by its own transfer function.

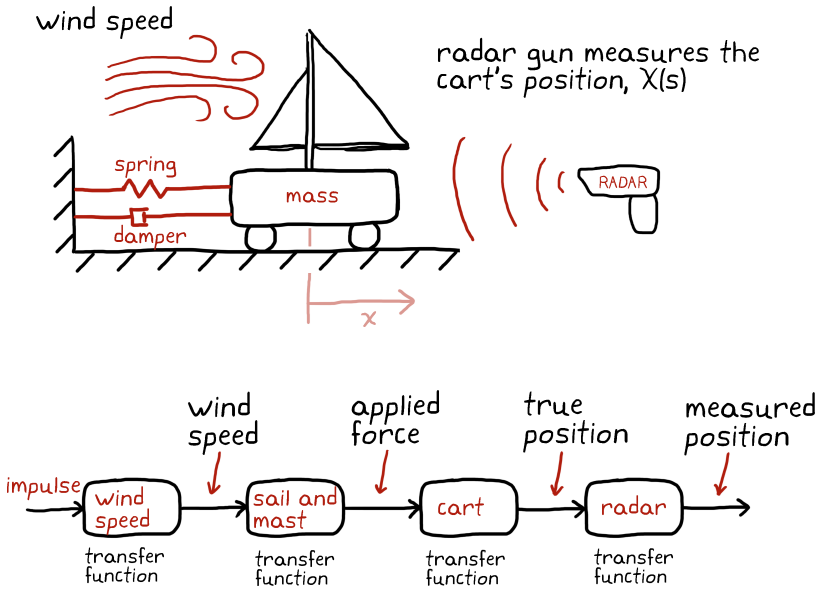


By combining these three together, you can predict the overall motion of the cart to the specified wind speed profile. Furthermore, one engineer no longer has to be responsible for deriving each of these systems. You could have a wind expert produce the wind profile transfer function, a sail expert model how the sail captures the wind and transfers the force, and a cart engineer to work out how the cart moves when a force is applied. The three engineers can come together afterward and produce the final model.

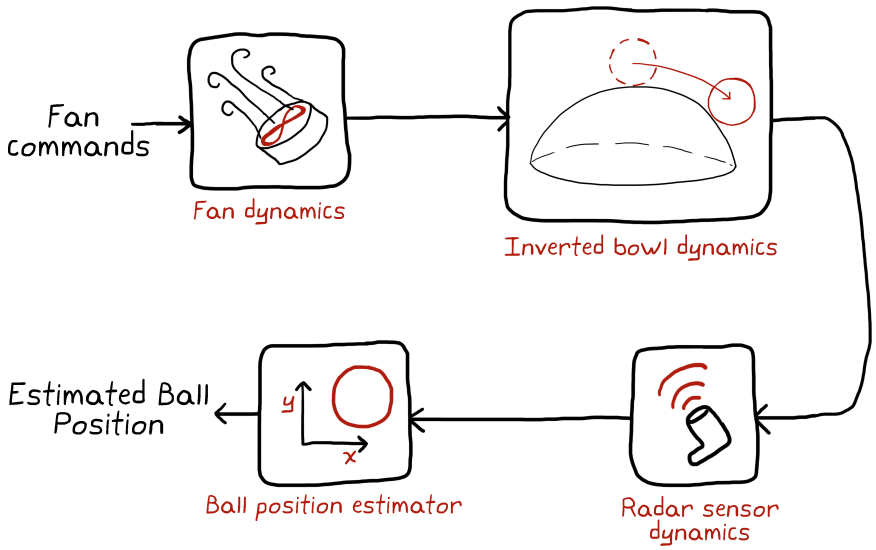
This way of building up models in a piecewise fashion also allows you to add new parts easily. You may find that you don't care about the true position

[23] Assuming the overall mass of the cart and sail stayed the same.

of the cart - because it can't be known exactly anyway - and that you want to know how well you can *measure* the position of the cart. You use a radar gun to measure the performance but it has its own dynamics, and therefore its own transfer function that needs to be accounted for.



Recall the inverted bowl problem that we set up at the beginning of this chapter. At the time it might not have been obvious how transfer functions would help us simplify this problem. Hopefully, now that you're at the end of the chapter that is no longer the case. The fan commands, inverted bowl dynamics, radar sensor dynamics, and ball position estimator would each have their own transfer function - possibly generated by a different engineer - and combined to form a model of the entire system.



With transfer functions, block diagrams are so much more than just abstract ways of visualizing and organizing your system - you can actually manipulate them mathematically. This will be hugely beneficial when we start describing systems that have feedback and other complex dynamics. We can use the fact that we can multiply and divide transfer functions to simplify the feedback loops in systems, and by simplifying them we can start to understand how they affect the larger system.

Lest you think this chapter was a pretty weak argument for why transfer functions are a good way to represent system models, just know that transfer functions have other great qualities as well. We haven't yet talked about the power of system analysis and manipulation that transfer functions give us. So just consider that a bonus thing to look forward to! For now, though, let's take what we know of transfer functions and move on to the next chapter to discuss block diagrams in further detail.

2.8 Try This!

1. Which of these system models are LTI? For the models that are not LTI prove which LTI property is not met.

a)

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

b)

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

c)

$$x(t) = 5\frac{d^2x(t)}{dt^2} + 3\frac{dx(t)}{dt} + 2x(t)$$

d)

$$x^2(t) = \frac{dx(t)}{dt}$$

e)

$$0 = \dot{x}(t)\ddot{x}(t)$$

2. Look around and describe the systems near you. What are the inputs and outputs to those systems? How could you divide them up into smaller subsystems that are interconnected? Describe whether you think it could modeled as an LTI system.

Can't think of any ideas? Here are a few examples to get you started. Flushing a toilet[a]. Changing the channel on your TV. Balancing on two legs of your chair as you lean back.

3. You are asked to generate the transfer function for the pitch control of an airplane. The input is the control column - pulling back causes the airplane to pitch up and pushing forward causes the airplane to dive down. The output is the measured pitch angle. The pilot does not want to produce an impulse input (basically pulling back and releasing as fast as he can) because he feels it is dangerous to the airplane. Instead, he pulls back on the control column quickly and then holds it in that position for 10 seconds. You are given the measured pitch angle from that maneuver in the s domain.

$$P(s) = \frac{s+0.2}{s^4+0.7s^3+s^2}$$

What is the transfer function for the pitch control of the airplane?

What would the measured pitch angle, $P(s)$, look like had the pilot pulled back on the control column gradually (ramp input) rather than the sharp pull (step input) that he did?

4. Prove that convolution in the time domain is multiplication in the s domain. **Hint**, we proved this already for the frequency domain.

5. Draw the approximate time domain graph for e^{st} at each of the following s locations.

a) $s = 0$

b) $s = 3$

c) $s = -1$

d) $s = j$

e) $s = 3 + j$

f) $s = -1 - j$

[a] Are you studying in the bathroom??

Chapter Credits

Meg Douglas	Kirkland, Washington	Gavin Kane	Emden, Germany
David Feinauer	Northfield, VT	Krunal Desai	
Wong, C.J.	Zootopia	Jeff Sprenger	USA
Tran Dam		Shuki Eisdorfer	Isreal
Hyungsouk Kang	Seoul, Korea		

Thanks for making this chapter awesome!

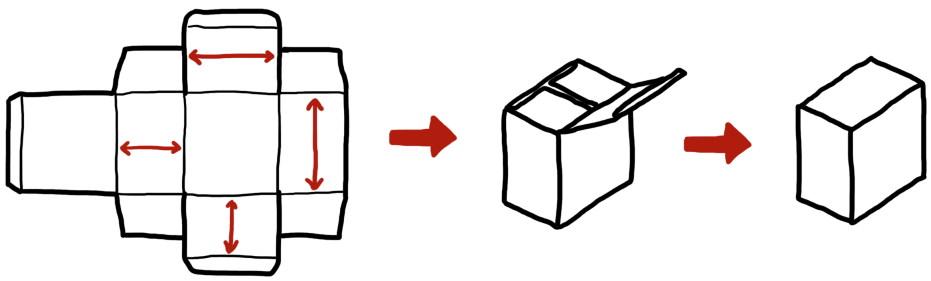
3 Block Diagrams

In this chapter, we're going to discuss one of the control system engineer's favorite tools for organizing, communicating, simulating and solving problems: the block diagram[1].

Block diagrams are actually pretty simple. You draw some boxes and connect them with arrows to represent your system, and voila, you have a block diagram, right? There is, of course, more to it than that. The usefulness of block diagrams goes well beyond just a way to represent your system graphically, and to understand why this is the case, this chapter will cover a few topics that won't necessarily help you complete a homework assignment, but will help you gain a better understanding of what makes block diagrams so powerful in our quest to be great control system engineers.

The first section covers how block diagrams fit into your toolbox of control system knowledge and why we learn them in the first place. We'll then cover some terms and their definitions so that we're all speaking the same language. We'll walk through some of the more common block diagram al-

[1] Don't confuse a block diagram with a diagram for making blocks!



gebra rules and then we'll present a few examples of how to use the algebra rules to simplify a diagram. Finally, we'll cover a few ways you'll use block diagrams in your everyday working life.

3.1 Why are block diagrams important?

Block diagrams show us the interrelationship of systems and how signals flow between them.

A block diagram is just one type of a huge variety of diagrams that cover all fields of engineering. A diagram is a graphical representation that gives us a way to abstract away the complexities of an idea so that we are left with a clearer, more focused, understanding. Another way to say this is that diagrams are simplified drawings that remove all of the stuff that is not needed so that the user can focus on the concept being described. With block diagrams, we focus on how systems are connected and the signals that flow between them. We omit the other engineering information that would just clutter up the diagram. For example, block diagrams don't show the physical location and drawings of the hardware.

A list of all engineering diagrams is too numerous to write out - and probably not that helpful in this chapter, but the following example gives you a sense of the breadth and usefulness of diagrams during an engineering project.

Over the course of the project, you'll have a schedule that is written out as a *sequence diagram* showing the interrelationship of activities and their allotted times to complete. When you're designing a part you might diagram out the electrical system with *schematics* and *routing layouts*. The software and hardware interaction might be planned using a *functional di-*

agram which shows how components work with each other. There might be *logic flowcharts* that show how the system is designed to reason through logical inputs and handle decisions. There are *process diagrams* that define the order in which activities and decisions are made. There might be a *system topology diagram* that shows how all of the hardware components of a network are connected. During the test and verification phase of a project, you might come across an error that requires a *cause and effect diagram* like the Ishikawa (or fishbone) diagram. And when you're confused about who to go to to ask your questions about all of these different types of diagrams you'll probably consult the *organizational chart*, which shows the hierarchy and responsibilities of the team members of a project.

When you first learn about block diagrams, you typically use them to sketch complex architectures and then use that sketch to simplify the system to get an overall transfer function[2]. This chapter is no exception; we will do that very thing in the next few sections. However, it is important to keep in mind while you read this chapter that in industry you will more than likely use block diagrams in a different way than what is presented in this book. You will probably very rarely have to determine an overall system transfer function from a block diagram, but don't let that discourage you from really understanding this material. Learning about block diagrams in this way is worthwhile because they give you a foundation from which you can build your understanding of more complex control theory topics. Without a good understanding of block diagrams, it will be very difficult to visualize and intuitively understand the concepts we'll cover in the rest of this book. This visual representation is powerful because we can use it to reorganize and simplify our systems and open up the possibility for collaboration between

[2]Later in this book we'll use this overall transfer function to make claims about the performance of our system and to design a control system

teams of people, oftentimes, who don't interact with each other, other than through these diagrams.

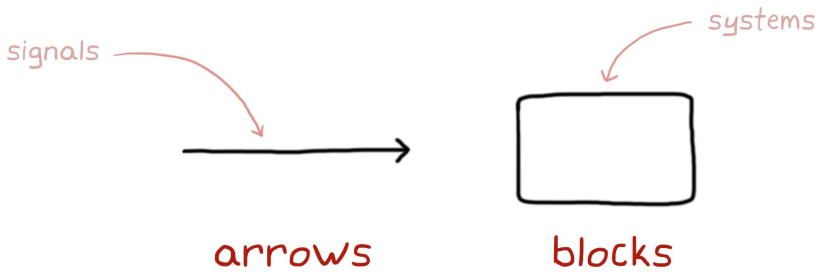
Arguably, the most important reason to learn block diagrams is that they have reach well beyond control theory. Treating block diagrams as a tool to sketch architectures and generate a transfer function is the beginning of understanding their usefulness in other engineering areas. Possibly, the most important benefit of learning block diagrams is to introduce you to the way dynamical systems are described in model-based design - which is used extensively in the automotive and aerospace industries. I will present a very short introduction to model-based design at the end of this chapter.

3.2 The nomenclature (let's all speak the same language)

We will use block diagrams throughout this book, therefore, it's important that we have a consistent nomenclature, or definition of the terms, so that we minimize any confusion there might be going forward. Keep in mind that I am presenting to you the nomenclature that I prefer to use. Other authors, lecturers, and coworkers may choose a different set of terms that mean essentially the same thing. In some cases, I will list out a few alternative names or styles, but in general, keep in mind that throughout your career you may come across slightly different ways of representing and talking about block diagrams. It is fine to have a different definition as long as you and the person you are communicating with have the *same* definition - so with that, let's make sure we have the same definition.

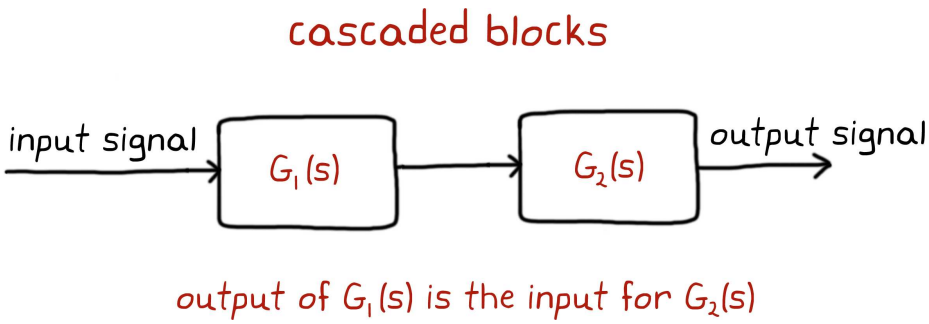
3.2.1 Arrows and blocks

The first thing you typically notice when looking at block diagrams is that they are drawn as a set of blocks that are connected in some way by a set of arrows.

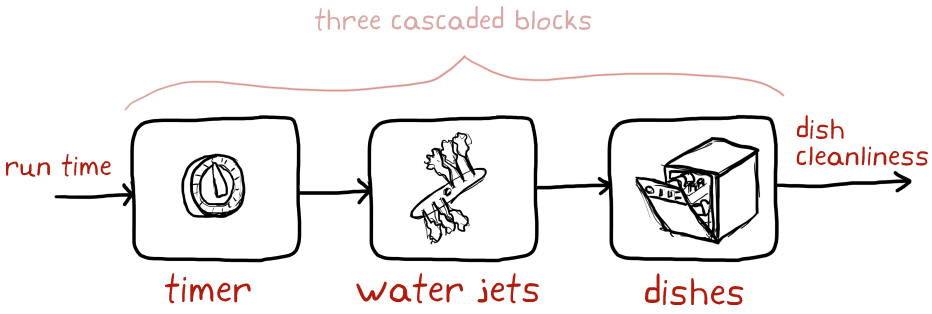


Each block represents a system and each arrow represents a signal that goes to or from that system. In this way, you can depict graphically how individual and modular systems interact with each other. In other words, you can see how signals are generated by and shared with systems.

In the example below, the output signal of the first block, $G_1(s)$, is the input signal of the second block, $G_2(s)$. This series of blocks is sometimes called *cascaded blocks* because of the way the signals progress from one to the next; much like how water cascades down from one rock to the next in a small waterfall.



From here, we could continue to add more blocks to this diagram in a cascaded manner to model more complex systems - just like we did with the three blocks in the diagram in chapter 1 where we described the open loop control system of a dishwasher.



The timer, water jets, and dishes are all modular systems depicted by single blocks. The run time is the input signal to the timer system and the dish cleanliness is the output signal from the dish system. The two arrows in the middle also represent some signal but are not labeled in this diagram.

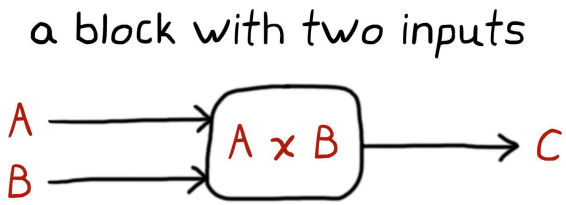
Blocks and arrows alone can create a cascaded system, however, they aren't enough to fully represent the complexities of a feedback control system. For that, we need to add summing junctions and take off points to our repertoire.

Multiple signals into and out of a system

When dealing with block diagrams where the systems are modeled as transfer functions, you will only ever see one arrow into and one arrow out of a block. This is because transfer functions are Single-Input, Single-

Output (SISO) functions so it doesn't make sense to have a block diagram with multiple inputs or multiple outputs[a].

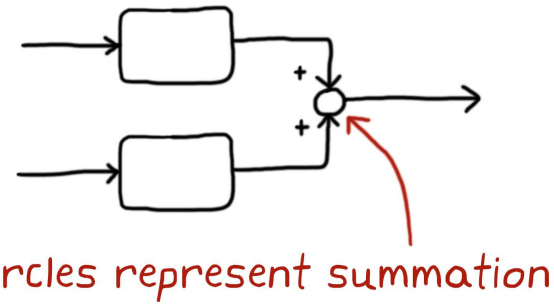
In general, though, blocks can have multiple arrows going into it or coming out of it. A block that has two arrows going into it and a single arrow coming from it means this system takes two input variables, performs some mathematical operations on them, and then generates a single output variable. For example, a system might take the two inputs and simply multiply them together to generate the single output.



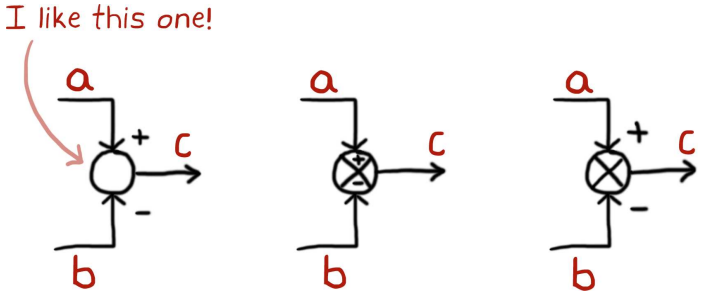
[a] State space representation does allow for multi-input, multi-output system, we're only talking about systems represented as transfer functions here.

3.2.2 Summing junctions and take off points

You can subtract or add signals using a summing junction. A summing junction is a circle with two or more input signals and a single output signal. For each of the input signals, there is a $+$ or $-$ sign indicating whether that signal is added or subtracted.

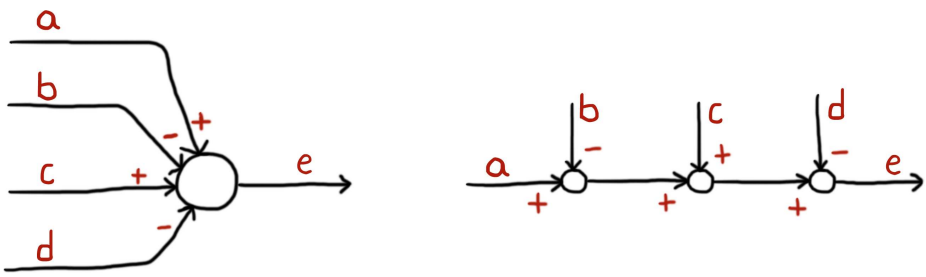


There are several different stylistic ways you can draw a summing junction; I show three different ways to draw $a - b = c$ below. I prefer not drawing the X in the circle because I think it has a cleaner look, and therefore to me, it's easier to read.

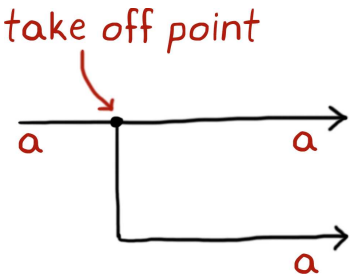


One purpose of a block diagram is to help you organize your problem so that it's easier to understand and share with others. In this case, communication is done through the clarity and organization of the diagram. Therefore, special care should be taken to make sure the diagram is clear and easy to read. You can create a diagram that is technically correct, but if it's not presented in a straightforward way it can cause confusion. For example, another stylistic choice you have with summing junctions is how to handle multiple input signals. Both of the following diagrams are technically correct, and both represent $a - b + c - d = e$, however, I find the diagram on

the right, splitting out each summation into its own junction, is easier to understand. Choose the method that you think looks the best for the diagram you're creating. If there had only been three inputs into the summing junction I may have preferred the style on the left.

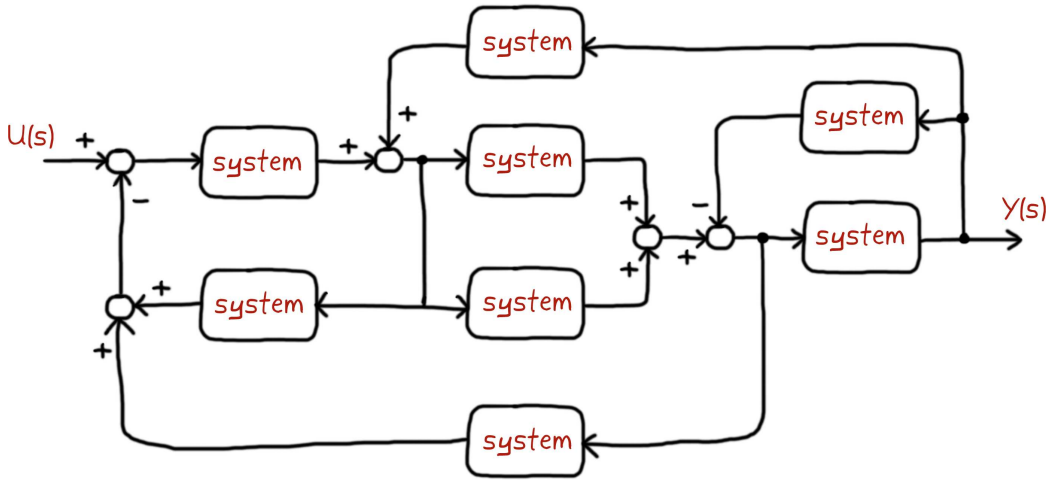


A take off point occurs when one arrow starts at another arrow. They are used to allow an unaltered signal to go along multiple paths. In the drawing below, the signal a is split through the take off point and is fed to both output arrows.



Take off points are used when the same signal is the input for multiple systems or summing junctions.

Blocks, signals, summing junctions, and take off points, are not very complex concepts, but you can build very complex block diagrams with them. The following diagram might look like a tangled mess, but you can see that it is created from the same four symbols that we've covered.

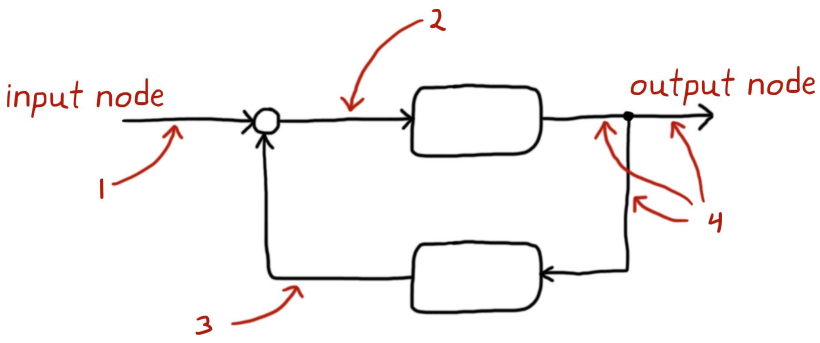


Even though the diagram above consists of only 4 different symbols, there is additional complexity in the patterns that are created. It is helpful to give names to and define some of the patterns that come up often in block diagrams.

3.2.3 Nodes

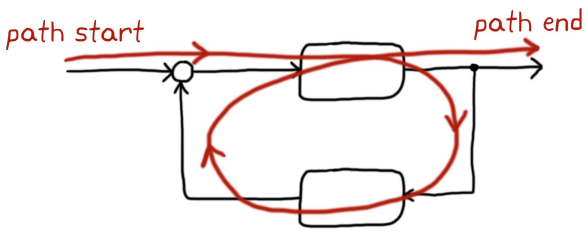
Nodes, in our context, are typically used in signal-flow graphs, a close relative to a block diagram. I will explain them here because they will help us define the patterns that we're interested in. Arrows are not the same as nodes. Arrows represent signals and the direction they flow, whereas nodes are the system variables and can consist of multiple arrows. A signal is constant at a given node regardless of how many arrows make it up. Summing junctions and blocks create new nodes (because they change the signal) but take off points do not. The following diagram consists of two blocks and one summing junction. The combination of those three elements creates four nodes. Notice that the output node 4 extends through the take off point

and consists of both arrows. The nodes 1, 2, 3, and 4 are each unique signals in this diagram.



3.2.4 Paths

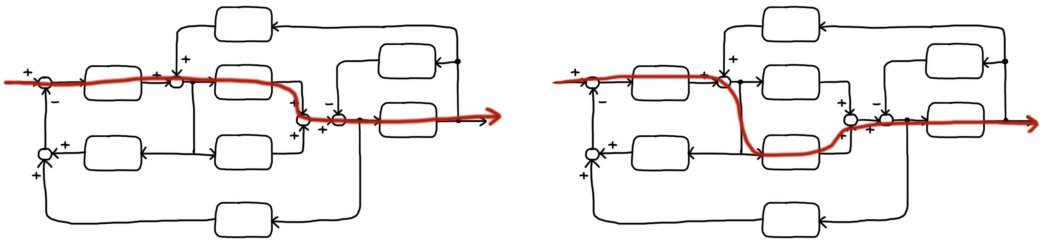
A path is a continuous line that is created if you place your pen on a node and trace the signal lines in the direction of the arrows. Start on any node, take any direction at a take off point, and stop on any node of your choice. You've just traced out a path.



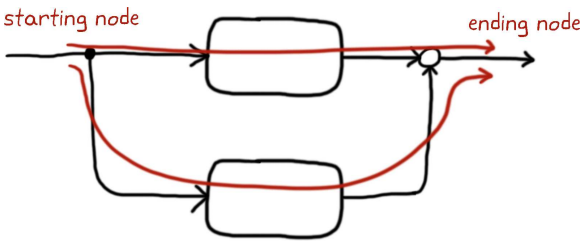
Let's walk through some paths that have specific names; these will come up often over the course of this book.

The forward path is any path that starts at the input node and ends at the output node without ever touching the same node twice. You can have more

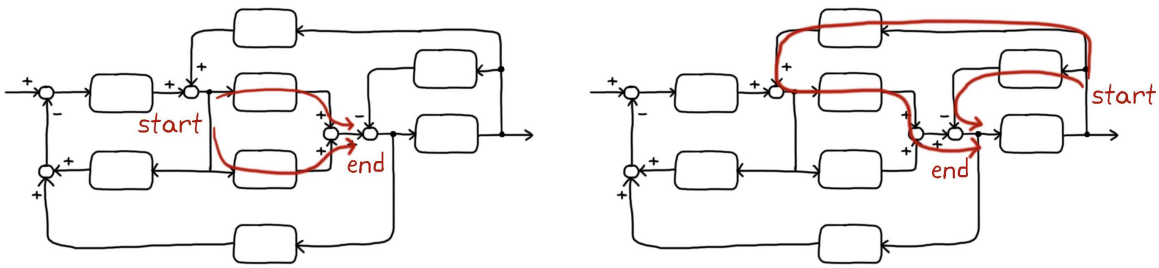
than one forward path in a block diagram. The following example shows a diagram with two forward paths.



Two paths are considered parallel if they both start and end at the same node while not sharing any of the same blocks.

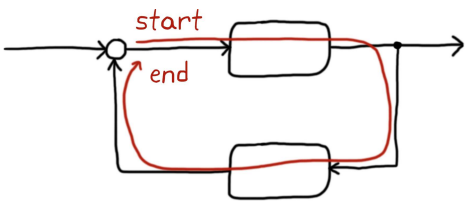


Parallel paths can be in the forward direction, like the above diagram, but they can also be in the reverse direction as well. You can have parallel feedback paths, and any number of complicated parallel paths, as long as they meet the definition of what it means to be parallel. The following example shows different parallel paths that exist in our complex block diagram.

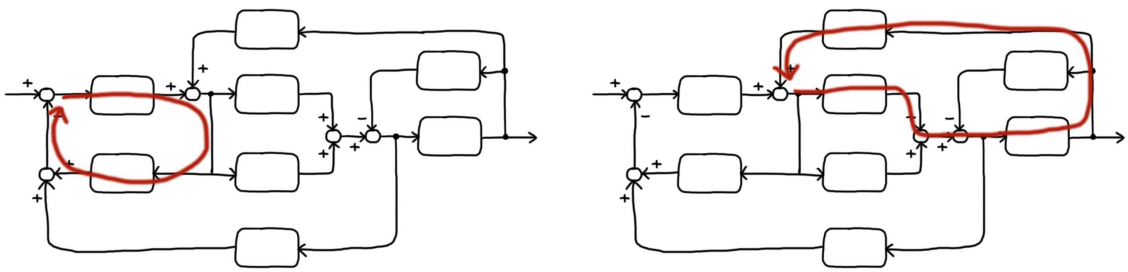


3.2.5 Loops

A loop is any path that starts and ends at the same node without ever touching any node more than once. The classic feedback system is often called a closed loop system and you can see how it clearly meets the definition of a loop.

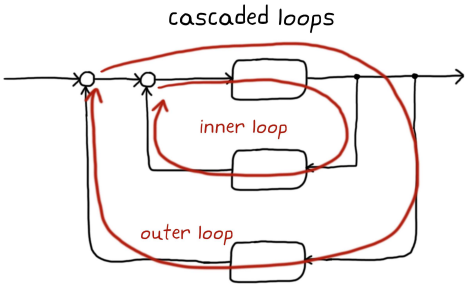


Loops aren't always obvious. Instead of a nice clean loop that takes up pretty much the entire diagram, they can be parts of a larger, more complex, diagram. The following example shows just two of the many loops that exist in our block diagram.

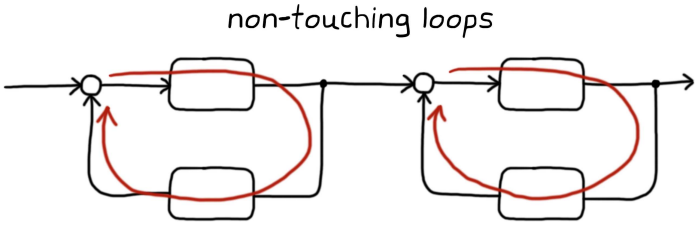


When you have more than one loop in a diagram they have special names based on how they interact with each other. When one loop is nested within another, this is called cascaded loops. The two loops are distinguished as the inner loop and outer loop. You'll notice that this pattern is made up of two feedback paths that are parallel to each other. In this way, you can start

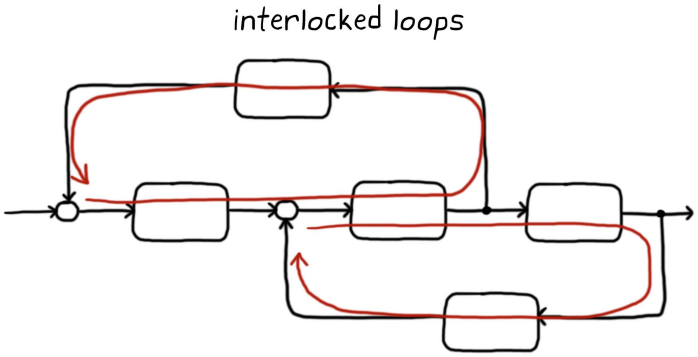
to see how we use the terms we're learning to describe the types of block diagram structures that systems can have.



The two loops could also be separate from each other; this is called non-touching loops. Non-touching loops occur when the two loops don't share any blocks or nodes.



Lastly, the two loops may not be fully nested but still share some of the same nodes and blocks. These are called overlapping or interlocked loops.



I think we should be able to get through the majority of the block diagrams we'll encounter in this book with just the terms I've listed. The thing to remember is that block diagrams are created using just four symbols; arrows, blocks, take off points, and summing junctions. The rest of the terms we covered, and most of the complexities of block diagrams, come from the patterns that emerge from those four symbols.

3.3 Block diagram algebra

Now that we have a common set of terms, let's walk through some of the algebraic rules that you'll employ when manipulating and simplifying block diagrams. However, before I start just listing out algebra rules, let's take a minute and explain why algebra even works in the first place. If you take nothing else away from this section, remember that the algebra rules we will cover here are only possible if the system is LTI and that the systems are represented by transfer functions - that is, in the *s* domain.

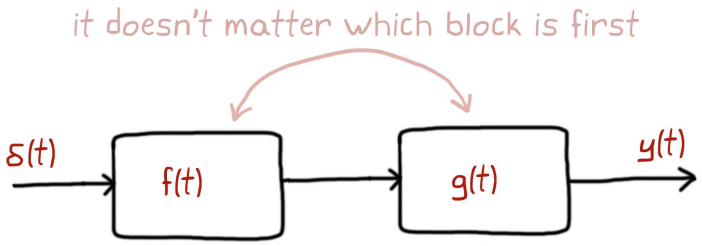
3.3.1 Why is LTI necessary?

Remember that LTI systems obey the properties of homogeneity, superposition, and time invariance. These properties are what allow us to move blocks around and simplify diagrams. If the system is LTI, then manipulating the blocks relative to each is very simple.

Take, for example, two cascaded LTI blocks. Since LTI systems are commutative, a byproduct of homogeneity, superposition, and time invariance, we can manipulate them by simply swapping their order without impacting

the output of the system. Don't believe that statement? Proving it is pretty straight forward.

In the following block diagram, we have two LTI blocks, $f(t)$ and $g(t)$.



It was shown in the last chapter $f(t)$ represents the impulse response of that block. We can 'play' the output from $f(t)$ through $g(t)$ with the convolution integral and get the output of the cascaded system, $y(t)$.

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

If we swap the order of the two blocks, finding the new value for $y(t)$ is as simple as swapping the variables in the convolution integral.

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

However, we're claiming that swapping the order doesn't matter, so $y(t)$ should be the same in both cases.

show that these are equal

$$\int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau = \int_{-\infty}^{\infty} g(\tau)f(t-\tau)d\tau$$

The simplest way to prove this is by changing the variable of integration on the right side of the equation from τ to u; where $u = t - \tau$. We can differentiate it to get du and also find the new limits of integration.

define new variable

$$u = t - \tau$$
$$du = -d\tau$$

adjust limits of integration

$$u(-\infty) = t - (-\infty) = \infty$$
$$u(\infty) = t - (\infty) = -\infty$$

Plugging all of this into the right side of our equation from above produces the following result.

$$\int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau = -\int_{\infty}^{-\infty} f(u)g(t-u)du$$

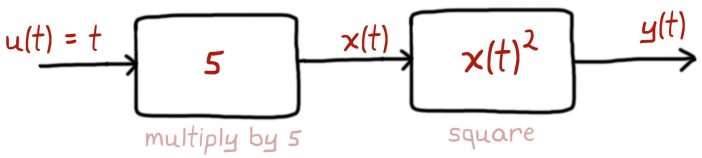
flipping the integration limits will cancel out negative sign

We have a negative sign in front of the right integral but the limits are swapped as well; they go from ∞ to $-\infty$. Luckily, swapping the integration limits is the negative of the integral so it all works out.We have shown that convolution is commutative and, therefore, swapping the order of the blocks has no impact on the output $y(t)$.

$$(f * g)(t) = (g * f)(t)$$

It is worth noting, however, that even though the output is the same in both cases, it should be obvious that the intermediate signal - the signal between the two blocks - does change when you swap the order. This is a common result we'll see as we manipulate block diagrams. The overall system is unchanged, but the intermediate signals do change, and frequently lose their real physical meaning.

We can see that even simple block diagram manipulation does not hold for a nonlinear system. It's harder to prove this mathematically with just a few algebraic steps, but fortunately, we can choose an arbitrary example and prove that in at least one case non-LTI systems do not commute. In the following example, we set $f(t)$ to a gain multiplier of 5, the input $u(t)$ to a ramp, and $g(t)$ to a squaring function; a decidedly nonlinear operation.



I'll let you step through the math[3], but I think you'll be able to find that in the original block order you're left with $y(t) = (5t)^2$ and after swapping the order of the blocks you're left with $y(t) = 5(t^2)$. Not the same result.

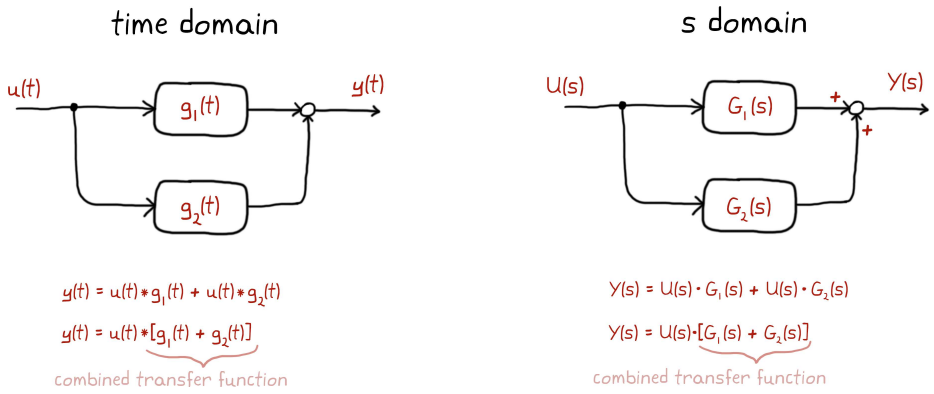
3.3.2 Why are transfer functions necessary?

We can see from the previous section why LTI systems are necessary, but why do we create block diagrams in the s domain with transfer functions rather than in the time domain? We learned the answer to this in the chapter

[3] Don't you hate when authors do this?!

on transfer functions and it is because when we're working in the s domain the mathematics become very easy. In the s domain, complex time domain operations like convolution, differentiation, integration, and time-shifting are accomplished with the four basic mathematical operators; $+, -, \times$, and $/$.

For example, let's compare two forward parallel paths in the time domain with the same system in the s domain. If our goal is to simply write out the combined transfer function of our system, then either the time domain or the s domain are equally easy to work in. In both, the combined transfer function is the summation of the two individual system transfer functions.

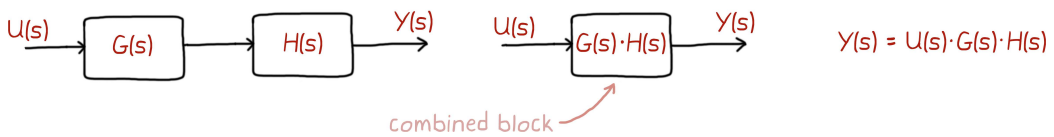


But as you can see, if our goal is to actually determine the response of this system given an arbitrary input, $u(t)$, then they are no longer equally simple. In the time domain, you need to solve the convolution integral twice and sum the results - this is a lot of math. In the s domain, the output can be calculated by summing the two transfer functions and multiplying them with the input function - a much simpler operation.

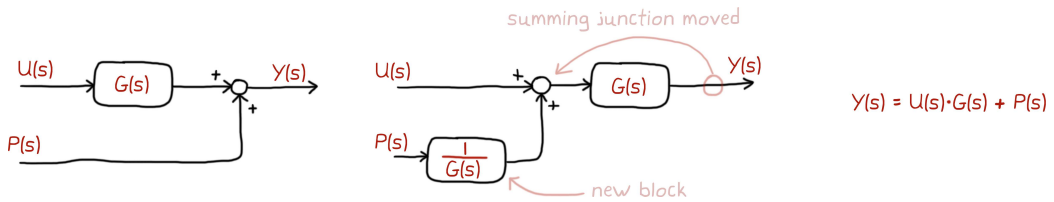
3.3.3 Common algebraic rules

The benefit of building block diagrams in the s domain with transfer functions is really evident when you start working with and manipulating more complex diagrams. Once you have those complex diagrams, your next goal will be to start simplifying them into something that is workable. To do this, we need to learn a few common algebraic rules for block diagram manipulation.

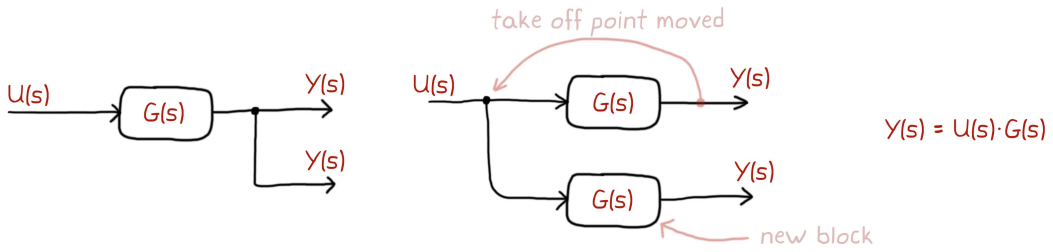
Combining two blocks in series - multiply the two transfer functions together. The output, $y(t)$, of the system is the input, $u(t)$, times the product of the two systems.



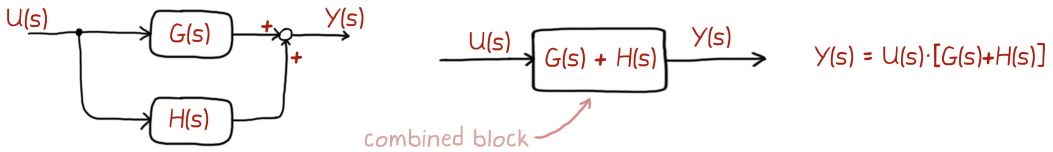
Moving a summing junction - whether you are moving the summing junction before or after a block, make sure the resulting algebraic equation, from inputs to outputs, is the exact same. Here, I moved the summing junction to before the system $G(s)$. This has the result of also multiplying the input path, $P(s)$ by $G(s)$ as well. Therefore, we need to add another block on the $P(s)$ path to divide out the extra $G(s)$.



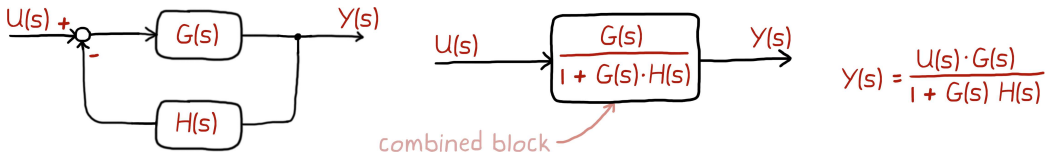
Moving a take off point - This is similar to moving a summing junction in that you need to account for the extra or missing multipliers that arise from the move. In this case, we moved the take off point to before the system, $G(s)$, and so that needs to be accounted for with an extra block.



Removing a forward parallel path - sum the two systems together. Of course, you do need to adhere to the signs in the summing junction. So in some cases, you will be summing a negative path which will result in subtracting the two systems.

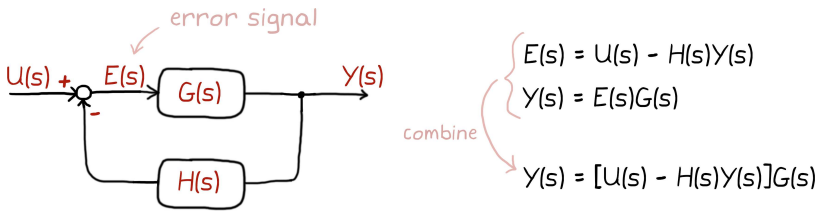


Removing a feedback path - replace the negative feedback structure with the equivalent transfer function, $\frac{G(s)}{1+G(s)H(s)}$.



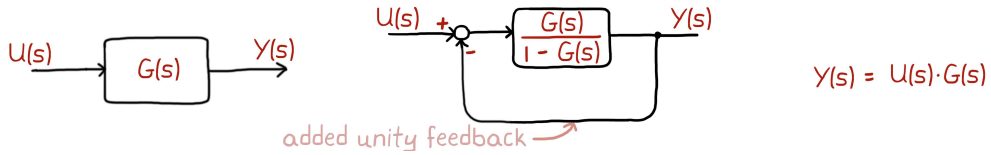
You will use this transformation a lot so it's worth memorizing, but in case you have a bad memory, it's really easy to solve for it. I start by labeling the

error signal after the summing junction and then setting up the two equations that will be combined into a single algebraic equation.



From here, you can solve for $\frac{Y(s)}{U(s)}$ and you will get a single combined transfer function for the negative feedback system.

Adding a feedback path - what's interesting about setting up the algebraic equations like we did in the negative feedback path example is that we can apply that exact same logic to a number of different situations. Here, we convert a transfer function, $G(s)$, into a negative feedback system with unity feedback gain.



We have to be careful here because once you start manipulating block diagrams you are deviating from real physical systems. For example, we just created a block diagram with unity feedback, however, we can't guarantee that the system truly has unity gain feedback, only that it has the transfer function $G(s)$. That means if we want to add a sensor to our system to measure the feedback signal, there might not actually be a real signal that we can measure. That signal is a construct of the block diagram manipulation we just did.

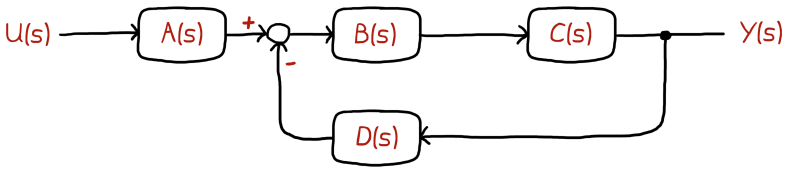
With that in mind, block diagram manipulation is still a worthwhile exercise because we can use these algebraic rules to simplify our diagrams and gain a better understanding of the system as a whole.

3.4 Simplifying block diagrams

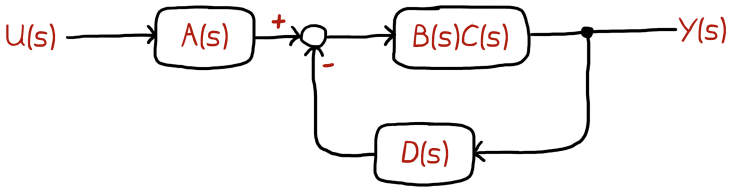
Let's apply the block diagram algebra rules to a few different block diagrams.

Finding the transfer function of a block diagram

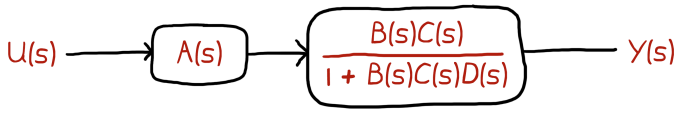
In this example, we have a classic negative feedback block diagram and we want to find the transfer function for this system.



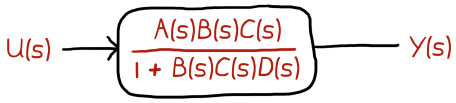
Systems B and C are in series (or cascaded) and so we can combine them into a single block by multiplying them together.



Now we remove the negative feedback path.

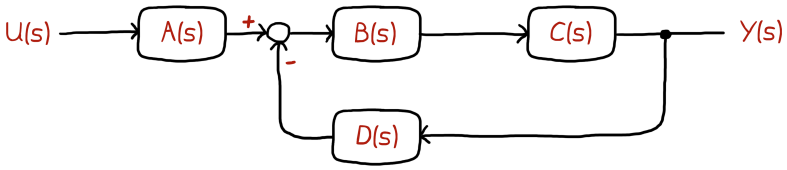


Lastly, we combine the two blocks in series and we are left with the transfer function for this system.

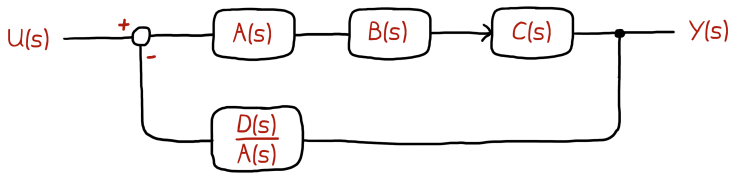


Another approach to the previous example

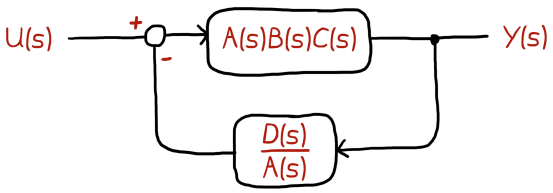
The approach taken in the last example is just one way to simplify the block diagram. But what we'll show now is that it doesn't matter what step you do first, you will always end up with the same transfer function at the end[a].



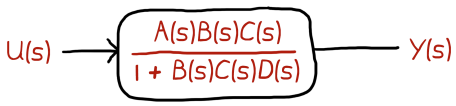
Instead of combining systems B and C first, we'll move the summing junction to the left side of system A.



Now we can combine the three cascaded systems in the forward path into a single block.



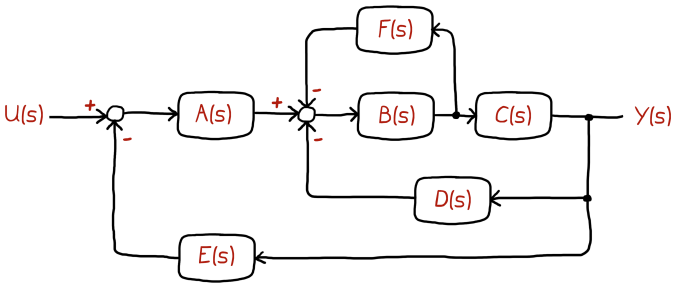
Once we remove the negative feedback path we find that we did, indeed, get the same transfer function for this system. This is important to realize because you may find yourself getting stuck on which simplifying step to do first. Remember that you can do any step first you want and as long as the algebraic steps are followed you'll end up with the same answer at the end.



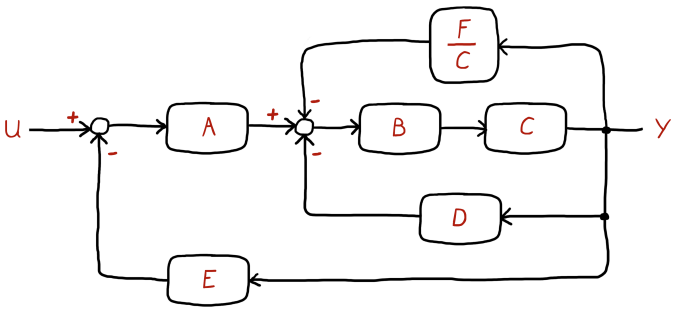
[a] Assuming you don't make any mistakes along the way!

Dealing with interlocked loops

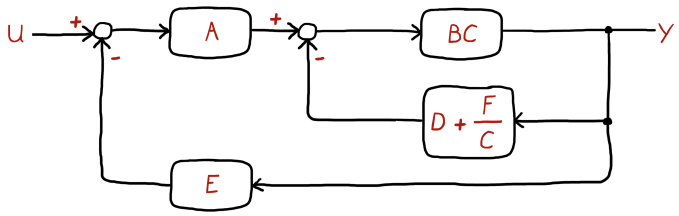
Let's try our hand at simplifying a system that is more complex than the previous negative feedback example. In this problem, we have a system is made up of nested and interlocked negative feedback paths.



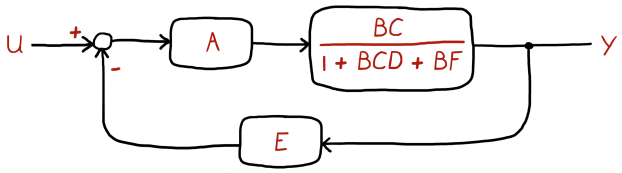
From this point, I've dropped the (s) to simplify the drawing and make the transfer function easier to read. The first step I took is to move the take off point to the node after the system $C(s)$.



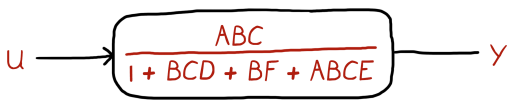
The feedback path with $\frac{F}{C}$ is parallel to the feedback path with D, therefore, we can combine them by summing them together.



We now have two nested feedback loops. The next step is to remove the inner loop feedback path.

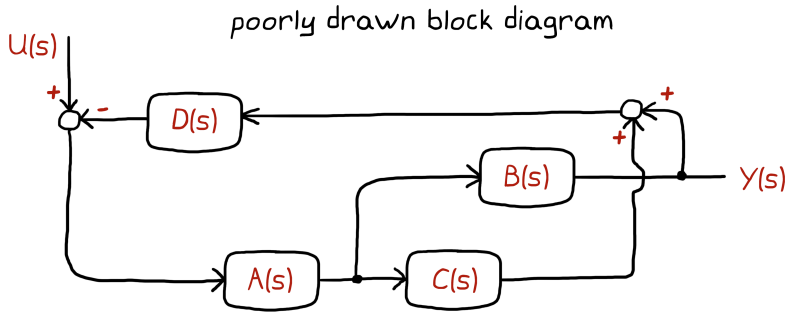


Lastly, we can combine the two systems in series in the forward path and remove the last negative feedback loop to give us the transfer function for this system.

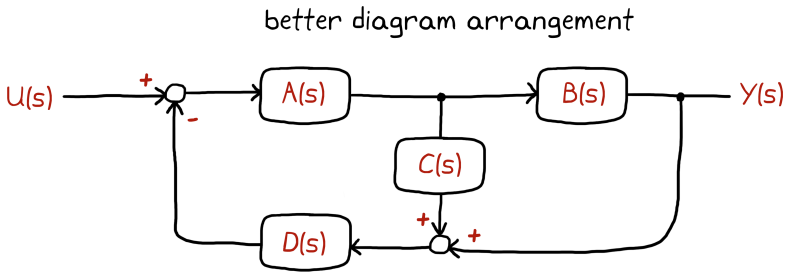


Poorly drawn block diagrams

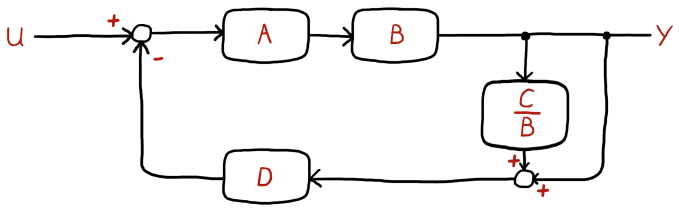
It is possible to draw a block diagram that is technically correct but drawn in a confusing way. Take the following block diagram as an example. When you first look at this diagram it might not seem too poorly drawn; all of the lines are straight and everything is well-labelled. However, the way I've drawn it makes it really hard to quickly answer questions like 'is there a feedback path and is there more than one? What is the forward path? Where do I start in order to simplify this block diagram?'.



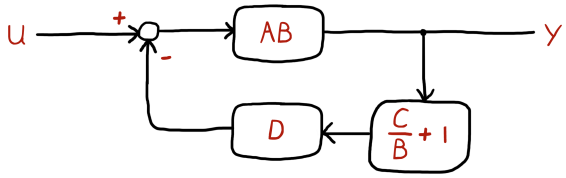
By simply rearranging the blocks and arrows - without any other simplifications or combinations - this diagram will be much easier to read. This might not seem like a big deal but one of the uses of block diagrams in industry is to communicate a design to other engineers. A poorly laid out diagram will hide the intent of the designer and will make it harder for a someone to peer review the design effectively.



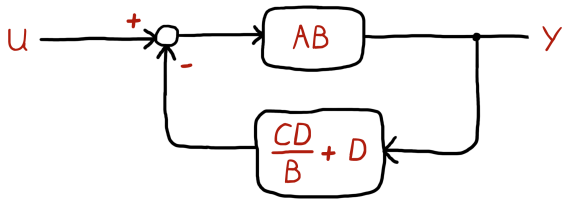
From here, we can continue to simplify the diagram using the block diagram algebra rules. Once again, I've dropped the (s) after each system letter to make the diagram easier to read. To start, we can move the take off point to after system B. This leaves us with parallel paths feeding back into the summing junction.



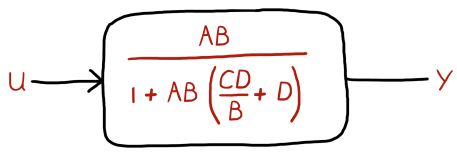
We can combine the parallel paths by summing them together. By recognizing that the unity feedback path has a gain of 1, the new feedback system is $\frac{C}{B}+1$.



Now, we can combine the two cascaded blocks in the feedback path. At this point, our diagram is drawn in the classic feedback structure. The forward path is AB and the feedback path is $\frac{CD}{B} + D$.



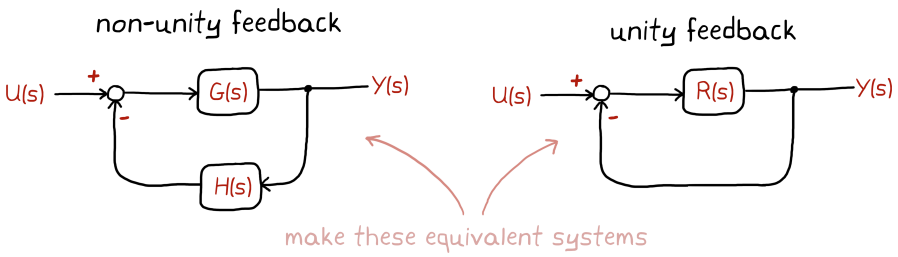
We can simplify further by removing the feedback path.



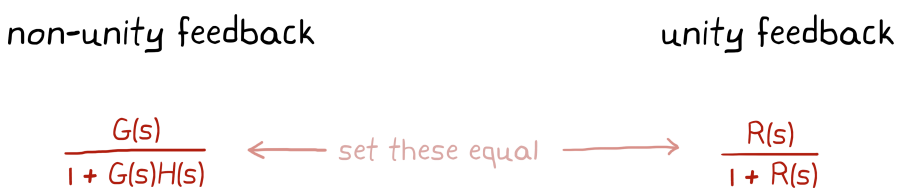
Finally, we can distribute the AB in the denominator to produce a very pleasing form of the transfer function.

Creating unity feedback

When you manipulate block diagrams, your goal isn't always to remove blocks to simplify the drawing, sometimes it's to put the diagram into a specific structure. For example, you may want to describe your system with unity feedback - that is with a 1 in the feedback path. One of the nice things about LTI systems adhering to algebraic rules is that, in general, we can fit a system to any structure we want as long as the resulting algebraic equations have a solution. Let me explain what I mean by this by walking through the conversion of a non-unity feedback system into a unity feedback system.



We need to determine an $R(s)$ value that makes the two systems equivalent. We can do that by writing out the transfer function for each system, setting the two equations equal to each other, and then solving for $R(s)$.



We can solve for $R(s)$ in this case because there is one equation and one unknown variable. Notice, however, that we wouldn't be able to easily use this method to go from a unity feedback system to a non-unity feedback system. In that case, there would be two unknown variables, $G(s)$ and $H(s)$. In order to solve that, we'd have to assume the solution of one of the systems and solve for the other.

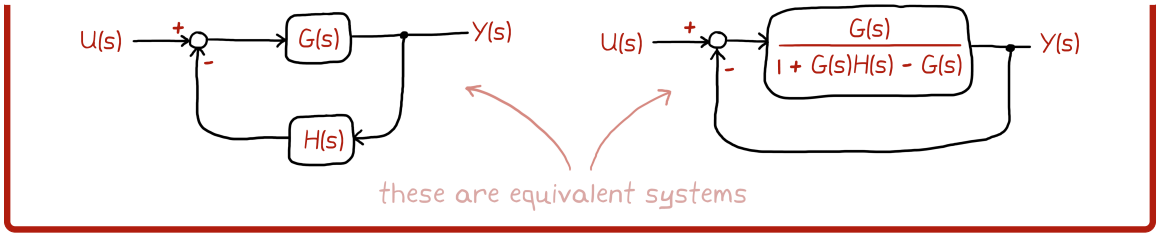
(1) $$\frac{G(s)}{1 + G(s)H(s)} = \frac{R(s)}{1 + R(s)}$$

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

(3) $$\frac{G(s)}{1 + G(s)H(s)} = R(s) \cdot \left(1 - \frac{G(s)}{1 + G(s)H(s)}\right)$$

(4) $$R(s) = \frac{G(s)}{1 + G(s)H(s) - G(s)}$$

By setting $R(s) = \frac{G(s)}{1+G(s)H(s)-G(s)}$ we are left with two systems with the same overall transfer function.



3.5 Model-based design

We will end this chapter with a short discussion on model-based design. The intent of this section is not to give you a full understanding of model-based design - that would require its own book - but to put the concept of block diagrams into the context of how you will more than likely use them in industry. This context will also help you to understand how control theory, and the algorithms you'll learn throughout this book, fit into the larger systems engineering process.

Model-based design, as the name implies, is a way of engineering that uses a model[4] as the main tool to help you develop your system rather than real hardware like prototypes and physical mock-ups. If you've never participated in a large-scale engineering effort, you might not understand what is meant by that last sentence. Therefore, I think comparing these two approaches will help you understand how a model can take the place of the physical hardware across the entire engineering life cycle - and ultimately, how block diagrams help with the modeling effort.

Engineering projects typically start with a needs analysis. Before you ever start designing anything you need to determine what you're trying to accom-

[4]Model in this sense means a mathematical model, not a scaled physical version of what you're building

plish. From there, you start to lay out a concept of a system that will meet the needs of the project and decide on the architecture you want to pursue.

It is at this point that a decision is made: do you want to take the model-based design approach or not? The answer is not always the same for every project. It depends on several factors like the size of the project, the cost of the hardware, how long it takes to build the hardware, and the capability of the team. Typically, large engineering projects like those you'll find in the automotive and aerospace industries rely on model-based design because the hardware is expensive, mimicking the real operational environment in a test is difficult, and it allows very large teams to work in the same environment effectively.

In model-based design, a lot of effort is put into creating very accurate mathematical models of the physical hardware and the environment in which it will operate. Building the model is typically done in parallel with the hardware and software engineers who are developing the system. If you recall from chapter 1 and the section on the three different problems control engineers typically solve (the system identification problem, the simulation problem, and the control problem), model-based design is what allows us to solve the simulation problem. We have our model of the system and we know the inputs to the system, therefore, we can use simulation to determine how it will behave given those inputs. If your models are accurate enough, then you can make a claim with some confidence that the real system will behave in a similar manner. Ultimately, it is simulation that takes the place of testing on physical hardware in the model-based design approach.

No matter which design approach you take, requirements are set at the system level and then as the project matures, they are flowed down to smaller and smaller components until you have enough of an understanding of how

the system needs to behave to begin implementing the design. In model-based design, the requirements can take the form of the block diagrams and mathematical models. In other words, the designers are trying to implement a system that matches the model. Of course, when you are creating your model, you have to make sure that it is something that can be realized in the real world. It does you no good to develop a model that is physically impossible to implement or has no basis in reality.

Modeling, simulating, and implementing the design don't occur in a linear fashion where you always progress from one to the next in order. In real engineering projects, all three contribute to each other as the design spirals into a more mature system. It is this quick turn between making a change to the model and then simulating it to determine the impact on the system that allows projects to catch design errors early. Without the model-based design approach, projects typically don't find design errors until the system is mature enough to build prototype hardware and a physical test can be executed. If your hardware is inexpensive and can be built quickly, then using the hardware early to find errors is a perfectly good way to approach your project. That approach might even be easier because you don't have the risk of making an error in your mathematical models. However, for many large engineering projects, the risk of modeling errors is much lower than the risk to the budget and schedule with multiple hardware design cycles.

Of course, even the model-based design approach doesn't remove the need for prototype hardware and physical test - it just reduces the number of hardware design cycles and design errors that you'll find while testing. When it comes to physical testing, model-based design can allow you to start testing hardware earlier in the project life cycle by using hardware-in-the-loop (HITL) testing. With HITL, you can selectively test individual physical com-

ponents as standalone units, even if those components won't fully operate without the rest of the system, by replacing the rest of the system with a simulation. Simulated inputs are fed into the physical component and the sensed behavior of the component are fed back into the simulation. In this way, you get the dual benefit of testing the real hardware as soon as it's available, as well as, the ability to precisely control the inputs to the unit under test.

Model-based design simplifies test in another way. One purpose of testing is to verify that the real physical system behaves the way you need it to in an operational environment. This can be difficult to accomplish if the operational environment is hard to achieve. Take, for example, verifying that a satellite is capable of pointing its antenna accurately in space. You can mimic the space environment in a thermal vacuum chamber and you can try to simulate the microgravity environment with clever rigging. However, you'll never be able to design a physical test that perfectly matches the space environment - and any test that comes close to it is very expensive to set up and run. Instead, you can develop models of the space environment and the satellite, verify each component of the model is accurate, and then use the model to simulate how the system will behave in space. This might seem like a risky approach, but a huge amount of money and time has been spent developing very accurate models in the aerospace industry.

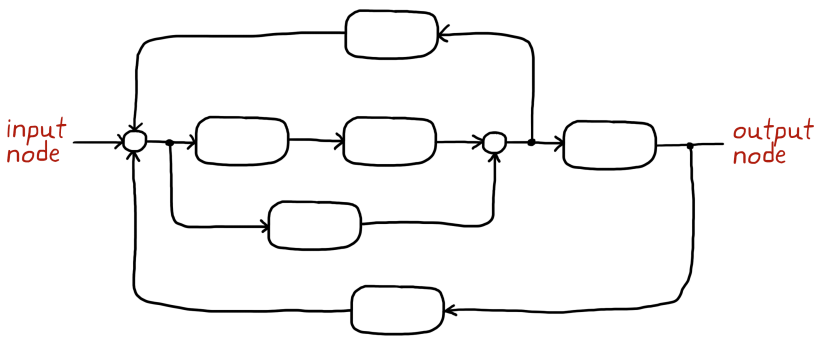
At this point, you might be wondering how any of this has anything to do with block diagrams. Well, if block diagramming a system wasn't so useful, then perhaps not much. I mean, we could simulate our system by coding the mathematical equations in a text-based programming language like C. Essentially, we would just be writing a piece of software that we could execute to generate the simulation results. However, engineers (especially control

engineers) have found it useful to construct their models as a block diagram rather than as textual code. This is because the systems we are creating are inherently hierarchical and lend themselves nicely to a graphical representation.

This has led to model-based design tools like Simulink from Mathworks and LabVIEW from National Instruments to use block diagrams as the way they describe and visualize dynamical systems. Of course, with these tools, you can build models that are nonlinear and in mixed domains[5] and the tool itself takes care of all of the transformations and keeps everything consistent. However, the understanding you get by working with block diagrams by hand with transfer functions, and especially when dealing with simplifying them, will go a long way to getting you comfortable working with model-based design tools.

3.6 Try This!

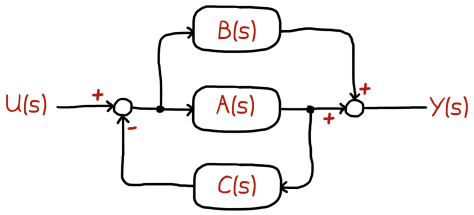
1. Find all of the forward paths, parallel paths, and loops in the following block diagram. How many nodes are there?



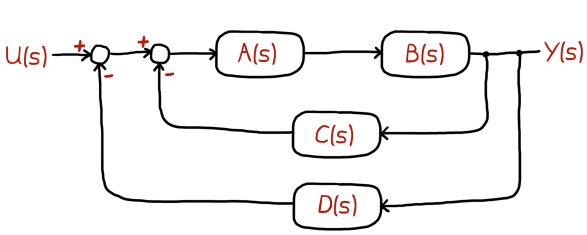
[5] Your model can have elements in the time domain, frequency domain, and the s domain simultaneously.

2. Simplify these block diagrams. Find the transfer function for each system.

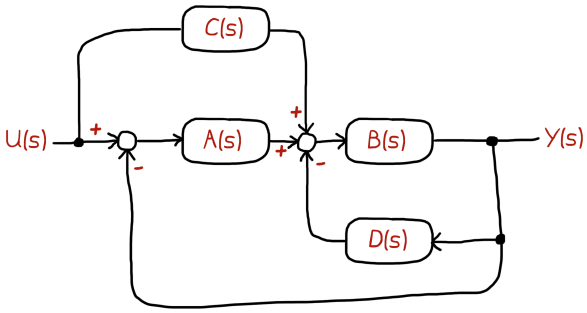
a)



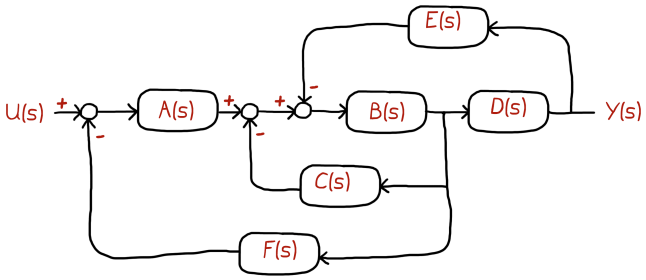
b)



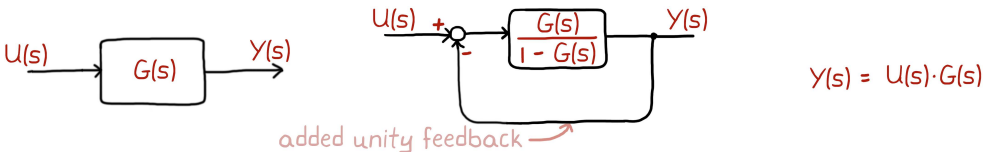
c)



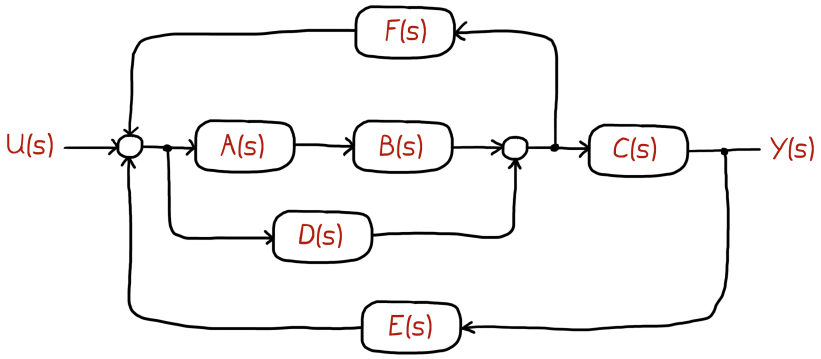
d)



3. Prove that adding unity feedback to transfer function $G(s)$ results in the forward path function $\frac{G(s)}{1-G(s)}$.



4. Reduce this block diagram to create a system with unity feedback.



Chapter Credits

Meg Douglas	Kirkland, Washington	Mariano Lizarraga

Thanks for making this chapter awesome!

Appendices

A Transforms

A.1 Fourier Transform

Fourier Transform

$$F(w) = \int_{-\infty}^{\infty} f(t) e^{-jwt}\, dt$$

maps time to frequency

Inverse Fourier Transform

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(w) e^{jwt}\, dw$$

maps frequency to time

Coming soon!

Appendix Credits

Meg Douglas Kirkland, Washington

Thanks for making the appendix awesome!