MITOCW | Lecture 1 | MIT 6.832 Underactuated Robotics, Spring 2009

RUSS TEDRAKE: OK. Welcome to Underactuated Robotics. I'm glad you're all here. Can I get a quick show of hands, actually, who considers themselves hardcore robotics types? Don't be shy. It's a good thing to do. Who's here to see what this is all about? OK. Good, good.

So I've thought of course twice before. This is the first time that it's a TQE course for Area 2. So I'm hoping to excite a slightly more general audience. And your responsibility, then, is to ask me questions if I say things that if I assume prior knowledge that people don't have and make sure that everything is coming across.

The only real prereqs that I assume for the course are basically some comfort with differential equations and Ordinary Differential Equations, ODEs. I assume some comfort with linear algebra. And we use a lot of MATLAB, so it helps if you know MATLAB.

What I don't assume is that you're-- I mean, it's great if you have taken previous robotics courses. That's certainly a good thing. But I'm not going to assume everybody knows how to crank out the kinematics or dynamics of rigid body manipulators and things like that.

So hopefully, if you've got a background in here, then everything else should follow. And the course has got a set of course notes that will be published on the website just after the lectures. And they should be pretty much contain what you need to know, or at least have links to the things you need to know.

OK. So today is the warm-up. I want to start with some motivation. I want to make sure everybody leaves understanding the title of the course, if nothing else. So I want to tell you some motivation why I think underactuated robotics is such an important topic in robotics. I'm going to give you some basic definitions, including the definition of underactuated.

I'm going to go through an example with working out the equations of motion for a simple system to a review of dynamics. And then I'm going to tell you quickly, in the last minutes, everything else you're going to learn in the course. And then we'll go through that more slowly in the next 25 lectures.

OK. So let me actually start with some motivation. That involves plugging in this guy. OK. How many people have seen ASIMO before? Good. OK. So ASIMO-- let's watch the little promotional video for Honda's ASIMO.

This is a few years old now. But ASIMO absolutely is the pinnacle of robotics engineering over the last 20 years, I'd say, even. So Honda, turns out, without telling anybody, was working on walking robots from the early '80s till they announced to the world in 1997 that they'd started building these walking robots.

And they started putting on these shows where they'd kick soccer balls and walk up stairs. It's absolutely, absolutely amazing. I mean, this is it. This is what we've been waiting for in robotics for a very long time. This is a humanoid robot absolutely doing fantastic things. It's a marvel of engineering. The precision, the amount of computation going on in there-- it's something we've been waiting a long time for.

OK. But let's watch it again and be a little more critical this time. So what's wrong with ASIMO when it's walking? OK, it looks a little stiff. It looks like a guy in a space suit. If you look carefully, you'll notice that it's always got one foot flat on the ground. That doesn't look quite right. OK, well, we'll forgive the goalie.

He's work in progress, I guess. But the whole thing just looks a little bit like a machine that's, let me say, not comfortable with its own dynamics. It's taking a very conservative approach. The fact that it can go upstairs is really remarkable.

It does have to know exactly where those stairs are and the geometry of the stairs. But you see, it's taking a very, very conservative approach to walking. It's always walking with its knees bent, its feet flat on the ground, and has this rigid astronaut walk to it.

OK. Why is that bad? It's bad because it requires a lot of energy, first of all. So just imagine walking around with your knees bent all the time. It turns out, ASIMO uses 20 times as much energy if you scale out mass and everything as a human does when it walks-- 20 times. And that makes a practical difference because the batteries it's got in its belly only last 26 minutes, and it's a lot of batteries.

It matters because it's walking-- because of this very conservative approach to walking, it's walking a lot slower than you or I would. They actually have top running speeds of ASIMO. They're six kilometers an hour. But that's a little bit below where you are I would sort of comfortably transition from walking to running if we were just going down the street.

So it's considerably slower than a human when it runs. And although there's some really amazing videos of walking on stairs and things like that, the videos you won't see are the ones of it walking on terrain it doesn't know everything about or even uneven terrain. It doesn't do even train particularly well.

OK. So in some ways, ASIMO is the very natural progression of what robotic arm technology, which started in factory room floors, matured into a walking robot. OK. It's a very high-gain-- we'll talk a lot about what that means. It's a very high-gain system. It's using a lot of energy and feedback in order to try to rigidly follow some trajectory that it's thought a lot about. And it's doing that in a very conservative regime-- feet flat on the ground, knees bent.

So there's a different approach to walking out there. This one was built by Cornell. It's called a passive dynamic walker. It's almost not a robot whatsoever. It's a bunch of sticks and hinges with ball bearings at the joints. But if you put this thing on a small ramp going down and give it a little push, look what it can do. That's just falling down a ramp-- completely passive machine powered only by gravity.

So these passive walkers are a fantastic demo. I mean, it's unbelievable that they can build these things that walk. It's a glorified slinky. But it's walking like you and me, right? Probably more so. Most people would say that looks a little bit more like the way we walk than ASIMO does.

But what's really amazing about it is it says that this really conservative, high-gain, feet flat on the ground approach to walking, it's certainly not a necessary one. And it suggests that if you really want to do some energy-efficient walking, maybe even more robust walking, then what you need to do is not cancel out your dynamics with high-gain feedback and follow some trajectory. You need to think a lot about the dynamics of your robot. So this is just the dynamics of the robot doing all the work-- no control system, no computers, nothing.

OK. So that's a story in talking about why maybe dynamics matter a lot. And we should really start by understanding the dynamics before we do a whole bunch of robotic arm control. It's actually true in a lot of different fields.

My other favorite story these days is flying things. So if you look at state-of-the-art military aircraft, this is an F-14 landing on an aircraft carrier. Even in the most highly engineered control systems we have, it tends to be that aircraft stay in a very, very conservative flight envelope.

The same way that ASIMO stays with its feet flat on the ground and does this really stiff control, the airplane stays at a very low angle relative to the oncoming flow and uses high-gain control in order to stabilize it. So fighter jets-- you might know, the patent fighter jets tend to be passively unstable, and control systems are doing amazing things to make these guys work. But they're not doing some basic things that you can see every time you look out your window.

So here's a cardinal doing sort of the same thing-- landing on an aircraft carrier, landing on a branch-- about the same thing. But unlike the airplane, the cardinal's got his wings spread out way out to here. And what that means, if you know anything about aerodynamics, if you take a wing and the airflow is moving this way, and you have it at a low angle relative to the oncoming flow, then you have a very simple attached flow on the back of the wings.

And it turns out that linear control and linear models of the dynamics work pretty well in that regime. If you go up to a small angle of attack, like the fighter jet's doing, then the air can bend around the wing. Everything still stays attached, they say, to the wing. And you can still do linear control ideas.

OK. But if you go up and stall your wings out-- that's what's happening here. If you go up to a higher angle of attack, the air can no longer bend around the wing fast enough. Something more dramatic happens. You get a big vortex in this picture. And what happens is the flow gets much more unsteady. It starts shedding vortices. And you start stalling your wing.

Now in these regimes so far, the dynamics have proven very, very complicated to even understand, even model, and considerably harder to control. The birds doing it every day of the week. Somehow, we don't know exactly how, but he does it all the time. And he does it with gusty environments. He does it when the branch is moving. He probably misses every once in a while. But he's doing a pretty darn good job.

There's a reason why he does it too, right? It's not just to show off or something. But if you are willing to go into this more complicated flight regime by stalling your wings, and if your goal is stopping, then, actually, you get a huge benefit by going up to that high angle of attack. Not only do you get more drag just because you have more surface area exposed to the flow, but when you start getting separation on the back of your wing you leave a pocket of low pressure behind the wing. The air can't quite come in and fill in the space right behind the wing.

And that acts like an air brake. It's called pressure drag. So the birds-- the planes are coming into this conservative approach in order to maintain control authority. The birds are going [GRUNTS], hitting the air brakes and coming to still somehow doing enough control to hit that perch, which is kinematically more difficult, I think, than even hitting an aircraft carrier.

All right. So In my group, we've been working on trying to make planes do that kind of thing. So this is our airplane that comes in and tries to execute a very high angle of attack maneuver in order to land on a perch. It's a slow-motion shot, slowed down so you can see what happens.

And actually, just to convince you that the flow is complicated. This isn't our best flow of visualization, but it shows you what's going on here. The airflow, as it comes in, this is now that same plane with-- we're emitting smoke from the front of the wing, the leading edge of the wing. And it comes in at a low angle of attack.

And you can see the air is mostly-- it's actually already stalled because it's a flat plate. But the air is on top of the wing in the same way I showed you in that picture. And as you go up to a high angle of attack, you get this big tip vortex that rolls over. Everything gets a lot more complicated.

So the point I'm trying to make with these two examples are first of all, robots today are just really, really conservative. Dynamically, they're very, very conservative. They're operating just a fraction of the level of performance that they should already expect given the same mechanical design. With the same mechanical design, a simple little plane but better control, we can start doing things that look more like birds.

And the second point that I'm going to make more formally in a minute is that the underactuated systems, underactuated robotics is essentially the art, the science of trying to build machines which use their dynamics more cleverly instead of trying to build control systems which use actuation motor output in order to override the dynamics. We're going to do an underactuated system that just pushes and pulls the natural dynamics, tries to do these more exciting dynamic things. As a consequence, we need to do smarter control.

This is a computer science course. So what have I said anything to do with computer science yet? I believe that there's new techniques from computer science, machine learning, motion planning that are basically changing the game. It's allowing us to solve some of these control problems that haven't been solved before.

Just to throw a few more cool examples out-- so if more willing to do more clever things with our dynamics, then there's just countless things that we can do. So if you just care about efficiency-- this is a wandering albatross. If you just measure the energy in some metabolic estimate of the energy it consumes versus the distance it travels, then it's about the same as a 747, which is actually cool because they're quite different sizes. But if you just do some dimensional cost of transport, it actually works out to be almost the same efficiency as a 747. So maybe we haven't gained anything.

But it turns out if you look more carefully, the albatross uses the same amount of energy when it's sitting on the beach as it does when it's flying across the ocean. So this guy can fly for hours, days without ever flapping its wings. Go across the entire ocean that way, into the wind, out of the wind-- you name it-- because they're sitting there and they're just riding on gradients due to the wind over the ocean. So the energetic cost this guy's experiencing is just digestion and things like that. He's actually not doing hardly any mechanical work in order to fly across the ocean.

OK. If you care about maneuverability, you know, so falcons have been recently clocked diving at 240 miles an hour. This one is pulling out of that 240-mile-an-hour dive to catch a sparrow. That's pretty good. I mean, planes-- in terms of sheer speed, planes are going to win every day of the week.

But if you look at some sort of dimensionless aspect of maneuverability, then birds are still the masters of the sky. Bats can actually be flying full speed this way in two flaps, which turns out to be just under half of the wingspan, they can be flying full-speed the other way-- two to five flaps. This guy's at Brown have been recording this.

Obviously, bats are actually some of the most maneuverable. They can fly at high speeds through thick rain forests. They can fly in caves with 1,000 other bats on top of them. At least, that's what I get out of the movies. [LAUGHTER] And they're doing all these just incredibly dynamic things in ways that our control systems can't even come close to right now.

And this is one of my favorite videos of all time. Again, the story about efficiency. This is a fish now, not a bird. But it's almost the same thing. They're both operating in a fluid.

This is a rainbow trout. So the rainbow trout are the fish that swim upstream at mating season. So every year, they make their march up the streams. It turns out if you watch these rainbow trout, they tend to hang out behinc rocks. Someone thought, it seems like it's tiring work going upstream maybe there's something clever going on when they're hanging out behind these rocks.

So what they did is they took that rainbow trout out, and they put it in a water tunnel. And you're looking at a view from above of the fish swimming in the water tunnel. This is George Lauder's lab at Harvard. And that's what it looks like when it's just swimming in open water.

If you take that same fish, put it in the water tunnel but put a rock in front of it-- now if you've looked at rocks in a river, behind a rock, you'll get some swirling vortices, some eddies. The fish behind the rock just completely changes the kinematics of its gait. So that's suggestive that there's something clever going on here.

But this is the clincher here. The dynamics matter if you're a fish. This is now a dead fish. It's a dead fish. There's a piece of string making sure it doesn't go back and get caught in the grates. That would be messy. But it's slack. As it's moving around, you'll see when the string is catching.

This is our rock now. It's a half cylinder. The water's going this way. There's going to be swirling vortices off the back of that rock. Let's see what happens if you put a dead fish behind the rock.

So the vortices are knocking the fish around. That's not too surprising. What's really cool is when the dead fish starts swimming upstream. That's pretty good. So the water's going that way, and the fish just went that way. And it's dead. So dynamics matter if you're a fish.

And if you care about birds-- mechanically, we're capable now of building robotic birds. This is our best copy of a hobby ornithopter. But we can build birds that fly around with Derek at the controls. But if you asked me how to control this bird to make it land on a perch, pull energy out of the air, we haven't got a clue.

We're working on it. We haven't got a clue. Yeah, that hit a tree. And Derek recovered. We've got the first flight of our big 2-meter wingspan. This is an autonomous flight. You can tell it's autonomous because it flies straight into it about runs into the building and then we hit the brakes, and it has to go in and hit the trash can.

But mechanically, we're close to where we want to be to replicate some of nature's machines. We've got a long way to go. But really, we're at the point where we have to figure out how to take these magnificent dynamical machines and control them. And that's what the course is about.

OK. So let's get a little bit more careful in what I mean by underactuated. Systems. So we'll give you some motivation. We're going to try to make robots that run like humans-- run like gazelles, swim like dead fish, and fly like that falcon that comes down-- so not a tall order at all, right?

So in order to start doing that, let's start by just defining what it means to be underactuated. Let me ground things in an example. Let's take a two-link robotic arm. I'm going to describe the state of this system with theta 1 and a relative angle here, theta 2. And let me quickly draw that twice as big.

We'll parameterize it here with the-- that'll be L1 with the length. This will be able L2 here. So we call this mass 1, mass 2. So we'll assume it's a massless rod with a point mass at the end, just to keep the equations a little cleaner for today. And it's got two angles that I care about. And there's a couple lengths at play.

So throughout the class, I'm going to use the convention that q is a vector of the joint angles or the coordinates of the robot. In this case, it's going to be theta 1 and theta 2. If I have motors on the robot, let's say, I can apply a torque here because I have a motor at the elbow, maybe a torque here. So I'll call this torque 1, this torque 2. I'm going to call that u, vector u. There's all the inputs to the system. So this is the joint coordinates. These are the control inputs.

OK. So it turns out if you want to write out the dynamics of this system, if you want to be able to, say, simulate the way that this pendulum moves, well, most of the robots we care about in this class are our second-order. So everything's based on their mechanical system. So we've got F equals ma governing everything. In this case, a is going to be the second derivative of q.

So what I really have-- I should also say that q-dot is going to be the joint velocities. And q-double-dot is the joint accelerations. So if I want to describe the motion of these systems, of this kind of a robot, then what I need is to find an equation that tells me the acceleration of that system based on the current position, velocity of the robot, and the current control inputs.

If we're living in second-order mechanical systems world, then that means I'm looking for a governing equation-- equations of motion of the form f is some nonlinear function of q, q-dot, u, potentially time too, if it's a time-varying dynamics, if there's something else going on clocked by time.

So basically, the entire class, we're going to be looking at second-order systems governed by some non-linear equations like them. It turns out that actually most of the robots we care about, there's even a simpler form. Turns out we can almost always find-- it's not always-- but for many robots, we find that the equations of motion are actually linear in u. If I'm going to put in torques or something to my robot, that it turns out that the way that the torques affect accelerations is linear in the input.

So let me write a second form, which takes advantage of that observation. OK. So I've said almost nothing here. I'm just saying there's some nonlinear terms that depend on the current state, and velocity, and time. There's some other nonlinear terms that get multiplied by u. But the only power in this is that I'm saying that the whole thing is linear in u. And it turns out to be-- I'll convince you as we go forward that that's true.

OK. So here's our-- we're finally grounding everything here. Let me tell you what fully actuated means. Just think about what this equation is too. So q is a vector. Q-double-dot is also a vector. In this case, q was a 2 by 1 vector. Q-dot then is also a 2 by 1 vector. Q-double-dot is also a 2 by 1 vector.

So this is a vector equation of a 2 by 1 vector in this case. This is some vector-- 2 by 1 vector. In my case, I had also two control inputs. So this is also a 2 by 1 vector, which means what is this thing going to be? That's going to be a 2 by 2 matrix in that example.

What matters, what makes life dramatically easier, and what most robots today have really assumed is that they assume that F2 is full rank. So a robot of this form is fully actuated if the rank of F2, q, q-dot, and time is equal-- I'll write the dimension of q here-- its full rank.

OK. Why does that matter? What does that mean first? What it means is that if I know F1 and F2, then I can use u to do anything I want to q-double-dot. OK. I'll show you that. I can say that right now pretty explicitly.

So let's say I know exactly what F1 and F2 are. Let's say I choose a control law. I want to choose u as some function-- I'll just call it pi-- of q, q-dot, and time. So I want to come up with a controller which looks at my positions, and my velocities, what time it is, and comes up with a torque. Let's say, I did F2 inverse q, q-dot, and time times negative F1, q, q-dot, time plus, I don't know, some other controller, but I want u-prime, let's call it.

So if the rank of F2-- if it's full rank, if it's got the same rank as the dimension of q, then that means that this F2 inverse exists. And I think that if you plug this in for u right there, what you're going to see is that this cancels out this. If I did it right, then, and this cancels out this. And what I'm left with is a simple system now. Q-double-dot equals u-prime.

AUDIENCE: Shouldn't that be q-double-dot [INAUDIBLE]?

RUSS TEDRAKE: Where do you want q-double-dot?

AUDIENCE: Is that u?

RUSS TEDRAKE: This is u-prime. So just some other u. So what I'm trying to do is now say that I'm going to effectively change the equations of motion of my system into this, u-prime. I might have called that maybe q-double-dot desired or something like that. That would be fine too.

So what did we just do? We did a trick called feedback linearization. I took what was potentially a very complicated nonlinear equations of motion, and because I could, using my control input, command every q-double-dot,

I can essentially effectively replace the equations of motion with a simpler equation. This is actually a trivially simple equation. For those of you that know, this is what would be a series of single inputs, single outputs systems. They're decoupled. So q-double-dot 1 is equal to the first element of this. It's just two vectors.

So that just looks like a trick. I'm going to ground it in an example in a second here. But first, let's finish defining what underactuated means. So what is underactuated going to mean?

AUDIENCE: That the other two matrices is important.

RUSS TEDRAKE: That's right. Good. Yeah, a system of that form is underactuated if the rank of F2, q, q-dot, and time is less than the dimension of q. In words, what underactuated means-- a system is under actuated if the control input cannot accelerate the system in every direction. That's what this equation says. If the control input u cannot produce.

That's just what the equation said. You could imagine if the form of the equations wasn't linear in u, then we'd have to adapt this rank condition to do this. But this, I think, for today is a very good working definition of underactuated. And we'll improve it as we go through the class. There's a couple of things to note about it.

First of all, as I've defined it here, whether you're underactuated or not actually depends on your state. So you could say that a robot was fully actuated in some states and underactuated in other states. Now why would that happen? Maybe there's a torque limit, or there's an obstacle or something like this that prevents you from producing accelerations when you're in some configurations.

Intuitively, what's happening in ASIMO is that it's trying to stay in this very conservative regime because then those are the states where it can act like it's fully actuated. And if it was running like you or me, then it's underactuated. But what I want to try to impress on you is that this dichotomy between fully actuated and underactuated systems, it's pervasive. I mean, so robotics, for the last 30-some, easily, years, has almost completely made this assumption that F2 is full rank when designing controllers.

If you learn about adaptive control, all these manipulated control ideas, computer torque methods-- all these things-- you're implicitly making this assumption that you can produce arbitrary torques. You can use arbitrary control effort to produce arbitrary accelerations. What that does, that's why all these proofs exist for adaptive control and the like because you can then effectively turn your system into a linear system that we know how to think about.

And the reason that dichotomy is so strong is because what happens if you're a control designer and you don't have the ability to take your nonlinear system and turn it into a linear system is that you have no choice but to start reasoning about the nonlinear dynamics of your system, reasoning about the long-term nonlinear dynamics of the system.

And analytics break down pretty quick. But computers can help. That's why we're revisiting this kind of idea. So factory room robotic arms tend to be fully actuated, except for very exceptional cases. Walking robots and things like that, as we'll see, are underactuated.

So let's do that example.

AUDIENCE: So are you implying that in order to have agile robots, we need to have underactuated robotics?

RUSS TEDRAKE: Absolutely. I'm going to finish making that argument, but absolutely. That's exactly what I'm saying. The question was, am I implying that we need to do underactuated robotics to have agile robots? Yeah. I would even say that I'm implying-- I'm a little biased-- but I'm implying that every interesting problem in robotics is interesting because it's underactuated. If you think about the problems that are unsolved in robotics-- maybe manipulation, walking, cool flying things-- if you look closely, if the control problem is considered unsolved, it's probably underactuated.

The things we know how to do really well-- picking and placing parts on a factory room floor-- that's fully actuated. Now manipulation is hard for many other reasons. You have to find the thing you're manipulating. You have to think about it. But there's something fundamental in robotics research that happens if your system suddenly-- if you don't have complete control authority. And all the interesting problems that are left seem to be underactuated.

OK. So instead of talking about abstract F's, let's make it specific. Let's write the equations of motion for our two-link robotic arm. So how many people have seen Lagrangian mechanics? Cool. So the class isn't going to depend on it. I'm going to do it once, quickly.

And it's in the notes. If you haven't seen Lagrangian mechanics, it's a good thing to know. And it's in the appendix of the course notes. It'll be posted. But I want you to see it once to just see that there is actually-- if what you care about first is just coming up with the equations of motion, then there's actually a fairly procedural way to do that for even pretty complicated systems. So let's do it for this not-very-complicated system.

OK. So let me do that in pieces. So let's say, this is mass 1. Let's say that it's that position x,1. If I call this x,1 and x,2-- x and y-- that makes them a coordinate system there. Let's say that the mass here is that x,1, and the mass 2 is x,2. So the first thing to do is just to think about the kinematics of the robot. And in this case, they're pretty simple.

So as I've written it, x,1 is-- what is it? The x position in x,1 here is l times sine or cosine? Sine of theta 1. And the other one is negative L1 cosine theta 1. Now we're going to get bored real quick if I don't adopt a shorthand. So let me just call that l,1, s,1. So that'll be shorthand for sine of theta 1. And this will be negative l,1 cosine, 1.

If I want to do the kinematics of x,2 here, that's going to depend on theta 2. It's actually also going to depend on theta 1 because I've got this in a relative frame. That theta 2 is relative to the first link. So it turns out the kinematics of x,2, we can actually start with just x,1. It's the position of x,1 plus another vector, which is l,2 sine of theta 1 plus theta 2. If you work it out, that's the right thing-- cosine theta 1 plus theta 2, which are shorthand as x,1 plus l,2 s,1 plus 2 negative l,2 c,1 plus 2.

OK. So the derivatives aren't too bad. I can do those. Let's see. If I want the rate of change of x1, intuitively, that's going to start depending on the joint velocities, right? So how does that work out? The time derivative of x,1 is going to be l,1 cosine theta 1 times theta 1 dot. And then l,1 sine theta 1 times theta 1 dot. And x,2 dot is going to be x,1 dot plus l,2 c,1 plus 2 times theta 1 dot plus theta 2 dot. And l,2 s,1 plus 2 theta 1 dot plus theta 2 dot. So we now have solved the kinematics of the machine.

To write the dynamics Lagrangian style we need to think about the energy of the system. So let's call T the total kinetic energy. And in this case, it's pretty simple. This is why I went with point masses. It's 1/2 mv squared, which in vector form, looks like 1/2x 1 dot transpose m,1 x,1 dot plus 1/2 x,2 dot transpose m,2 x,2 dot.

OK. And then we're going to define the total potential energy as u. And this it's just mg times the vertical position of the thing. So it's just mass 1 times gravity times I'll call it-- I guess I'll just call it y,1, which is the second element of that. I want to even not introduce new symbol. We'll just do l,1 c,1 negative. And this is minus m,2 g, y,2, which is l,1 c,1 plus l,2 c,1 plus 2. Sorry for going into the corner.

OK. But you can all write the kinetic and potential energy of the system. So Lagrangian derivations of the equations of motion just uses this Lagrangian, which is the difference in the kinetic minus potential. And I think a very good exercise is to understand the reason why this works. But for our class, we can actually just use it as a crank that we can turn.

If we write this out, and then you do some simple math on it, where this is called a generalized force, it turns out if you plug these in to this equation, turn your calculus crank, then you end up with the equations of motion. You end up with two equations that have the form-- they give you some equations in terms of F, q, q-dot, q-double-dot, it's some function of q, and this is actually where the u's come in. So in the simplest form, it comes up like this. And with a little work, let the call that F-Lagrangian so it's not the same F. With a little work, you can separate out the q-double-dots and get it to the vector equations we were talking about before.

OK. If you take those equations that you get and you pop them into MATLAB, and it's pretty simple to start simulating the equations of motion of the two-link arm. This is with zero control input. So this is just what happens if you take some two-link arm, apply no torque, let it go. Then you get this. I put some damping in there extra so we didn't have a demonstration of chaos.

But there's a pretty procedural way to go from very simple kinematics, doing some pretty simple energy calculations, and getting yourself to a simulation of even very complicated mechanical systems. So the forward dynamics, we understand. Now it turns out, there's actually very good algorithms for this too. If you have a 100-link robot, you certainly wouldn't want to turn the crank by hand. But you can download good software packages that write recursive versions of this algorithm that have very efficient computation of those dynamics.

OK. Now let's start thinking about what it means to have control in that system. It turns out, if you enough of these equations, if you punch in enough different robotic arms and walking robots or whatever-- oh, John yeah?

AUDIENCE: I think maybe minus 3L/3q.

RUSS TEDRAKE: Yeah, OK. Good catch. OK. So if you start punching these equations enough, then and then you start noticing a pattern. Turns out, even very complicated robotic arms tend to have equations that fall into this stereotyped form.

OK. This is almost just f equals ma. This is the mass matrix, the inertial matrix. The c here is the Coriolis terms. The g here is the gravitational terms-- potential terms. And then this is the torques.

These are called the manipulator equations. We're going to revisit them. You don't have to have complete intuition about them right now. But what I want you to understand is that if you take the Lagrangian dynamics on some rigid body manipulator, then you're going to get something out in a form that looks like this. Now this is actually a pretty powerful equation. It tells you a lot of things.

So there's a q-double-dot term that's multiplied linearly by something that only depends on q. So by leaving q-dot here, I've already strengthened my form by putting q-double-dot in linear here. So arbitrary equations don't fit this. This is a pretty special set of equations. There's some terms that depend on q-dot. And then there's some potential terms which only depend on q.

And then we have our joint torque king of things over here. And in fact, there's actually a lot of well-known structure in these equations. So it turns out, I could have written the energy of the system as 1/2 q-dot transpose H, q, q-dot. This inertial matrix analogous to mass is related to the kinetic energy of the system.

And what that means actually, just by thinking of it this way, it's well-known that H is positive definite. It's uniformly positive definite. You can't have a negative kinetic energy. And that manifests itself that this matrix H, which appears all the time, turns out to be equivalent to its transpose, its symmetric, and its positive definite. That's shorthand for positive definite. It's a matrix greater than zero.

And in fact, if you look at the equations I punched in for the robotic arm, it's exactly just a matter of computing H, C, G, and B, which is the matrix that maps your control inputs into joint torques. So H is a inertial matrix. C is Coriolis. G is gravity. I think B was this because people were running out of letters. I don't know. I don't know a reason to call it B. But in general, B could be a function of q maybe. But it's just some mapping between your control inputs in the torques that you want to get.

OK. So knowing that I've taken my simple manipulator, I found equations of motion that take this form. If I have torques to give-- torques at both the elbow and the shoulder, then it turns out for that example, H and c and G all just come from the Lagrangian. And B-- what's B going to be in that example? What size is it going to be first?

AUDIENCE: 2 by 2.

RUSS TEDRAKE: 2 by 2. And if I'm assuming that my control inputs are exactly the torques, then B is just the 2 by 2 identity matrix. Is this system fully actuated?

AUDIENCE: Yes.

RUSS TEDRAKE: Why is it fully actuated?

AUDIENCE: Because the rank of the [INAUDIBLE] matrix is 2.

RUSS TEDRAKE: OK. But there's one other term that was one other part of that statement.

AUDIENCE: That's equal to the dimension of q-dot.

RUSS TEDRAKE: But I need to get the mapping from q-double-dot u.

AUDIENCE: Oh, matrix is positive definite.

RUSS TEDRAKE: Yeah. Because the inertial matrices are always also positive definite, that if I actually write out q-double-dot that for these systems, I get an H inverse q times all that stuff-- B, q, u minus C-- oh leave the q-dot G. And we know H inverse exists. I told you it's positive definite. So as long as this thing is full rank, which as you said, it is, then that system's fully actuated.

OK. That means I can do anything I want to that system. What should we do that system? What should we do? Let's replace the dynamics with something else. Well, I can't do anything. It's going to have to be two variables or less, the system I want to simulate. I can't make it simulate a whip if I've only got two. But I can make it simulate any sort of two-dimensional second-order system.

OK. How about we take our two-link pendulum and make it act like a one-link pendulum. That's a simple enough thing to do. So what I'd do is I'd find the equations of motion for the one-link pendulum, and I just do my feedback linearization trick. I'd cancel it out, and I'd replace the dynamics with the one-link pendulum.

All right. So if you can see this, it's just a matter of saying u is C times x-dot. In my MATLAB code and in lecture, I'll use x to mean the combination of q and q-dot. I can just do my exact feedback linear trick-- u is C plus G.

Let's see if I can make this a little better. And there's the equations of a simple pendulum with a little damping. In my control system, if I say, lecture1-- I think I put under simple pend, then suddenly, my two-link pendulum-- the dynamics of my two-link pendulum, when I'm simulating those entire dynamics, work out to be the dynamics of my one-link pendulum. So it's maybe not a useful trick. If I really wanted a one-link pendulum, I could have done a one-link pendulum.

Let's say I want to do something more clever maybe. Let's invert gravity. Let's take my inverted pendulum problem and make it work by just replacing the dynamics of my pendulum with an upside-down pendulum. So maybe if I want to just get the pendulum to the top, let's just make it act like an upside-down pendulum. So we can do that too. Woop.

When I say it the way I'm saying it, I hope it sounds like, of course, if the system's feedback linearizable, you can do whatever you want. It's easy. It's not worth thinking about these kind of things. I mean, that's what I'm trying to communicate. But almost every cool robot that works because of these kind of tricks. They're hidden, but they're there. A lot of the reason robotic arms work as well as they do is because you can do this.

Now there's limits, right? You can only do this if you have unlimited torque. In practice, a lot of robotic arms have almost unlimited torque to give. They've got big gearboxes, right? You'd be surprised how pervasive this idea is.

So what this class is about is what happens if you can't do that? All right. So let's take our hour two-link arm. How are we going to break it? How are we going to make it so we can't do that anymore? What's a more interesting problem?

AUDIENCE: Get rid of one motor.

RUSS TEDRAKE: Get rid of a motor. Let's get rid of the shoulder motor. That seems like an important one. Let's see what happens if we take that right out of there. So the equations of motion actually stay exactly the same, except for now, B of q is going to have to be smaller if u is now just one-dimensional. I've got a single control input. And B of q is just going to be what size?

AUDIENCE: 1 by 1?

RUSS TEDRAKE: It's going to be-- it's got to get to a two-dimensional thing. So it's going to be a 2 by 1. And let's say if as I drew it, that 2 by 1 is going to have nothing to do to the shoulder motor. If I assume the first one is the shoulder, it's going to have direct control of the elbow. Suddenly, it's a whole different game. Turns out, you can still solve that problem. I wasn't thinking of showing this. But let me preview something to come quickly here.

This is exactly that system. It's a system we're going to talk about. It's called the Acrobot. It's got inertia in the links instead of the mass. And if you take these computer science techniques I'm going to tell you about, then you can, for instance, find a solution for the torque at the elbow to try to make this thing go to the top.

If you think about it, it's actually-- it's called the Acrobot because it's like an acrobat on the high bar, where you don't have-- you can only give a little bit of torque at the wrist. You can do a lot with your waist, potentially. So this, if you do a clever job, you can actually pump up energy and swing up and get to the top. But that's a lot harder problem. And I can't write that down in a single board here at 72-point font. But we're going to do that very, very soon.

So I hope you know what underactuated means now. Why would I care about a system that's missing its shoulder motor? That seems pretty arbitrary. If I'm building a robot, I might as well order enough motors to put them everywhere.

It turns out if you care about walking robots, one of the simplest models of a walking robot is called the compass gait robot. It's got a mass at the hip. It's got two legs. We can even assume it's got a pin joint here. That's the connection to the-- that's the foot on the ground. And it's got to torque to give here at the hip.

But it can't apply torque to the ground. It's not because it's not an artificial. If I had a foot, then suddenly, my toe-- somewhere, you're not bolted to the ground. So you've got a bunch of interesting links, and you can apply torque between your links.

But the place where you might want it the most-- your shoulder motor, your elbow motor, whatever it is-- the place that connects you to the ground, you don't have a motor. And you can't have a motor unless you're willing to stick yourself to the ground. Suction cups are a viable thing for walking robots, I guess. But the more interesting problem is how do you do control if you don't have to be stuck to the ground?

So that two-link simple point mass thing is actually exactly the dynamics of the compass gait walker that we'll talk about fairly soon. OK so I've got no torque here. Torque equals 0 there. Every walking robot is underactuated.

The same thing is true. If I'm a humanoid, I'm trying to control all of my state variable. That's the question, right? Do I have enough motors to instantaneously affect every state variable? That's the question. If you count the number of motors on me, it's a lot. They might not be as strong as they used to be. But they're there. There's a lot of them. If you count the number of degrees of freedom, that's hard too.

But no matter what your count, your tally adds up to, if I jump, when I'm up in the air-- I'm not going to do that for you. But when I'm up in the air, none of those motors, no matter what I do, I could do something with my arms, whatever, ignoring aerodynamic forces. None of those motors are going to change the trajectory of my center of mass. There's nothing I can do to change the trajectory of my center of mass.

I can move relative to my center mass, change my angle of momentum. I can serve angular momentum. I can move things around. But nothing I can do is going to move my center mass. A walking robot-- a jumping robot, for sure, is underactuated. A flying machine is underactuated. I mean, fighter jets are a good example. You can go that way pretty well. You know, they don't go backwards so well, for instance.

All right, they don't go directly up so well. Although, I can show you videos of that kind of thing. birds-- you name it. These systems tend to have control variables that you're not in complete control of. Manipulation-- if I'm throwing this chalk around, I don't have complete control of that chalk. If I form a force closure with it, then you can maybe start thinking I'm a fully actuated system. I can move this thing around. That's fine. But I think the interesting part of manipulation is before you get that force closure.

OK. So every interesting problem in robotics is underactuated. I'm going to give a quick sketch of what the rest of the term has for you. And then we're actually-- we're going to try something new on the website. So the website's going to contain everything. After today, we're a paperless existence. The website will have your problem sets. It'll have the lecture notes. You can submit your problems that's on the website.

We're also going to try a new thing. When I post the PDFs of the problem set, you'll be able to download them and print it out if you like. But you'll also be able to use this sort of interactive PDF viewer where people, instead of having a forum or something on the website, you could go right into the PDF and mark, say, I don't understand what this means.

You can choose whether it's anonymous. You can choose whether everybody knows who said it. You can choose if it's just private. In just a minute, I'll show us a demo of that. We'll see if it works. And it might be a cool way to communicate outside of the room.

But let me tell you-- let me forecast what's coming here. I haven't actually told you why this is a computer science class yet. So I can't let you leave without that. Here's roughly what we're doing. On Thursday, we're going to talk about the simple pendulum. So we talked about a two-link pendulum just now.

We're going to take a step backwards on Thursday. We're going to talk about the dynamics of a simple pendulum. But we're going to talk about everything there is to know about the simple pendulum. And we're going to really think about the nonlinear dynamics and how to think about that.

And then we're going to think about how to control that simple pendulum. But as we go in the class, we're going to get to more and more interesting systems. We're going to get to a cart pull system. These are some of the model systems in underactuated robotics. We're going to get to the Acrobot system I just showed you, a two-link thing with a torque here and no torque there. This one has a force here. We're going to think about the toy systems for underactuated robotics.

And then we're going to start splintering into different domains. If we care about walking, then we can start thinking about these compass gait-type robots. And we'll talk about more complicated robots. And the key difference between here and here is just we added a few extra degrees of freedom. Here to here, the dimensionality of walking robots isn't actually necessarily that high. But what happens is you have to think about systems with impacts. And you have to think about limit cycles. We'll develop some of those tools.

OK. And then we're going to think about how do you get away from these toy systems by making higher-dimensional systems? And it can come from walking too. We'll have, for instance, multi-link robots. And think about how to control the higher-dimensional systems, more Degrees Of Freedom-- DOFs. Then we're going to think about what happens if I take these model systems and add some uncertainty or stochasticity.

So a toy example for that might be a walking robot walking on rough terrain, let's say. And then we're going to think about how to take these model systems and what happens if we don't know the model. And that's certainly the case if you've got a little perching airplane, for instance, or a little robotic bird. I have a two-year-old daughter. And I've started being asked to cartoon everything I say. So I'll subject you to some very bad but quick cartoons.

OK. So that's the systems we're going to think about and the reasons that they're interesting. Turns out, we're going to take a very computational approach to them. So in this system, we're going to start introducing optimal control. We're going to say, let's say, I want to get the pendulum to the top, but I want to do it, for instance, by minimizing energy or minimizing the time I get there. So we're going to talk about optimal control. And as much as we can, we're going to talk about analytical optimal control.

But pretty quick, we're going to run out of things we can do analytically. And we're going to start looking at numerical optimal control-- computer science, again, based on dynamic programming. And that's going to get us somewhere.

When we start taking these slightly more interesting systems like this, we're going to develop some better tools. We're going to do numerical optimal control with something called policy search, which is a combination of tools from reinforcement learning, machine learning, and numerical optimization. We'll be able to do some of our impact modeling with that too, I guess.

When we start getting into higher and higher dimensional systems, we're going have to give up on the opportunity to completely solve an optimal control problem numerically or analytically. And we're going to start talking about approximate policy search and motion planning.

And I'm drawing it like this because I want you to see that we're taking a very spiral course through the class. We're going to develop tools that every time I develop a new tool, we're going to make sure we understand what the heck they do to the pendulum, the cart pull, and things like that, and work back up.

So we're going to cover motion planning. If you know randomized motion planning-- RRTs feedback motion planning-- you're going to see that here. And then when you get into the really good stuff here, when you've got uncertainty, stochasticity, and unknown models, then we're going to have to get into pure machine learning approaches in some cases. That looks just like my yellow one. Control based on reinforcement learning, for instance. And that's how we're going to address some of these systems that are more complicated still.

OK. So we're going to route everything in mechanical systems because that's what I care about. I want things to move. But we're going to do it in a pretty computer science-y way because I think the computer scientists have turned a corner, and they're going to solve all these problems.