PROFESSOR: Hi. Today I'd like to talk to about a new module called Signals and Systems. Our previous module on programming, we focused on object-oriented programming and state machines as different methods by which we could model the physical world. In this module we're going to talk about a new way to model the physical world called the discrete linear time-invariant system.

You might say, Kendra, why do I need to do that, or why would I move away from state machines. I seem to be able to model everything using state machines. The short answer is, you want to be able to move away from state machines, because you want to be able to predict the future. But I'm not really going to be able to get to how you're going to be able to predict the future until two videos from now.

So for now we'll introduce discrete linear time-invariant systems and also talk about different knowledge representations you might encounter them in, so that you recognize them when you see them and can manipulate them as needed to talk to other people about them. Let's go to the board.

Last time we talked about state machines. We're able to use them to model pretty much any system. We can model the evolution of a particular process over time using a record of all previous inputs and outputs and possibly previous states, if we included that in our output.

But the problem with state machines is that they, like most programming, encounter the halting problem. At some point your program can reach a little complexity at which you cannot determine what it's going to do without actually just running it and seeing what happens.

This is great for researchers. This is the method by which people go out and have to find something or discover something instead of being able to like, simulate it using a machine. But if you want to be able to promise things to other people or predict what's going to happen, then it helps if you can model something that has a high level of complexity by abstracting away the complexity and interacting with it as

though it has features that indicate that it's going to behave in a particular way.

So if you want to make a control system that does a particular thing, or look at an existing system and say, well, I know that your power plant's going to blow up in five years, because of this, it helps if you have some understanding of particular classes of systems and what kind of long term behaviors those particular classes of systems generate. And that's why we look at discrete LTI. It's a particular class of system -- in particular discrete LTI is what we're going to look at in 6.01 because it's fairly simple. But once you've learned how to deal with the system in that manner, you can use the skills that you've learned there to approach more complex feedback problems or more complex control problems.

First, an incredibly brief review of linear time-invariant systems. When you're talking about a linear time-invariant system, both the inputs and the outputs are going to be real. We're not going to deal with the complex plane at all. If you were to model your LTI system using a state machine, that state machine would be dependent on a fixed amount of previous inputs, outputs or states.

You give some amount of leeway for start state or the fixed number of states before you get to your fixed amount of data structure that represents your current state. But if you're running a linear time-invariant system, the amount of information that you need to figure out your state in a long term sense is always finite and fixed.

In terms of functional expressions, you've probably seen these associated with linear time-invariant systems. Know that this also means that you can represent linear time-invariant systems as additions or scalar multiplications of your existing function.

So that's out of the way. These are all true of LTI. We're going to focus on discrete LTI. One, because we can use a discrete state machine to model it. And two, because it makes it easier to represent using computers. Once you've got that digital abstraction, you don't have to worry about having access to a truly continuous function.

All right. Now that you know what we're going to talk about, how do you talk about it to other people? Here are the different representations that you might see a discrete linear time-invariant system in. One of them is called a difference equation. And you've probably encountered this in one of your math classes. It says that you're going to take a sample from a signal y at some sort of given time step, typically n. And when you're writing out a difference equation that represents an entire system, you usually see y of n on one side on the left, and then everything that represents the functional component on the system on the right, which in this case determines what determines the output at every time step.

In this very particular case, we're talking about an accumulator, which means that at every time step the output is determined by the input plus the previous output. So at every time step we take the input and whatever it is we were outputting before and output it again. Pretty straightforward. If somebody described to you a difference equation in words, you could probably turn it into a equation at this point.

Note that even though there are variables associated with these samples, they are still very particular samples. And the thing that you're interested in is probably not even the samples at all. It's the relationship between the samples, when they were sampled, and relative to one another, and how that relates to the output at a given time step. If you then want to talk about an entire signal, then you can use an operator equation. And you'll probably see these things when you do any sort of research on control theory or feedback.

Instead of representing the sample of the signal y at a given time n, we're just going to talk about the overall signal capital Y. Likewise, instead of talking about x at a given time n, we're going to talk about the signal capital X.

Remember that I said that the most important part of this equation is the fact that there's a different relationship between these two signals and when you were sampling from them. When we want to represent this relative delay, we represent it using an R. And in particular, I want to note the fact that the degree of R represents the amount of delay associated with the sample from a particular signal. So if I

wanted to make this n minus 2, I would reflect that change in my operator equation by changing the degree of R. And because we're working with a linear time-invariant system, if we wanted to scale this by 2, it'd be the same as scaling this by 2, et cetera.

Now I can talk about signals. What if I want to talk about a physical manifestation of the relationship between these signals, and/or what if I want to use something kind of like circuit diagrams to talk about what kind of signal manipulation I'm going to do? This is where block diagrams come in.

Block diagrams are incredibly PowerPoint-friendly. Block diagrams mean that everybody is on the same page, because all you have to do is trace out the arrows. Block diagrams mean you're going to do a combination of gains, delays and adders to visually represent the relationship between your output signal and any input signals that you have.

Up here I have actually drawn an accumulator. And it's got a box around it, which will be relevant in two minutes. But right now you can ignore it. I've got my input signal, or my input signals, are typically on the left. And the progression of gains, delays and adders associated with the block diagram represent the relationship between the input signals and the output signal. And then I'll have my output signal on the right.

Most people indicate flow with arrows, in part to make things easier to read. But typically, you only need the arrow indicating where you're sampling from and what your output's going to look like. In this case in order to get Y, and in the general sense, I can backtrace from the signal that I'm interested in through my diagram and figure out what values I'm actually interested in. So in this particular case, Y is a linear combination of X and whatever this represents, which is RY.

If I did want to put a 2 here, I've been talking about gains, delays and adders, but this diagram doesn't contain a gain yet. So let me show you how you would include a gain. Gains are typically represented by the value of the gain inside an arrow. It looks like an op-amp from schematic diagrams. We'll get to those later in circuits.

It's not essential that the arrow point in the direction of flow, but people might look at you funny if you don't put it in the direction of flow. And the other thing that I think I want to note here is that if you see a minus sign here, it means X minus 2RY, which is the same as putting a negative value on your gain. So right now, we're back to X plus 2RY.

OK. We've got to block diagrams. You might be saying at this point, Kendra, what are block diagrams good for other than PowerPoint presentations and possibly arguing with my friends over what's going on. And I say, they're really good for abstraction.

I could draw a box around this and abstract it away into a function and say, if I put something into this box, and I want to get something out where the action that happens in here is identical to this schematic, then I can find an equation that actually represents that operation. And the way I do that is I'm going to take my operator equation and solve in particular for Y over X. In this case, with the 2, Y over X is going to be 1 over (1 minus 2R). Note that if H is equal to the expression Y over X, and I multiply H by X, I get out Y.

This concludes my coverage of motivations for learning about discrete LTI systems and also the different representations that we will want to use when talking to other people about discrete LTI. At this point we can also start talking about how we get to the point where we can start predicting the future. And then in the video after that I'll actually start talking about poles.