MITOCW | Lecture 3 | MIT 6.832 Underactuated Robotics, Spring 2009

RUSS TEDRAKE: OK, so last time we talked about non-linear dynamics. We drew phase plots. We talked about basins of attraction. We talked about fixed points. And we hinted at control, and I tried to motivate control not as some nice matrix manipulation of equations, but actually by thinking about phase plots and saying, you're going to move that phase plot a little bit. You're going to reshape it in order to bend the system to your will right-- but just a little bend. You're only allowed a little building in this class.

OK, so today we're going to make good on that idea, but we're going to do it on an even simpler system first, just today. We're going to do it on the double integrator, which is q double dot equals u-- because here I can do everything analytically on the board. If you want a physical interpretation of that-- which I always like-- you can think of this as a brick of unit mass on ice, where you provide as a control input a force, like this.

[INAUDIBLE] force equals u, and there's no friction, and mass equals 1. What we're going to try to do with this double integrator is roughly, we're going to try to drive it to some-- to the origin. We're going to try to drive it to zero position-- I guess that's negative x in this picture-- and with 0 velocity.

It turns out there's lots of ways to do that. And the goal here is to make you think about ways to do that that involve invoking optimality, because that's going to be our computational crutch for the rest of the term. OK. I've been trying to bring the tools from the different disciplines all together. So let me start by doing just a quick pole placement analysis, for those of you that don't think about poles and linear systems that much.

So if I want to write the-- a state space form this equation-- again, I've always tried to use q just to be my coordinates, and I'll use x to be my state vector. So a state space form of this is going to use vector x to be, in this case, q and q dot. And that dynamics there is the simplest state space form you're going to see, but a state space linear equation will have the form Ax plus Bu. In our case, it's going to be the trivial 0,1' 0, 0; and x plus 0, 1 times u. OK, it's not going to get easier than that, but we're going to use that form, because that's going to help.

OK, our goal now is to design u. We want to come up with a control action u-- which you can think of as being a force on the brick, let's say-- which drives the system to 0. So in general, our goal is to design some feedback law-- I use pi for my control policies-- which is a function of x.

Let's start by doing the linear thing. Let's start with considering [INAUDIBLE] of the form of negative kx, where k is a matrix. Well, actually, what is k in this case?

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: 1 by 2, right? So it's going to be k1, k2 times x, which is my q, q dot-- equivalent of saying negative k1q minus k2 q dot. So many of you will recognize this as a proportional derivative controller form.

OK, so if I take this u equals negative x and I start thinking about what that-- if I change k, what happens to my control system? That's easy to do in linear systems. So if I stick that gain matrix in, then what I get is a closed loop system, which is A minus-- sorry-- minus Bk x, which is just the system 0, 1; negative k1, negative k2; x.

OK, and if you've had a class on differential equations, you know how to solve that. The solution uses the eigenvalues of the system. You can quickly take the eigenvalues of that matrix. Characteristic equation out to be k squared minus 4k1 over 2, with eigenvectors-- v1 is this, v2 is this. That's just the eigenvalues and eigenvectors of this matrix. So what are the conditions on the eigenvalues to make sure the system's stable?

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: [INAUDIBLE] both negative. Potentially, we care about whether the system has any oscillations or not, which manifest themselves and whether that's-- whether the thing's complex-- [INAUDIBLE] complex eigenvalues. This is all things you've seen in plenty of classes, but the only way it's going to be complex is if this thing goes negative, right?

OK. So we want a couple of things. We want both of them to be-- both of these to be less than 0, which we can get pretty easily. And we want k2 squared to be bigger than 4k1. k2 squared is bigger than 4k1-- then the system is actually overdamped. If it equals 4k1, it's critically damped. And it's underdamped if it's less than 4k1. For stability, we want lambda 1 and 2 to be less than 0 for stability.

OK, so just to connect this to the phase plots we were talking about yesterday, you've seen-- you might have seen phase plots first in this context, in the linear systems. context . The reason to do this eigenvalue decomposition is that you have these beautiful graphical interpretations of the solutions to the system. Let's choose a particular case.

What did I pick? So let's say k1 is 1. That means I'm going to want to think about an overdamped system. I want k2 to be at least greater than 2. So I'm going to choose k2 equals 4. If I do that, then my eigenvalues work out to be negative 4 plus or minus 16 minus 4 over 2, which is negative 2 plus or minus the square root of 3. Square root of 3 is about 1.75. So I get negative 0.25 and negative 3.75 for my eigenvalues.

OK. And the eigenvectors are going to be just this form. So what that allows me to do is make my same state space plots we were making yesterday where now I have q and q dot. And my first eigenvector is going to be 1, negative 0.25.

I'll use these as quarters. So I go minus 0.25 [INAUDIBLE] 1 here, so I get a line that looks like this. That's v1. And I get a line that goes almost 1, negative 4-- so a little bit under that here. v2 is over here.

OK. And the eigenvalues on this-- if you've seen these plots before, we typically [INAUDIBLE] They're both negative, so we're going to draw an arrow like this. Systems-- initial conditions that start on this will just get smaller. Initial conditions that start on this will also get smaller, but they can actually get smaller a lot quicker.

Just to say a few of the subtleties, so I an overdamp system so I don't have repeated eigenvalues. I chose a overdamped system so I don't have oscillations, because I can make the same plots. But for the overdamped system, this is a great way to think about things.

When I don't have repeated eigenvalues, the-- any initial condition of the system can be written as a linear combination. That means that, when the system doesn't have repeated eigenvalues, the eigenvectors span the space. Any initial conditions can be written as a linear combination of the two eigenvalues.

And the dynamics from this point are just the exponential dynamics on the-- of the two components. I don't know. Tell me if you want me to say that again in a more-- it's not really the focus, but if you understand that, you got the whole thing here. So what it means is you could take any initial condition-- it turns out, any initial condition when I have eigen vectors which span the space. I'm guaranteed to have that, if I have unique eigenvalues.

Then I can write this as a combination of these vectors. I've got one component like this and one component like this. This initial condition-- if I say this is x0-- so I can say x0 is alpha 1 times v1 plus alpha 2 times v2.

We know that initial conditions that are just on the line v1 go e to the negative lambda-- or e to the lambda t v1, so the whole system goes alpha 1 e to the negative lamb-- oh, sorry. I've got negative eigenvalues-- to the lambda t v1 plus alpha 2 e to the lambda 2 t v2. That's the great thing about all these linear systems.

So what that means is, if I've drawn the eigenvectors, then I know exactly the entire phase plot of the system. So we're connecting back to the pendulum. I went through in all the different places. I thought about what the contributions were and mapped it out. Here I don't have to do that. I know that this system is going to-- the component-- the space that has-- in this eigenvalue is going to decay quickly towards 0.

And it's going to decay faster than this one, which means an initial condition like this is going to go like this. I should have used one of my many other colors to do that. Looks like a blue-- so trajectories from that initial condition are going to do that. And trajectories from this initial condition are going to-- what are they going to do, if I start over here?

They're going to go mostly down, and eventually we think it's stable, so it's going to get to the origin. I can even do my filled in circle there. When are they going to start bending towards the origin?

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: Later-- they have to pass this before-- they have to pass that and get the negative velocities before they can hook back in. So the trajectories look like this, and go in. Yeah? And likewise, so all these trajectories-- you can map out the entire phase portrait of the space pretty quickly just by understanding the eigenvalues and eigenvectors. Same thing's true over here.

OK, so there's another example of a phase plot that we have. In the linear systems, it works out to be clean. You can just do these eigenvalues, eigenvectors. OK, now, control-- we're allowed to change k1 and k2. Changing k1 and k2 is going to change the phase portrait. It's going to change those vectors. I want to change them to make it do whatever I want. The first discussion is, what do we want to make it do?

Maybe even before that, I should observe that, without thinking about optimality at all, it would be easy to stop here, because I-- if I look at this carefully, as long as I choose k squared-- k2 squared greater than 4, K1. I know I'm going to not oscillate. And I can just start driving k1-- and correspondingly, k2-- up as high as I like, and make the system get to the origin as fast as I want, and it won't oscillate. Why not just drive them all the way to infinity?

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: You can't-- don't have a motive to do that. That's the first unsatisfying thing-- absolutely. Probably there's some unmodeled thing. Even if I did have a motive that could do that, there's probably some unmodeled things that I might excite, and cause bad things to happen anyways.

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: You could melt the ice and it'll break. That's right. That's right. I guess I could have said wheels, and then maybe they'd melt the tires. OK. And you can see that here, actually. Remember? What is the unactuated phase plot of the system look like? I can just draw that. If u was just uniformally 0, if k1 and k2 were 0, what would the phase have looked like there?

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: It would have just been x dot equals A of x, where A is this guy. So it wouldn't have been as interesting. Every point would have just had a vector like this. It would have been a little bigger with bigger velocities, but it's just-- it would just be a flow like that, which I hope is what you'd expect it to do, since it's an integrator. Things are just going to-- off into the ether.

OK. So if I consider-- I started with this, and I'm getting out things that look like this, I'm already-- in my unitless cartoon here, it's sort of already looks like I'm using a lot of torque to do what I'm doing. I'm using a lot of force. I'm really significantly changing those dynamics in order to bend this thing to come around like that. That's OK, but we can do better.

So today I want to use this system, which I think it's quite easy to have strong intuition for, to start designing optimal feedback controllers. So let's address the we don't have infinite torque problem first.

One more comment on this-- I didn't actually call them poles-- there's a pole placement version of this too. It's exactly the same thing. If you were to draw a root locus, what would the system look like? The typical root locus would be you're multiplying the entire feedback by some linear term. You're not scaling them in some squared law, so what you get is, for very small feedback gains, you get oscillations.

As you crank it up, the poles connect in the left half plane, and then they separate. And as I keep turning up my gains, one of them creeps towards the origin. The other one goes off really far to infinity. So just those of you think about poles and zeros, this is exactly the same way to say that. I didn't do a root locus because I was changing two parameters, but it all connects.

OK, so now, let's say I have a hard constraint on what u I can provide. Let's just say that I have an additional constraint that's, let's say, the absolute value of u has got to be less than 1. Well, that changes a lot of things.

My linear system analysis is impoverished now. If you want a graphical version of what that's doing, that's-- my zero input looked like that. I wanted to go like this with my linear controller, but maybe it's capped at something like this. OK, so what's that going to do to your system?

If I just ran the policy u is some saturation, say, on negative kx, I took my same feedback-- linear feedback controller and I just said, if it's greater than 1 it's 1; if it's less than negative 1, it's negative 1, I think you're still OK. Trajectories are still going to get to the origin. They might take fairly long routes to the origin.

You're not going to lose stability in this case because of that, but it starts to feel like, man, I should really be thinking about those hard constraints when I design my controller. All right. So how do we do that? One way to do that is optimal control. It's not the only way to do it. Let's formulate an optimal control problem.

Let me sync up with my notes so I don't go too far afield. OK. Let's say my goal in life is to get to the origin as fast as possible in minimum time. But I'm subject to this constraint. So that's the famous minimum time problem-- subject to that constraint. OK.

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: Yes. What do we want? Both the position and the velocity to be 0. Turns out you need this constraint for it to be a well-posed problem. If I didn't have constraints on u, then, like I said, I would just use as much u as possible. I would get there infinitely fast, and we haven't learned a whole lot.

There are other ways to penalize u or something like that, but we're going to put a hard constraint on it here. OK, now, muster all your intuition about bricks and ice and tell me-- if I've got limited force to give and I want to get to the origin as fast as possible, what should I do?

AUDIENCE: Bang-bang.

RUSS TEDRAKE: Bang-bang. Good. He knows the answer. What should I do? People haven't thought about it and don't know bang-bang is.

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: Right. Right. So if I want to get there as fast as possible, I'm going to hit the accelerator, go as fast as I can until some critical point, where I'm going to hit the brakes. And I'm going to skid stop right into the goal. There's nothing better I can do than that.

We're going to prove that, but I want to see-- that's a fairly complicated thing. It's something you can guess for the double integrator. You can't guess for a walking robot, for instance. But we want to get that out of some machinery that's going to be more general than double integrators.

OK, so the proposition was bang-bang control. You might hear people casually say, bang-bang control's optimal, and that is-- if you have hard limits on your actuators, it's very common that the best thing to do is to be at those limits all the time.

If that's the way you've defined the problem, bang-bang control solutions are pretty ubiquitous. They don't always work that well in real robots, because actuators don't like to produce zero-- infinite force and then-- or max force and then negative max force within a single time step.

Good-- OK, so I think the only subtle part about it is figuring out when I need to switch from hitting the gas to hitting the brakes. So let's see if we can figure that out first. I think a pretty good way to do it is to think about what happens if you hit the brakes. And then you want to hit the brakes and arrive directly at the goal.

There's only going to be a handful of states from which, if I was going at some-- if I was some position and some velocity and I hit the brakes full now, I'm going to land exactly at the goal. Let's see if we can figure out that set of states first.

Let's think about the case where q is greater than 0 first. Just pick a side. So in that case, hitting the brakes-- is that positive 1 or negative 1?

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: Negative 1-- no, it's positive 1. You almost got me. If q is greater than 0, it's positive. q is greater than 0-- then u is positive. I want to be pushing back in the direction I'm already coming from, so u is positive 1.

All right, so now, we're going to have some math ahead of us. See if we can integrate this equation. I can do that on the board for you. q dot of t-- I better get it right.

[LAUGHTER]

ut-- so in this case, it was 1-- plus q dot of 0. I'll just make it a little bit more [INAUDIBLE] This case, it was 1, and q double dot of t is-- sorry-- switch orders-- q0 plus q dot 0 t plus 1/2 ut squared. OK, I want to figure out--

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: Did I screw it up? What?

AUDIENCE: [INAUDIBLE]

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: Oh, sorry. Sorry. Thank you-- good. Thank you. OK, so let's figure out, if u is 1, what trajectories are going to get me so that q-- at some t final, qt and q dot of t are 0-- simple enough-- little manipulation. So it turns out I'm going to solve for q0 and q dot 0.

So q dot 0-- looks like that's going to be negative u of t. It's a little bit weird, my notation here. I'm saying that the initial conditions are moving backwards. The equations are simple enough. I hope it's OK. And q0 t had better be-- it turns out to be 1/2 ut squared. So q dot of t is negative ut. Add those together. q0 is going to be 1/2 ut squared.

If I solve for t-- solve out t-- in this case, u is 1 so t, say, is just negative q dot. So q of 0 is just 1/2-- let's keep that. This is just 1 t squared q dot 0 squared. If I plot that, what I've got in my state space-- q, q dot-- is a manifold of solutions, which starts at 0.

And then I said that I did this for u is positive. And it goes like this. This one's not a solution. Where did I get that out? And one of my assumptions here when I inverted t or something that I-- that solution disappears. You can't have negative time.

In fact, in my notes, I did it. I solved for the other t, which would have been better. Sorry. OK, so there's a line of solutions here-- which, if I started this q-- this is actually the positive queue, negative q velocities. I hit the brakes. I go coasting into the stop at the origin.

Turns out, if I do-- if I think about the negative q case, I get a similar line-- similar curve. [INAUDIBLE] quadratic curve over here. You know what-- let me be a little bit more careful. Let me make that one pink, because this is the now u is negative 1 case.

Good. We figured out the line of solutions where, if I hit the brakes, they get to the origin. Harness your intuition again. What do I do if I'm here?

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: Right. This was the stopping all the way to the goal. So pretty much, from anywhere else, I want to accelerate. So what does accelerating look like when I'm here?

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: It's going to put me going up. And what happens is, any time I'm below this curve, I'm going to drive myself up. I can't go backwards like that and drive myself up, hit that, and ride it in. And if I'm above the curve, what do I do?

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: Have to overshoot a little bit-- I can't bend down more than this, so I'm going to ride it all the way over to here, connect up to this surface, and ride it in. And it turns out, any time I'm over here, the best thing to do is to-- did I get my colors wrong? Got my colors wrong-- let me fix that. Sorry. It's confusing.

This is the accelerate, and then break. And this is the break. Let me just recolor it for you to make it a little more clear. Sorry about that. So let's say I'm pink over here, blue over here. OK.

I want to be pink. I want to decelerate just like this if I'm above it because I want to take these curves that are almost there. If I've got extra time, I'm going to accelerate to the point where I decelerate again, so down here should be blue. And then this is, again, the case where I decelerate as much as I can until I take the pink line.

This was the u equals negative 1, and this was the u equals positive 1. OK. Is that at all satisfying? We can now connect this back again to your phase plot pictures. We had our initial lines that looked like this. [INAUDIBLE] allowed to apply a bounded amount of torque. So the best thing I can do, if I'm right here, is I can warp this thing down to the point where I get right there.

And if I'm here, I can warp it up to push me here, and then ride it down. The hard part is actually showing that that's optimal. And the reason I'm going to go through it is because it's-- it forms the basis for all the algorithms that are going to be more general. So let me show you that that's optimal. To do that, I need to introduce our first optimality ideas for the course. Are people OK with the-- that picture?

AUDIENCE: [INAUDIBLE] below the line.

RUSS TEDRAKE: Mm-hmm.

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: So tell me where. Bottom right? OK. So this is the place where, if I decelerate, I get to the origin. If I'm here, then I have a little bit more velocity in the same position. So if I hit the brakes, I'm not going to stop in time.

I don't quite decelerate fast enough to get here, because there's limited torque, so I just slip past it until my chance to come in the other way again. The only separation is that curve.

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: Everywhere up here, you want to be-- top left you said?

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: Here, you're blue. The same way, you want to accelerate as much as you can, because you want to get to the place where you have to hit the brakes. This is me. I'm at some position. I don't have enough velocity that I have to just hit my brakes, so I'm going to gun it until I'm at the velocity where I just have to hit my brakes, and then ride it in.

Up here, I'm at the point where I have too much velocity. Even if I hit my brakes, I'm still going to overshoot a little bit, which means I'm going to have to-- and so you could think of it as now-- the word brakes maybe flips when you flip that cross across. Maybe that's the right thing. But the action I take is only changing based on that switching surface-- which, as you know, will be a nightmare for a lot of our reinforcement learning algorithms. This is a hard cusp.

So if you have a control system that has a hard non-linearity like this, which is-- I'm doing one thing here, and I'm immediately, at some discrete place, doing a different thing-- that's a very non-linear event. And it's hard to get analytically when you're doing something more complicated than a double integrator, and it can be hard to get computationally. But we'll talk about that when the time comes.

OK, so how the heck do I make an optimality argument about this? I want to introduce Pontryagin's minimum principle. OK.

This is going to be a load of equations real quick, and we're going to tease them out. In general, optimal control problems are going to go-- are going to be formulated by designing a cost function. That cost function is some scalar quantity that I want to minimize.

I'm going to use the symbols g of x you as an instantaneous cost function. I'll use h of x to mean final costs. I'm going to show you what this means in a second. And I'm going to use J of x to be a cost-to-go.

It's very important-- all of these are scalars-- not vectors at all, just a scalar quantity. So a typical optimal control problem will be formulated as something like h of x capital T plus integral from 0 to T g of x u dt, subject to x dot of t is f of xu. Let's say the dynamics and x0 t is some-- let me it call it x0 here-- x0.

This is a general cost function-- cost-to-go function form for optimal control. We're going to use it a lot. There's just a couple of things to note about it. So just to get some intuition, so g of xu-- that's things I'm penalizing throughout the trajectory. So maybe I want to do small actions in general, in which case, I could put some term in here, which penalizes me for having a u. I could put a u squared in here or something like that.

Or maybe I want to just worry about being a long way from the origin. Maybe I'll put an x squared in here or something like this. h is a final cost. It's some function that only penalizes the final state of the system. So maybe I don't care what I'm doing for the first capital T seconds, but at time capital T, I want to penalize it for being away from 0-- x squared here or something like that.

There's lots of different forms. The only thing that's really important to note about this, the only really restriction in the forms that you can play with, is that we do tend to assume this form, which is additive. It's integrable-- integrates some scalar cost g. So I don't look at multiplicative contributions of-- from x at time 1 and x time 4 or something like that. I'm only looking at additive cost functions. Assuming that form-- that additive cost form will make all the derivations work, roughly.

OK, so for the minimum time problem, what is that form? You could formulate it a couple of different ways. In this case, I could actually have g of xu equals 1, and have capital T defined as the time, and have h of x equal 0. That's a perfectly reasonable optimal control formulation. So it certainly fits in this general optimal control form.

OK, so now we need to know how to-- we've got this guess. I'm going to leave that hard-earned picture up there. I like this one too, but-- let me just say what Pontryagin's minimum principle is first, and then we'll make sure it makes sense.

So for this general form, J of x is h of xT plus integral from 0 to T g x, u dt, subject to-- and I'm going to try to be very careful about writing these all every time.

Let's assume for-- to begin with, capital T is fixed, just a parameter somebody chose. Let's say u is bounded to some set u. In our problem right now, it was negative 1 to 1, right? The minimum principle goes like this.

We're going to define this new auxiliary function, the Hamiltonian, capital H. If I have found some optimal control solution-- I'll think of it in terms of-- the solution right now in terms of a trajectory, which is some sequence x star of t, u star of t. Then it must satisfy the following conditions.

First of all, we know it must satisfy f of x star u star. That was already one of our conditions. And it has to satisfy the-- OK. There's a significantly less trivial one, which is that p dot of t has got to equal to negative partial h, partial x evaluated at x star, u star, p-- which, if h is what I had up there, works out to be partial g, partial x, plus partial f, partial x. Transpose p.

And this auxillary variable that we had has to be the gradient of partial h evaluated x star t. One last condition-- this is 1, 2, 3. u star t had better be the argmin over u of h of x star, u, p. OK. Sorry. Cut that out. We're going to make sense of it now.

OK, before we derive it-- and I'll just do a sketch of the derivation-- but before we derive it, let's just think about the implications. First of all, this says the optimal control trajectory must satisfy, which means it's a necessary condition for optimality.

If I found some optimal trajectory x star, u star, some trajectory x, u, I can verify that-- a necessary condition is that all these things are hold, but that's actually not a sufficient condition in general. For linear systems that are convex-- linear dynamics that are convex and the cost function, it turns out it's OK, but in general, it's not always sufficient.

And it says that, if I take my x and I integrate it forward in time, solving x by integrating my dynamics forward, and then I take this other function-- this new set of variables p, which happens to have the same size as x-- we'll see that-- and integrate it effectively backwards in time, because I have final condition on p-- if I do both of those things, and I can write down u as being the argument of what's left-- h, x-- then I've satisfied a necessary condition for optimality. Let's try to make sense of that.

How many people have done optimization before at all? How many people have seen Lagrange multipliers before? OK, good-- so let me say a few things but not dwell. And there's a lot of information in the notes-- as fast as I can type.

All right, in general, what I'm trying to do is I'm trying to minimize some function. In this case, I'm trying to minimize J. I'm trying to minimize this J of x by finding the u [INAUDIBLE] which minimizes then. But let's make it a little simpler just to make sure we get the basic idea.

Let me just say J is some function of some parameter alpha. I'm trying to minimize J-- I can even do it even simpler. Let's just say minimize over x J of x. So if I have some function of x-- J of x looks like this-- I want to find the minimum.

The first condition, the necessary condition, is that, at the minimum, the derivative of that thing better be 0. So I can check by just checking if partial J, partial x equals 0, that I've got a necessary condition for a minimum.

That's actually a lot of it. The second part is the Lagrange multiplier part. Let's say x is a vector now-- a two-dimensional vector. Let's say I want to do the optimization min J of x, subject to the constraint that x1 equals x2-- or let's do something slightly more interesting. x1 plus x2 is 3.

Turns out, thanks to the method of Lagrange-- one of his many methods-- solving this problem is no more difficult than solving this problem-- finding necessary conditions for this problem. By just making an augmented function, you can now minimize x and lambda of J of x plus lambda times this constraint-- which, in this case, is x1 plus x2 minus 3-- has to equal 0.

It turns out, if partial J, partial lambda equals 0, then that means the constraint is enforced. Partial J, partial lambda in this is x1 plus x2 minus 3. If that equals 0, which is the condition I'm looking for anyways for the minimum, then I've now-- not only have I satisfied my constraint, but the remaining minima-- the minimization of this is this constrained solution to that optimization. The Lagrange multiplier method is very, very useful. If you don't know it, look it up. It's very good. Yeah?

AUDIENCE: So in the partial J, partial lambda, that J [INAUDIBLE] partial is this new J--

RUSS TEDRAKE: Oh, sorry. Thank you. Thank you. Yep. Good catch, good catch-- thank you. Partial of-- I don't know-- that whole thing-- partial lambda-- thank you-- good catch. And in the method of Lagrange multipliers, lambda has an interpretation of a constraint force.

What you're about to see is that all I'm saying in Pontryagin's minimum principle-- which is an absolute staple in optimal control-- all I'm saying is that J of x is-- which is my cost function-- my cost-to-go function-- is at a stationary point with a Lagrange multiplier which enforces this dynamic. And that Lagrange multiplier happens to p. OK? So let's just see how that plays out.

OK, so this is a sketch of the derivation of Pontryagin's minimum principle, which, I think-- I'm just going to do enough so you see where those things are and have some intuition about them-- a sketch of it based on the calculus of variations. So there's many other ways to do it-- calculus of variations, which is a scary name for a very simple thing.

This is the problem solving-- J of x0 is h of x plus integral over g, subject to those constraints. So how do I write that in terms of a Lagrange multiplier? I'm going to do a second function, which I won't make the mistake of calling J again here. Let's call it S. Some function S is going to be h of xT plus the integral from 0 to T of g of xt, ut plus some Lagrange multipliers-- p in this case-- times my constraint, which is x dot minus f of xu. I was trying to use T's everywhere. [INAUDIBLE]

OK, so now I can explicitly try to find the place where S-- which is my Lagrange multiplayer version of my problem, which has the explicit cost that I'm trying to minimize-- subject to the constraints that x better be-- satisfy my dynamics-- exactly the same as that two-second Lagrange multiplayer introduction.

Now, getting it right is a little bit funny. So S is now-- you could think of S as a functional [INAUDIBLE] a function of functions. If I take a variation, this is just-- it's going to be exactly like your basic calculus, but the calculus of variations uses these symbols for a variation on a function is just going to be partial h, partial x times the variation in x of T plus the integral 0 dt

Notice quickly that this thing inside here is just h. That's my Hamiltonian. So that thing inside there I can just-- OK, this says this is a variational analysis of S that says, if my function changes by some small amount in x tilde, this is the result of-- in changing S-- in S. Similarly, if my thing changes by a little bit in xt, or in ut, or pt, or in all of them simultaneously, this tells me what the variation's going to be in S.

The stationary conditions then-- if I'm at an optima, what I care about is that-- if I change u a little bit, if I change x a little bit, if I change p a little bit, S better not change. That's my condition-- my necessary condition for optimality.

If partial h, partial p is 0, then I know that changing p isn't going to change the solution, so I can look for stationary points with respect to p. Partial h, partial p better be 0. What's partial h, partial p? Well, it turns out it's just x dot minus f of xu, which is my forward dynamics. So if I've integrated my system forward in time, then this thing's going to be true, and [INAUDIBLE] steady state with respect to changes in p.

OK, let's look at the changes with respect to x. So to get the contributions from x correct, we first need to worry about this x dot. We don't want to have that x dot floating around in there, so let's integrate by parts to get that out of there. We're going to look at this partial h, partial x-- the variations-- but first, integrate by parts.

The integral of my p of t x dot t dt from 0 to t is just going to be p of T, x of T minus p0, [? x0 ?] minus the integral capital T of p dot t x of t-- I forgot my transpose-- dt-- basic integration by parts. OK, if I now take my variation in partial x, having used that, then what I get is-- which gives me-- which, in this case, was partial g, partial x transpose plus partial f, partial x transpose p of t.

My goal is to show you enough of the derivation that you understand what these terms are, and not so much to get completely bogged down in it. If you want a good treatment, a careful treatment, you should see Bertsekas optimal control book. When I say careful treatment there, it's going to be 5 pages or more at least.

OK, so Pontryagin's minimum principle says that, if my constraint is satisfied to x dot, and if I can just integrate back to this p dot backwards in time from some final conditions-- which are the same basic variation argument that drives this-- then I found Lagrange multipliers which satisfies that constraint. And the final variation says that u had better be the minimum of h of x.

That puts me at a local minima in my constrained optimization-- big pill to swallow, but this is the way we're going to show that the brick solution is optimal. People are much more enthusiastic when there's bricks. It's OK. I understand.

OK, so let's turn the crank and use this tool now to see what the heck-- see if we can verify our original bang-bang policy is optimal. So for the bang-- Pontryagin bang-bang double integrator-- what's the [INAUDIBLE] look like for this thing?

g of xu we said was just 1-- and p transpose times x dot minus f of xu. I'll just write it out in elements forms. It's so simple. It's p1 times q dot plus p2 times u. OK?

If we had derived our bang-bang controller just like this, then we could actually immediately say, what's the optimal control solution? If I want to take u star as the argmin, u in negative 1 of 1 of h x, u, p-- and what's it going to be, just looking at what I've got on the board here? This is a good time to make sure you get it.

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: Yeah-- good. So these terms are-- have no impact. If p2 is positive, and I want to minimize this thing, then u better be negative-- as negative as possible. Negative as possible means negative 1. And if p2 is negative, then you should be as positive as possible to minimize that thing.

So it turns out our same policy that we worked hard for in the Pontryagin, in terms of the Lagrange multipliers, works out to just be p2-- sine of p2 of t-- negative sine, sorry. So the sine function is just 1 if it's greater than 0, negative 1 if it's less than 0.

My equations for p-- which, if I didn't use the word adjoint equations yet, I should have-- these equations for p are called the adjoint equations. My equations for p are pretty painless. So p1 dot is negative partial h, partial x1. x1 is q, so-- doesn't appear at all. That's 0. So that Lagrange multiplier isn't going to change at all. That's pretty painless. And p2 dot is negative partial h, partial x2, and that's negative p1 t.

OK, so it turns out that p1-- my Lagrange multiplier's just going to be some constant. And p2-- t is just going to be the interval of that constant-- c2 plus c1 times t. Try and debate how much to squeeze in the next few minutes-- you know what, let's-- let me do it tomorrow for real-- or on Thursday for real, because I don't-- because it's going to take 10 minutes to finish, but it's worth doing it right.

So for homework, to yourself, see if you can work it through. Take the equations that we had and show that these-- those four conditions are satisfied, and I'll spend the first 10 minutes of class doing it properly on Thursday. I don't want to rush through it and have it mean nothing. Sorry. I wrote 3 here. There's a condition-- a final condition on p2-- so three conditions by this number. Awesome.