MITOCW | watch?v=TeVSxZgIHAA

PROFESSOR: I wanted to give a lecture, because as I told you in presentations, I love feedback. In fact, I love it so much, that I think the examples we're going to do, we can do analytically are not maybe sufficiently compelling for you to believe how exciting control can be. So let me just start by giving a few minutes of a research result we had a few years ago.

That's why you've got color images-- everybody noticed the color images on the slides. I brought this robot in to the first recitation of [INAUDIBLE]. It's a big 2 meter wing span ornithopter, and in my case, we've been trying to design control systems to make airplanes move more like birds. Here's one example of that. One thing birds do that planes don't do very well is land on a perch. So we asked the question, can we take a simple airplane with fixed wings, no flapping allowed, and make it land on a perch-- UAV stands for unmanned aerial vehicle-- land on a perch like a bird.

The story I want to tell you is that with feedback design, you can.

Here's why it's interesting and hard. The reason airplanes don't land on a perch is because, when your wing is at a low angle, which a plane normally is when it's flying, the air flow around the wing is relatively easy to model. It's easy to write a block diagram description of. But if you go to moderate angles, that's still true. The air stays attached to the wings and everything-- [INAUDIBLE]

At a low angle, the air is still attached to the wing, and so we have very good models of the airflow around here, and the lift and drag forces that you get when you're flying in these regimes. But if you go up too far, like this, then everything changes. So you think, the air can't quite stay around the wing. You get separation. You get big vortices pulling off the back. It's very nonlinear, very unsteady. Very complicated flight regime. And as such, our best aircrafts to date mostly stay down in this regime where we have good models and we can design conventional control systems for.

But birds don't do that at all. So birds are, often, when they're landing on a perch, they're up even far beyond this. Maybe even 90 degrees. "Angle of attack", it's called. They're way into this deep stall regime-- that's called stalling your wings when that happens. And it seems like they're doing a lot better landing on a perch because we don't see our airplanes do it.

But you can actually try to quantify that. So if you want to compare the performance of a bird landing versus a plane landing, the first thing you have to do is you have to take out the differences in mass and wing area and all these things. But you can do that. And a fair comparison is the distance averaged drag coefficient, which is just a way to scale out the effects that you'd expect from having a bigger wing or a bigger mass. You can plot this drag coefficient for a few different vehicles.

This is a standard runway landing of a 747, would get the drag coefficient, just so you have some calibration, of about a 0.16. The X-31 was a super maneuverable research vehicle. It was designed to do super short runway landings, and it got a drag coefficient during those landings of about 0.3. There's a few other projects out there about trying to make perching planes that were getting similar numbers.

But then we went out and looked at nature. I have some collaborators at Harvard that work with real birds. It turns out-- I wanted him to tell me the numbers from some really elite bird, like at least a hawk or something like this. But they work on pigeons. So that was the only number we could get, was a pigeon. They actually convinced me by the end that pigeons are really good. The way they can sort of dive through the fence and get your lunch by the food trucks? I mean, that's actually-- they're seriously skilled birds.

We even had-- one of the fun things about doing this kind of research is you get visitors. We had will.i.am from the Black Eyed Peas come to the lab. And he said-- after we told him the story, he said pigeons are ghetto birds. They got mad stop. That sort of summarizes it.

You already know the answer. But if you look at the drag coefficient that a bird gets when it's landing, they get a 10. So they're doing orders of magnitude better than our best planes in terms of stopping. If you just want to be fun about it, you could say, what would it take for a 747 to get a 10 to impress will.i.am. And it turns out that 747 would have to go-- they fly at about 450 miles an hour cruise speed. It would have to stop in 40 meters to do what a bird's doing. The wings would probably pop off. There's problems with that. But the dynamics are impressive, of the birds.

And what's more, when they're doing that, they're getting incredible accuracy. This is one of my favorite videos of a perching owl. If you watch really closely, you can see the airflow get complicated on his wings. You can see his leading edge feathers start to flip over. Boom. The way you do that is you put food by the camera.

We tried to build these simple planes and ask if could they do what the owls are doing, what the pigeons are doing. Very simple planes. We did them in an indoor environment. We made it so it was basically a boring plane, but it can do very interesting things in pitch, up and down.

Then we spent some time trying to build a dynamical systems representation of the plane. And how do you do that? Well, you shoot the plane a bunch of times into a fishing net. And you collect a bunch of data. And like we talked about in recitation, you can potentially get from that data a model of the dynamics. And turns out, we had to do a nonlinear dynamical model for the vehicle, but we could do a linear actuator model plus delays and everything. We built a pretty good model of the plane, which is sort of-- for an aerodynamics crowd, this would tell you that this is a surprisingly good fit to the lift and drag forces over a very large range of angles.

Then we had a 6003 problem, basically. The nonlinear terms make it a little different. And what we did as a group was we innovated a way to design a feedback controller that could make this thing very accurate during very high performance maneuvers. And a long story short, we can now shoot an airplane into a motion capture arena. This is slowed down about 11 times. It's this airplane right here. And it goes into a very deep stall. And we can land with an accuracy, enough accuracy to land on a perch.

It turns out if you can build a good enough control system that can handle the complexity of the dynamics, then you can make these things happen. This is just-- to convince you that the dynamics are complicated, this is the flow visualization. We built a wind tunnel releasing smoke from the leading edge and took a picture of it to show you how complicated the dynamics were.

So control is potentially an incredibly powerful idea. You can make-- we try to make robots that run like ostriches. We try to make all kinds of things happen. You could imagine improving wind energy. You could imagine all kinds of things where control is an integral part of it.

The feedback in there was absolutely essential. If I took the same plane and thought about the model a lot and then designed a controller, just a set of commands to the elevator, to try to make it land on the perch, it misses the perch every single time. And with feedback, we can hit-- it's only with feedback that we can hit the perch every single time. And there's a reason for that.

You guys probably heard the idea that fighter jets are unstable without the control system, right? Fighter jets are-- a lot of times, the systems that have peak maneuverability, that's often at odds with stability. In fact, when you try to make a very high performance fighter jet, and you want to be able to turn, you actually-- if the control system is off, this thing should be unstable. It's not a complete pathology, but that's pretty true, because what you want, you can elicit a very fast turn if you basically let the system go unstable. Then you can let the unstable dynamics of the system make a very rapid turn. In fact, it's common to build a system that is unstable, and it's only stable when there's feedback in the loop so that you can have very high maneuverability when you need it.

You've already done feedback if you took 601, right? So what I want to do today is-- I taught 601 when some of you were in there. I want to go through the example that you already worked through in the 601 lab for those of you that took it. I think it's sufficiently complete here if you didn't take it. But we want to use that-- I'm going to use that as an example to build on what you already know and to show us with the new tools how far you can get in thinking about what that robot did for you last year. Do you remember this example of the little pioneer robot that had to go to the wall, the foam wall, using the noisy sonar sensor? And we wanted to get a desired distance from the wall? And by the end, you guys were moving the board around and it was trying to track the wall?

In that exercise, in the 601 lab, you guys tried a bunch of different gains. For some gains, we saw responses that sort of looked like this. If you gave a desired distance to a wall, it would go up to that desired distance. For other gains, you saw some oscillation. You saw some big oscillations if you tried all the gains we recommended. Now you guys have a deep understanding of what kind of things can cause that type of response.

The way we told you to think about it in 601 is actually the way that we often think about control systems. Typically, we have something called the plant, which describes the dynamics of the robot or the thing we're trying to care about. You guys know why it's called the plant often? What would be the history of calling it "plant"? It's a weird name for it, right? Actually, some of this stuff grew up in the chemical industry, in chemical plants. Is that what you said? No? It's actually-- it has a history in chemical plants. But now everybody calls their robot plants.

The plant is the dynamics of our robot, for instance. You've got some output of that. In this case, it's the position to the wall. A sensor that reads it. It may have its own dynamics. And

your goal is to take some reference command, a position you want to go to, let's say, compare it with what the sensor is saying and build a controller that takes that error and makes the robot or control system do what you want it to do.

In this example, the one you studied in 601, the plant was a very simple model of the dynamics of the robot. It was just a first-order model. We said that the output, the distance from the wall, the distance in front of the wall, at time n was just the distance in front of the wall time minus-- because, you remember the frustration of the flipped sign, too? But I kept it consistent here. So it's T times the velocity, just happens that since the velocity is going this way and the distance is getting smaller, you get a minus T. This looks almost like the first-order approximation of a CT system, right? So that's the dynamics of the vehicle.

The sensor, we're going to assume for now, is just perfect, that it just immediately tells me the distance to the wall and gives the perfect feedback. And then we designed a controller, which just takes the error, the difference between the desired position and the actual position, multiplies it by a constant K and comes up with a velocity command that goes to the motors.

You can visualize that as a block diagram, of course. You get that this plant model here-- it's got a minus one here and a minus one here, so that's equivalent having a delay in the forward path. And then it's got feedback, because the previous signal was directly put around to the feedback. So this part here is the model of the plant. Then we put the T in, with a gain here, so I guess this part here is the plant. Here's our simple controller and our sensor is perfect. So that's our block diagram. We've just turned our robot into a block diagram. And we know everything about how to analyze those things. You can use the operator notation. You could think of it as a system function, too.

This guy here you know is-- you guys can do this in your sleep now, almost? But if I just take that by itself, what does that come out to? It's a plus here. This is x and y. So I get y is R x plus y. Or y over x is R over 1 minus R. Then I multiply that by K and a negative t. And then I do the exact same computation again to get this loop around. And this is our input output system function. Simplify it a little bit, I get this. Simplify it a little bit more to identify the pole, and you can see that the pole now looks like 1 plus KT. T represents the time step between updates, K is our feedback gain.

Now, we can start thinking about what happens if we choose different-- let's just lump K and T together, because the K you choose is going to be intimately connected to time. In this system,

there's no point in separating them. So we just talk about KT as a system. If we chose KT-- KT is almost always going to be negative. You want negative feedback, in general, for making things stable. If we choose KT equals 0.5, then with that system function, there's a delay in the forward path, so it's going to be offset by one and then it's just got the simple single pole response. The unit step response similarly looks like this. So the question is, what determines the speed of that response?

Here you go. I have to get you to talk to your neighbor the way Denny always seems to get you to talk to your neighbor. Take a minute. Figure out which of these or none of them do you think would give you the best response for this simple block diagram system. Talk out loud to your neighbor.

OK. Show of fingers, what do you think it is? OK. A lot of right answers. Let's just do it real quick. The fastest possible convergence is going to be at the pole zero. Pole zero, what's it going to look like? It's going to give you-- the best you can do if you zero this out, you get the impulse response. The unit sample response would just be R. You're going to have a delay of 1, that's inevitable. You get one non-zero entry and then zeroes everywhere else. In discrete time systems, you can actually kill things in a single step. You can set a pole at 0 that's the fastest possible response.

But if you think about the different responses for different values of K, you can use the pole zero diagrams to pretty much understand everything there is to know about it. For KT less than 0 to negative 1-- like I said, we want to think about negative feedback-- that's going to take us from out here on the unit circle back towards the origin. If I keep making KT more negative, it's going to go out here. If I make KT too negative, bad things are going to happen again. The poles are going to go outside the unit circle and will actually have alternating diverging responses. The answer is negative 1. Negative 1 puts you at a pole of 0.

Here's to think about it physically. If I choose K correctly, since KT was negative 1, that means K's going to be negative 10. That's going to be commanding exactly the right speed so that the robot, after the one tenth of a second, let's say, it gets you exactly to the right position. Unit sample response in this case would be saying, the robot's at zero, I want it to be at 1, and then I want it to be back at zero. The command is going 0, 1, 0. And the robot is going to be almost doing exactly that. It's going to go from 0 to 1 in a single step. And then back to 0 in a single step. It's just going to be delayed by one. So you give it a command saying, go here, go here,

go back. And it's going to do exactly the right thing-- boom, boom, except for one step delayed, because you can't get rid of that R.

So if I plot it-- let me draw a stem diagram, but coming down in time so that I can line up with the axes up here. If I start with an initial position here, and I command it to go to the desired front position, it's going to go boom right to that front position with one unit of delay. And then it's going to stay there for the rest of the time. It would be a unit step response.

But that's not what we got to see on the robots in 601, right? The real robot didn't work like that. And the way we made the robot model more realistic was we said, OK, but your sensor's got some delay. And actually, if you knew what was going on behind those 601 robots, it's actually had a lot of delay. There's Python running serial interfaces to over the serial link to the fairly old controller in the pioneer robot. So the delay is real. There's a delay of about 1/10 of a second in the sensor.

If I take that exact same controller, exact same gain that I already did, now put it in this new system that has an extra delay, then what happens? I get a velocity of 10 after one step, looks good. But then, uh-oh, there was some delay in the sensor I didn't realize was here, so I'm still going to take corrective action trying to get me there. It's going to move me all the way to the wall, smash into the wall, and then it's going to realize it was zero. It's going to-- there's some delay in seeing that. It's going to move me back. And you're going to get oscillations.

You can see that now by just adding the model to our block diagram. If we put the delay now in the feedback path, otherwise, keep the block diagram exactly the same. Now, you can write the system function. What's the resulting system function given that this thing is in the feedback path?

Go ahead and put your fingers up when you think you've got it. All right. Fantastic.

Just like I did here, you can just quickly replace the accumulator there with the equivalent block diagram. Do the loop again. It's going to be exactly what we did before, but it's got a new R in the feedback path. Giving us the R-squared here, just on the feedback path. It's exactly the same otherwise except for that R, and that simplifies out to this. So the answer was 4. That's just operator notation, polynomial algebra.

If we want to find the poles of that system, we can just go ahead and factor the quadratic form in the denominator. The roots of the denominator look something like this, which is a little ugly

to think about. But we can think it through. For general KT-- when KT's small, KT's about zero, then I'll go ahead and simplify that a little bit to say, KT could sneak inside the-- KT and KT-squared aren't so different. We're going to sneak it inside here. And then you can see that the poles end up at-- around K equal to zero you get a pole at the origin. But you also get a pole up by the unit circle. Around 1 and around 0.

Remember, when you've got multiple poles in the system, the total response is going to be dominated by whatever is the slower pole. The total system response is going to be dominated by the slower pole. In this case, the slower pole's the one closer to the unit circle.

What about if KT equals negative 0.25? We can pop that in and solve it. Exactly-- math works out nicely. We get poles at a half, two poles at a half. And in fact, there's a smooth transition between-- if you look at the numbers between KT equals zero and KT equals 0.25, you'll see as you vary that gain, the poles move together along the real axis until they come together at a half. So here, you've got a purely real response dominated by these poles at a half. System's stable. If you keep changing K though, the poles came together and then they split off and start going this way. And in fact, if you look at negative 1, which is the one that was the best response, it put a pole exactly at 0, for the system with no delay. If you put it at the system with delay, they land exactly on the unit circle. Right there. Complex poles on the unit circle. You're going to get a stable oscillation. Which is exactly what we saw there.

Just a quick-- you know this like the back of your hand now, but what's the period of that oscillation? You've got two poles on the unit circle right there. What's the period of oscillation? Put your fingers up when you think you know. Yep. Good. Most people got it. This thing was a half, so it's 1/2 square root of 3 over 2. So that thing had to be at pi over 3. It's actually the same pole that I was using in recitation yesterday. So if you have a pole at pi over 3, it makes it around in six steps. The period of the oscillation is six.

This is generally true, that if you put a controlled gain, a feedback gain, into the closed loop dynamics, then even a simple gain can allow you to really shift around the poles of the system. And since we know the response of the system, the zeros matter, but for convergence, for the rate of convergence, it's the biggest pole that dominates. It's very nice to understand how those poles move with K. If you change the system, the way they move with K is different. And you can just change K to tune the response to be what you want, from KT equals 0 to infinity, the poles go towards here and then off in both directions, actually, to infinity.

So, KT equals negative 1 was the fastest possible response for the system without any delay in the sensor. What's the fastest possible response for this one?

Oscillations are allowed. I just want the fastest possible response.

Yeah. Looks good. So where do I want the poles to be for the fastest possible response?

These poles are still stable and oscillations are fine, but the absolute-- the magnitude of those poles is larger when they're out here in the complex. When I'm down here, I have got a pole over by one, that's going to dominate. So the best I can do is if I put the poles at a half. That gives us the largest pole, the smallest possible magnitude. And that happened if the poles-- the double poles at a half happened when KT was negative 0.25. Most of you said, the answer is 2.

In general, delay is a bad thing. In DT systems, we have good representations of delay. It's even worse in continuous time. Delay is a bad thing. It tends to make control systems not work as well. If you just took the ideal sensor, we had a K equals 1, we had a response that started here, we could put it anywhere we wanted along the real axis. As far negative as we wanted, of course, what we chose was to put it right at the origin. But as soon as we just added that one piece of delay, the things we could do with proportional feedback changed completely, and ultimately, got worse because I have two poles now, first of all, and I can't simultaneously get both of those poles to go to zero. In fact, as I change them, they go off into complex. The fact that it's complex isn't bad. But the fact that there's two of them, and the best I can do now is get to a half, which is a much slower response, the best possible response.

If I added even more delay, things get even worse. If I had two units of delay and I went through the same exercise, what you'd see is that the poles would start in the same place. They'd come together and go there, and there's another pole that goes off. Just because there are three poles in that system. Two poles come together and split off this way. And this one goes off this way. And the place you probably want, depending on how fast this one splits off, but the place you probably want to put it is when the two poles are together right there. That's going to give you the fastest response. But that fastest response is still slower than what we had-- it's a bigger number than a half. It's 0.682. It's going to be a worse response

than when I had a delay of one, which is intuitive.

That's a quick reminder of something you already did in 601, of using feedback and the tools we've already got, which is poles and zeros and everything, to understand how to design feedback gains. We're going to get more into it in the next couple of lectures. But let me just convince you that this stuff is real. And I showed this diagram once in 601, but it probably means even more to you now. This is an F-14. It's one of the best modern engineering control systems ever built. It was built for this F-14. They got more research money and modeling this vehicle, designing ultimate gains for it. It's such a success story that you can actually, if you're in MATLAB, you can open up the F-14 demo and see what a flight control system for an F-14 looks like.

MATLAB-- I don't know if you've played with Simulink-- MATLAB has a language called Simulink, a graphical language that allows you to draw the block diagrams and simulate them and even design controllers for them. And it turns out you can make a block diagram description of an F-14 using only tools from 601 and 6003. You can see all the same adders and gains, transfer functions, system functions like this. The only essential difference is that in some of the diagrams, you see multiple inputs coming in. In this class, we've restricted ourselves so far to thinking about single input, single output systems, which keeps everything clean. All the intuition scales to multiple inputs and multiple outputs. But that's sort of the only big addition of complexity when you go to this model from what we've done before.

If you zoom in onto the controller here, you can-- these block diagrams in this language allow you to abstract away a bunch of hidden things with a single transfer function. If I zoom in, then you can see things you recognize. It's got a low-pass filter, which is just a system with a pole at negative 1, exactly what we did in recitation the other day. This is what people-- this is what MATLAB uses on an F-14. I guess it might not be what the military uses. But it's modeled after what the unclassified documents say is happening on an F-14.

Everything here, you understand, right? A proportional controller in the end. One difference in F-14s is that-- and in general for these more complicated systems-- they'll design slightly different controllers given the situation. So if you have an altimeter on there and pressure sensors, an inclinometer will tell you the angle of the plane-- given those sensors, they'll pick a different K for a proportional controller out of a pot of a library of controllers they've already designed. So it's called gain scheduled control. But the analysis and design of each K is a linear systems design and analysis.

This is super powerful stuff. I think-- signal processing is good, too, but control is where it's at. I guess I went a little fast, but that's your introduction to feedback. We'll do DT and CT feedback in the next couple of lectures. And if anybody hasn't picked up their exams, we have them over here. And we'll see you in recitation tomorrow.