PROFESSOR: Hello, and welcome. Last week we started to think about programming. Programming was the first module in this class. And it was important for two different reasons. First, we're going to use programming throughout the term in the study of all the different things that we do. So it's important that you learn to program now, just so you can use that tool.

More importantly perhaps, we didn't just learn how to program, we focused on how to program in a fashion that would let us construct complicated systems out of a simpler systems. This is the way that we can manage complexity. This is the only possible way that we can make complicated systems.

So that was the more important intellectual theme from the first part, where we introduced our mantra, PCAP -- primitives, means of combination, abstraction and identifying patterns. That's the key to this modular approach to hierarchical kind of design.

Today what I want to do is start the second major theme. First theme was the design of complex systems, we saw that by reference to programming. Today we're going to start thinking about modeling and controlling physical systems.

The idea here is not so much how you construct systems-- but we will get back to that. The idea is to characterize the systems that you've constructed and say something about their metrics as being positive or negative. So what we want to do is, in fact, focus on behavior.

So to illustrate that, I'll start with an example. This is an example that you did in design lab last week, or for some of you, yesterday. The idea was to program the robot so that it could sense the distance to a wall.

Represented two ways here, sort of the view that you would get from Soar, and a more schematic representation showing the position of a robot, the position of a wall. The idea is that you can sense the position to the wall using the sonars. You know where you would like to be, because some user told you. You'd like to be say, half a meter away from the wall. And your job was to write a program that moves the robot from where it is to where you'd like it to be.

So here's the kind of behavior we might have liked-- so I'll do that again. So we might have liked that if you started here, you have a nice smooth progression up to where you'd like to be. Very graceful, ballet type, and you just sort of smoothly glide into the position that you'd like.

Some of you probably achieved that behavior, and some of you probably did other things. So that might be the intended behavior. One way to achieve the intended behavior is to use what we call a proportional controller. In a proportional controller, you make the command be, in some way, proportionate to the intended response. So imagine this code, which might establish a class for finding the wall.

So the important thing-- as we saw, for all state machines, the important thing is to define a start state and a getNextValues routine. What this getNextValues routine does is it establishes the desired distance to be a half a meter. It figures out the current distance to the wall by reading the sonars, and then it specifies an action.

So the first question is what would you like fvel to be in order to make a proportional controller? Which of those expressions makes sense? Take 30 seconds, talk to your neighbor, figure out some answer between (1) and (5). I will ask you in 30 seconds to raise your hand with that number of fingers.

AUDIENCE: [INAUDIBLE]

PROFESSOR: So what's the right kind of expression if we wanted the controller to be proportionate? Everybody raise your hand, show me some number of fingers. OK, the vast majority is saying (2). Everybody likes the idea of current minus desired.

Why is that the right answer? That is the right answer. Why is that the right answer? How do you prove that to somebody?

AUDIENCE: Well, in current divided desired, once you meet your desired distance, you're still going to have to [UNINTELLIGIBLE] velocity [UNINTELLIGIBLE], which really doesn't make sense at this point.

PROFESSOR: Exactly. So that method, we might call extreme cases. Think of the simplifying cases that give you some insight into the problem. So one simple case is what if desired and current were the same? You'd better stop. OK, that's a simple case.

So the simple case says that you better have one of these-- whatever the right answer is, it better have the property that when current equals desired, fvel is 0. Otherwise it's not going to work. And in fact, using just that one simple case, you can eliminate all the ones except (2).

There are some other simple cases, what are some other simple cases? Simple cases. Yeah.

AUDIENCE: currentDistance greater than desiredDistance?

PROFESSOR: Current bigger than desired. So if current were bigger, that would mean that you were starting out way over here someplace. So if you were way over there someplace, you'd want the velocity to be positive. The forward velocity should be positive. That's how you would disambiguate the sign. Similarly, if the current were shorter than desired, if you were too close to the wall, you'd like the forward velocity to be negative. OK.

So that's the proportional controller that we'd like. So we might fill in our wall finder class this way. And then when we built it, if things went really well, we would get exactly the behavior we wanted. But if things went more naturally, it wouldn't quite work that way, and you might get a different kind of behavior.

Let's do that again. So here is the resulting behavior for that simple controller that I just showed. So in some sense, that's not as good behavior. And the way we want

to think about behaviors-- the way we're going to develop today, is to think about behaviors in terms of signals. Plots.

So the first question is, what plot best represents the behavior that I saw here? So which of those plots best represents that behavior? Take 30 seconds, talk to your neighbor, figure out what's the right answer?

AUDIENCE: [INAUDIBLE]

PROFESSOR: OK, so which behavior best represents the illustrated cartoon that I showed previously? Raise your hand, show some number fingers so that I can see if you're with it. Not all correct, but more than 90% correct. The more than 90% answer is number (2). What's good about number (2) that's not good about numbers (1), (3), and (4)?

AUDIENCE: The initial value is different than the final value.

PROFESSOR: Initial is different. So here, I try initial and final being roughly the same. So what's showed here, is a plot. Current distance on the y-axis, step number on the x-axis. And so we can see from here-- and the implication of the axes, by the way, is 0.

So the implication of the vertical axis is it intersects this at 0, unless I label it otherwise. And similarly, this horizontal line intersects the vertical at 0 unless I label it otherwise. So that's the point (0, 0).

So the implication here is that the initial and the final values are the same, as they are here. Here they're not, what makes (2) better than (1).

AUDIENCE: [INAUDIBLE] the current distance is decreasing [UNINTELLIGIBLE].

PROFESSOR: The current distance starts out bigger than it ultimately is. So we start out bigger than the final value, the final value is presumably half a meter. We start out roughly twice that far, and then we see some approach. The approach is not monotonic. So the answer is (2).

Why do you think it under-shot? So on the way to going from one to one-half, it

transiently went through something smaller than one-half. Why do you think it did that? Yeah.

AUDIENCE: Because [INAUDIBLE] this way, there's a small-- there's a small interval of time when its moving at the velocity where it couldn't [INAUDIBLE].

PROFESSOR: So it's a small interval of time between when it does one thing, and it does something else. When it senses and moves, for example. So there could be a time delay, in the system and in fact, that's true, there is a time delay.

So it takes some amount of time for the sonars to register the distance. Then it takes some amount of time for the computer inside the robot to register that the sonar has told it something different.

Than it takes some time from the time the computer commands the wheels until the robot starts moving. All of those cumulative effects mean that you have the potential to overshoot where you're going, because there's delay in the system, there's inertia in the robot.

All of those reasons can lead to overshoot. And the point of today is to figure out some way of predicting and correcting for those kinds of unintended behaviors. So what we will do then, is develop an approach focused on signals, not systems.

So, so far, we've been thinking about how do you build the system? Now we're going to think about behavior. We're going to think about analyzing that behavior. And the focus is going to be on what was the input, what was desired? What was the output, what was achieved? So we're going to be looking at those output signals in order to figure out how good the behavior was.

That approach is called the signals and systems approach. The idea is characterize your system-- whatever that system is, a physical system, a mathematical system, a computational system, whatever it is, think about it by the way it transforms an input signal into an output signal.

That's kind of a bizarre way to think about systems. So let me just illustrate that by

way of a system that you've all seen before. OK, here's a simple system, right? You've all seen this, right? Anything like 8.01 ring any bells?

OK, so that's a simple system, you all know how to solve it. There's a gazillion ways you could solve this system, right? Free body diagrams, kinetic energy, potential energy, there's a gazillion ways you could do it. You all know how to do it, that's not the point.

The point is, that we're going to learn a different way to solve it. We're going to think about the mass and spring system-- not like potential energy and kinetic energy, not like free body diagrams, we're going to think about it as transforming an input signal into an output signal.

So the input signal-- it's kind of arbitrary what I use besides the output, but clearly the thing I have control over is my hand. So it seems natural to associate a variable with the position of my hand, that would be x. Also it seems natural to associate a variable with the position of the mass, that could be the output y. That's not unique.

Every time we try to solve a problem we ask ourselves, what's the meaningful input, what's the meaningful output? The meaningful output could have been the force on the spring. I'm just sort of arbitrarily saying for the purpose of my analysis, I'm going to consider the input to be the position of the hand, the output to be the position of the mass. And I'm going to think about the mass and spring system as a box that transforms x into y.

So rather than thinking about it in terms of free body diagrams, and kinetic energy, and potential energy, I'm going to think there's some input signal x, and there's some output signal y. And what I'd like to do is given x, calculate y. OK, bizarre, why would I do that?

One of the reasons I want to do that is that it's a very general way of thinking about behaviors. It works for the mass and spring system, it works for water tanks. What happens if I have water flowing into a tank that's leaky? Well, it leaks into another tank which is leaky, and that leaks more.

Completely different physics. Probably wasn't covered in 8.01. Probably is something you could figure out. The point is that from the signals and systems point of view, I'm going to map this physics, whatever it is, into this structure.

Think about the tank system as the system that transforms some input. The input is that there's water spurting for some small interval of time. The output is that there's water coming out. And the idea is that I'm going to characterize the system-- whatever it is, as the rule that transforms the input signal into the output signal.

Here's a third example. I could think of a cell phone system. Here again, there are very complicated ways we could think about the system, but I'm going to take the particularly simple approach called signals and systems. Then I'm going to characterize the phone system by the way it transforms some input sound into some output sound. And as you can imagine, there's a way of thinking about performance in terms of that transformation. Ideally, we would like this to bear some resemblance to that.

So the point then, is that we're going to focus on behaviors. And to do so, we're going to think about the signals and systems approach. Represent a system, whatever it is, by the way it transforms inputs into outputs. One of the reasons we like the approach is that it's so general.

So you can use it for virtually any kind of a system for which you can develop a mathematical underpinning. You can use it to analyze electrical systems, mechanical systems, optical systems, acoustic systems, biological systems, financial systems if you're on the dark side. So there's lots of different kinds of systems that are amenable to this kind of approach.

Also, this approach has a nice modularity. Having represented a cellphone by transformation from sound in to electromagnetic field out, as illustrated by this cartoon depicting sound going in and having a radio wave transmitting to a tower. Then we represent tower to tower communication some way, maybe via an optical fiber, maybe via a satellite. Then tower to cell by the same kind of reverse transformation that we used in the first one.

We can piece those all together, we can treat them as modules, because each box takes an input signal and makes an output signal. The method is oblivious to the underlying physics. That affords us a certain amount of power.

And in particular, it's very modular. You can put together modules that represent very different physical substrates. That allows us to go back to PCAP.

If the underlying representations for the different physical substrates are the same, we will be able to-- and we will over the next three weeks, develop a bunch of techniques for combining multiple modules into one. That provides the same kind of abstraction and modularity that we saw in programming for the last two weeks.

So that's the idea. What we want to do is represent a system by the way it transforms an input signal to an output signal. There are many different kinds of inputs and outputs, but a fundamental distinction that we are going to have to make is continuous time, and discrete time.

This system works in what we will refer to as continuous time. Because my position of my hand is a continuous function of the continuous variable, time. The robots, by contrast, work with what we will call, discrete time. Steps.

It turns out the math for those two different approaches are very different. And we will focus entirely in this class on discrete time, because our area of application is the robot. It's not that continuous is deeper, or harder, or anything like that, it's different.

So we'll focus on discrete time. And the point of today then, is to develop some representations for signals and systems of this free time nature, that will let us analyze and predict behaviors of systems like the robot system.

The first class of methods for representing such systems is difference equations. Difference equations are a lot like differential equations, except there's no differentials, there's differences. Difference equations are the discrete time analog of differential equations for continuous time systems.

Simplest possible example here. Say I have an output y, that for reasons that I don't care about, I know can be represented as the difference between two values of the input -- x at time n, and x at time n minus 1. That's a way to represent the behavior of a system, a discrete time system, by using a difference equation. That's in fact, almost a complete description of the system. So let me just explain the way you would use that.

It's almost-- because I didn't tell you anything about the input. I'll tell you something about the input for the purpose of example, where I'll use the simplest possible input I can imagine. Something that we'll call a delta function. A delta function is a signal. It's a discrete time signal. It has the value 1 if the time index is 0, and has the value 0 everywhere else. It's, in some sense, the simplest possible signal that we could imagine. So it's natural to start there.

So what I want to do now is think about if this were a characterization of the system, and if this were the input to that system? What would the output be? That's after all-- that was the question that I posed at the beginning.

We would like to build a representation for a system so that we can predict the output, given the input. So how does that work? So given the difference equation, all we need to do is step through it. OK, it's trivial. We call that analyzing by step by step.

So given the difference equation, given the input signal, all we need to do is sequentially go through the different values of n, and think about the implication of the system on that input.

So if I were to use the value of n, being minus 1, this general form of the difference equation tells me that the minus 1 value of n for the output is related by this difference, with the input.

So y of minus 1 is x of minus 1 minus x of minus 2. Since both of those are 0, it says that the output at time minus 1 is 0. Trivial, right? Trivial.

And similarly, we can just iterate through the solution to the whole signal. So y of 0 is x of 0 minus x of minus 1. x of 0 is that special one, that is 1. So now we get 1 minus 0, which is 1. y of 1 is x of 1 minus x of 0. Now the special one is on the other side of the minus sign, so the answer is minus 1. y of 2 is x of 2 minus x of 1 -- they're both 0. And in fact, all the answers from now on are going to be 0.

So what I just did is a trivial example of -- I use a difference equation to represent a system, and I figured out the output signal from the input signal. That's the method that we call-- that's the representation for discrete time systems that we refer to as difference equations. Difference equations are very powerful. As we will see, of all the representations we look at, difference equations is the most compact representation.

But there are features of other kinds of representations that are also valuable. So the next representation I want to look at is a block diagram. What I'm trying to show here is a picture, a diagram, that represents the same system that we just analyzed with difference equations.

Here though, I'm thinking about it as a signal flow path. I'm thinking about what's the cascade of operations that you need to do on each sample in order to get from the input to the output?

So the difference equation said every value of the output should be equal to the corresponding value of the input less the value before that. I represent that in a block diagram by saying there's a straight through path, y of n is equal to x of n. The plus just adds the signals on these two paths.

So y of n is equal to x of n. Subtract out, because I'm multiplying by minus 1. Delay, so I'm getting the one from before. So this block diagram just is a symbolic representation of that difference equation.

The value of the block diagram-- we'll see several of them, but one of them is focus on signal flow path. If you want to visualize the transformation from input to output, the block diagram can provide visual insight into what that transformation is like.

So as before, if I gave you this representation rather than the difference equation, you could still step by step and figure out the way it worked. It's still easy.

There's one new caveat here. We have to start the system in a state. The state that we will usually talk about is what we will call, at rest. At rest just means all the outputs of all the delays are initially 0.

So that specifies the state of the system at the time the signal was turned on. So the system is at rest, which means that all the delays start out with output equal to 0. So at rest means this delay has an output of 0.

Well, if I tell you the output is 0 times 0 for this delay, then it's a simple matter of stepping through what is the output for each corresponding input? So if I-- the special value of the delta function is that it's 1 at the time 0. So at the time 0, there's 1 coming in. That 1 makes it through the adder, adds to 0. Well, this is 0 because it was at rest. So the 1 adds to 0 and the output becomes 1.

Notice that the 1 also goes down this path, and goes through the gain of minus 1 to give me minus 1. But I'm in step 0. So at step 0, the output of the delay is 0, not minus 1. So the output then, for time equals 0, is y equals 1.

Just like we solved for the difference equation-- after all, I'm hypothesizing that those two systems are the same, they better give me the same answer. So then at the next instant, as the time index goes from 0 to 1, two things happen. The input goes from 1 to 0, and the delay box gets updated. The delay is now reporting to me the value that was at its input. So the output of the delay, which had been 0 because it was at rest, becomes minus 1.

So then what happens? The 1 goes to 0, the 0 goes to minus 1, the 0 adds to minus 1, and we get an answer which is minus 1. Then the input becomes 0. That 0 comes down here, the minus 1 goes to 0. We end up with 0 being added to 0, and the next answer is 0, et cetera.

So the idea then, is that you can step through the block diagram representation, just like you would a difference equation, it's just that now we're thinking about these

blocks characterizing the system, rather than thinking of math as characterizing the system.

Why on earth would you do that? What's good and bad? What's the relative merits of difference equations verses block diagrams? Take 30 seconds, talk to your neighbor, figure out some good feature of each.

AUDIENCE: [INAUDIBLE]

PROFESSOR: OK, we'll start with the easy one. What's a feature, what's a property of the difference equation that makes it very good. I already said it, so. What's good about the difference equation? Yeah.

AUDIENCE: They can be solved mathematically.

PROFESSOR: They can be solved mathematically. The block diagram could be solved, maybe not mathematically, but kind of. Could you refine that a little more? What's special about difference equations that's different from block diagrams? Oh come on, I said it.

AUDIENCE: Use math?

PROFESSOR: They use math, yes yes. They're concise, a little, right? It's a very simple expression to say that. It's by contrast, a bit more complicated to draw this picture, right? It's mathematically concise, right? It's completely accurate, self-contained, concise, small. It's a very concise representation of a system.

So why would we want to use block diagrams? Can anybody think of any good reason for block diagrams other than Freeman's up front saying, do block diagrams?

AUDIENCE: Electrical engineering?

PROFESSOR: Electrical engineering. There should be a deeper reason, I would hope. I don't disagree with that reason. Why do electrical engineers like this? Yes.

AUDIENCE: It's a more physical representation of this.

PROFESSOR: It's more physical? Is there anything that you can see that you can't see in math? Yes.

AUDIENCE: In the way that you're actually going to be programming it. Like, you're gonna make the delay machine, you're gonna make the state machine that multiplies them by negative 1 and you're gonna see how to connect them.

PROFESSOR: That's a really good point. It's kind of isomorphic to the implementation. Everybody get that? It's kind of a picture of the way you would build the system. Along those lines, there's some bit of information-- there's actually more information in this one, than there is in that one. There is exactly one bit more information in the block diagram, what's that bit?

AUDIENCE: Delay?

PROFESSOR: Delay? There's kind of delay by the n minus 1. Yeah.

AUDIENCE: [UNINTELLIGIBLE]

PROFESSOR: The input and output are explicit. Yes. The arrows are the big difference. You can't tell from the difference equation, what's the input and the output? You can tell from the block diagram, what is the input and the output by the direction of the arrows.

So there's more information in the block diagram. There's another way of thinking about it, and this is kind of a summary of several comments that came from the audience. The difference equation is declarative. It tells you a true statement about what the system will do.

The block diagram is imperative. It tells you what to do now. Take the input, put into an adder. Take the input, multiply by minus 1. Put it into a delay. Take the delay output, put it into an adder.

The representation with a block diagram is imperative. It tells you what to do. So there's extra information, but it comes at a cost. It's a more complicated representation. It's a whole picture instead of just an equation.

What we'd like to do, and what I'm going to do now, is develop a different mathematical approach where you get a difference equation that has the same properties of concision, the same conciseness, but also contains all of the information that was in the block diagram. And the way to do that is to change our focus. And this is the big abstraction of the day. Change our focus from thinking about samples, to thinking about signals. Stop thinking about x of n, start thinking about the input signal x.

This is the same kind of lumping that was key to abstraction in Python. Put all the interesting data together into a list, put all the interesting operations together into a definition. Here put all of the interesting samples together into one signal.

So what we want to do, is develop a math by which we can operate on signals instead of samples. So what I'm going to do is replace the representation x of n, little x of n, with cap-X. Cap-X means, all of the n's, it's the signal X. Cap-Y means the signal Y. And I'm going to reinterpret all the boxes.

This box means, take this signal, the whole thing, all n values on it, multiply it sample by sample, by minus 1. Flip the whole signal upside down. So I'm going to reinterpret all of the operations on the block diagram in terms of signals, rather than samples.

To do that, I need a representation-- a mathematical representation, for the delay box. And I'm going to call that R, the right shift operator. If you apply the right shift operator to a single X, it takes the whole signal X and shifts it to the right one. That's all it does.

So I'm going to say Y equals R applied to X, or more abbreviated, RX. Simply says, let Y represent the signal that is the same as X, except every sample was shifted to the right. The entire signal was shifted to the right. That's going to let me represent this block diagram this way.

Y-- the whole signal Y, is the sum of the whole signal X, subtract out R applied to X. Or, even more concisely, calculate Y by applying to X the operator 1 minus R.

So I'm thinking now of an operator. An operator is not something that works on a number. Operations work on numbers, operators operate on signals.

So I'm thinking about operator expression. I'm going to try to formulate the transformation from the input to the output in terms of operators. The way I take X, which is the input, and turn it into Y, is to operate on it with the operator 1 minus R.

OK, just to see that you're with me, connecting signals and samples, assume the Y is RX, which of the following is also true? Take 30 seconds, talk to your neighbor. Figure out some number -- (1) through (5).

AUDIENCE: [INAUDIBLE]

PROFESSOR: OK, so which representation works best? (1), (2), (3), (4), or none of the above? Everybody raise a hand, tell me some number. OK, virtually 100% correct. The answer was (2), why is the answer (2)? Can somebody explain that concisely? No no,no. I asked that wrong, of course everybody can explain it concisely. Do I have a volunteer to explain it concisely? Yes.

AUDIENCE: The R operator just shifts all of the values in X of n to the right. So you just add 1 to each of these basically?

PROFESSOR: So you think about it-- so a good way to think about this is to think about simple cases. Right? That's the same thing I talked about earlier.

What's a simple case? Well what if X-- let's imagine that X is simple. So let's say that X looks like that. So X is delta. What would happen, what is the signal RX? It's a right shift operator. So what does the signal RX look like?

Shifted to the right, right? That's the whole point. So the right shift operator gives you that signal, and we've said that that's Y. So is it true that Y of n is X of n, for all n? No.

Is it true that Y of (n plus 1) is equal to X of n for all n? Well, is it true for n equals 0? So if we did n equals 0 we get Y of 1, which happens to be 1. And X of 0, which also

happens to be 1. And if we choose any other n, we would get two 0's.

So at least for this simple case, and it seems to be true, and if you think about building upon this simple case, you can convince yourself that number (2) is always true. And in fact, the general rule is going to be that the left hand side has to have a number that's bigger than the right hand side. Which is only true for number (2).

So the idea then, is that by changing our focus, by looking not at samples, but looking instead at signals, we can generate an algebra that looks for all the world like difference equations. Except it knows the direction from the input to the output. So it's more powerful.

And in fact, this new algebra is going to obey some very simple properties which we can get a hint at here. If we were to cascade two systems, imagine this system which looks just like the system we've been looking at, but it's cascaded with a clone. The question is, what would be the behavior of that cascade?

Well, according to our operator representation, this Y1 signal is just the 1 minus R operator applied to X. Analogously, the Y2 signal should be a similar 1 minus R operator applied to the Y1 signal, which gives us a very concise representation for the cascade. The point of the slide is that the operator representation gives us a representation that is just as compact as difference equations.

It has other features, that it can be manipulated just like difference equations. So if we continue with this same example and try to think of the transformation on a sample level, we could say that Y2 is Y1, the straight path, minus Y1 delayed. But then we could substitute for Y1 of n, that Y1 of n is X of n, this path, subtract X of (n minus 1). Similarly, collapse, and we get a simple expression.

Now if we think about that same sequence of operations in operator notation, we get a much simpler expression. Throw away the index arithmetic, it's just R. So the Y2 operator is 1 minus R applied to the Y1 signal. The Y1 signal is 1 minus R applied to the X signal. The total is 1 minus R the operator squared, which by polynomial math is just 1 minus 2R plus R squared, the same thing we got there.

The point is, the operator notation is just as compact as the difference equation representation. And it contains all the information that's in block diagrams. And it's just as easy to manipulate as a polynomial. So it's got lots of features that make it superior to difference equations.

The most important of which, is you will be able to understand all systems that we represent using the R operator using polynomial arithmetic, something you learned in high school. There's nothing new here. That's what we like. Representations that simplify the task of finding an answer. We'll be able to find the answer to these operator expressions by treating them as polynomials.

So you can get a feeling for the way that works by looking here, the power of this. So the power here is that, among other things, you'll be able to use the operator representation to prove equivalences. The idea is that here's a system that we looked at before, that was the cascade of two simple delay systems. Here's a somewhat more complicated, somewhat simpler, representation.

The point is, it's different. And if we compare the operator representations for the two, we see that they are the same. And what I'm about to prove, is that when the operator representations are the same, the systems represent the same transformations-- provided they all start at rest.

The provided is important. Obviously since they have different delays in them, if the delays didn't all start out at 0, the differences in the delays could propagate into the output. So all of the statements that I'm making are premised on the idea of initial rest.

The other important thing to remember about operators is that it's a higher level of abstraction. We can think about the operator as composing things, but the things that are composed are whole signals, not samples. And here's an illustration of how to think about that previous example.

How do you think about this transformation, 1 minus 2R plus R squared. What happens when you apply that operator to an input signal X? Well let's say that X

was our unit sample signal. In order to apply this operator 1 minus 2R plus R squared, all we need to do is think about each component.

1 times X is X. Minus 2R applied to X is minus 2RX. Plus R squared X is just plus R squared X. So we start out with X, which is a unit sample, minus 2RX means shift it to the right and multiply by minus 2. Shift it to the right, multiply by minus 2. Plus R squared X means shift it twice to the right.

So the result, this operator 1 minus 2R plus R squared applied to X is just the sum of these things. So you can think about the operator expression, it's just like algebra, except that the elements are signals, not samples.

And as I alluded previously, you can make powerful statements about the way these operators work which map isomorphically onto polynomial math. So for example, it's easy to prove that if you were to cascade the 1 minus R operator with R. So start with X, apply 1 minus R. Start with X, apply 1 minus R. Then apply R. That's going to result in the same signal, assuming initial rest, as if you were to flip those operators.

The way I can see that is by thinking about signal flow. You'll remember that I said, one of the features, one of the powers of the block diagram representation is that we can look at signal flow paths. We can use that as a way of proving things.

This system has two signal flow paths, the one that goes straight through that way, and the one that goes down this way and up that way. Because of the addition, all it does-- the adder just adds the result of those two flow paths. So the first flow path introduces one delay, the second flow path inverts and puts in two delays.

So there are two ways to get from the input to the output. Similarly here, there are two ways to get from the input to the output. One of them goes through a delay and then goes through the adder, the other goes through two delays at a minus 1. But that's the same two signal paths that were in the top one.

So that's a way of using the block diagram to prove a relationship about the operator. What I've just showed, is that the operators obey commutativity. So what I

was able to show is that I can commute these two operators-- doing a general proof is slightly more complicated, I proved it in a special case. But the general case works too.

Just like polynomials, operators commute. And I indicated why you should think that's true by thinking about signal flow paths. Multiplication distributes over addition.

I apologize, the diagram in your notes is wrong. This is right. I will always post the notes-- I get up in the morning, I make coffee, I read the lecture notes, and I say, oh my goodness, there's a wrong figure.

In this particular case, one of the staff members wrote me an email and said, Hey Freeman, your slides, something or other, is wrong. He was right. So this is the right diagram.

So the idea is that if multiplication distributes over addition, we should expect that R applied to 1 minus R, would give R minus R squared. And we can again get a feeling for why that ought to be true by thinking about the signal flow paths.

The two signals flow paths that represent-- so the signal flow path that represents here, this says take the 1 minus R operator and apply it to X, then apply R to the 1 minus R operator, result. As opposed to this one says, apply the R operator to X, then apply the R squared operator to the X and subtract them.

If you think about the signal flow paths through those two systems, they're also the same two signal flow paths. And here's a more complicated example that shows associativity. You can think through that, same idea.

The reason that the idea-- the big important point is, difference equations are a good representation for discrete time systems. They're mathematically compact. Block diagrams are a good representation, but they have more information. They tell you what is the input, what is the output, and what are all the different flow paths through the system?

Operators kind of combine the best features of both. It's mathematically concise, it tells you which is the input and which is the output. And you can visualize all the flow paths by thinking about all the ads in the operator expression.

OK, to make sure that everybody's up with me, how many of the following systems are equivalent? You have 30 seconds.

AUDIENCE: [INAUDIBLE]

PROFESSOR: OK, so how many of those are equivalent? More hands, more hands. Not necessarily more fingers, but more hands. OK, about 75% correct, roughly speaking.

OK, how many distinct signal flow paths are going through the first system? How many distinct signal flow paths can you see? Well, here's one. How many more are there?

AUDIENCE: Three.

PROFESSOR: Three more. So here's one, here's one, here's one, and here's one. All you need to do to think about this system, is think about all of the signal flow paths through all of them, make a sum, and see how many of them have the same sum.

So the path with the greatest delay through this path is two times two delay delay. Two delays with a multiply by 4. What's the path with the biggest delay through this one? Also straight through. Also delay of two, also a coefficient of 4. How about this one? So that's this way.

So they all have the same paths with maximum delay. The delay is 2 and the coefficient is 4. This one has four possible paths, this one only has three. So there's one straight through this way. There's one that has one fewer delay. And there's one that only has-- so there's a straight through one, there's a delay of 2, and there's a delay of 1.

So let's do the straight through one. This one has a straight through path. No delay, coefficient is 1. This one has a straight through path, coefficient is 1. This one has a

straight through path, coefficient is 1. So all three systems have the same maximum delay path, they all have the same minimum delay path, we only have one in the middle yet.

This one has two ways to get a delay of 1. 2 delay, 2R, or 2R. Since they're both 2R, they sum, so that's 4R. This one only has one way that we can get 1 delay, and that is to come this way and then go that way, that's 4R. This one, to get 1 delay, I take the center path, and that's 4R.

Each path has the same ways to get through the system with zero delay, 1 delay, and 2 delays. They're equivalent in the sense that if you started them with initial rest, they would all generate the same output given the same input. So the answer's (3). OK?

So far I've only worked with systems that propagate the inputs systematically through to the outputs. We call such systems feedforward. Things are a little bit different when you have cycles. We call such systems feedback.

So what I want to think about now is how do you think about a system that has a feedback loop in it? The interesting thing that happens when you have a feedback loop is that the operator expression no longer represents a simple sum of input signals. Let's look at what happens here.

So Y is apparently the sum of two things. It's the signal RY, which comes around that way. Does everybody see that? So if I think about labeling this input as X, labeling this output as Y, then the correct label for this point is-- don't everybody shout at once. If this can be labeled as the point Y, what is the correct label to a label at this point? RY.

So the signal Y must be, RY plus X. What that says, is that if I apply the 1 minus R operator to Y, I should get X. OK, that's a fine operator expression, except that it's not a formulaic operator applied to the input. The operator is applied to the output.

OK, the difference here is the difference between an imperative system, like we

talked about block diagrams when we were thinking about samples. The block diagram tells you what to do step by step. Regardless of whether you have feedback, the block diagram always tells you take X of n, add it to Y of n, whatever. There's an imperative rule, do this.

We took the block diagrams and we turned them into operators, and we ended up with something that is not imperative. This is much more the kind of statement we got when we did difference equations. This is a statement of truth, it's declarative.

If you tell me the signal X, it must be true that the resulting signal Y when operated upon by 1 minus R, is X. So the idea is that it's a declaration, it's not an imperative rule. Does everybody get that?

So this statement up here told me a rule, start with X, apply the 1 minus R operator and you will get Y. That's an imperative operation, do this. This is a declaration. If you tell me X, Y must be the signal that when operated on by 1 minus R, gives you X. But it doesn't tell me a way to find it. It tells me a truth, but it doesn't tell me how to find that truth.

So let's go back, let's back up. OK, we got a representation, we like the representation. It's concise, it has many of the features of block diagrams, it doesn't seem to be imperative. Well, that's a problem. So let's back up, think about how this same system that ran into a problem with the operator, think about what must the answer be.

Well the answer we can figure out by doing step by step. Imagine that it starts at rest, so the output starts at 0. And now I just tick through the samples.

So when the first sample comes in, X is equal to 1, I'm thinking about the unit sample response. We call the delta function the unit sample. When the unit sample at time n equals 0 comes in, it has a value of 1. The 1 adds to the initial condition, which is 0, to give me 1.

Then, this output is 1, so when the clock ticks, the input goes from 1 to 0, but the output of the delay goes from 0 to 1. So when the clock ticks, I get another 1. And

that persists. Does everybody see what's going on?

So I initially had a 0 coming out of the delay. The unit sample made the first output be one, but then that 1 fed back in to make this be 1. Which combined with the next 0 to give me the same 1, and that state persisted.

What's different, is that the output signal persists long after the input went away. In fact, there is a prescriptive way to figure out the relationship between the input and the output. It's just that it takes an infinite number of delays.

Here's an alternative system that would generate the same response to a unit sample signal as was generated by the simple feedback system. It needs to generate the answer 1, 1, 1, 1, 1 -- when the input is just 1.

Well, the output at 0 happens through this path. The output at 1 happens through this path. The output at 2 happens through this path. 3-- et cetera.

There's a separate path for every one of those separate components of the output. That's how we can think about this construction. The input had a single non-zero entry, the output has an infinite number. We can think about that as resulting from an infinite number of paths, or something similar, about the simple feedback system, which can be represented by that operator representation, and the infinite feedforward system.

This is a simple feedback system. This is an infinite feedforward system. There's something the same about those two. In fact, they're equivalent in the sense that if all the delays start out with initial conditions of 0, they will generate the same response to all possible input signals. Those two signals are equivalent, and that's proved here.

All you do is you say, OK, Y2 depends on X2 this way. If X2 is the same as X1, I can substitute it. But X1, according to this rule, looks like 1 minus RY1. When you multiply out this mess, you get Y1.

What I just showed is that if X1 is equal to X2, then Y1 is equal to Y2. Those two

systems are the same. Well that's weird. So there's something the same about that operator and that operator.

We write that this way. So here's the feedback system, we think about that as representing the operator Y over X, 1 over (1 minus R). So in order to calculate X, cross multiply by X. Y is the operator, 1 minus R applied to X.

So we want to say Y is the operator 1 minus R applied to X. What is the operator 1 over (1 minus R)? Well, if you didn't know anything but polynomial math, you might have expanded this in a series.

And in fact, that gives you exactly the right answer if you were to expand 1 over R in a series. So, for example, evaluate it using synthetic division, evaluate it with a Taylor series, however you want to do it. Think about R as though it were a number, just like you would if it were a polynomial. Expanded just like you would if it were a polynomial.

And what you see, is that there's a representation for this operator 1 over (1 minus R) that is equivalent. That's exactly the same as if I applied the operator 1 plus R plus R squared plus R cubed to X. Those two are equivalent in the sense that if both systems start out at rest, and if they are both applied to the same input, they both generate the same output. So that gives us that way of thinking about operators that have numerators and denominators.

So, to make sure you're up to speed, a system is described by the following operator expression. Determine the output of the system when the input is a unit sample.

AUDIENCE: [INAUDIBLE]

PROFESSOR: What's the first thing I should do? OK, this is one of those systems that has the R polynomial in the bottom. So it says that X must be the same signal by cross multiplying-- X must be the same signal as the (1 plus 2R) operator on the Y. OK, that's backwards. That's not the way I want to think about it.

How do I make that into a forward statement that tells me what operator gets applied to X? The answer is that. So what do I do? Multiply by-- actually you could cross multiply. How do I convert this into an operator that looks like just the numerator times X. Yeah.

AUDIENCE: 1 minus 2R plus 4R squared.

PROFESSOR: Exactly. What I want to do is convert it by synthetic division, Taylor series, whatever method. I want to think about what would 1 over (1 plus 2R) look like? What's the reciprocal of 1 plus 2R? That is 1 minus 2R plus 4R squared, et cetera. So now I have this, which I apply to X, which is a unit sample.

So now I want to think about applying this operator to the unit sample signal. But that's easy. The first term just brings out delta. Minus 2R applied to delta shifts the delta by 1. Gives me delta of n minus 1, and multiplies by minus 2.

And that whole mess then just says that my output looks like this. If the input was X, which was a unit sample signal, my output has an infinite number of terms. Each one is a delayed version of the predecessor, and the weights go 1, minus 2, plus 4, minus 8, plus 16, and diverge.

So what we just did was pretty complicated. We just solved a block diagram, but we did it with polynomial math. We did it with math that you learned in high school. That's the point.

In fact, the point of today is that any system that's built out of simple parts-- delays, adders, scalers, that sort of thing-- can be represented by a difference equation. Fine, that's good, difference equations are wonderful. They can equivalently be represented by an operator equation.

The operator equation has more information in it. It knows how to get from the input to the output. It's imperative. It's easy to manipulate. You use the same rules that you use for polynomials.

So all in all, this is a more powerful kind of representation. And any system that can

be represented by a difference equation can similarly be represented by an operator equation. That's why we're focusing on operators.

So final question. Think about-- do everything backwards now. Here's a block diagram, find the associated difference equation. And the idea is to take advantage of operators. In the interest of time, let me just do it.

If we wanted to-- because I'm running out of time. So I could start with the block diagram, I could stay in block diagram domain. Presumably that will work, that's hard. I want to do the easy way.

So convert it to operators. How do you convert a block diagram to operators? Replace the delays by R, label all the signals. x becomes X, y becomes Y. I don't have a name for this, so I'll call it E, error. I don't have a name for this, so I'll call it W, who knows.

And then I'll express each of the relationships imposed by the plus sign, this R or this R, by a line of operator reasoning. The plus says that the E signal is X plus W. The R says that the Y signal is R applied to E. This box says that the W signal is R applied to Y.

I get three equations in R. I just solve algebraically. None of this difference stuff, none of the square brackets with n's in them. I just use algebra. So I solve it algebraically and I get this. And that translates into a corresponding difference equation, showed here.

The point. The point is three different representations. Difference equations, block diagrams, operators. Operators are easiest. Even when I was asked to solve a problem that has no operators in it, it's easier to cast it into an operator expression, solve it in the operator domain, and then turn it back into a difference equation.

Starting next week, we will figure out much more powerful things that we can do with operators. This is just the beginning. So with that, let me just summarize that we looked at three representations, and the point of the labs for the week are going to be to exercise this, to get some experience with representing signals in Python.

6.01SC Introduction to Electrical Engineering and Computer Science

Spring 2011