What's a Control System and Why Should I Care? A whirlwind tour through the basics of control systems for students about to take that first class. [⋆]

Daniel Y. Abramovitch [*]

[*] *URL: dabramovitch.com (e-mail: abramovitch@stanfordalumni.org).*

Abstract: This paper is designed as a primer for college level STEM students about to take their first formal class in feedback control systems. This means that the explanations assume the reader has had some necessary math, science, and programming classes. With that in mind, this document aims to provide not only an overview of what the student will see in learning about control, but context for why those methods are used and why the whole field is important. Even for the scientist or engineer who has long ago finished college or turned in that doctoral thesis, this basic context for the field of feedback systems can provide some useful understanding.

Keywords: Control primer, tutorial, second-order systems, closed loops, computers, transforms.

1. INTRODUCTION

As Dennis Bernstein so eloquently wrote in in his seminal 2002 paper (Bernstein (2002)) feedback is the hidden thread of many world changing technologies of the 1900s. As we progress through our ever automating world of the 2000s, Dennis' insight appears to be the proverbial tip of the iceberg. With agents, networks, robots, self driving cars, smart grids, swarms of drones, etc. the prevalence of human built feedback systems in our lives is increasing exponentially. In parallel, the growing consciousness of the role of feedback in biological and environmental systems – and the need to model and quantify these – cries out for a public understanding of the need for, uses of, and pitfalls in feedback systems.

At the same time, the prevalence of cheap real-time computing platforms and hobbyist accessible programmable cars, house automation, robotics, and drones means that there will be a proliferation of computer controlled devices, with most of the code written by people with not even a basic understanding of feedback systems.

This document is written as a primer, a cheat sheet, an introduction for students about to take their first college level class in control systems. Having taken a few of those classes myself, it is a wonder that I ended up working in the field of control. What is often missing from those first classes is context (the "Why are we doing what we are doing?" part) and a motivated road map. The course syllabus might point out a bunch of topics to cover, but why they are being covered is often left as a meditational exercise for the student. Being of a simpler, more direct mindset, I thought I'd go through that stuff so that you (the student) can actually be motivated to learn the material. You are still free to align your chakras. Namaste.

It is important to me (and to the larger world) that you actually have a clue about this stuff by the time you finish the class. Not being in control of your class (no pun intended) I can best affect this by giving you a clue before the class starts. Why does this matter? Isn't machine learning (ML), big data (BD), and artificial intelligence (AI) a lot hotter this days? Yeah, they are but these fields are not mutually exclusive and while system theory (which encompasses controls and signal processing) can function without AI and ML (and have been doing so for years), the converse is not true (at least in the physical world). A good analogy can be made from riding a bike. The visual processing, the path planning, the adjustment to weather conditions, the improvement of technique is all a higher level learning process, akin to AI and ML. However, none of that can happen if you can't keep the bike upright or steer it away from danger, or simply get it to move in the correct direction. These more low level tasks are akin to the control and signal processing fields. They respond more quickly and more simply to instantaneous events. Improving one's biking ability might be machine learning, but the instantaneous operation is all about system theory.

What's more, there are some bedrock principles that get exposed when one learns system theory that are extremely useful "laws of physics" type rules that can guide and sanity check machine learning. Anyone who has done any programming, built any machine or circuit, or tried to make some sort of algorithm work learns very quickly that it's important to have sanity checks put in along the way to make sure we are creating something that has a chance

[⋆] Daniel Y. Abramovitch is research engineer working in Silicon Valley. Copyright (©) Daniel Y. Abramovitch. Permission to share document in an unaltered form is granted. Permission to excerpt, alter, or sell any portion of this document is withheld without the express written permission of the author. This document represents the best effort of the author. There are no warranites expressed or implied about the accuracy or utility of its content.

of working. So it is that system theory principles can be applied to sanity checking machine learning. (Analogously, when we do signal processing and/or control we have to be careful not to violate the laws of physics. Any algorithm that depends on a violation of the laws of physics is likely to fail.)

So, I hope you will find this useful, both as a standalone document, but also as a cheat sheet to help you with that first (and perhaps only) controls class. The goal isn't to beat theory and proofs into you. That's a good job for professors. The goal is to help you see why all that stuff they are yapping about and all those theorems and proofs might just be worth your while.

2. UNDERSTANDING THE AUDIENCE AND PREREQUISITES

In any attempt to teach material, it is important to understand the level of the audience so as to create an "impedance match" between the teacher and audience. As this document is a primer, a cheat sheet for those of you entering your first formal control systems/theory class, I have to assume a few things about what you already know. Some things are almost mandatory: that is, you will get something out of this document but then the actual class will be a struggle. Some of it is optional: your understanding will be a lot deeper, but you shouldn't flunk that first class. By the same token, if I'm going to be enough of a tool to tell you that certain things are necessary, I should at least be kind enough to tell you why I think that's true. But enough about me; let's talk about you.

This document assumes that you are a third or fourth year college student studying a STEM (Science, Technology, Engineering, and Math) field and you are about to start your first class on control systems. Maybe it's called control theory or automatic control or feedback systems. That particular detail isn't important because no matter what they call it, they'll show you a few drawings of systems that might use what you are about to learn and then dump a crapload of math on you, so that at the end of the quarter or semester, you are still wondering what you learned and why it is at all useful.

By this time in your STEM college career, you should be completely done with calculus (differential, integral, multi variable), will have slugged your way through that first class in differential equations, should have had a class in linear algebra, and have had some transform theory (and probably hated that, too). You will have had college chemistry and college physics, although if you are a computer science major, it's not clear that the last statement is true. You will have had at least one programming class, although that might very well have been a class where you are programming in Matlab or Python. Still, the idea of writing code to implement a mathematical algorithm should not be foreign to you. All of these have a critical piece to play in understanding the mathematical underpinnings of control systems. Sadly, the how and why of each of these pieces is often neglected in the class themselves and in the control systems class you are about to start. Sorry; not my fault; trying to help.

Depending on your major, you may have had some biology. At this point, it is not clear if you have had probability and statistics. The last three are very useful, but not critical to learning basic control systems work. Similarly, you may have had a first class in circuits (if you are an electrical engineer), or dynamic systems (if you are a mechanical engineer), or process dynamics (if you are a chemical engineer). Again, very useful for context but not absolutely necessary. Even those that have had a few circuits classes have probably not gotten to the point of op-amp (operational amplifier) circuits which is really the part that applies best to controls work.

So, why are these subjects important? I'll stick to the critical ones here. We observe/model the physical world with science. Some science is entirely experimental: observe stuff, write it down. However, for science to actually be useful in doing more stuff, the observations must lead to an abstraction, a model of what is going on. One of my favorite quotes on models comes from Stephen Hawking in his book, *A Brief History of Time*:

> I shall take the simple minded view that a theory is just a model of the universe, or a restricted part of it, and a set of rules that relate quantities in the model to observations that we make. It exists only in our minds and does not have any other reality (whatever that might mean). A theory is a good theory if it satisfies two requirements. It must accurately describe a large class of observations on the basis of a model that contains only a few arbitrary elements, and it must make definite predictions about the results of future observations. (Hawking (1988))

Science gives us a mathematical descriptions/models. Usually, the more descriptive and precise the model, the higher its complexity, which can make it hard to actually work with. To do something with a model, we often have to simplify it. The big question is whether our simplifications take us too far from describing the physical system behavior that we observe.

Now, for some specifics. Physical systems are often described by differential equations. Of great interest are linear, time-invariant differential equations (linear: double the input, you double the output; time-invariant means that the equations describing the system don't change if the system is hit with an input tomorrow versus if it's hit with an input today. We can't guarantee that the world can be modeled this way and in fact,it often can't, but there are a lot of advantages to making this modeling approximation, and so we do it whenever it's at all reasonable.

It is for this reason that we start by using science to give us a model of some system that is often in the form of a differential equation. If it is linear and time-invariant, great. If not, we likely will try to linearize it and assume that the rate of change of the system parameters are slow enough relative to our signals in the system so that we can more or less assume that they are time-invariant.

When the differential equation that describes the system behavior is less than ideal (almost always), we would like to change this. We do this by feeding back a portion of the system response (there might be more than one measurement for different points of the response, but for now let's act like one of those old, terrible "Highlander" movies and assume there can only be one). When we feed back that portion of the system response and combine it with the reference input to the system we produce a new system and the governing differential equation of that new feedback system (as the output was fed-back) is different from the original system.

In other words, with feedback, we have changed the characteristic differential equation of the system we are trying to control. That's it. That's what feedback control does: it allows us to change the equations governing our system's behavior. Now, that can be good or that can be bad, and the rest of the field of control theory is about doing this in a way so that it is good. The "how-it's-done" details are what the whole class is about, but it's good to know why we are here in the first place.

Okay, so we are going to change the behavior of differential equations, but if you remember that class your first or second year, that wasn't a whole lot of fun. It's hard to get intuition beyond a first order system. Okay, you worked out the second order response with those sine and cosine terms and that decaying exponential, but life's too short to work out that 10th order solution. Especially in the days before digital computers (what you call computers, but they were once made with analog circuits as well). That's why there are tricks to get around this stuff and the two most useful are transform theory and linear algebra.

Transform theory passes a differential equation through a special integral to map that equation in the signal domain, say time (t), to some frequency variable, say s. In the case of a Laplace Transform, s is a complex variable valid all over the complex plane, but in the case of a Fourier Transform, we restrict s to a single line, the imaginary or $j\omega$ axis. When the differential equation is linear and time-invariant, there is a one-to-one correspondence between the signal domain and the transform domain. This means that we can take a signal/description/model in the time domain, map it to the transform domain solve the problem there, and then transform it back to get the answer we want.

At this point one might ask why anyone would go through those extra steps. The answer (and there has to be a reason to incur what seems like a lot of extra work) is that that long path is often a lot simpler, with a lot more intuition. One of the main reasons is that when a signal described by a time domain function enters a system described by a differential equation, the response is obtained by a convolution integral. Not only is the convolution integral a pain, but one has to redo it for each new signal. On the other side of the transformation, though, the signal transform is multiplied by the model transform and multiplication is a lot – and I mean a lot – easier than convolution. Plus, once we have that transform of the system model, we can use it over and over again by multiplying it via other signal transforms.

When we work with systems then, there is an input/output relationship of the system in the transform domain called a transfer function. The roots of the characteristic polynomial of the transfer function correspond to the roots of the homogeneous solutions of the differential equation, but in the transform domain, we are doing algebra in place of solving differential equations, and algebra is a heck of a lot simpler. And that's it: we need to know the transform domain stuff because it allows us to solve the differential equation more simply and get a lot more intuition than we can get from the differential equation on its own.

What about that linear algebra? Well, you may or may not remember that we are able to turn an n$^{\text{th}}$ order linear, time-invariant differential equation into a series of n first order differential equations, stack them into a matrix, and then the properties of the matrix tell us about the behavior of the differential equation. In fact, the eigenvalues of that matrix correspond to the roots of the denominator of the corresponding transfer function. Different ways of looking at the same problem.

So, we need to know differential equations as those will describe our systems. We need to know transform theory and linear algebra so as to simplify dealing with interactions between multiple interconnected systems. We need science for our initial modeling and understanding. We need some programming because when we design the thing that will adjust the signal that was fed back, the feedback signal, we will almost certainly build that in a computer program.

3. BASIC PRINCIPLES AND TOPIC AREAS

The basic principles for teaching control in this document are to start at a high level with physical examples, and follow up with the underlying principles that tie those examples together. The process iterates as we select topics that are of importance in the study of control systems, with each level adding both physical and general insights into some aspect of control systems.

The examples are drawn from both history and everyday life. Since we are all humans in the loop at many points in our daily lives, these systems are particularly helpful. They are physical, they are ever present, and the decision process that humans have to make can be easily visualized. It is a simple step to then say that what we are really doing is teaching machines to do the same thing.

Likewise, simple mechanical feedback systems provide a very visual understanding of human built feedback systems. They span history, from the earliest outriggers and water clocks (Abramovitch (2005); Mayr (1970)) to the ubiquitous toilet examples. It is a simple process to turn these systems into a block diagram and then discuss how we would teach a machine to make those decisions.

Along with that, there are some fundamental topics that need to be covered for any students to begin to have an understanding of control systems. They include:

- Outer loops (big picture) versus inner loops (small picture).
- Discretization: why it's a big deal, what are the benefits, and what are the pitfalls.
- Delay and latency: how it matters little in signal processing and how it is all defining in feedback systems.

- Modeling of systems: how we get models from science and why we need them.
- Math: where does it come in with modeling, and how does it help us understand our systems.

4. TOP LEVEL: DEFINING SOME BASIC TERMS

One place where the first controls class is often a bit meager is to motivate and define the basic terms used. I'll give a few basic definitions here, but there should be a more complete practical glossary at the end of this document (Section 23). At a top level, the following definitions are useful:

Control: Make something move where you want. That's pretty much it: control is about making stuff move in a way that you want it to move. Originally, we would have been talking about moving physical objects around, but the idea of moving stuff can refer to electricity, chemicals, data, etc. Still, when we talk about moving something in a way that we want it to move, we are talking about control.

Feedforward: Estimate (guess) how to push it but never use where you see it going to adjust how you are pushing. If we think about how we often think about doing things for which we are very familiar, we do it in a feedforward way. When we are cooking something that we have cooked a hundred times and we know the recipe cold, we just execute it without having to taste the food in intermediate steps. That's feedforward and when we know things really well, it's a very efficient way to do things.

Feedback: Look at where it's going as you push it and adjust how you are pushing. When we aren't familiar with the recipe, or we are working in a new kitchen with new tools, we taste the food a lot to see if it's cooking the way we want it to. This is feedback: make a sample of the output of what we are doing and compare it to where we want to be moving to. Again, this can be feedback of signals moving inside a computer as much as a car moving down the street.

With these definitions, we span the basic field of control. One thing to realize is that the principles of control and the phenomena of control in nature is fairly ubiquitous. **Control in general and feedback in particular happens everywhere in nature and that people are doing control all the time (when we throw a ball, ride a bike, drive a car, put a key in a lock, or find a keypad on a phone). Once we internalize these examples from everyday life, we can get to what we are really doing with control design: we are teaching machines to do what we humans do all the time and what happens in many natural systems.**

Okay, we have a few definitions. Let's look at a control system that is hopefully fairly universal. There are lots of choices here, but I have found the shower loop example to be most universal. This is displayed with the diagram on the left of Figure 1, and the "almost block diagram" on the right. The steps are phrased in both the physical steps that the person takes and in their control system nomenclature. From here, it is easy to see that this is an example of a universal set of steps, that show up in all feedback loops:

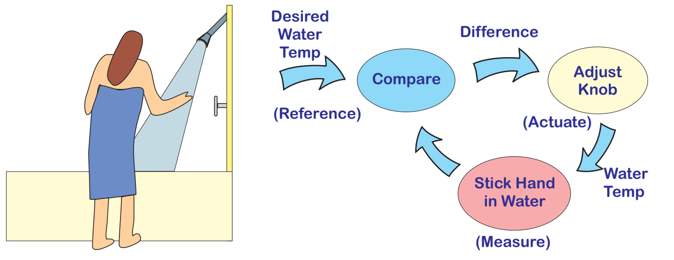

Fig. 1. A common feedback system, and the "loop" abstraction.

- a reference signal,
- a measurement,
- a comparison element, and
- an adjustment mechanism.

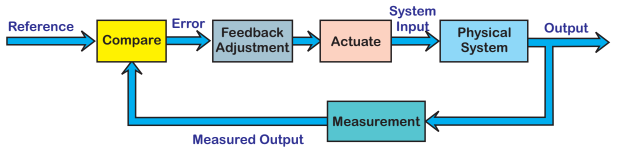

Fig. 2. The basic elements of a feedback loop.

We now move from the "physical" block diagram on the right of Figure 1 to a more generalized block diagram of Figure 2. Figure 2 names the internal elements of the loop more clearly, and adds something about its function. Every feedback mechanism has (Mayr (1970)):

- a sensing element (that which makes the measurement),
- a comparator (that which compares the sensed value to the reference signal and turns it into an adjustment),
- and an actuator (that which physically makes the adjustment).
- Furthermore, it runs in closed-loop – that is – a portion of the sensed output is fed back into the system.

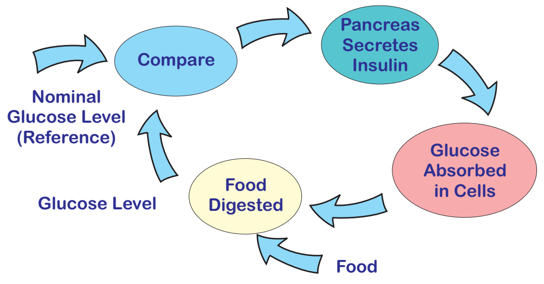
Compare
Pancreas Secretes Insulin
Nominal Glucose Level (Reference)
Glucose Absorbed in Cells
Food Digested
Glucose Level
Food

Fig. 3. A simple blood sugar feedback loop.

With the generalization of Figure 2, let's look at another feedback loop that is present in nature: the sugar loop in which our bodies regulate our blood sugar, shown in Figure 3. Now, this loop is internal and relates to something that many folks understand, that the pancreas wants to maintain a steady range of blood sugar, despite disturbances (food input, stress, etc.). Historically, understanding relationship between insulin and blood sugar led to first diabetes treatment. As most folks know, diabetics need to measure their blood sugar and adjust (by eating or injecting insulin). In other words: diabetics are closing the loop themselves.

This also brings a natural discussion of sample rates into view, since most diabetics will only do this check 4-6 times a day, while a non-diabetic system will make this adjustment constantly. In the past few decades, insulin pumps have been developed, but this run open loop, that is without being regulated by a measure of the person's blood sugar. This is a natural segue into the area of artificial pancreas (Haidar (2016)), which are a big topic of controls research and practical application.

One of the major differences between the manual measurements and adjustments made by diabetics and the automatic ones done by a healthy body is that the healthy body makes these measurements and adjustments on a continuous basis. The diabetic must make periodic measurements and adjustments throughout the day. How often do they need to measure? How representative of their actual blood sugar are their periodic measurements? How well do their periodic adjustments represent the ideal adjustments that would be made continuously? These are all issues of sampling and discrete-time feedback. It is a subject unto itself, but it is critical to the understanding of how we do control on physical systems with computers, which can only look at the data at discrete measurement times (Sections 7 and 19).

Lest you think that human built control systems are a recent thing, I have a couple of examples that date back in history. The first is the water clock of Ktesibios, an ancient Greek inventor who lived in Gaza, in the third century B.C.E. (Mayr (1970)), shown on the left side of Figure 4).

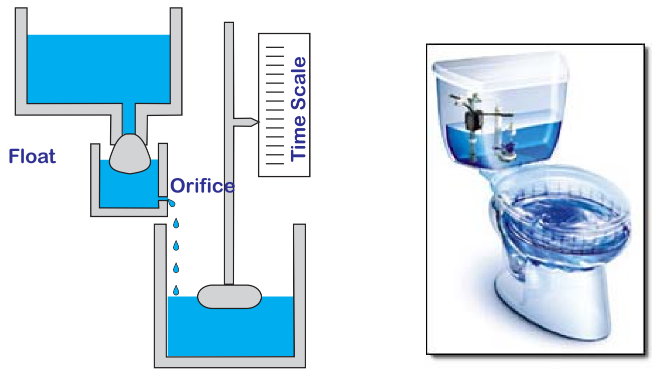

Fig. 4. The float valve of Ktesibios and its modern day descendant.

Ktesibios lived in a time when the primary mode of time-keeping was with a sundial. Not only do they not work well at night or on cloudy days, but the hours are literally longer in summer than in winter. The mechanical clock wouldn't be invented until sometime in the 13th century CE, somewhere in Central Europe (Jespersen and Fitz-Randolph (1999); Barnett (1998)). Ktesibios, living in a warm Mediterranean region, created a water clock that measured the flow of time by how fast the bottom reservoir was filled from the top reservoir (on the left of Figure 4).

The issue is that water flows faster when the supply is higher, meaning the measure of time in the lowest chamber (marked by water level) would not be uniform. Ktesibios' clever solution was the float valve, which kept the level of an intermediate reservoir roughly constant, allowing the lower reservoir to fill at a more constant rate. The float valve is a feedback mechanism, as it regulates the level of the middle chamber based on a combined measurement of the level and actuation (sealing the inlet). The feedback accomplishes the task of stabilizing the flow of water into the lower chamber to an effectively constant rate. Water clocks don't work too well in places where water freezes, and I've never seen a water wristwatch, but it was a clever invention. Float valves are still used in the modern day toilets (another ubiquitous example). Lift the tank lid on your home toilet and you will see a float valve that seals the inlet when the water level in the tank rises to the proper level. Impress your friends at parties by taking them into the bathroom, removing the toilet tank lid, and regaling them with the history and significance of the float valve.

Fig. 5. The outrigger is also a great, physical example, and much older than the float valve (Abramovitch (2005)).

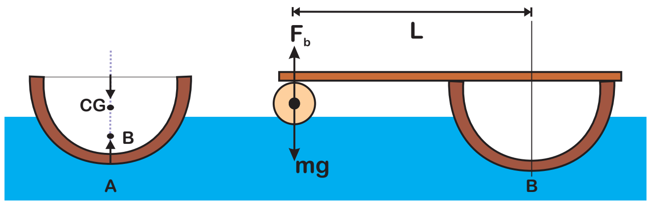

Fig. 6. Diagram of the operation of an outrigger. On the left, the half cylinder canoe has a very small righting moment and can easily capsize. On the right, the addition of an outrigger uses a combination of buoyancy and weight to stabilize the roll of the boat.

I have a particular affinity for the outrigger (as seen on an outrigger canoe) (Figures 5 and 6) as an example of ancient feedback, since it is more fun than a toilet, and is at least 1200 years older than the water clock (Abramovitch (2005)).

I only realized it was a feedback mechanism after my then three year old son pointed to an outrigger on the beach and asked, "How does that work?" "Well, it's a float so when the boat tips one way, the buoyancy helps straighten it out. It's also heavy so when the boat tips the other way, the weight of the float tends to bring it back down." And as I spoke, I realized I was describing a feedback mechanism. Some research led to the study in (Abramovitch (2005)).

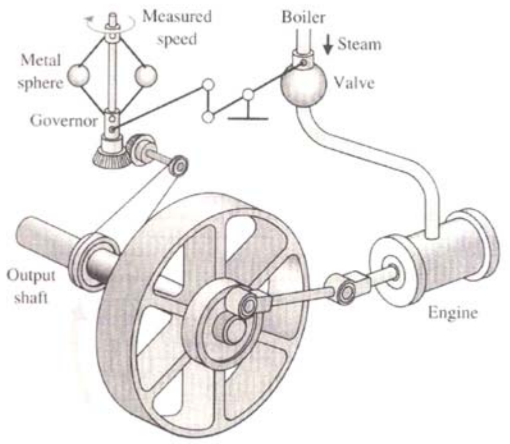

Fig. 7. Drawing of a Watt flyball governor attached to a steam engine. (Bernstein (2002))

Now, both of these examples are very simple in their operation and can be understood quite intuitively. In fact, little in the way of math was applied to analyzing either one of them. All that changed late in the 19th century, CE, when the industrial revolution was in full swing and steam engines needed to be regulated. The solution was a flyball governor (Bernstein (2002)), sketched in Figure 7. The operation of the governor is that as the steam builds up, the engine speeds up, causing the balls to fly up farther. They are connected by levers to a valve that opens up as they rise, literally "letting off steam" and slowing down the engine. Thus, they regulate the speed of a steam engine with feedback.

The main historical note about the flyball governor is that it is when feedback control first "went viral," at least in the pre-electricity days. Because it was so useful in the operation of steam engines, and coming right at a time when mathematics was first being applied to engineering, it represents the first feedback device for which was studied with significant mathematical analysis. The first technical articles trying to explain the behavior of the feedback mechanism were written about flyball governors, one of them by James Clerk Maxwell (Maxwell (1868)) (he of the Maxwell Equations from electricity and magnetism).

It is also worth emphasizing that there are two types of feedback. Positive feedback in devices as in life, causes something or someone to amplify (or do more) of what it was doing before. Positive feedback will cause a system to amplify the input. If the gain is high enough, the output can grow until one of two things happens: something saturates (hits some limit and then bounces back and forth between limits, which is useful for building devices such as oscillators) or something blows up (which is generally not that useful and can be directly related to such disasters as the explosion of the Soviet nuclear reactor at Chernobyl (Figure 8,Stein (1989, 2003))).

Fig. 8. Picture of the reactor at Chernobyl that exploded after it went into thermal runaway. The final trigger was caused by conditions in the reactor having changed so that the control being applied was positive feedback.

Negative feedback (in both devices and life) causes something or someone to deviate less from some desired path than before. Negative feedback will cause a system to become less sensitive to changes. It generally trades absolute gain for stability of gain. For example in building electronic amplifiers, their overall gain is limited by using feedback, but that gain remains steady despite temperature changes, wear, etc. which allows us to build devices that behave reliably the same way time after time.

It turns out that much of control design and a large part of your first control class will be doing math to show that the model describing the feedback system is stable: that is that the roots of the differential equation of the feedback system are all stable. After all, it makes no sense to try to improve the behavior of a system with feedback only to have it oscillate wildly out of control. It's as if one has a steering augmentation system in a car that makes it fly off the road. We will talk about some of the basic math behind this in Section 13, but for now it's just good to know that this is a thing.

One of the repeated themes will be that we can have a basic feedback control design that is stable for some parameter values and is not stable for others. A big part of our design then is to improve performance while still providing some margin, some slack for stability. There will be parameter values for which our system goes unstable, i.e. for which oscillations start growing until something breaks. Bad things happen when negative feedback becomes positive feedback. Chernobyl is the perfect example. A simple explanation for what usually causes this is that the correction lags the error by too much time, a too much too late scenario.

In this section we've gone from broad definitions to something that is a current topic of control research and application in a few minutes. Each step builds on the use of physical examples, to abstraction, to application to more physical examples. If we stopped at this point, you would be able to tell someone what a feedback system was, describe its components, and give two simple examples. At this point, we are able to circle back and give you considerably more detail.

5. ABSTRACTION TO BLOCK DIAGRAMS AND THEIR COMPONENTS

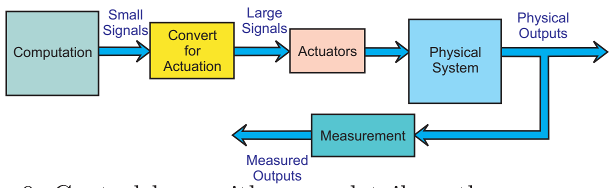

Fig. 9. Control loop with more detail on the components. This loop lacks feedback.

Now that we've gone through a bunch of principles and examples of feedback, we're going to want to generalize. We need to go up lone level in abstraction and talk about the generic block types in a control loop block diagram. Figure 9 is a nice starting point for a discussion about the components of a control loop and how they affect the size and complexity of systems for which we can build controllers. Each of these components needs to be defined. For example:

Sensors: tell us what is happening.

Converters: change what sensors see into something we can compute with and back again into something that does something about it.

Computation: make decisions about what to do.

Physics: what the real world is really doing.

Modeling: how we describe this to our computation.

Actuators: do something about it.

With these items defined, we can go into a bit more detail. I have assumed that all of you have taken at least one programming class. Here is where that becomes useful. Simply writing down the math of the feedback controller does not get it built into a working system. It has to be implemented. The list above talks about the pieces of the block diagram that make a control system. The top diagram of Figure 9 has a physical system on the far right. This is what we want to control. We can push on this system with an actuator, a physical device that translates electrical signals into specific actions that affect the system (upper branch). We can measure the outputs of this system (lower branch). To drive the actuator, we have commands coming from some form of computation. Those commands need to be translated into signals that make the actuator do something. In humans, the computation is in the brain, the conversion of signals is through the nerves, and the actuation is via muscles that cause part of the body to move.

In a modern day car, the computation is via a specialized computer that sends signals to converters, that in turn scale up those signals to cause the tires to turn, or the fuel injection to increase, or the brakes to be applied. The generic blocks are always there in most systems, but their specific realizations are all over the map. Note that in this diagram, the measured signal does not get back to the computation. There is no feedback, and so the operation of this particular diagram is open loop.

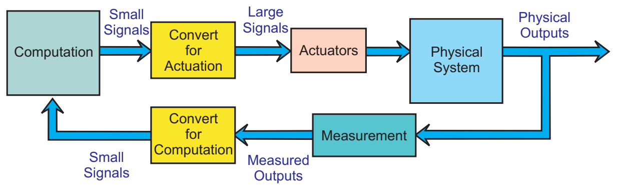

Fig. 10. The diagram of Figure 9 with feedback added.

Figure 10 is our first iteration on this diagram, in chich we have fed the measured output(s) back into the computation engine. Important side note here: we are feeding back the measured output, which may or may not be the output we need to control. One might correctly guess that there is another whole branch of this that involves how me make measurements and where we put our sensors. Suffice it to say, measuring the speed of a bicycle is a lot more benign than trying to measure the temperature inside a nuclear reactor. For right now, it's enough to realize that a measurement is a representation of the signal we want to know about and the speed and quality of that measurement may (and does) affect what we can do with the feedback loop in a major way. For simplicity, we will stick with single-input, single-output (SISO) systems in this document, but it should be obvious that we can measure a lot of different signals on a system and we can put multiple actuators on it to push on different places. It just makes understanding the principles a lot more complicated.

With Figure 10 we now have a feedback loop, in which some portion of the measured output or outputs are fed back into the computation engine in order to generate an error signal (or signals). Our computation can take that error signal and massage it into a form that allows us to give the right commands to the input of the physical system (going through the signal converter and actuator). Honestly, the point of a controls class is to teach you mostly how to do that massaging, that shaping of the response to the error, given what you know about the other pieces in the loop.

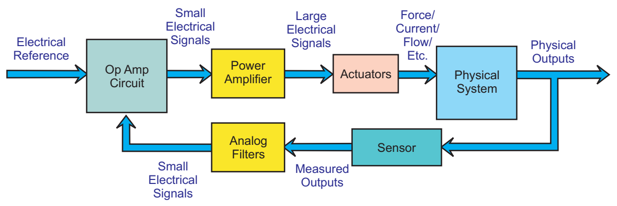

Fig. 11. The diagram of Figure 10 where the computation is done via analog circuitry.

We haven't said anything about how we are doing the computation, but Figures 11 and 12 show two iconic and important cases. Figure 11 involves implementing the math

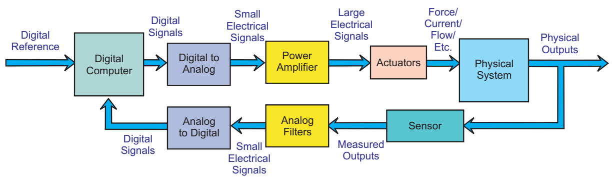

Fig. 12. The diagram of Figure 10 where the computation is done via digital computation.

computation in analog circuitry. This was the dominant form of electronics from the 1930s well up to the 1980s. The measurement would return an electronic signal and that signal could be routed through circuits to modify it accordng to the differntial equations that described the circuit components. Most of the time, folks thought of this in the Laplace Transform domain, converting ths system model, the models for the actuators, sensors, and conversion circuits, and the control circuits into their transfer function form. The Laplace Transform turned the issues of stability of the closed-loop differential equation into an algebra problem, which was much easier to understand. A huge number of devices were built this way, including the early age of communications, flight, the space race, radio, television, etc.

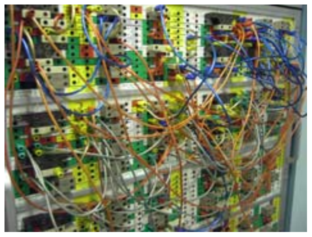

Fig. 13. A "program" on an analog computer. Can you spot the "typo"? Yeah, me neither.

One issue with analog circuitry is that the input and output values of any circuit can take on a multitude of values. Secondly, the behavior of the circuits may change over time or due to environmental conditions or aging of the circuits. Finally, and perhaps most importantly, it is easier to change a computer program than it is to rewire a circuit. The mess of wires in Figure 13 is an actual program in an analog computer. No idea what the code is saying or how to debug it. This means that if we can somehow replace the circuits to do the math required to implement a controller, then we will almost always choose to do that. The issue that dogged the latter approach for many years was that implementing real-time math (math that had to keep up with things going on in the real world) in digital computers was difficult and expensive for many years.

It was NASA's Apollo program of sending people to the moon and bringing them back safely that pushed digital real-time computation. For all the benefits above, NASA made the very prophetic choice to go digital. In the years since then, the march of Moore's Law and the ever shrinking of both the size and power requirements of computation, while raising the available computation speed has made these issues go away for entire classes of physical problems. Thus, digital computers (what you know simply as computers) replaced analog computation for most applications. Looking at Figure 12, we see the main addition is an extra set of converters to take the electrical signals into and out of a digital form that is usable by the computer.

Analog circuit implementations of control laws are a far less significant part of the modern control landscape than even a few decades ago. An emphasis that makes sense is one that replaces the op-amp circuit computation of Figure 11 with a digital controller shown in Figure 12.

That being said, I wouldn't be fair to history if I jumped straight into digital control examples without giving examples of things that were made possible by the first analog electronic circuits and their use in control systems, as shown in Figure 11.

Analog feedback was significant in the development of telephony, both positive feedback for oscillators and negative feedback for amplifiers (Bernstein (2002)). The idea of using negative feedback in amplifiers was first proposed by Black (Bernstein (2002)) when working at Bell Labs. It worked amazingly well, although engineers there noticed that things could go unstable and started trying to understand the mechanisms that caused this. The names of Nyquist and Bode become prominent in this time as they did the first work on understanding how to keep those "operational amplifiers" (op-amps) stable. At no point did these folks connect the stability of these amplifiers to the behavior of flyball governors. That "aha" moment had yet to arrive.

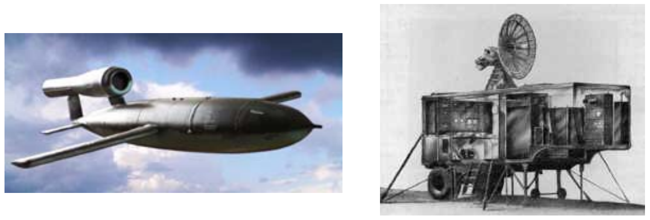

Fig. 14. Pictures of the German V1 flying bomb, which flew on a straight line to the target using automatic feedback control provided by analog electronics. On the right is a picture of the American M9 gun director which directly tied radar into the pointing mechanism for the gun and calculated by how much to lead the target, also using analog electronics. In 1944-45, these devices engaged in what was the first battle between automated devices.

At the same time, World War II was on the horizon and the Americans and Germans were both working on rockets and controlling them. The British and Americans were working on radar, and the issue of using radar to guide anti-aircraft guns had become critical as humans really could no longer track fighter planes or bombers manually. David Mindell's book and articles (Mindell (1995a,b, 2002)) bring to life the fascinating parallel evolution of German rockets guided by electronic feedback loops and American anti-aircraft gun directors, which used analog electronics to close the loop between the radar tracking and the accurate firing of the guns. This resulted in the first "robot battle" between the German V1 cruise missiles and the M9 Anti-Aircraft gun director (Figure 14).

For all the potential issues with digital control which we will discuss in Section 7, the analog computer program of Figure 13 should convince anyone that digital is the way to go. That being said, even with digital controllers, we still use analog circuits to help us "touch the real world", but these circuits are now signal conduits, not decision engines. The picture of an analog computer patch panel makes the reason for this obvious: it is really hard to program and debug anything complex on an analog computer (Figure 13). Again, what we are all about in designing control systems is teaching a computer to do what a human does, but now we've added the detail that we are using sensors and actuators to intelligently move stuff around.

6. DELVING DEEPER: FEEDBACK LOOPS OCCUR AT MULTIPLE LEVELS

One thing that we all kind of know, but we don't intuitively acknowledge is that feedback loops occur at multiple levels. In many cases, a subsystem of a larger system controlled via feedback has its own feedback loop. We can consider the simple example of a plane flying from California to Maui, shown in Figure 15.

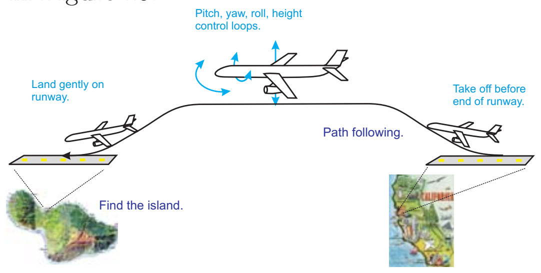
Pitch, yaw, roll, height control loops.
Land gently on runway.
Take off before end of runway.
Path following.
Find the island.

Fig. 15. An example that has loops within a loop.

This diagram is illustrative of the different levels of feedback loops in everyday systems. There is the big path, which is the main mission of the plane.

- Take off from the runway.
- Find the island.
- Land on the runway.

Within the big path, there are small feedback loops that help ensure that the main path can be accomplished, with the added bonus of the passengers and crew being alive at the end of the flight:

- Roll control (don't flip the plane).
- Height control (keep the plane at the correct height).
- Yaw control (keep the plane pointing in the right horizontal direction).
- Internal air pressure control (keep the passengers alive by providing air at 35,000 foot altitude).
- Temperature control (it's cold at 35,0000 foot altitude).
- Other controls: engine, flight control surfaces, etc.

When we look at any complex system, we can see these kinds of loops within loops. From the perspective of the inner loop, the outer loop provides the reference commands and does something with the measured output. From the perspective of the outer loop, the inner loop becomes simply a component with a well defined behavior.

7. DISCRETIZATION

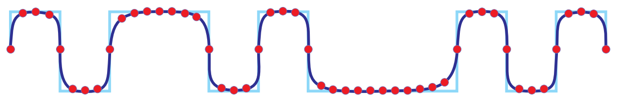

Fig. 16. A pictorial representation of sampling.

One of the fundamental differences between how control systems have been implemented in the past 50 years has been that the implementation of the math to build compensators has been done increasingly via digital computers. Even though this has become almost universal in the past 30 years, dealing with sampled data systems (where the real-time data is sampled at discrete-time intervals) does not usually get taught until the second controls class, one on digital control. Still, it is worth understanding what happens to the data when it has go get into a computer. I'll start with the generic sampling diagram of Figure 16.

Fig. 17. Analog entertainment. On the left is a reel-to-reel tape recorder favored by serious audiophiles. On the right is an old analog color television console.

Fig. 18. Digital entertainment. On the left is a smart phone that holds more music that your grandparents' entire house. On the right is a high definition digital television, with much more resolution, a much larger screen, and a much smaller depth than it's analog ancestor.

The diagram of Figure 12 and the common experience of most of our lives indicates that there is a lot of stuff being done with digital computers, but you really have not been told what is up with discretization. That is, nobody has really told you how the data gets into and out of the computers. Prepare for a crash course in the form of a short paragraph.

Computers can only work in discrete cycles: logic combinations flip from one set of 0s and 1s to another in multiple layers throughout the system. They are interpreted in different ways, but there is always a clock. Furthermore, since there are only 0s and 1s, any real number, anything that might represent the real world can only take on a finite (but possibly huge) set of values. To bring data into a computer then, we need to first convert it into an electronic signal (that's the job of a sensor) and then sample the signal from this sensor at discrete-time intervals. Furthermore, the data can only take on a finite set of values. This is usually accomplished with a circuit called an analog-to-digital converter (ADC). To get data back out, to the real world, one has to do the inverse of this path, eventually putting out an analog signal to an actuator via a digital-to-analog converter (DAC).

If one looks at the true square wave signal in Figure 16, it has beautiful, sharp edges. By the time it has gone through any analog circuitry, those edges get rounded by the "low pass" nature of most things. The rounding might be trivially small, but it is there. Sometimes it is significant. We sample not the perfect signal, but that rounded signal, and then we only get a finite number of values with which to represent it. An 8-bit ADC can generate only $2^8 = 256$ possible values, a 16-bit ADC $2^{16} = 65536$. Finally, relating system models based on differential equations to what ends up inside a computer requires a slightly different set of math. Playing the Devil's Advocate here, this means that digital ("computer") control is bad because:

- The mathematical descriptions are less like the real world, so modeling the real world is harder.
- Signals are approximated by a certain number of bits (Figure 16) and are sampled in time which means that we only look at data every so often and assume it behaves.

Logically, the devices of Figure 18 should never replace those of Figure 17, and yet as you check your smart phone between any two paragraphs in this document, we know this has happened. Why?

Well, while sound quality on perfectly tuned, high grade, analog audio, played from reel to real recording (your grandparents may still have one of these) beats almost any digital music recording, actually getting everything tuned just right, keeping the analog recording medium from getting noisy or distorted, due to degradation over time is really hard. On the other hand, digital methods allow a cookie cutter approach. Bits are 0 or 1 and we don't care if it's a large signal or small so miniaturization happens. Miniaturization leads to less power and more speed for the same math functions. Wiring gets replace by computer programs, and computer programs are a lot easier to implement, debug, and expand than wiring. Copies from one instance of something to another are exact.

All of these things mean that we can do far more accurate and complex things with the "inexact" samples and models of digital methods than we can with the analog methods. We still have to do that conversion back to and from the real world for anything real to happen. This is the difference between something happening in computer graphics and in real life.

8. SENSITIVITY TO DELAY: THE DIFFERENCE BETWEEN SIGNAL PROCESSING AND FEEDBACK CONTROL

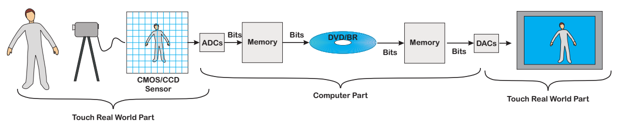

Fig. 19. Discretizing image of human for creating and playing back a DVD/BRD.

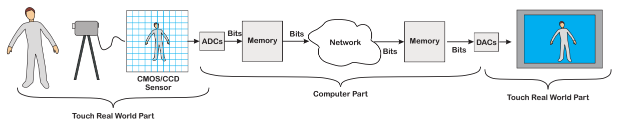

Fig. 20. Discretizing image of human for transmitting across a network.

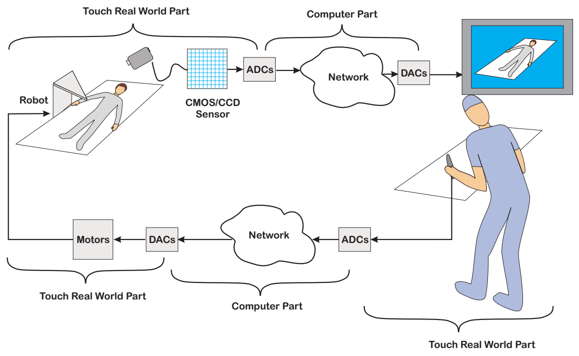

Fig. 21. Remote surgery involves feedback from the images, making latency relevant.

Another useful topic comes out of the whole discussion of discretization and can be easily illustrated using Figures 19 – 21. The top Figure 19 illustrates the process of taking an image or sets of images into an optical disk recording (DVD or BluRay) and playing it back. The important illustration is that the real world images are discretized (using an ADC at a particular sample rate), processed, and stored on the disk. At some arbitrary point later in time, the digital information can be retrieved from the disk, and returned to an analog form (via a DAC) for display on the screen. In the case of a digital monitor, the actual conversion back to analog form is on the screen itself. The individually addressed pixels are integrated by our eyes to produce the analog image that our minds can see.

When we stream these images across a network (replacing the disk, Figure 20) we again have the same discretization and return to analog processes. It's just that the middle part has changed. In either case, there is no great worry about the delay between when a particular image was taken and discretized, and when it shows up on the screen. We don't care whether the delay from the disk to our eyes is 0.1 or 1 second, so long as it is consistent, and while we might want to see some sporting event live, we don't worry whether the screen images are delayed from the stadium on the other end of the world by 1 or 10 seconds, unless our neighbors start cheering before we've even seen the play.

This all changes when those network images are being used in remote surgery (Figure 21). In this example, the surgeon is operating on a patient far away using a remote robot. Now, it becomes very obvious that any delay in communication of those images to the surgeon and in translation of the surgeon's controller movements to the robot must slow the surgeon down in their movements.

This is why we are so concerned with delay in feedback systems. While it is a minor annoyance in the first two examples which would be considered signal processing problems, it becomes one of the key limiting factors when one wishes to close the loop.

9. DO THE MATH

If you've stayed with this so far, congratulations! We are past the "control systems for poets" part of the document, that gave history, context, and a lot of the main ideas of control without doing any of the math. That's great, but you won't be taking a controls class to simply waive your hands and give folks that meditative feeling of control. No, you will be taking a controls class to supposedly help you understand how to actually build a controller for some physical system.

The radical assumption here is that you want to have a clue how to "turn the knobs" when you end up building or buying a PID controller. For that, we need more than to align our chakras: we need to do some freankin' math! Let's not panic here: it's all math you supposedly have had already, but probably until now you never had to put it together into trying to make a system work. The way many controls classes are taught, you still might not have that feeling at the end of the quarter/semester, but I'll try to explain where the different math tools show up and when the different ones are useful.

In teaching these concepts to middle and high school students, we have a major issue at this point, because they haven't yet had the prerequisite math and the math based science classes to make the mental leaps needed to get from the physical concepts above to knowing how to actually get parameter values to execute a good feedback loop. We have to lead them through a few "trust me" moments to turn all of the above into some algebra based root finding (which they've seen and never thought would be useful). You folks are lucky, as you have taken the "how to get from there to here" classes and can see how those pieces align without too many leaps of faith.

The science that you have studied to this point allows you to move from physical representation to mathematical models, many of them will involve differentiation, integration, and differential equations. The math classes will give you some idea how to solve those differential equations. In particular, classes on linear algebra and Laplace Transforms allow us to turn the difficult problem of solving differential equations into a tedious, but much more straightforward problem of using algebra to solve for roots of characteristic polynomials that define the core behavior of the system. Finally, the fact that we can multiply models and signals in the Laplace domain in place of convolving them in the time domain makes it far simpler to get some intuition about complex and interconnected systems.

In the sections that follow, I will go through some examples using some of what you should already know, to give a framework for how one works with feedback control systems.

10. NEWTON AND THE RIGID BODY

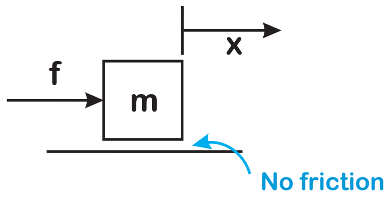

Fig. 22. A rigid mass on a frictionless plane is the starting point for understanding Newton's Law.

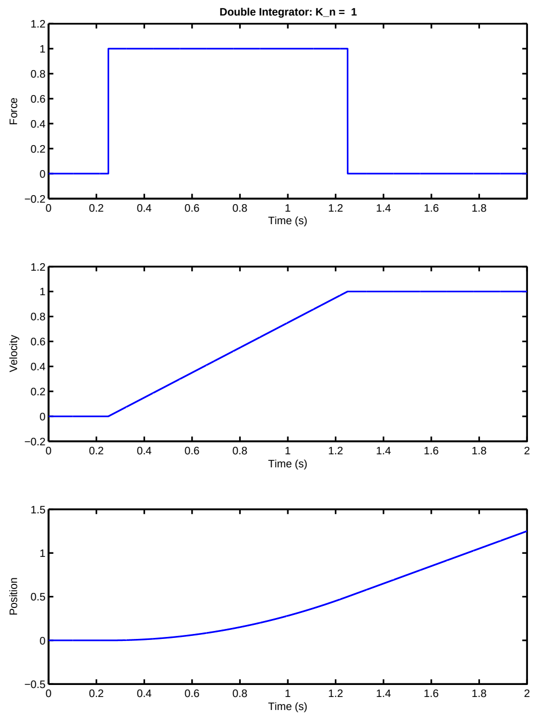

Fig. 23. Response of double integrator to force applied over finite time.

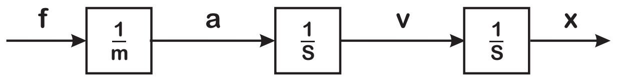

Fig. 24. The block diagram of the double integrator achieved when we ignore the initial conditions and use Laplace Transforms.

By now, almost all of you should have had that physics class where you were initiated into Newton's Second Law: $f = ma$. We can tie this to a simple mass on a frictionless plane, as shown in Figure 22. Remember that acceleration is the second derivative of position, thus we can rewrite $f = ma$ as:

$$a = \ddot{x} = \frac{1}{m} f \longrightarrow v = \int_0^t a dt + v_0 \longrightarrow$$

$$x = \int_0^t v dt + x_0 = \int_0^t \left(\int_0^t a d\tau + v_0 \right) dt + x_0. \qquad (1)$$

The integrals allow us to describe the physical system of Figure 22. Equation 1 has two integrals and so this system is called a double integrator, and it shows up a lot. One can imagine an air hockey puck on an infinite table. This also happens to describe a lot of the the early spacecraft control problems (Franklin et al. (2006); Ogata (1970)) where the spacecraft bodies could be considered rigid bodies that did not have any significant vibrations and were merely pushed around by the different maneuvering rockets. In each case, if one taps the puck or gives a short burst of a maneuvering rocket, the rigid body has a constant velocity. However, if we push it with a constant force, it accelerates as long as the force is being applied. This is displayed in Figure 23. The top plot is the applied force, the middle plot is the velocity of the block, and the bottom plot is the position. We will use these types of plots to explain the behavior of different systems under control both open loop and closed-loop.

Here is where we put together several math subjects from a smattering of the classes you have had. The transform domain allows us to do a set of special integrals (Boyce and DiPrima (1977); Bracewell (1978)) which allow us to transform equations such as Equation 22 into algebra. For continuous-time calculus and differential equations the most prominent transforms are the Laplace Transform and the Fourier Transform. If we pass our differential equation through a Laplace Transform (LT) and ignore the initial conditions, we get:

$$x = \int_0^t \int_0^t (a) dt dt \longleftrightarrow X(s) = \frac{1}{ms^2} F(s)$$

Time Domain (LT) Transform Domain (2)

When we are done, from the transformed math we know that this system is not stable. That is, if we push the block on the frictionless plane, it will go on forever. Once we stop pushing, the velocity remains constant (middle plot), but the position keeps increasing. That being said, it is easy to control. That is, if we apply the same amount of force in the opposite direction for the same amount of time, the block stops. Finally, a lot of systems look like this (assuming we do not look too closely). Even without imagination, this shows up in most spacecraft control problems, since there is no air to generate friction and no spring force of gravity.

The above discussion has shown how we can lead the students along a path that is supported by a lot of knowledge they already have to the concept of a dynamic system and its stability. Not only have we discussed it qualitatively, but we have tied it into the physics of the problem (using an equation that they almost certainly know), and we have discussed some of the useful tools (transforms) that we use to understand the problem. Finally, they have seen an example of this behavior plotted out. With this, they are ready for the next step: adding feedback into this problem.

11. ADDING FEEDBACK TO THE DOUBLE INTEGRATOR

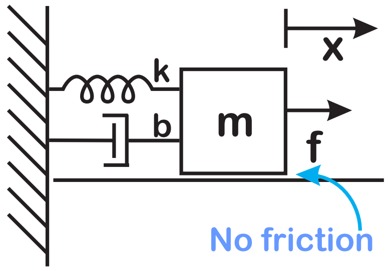

Fig. 25. Adding a spring and a damper to our original mass block.

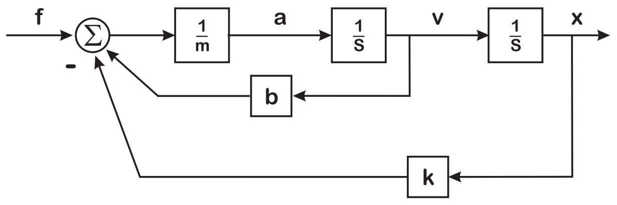

Fig. 26. The block diagram of the double integrator with velocity and position feedback.

One of the great pedagogical things about laying out the double integrator as we have done above is that now we can transform it to a spring-mass-damper system by adding a position feedback (k) and a velocity feedback (b). We are in a position to describe these not only in the picture of Figure 25, but in the block diagram of Figure 26. Equation 1 gets modified to be:

$$m\ddot{x} = f - b\dot{x} - kx \longleftrightarrow \frac{X(s)}{F(s)} = \frac{\frac{1}{m}}{s^2 + \frac{b}{m}s + \frac{k}{m}},$$

Time Domain (LT) Transform Domain (3)

where we see these feedback terms showing up explicitly in the time domain equation and this is transformed on the right into a relationship the transfer function on the right. On the transfer function side, we see that if we set k and b to 0 we are back at our double integrator case. Furthermore, we can relate the second order relationships of k and b and m to those of an oscillatory system by matching coefficients in Equation 4 which relates the spring and damper parameters to that of a simple resonance:

$$\frac{X(s)}{F(s)} = \frac{\frac{1}{m}}{s^2 + \frac{b}{m}s + \frac{k}{m}} = \frac{K\omega_d^2}{s^2 + 2\zeta_d\omega_d s + \omega_d^2}, \qquad (4)$$

where

$$\sqrt{\frac{k}{m}} = \omega_d = 2\pi f_d \Longleftrightarrow f_d = \frac{1}{2\pi}\sqrt{\frac{k}{m}}, \qquad (5)$$

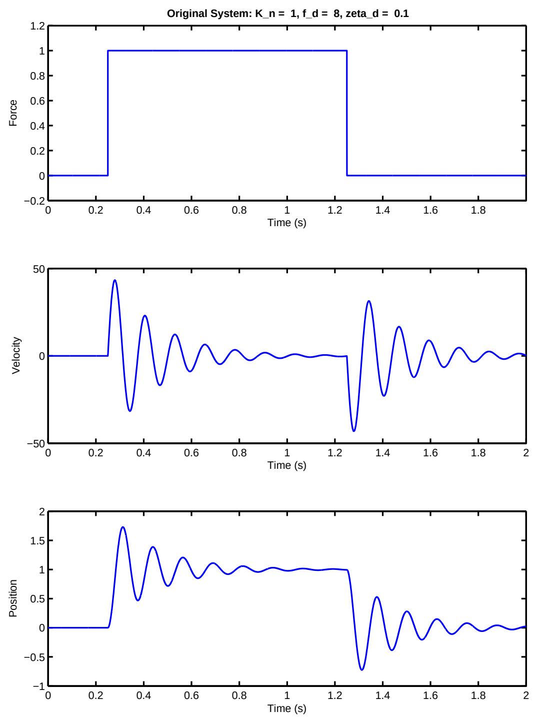

Fig. 27. Response of double integrator with velocity and position feedback to force applied over finite time.

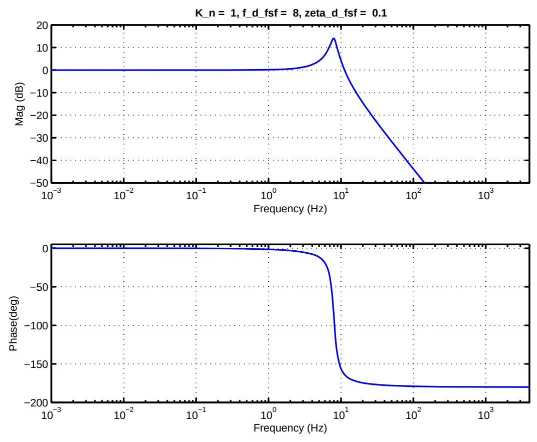

Fig. 28. A Bode plot of the double integrator with position and velocity feedback. Note that we care about both magnitude and phase, and that we use a logarithmic frequency spacing and a logarithmic scale for the magnitude plot.

and

$$\frac{b}{m} = 2\zeta_d\omega_d \Longleftrightarrow \zeta_d = \frac{b}{2\sqrt{km}}. \quad (6)$$

Let's remember that when we try to solve for the roots of a quadratic polynomial and the number under the radical is greater than 0, the roots are real, but if the quantity is less than 0, things get weird. Well, it was weird in high school, but now we know that this produces complex numbers and those numbers are useful in understanding signals and systems.

I've got to jump the gun a bit and go straight to Bode plots without the full explanation (which will show up later). However, if one takes a transfer function such as Equation 4 and evaluates it at positive values of the imaginary axis, i.e. if we set $s = j\omega$ where $0 < \omega < \infty$ and we pick a set of values of $\omega = 2\pi f$ (where f is frequency in Hertz), then we are effectively stimulating the transfer function with a sine wave for infinite time and seeing what happens. What happens is that we get a complex value for each of the frequency points. That is, we get the response of the transfer function at that pure frequencies with no other input. Do this for a set of frequencies that that covers all of the system dynamics we care about and we get a frequency response function (FRF). That is, the value is related to how much the physical system has altered the magnitude and phase of the input sine wave. Okay, so we can plot it out and there are lots of different options, but the Bode plot evaluates the magnitude and the phase of the transfer function response at each frequency and puts them on separate plots. Furthermore, for reasons that I don't want to get into right now, the frequency axis is typically logarithmically spaced. On the magnitude plot, we get a more usable vertical spacing by computing the logarithmic value of the magnitude, in decibels, i.e. $20 \log |H(j\omega|$. So, we plot the magnitude and the phase, and while some like to plot the phase in radians, I'm partial to having a clue and so I like to use degrees. Hey, it's $180/\pi$ but makes things easier to understand.

Folks often confuse transfer functions (TFs) and frequency response functions (FRFs), but the former is a parametric model and the latter is a set of ordered pairs of data: the real frequency and the complex response. Going from TF to FRF is easy (just evaluate at the frequency points), but going the other way is hard.

Let's look at the Bode plot in Figure 28. Many of you will have seen something like this when you looked at specifications for things like headphones (or ear buds). In the audio case, they mostly show you the magnitude plot as time delay and phase don't matter so much in signal processing, but here we have both. With audio equipment, you generally want the response to be flat out to some corner frequency, at which point it falls off. That means that the device reproduces sound out that that frequency without much distortion. In buying ear buds, you would never see the lower plot, which is the phase – the relative angle of output to the input. Again, worrying about this lower plot is one of the main differences between folks who work in signal processing and folks who work in feedback systems (Section 8). The plot relates the response and helps us understand what is going on even without a computer. From a plot like this, we can tell that the roots of that denominator polynomial are complex, and that if the system gets hit with something like a step in force, it will oscillate (ring) back and forth before settling down.

Now, we have gone, in very straightforward steps, from a double integrator system to one with feedback from the outputs of both integrators. We have shown that using some math (differential equations, transform theory, algebra, and root finding) that we can understand or "model" the behavior of this very physical system. Equations 5 and 6 tell us about the behavior:

- k/m tells us how fast it rings.
- b/m (in relationship to k) tells us how long it rings
- Making k bigger means the spring is stiffer, which results in higher frequency ringing.
- Making b bigger relative to k causes the ringing to damp out faster.
- Because denominator polynomial is 2nd order, we can get roots with quadratic equation.
- Any polynomials that are more than 2nd order are a lot harder.

One of the great things about having cool computer tools is that we don't have to just look at one type of plot, so we can take the transfer function above, generate a (discretized) computer version of it, and simulate the time response to something like a step (as we showed earlier). In this case now, we can vary the physical parameters and see what effect it has on the step behavior of our transfer function.

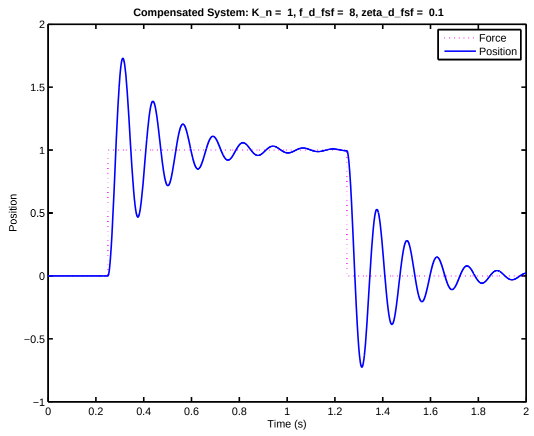

Fig. 29. Double integrator with spring and damper feedback. Resonant frequency (ω_d) at 8 Hz. We show the effect of changing the damping (ζ_d) from 0.1 in this plot, to 0.3 in Figure 30, to 1 in Figure 31.

To illustrate the changes we can make by changing k/m and/or b/m we can show some relatively straightforward plots in Figures 29 – 31. Since we have described the damping and the oscillatory frequency as functions of b, k, and m, we can see how changing their relationship can change the damping and dramatically change the behavior of the system. We have introduced these as properties of the physical system itself. We are trying to understand/model the behavior with these equations and plots to gain insight.

This structure has set us up for the next step: introducing our own augmentation to nature's parameters.

12. INTRODUCING HUMAN AUGMENTATION OF NATURE'S FEEDBACK

In the epically great movie, "Shrek," when Shrek and Donkey emerge from the cornfield, they look up at Duloc

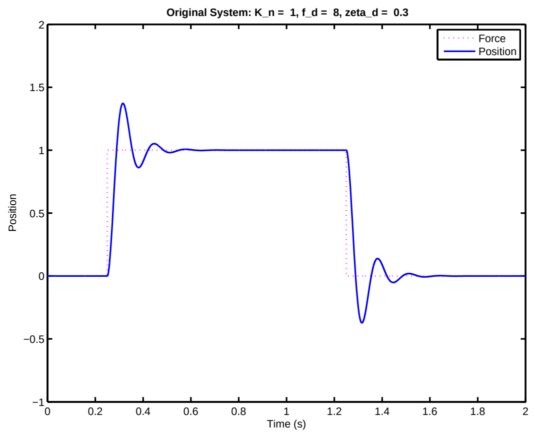

Fig. 30. Double integrator with spring and damper feedback. Resonant frequency (ω_d) at 8 Hz. Damping factor,ζ_d = 0.3.

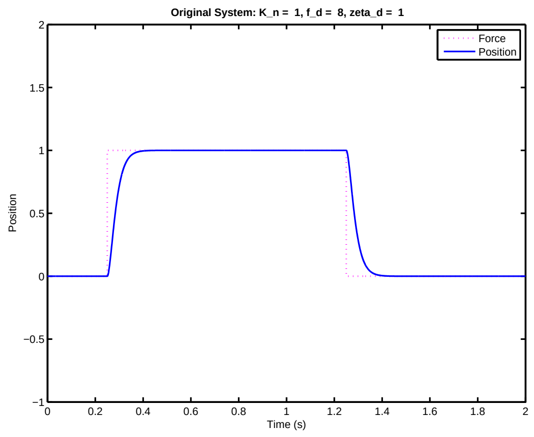

Fig. 31. Double integrator with spring and damper feedback. Resonant frequency (ω_d) at 8 Hz. Damping factor,ζ_d = 1.

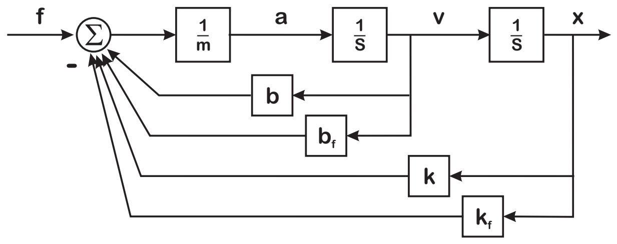

Fig. 32. Adding our own feedback to the spring-mass-damper system.

Tower and Shrek quips, "So,do ya think he's compensatin' for something?" Metaphorically, we've just emerged from the cornfield.

I have shown you the effects of nature's feedback parameters on the behavior of our simple system, and it is natural to ask, "What if nature's k and b are lame? Can we compensate?" The answer is – of course – yes (or we control engineers would all need to find new jobs), but we can use this model to show how we introduce augmented feedback into the system as shown in Figure 32. We can describe this as adding our own signals to feed back the output of each energy collector (the integrator, which we call a state) back into the input of the system. If we do it right, we are – in the words of Shrek – compensating for something. We can analyze this to pick our parameters by looking at modifications of Equations 4–6.

$$\frac{X(s)}{F(s)} = \frac{\frac{1}{m}}{s^2 + \frac{b+b_f}{m}s + \frac{k+k_f}{m}} = \frac{K\omega_d^2}{s^2 + 2\zeta_{df}\omega_{df}s + \omega_{df}^2}, \quad (7)$$

where

$$f_{df} = \frac{1}{2\pi}\sqrt{\frac{k+k_f}{m}} \text{ and } \zeta_{df} = \frac{b+bf}{2\sqrt{km}}. \quad (8)$$

Okay, cool! We're all done. Almost. I have shown you in a very straightforward way that we can augment nature's feedback with our own. The particular form of feedback that I used above involved augmenting every feedback path from every one of the model's energy storage elements (the outputs of the integrator blocks), and this is called "full state feedback", which is the 800 pound gorilla of control.

(You can Google or Bing that. It's a joke from your grandparent's generation that essentially asks, "Where does an 800 pound gorilla sleep?" The answer is, "Anywhere they want.") Now, let's simulate our 800 pound gorilla version of control and see what we get in Figures 33 – 35.

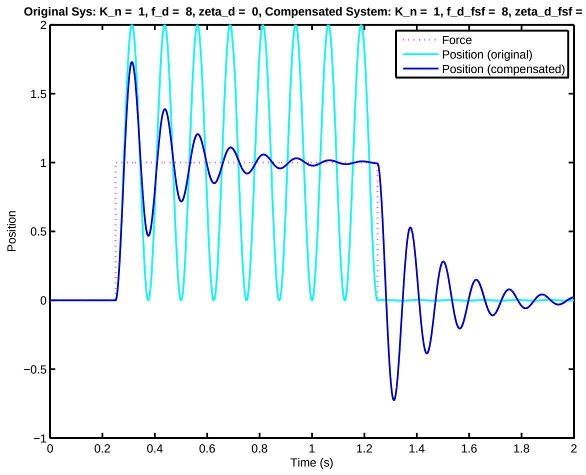

Fig. 33. Our spring mass damper with $\zeta_d = 0$ and $f_d = 8Hz$. The cyan curve shows this response, which rings without stopping. The blue curve shows the response of the system to the same input, when we humans have augmented nature's feedback. We then use our augmented feedback to change ζ_d to 0.1 (here), ζ_d to 0.8 (Figure 34), and even change the frequency, f_d to 20 Hz (Figure 35).

That's cool. We can see that by cleverly choosing our feedback parameters, as shown in Figures 33 – 35, we can dramatically improve the system's behavior. By adding a little bit of damping (on the left) the system rings but eventually settles down. A bit more damping (center) and the system settles without ringing. If we maintain this new damping and artificially increase the stiffness of the spring, we get a higher frequency which means that the system

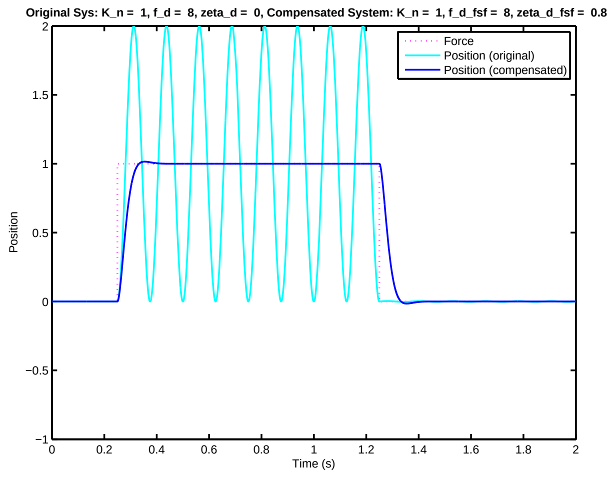

Fig. 34. Our spring mass damper with $\zeta_d = 0$ and $f_d = 8Hz$. The cyan curve shows this response, which rings without stopping. The blue curve shows the response of the system to the same input, when we humans have augmented nature's feedback. Augmented feedback changes ζ_d to 0.8.

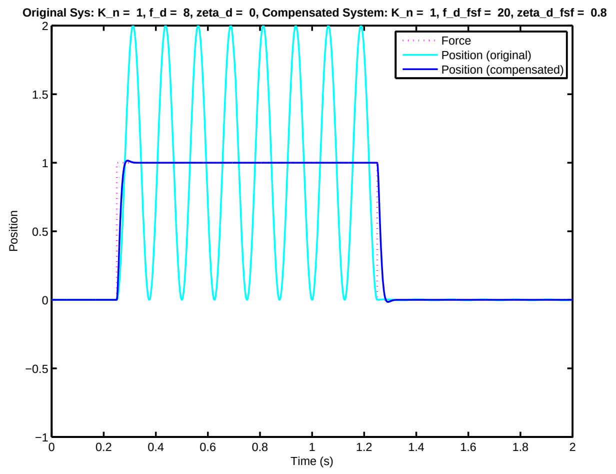

Fig. 35. Our spring mass damper with $\zeta_d = 0$ and $f_d = 8Hz$. The cyan curve shows this response, which rings without stopping. The blue curve shows the response of the system to the same input, when we humans have augmented nature's feedback. WAugmented feedback changes ζ_d to 0.8 and f_d to 20 Hz.

responds even more quickly. This is analogous to cars that can adjust their suspensions to the driving conditions.

Cool, but what happens when we get the feedback parameters wrong? This can be simply illustrated by showing cases where the damping gets set to 0 (on the left of Figure 36) or even negative (on the right of Figure 36). The negative damping results in positive feedback and as I explained earlier: when negative feedback becomes positive feedback, bad things happen – as illustrated by the ever increasing oscillations. Again, this can be tied back to physical examples such as a car swerving erratically because a driver under the influence is compensating for the weaving far too slowly. Another visual analogy is to imagine a tightly spaced

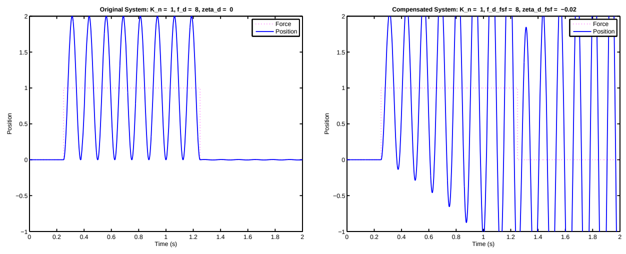

Fig. 36. If we accidentally set $b_f = -b$ (left plot) we zero out the damping of the original system and now it rings forever. If we accidentally set set $b_f = -1.02b$ (right plot) we have negative damping which gives us positive feedback, and this is bad.

margin band trying to maneuver with one band member 180^o out of phase with all the rest. It makes for a funny video.

You might ask yourself: what kind of idiot would use the wrong feedback values? The answer is nobody, assuming they knew what the correct values were. However, if we have the wrong values for our model of the physical world, how can we know what the correct values for our feedback compensation should be? This is the essence of modeling: we need correct parameters to have a good model and if the accuracy of our model is limited, then the accuracy of our control scheme goes with it.

Looking back at what we have done: we have introduced a simple system, shown how nature's feedback affects its behavior, added in our own feedback to improve the behavior, and shown the pitfalls of making a mistake. We have shown modeling, feedback, full state feedback to augment nature's feedback, stability, and instability in a very simple progression. What is left to do? Can you skip that whole quarter/semester of class? No way am I putting myself out of business here, but in fact, no you can't as there are other things that make it not so easy. In other words, you don't always have access to an 800 pound gorilla. More specifically, not all systems are second order and we can't always measure everything. It was nice while it lasted, but to deal with these issues we have to go a bit deeper.

13. FEEDBACK CONTROL WITH FEWER MEASURED OUTPUTS

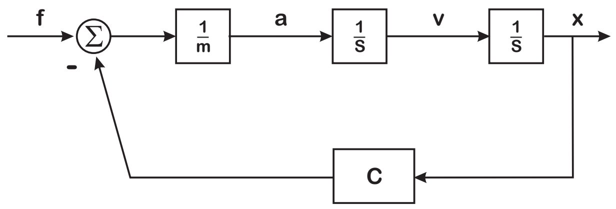

Fig. 37. The double integrator with feedback only from the position state.

So, as I alluded to at the end of Section 12, the problem gets a lot harder when we cannot measure the output of each state. In our current example this might mean that we can measure position but not velocity as shown in Figure 37. What happens if we strap on our Nike's and "Just do it?" The following example illustrates what happens.

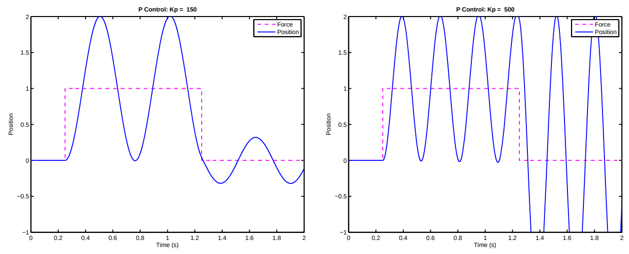

Fig. 38. Our double integrator system when we only feed back position. Here we have position feedback scaling (gains) of 150 (left) and 500 (right).

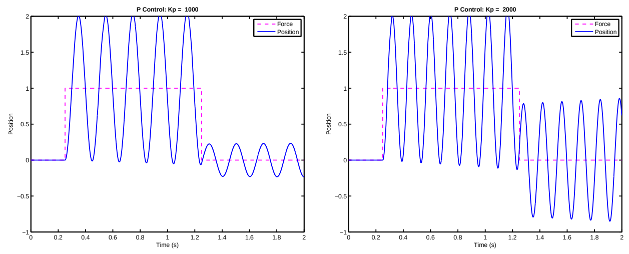

Fig. 39. Our double integrator system when we only feed back position. Here we have position feedback scaling (gains) of 1000 (left) and 2000 (right).

Looking at Figure 38 shows that didn't turn out nearly as well as before. Instead of getting a stable system that settled down (with or without ringing) the thing just seems to ring the whole time. Now, there is a tendency to thing if we simply try harder or "turn it up to eleven", things will work out, so in Figure 39 we increase the amount of feedback from position that we are using. All that happens is we ring faster.

What has happened is that we have run up against a problem that is easy to solve, but not with the simple tools we have discussed so far. What we need are the sets of tools developed over many years to understand control systems. These are the tools on which you will likely spend the bulk of your first controls class, so let's look at them in context.

14. POLYNOMIALS THAT ARE BEYOND QUADRATIC

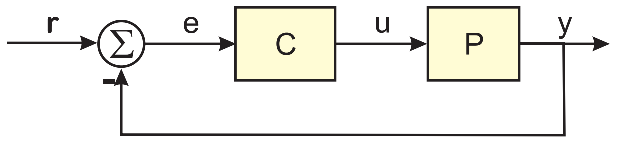

Fig. 40. Simplified abstract control loop. Transitioning from the complex block diagrams to this simple abstraction allows us to explain the uses of Equations 9 and 10.

The curves of Figures 38 and 39 show us that when we only feed back information from position in this particular problem, something goes horribly wrong.

Understanding feedback control is all about understanding the solutions to differential equations that have been modified by a new set of signals based on measurements of the output(s) of the original system. At its core, it is still about what the Hawking quote from Section 2 said: explain what we have seen and help us make predictions about what we will see. The problem is that this is generally hard to do for differential equations that are higher than second order. We can turn them into transfer functions with Laplace Transforms, but we are still trying to find the roots of a modified denominator polynomial, and finding the roots of polynomials gets a lot harder when they are higher than second order.

Looking at the simplified model of Figure 40, we know that in the transform space, we get ratios of polynomials from the models comprising Figure 40:

$$\frac{Y}{R} = \frac{PC}{1+PC} \quad \frac{E}{R} = \frac{1}{1+PC}$$
$$\frac{U}{R} = \frac{C}{1+PC} \quad \frac{Y}{U} = \frac{P}{1+PC}. \tag{9}$$

These transfer functions are all functions of s, i.e. $P = P(s)$, $C = C(s)$, but for readability, we will leave the (s) out of a lot of what follows. All four of these have the same denominator and this denominator governs a lot of behavior. We want to know the roots of the rational equation, where $1 + PC = 0$. Now, we can go into a little bit of detail here in that if we assume both $P = P(s)$ and $C = C(s)$ are ratios of rational polynomials themselves, then $1 + PC$ looks like:

$$1 + PC = 1 + k_0 \frac{(s+b_1)(s+b_2)\cdots(s+b_m)}{(s+a_1)(s+a_2)\cdots(s+a_n)}. \tag{10}$$

Now, the critical points of PC are often very easy, but the critical points of $1 + PC$ are actually pretty hard – especially when one is in the 1940s/1950s and doesn't have a digital computer to help. We know from all the equations in Equation 9 that things get bad when $1 + PC = 0$. Here historical necessity was the mother of invention, and we should thank ourselves that the smart folks who had the following insights did not have computers. These insights are so deep, so useful, provide so much intuition about the control system, that even when we have powerful computers available, we often use them to help us implement these methods that originated before the computer age.

Here, we have defined $P(s)C(s) = PC$ as:

$$PC = k_0 \frac{(s+b_1)(s+b_2)\cdots(s+b_m)}{(s+a_1)(s+a_2)\cdots(s+a_n)}, \tag{11}$$

with an assumption that $n \geq m$, that is the denominator is at least the same order as the numerator. This means that

$$\frac{PC}{1+PC} = \frac{k_0 \frac{(s+b_1)(s+b_2)\cdots(s+b_m)}{(s+a_1)(s+a_2)\cdots(s+a_n)}}{1 + k_0 \frac{(s+b_1)(s+b_2)\cdots(s+b_m)}{(s+a_1)(s+a_2)\cdots(s+a_n)}} \tag{12}$$

$$\frac{PC}{1+PC} = \frac{k_0(s+b_1)(s+b_2)\cdots(s+b_m)}{(s+a_1)\cdots(s+a_n) + k_0(s+b_1)\cdots(s+b_m)} \tag{13}$$

We can make some interesting observations here. We see that the closed-loop poles (the roots of the denominator) will be different from the open-loop poles, but the closed-loop zeros (the roots of the numerator) will stay exactly where they were in the open-loop system.

14.1 Routh-Hurwitz

The first insight into this is to figure out what we can say about the closed-loop poles, the denominator of Equation 13. In the age of modern computing one can multiply out all the polynomial terms and evaluate it numerically, or even symbolically, but before there were computers, this was harder. Routh and Hurwitz were two mathematicians who independently came up with a similar criterion that now bears both their names Franklin et al. (2006); Ogata (1970). It involves multiplying out all the terms in the denominator of Equation 13 and then checking the coefficients. There are things one can say about the coefficients which give away if the roots can be unstable. These involve checking for missing coefficients or whether there is a sign change between any two coefficients.

This will likely be one of the first methods taught and it can be useful, especially when one can multiply out the coefficients symbolically. This usually means a relatively low order polynomial, say less than 10th order or so. It does allow one to say if the roots of the denominator polynomial are stable. As s is a complex frequency variable, the real part of the root in the s domain has to be negative to ensure that the time domain exponential corresponding to it is decaying.

It has limitations, in that it doesn't really tell us much about how to do design. It doesn't really tell us anything about margins. Worse yet, it requires us to multiply out the denominator polynomial. Finally, it tells us nothing about the interaction with the zeros of the system. It's more of a go/no-go type evaluation.

14.2 And now for your moment of Zen: Solving without Solving

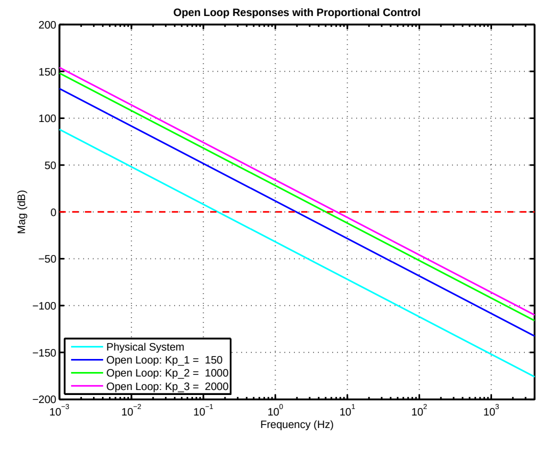

Fig. 41. Bode magnitude plots of the different levels of gain show that all we did was change the level of the sloped line, but did not alter the shape.

A long time ago, someone had the insight that this means:

$$1 + PC = 0 \Leftrightarrow PC = -1 \Leftrightarrow$$
$$\|PC\| = 1 \;\&\; \angle PC = -180^\circ. \tag{14}$$

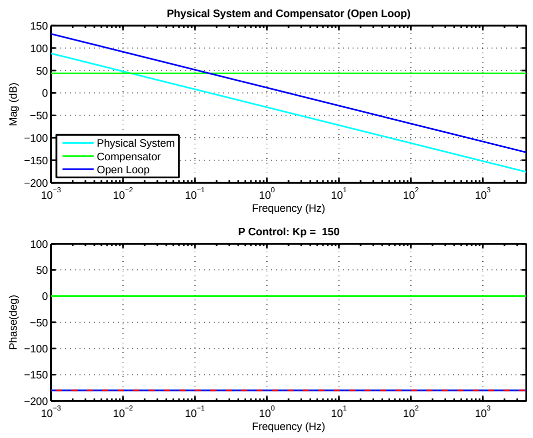

Fig. 42. Again, we use a Bode plot. Here, our compensator gain is 150.

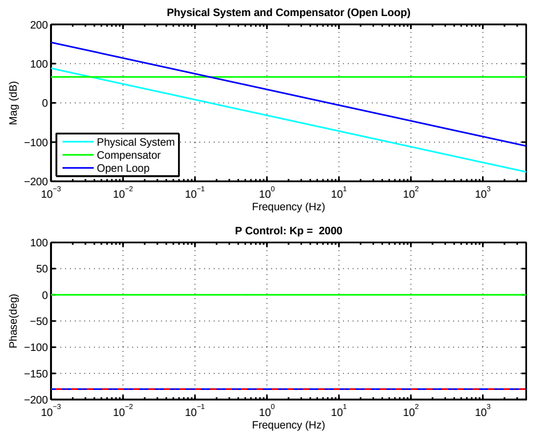

Fig. 43. Again, we use a Bode plot. Here, our compensator gain is boosted to 2000. However, as neither this gain nor the one in Figure 42 has changed the phase, we eventually have a ringing of the system.

The great thing about this (and it was applied in several ways) was that we did not have to solve for the solutions of $1 + PC = 0$ but instead could look at the magnitude and phase of PC, which was relatively easy to do by hand.

This means that if we can check the magnitude and phase against each other, we can be careful that the magnitude should be less than 1 before the phase gets to $-180°$. This explains our fascination with Bode plots. The plots of Figures 42 and 43 show that no matter how big we make the gain, we have the problem that the phase is always at $-180°$ and so when the gain gets to 1 (which is 0 dB on this logarithmic plot), it will ring. It will ring at different frequencies depending upon our gain, but it will still ring.

Okay, so we have analyzed why the position only feedback on the double integrator was not stable. In Section 15 we'll get into how to get around this problem, but it's worth taking a moment to discuss the entire area of Bode plots, which are part of what is called "frequency domain" analysis. As I mentioned when I jumped the gun on Bode plots in Section 11, we get to the frequency domain by taking a transfer function such as Equation 4 and evaluating it at positive values of the imaginary axis. That is, we set $s = j\omega$ where $0 < \omega < \infty$ and we pick a set of values of $\omega = 2\pi f$ (where f is frequency in Hertz), then we are effectively stimulating the transfer function with a sine wave for infinite time and seeing what happens.

What happens is that we get a complex value for each of the frequency points. That is the value is related to how much the physical system has altered the magnitude and phase of the input sine wave. Okay, so we can plot it out and there are lots of different options, but the Bode plot evaluates the magnitude and the phase of the transfer function response at each frequency and puts them on separate plots.

The frequency axis is typically logarithmically spaced because when we pass open-loop poles and zeros, the result in a corner point and a lot of close to straight lines that make the plot easier to interpret. On the magnitude plot, we get a more usable vertical spacing by computing the logarithmic value of the magnitude, in decibels, i.e. $20 \log |H(j\omega|$. So, we plot the magnitude and the phase, and while some like to plot the phase in radians, I'm partial to having a clue and so I like to use degrees. Hey, it's $180/\pi$ but makes things easier to understand.

Although it's not discussed this way in the textbooks, the frequency domain is to the transform domain as time simulation plots are to time domain equations. We evaluate the models at different frequency or time points, respectively.

The most complete version of frequency domain insight one can get is from Nyquist plots and the Nyquist Criterion (Franklin et al. (2006); Ogata (1970)). It takes into account poles and zeros, and we can work from the open-loop quantities without multiplying out the denominator polynomial. It predicts the behavior of the system beyond a stability go/no-go. Furthermore, it gives margins for error, how far we are from $|PC| = 1$ when $\angle PC = -180°$ (called gain margin) and how far we are from $\angle PC = -180°$ when $|PC| = 1$ (called phase margin).

Unfortunately, Nyquist plots are a bit hard to visualize. Bode found a way to simplify our understanding by unwrapping the plot of Nyquist so that magnitude and phase were still plotted relative to $j\omega$, but they were separated from each other. Bode plots are not quite as general, but the visualization is so much easier to see that for all intents and purposes, folks use Bode plots.

One of the main drawbacks of the frequency domain is not in using it for analyzing models but in making measurements. A lot of systems, in particular systems with slower dynamics, such as chemical process control (CPC) are not well suited for frequency domain measurements.

The other way of saying something about the closed-loop poles working just from the open-loop polynomials is what is called the Evans Root-Locus (after its inventor) or simply Root-Locus. Again, we work with the poles and zeros and mark those on a graph that is made on the complex s plane. Poles are marked with $\times$ while zeros are marked with $\circ$. The use of the realization about the roots is that the locus of roots will move from the open-loop poles (when $k_0 = 0$) to the open-loop zeros (when k_0 is very large). (See Equation 13.) The Root-Locus method says that as we move that all the poles should be at an angle of $-180°$ and so we can sketch out the path (or locus) that the closed-loop roots take by measuring angles from all the poles and zeros and making sure they sum to -180. Evans even created a combination ruler and protractor called a Spirule that we all had to buy to participate in class. There were rules about asymptotic behavior that allowed you to quickly decide where the toots would end up in different situations. It was very visual. These days, it's all done simply in a computer; no need to draw. (Some of us still have a Spirule in the desk just to wave it at young engineers when we shout at them to "Get off our lawn.")

The path of the roots can be very insightful, but again this says nothing about the path of the zeros and nothing about margins. Furthermore, we only do this evaluation with all the controller components fixed except for the k_0 term that is being varied. It shows itself as being of a mentality of analog circuitry, in which a fixed controller would get a fixed control design, with the gain being the only value that was modified easily.

In summary, Routh-Hurwitz can give us a go/no-go decision on loop stability, root-lccus can show us where all the closed-loop roots go as we vary the feedback gain, and frequency response plots and tell us about stability, performance, and margins of our system. None of them require a solution to the differential equation or finding the actual roots of the closed-loop denominator polynomial. You should expect to have to solve problems using all three of these in any first controls class.

In summary, solutions to LTI differential equations can be composed of a sum of (possibly complex) exponentials.

- **If the differential equation has real coefficients, then any complex exponentials come in complex pairs, with the same real part and complimentary imaginary parts. Those can be rewritten as a sine or cosine with an damping factor.**
- **If all of those exponentials have negative real parts, then the equation has a stable solution.**
- **Feedback provides an opportunity to change the solution exponentials, to go from the original set with hopefully better characteristics.**
- **In the Laplace Transform domain, negative real exponentials correspond to roots of the transfer function denominator in the laft half of the s-plane.**
- **If the LTI differential equations are sampled at a fixed rate, the sequence of sampled exponentials become geometric progressions and negative real exponensts correspond to a ratio, r_i, of each of the progressions with $|r_i| < 1$. That is, in discrete time, the left half of the Laplace s-plane should map to the inside of the unit circle on its discrete counterpart, the z-plane. In other words, the discrete transfer function has all roots inside $|z| < 1$.**
- **Delay in time maps to a unit gain complex exponential in the s-plane whose only effect is to provide negative phase. This is only a limitation when the delayed signal is used in feedback.**

15. FAKING MEASUREMENTS: ESTIMATION

Okay, with all that extra insight into math tools, what about our problem. We were doing fine with full-state feedback, but take away the velocity measurement and our system is no longer stable. How to we fake our way into something that looks like a measurement of velocity?

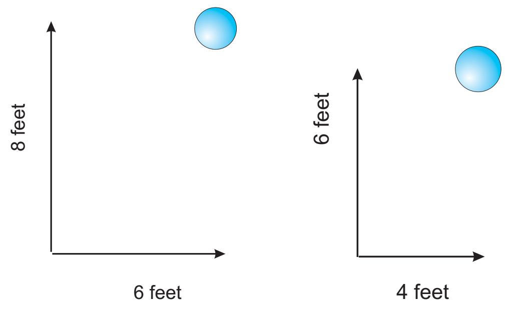

Fig. 44. Two images of a ball in the air with only position information for each of them. On their own, there is not enough information to catch a moving ball from these images. A time stamp is needed on each.

In order to fix (or compensate for) our lack of velocity measurement, we know that we need extra information. This is pretty obvious to the students. While control theorists understand that we need some sort of estimate of velocity (or derivative information), this is hard to explain to students who have not yet computed a derivative. I have found that the example of Figure 44 works spectacularly well, since most of us have thrown and caught a ball. The figure represents two images of balls in the air with only position information. Having been through the prior material, we can all conclude that it is impossible from only that information for anyone to catch the ball. We then produce a soft ball and toss it to one of the students who almost always catches it. We then get a chance to explain the contradiction, that if the speaker is not a nonsensical liar, something else must have taken place.

That something can be explained as follows. Our eyes take repeated still images (position information). Our brains take the difference and add a time stamp. Our brains (meat computers) are estimating the velocity of the ball from changes in position over time. In fact, for many North American kids who have played baseball or softball, we can relate this to learning to play catch, how as our skill level improved, we could catch balls thrown much less accurately and with greater speed. As your brain learned to produce better estimates over time, you could throw and catch much better, with far greater speed on the ball. For those of you from other parts of the world, similar analogies can be made with soccer (football), tennis, etc. The message here is that we need to teach our machine how to estimate the velocity.

Our system that fed back everything had no issues because it was getting velocity information. Can we somehow

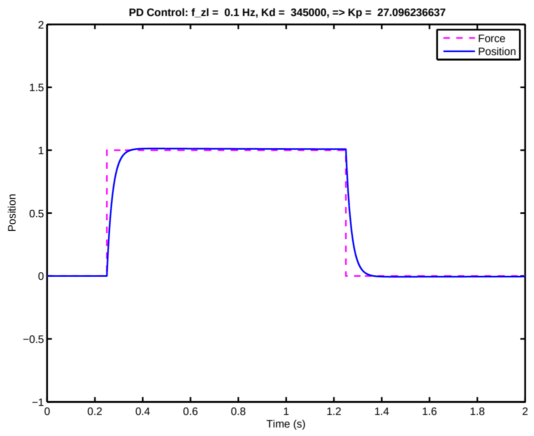

Fig. 45. Our double integrator system when we only feed back position information. Here we have used position and the derivative of position to make the feedback effective and stop the ringing.

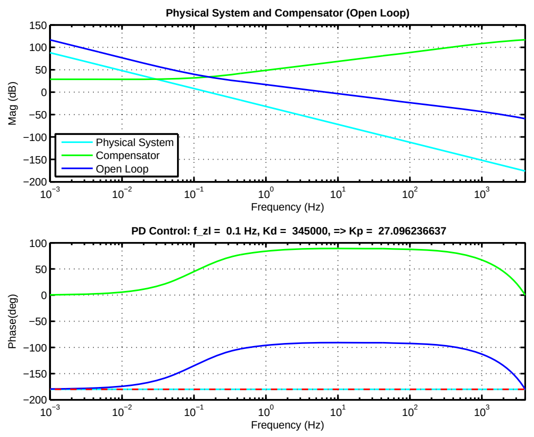

Fig. 46. Our double integrator system when we only feed back position information. Here we have used position and the derivative of position to make the feedback effective and stop the ringing. We see that the Bode plot helped us predict that this would happen and how to choose the relative and overall gains.

estimate velocity from our position measurement, and to do this, we need to differentiate, that is see how quickly our position is changing over time. To replace our direct measurement of velocity, we need some sort of estimate of velocity based upon stuff we are measuring – in this case position. We know from differential calculus that differentiating position will give us velocity, but the engineering know how part of this method also tells us that differentiating also can amplify noise, so we really want to limit our differentiation to what we need to maximize the tradeoff between getting information and not amplifying noise. A transfer function of a pure differentiator is:

$$D_1(s) = s. \tag{15}$$

Furthermore, there are controllers called proportional plus derivative (PD) controllers that use this:

$$C_{PD,1}(s) = K_P + K_D T_D s. \tag{16}$$

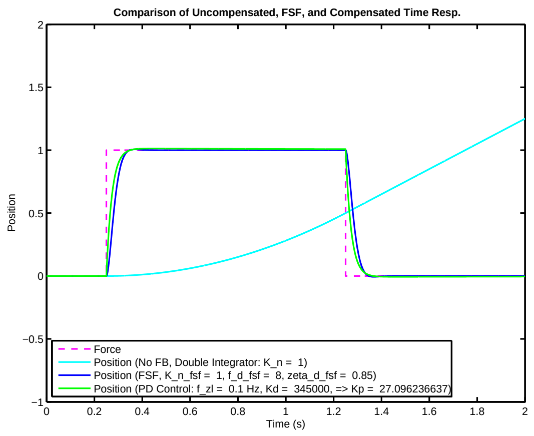

Fig. 47. Here, we repeat the plot on the left of Figure 45, because we have curves from full state feedback as well. Thus, we are showing that using some signals to estimate others can give us almost as good a result (sometimes) as when we can measure everything.

Here we've included the differentiation time, T_D in the equation as a separate part of the differentiator gain. We will see why this is useful in Section 20. Practically speaking, an unfiltered differentiator does not make sense since it implies infinitely high response at infinite frequency (that's called the Big Bang). We console ourselves in one of three ways:

1) The actual physical system and any circuits that implement the PD controller will naturally roll off at higher frequencies, even if it's not reflected in Equation 16.

2) We can always add a low pass filter to the differentiator term or to the entire compensator, clearing up this problem.

3) If we discretize this with a backwards rectangular rule (Section 19) the problem goes away in discrete time. This is very useful for making a digital computer implementation of a PID controller (Section 20).

Considering the second option, we can apply a low pass filter with DC gain (when $s = 0$) of 1 to the differentiator on its own:

$$C_{PD,2}(s) = K_P + K_D T_D s \left(\frac{a}{s+a}\right) \tag{17}$$

$$= \frac{K_P(s+a) + K_D T_D s a}{s+a}, \tag{18}$$

$$= \frac{(K_P + K_D T_D a)s + K_P a}{s+a}, \tag{19}$$

$$= (K_P + K_D T_D a)\left[\frac{s + \frac{K_P a}{K_P + K_D T_D a}}{s+a}\right], \tag{20}$$

or we can apply it to the whole PD compensator:

$$C_{PD,3}(s) = [K_P + K_D T_D]\left(\frac{a}{s+a}\right) \tag{21}$$

$$= \frac{K_P a + K_D T_D s a}{s+a}, \tag{22}$$

$$= (K_D T_D a) \left[\frac{s + \frac{K_P a}{K_D T_D a}}{s + a} \right], \qquad (23)$$

We can see that $C_{PD,2}$ and $C_{PD,3}$ have the same structure and one can be turned into the other by how we choose the values of K_P and K_D. They relate to a more practical implementation of derivative circuits over a finite frequency interval that looks like:

$$D_2(s) = \frac{(s + b_f)}{(s + a_f)} = \frac{(s + 2\pi f_b)}{(s + 2\pi f_a)}, \quad 0 \leq f_b < f_a, \qquad (24)$$

Because the zero of $D(s)$ is at a lower frequency than the pole, and thus it will amplify signals in the range between f_b and f_a. However, unlike an ideal differentiator, it won't continue to amplify, but instead will flatten out at frequencies above f_a.

Thus, the generic PD controller ends up being what is called a lead circuit in the texts.

$$C_{lead}(s) = k_f \frac{(s + b_f)}{(s + a_f)}, \quad 0 \leq b_f < a_f, \qquad (25)$$

and point out two important things: that there is a set of algebra from our transform space that allows us to predict what math we need to do to estimate velocity and that this math is often very simple. We can also point out that a lot of very useful control systems have simple guts such as this.

When we do this, we get the response in Figure 45, which looks pretty good. On the right, we see that our Bode plot shows some differences, but the main thing that we show them is that the phase is well above $-180°$ when the gain gets to 1 (0 dB). In fact, we can splurge a bit and show them Figure 47 which has the results of both the full state feedback and our PD feedback and we are showing them that with a good estimate, we can do almost as well as when we measure everything.

Again, it is worthwhile to look at the path we have taken with a set of bright students who have not yet studied calculus. We have shown them that we can do something with a limited set of signals to produce estimates of the signals we need to cause the system to behave the way we want. We have shown them that some math that they don't yet have leads to some math that they already know and that the math they already know tells us how to fix the problem, sometimes simply. We have attached physical meaning to the equations and examples that we use, so that even if they don't understand the details, they have a very strong sense that they get the basic idea.

16. SOME DEPTH ON ESTIMATION

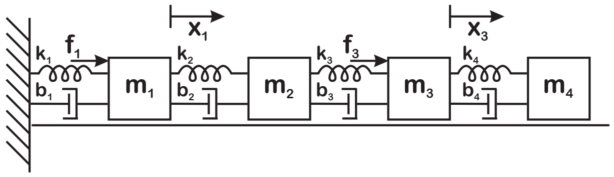

Fig. 48. Extending our spring mass damper system to one with a lot more springs, masses, and dampers.

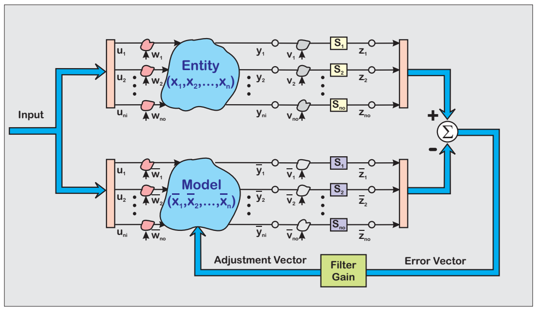

Fig. 49. A generalized view of model based filtering

While we have taken the students a long way in a short time with the previous discussion, there are some extensions and abstractions that are useful because they tie what we have been discussing to problems that they hear about every day. We start with the example of Figure 48. Here we have extended the simple physical example. We can then ask:

1) What if we can't measure every "state"?
2) What if the physical system is more complicated than our model?

The trick is that since (2) is always true, then (1) is always true. This kind of modeling allows us to discuss really big, big systems with lots of stuff that we can't model exactly and can't measure fully. Power grid, air traffic control, automated highways, systems biology, have many more things (states) than can be measured. However, the world is increasingly comfortable with computers, with immersive video games, with simulations. This means that we can refer to Figure 49 as a general metaphor for how we handle these problems. The general process steps are:

- Build a simulation.
- Run simulation in parallel with real world.
- Compare simulation output to things we can measure in real world.
- Correct simulation with measurements from the real world.

In doing so, we get a lot of advantages, including that the "inside" of our simulation will have useful information we could not measure in the real world. Plus, we get to tell our friends that we work with models.

Model-based simulations corrected periodically by measurement data is an estimator, and this is really the basis for actual model-based methods – often called state-space methods. With a good model and clean, frequent measurements, an estimator can tell us a lot more about what is going on inside a physical system than we could from a transfer function (which is really an input-output relationship). The key issues are in getting clean measurements and in getting an accurate model. This is often the place where state-space methods fail in practice, but it does not have to be (Abramovitch (2015b)).

17. BEING AWARE OF DELAY

The telerobotic surgery example of of Section 8 gave us an intuitive feel for how time delay changes the control problem: the more delay is in our loop, the more slowly we have to move. Where does this show up in the math?

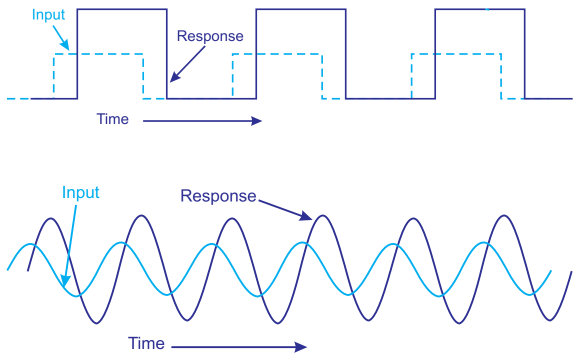

Fig. 50. Simple examples of pure time delay in a system. The output reproduces the input, with perhaps some scaling, but no other distortion.

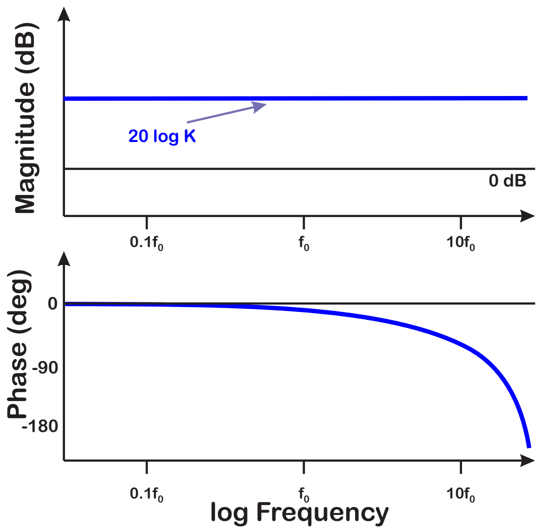

Fig. 51. A sketch of the Bode plot delay produces: flat gain ($20logK$) and phase going from 0 to increasingly negative with high frequency.

We can also have a model that is a pure time delay, that is, the output reproduces the input but delayed T_D seconds. The transfer function for this is:

$$\frac{X(s)}{F(s)} = Ke^{-sT_D}, \tag{26}$$

and it has a gain of K for all frequencies, but an ever decreasing phase. Signals passing through lines without attenuation, or simply data passing through a computer system exhibits this kind of behavior. Two examples of such signals are shown in Figure 50. Figure 51 sketched the resulting Bode plot for the pure delay with gain K.

When delay is part of a system – and it always is – it will result in negative phase at higher and higher frequencies. At some point, just the delay along takes us past -180° of phase and we are done, but even at lower frequencies, delay erodes our phase margin (how much phase is left when the open-loop magnitude gets below 1). Thus, delay always limits how fast we can go. To go faster, we must minimize delay in the feedback loop. Without a time machine, there is no way around this.

One more thing to keep in the back of your mind: it is hard to put the transfer function of Equation 26 into a root locus and I don't remember it being something that could be handled with Routh-Hurwitz analysis. The Bode plot of Figure 51 makes it kind of obvious, but sometimes folks want a transfer function. The most common fix for that is a Padé Approximation (Franklin et al. (1998)). Let's just show a first order Padé approximation to the transform of a delay of length T_D:

$$e^{-aT_Ds} \approx \frac{1 - \frac{1}{2}aT_Ds}{1 + \frac{1}{2}aT_Ds}. \tag{27}$$

As with Taylor series approximations, we can always add more terms, but this gets the main idea across. The Padé approximation of Equation 27 does give us a rational transfer function for the delay, but it has a zero in the right half of the s plane. While these kinds of systems aren't as hard to control as if the system were unstable (if there was a pole in the right half plane), the presence of this non-minimum-phase (NMP) zero makes control a lot harder. So, this is just another way of saying that time delay limits how fast we can go in a feedback loop.

18. A LITTLE BIT MORE MATH: INTEGRATORS AND THE FINAL VALUE THEOREM

The Final Value Theorem (FVT) relates the limit time value of some response to the zero-frequency, DC response of a transform. For control systems, it's main use is to tell us how many integrators we need in the forward path of a feedback loop in order to achieve 0 steady state error to some level of input (step, ramp, quadratic, etc.). In continuous time, the theorem says if a function $f(t)$ is bounded for $t \in (0, \infty)$ and the $\lim_{t\to\infty} f(t)$ is bounded, then

$$\lim_{t\to\infty} f(t) = \lim_{s\to 0} sF(s), \tag{28}$$

where $F(s)$ is the Laplace transform of $f(t)$. The transfer function from the reference to the error signal is:

$$\frac{E(s)}{R(s)} = \frac{1}{1 + P(s)C(s)} \tag{29}$$

and if the input, $r(t)$ is a unit step function, its Laplace transform is $1/s$. This means that for a step:

$$E(s) = \frac{1}{s}\frac{1}{1 + P(s)C(s)}. \tag{30}$$

Let's call $G(s) = P(s)C(s)$. The final value of this is given by

$$\lim_{s\to 0} sE(s) = \lim_{s\to 0} \frac{1}{1 + G(s)}. \tag{31}$$

Now, if the open loop transfer function has an integrator in it, then $G(s) = \tilde{G}(s)/s$ and

$$\lim_{s\to 0} \frac{1}{1 + G(s)} = \lim_{s\to 0} \frac{s}{s + \tilde{G}(s)} = 0. \tag{32}$$

So, that's it: if $P(s)C(s)$ has an integrator in it, then the error to a step eventually goes to 0. For this reason, many controllers add an integrator into their transfer function so as to guarantee zero steady state error to a step input. An extremely common simple control structure includes both proportional and integral feedback with different gains, i.e.

$$C_{PI}(s) = K_P + \frac{K_I}{T_I s}\frac{K_I}{T_I s} = K_P \left(\frac{s + \frac{K_I}{K_P T_I}}{s} \right). \tag{33}$$

This is known as a Proportional plus Integral (PI) controller, and it is one of the most common controllers in industry. It works on a variety of low order system models where the first resonance or pole frequency is far enough out that we can apply integral action to get zero steady state error, but mostly we rely on the proportional feedback to stabilize the system. When we need some differentiation for stability and still want the integrator for zero steady state error, we turn to that workhorse, the proportional-plus-integral-plus-derivative (PID) controller, which we will talk about briefly in Section 20.

19. SOME MORE ON DISCRETIZATION

A stable differential equation has roots of the characteristic polynomial that all have negative real parts. The simplest, first order, linear, constant coefficient differential equation,

$$\dot{x} = ax, \tag{34}$$

is stable when $Re\{a\} < 0$. Now, if we sample the time axis at a constant rate, say $f_S = 1/T_S$, we have a spacing between points of T_S and if we took the values at those points, they would end up as a geometric progression with a ratio. Call this ratio α. The essence of understanding discretization of physical signals is to relate quantities such as α to a and T_S. What we know is that for the differential equation to be stable, $Re\{a\} < 0$. For the sequence of pulses to decay, we also know $|\alpha| < 1$. Thus, somehow these are related. We can understand this by looking at the transfer function of a low pass filter:

$$L(s) = K\frac{a}{s+a}. \tag{35}$$

This has a low pass (when $s \ll a$) gain of K, with a corner frequency at $s = a = 2\pi f_c$.

There are lots of ways to generate a discrete transfer functions so that we can do this math on a computer. One method that we remember from numerical solutions of integration is to use one of the simple integration approximation rules. For example in the Trapezoidal Rule integration, where we average the heights of the two points and multiply by the distance between to get the area of the interval. In that respect, the differentiation operator of a Laplace Transform, s, is substituted by:

$$s \longleftarrow \frac{2}{T}\frac{z-1}{z+z} = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}. \tag{36}$$

Making this substitution, we end up with

$$L_T(z) = K\frac{\beta_T(z+1)}{z-\alpha_T} = K\frac{\beta_T(1+z^{-1})}{1-\alpha_T z^{-1}}. \tag{37}$$

where $\beta_T = \frac{aT}{2+aT}$ and $\alpha_T = \frac{2-aT}{2+aT}$. This filter is stable when $|\alpha_T| < 1$, which is true whenever $a > 0$, so our digital version has the same stability conditions as the continuous-time version.

For a variety of reasons, folks sometimes use another integration rule, the forward rectangular integration rule:

$$s \longleftarrow \frac{z-1}{T} = \frac{1-z^{-1}}{Tz^{-1}}. \tag{38}$$

$$L_F(z) = K\frac{\beta_F}{z-\alpha_F} = K\frac{\beta_F z^{-1}}{1-\alpha_F z^{-1}}. \tag{39}$$

where $\beta_F = a$ and $\alpha_F = 1 - aT$. This filter is stable when $|\alpha_F| < 1$, which is true whenever $0 < aT < 2$. Now, $T > 0$, so we are really placing conditions on a, not only that it be greater than 0 as before, but there is also a requirement that the sample period, T, be small enough in relation to a so as to keep the combination below 2. This is a requirement on how fast we need to sample, just to keep the digital filter stable, and is a direct consequence of how we chose to represent the continuous time quantities inside a computer (discrete time).

This is the main lesson of this section: when one samples the data, an entirely new set of issues comes up related to how fast we sample, how we represent the physical system, and how we convert the data. It doesn't mean that it's not worth doing; it usually is. It simply means that we have to be aware of those new issues.

One more simple discretization is worth mentioning here, for the simple reason that it shows up in generating the discrete-time version of a PID controller (Section 20). As this latter device is what most of you will end up doing if and when you actually implement a control system, it's worth mentioning. The rule is the backwards rectangular rule, where we integrate by looking backwards from the latest sample. (The forward rule looks forward from the next to last sample.)

$$s \longleftarrow \frac{z-1}{Tz} = \frac{1-z^{-1}}{T}. \tag{40}$$

$$L_B(z) = K\frac{\beta_B(z)}{z-\alpha_B} = K\frac{\beta_B}{1-\alpha_B z^{-1}}. \tag{41}$$

where $\beta_B = \frac{aT}{1+aT}$ and $\alpha_B = \frac{1}{1+aT}$. This filter is stable when $|\alpha_B| < 1$, which is true whenever $aT > 0$ or $aT < -2$.

Now, this is weird, because the backwards rectangular rule discretization of our filter is stable not only when the continuous-time filter is stable, but even in some cases when it isn't. We refer to this as the backwards rule being an overly conservative discretization. Nevertheless, it shows up as the main way of discretizing continuous-time PID controllers. Even if we are not delving into digital control much here, this is a useful point of reference. I'll discuss that briefly in Section 20.

Finally, there are many other ways to discretize a continuous-time model. None of them are perfect, because by looking at the data only at discrete points separated in time, we have to throw away some information. Intuition tells us that if those points are close enough together, we will be okay. In fact, the Nyquist Sampling Rate (Bracewell (1978); Franklin et al. (1998)) is based on a way of establishing a minimum sampling rate based on the continuous-time model. By far the most common discretization method used in textbooks (and in Matlab) is the Zero-Order Hold Equivalent (Franklin et al. (1998)), which has the property of being exact at the sample points. It also has the property that it destroys all physical intuition that might be in the continuous-time model as part of the discretization, so there's that.

This section described a bit about discretization, even though you likely will not see it in your first controls class. The reason for this is that when you go to generate your first control system, you will almost certainly be faced with discretization as you implement the controller on a computer. Even some mild awareness of some of the issues here is better than the "What just happened?" feeling most students are left with when they are asked to do control without knowing anything about discrete-time representations of sampled data systems.

20. THE GOOD, THE BAD, AND THE PID

It's a plot that repeats itself almost as often as people shaking their heads at one of the newer *Star Wars* movies: you go through 90% of a class on controls and at the end there is a brief discussion of proportional-plus-integral-plus-derivative (or proportional-integral-derivative, PID) controllers. Then you look at almost any real world control system and the odds are you are looking at a PID. They probably deserve more respect than they will get in your class.

If we think about the control schemes discussed above, we had full-state-feedback, where we did proportional feedback from every energy storage element (state) of the system. When we couldn't measure all the states, we used a differentiator or limited differentiator (lead filter) as an estimate of at least the velocity. That gave us the proportional-plus-derivative (PD) controller (Section 15). Finally, the Final Value Theorem convinced us that if we wanted zero steady state error to a step input, we wanted an integrator in the loop. That gave us the proportional-plus-integral (PI) controller (Section 18).

It should be noted that many of our examples had physical systems models that were second order or less. This was convenient, but it should not be a total shock: engineers often beat systems into a form so that the dominant characteristics that the control system needs to deal with (at least at first pass) is a second order system. With that in mind, what happens when we feel we are most likely to have some second order physical system and we may need PD and/or PI control? What if we had one controller that had enough knobs to turn on or off any of those characteristics?

Enter the PID controller. For a lot of the logic described above, PID controllers dominate the industrial landscape. They are either built in-house or bought as add on controllers. Ironically, while PID controllers are the most standard controller used in practice, the PIDs themselves suffer from a lack of standardization. Virtually any control structure that has a proportional gain, an integrator with its own gain, and a differentiator (with or without filtering) with its own gain can be considered a PID controller, but the specification of gains can have a multitude of styles.

Consider the following three PID controller equations. The center term for each is the typical "three parameter" form, while the rightmost term puts the PID in the form of an analog filter (which we would use for analysis). For brevity, we have omitted the cases when a low pass filter is added to the differentiator term that we showed in Section 15, although this would be necessary if the PID were to be discretized with a Trapezoidal Rule Equivalent (Abramovitch (2015c)). Still, these three forms are close to the style of specification of most continuous-time, unfiltered PID controllers that may be encountered, either in analysis or in commercial controllers. Recognizing one of these forms, we should be able to easily translate the control parameters into one of the other forms. This is extremely useful in getting an apples to apples comparison of different commercial PID controllers.

$$C(s) = K\left(1+\frac{1}{T_I s}+T_D s\right) = K\left(\frac{1+T_I s+T_I T_D s^2}{T_I s}\right) \tag{42}$$

$$C(s) = K_P + \frac{K_I}{s} + K_D s = \frac{K_I + K_P s + K_D s^2}{s} \tag{43}$$

$$\begin{aligned} C(s) &= K_P + \frac{K_I}{T_I s} + K_D T_D s \\ &= \frac{K_I + K_P T_I s + K_D T_I T_D s^2}{T_I s} \end{aligned} \tag{44}$$

The form of Equation 42 is typically – but not always – associated with process control applications, temperature, and pressure control. There is an overall controller gain, but the relative gains of the three parts are adjusted via the terms T_I and T_D, which nominally are meant to refer to the integration time (time over which the integral takes place) and differentiation time (time over which the derivative takes place), respectively. Even here, the terminology is confusing, since as written the integral and differentiator take place over all time. These terms take the place of the integrator and differentiator gains. Still, they are a bit confusing since one would naturally assume that integration and differentiation times would be characteristics of the device or a measurement parameter, rather than a control gain.

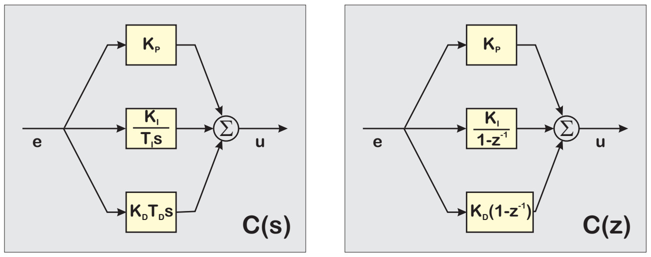

Fig. 52. A parallel form topology of a simple analog PID controller (top). The digital PID controller (bottom) shows much of the same structure.

The form of Equation 43 has three independent PID "knobs" with no hint of relation to the integration and differentiation time. The form of Equation 44 breaks the integration and differentiation times out from the integrator and differentiator gains. While this last form may seem a bit tedious due to the extra coefficients, it has a distinct advantage when discretized using a backwards rectangular rule equivalent where $s \longrightarrow \frac{z-1}{T_S z}$. If one sets $T_S = T_I = T_D$, which aligns with integration and differentiation over a single sample period, we get:

$$C_B(z) = K_P + \frac{K_I}{1-z^{-1}} + K_D(1-z^{-1}) = \frac{(K_P+K_I+K_D)z^2 - (2K_D+K_P)z + K_D}{(z-1)z} \quad (45)$$

What is most revealing about this form is the similarity between the structures of Equations 44 and 45 as demonstrated in Figure 52. The continuous and discrete parameters are related via $T_I = T_D = T_S$, while the structure remains essentially the same. Furthermore, the backwards rectangular rule discretization has taken a non-proper analog PID form and made it proper and therefore directly implementable.

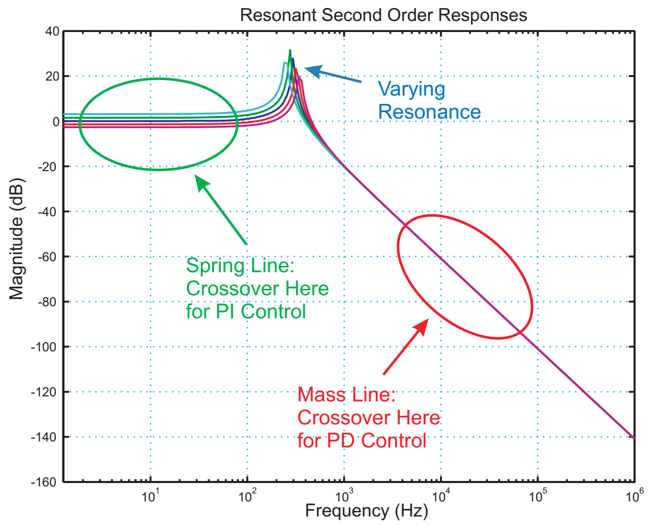

Fig. 53. The typical ranges in a mechatronic system and how they relate to PID control.

While the denominator of Equation 45 does not change with PID parameters, the numerator can produce anything from a lag filter (set $K_D = 0$) to a lead filter (set $K_I = 0$) to a notch filter (Abramovitch et al. (2008); Abramovitch (2015c)). Figure 53 shows the ranges of a typical mechatronic control system. If the loop is closed well below the resonance, a PI controller may be used. If the loop is closed far above the resonance, a PD controller may be employed. It is when we try to close the loop in the vicinity of the resonance that we need to carefully use all the PID coefficients, and so we need a far more accurate model than the previous two cases.

These types of insights move the PID from something separate from the rest of control design to simply being a particularly useful substructure of control design. It ties advanced methods to that workhorse of implementation, the PID, even if that starts simply with improved modeling of our system so that we can obtain better PID parameters.

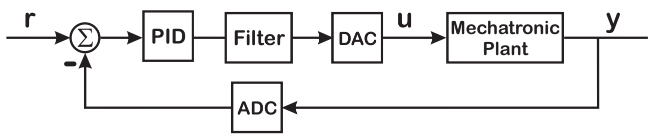

Fig. 54. A practical digital control loop for a mechatronic system. The digital controller is often implemented as a PID like controller in series with filtering to lower the effect of high frequency resonances.

Another bridge between PID controllers and advanced methods is to realize that almost any linear SISO state space controller can be formulated as a PID plus a group of matched filters as shown in Figure 54. Looking at it this way, the matched filters correspond to the estimator model and neither of them works well if the modeling is inaccurate. Generally, for systems with more than second order dynamics, successful modeling involves a deep dive into measurements.

A few things stand out differently from much of what you will learn in your first class and this disconnect is often very confusing, like an M. Night Shyamalan movie script. You are stuck shaking your head wondering, "What the heck just happened?!?"

The first is that most of the controls class you will learn will be about developing transfer functions for controllers that then result in some plotted response that looks reasonable. Furthermore, that transfer function has a single gain knob to adjust. With a PID controller we end up with three parallel mini-controller blocks, each with their own gain.

The second is that when we are taught to do computer control by discretizing the system model and then creating an all digital controller (Franklin et al. (1998)). However, with the PID, we take an analog controller structure and discretize that.

The third thing is that almost all controllers used in industry are PID controllers. The estimates are easily 90% or more. Thus, it is worth understanding their relationship to other forms of control if we want to do something more intelligent than randomly turn three knobs (the PID gains).

But my main point is that if you really do control on real systems, you will work with PIDs at some point (perhaps often). They are not a separate thing, but a particularly easy-to-use, robust, and flexible structure for implementing controllers. They are not a seperate art and should not be treated as such. In dealing with these, Yoda said it best: "Use what you have learned: serve you it can."

21. CLOSING COMMENTS

With such a whirlwind tour of control and system theory, I hope that I have given you not only some context for why control and the encompassing field of system theory is important, but also what to expect when you first learn it. I assumed that you had certain math and science classes under your belt. While not mandatory, it does make the description above a lot easier to handle. These are likely the prerequisites for your control class anyway, so this shouldn't come as a big shock to any of you.

Another prerequisite that I assumed was that you had some experience with programming as a means of implementing algorithms. As most control systems one will see involve a computer to execute the control decisions, this is kind of a basic necessity in the modern world. One caveat is that computer science is great, but has no concept of time constants. It is thus up to someone with some idea of these things to teach programmers how to make their code interact with the real world, to keep up with nature's clock.

With that, you can expect that your controls class will go through a lot of the topics described above in far greater mathematical detail. That's great. I hope that the preview of and context for the different tools allows you to see where they all fit in. I wish someone had told me those things back when I was your age.

There is a lot more to control, signal processing, and system theory than can be understood in just one class, and certainly in one primer/cheat sheet document like this. Every problem has its practical limits, but for different subbranches of control, they show up differently. For example, chemical process control systems are large and complex, but the model orders are relatively low and the time constants are absurdly slow, so one would not be concerned with the computer being able to keep up with the needed sampling rate. On the other end of the spectrum, small mechatronic systems often have multiple resonances that oscillate in the tens or hundreds of kiloHertz. Even the fastest processing chips have issues doing complex operations between samples coming in at a 50 MHz rate. On the other hand, chemical process control systems are often limited by extreme nonlinearities and limitations on what can be injected as a test signal and what can be measured. To paraphrase Sun Tzu (Tzu (1983)), "Know your time constants and know your model, and you can close 100 loops without disaster."

It is best to take a step back and give them an overview here, some thoughts about control systems and tech work in general:

- A lot of this discussion shows up in all technical work. Ideas about making measurements, using science to generate models, transforming those models into mathematics, using those math models to make predictions and improve design, and implementing things using computer programming are fairly universal..
- Computation has gone digital, not because digital gives better performance, but because digital gives cookie cutter, which gives miniaturization which gives more capability in small spaces . . . everywhere.
- To do useful things that touch the real world, you have to understand the real world and this usually takes modeling, and modeling takes math. (This answers the high school students inevitable questions in math class: "Are we ever going to use this?")
- Some folks do math for its own sake. They are called mathematicians. Scientists do math to understand the world. Engineers do math to do something about it. The lines often cross, but one doesn't have to prove theorems all day to find math that makes a lot of things understandable and allows us to build better solutions to problems.

I'm not saying learning about control systems is easy, just that with some guidance and context it can be a lot less inscrutable than it has been since I was a student. Why is it worth learning about feedback and control (apart from the class being required to graduate)? Well, as we move to a world of more and more automated devices interacting with each other and with humans and the environment, the physical need for measurement and feedback is obvious. Without the basic understanding of the principles and practices that give understanding and intuition in these areas. most folks are "practicing control without a license." It doesn't take a science fiction writer to imagine what kind of chaos machines that aren't properly regulated will cause.

Instead, a little bit of control systems knowledge goes a long way. It goes beyond the theory to an understanding that there are certain limits that we invariably hit and one should be aware of these before one blindly uses the numbers out of an optimization algorithm.

REFERENCES

Abramovitch, D. (2005). The outrigger: A prehistoric feedback mechanism. *The IEEE Control Systems Magazine*, 25(4), 57–72.

Abramovitch, D.Y. (2015a). Built-in stepped-sine measurements for digital control systems. In *Proceedings of the 2015 Multi-Conference on Systems and Control*, 145–150. IEEE, IEEE, Sydney, Australia.

Abramovitch, D.Y. (2015b). Trying to keep it real: 25 years of trying to get the stuff I learned in grad school to work on mechatronic systems. In *Proceedings of the 2015 Multi-Conference on Systems and Control*, 223–250. IEEE, IEEE, Sydney, Australia.

Abramovitch, D.Y. (2015c). A unified framework for analog and digital PID controllers. In *Proceedings of the 2015 Multi-Conference on Systems and Control*, 1492–1497. IEEE, IEEE, Sydney, Australia.

Abramovitch, D.Y., Hoen, S., and Workman, R. (2008). Semi-automatic tuning of PID gains for atomic force microscopes. In *Proceedings of the 2008 American Control Conference*. AACC, IEEE, Seattle, WA.

Åström, K.J. and Wittenmark, B. (1990). *Computer Controlled Systems, Theory and Design*. Prentice Hall, Englewood Cliffs, N.J. 07632, second edition.

Barnett, J.E. (1998). *Time's Pendulum: From Sundials to Atomic Clocks, the Fascinating History of Timekeeping and How Our Discoveries Changed the World*. Harcourt Brace & Co, San Diego, New York, London. ISBN: 0-15-600649-9.

Bernstein, D.S. (2002). Feedback control: An invisible thread in the history of technology. *IEEE Control Systems Magazine*, 22(2), 53–68.

Boyce, W.E. and DiPrima, R.C. (1977). *Elementary Differential Equations and Boundary Value Problems*. John Wiley & Sons, New York, third edition.

Bracewell, R.N. (1978). *The Fourier Transform and Its Applications*. McGraw-Hill, New York, 2 edition.

Franklin, G.F., Powell, J.D., and Emami-Naeini, A. (2006). *Feedback Control of Dynamic Systems*. Prentice Hall, Upper Saddle River, New Jersey, fifth edition.

Franklin, G.F., Powell, J.D., and Workman, M.L. (1998). *Digital Control of Dynamic Systems*. Addison Wesley Longman, Menlo Park, California, third edition.

Haidar, A. (2016). The artificial pancreas: How closed-loop control is revolutionizing diabetes. *IEEE Control Systems Magazine*, 28–47.

Hawking, S. (1988). *A Brief History of Time*. Bantam Dell Publishing Group, New York, NY. ISBN: 978-0-553-10953-5.

Jespersen, J. and Fitz-Randolph, J. (1999). *From Sundials to Atomic Clocks: Understanding Time and Frequency*. Dover Publications, Inc., Mineola, New York, second revised edition. ISBN: 0-486-40913-9.

Maxwell, J.C. (1868). On governors. *Proceedings of the Royal Society of London*, 16, 270–283.

Mayr, O. (1970). *The Origins of Feedback Control*. MIT Press, Cambridge, MA.

Mindell, D.A. (1995a). Antiaircraft fire control and the development of integrated systems at Sperry, 1925-1940. *IEEE Control Systems Magazine*, 15(2), 108–113.

Mindell, D.A. (1995b). Automation's finest hour: Bell Labs and automatic control in World War II. *IEEE Control Systems Magazine*, 15(6), 72–78.

Mindell, D.A. (2002). *Between Human and Machine: Feedback, Control, and Computing before Cybernetics*. Johns Hopkins Studies in the History of Technology. The Johns Hopkins University Press, Baltimore and London.

Ogata, K. (1970). *Modern Control Engineering*. Prentice-Hall Instrumentation and Controls Series. Prentice-Hall, Englewood Cliffs, New Jersey, third edition.

Stein, G. (1989). Respect the unstable. Bode Lecture presented at the 1989 IEEE Conference on Decision and Control, Tampa FL.

Stein, G. (2003). Respect the unstable. *IEEE Control Systems Magazine*, 23(4), 12–25.

Strang, G. (1980). *Linear Algebra and its Applications*. Academic Press, New York, second edition.

Tzu, S. (1983). *The Art of War: Edited and with a Forward by James Clavell*. Dell Publishing, New York. ISBN 0-385-29985-0.

22. WHAT ELSE?

So, we're done, right? If we've mastered the stuff above in our first controls class, we know the whole field. Not so fast, Kemo sahbee. If you you understand everything from the first class, you have the tools to become functional with feedback control, but we've only scratched the surface. The stuff outlined above that you should learn in that first controls class will get you going, but there are a ton of other things to consider if you want to go deeper.

Resonances: I haven't really gone into it, but not all systems can be effectively modeled by a low order transfer function. In fact, as we know from being around anything physical, things vibrate. The vibration modes, which look like combinations of second order underdamped poles and zeros are difficult to control because of the combination of peaks and notches in the frequency response magnitude accompanied by ± 180° phase changes. From a root locus perspective, these are pole/zero pairs that are near the instability line of the $j\omega$ axis. From the Bode plot perspective, we see potential for the open loop gain to pop back up above 1 (or 0 dB) when the phase is below 180°. It's all a matter of how fast these dynamic features are relative to the speed at which we want to control. Most PI controllers will limit their bandwidth to about 1/4 the frequency of the first resonance. Controlling through resonances and notches (anti-resonances) involves having a really, really good model of those features (Abramovitch et al. (2008); Abramovitch (2015b)). That can't be done without a lot of precise measurements (Abramovitch (2015a)) and modeling.

Digital control: Also called computer control (Åström and Wittenmark (1990)), discrete-time control (Franklin et al. (1998)), etc. While I gave some indications of the issues in digital control in Sections 7 and 19, that's barely scratching the surface. Careful consideration of what it means to sample real-time signals, the limitations and the needs for doing so, and the design modifications that should be in place when we really want to properly take advantage of computer control are design disciplines on their own (read another semester at least). What does the Nyquist Sampling Theorem (Bracewell (1978)) tell us about sampling and is it's admonition to sample twice as fast as things are changing really enough? What are the ramifications for delay of sampled systems? How do we move between the continuous-time physical models of the real world and the discrete-time models in the computer without losing all intuition for what is going on?

State Space: Your instructor may or may not bring this up in your last week of class, but if you remember the 800 pound gorilla metaphor, you've already seen part of this. State space, also known as model based methods, revolve around taking the linear algebra way of solving differential equations. You turn an n^{th} order equation into a system of n 1^{st} order equations that fit conveniently into matrix algebra (Strang (1980)). Rather than being limited to the input-output relationship of transfer functions, we now get the states which are all the internals of the model. If we could directly measure them all, we'd have the best control.

The issue is that since we can't measure them all, we have to estimate them. I touched on this in Section 15. Both the estimation and control depend upon having a good model of the system, and a reality based model of the system requires a crapload of measurement and model fitting. That part often gets ignored, but the math of state space is beautiful. Theoretically, you can do a lot more, so it is a big focus of academic methods.

Modeling/Identification: I've alluded to it quite a bit in this document, but almost all of control system work depends upon having some sort of model if the physical system. Often the quality of the model determines the quality of the controller. This is particularly true when we are trying to represent all the internal states of a system, such as with model based control. **You can't control what you can't model and you can't model what you don't measure.**

Measurements: Speaking of which, what kinds of measurements can we make on our systems, whether to get a model or to simply monitor performance? Where can we put a sensor and how does that sensor works? Can we put in our own test signal (what they would call training data in machine learning (ML)) or are we stick with only operational data? How do we wire our measurements so that the data goes right into our analysis and design tools?

On-line and self tuning: Wouldn't it be great if we could attach a controller to any physical system turn it on and have it self adjust itself to get the best performance as a control system? It seems great, but there are limitations. You wouldn't want to do that right before you launched that rocket. You might not want to do that on that nuclear reactor, that airplane, that automated car. Self-tuning and learning tend to be slow processes so as long as the thing to which they are being applied it well behaved on its own, they can produce some impressive results. However, for things that are hard to control, this idealization leads to disaster. We need some sort of nominal model and controller in operation to keep these systems stable while the learning algorithm learns the appropriate model. In other words, something has to buy time for the learning algorithm on hard to control systems.

There is a whole branch of study on this and at its heart are the origins of what is now called machine learning (ML) which has now become all about neural networks, which are tuned by the same core algorithm that cleans up the performance of our phones and televisions.

Noise and disturbances: Speaking of all that modeling and tuning, one of the main limits is that measurements are never perfect; they are inherently noisy. How noisy is noisy and what limits does this put on our ability to measure for modeling or apply self tuning or simply operate the control system?

Feedforward: For many years the forgotten step-child of the control world, feedforward control has made a resurgence in the past 30 years. The term, feedforward, is a applied to many different control schemes, but most broadly, it can be considered as being any control scheme that uses a signal not directly extracted from a measurement of the signal being regulated. In other words, the signal that drives that part of the controller is not a feedback signal. Because of this, feedforward control doesn't need to worry about destabilizing a system via feedforward. A well designed feedforward system can inject the reference signal in a way as to minimize the feedback error, leaving the feedback controller to mostly deal with disturbances. Other forms of feedforward can use a calibrated auxiliary sensor signal to anticipate an precompensate for disturbances. The list is pretty long, but it's important to realize that most feedforward systems are really combinations of a core feedback system augmented with a feedforward component.

Time Varying: We got a lot of mileage assuming that the differential that described the physical system did not vary with time, but that's often not true and it makes the math a lot harder. It is very hard to use any transfer function math (they kind of depend on that LTI thing) and even the time domain solutions are hard. Grad school, anyone?

Nonlinearities: Yeah, the other part of not being LTI is not being linear, i.e. the differential equation being nonlinear. Or having nonlinear components in the loop such as limits on how large a signal can be (saturation) or quantization in converting signals between the analog and digital worlds. The question really isn't whether there is nonlinearity, but how much nonlinearity there is and how much it affects what we are trying to do.

Integrator wind-up: One of the main nonlinearities is saturation and this is particularly an issue when the actuator limits and our controller things it still has errors to kill. Even in a PID, the integrator can continue to accumulate errors that the actuator cannot do anything about so that when the actual error gets small enough not to saturate the actuator, we still have this large container of old, useless, integrated error to deal with. This is called integrator wind-up and there are anti-windup schemes to limit the behavior of the integrator when we hit saturation.

Multi-Input, Multi-Output: We have focused on control problems where we had a single input to the physical system and a single measured output (single-input, single-output; SISO). The fact of the matter is that most real systems have lots of potential inputs and outputs. These multi-input, multi-output (MIMO) systems are far more complex, as is the math to deal with them. This is one of the great promises of the state-space methods above: at least from a mathematical structure point of view, they seem to easily adapt from SISO problems to MIMO problems.

23. GLOSSARY

ADC: An analog-to-digital converter. A circuit which takes an analog signal level (usually in the form of a voltage) and converts it into a computer readable signal (represented by a collection of bits). ADCs are characterized by their precision (the number of bits, M, which allows for up to 2^M distinct voltage levels to be identified and thus a resolution of up to $R/2^M$ volts, where R is the input voltage range. They are also characterized by how long they take to do the conversion and what circuitry is used to actually do this. The choice of circuitry affects the speed, accuracy, and cost.

Control: Make something move where you want. That's pretty much it: control is about making stuff move in a way that you want it to move. Originally, we would have been talking about moving physical objects around, but the idea of moving stuff can refer to electricity, chemicals, data, etc. Still, when we talk about moving something in a way that we want it to move, we are talking about control.

DAC: An digital-to-analog converter. A circuit which takes a digital signal in some number of bits and produces a voltage level.

Feedback: Look at where it's going as you push it and adjust how you are pushing. Ah, when we aren't familiar with the recipe, or we are working in a new kitchen with new tools, we taste the food a lot to see if it's cooking the way we want it to. This is feedback: make a sample of the output of what we are doing and compare it to where we want to be moving to. Again, this can be feedback of signals moving inside a computer as much as down a street.

Feedforward: Estimate (guess) how to push it but never use where you see it going to adjust how you are pushing. If we think about how we often think about doing things for which we are very familiar, we do it in a feedforward way. When we are cooking something that we have cooked a hundred times and we know the recipe cold, we just execute it without having to taste the food in intermediate steps. That's feedforward and when we know things really well, it's a very efficient way to do things.

LTI: Linear, time-invariant. It's a property of certain systems (or maybe an idealized pair of properties) in which the the differential equations describing a system are both linear (double the input and the output doubles) and time-invariant (the parameters are fixed in time). While the term linear is prevalent, the sophisticated kids use the term affine, which means there can be an offset that does not double (is fixed). In other words, truly linear functions pass through $(0, 0)$ but affine functions don't need to do that.

MIMO: A system with a multiple inputs (MI) and a multiple output (MO), hence making it a multi-input, multi-output (MIMO) system. By convention, the number of inputs and outputs is usually counted on the physical system itself, not on the controller. When life is good, we can decouple MIMO systems into a set of individual SISO systems that don't interact, but as one might guess, nature is usually not so benign. MIMO systems require another level of control design because the interconnection between the different axes of the system means that good things we do on one loop may adversely affect the behavior of another loop.

Sample and Hold: An analog circuit that samples a voltage and then holds that voltage level at its output for a specified amount of time. These are typically used in conjuction with ADCs to hold the signal at a given value so that the ADC can do its work of converting the analog signal to a digital value. Most hold circuits in the literature are zero-order holds, which means that they hold the sampled value flat until the next sample instant. A first-order hold would produce an output signal that was a line (first order polynomial) between the last two sample points (which would add some derivative information at the expense of some delay).

Sampling: Refers to looking at a signal at discrete-time points. Usually, but not always, the separation between points in time is a constant, denoted T or T_S. This sample period is the inverse of the sample frequency, f_S, so $f_S = 1/T_S$. Generally, the faster one samples, the better the representation of reality, however, faster sampling also means less time between samples for the computer to do the necessary calculations. For many problems these days, the overwhelming speed of microprocessors compared to the physical problem time constants makes this a non-issue. However, for high speed applications, this becomes critical.

SISO: A system with a single input (SI) and a single output (SO), hence making it a single input, single output (SISO) system. By convention, the number of inputs and outputs is usually counted on the physical system itself, not on the controller.

Transform: Usually refers to an integral (continuous-time) or sum (discrete-time) in which a signal (usually in time or space) is transformed into an alternate representation usually with some sort of complex frequency significance. In continuous time, the most commonly used transforms are the Laplace Transform (frequency variable is the complex s) and Fourier Transform (frequency variable is the imaginary $j\omega$. In discrete time, the most commonly used transform is the Z-transform (frequency variable is z). Transform methods have the advantage of turning convolution integrals/sums in one domain into simple multiplications in the other domain.