The following content is provided under a Creative Commons license. Your support will help MIT OpenCourseWare continue to offer high quality educational resources for free. To make a donation or view additional materials from hundreds of MIT courses, visit MIT OpenCourseWare at ocw.mit.edu.

RUSS TEDRAKE: OK, let me say the big idea for LQR trees again, and you can tell me where you want more details. So here's the basic story. I've got some goal I want to get to, and I've got a lot of potential-- I'd like to be able to get there from every initial condition, let's say. The idea was we know how to design-- we know how to stabilize trajectories. So let's just pick a point at random, design a path to the goal like that, design the LTV LQR stabilizer on this.

The great thing about that is, first of all, that it will locally stabilize the trajectory, but second of all, because we can compute not only-- because we are given both u equals some time varying, we have a feedback policy from that, but we're also given an estimate of the cost to go, which is in this time varying matrix, yeah, the Riccati equation backwards. Because of that, when we're doing our LQR design, we actually have a good candidate for a Lyapunov function for the system around that trajectory. So that's an important observation, that when we do LTV LQR on a trajectory, we get both, OK.

Now this thing, for the linear system, even the linear time varying system, this is a true Lyapunov function. From all initial states, this function will just only get smaller with time, OK. But since the actual system is non-linear, as I get further from my trajectory, some of the non-linearity is going to come in and corrupt my Lyapunov function. At some point, when I get far enough from the trajectory, those higher order terms are going to mean that this thing doesn't have a negative time derivative.

But what I care about for a Lyapunov function is that this thing is less than or equal to 0. So the idea with the certificates is to do a higher order polynomial expansion of the dynamics along this trajectory and try to estimate the threshold where this stops being true, OK. And that gives me, essentially, a funnel. If I do it everywhere in time, that gives me a funnel along the trajectory over which I know the LQR cost to go is a Lyapunov function for the nonlinear system.

And it's mostly conservative. The way that we construct that threshold is mostly conservative. The only weakness of it-- meaning the real basin of attraction should be bigger than this estimate. The only weakness is that I'm only doing it by doing a polynomial expansion of the system here. So if there was a hard discontinuity right next to the trajectory that didn't show up in the Taylor expansion, then I wouldn't ever see it.

I'm not exhaustively searching the non-linearity around the trajectory. I'm just saying, along this trajectory, what are the higher order expansion? I do a Taylor expansion-- a third order, fourth order, whatever it takes, and I use that to check when it breaks the cost to go, OK.

So that's the certificate. That's just, if I have a single trajectory, that I can use this. The real cool thing is that thanks to Pablo and [? Sasha ?] [? McGretzky, ?] we can do that efficiently with a convex optimization. And then the idea is, if I can do it for a single trajectory, then why not put that back into sort of an RRT kind of framework and try to build lots of trajectories that are stabilized?

OK, so the first step was just to pick a point at random, design a single trajectory. The second step is, let's pick a new random point. I don't actually have to go all the way back to my goal. I just have to go to the nearest point on the current tree and stabilize that. And then I pick another point, find the nearest point on the tree, build the trajectory in like that, stabilize that.

If I pick a new point and it's already in the basin of attraction, I don't need to add an edge. That would be a waste of my time. So the effect is I get these-- I didn't say carefully what the results are, because I can't prove them yet. This is still hot off the press. But I think that I can say that it probabilistically covers the reachable state space.

Every place that I can get to the goal from, there exists a controller. If I design enough samples, I should be able to get to the goal, given I do enough steps in my LQR tree. So as time goes to infinity, the entire space will be covered with basin of attraction of a controller that gets me to the goal, which is a pretty powerful thing to say, if you're willing to wait till time goes to infinity. And the practical thing that's nice about it is it seems to happen pretty quick with a handful of-- with a fairly small number of trajectories. Yeah? [? John. ?]

AUDIENCE: If you want to guarantee [INAUDIBLE] [? in your ?] system has that, which property would you have [? to let it ?] sample inside the basins of attraction?

RUSS TEDRAKE: Yeah, so that's why I have to qualify the guarantees. I'm only saying the nonlinear system as represented as the Taylor expansion around the trajectories. That's the weakness. So it really only works for smooth systems. If there's a cliff here and I design a trajectory right here, it might come up with [? a ?] saying a basin of attraction's here, even though that's just not true. OK.

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: If I smooth the cliff-- [? sorry, John. ?] If I smooth the cliff and therefore, when I Taylor expand here, I can see that there's a cliff there, then I'd like to think that the certificates would do the right thing. But if it's a hard cliff where the higher order expansion here doesn't see that cliff, then there's no hope.

AUDIENCE: [? If ?] [? you ?] [INAUDIBLE] inside the basin, [? though, ?] couldn't you [INAUDIBLE]?

RUSS TEDRAKE: That's a good question. So it's just building more and more accurate certificates as time goes to infinity. It could be. It could be a good idea.

My suspicion is that, in practice, this is going to be good enough for-- and we're not going to want to do the-- maybe I'm just the kind of guy that cuts the corners, but I think this is a pretty good solution. Ernesto.

AUDIENCE: [INAUDIBLE] [? basin ?] [? of attraction ?] [INAUDIBLE].

RUSS TEDRAKE: So I use-- whenever I design each trajectory, I use [? DR ?] [? call. ?] Why not? But it's so I could say they're locally optimal in the branches that they are, just because that's easy to accomplish. But there's no sense in which it's globally optimal. So I just try to say that every state will eventually get there and not that it'll get there optimally. mm-hmm.

AUDIENCE: But again, if you allowed it to sample inside the basins of attraction, you would probabilistically get some kind of convergence to global optimal.

RUSS TEDRAKE: There are ways to try to get at that problem. I don't think it's a one-step change from what I said. I think you have to do something more. I mean, any one of these trajectories-- maybe that's getting busy. Any one of those trajectories is just locally optimal.

So let's consider the case where the goal is here. There's something here, and I did this. And for some reason, I got a trajectory like that, which is locally optimal but not globally optimal, because there's a shorter path here.

So I need to somehow-- I would need to somehow include the mechanism, which is, I think, what you're getting at, to eventually find a different path to the same place and overwrite the old path. And I don't have any mechanism for that inside. And it turns out to-- that's actually what I was going for initially. Turns out to be a little bit nontrivial to do. Chris Atkeson has a nice idea that goes towards doing that a little bit more on composing trajectory libraries. But I gave up optimality and tried to get coverage with a basin of attraction in that design.

So some of you have been talking to me about the projects and have been asking about these things. So I think Mark and Matt are thinking about trying to help with this-- I mean, this is new. I mean, you guys can definitely help with the idea.

So we're trying to figure out, for instance, if there's actuator limits. These guys are talking about trying to figure out how to compute the certificates, and even the LQR stabilizer, if you not only have a quadratic [INAUDIBLE] on u but have some hard limits on u, because that's a problem we actually hit in practice. So one of the final projects in the class, I think, is going to be helping with that.

There's problems of, how do you design the stabilizer through impacts and in periodic systems. There's lots of good questions. So if you're excited about that idea-- I'm definitely excited about the idea right now-- then I'll see what you wrote up today, but we could maybe try to find a way to connect that. I'd be thrilled to have your help in making that idea more relevant. Yeah?

OK. Does that cover? Now for something completely different, although related, of course. So in the LQR trees and basically in everything that we've done so far in class, we've made a handful of important assumptions. The most important one, probably, is we've always assumed that we have the model of the system.

So next week we're going to get at, how do you do some of these optimizations when you don't even have a model, OK. We've also assumed so far that the current state of the system is known. In other words, the state of the robot is fully observable. If I stop assuming that, we start getting into discussions about state estimation, and we will towards the end of the class.

And there's another assumption we've been making, which is that the dynamics are deterministic, let's say, that pretty much, the system goes where I think it's going to go. Now I chose to write that and then cross it out, because that's not quite what we're assuming, OK. I want to write that.

But maybe what's a-- if we really were assuming that the dynamics were deterministic, then we wouldn't have been spending much time talking about feedback at all. We would have been just focused on open loop things. So let's think. Let's have a short philosophical argument about what we're actually doing, yeah?

So we're not quite assuming that it's deterministic. Maybe a better way to describe what we're doing is we're assuming something specific about the disturbances in the system. We're assuming that disturbances look like a change in initial conditions. That's one way to say it, an un-modeled change in initial conditions.

By virtue of talking about these feedback stabilizers, the motivation is that, OK, I'm following this trajectory. When the model's right, all is good. If something does happen, then OK, it's going to move me somewhere in state space. But as long as that's in the basin of attraction-- the whole idea of really a basin of attraction is going at the idea that I can handle disturbances by just being robust in state.

But that's actually a subtle thing to assume. I want to be a little bit more explicit about what that means. Really, it sort of implies that we're assuming that disturbances are impulsive, not constant, instantaneous, Impulsive.

If a disturbance for my walking robot was someone put a new weight on my leg, then that's not something that our designs so far have handled, because that's more like the model changed. We're talking about something that it moved me, and now I have to deal with it. So if the disturbance lasts for a long time, then that actually feels a lot more like a model change. But if it's impulsive, I can think of it as a change in initial condition.

And the other thing that that implicitly assumes is that the disturbances are rare. If I got impulsive disturbances 1,000 times a second, then, again, my completely deterministic analysis isn't probably the relevant one, OK. So those are-- implicitly, we've been making these assumptions the whole time. The whole idea of talking about basins of attraction of deterministic systems and doing feedback design implicitly makes that assumption.

So there's three things there, the fact that it's un-modeled and impulsive and rare, I guess. If your disturbances are not instantaneous rare disturbances, then I think I would advocate quickly for starting to think about your dynamics as not deterministic dynamics but as a stochastic dynamics, OK. And even if they are impulsive and rare, but if you have a model of them, then you should still be able to do better by explicitly letting your feedback design reason about the stochastic dynamics, OK.

So today I want to start talking about-- I want to start breaking down our assumptions. And throughout the rest of the class, we're going to try to break these down. Let's start breaking down the assumption of deterministic dynamics, OK.

AUDIENCE: [INAUDIBLE] [? or ?] talking about the LTV LQR.

RUSS TEDRAKE: Mm-hmm.

AUDIENCE: [INAUDIBLE] actually saw something, like if we the actually have this [? passive ?] transition model and we move somewhere, and then we're a little bit off from the trajectory, then you look at the policy, which would bring this back [INAUDIBLE] [? projected ?] [? and ?] go back.

RUSS TEDRAKE: I think the answer to your question is yes. So one of the things we're going to do today is show that the-- it's sort of subtle that that's yes. But anybody who knows linear quadratic Gaussian control, yeah, well, it turns out that we're actually doing linear quadratic Gaussian control, but that's a surprising result, OK.

So in the specific case of linear dynamics, quadratic cost, Gaussian noise, then what you said is true. So that's a special case that we'll see quickly. But in general, it's not true.

Right, so and again, this is leading into next week. We're going to start talking about doing these optimal controlled derivations without any model. But today let's think about what happens if we have a stochastic model.

OK, lots of ways people-- there's lots of different notations people use to talk about stochastic dynamics. The one I'll use is, I think, the most popular one. We still got our standard equations of motion, but we're going to add an additional input, which is some disturbance w as an additional input into our dynamics, where w of t is some noise process.

OK, but if we let the noise come in through an input here, then we can still think about it as a deterministic function, our dynamics as a deterministic function, and keep it mostly the same as what we've been thinking about. Now some people don't like defining noise processes in continuous time. It's a little bit more natural to describe them in discrete time, and as we've done in the other class, let's do the discrete time case first.

So in discrete time, we'll do the same thing. But now maybe we can-- it's easier to think about what w of n might be. So for instance, w of n might be some iid Gaussian process or something like that, which is just a complicated way to say that at each n, w n is sampled independently from a Gaussian. Let's say a normal distribution, like something like that.

So when I'm simulating this in MATLAB, if I have-- I can simulate an iid Gaussian process by just every time step calling rand n, yeah, as if-- and it's independent from the [? w's ?] I picked before. That's a pretty good model.

Now one thing I want to avoid talking about today so far is, let's still assume that we have perfect state information. So I don't want to worry about sensor noise yet. It turns out it'll be pretty natural to think about it with some of the same tools, but let's just assume for now that when I'm in state x, I know I'm in state x exactly. But if I'm trying to think about, into the future, what's going to happen, what's the optimal thing to do, I have to worry about the fact that the noise in the system is going to push me around.

OK, so we updated our dynamic equation. We better start-- if we're thinking about this in an optimal control sense, we better also update our definition for optimality. So it used to be that we said J in the discrete time case was just some sum from n equals 0 to n of g xn un. The problem with that now is that xn, this is going to be-- this is now a random process, yeah?

If I run from the same initial conditions with even the same open loop tape, let's say, the system five times, this is going to be a different value every time I run it. Exit time three will be different every time I run it, OK, which means J is also going to be a random variable, a random. So it doesn't quite make sense to say my notion of optimality is some random variable. We want to choose a property of that random variable that we care about.

There's different schools-- again, being philosophical a little bit, there's different schools of thought in control theory. Some people say the thing you should care about, the only thing you should worry about is the worst case behavior. You want to worry about the tails of your distributions and make absolutely sure that my plane-- if I'm riding on a plane from here to California, it's not going to fall out of the sky with five nines of probability or something, OK.

I actually don't subscribe-- OK, when I'm riding on a plane, I do subscribe to that, OK. But when I'm building robots that I don't have to put my life in jeopardy for, I don't believe in that. And I actually don't think animals do that. I think if animals were so conservative-- so that's the robust control approach, what I just described, worrying about the worst case.

And the problem with robust control, the well documented, well discussed problem with robust control is it tends to come up with conservative control strategies. So if I'm worrying about never running into the table, then I'm never going to go anywhere near that table. There's another approach, which I prefer, that tends to be more common in the optimal control community, is to worry about maximizing the expected returns, OK.

So now let's define J as the expected value of this cost function. There's a really obvious reason why the optimal control people choose to do that. It's because we already made a decision to do additive costs. Expectations play beautifully with summations, and so our life is going to stay clean and good if we're willing to do the expected value derivations, OK.

But also philosophically, I think that me as an animal, I maximize my expected reward. Or a gazelle running through the field, I think, is maximizing expected reward. I mean, on average, it's doing spectacularly well. And every once in a while it wipes out and falls down and breaks its leg and gets eaten. But if it was worrying about that all the time, then it would never run as fast as it does. So-- or as aggressively as it does.

So I think, personally, if you want to build aggressive robots, you've got to stop worrying about guarantees of stability, of performance. Just try to maximize the average performance. I think that's where I've put my chips.

OK, so now, just to be clear here, so x-- again, every time I run it, even with an open loop u, if I completely control u, so u is not a random variable but some open loop tape, lets say, x will still be a random variable, OK. But J is not. J is saying that, given some control policy, some initial conditions, there is a well defined cost that I-- expected cost that I receive. And that's something I can optimize, OK. Does that make sense?

OK, so I want to think about the implications of having these things turn into random variables with a simple example, OK. And that example I like, the one I like is a particle sitting in a bowl. Let's say some particle in a potential well. So let's say gravity is like this, and I've got some bowl I'm sitting in, and this particle is going to want to roll down that bowl, OK.

But to make it interesting, we're going to say that this particle is subject to Brownian motion. Do you guys know what Brownian motion is? I guess-- was it-- I guess if you look down at a Petri dish of very small things and they don't sit still, there's debate about exactly the physical mechanisms of it, but phenomenologically, you can see very small cells that are not actively motile by themselves move around in a random fashion, doing random walks and things like this. So people-- I won't get into the philosophy of-- the philosophical debate of whether there exists stochasticity in the world, but I think stochasticity is certainly a relevant model for a lot of things we're doing here. And I don't want to get quantum in class.

OK, but let's just say that this guy is subject to the dynamics of this bowl. But on top of that dynamics of this bowl, it has a tendency to jitter around a little bit, OK. So I'll write down the dynamics. I'll keep it discrete for now.

So let's say at every time step, the difference-- the update looks like the gradient of that bowl it's trying to go down that bowl plus some random noise. I'll call it z of n. And this, again, I'll assume is iid, Independent Identically Distributed Gaussian noise. So if you've never taken a class with all this random variables and everything, I hope most of this will still come through.

I'll throw a few of these words and try to be soft about it. If you have questions about any of these, just ask me. I think that we're going to be able to say things that are fairly mechanically intuitive and helpful. So I hope it's accessible to everybody. And if it's not, ask me.

OK, so this is a reasonable dynamical system now. It's attempting to, on each update, having gone down the gradient plus some random noise, OK. Let's make our lives easier by choosing u of x to be-- how about that? That's pretty nice, right? It makes our life good if everything is quadratic. Then this thing turns out to be negative alpha x.

OK, so just to make sure we're thinking about it, if the noise is 0, then what's going to happen in this system? I've got a discrete time system, which, if I move that back over to the other side like we're accustomed to, it goes like this. So how does that thing behave?

Where's the fixed point? At 0. And when is it stable? When's it stable? Discrete time linear system.

AUDIENCE: [INAUDIBLE] equals 0 [? of 1 ?] but [INAUDIBLE].

RUSS TEDRAKE: Yeah.

AUDIENCE: [INAUDIBLE] [? positive ?] [? x? ?]

RUSS TEDRAKE: No, you're thinking too continuous time. In discrete time, your bounds on your eigenvalues are between-- the absolute value has to be less than 1, which I think, in this case, means alpha has got to be between 0 and 2, yeah? Everybody OK with that? Yeah. That wasn't supposed to be the big insight for the lecture, but.

OK, so the reason I wanted to say that-- OK, so we definitely have some stable dynamics pushing us this way in this bowl. The picture tells you that. The math tells you that. We have some stable dynamics that's pushing us towards here, and then we have something that's pushing us out, which is that noise.

If I was-- maybe just as a thought exercise, if my bowl had been flat, if alpha was 0, and I've got this thing subject to Brownian motion, then if I look at it at time 100, where's it going to be? I mean, it could go sort of anywhere, yeah? If it's in this bowl and it's subject to Brownian motion, then where's it going to be at time 100?

AUDIENCE: [INAUDIBLE] close to [INAUDIBLE].

RUSS TEDRAKE: You'd expect it, with high probability, to be around here. There's a chance-- I mean, even if it turns-- even with small noise, if I were to get some abnormally rare large force in this direction 10 times in a row, if I watched this Gaussian process long enough, I might look and find it here. But that's going to be very low probability.

So what I'd expect to find is if I watch it for some amount of time, and I look at it, and time is 100, that it's probably going to be here. Going to draw some probability distribution here. And sure, there's some tails here that'll say if I looked, maybe I'll find it there, but that's pretty unlikely, OK. So hopefully when we're all done with this, we're going to get that out. And it's not too hard to see it, actually.

So what's the best way to say it? So let's pretend that we're going back to our-- so this was the-- we're back to the noise case again, putting epsilon back in. Can we compute, then-- so if I know where I am at time n-- if my sensors are perfect, like I said, I know where I am at time n, where am I going to be a time n plus 1? So let me write that as some probability distribution that-- I want to write, where am I at time n plus 1 given I know where I am at time n? What's that going to look like?

AUDIENCE: [INAUDIBLE] Gaussian function [INAUDIBLE] 1 minus [INAUDIBLE] n by some variance, which is kind of [? problematic, ?] actually.

RUSS TEDRAKE: Awesome. Right? So probably, I'm going to be 1 minus alpha xn away from where I was, but then some Gaussian distribution centered around that point. The deterministic part puts me here, and this part then adds noise to that, OK. So that's going to be my full Gaussian here, 2 pi sigma squared e to the negative xn plus 1 minus 1 minus alpha xn, that whole thing squared, all over to sigma squared, yeah?

You agree with that? If I know where I am at time xn, which, by the way, is equivalent in probability space to saying I have a delta function here-- I know where I am. My probability distribution is a delta function.

Then on the next step, I'm going to be 1 minus alpha times that, and I'm going to have a distribution, which is mean of this. This is a Gaussian distribution x minus mu over 2 sigma squared, squared. This is the new x. This is mu. Yeah?

Good. So now we have everything we need, really, to proceed. So let's say I knew where I was at x0. On x1, I'm going to be at the function described by this. Where am I going to be at x2? Well, in general, I have to do the update. And let me use the notation P of n plus 1, meaning the probability distribution over x at time n plus 1.

So think of it as a different function at each discrete time. Notationally, it's the cleanest, I think. Well, that's going to be-- it's going to be-- I have to think about-- oops, I did-- sorry, this is a y. Yeah. My fault.

So if I was-- I have to think about all the possibilities. I want the probability that I was in y equals 0 and then the probability of being in x, considering I was in y0. But also, I have to think about, what if y was 1? Well, what's the probability of that? And I'm going to sum the whole thing up in a continuous way, and I'm going to get my new probability of being at the new place. You guys OK with this equation?

All right, I have to consider all possible cases of where I was at time n-- that's given by this-- and then apply my dynamics, which was given by this, to get my new distribution, OK. Hope that's OK, but even if it's not, it still should be-- you'll still be OK here in a second.

So we can do that. So let's say that P of n y is a delta function. That's what I said. So I know my initial conditions.

I should have drawn it right here to match that plot. Well, then after one step, it's going to just pick out the P of this Gaussian centered at that y. And at one step, then, I'm going to be at a Gaussian distribution 1 minus alpha from where I was, centered around 1 minus alpha from where I was.

Now in the next step, I have to consider the fact that I could be anywhere in that Gaussian, weighted appropriately. And I have to consider all of the updates from those. That's what this integral is doing. And it turns out the magic of Gaussians and linearity is that if P of y is a Gaussian, and I multiply it by another Gaussian and integrate, I get out a Gaussian. Yeah. Life is good.

So I'll leave the actual math to your-- eh, what the heck. I'll just-- what you get-- I can write the answer-- if you push a Gaussian through, turns out to be 1 over square root of 2 pi sigma squared integral from negative infinity to infinity of e to the negative-- let me actually write the-- skip that one line just to keep things moving.

Let's do 1 over 1 minus alpha square root of 2 pi sigma squared. I have the probability of n at y prime minus 1 alpha dy alpha, where y is 1 minus alpha y. It's just-- I haven't done a lot of work yet. I just changed coordinates into y prime.

And it turns out, for instance, if I look for a-- if I guess that the steady state is a Gaussian form, and I look for a place where this and this can possibly be the same function, if I want to look at the steady state of this dynamics, then I find that P star of x is 1 over-- oh-- square root of 2 pi sigma 0 squared e to the negative x squared 2 sigma 0 squared. It's a Gaussian. It's actually a mean 0 Gaussian, is the steady state of that update, OK, which is just a few lines in the notes, where sigma 0 squared is sigma squared, which is the noise from the Brownian motion minus alpha squared.

OK? So I think, actually, these equations tell the entire story. If I start my system even from some-- well, specifically, if I start my system in a delta function or in a Gaussian distribution of initial conditions, then I'm going to be Gaussian for the rest of time. Turns out even if it's not, it'll go to a stable Gaussian.

And if I watch-- if I look far enough into the future, at the steady state of that probability distribution, then it's actually going to be what we hoped we'd find. It's a mean 0 Gaussian. This was 0 in my plot. And its width, its variance is given by two competing terms.

You've got the noise from the Brownian motion trying to push you out into larger variance, and you've got the competing force of the stability of the dynamics pushing you back in, OK. So if alpha gets-- and you actually-- this also is valid exactly when alpha's between 0 and 2. Doesn't go to 0 in that regime yet. OK.

So if I want to look at that particle at time 100 or time 10,000, then I should expect to see it somewhere with probability given by this distribution in the vicinity of that 0, OK. And that comes out of simple math of pushing this Gaussian through this equation, OK. Now you all probably-- most of you probably knew that before. Why do you know this before? Maybe not in this level of detail, but why do many of you know this already?

It's a Kalman filter, right? The Kalman filtered forward process takes a Gaussian, shoves it through a linear system, stays Gaussian. This is the single variable, little more careful version maybe than you'd do if you call MATLAB's Kalman stuff. But yeah, so that's not a surprising result.

But I think it forces you to think about a couple things. So like I said, the stability of the system is critical in determining that final distribution. But even more significantly than that, there's some implications of having noise in the system.

Implications of having stochastic dynamics, OK. If you want to reason about the cost that you're going to incur, given some policy, for instance, in order to reason about the future dynamics, even though you know current state, so even given initial conditions, you have to reason about-- it's not enough to just reason about x. You have to reason about that entire distribution, what I called Pn of x, not just x.

OK, so as soon as we start doing stochastic stuff, we have to change our view of the world. The state x is not the only thing you care about moving forward. You care about the probability distribution of states that you live in.

What would you say about the stability of this system? If I asked you, is that a stable system, what would you say?

AUDIENCE: [INAUDIBLE] [? stable it ?] [? is ?] [? or something? ?]

RUSS TEDRAKE: I'm asking-- and we're going to do that, but I'm asking you for your intuition. What would you guess? Would you feel comfortable if I said, it's a stable system, let's move on?

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: In some ways, it's OK, because the distribution is stable. x is not stable, OK. If I look at a-- if I'm at-- there's no fixed point in x. The noise is going to keep moving me around, OK. But I told you that P of x, actually-- well, I only told you it's a fixed point, but I can tell you it's a stable fixed point. P of x is actually stable, OK.

OK, so basically, this equation right here, this update right here, it's a very famous, important thing. Well, so important that it's called the master equation, yeah. I mean, don't just name it after some guy. Call it the master equation. And there are various versions of the master equation and specific problems that are named after people's last names, but in general, it's the master equation, OK. So you can't forget how significant it is.

And the idea is that in the master equation, you're looking at the dynamics of the probability distribution. Not the dynamics of a single state, the dynamics of a probability distribution, OK. So that probability distribution, actually, in the master equation, is stable, OK. Now there are more complicated cases, actually.

So what about this one? What's this thing going to do in the long run?

AUDIENCE: Bimodal distribution.

RUSS TEDRAKE: It's going to have a bimodal distribution in the long run. So it would be inappropriate to say either of those points-- I mean, that either of those points are fixed points. For the deterministic case, they are. For the stochastic case, they're not. And you're right. It's going to go to some distribution like this, OK.

I have a decision point, which of my 10 pages I should do. I could talk about some examples of just stochastic dynamics, for instance, on walking robots, or I could get to the optimal control of the more simple systems. I have to set John up for next week, so I guess I got to just tell you that, actually, we've done some work thinking about, for instance, the rimless wheel-- I'll just tell you the setup, and I won't tell you all the details.

So here's a realistic example of that sort of dynamics. Take your rimless wheel, OK, dynamics. And let's say, instead of walking down some constant ramp, let's say on every step, the ramp angle is drawn from some distribution. OK, so this is passive walking on rough terrain, mm-hmm.

And it turns out it's not following a limit cycle anymore. But it's always-- its long-term probability distribution is-- well, there's a slightly more complicated story. If I look long enough, this thing has an absorbing state. So if I take a big enough step, then eventually, I'm going to lose enough energy. Remember, the deterministic system had two fixed points. One was standing still. The other one was rolling at a constant speed, OK.

The standing still fixed point on rough terrain is an absorbing fixed point. If I get there and I never take another step, then I'm never going to leave that fixed point. So that is actually a true fixed point, yeah.

The rolling fixed point, you're going to tend to bounce around that fixed point. So maybe this picture is something more like this, yeah. OK? So the standing still fixed point, if I get in there, it's absorbing. I'm never coming out.

But the rolling fixed point, you tend to bounce around this limit cycle, OK. And then every once in a while, in the stochastic dynamics case, they say that these particles make an escape attempt-- that's what they call it. OK-- and maybe shoot over and fall down, OK.

And it's really very beautiful. If you watch the probability distributions as they propagate through the rimless wheel equations, or the compass gait equations or you name it, then what you get is you get this probability mass around here. For a long time, it's pretty likely that I'm in the vicinity of that limit cycle. And then slowly, as the time goes on, the escape attempts continue to the point where this thing gets smaller and smaller until, as time goes to infinity, I'm only going to be standing still, OK.

So the negative-- the pessimistic view of that work is to say that you're always going to fall down. You can build the best robot you want, but if you have a reasonable model of the dynamics, it's always going to fall down. If you wait long enough, a Mack truck is going to come along and hit it or it's going to walk into a door or something like that. You can do the best you want, but it's going to fall down, and it'll end up on YouTube, probably, right?

[LAUGHTER]

So--

AUDIENCE: So you're assuming your ramp distribution is actually Gaussian?

RUSS TEDRAKE: That's what we decided, yeah.

AUDIENCE: OK.

RUSS TEDRAKE: But that doesn't actually imply that the posterior is Gaussian, because it's going through nonlinear dynamics.

AUDIENCE: Right, but I mean, in reality, the random distribution is never going to be truly Gaussian, right? Because--

RUSS TEDRAKE: Everything's Gaussian if you get enough of-- I don't know. I mean, I think that's-- so you want to do stairs or something more specific?

AUDIENCE: Oh, well, I'm just saying, if there is a hard limit on how steep your ramp is--

RUSS TEDRAKE: Oh.

AUDIENCE: --you could guarantee that--

RUSS TEDRAKE: Good. Very good point. All right, so of the distributions didn't have tails, then there are cases where I can bound it never going over. But those tails actually have to be pretty steep-- the limitations have to be pretty steep, because you have to make sure that, on a single step, of the damping overcomes the biggest possible perturbation. If on a single-- if your noise can ever be bigger than what you can take out in a single step, then you will eventually, as time goes to infinity, find a way to get out. Yeah?

So Katie Mill did some nice work in quantifying the metastable dynamics of walking. And actually, I think that-- so we call it the metastable, because that distribution is long-living. It still makes sense to talk about where you'd expect this thing to be while it's walking. But eventually, we have to admit it's going to be-- it's going to go to falling down. Like a diamond is a diamond for a very long time, but eventually, it'll turn back into graphite.

OK, so good. So there's actually a beautiful-- even if you don't care about control, I think there's actually beautiful things that happen in stochastic dynamics. But the thing that matters here is, we've switched hats. We've now started thinking about probability distributions and how they evolve with dynamics, and how we can change those probability distributions with control. If I could if I could control the shape of that, then I can control those probability distributions, for instance.

OK. So it turns out it's sort of trivial to work stochastic-- to solve stochastic optimal control problems, at least with dynamic programming, OK. And it works out, because of this additive cost structure, that it's roughly no more expensive to solve the stochastic optimal control than the deterministic one. And that matters. Maybe I should even make the point that it matters.

So if I have a stochastic process-- and in general, the optimal policy that you get from stochastic optimal control is going to be different than the one you get from deterministic optimal control, so potentially, in dramatic ways. Let me try to make that point here. So imagine I've got my-- I've got a trashcan robot. I shouldn't call it a trashcan. I've got a-- what are they? What are the names of those little red robots? Pioneer robot or something like this, yeah?

And I want to get it from-- to this goal. Let's say I've got a cost function like this, and I start over here. And as I go, I know that my wheels slip or something like that. My distributions are going to grow as I go, yeah. And I've got some ability to control them, so they're not going to grow unbounded, but let's say they're going to grow in my path to the goal, OK.

There'll be two competing forces. There'll be my ability to measure and fight against disturbances, and there'll be the inevitable disturbances. And those two will again combine into some sort of distribution over time, OK.

Now imagine-- like the scenario we talked about in the feedback case, imagine my cost function is 0 everywhere, negative 1 at the goal-- I want to get to the goal-- and say something really big here, yeah. There's pits of fire in the middle of the lab. OK? No, I mean, right, so we've got to make the point. If it was just 1, I wouldn't make-- be as dramatic.

But OK, so long story short, a stochastic optimal control solution is unlikely to choose this path, because 0-- even if the distributions are fairly tight, 0 times a big part of the probability distribution plus 1 e to the 6 times even a little part of the probability distribution is still a big number, OK. And so therefore, the expected value of going through here is that I'm going to incur quite a bit of cost. Does that make sense?

So if I just did deterministic optimal control, we talked about using feedback to try to motivate not going through there. But really, the more direct way is to think about the probability distributions. So if I can control my probabilities to the point where I know 0 probability is going to be in here, then sure, go ahead through there. And the deterministic one will probably find that. But the stochastic one, if it realizes there's something, will probably try to find a different path, OK.

So that's one example. But in general, the stochastic optimal control policies are going to be different than the deterministic ones, and better. If you have a reasonable model of the disturbances you'd expect to encounter, then you should allow your optimal control tools to think about them, OK.

AUDIENCE: [INAUDIBLE] stochastic environment s times 4 [INAUDIBLE] [? you have ?] [INAUDIBLE] [? more states, ?] because potentially, each action can move you to any of the possible states while in deterministic case, you can go to one state. So when you do [INAUDIBLE] [? worse ?] [? case. ?]

RUSS TEDRAKE: Right. I knew someone was going to-- so I would say essentially no. I almost wrote, essentially no. And the reason I want to make that comparison, actually, is because I want to compare it directly to the barycentric interpolation that we were doing before, which is what I'm going to do in a second. And that is already doing an interpolation, already going through some probability, some transition matrix, yeah.

And it's probably true that the-- it may be true, depending on your noise distributions, that if you add a lot of noise, that transition matrix will be more dense, and therefore, it might take more time, depending on how you computed. My MATLAB implementation, it's the same. Yeah? Is that fair?

And it's no more complicated to write down. How about that? OK? We'll completely understand what [INAUDIBLE] was asking in just a second. OK, so why is it no more complicated for me to write down on the board the stochastic optimal control case in dynamic programming?

Remember, I said now this is-- I'm going to take the expected value of my additive cost. First let's just think about what the implications are for doing optimal control. So first of all, I can take that expectation inside. And now what's the probability of g-- or what's the expected value of g at xn un? Well, x has got some distribution given by P of x, and yeah.

So this thing is going to work out to be-- you can always take the expected value of a function of x by just that function times its distribution integrated. This thing's going to work out to be so an integral over all possible states of g of x u of n times P of n x dx. Right? OK, so you could imagine computing optimal policies by figuring out the state distribution by that evolution I was talking about before and then integrating over the possible states, yeah? The costs for each state, and figuring out our J, figuring out a way to minimize that.

I only wrote that down to make it look hard, OK. It turns out, again, just like before, the recursive form is beautiful and simple, OK. So you can imagine doing it that way, and that's correct, but just like before, the dynamic programming solution exploits the recursive-- the additive form and does a recursive solution which just works out beautifully, OK.

So if I do J of x from time 0 being the expected value of, let's do the final cost also, then what's J of x capital N? My cost to go, given I'm at the goal. The time has expired, and I'm at state x.

AUDIENCE: [INAUDIBLE] h of x.

RUSS TEDRAKE: Is it expected value of h of x, or is it just h? What is it?

AUDIENCE: h is [? deterministic ?] [INAUDIBLE].

RUSS TEDRAKE: Awesome. Yeah. If I know I'm already in x, then there's no probabilities left, yeah. OK, and then if I go backwards, J of x at time k is going to work out to be min over u-- I should say J star of x. Sorry. J star of x is going to be min over u the expected value of g x, u plus J star of f of x u w of n k plus 1. Can you buy that?

OK, so the reinforcement learning people always like to say that the reward or the cost can also be a random process a random variable. I'm always in this case where I design the cost, it's a function of some random x, but g is deterministic. So actually, I could take that expectation right inside here, and I just have to do a min over u of g of x times the expected value of my cost to go, OK.

The nicest way to see how to implement that is let's go ahead and-- we've already discretized time. Let's discretize state and actions. So now I have S of n plus 1-- remember I switch to S's and a's when I discretize things-- is now f of S n times some action times my noise. And the advantage of discretizing stat and actions is now I can do P of n plus 1, which is a function of S. I could think of that as just a vector where the ith element is the probability that S of n plus 1 equals Si. Is that OK?

The probability distribution, remember, in general, was a function. In the particle in a bowl case, it was a continuous Gaussian function. If I discretize the state there, then I can represent that as a vector, saying, what's the probability of being in state one, what's the probability of being in state two, probability of being in state three, and so on, OK. So the reason to discretize states is I can turn my continuous function into a vector. Yeah.

And I can turn this function into a transition probability matrix. I can-- so f goes to Tij, which is the probability of landing in Sj-- I should-- it depends on the actions-- given I was in Si and I took action a. This is a matrix. It's the transition matrix.

And now the state distribution dynamics are going to just turn out to be a pretty simple matrix equation. That's in-- oops. Tij Pj at time n plus 1. Yeah, so let me actually write the whole vec-- the real vector form. That's really for-- that's for a single element of it.

I could just write, if I'm doing it in column vectors, then it's actually going to be P of n times T of a, OK. Where these two are vectors, that's a matrix. So and this is the discreet time, discreet action, discreet state master equation.

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: I use it as a column vector.

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: Let's make sure. So the probability of being in state J given I was in state i should be the sum over-- write this thing. Probability of being of S n plus 1 is a probability-- what's that? I think I had it right, right? It's not a true loop, but I think that's right.

AUDIENCE: [? Don't ?] [? you ?] [? need a ?] transpose?

AUDIENCE: Yeah.

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: I need a T transpose?

AUDIENCE: Get a T transpose.

AUDIENCE: You just can't hit the--

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: Oh, of course, because I did this. OK, yeah, so sorry. Yeah, I think the way I've got T defined-- but thank you. That should be a transpose, yeah. Good.

AUDIENCE: [INAUDIBLE] T transpose on the other side [INAUDIBLE].

RUSS TEDRAKE: In which case, I could have written it with the T as a transpose too, but if I transpose the whole thing. Doesn't really matter if you like row vectors or column vectors. The point is the master equation, which looked a little scary before, yeah, turns into a simple matrix update in the discrete time, discrete state case. Yeah?

AUDIENCE: So in this [INAUDIBLE] [? truncated ?] [? the actions ?] or you'd just not [? capture things? ?] Because you have hard limits if you discretize things.

RUSS TEDRAKE: Good. So there's a question again, just like we had the question when we did dynamic programming for the value iteration, of how you go from the continuous probabilities and continuous states back to the other one. So again, yeah. So I would sample for my Gaussian and fill out a transition probabilities as a close, truncated representation of the Gaussian and still interpolate with the barycentric interpolators.

And now the DP update works out to be just a simple. The probability of being in S-- let's see S-- J is the min over a. The expected value from this equation can be taken care of with just the transition matrix over here. And I get maybe my vector g of a, which is the cost of being in each state given I took that action, sum over j Tij Sj J of Sj k plus 1. Yeah?

And I get rid of my expected values by just using that-- working directly with the transition matrices. i on this side, j on the side. Many apologies.

This turns out to be exactly-- the reason I said, basically no more expensive to solve than the deterministic case is we already used this form when we were doing the barycentric interpolation. Because our problem in the dynamic programming originally was that when we started simulating this thing from one node forward, it didn't end up-- unless you were very, very lucky, it didn't end up right on top of another node. So we already had said that we're going to estimate the new cost to go as an interpolation of the neighboring points, of the value of the neighboring points, where that weighting came from the barycentric interpolators, OK.

We're doing the exact same thing now. In fact, you could actually think of the barycentric interpolators as turning your deterministic problem into a stochastic problem, where the probability of going into each of these neighbors is the interpolant, OK. So the reason the barycentric works beautifully is that it turns the deterministic case into a stochastic case. Yeah? And that's why I wanted to say that it's no more complex, OK.

T might be more-- have less 0s than in the general case. If maybe-- with some probability distribution, it might be that I have to worry about hitting a lot more nodes. So it might take few more cycles. But the equations are the same, and my MATLAB code is the same, OK.

Simultaneously, or maybe conversely, this helps-- actually tells you why we have problems with the barycentric interpolators. This is the-- remember, the fundamental problems with the barycentric interpolators is that things leaked away from the nominal trajectories, and we had our chattering happening in the-- and our bang bang solution wasn't quite right. Because now you can think of it as, is my deterministic problem assigning some probability of going to each of these neighbors? And you can see that that distribution's going to start slipping away from the nominal trajectory. OK, excellent.

So this is actually very important. Stochastic optimal control is a beautiful thing. If I can model the disturbances in any reasonable way, then I can get better policies by explicitly reasoning about them. And just like we said dynamic programming for low dimensional problems solves all these really hard problems that are analytically intractable and things like that, it can even solve a stochastic problem with almost no more work, OK. The low dimensional problems, even complicated ones with complicated distributions, DP can do the work for you.

OK. So a few more things to say. There's one particular result, which we already mentioned early, that I have to mention here. This dynamic programming update, we use this in our analytical optimal control, too. We use this as the basis to start designing things like our LQR controllers that we turned it into the HPJ. And in the finite time case, we didn't even turn it into the HPJ. We just started-- we started-- we can back this out with dynamic programming, OK.

So we can actually use the same thing to analytically try to design some controllers for stochastic optimal control cases. And just like in the deterministic case, there's one outstanding result that everybody knows and uses, and that's the linear quadratic regulator with Gaussian noise. LQG is the shorthand.

There's two forms of it. One of them is Gaussian noise also on the sensors, but let's just worry about the case where we know there's no uncertainty in the sensors, only the dynamic noise. So x n plus 1 is a. It could be a of n. It could be time varying or not. xn plus B n u of n plus wn. Cost function, again, is the quadratic regulator.

What do you think's going to happen with that problem? Someone who hasn't used it extensively in your work, what's going to change about our LQR solution? Think about stabilizing the pendulum or something, OK. Let's say we're doing optimal control on the simple pendulum linearized around the top, and now there's disturbances bouncing me around.

How would you act differently given some model of disturbances in the linear case? How would you act differently if you know that somebody's going to be bumping me around with a mean 0? Let's keep w mean 0. How would you act differently if you're a simple pendulum around the top?

AUDIENCE: [INAUDIBLE]

RUSS TEDRAKE: Anybody else want to weigh in?

AUDIENCE: Increase your gain.

RUSS TEDRAKE: What's that?

AUDIENCE: Increase your gain.

RUSS TEDRAKE: You might increase your gain or something like that, you'd think. Yeah, it turns out you wouldn't act different. It's one of the most surprising results, I think, of stochastic optimal control.

It turns out that you work it all out, you put your costs like this, you wouldn't-- you don't turn your gains up because of the disturbances or anything like that. The optimal solution for the stochastic case is the same policy as the optimal solution for the deterministic case, OK. It's also true in continuous time.

R inverse B transpose S of n x. Yeah, it's the same B of n here. Did I write something funny?

AUDIENCE: Did you know that that clock isn't moving?

RUSS TEDRAKE: No, I didn't. Am I way off time?

AUDIENCE: It's about 4 o'clock.

RUSS TEDRAKE: OK. Thank you for telling me that. I thought I had time. Nice.

AUDIENCE: Is the S the same?

RUSS TEDRAKE: This S is the same too. Oh, sorry, sorry. Good. The S I wrote here is the same, OK. That is the same S that you get from the Riccati equation, but the total cost is bigger in the-- so the policy is the same, but-- so how much time do I really have? Do I have negative time already, or am I--

AUDIENCE: Yeah, kind of.

RUSS TEDRAKE: Sorry. OK. But J of n-- I thought it was just very exciting-- is this plus some expected value of the noise. I was going to keep going for a long time, probably. OK.

So it's the same S that you had before, but the cost to go that you get is actually higher, in a way that you might expect with-- it actually depends on the cost on the other S. And so the S of n comes from the deterministic Riccati equation, but the cost to go gets bigger. Yeah?

So that's one of the most surprising, I think, results from stochastic optimal control, is that in one case, it tells you it's OK to do deterministic optimal control, yeah. In most cases, it's not OK. It won't give you the same thing.