IntroToLinearDynamicalSystems-Lecture20

Instructor (Stephen Boyd):I guess we're on. This is the last lecture – you can turn off all amplification in here. I don't need any amplification here. Thanks. We'll finish up a couple of topics in reachability and I'll just do a couple of topics in observability. It's not a hugely important topic. And then I wanna reserve some time at the end of this class to just say a few things at the very highest levels about what the class is, how all this stuff fits together, and all this sort of stuff, so that's what we'll do. So continuous time reachability we looked at last time, which in fact was I guess for the people here in real time was this morning, and we looked at what's the reachability subspace for a continuous time system, so that's the set of all points you can hit in T seconds starting from zero in a continuous time system. The answer turned out to be real simple in this case. It's not interesting or subtle like in a continuous time system where at each step for a while anyway interesting things can happen. You can sort of hit a few points and some more, and all this kind of stuff. It just all happens at once. What happens is if T is positive, then the set of points you can hit is the range of controllability matrix period. It doesn't change with T if T is positive. What will change of course is the size of the input required to hit a target state if you do this real quickly, as opposed to if you do it on a long leisurely timeframe. So we also looked at this time. If you wanna look at the least norm input for reachability, you're looking for the input that minimizes this integral of the square of the norm of U. And we worked through that by discretizing it, and the formula that came out was this, quite straightforward. It was this integral of a positive semi-definite matrix. The integral itself is positive definite. There's this inverse, and then that's multiplied by this thing over here, which is B transpose time E to the T minus tau. Now this is actually – that's the system running backwards in time. I think I commented on it this morning. And all the parts of this correspond perfectly to the discrete time case. Instead of having an exponential, you have a power. These are just time propagators for example. The other thing I should mention is that this whole thing is really something just like C transpose C C transpose inverse. The difference is C is now like this kind of big complex operator. It's not a matrix as it was in the discrete time case. But this is the analog of CC transpose here – or sorry, CC transpose, and that's C transpose here is what this is supposed to – this is the analog. What happens then is that you can work out a formula for the minimum energy to hit the state in the amount of time T. It's a quadratic form, and it's a quadratic form that depends on Q of T inverse where Q is this integral, so it's an integral of E to the tau A BB transpose E to the tau A transpose, very familiar here because this is nothing more – this is sort of a – that's the time propagator for what an action at now – what effect it has on the state tau seconds in the future. You integrate this action – that's sort of like the same as summing in the discrete time case – and you invert. And this in fact – I guess you had one number problem on this – this is a Gram matrix in fact. This is an integral of – that's a Gram matrix, and it's an integral of something times something transpose. That's a Gram matrix. Now we can conclude all sorts of interesting things here. For example, if T is bigger, you integrate, you add more positive semi-definite terms or whatever you wanna call an integrand to this thing, and Q gets bigger, and that means Q inverse gets smaller. So here again, you have the idea that as the time horizon gets longer, the minimum energy required to hit a target in longer time goes down. The difference now is that T can be arbitrarily short, and you may need a very large input. By the way, as T gets shorter and shorter, this generates inputs, which in fact converge into something like the impulsive inputs that we looked at earlier. So there are impulsive inputs that get you to transfer U to a state immediately. Of course, an impulsive input is extremely large. Here, this will actually construct the impulsive inputs. Now we could also look at what happens when T goes to infinity. This matrix always exists because this thing is monotone increasing in the matrix sense. That's non-decreasing, and therefore the inverse – this is a positive semi-definite matrix – sorry, a positive definite matrix, and it's decreasing as a function of T, so this goes down. It's the quadratic form that tells you how much energy it requires to hit a point, any point in state space. This goes down. It has a limit therefore, and we'll call that limit P. It's the same as before. And that gives you the minimum energy to reach a point arbitrarily leisurely. So that's what that means. It's the same story. If the matrix A is stable, then this integral converges here. Nothing goes to infinity here if A is stable. Why? Well, because if A is stable, E to the tau A, all the terms in it look like polynomials times decaying exponentials. So this term simply converges, and therefore P converges to the inverse of that, and is therefore positive definite. And that says basically if the system is stable, then the amount of energy required to hit a point arbitrarily leisurely is gonna always cost energy. So there's no point you can get to for free for example. There's no point you can get to with arbitrarily low energy. It makes sense. If it's stable, that sort of means that left to its own dynamics, the system state will go back to zero. So it means that if you wanna get to some state, you have to fight against that, and that's what tells you that P is gonna be positive definite. So that's basically the idea here. We can look at the idea of general state transfer. That's transferring a state from an initial condition TI to some final state T desired. It's all very similar. It's the same as the discrete time case. What you do is you say this. You say that the final state is equal to this. It's the initial state then – that's the time propagator. That propagates you forward in time TF minus TI seconds here. That's this part. So this is where you would land if you were zero. If you did nothing, this is where you would land. This is the effect of the input over that interval like this. So the whole thing, you realize the following. This is equal to that. This is equal to X desired if and only if you put this on the other side, if U steers X of zero to X of TI minus TF to this thing here which is X desired minus this time which sort of compensates for the drift term. And in fact, all of this kinda makes sense. This says that if you wanted to steer something to a certain place, what you'd do is this. You first find out what would happen if you did nothing, and that would be over here. I subtract that from this, and then I act as if I'm going from zero to hit that point, and everything will work out. That's by linearity, so this all works out. Now for continuous time it's a lot easier because basically the system is either controllable or it's uncontrollable, and that means you don't get any of the subtleties. It means basically if it's controllable, you can get from anywhere to anywhere else in arbitrarily small time, possibly with a very large input, but nevertheless you can go from anywhere to anywhere else in any amount of time as long as it's positive. You can do it in zero time if you're allowing yourself to use impulsive inputs, so that's how that works. Okay. So let's just look at an example to see how this works. Here's three masses stuck between two walls, and we have some springs, and we have some dampers, and we can apply two tensions here. I think we've seen this example before. So okay, and your state is six-dimensional. It's the positions and the displacements of all – sorry, it's the displacements and the velocities of the three masses, and the system looks like this. The blocks here make sense. This basically says that – this top three are the positions, and it says the position dot is equal to – you read this as zero I, so it says that the top three DDT are equal to the bottom three. That's by definition because the bottom three states are the derivatives. And then these two come from the stiffness and from the damping. And then these two tell you how the two tensions apply forces to the different masses here. That's what this is. So we wanna steer the state let's say from E1 – so E1 means the following. E1 means that this mass displaced one unit to the right, and the other two are at zero, and everyone is at rest. That's what E1 is. And we wanna take that to the zeroth state. By the way, if we do nothing, it's gonna go to the zeroth state asymptotically anyway, and I can't remember what the eigenvalues are, but we looked at this example before and I think the eigenvalues are on the order of one. Maybe there's a slow one on the order of 0.2 or something like that. It doesn't matter. The point is that the whole thing will decay to zero at some rate. By the way, that builds the intuition here easily. It says that left to its own devices, this is stable. The state is going to go to zero anyway. Therefore, if I give you a long enough time period, long enough that just by through natural dynamics the state has decayed, it's gonna cost very little to drive it exactly to zero. By the way, with its own natural dynamics, the state is not gonna go to zero exactly. It will just get small. They get very small. But the point is once it's small, it takes a very little kick to actually move it directly to zero, and therefore we expect it to take very little energy. So what happens is the following. Here's a plot of the energy required to steer it from one, displacement one, then zero zero, and zero initial velocity to zero in T seconds. Very interesting plot, it says basically for six seconds and more, the amount of energy required is extremely small. By the way, is it zero? It can't possibly be. If it were zero, it would mean you'd apply no input. If you apply no input, it never gets exactly to zero. But the point is it's extremely small. So this is explained by the fact that the natural dynamics has made it small anyway. What's interesting here is that this thing now ramps up very aggressively. By the way, does this thing go to infinity or stop like in the discrete time case? No, this keeps going. So you could do this 0.01 seconds. There's an input. The energy required is absolutely enormous. You can work out what it is, so you can do it arbitrarily fast. But you can see exactly what's happening here is that you can move the whole thing to zero as quickly as you like. It just takes more and more energy, and this is quite obvious here. By the way, the practical use would be somewhere around here where you would be pulling this thing back to zero a lot faster than if you just waited for it settle out. Actually, I should mention something about this. There are lots of applications of this, tons. A lot of them have to do with things where you have a mechanical system where you move something from one place to another. I just thought of another one that's not mechanical. You move from one place to another. When it arrives there, it's sort of wiggling a little bit, but you can't use it until it stabilizes. The first example is a disk drive head positioned. So you say please seek Track 255 or something. So this thing zooms over here, lands close, but when it ends there, it's sort of shaky. It's gotta track the track very carefully. You can't use it when it's shaking. So in fact, what you do when you land there is you do something like this. You actually actively apply an input that will basically do active damping, that will remove all the wiggling in it and allow you to track faster. So things like this are actually used now. The other applications I know in X Y stages, so in lithography. These are extremely expensive machines for printing integrated circuits and things like that, and they move over a couple centimeters at a time, and they move, and then they align themselves to accuracies that are just unbelievable. It's like at the nanometer level. It has to be if you're looking at print technology, which is actually quite ridiculous to imagine something that moves two centimeters then – this state isn't even wiggling at the nanometer level, and stabilize itself absolutely to something at the nanometer level. So the way that's done – oh, and by the way the reason – it's of course stable, so you can move over there and just wait for the whole thing to stop wiggling, and then expose your wafer. The problem is this machine costs some huge sum of money, so you can have the cost divided by the throughput. So if 90 percent of the time, you're waiting for the thing to stop wiggling so that you can do your lithography, that's not a good investment, so in fact they use active control to do this where you move it over two centimeters, and you definitely apply inputs that will do active damping, and will cause it to stop shaking early, so that's one. And I'll mention one more just for fun from MRI. So in MRI you do these things. I don't know any of the details. I don't know any of the physics, but you line up all the magnetic axes, and then you let go, and they slowly drift down like this, and you don't do another scan slice until they've totally relaxed again, or something like this. That's the method. You can actually use a method where you act – instead of waiting – I guess they have times like T1 and – does anyone here do MRI? Too bad, someone could've got me out – they could've told me what T1 and T2 – there's like T1 and T2 are the two relaxation times or something like this. You wait one of these amounts of times or something like that to do it again. If you wanna do fast throughput MRI, you don't wait. You actively force the spins down to zero. How? Using these formulas here like this one. And you improve the throughput. And again, you have a big expensive machine whose throughput is now better because you're actually not waiting for something to stabilize out. You're forcing it to stabilize it out fast, and therefore it's ready for reuse again. Sorry. That was just a weird aside. If you wanna look at what some of these inputs look like, we can do that. Pretty much almost for the dot that I drew which was in three seconds, this is the input that you would apply. These are the two forces. It's symmetric, but then the whole problem is symmetric. These are the two tensions you would apply. I don't think they're obvious, so this is what it looks like for – this is the least norm input for bringing the system to zero in three seconds. If you relax that just to four seconds, it gets a lot smaller. Why? Because now the natural dynamics itself is helping you, so that's four. And if you look at the output for U equals zero – this is the – I guess now we have our answer. This says if you just kind of let the thing go with no input, you'd see that it's pretty small in eight seconds. Of course, it's not zero, but it's already kinda – so when you're taking it to zero in three seconds, you can see you're actually doing something. Depending on the accuracy with which this thing has to be zero before it can be reused, you can see that you're actually doing something, and you're doing something that is substantially better than just letting this thing settle out by itself. So I think these are all kind of obvious here. Let's see if I can – yeah, these just show things like the output for three and four, and things like that, but they're kind of obvious. So I think that finishes up this topic of controllability. I wanna emphasize that except for the fact that some of it was continuous time, there's absolutely nothing new in controllability. It's just applied linear algebra. If you know what range is, if you know how to solve linear equations, if you know how to find the least norm solution of a set of linear equations, then all of this is just an application of that. Okay. Now we'll look at the last topic in the class. Like controllability, it's not important. I think you've even done, or maybe – controllability stuff I think you were doing well before it had a name, I think. Didn't we have a problem on the midterm where you had to find an input that took you somewhere? There was one. Yeah, okay. So that's my point. All we did now is just give a name to it. That's all. So the same is true of the observability and state estimation. These are just exercises. So what we'll do is we'll look at this idea of state estimation and discrete time observability and stuff like that, but let's get the setup here. So we consider a discrete time system – this is in the general case that the setup would be something like this. You'd have X of T is A of X of T plus B U of T plus W of T, and W of T is called a state disturbance or noise. Another name for this is a process noise. That's another one you hear. And you have all of this in economics, too, and it's got a different name there, and I forget what it's called, but it's got – in things like chemical process control, this would be called the process noise. It's got all sorts – I'm trying to remember what it is in economics, in econometrics. I can't remember, but maybe I'll remember during – it doesn't matter. It's got some other name, but it's identical to this. And here also you have a V of T, which is a noise on your measurement here. So that's called sensor noise or error. It's also called measurement noise. It's got all sorts of names. The assumption here is that A, B, C, and D are known, but what you want to do is you're gonna observe input and output over some time interval, and you want to estimate the state, and there are lots of problems about estimating the state. You might wanna estimate the current state, the initial state. You might wanna estimate the next state – in finance, that would be of tremendous interest, right? To estimate what the returns for tomorrow would be. That would be of great interest. So there are lots of problems you wanna do, and this is gonna be based on U and Y. And this is basically gonna be a particular application of just this, and I'm gonna do some horrible notation. What I'm writing down now is notation from Week 5 or something like that. It's gonna be just a big problem of this form, which basically is you have some parameters you wanna estimate. You call it X. You have a linear mapping that maps X into some measurements, plus a corrupting noise, and then you ask a question like based on Y, what can you say about X. Okay? Well of course, there's the – we can do the platonic case. We'll get that. That's easy. The answer is it depends on the null space of A or something like that, but we'll get to that later. Then once you introduce the noise, it gets more interesting, and now you can say interesting things about how to do this. For example, you might use least squares here, and we'll see that's what's gonna be the same here, but it's gonna be more complicated, but you could map all of this big complicated thing into something that looks like that. It's bad notation because why? We're already using here – X is here and A is different from what it is or something like that. So let's see how that works. So the state estimation problem is this. You want to estimate the state at some time S from the inputs up to time T minus one from the outputs up to time T minus one. And if S is zero, you're estimating the initial state. That's initial state estimation. If you're doing S equals T minus one, that's current state estimation. If you're doing S equals T, that's called the one step ahead predictor is what you're doing because you're trying to guess what it's gonna be on the next step. And there's all sorts of other variations on it. If S is something like T minus five, some people call that a smoother or something like that because you're sort of using future observations to go back and make your most intelligent guess about what your state was five samples ago. If you were actually trying – if S is T plus five, you're doing a five state predictor, a five epic predictor. You're trying to predict what will the state be in five samples, five epics in the future. By the way, in a lot of these problems, despite using all sorts of fancy methods needless to say, your answer could be useless. That kind of goes without saying. If I ask you – if you can't predict tomorrow's returns, you certainly can't predict next summer's returns or something like that. That's silly. And that's fine. Of course, that comes up here. We don't dwell on it really there, but it's a fact of course. If I don't give you good enough measurements with which to estimate the parameter, then of course you can be as sophisticated as you like, and you won't predict anything that's worthwhile at all, and the same is true here. Maybe here it's kinda more dramatic. So if you have something that estimates a state, it's sometimes called an observer or a state estimator, and it's got other names in other fields, and it's used in lots of other fields. Now a generic notation which is pretty widely used is something like this. You would call X hat of S given T minus one is an estimate. It's denotes an estimate of X of S given input and output information all the way up to T minus one. Now the notation of course is meant to suggest something like a conditional expectation. This is again if you have that background, and there are in fact cases where X hat of S given T minus one is nothing more than the conditional expectations of X and S given input and output information up to T minus one. But here I'm just using it to say it is an estimate of X of S based on that information. It doesn't have to be a statistical method you use. We'll show you how to use least squares or something. That's the idea. Now by the way, I claimed that you can just do this right now because all you would do is put it in this form. I mean you would have to stuff the right matrix A and it would be a pain, and there'd be some debugging you'd have to get right, but I promise you all of those state estimation [inaudible], they look just like this. You have to figure out what X is. You would have to stuff A in the right way, and all that kind of stuff, and then you'd use least squares, and that would be that. Actually, that wouldn't be a bad – you'd actually make estimators that would be pretty good. We'll go through the more standard discussion of all this. Let's start with a noiseless case. Actually, that's probably a good idea to start for anything because if you can't do it in the noiseless case, then you shouldn't be trying to do it when there's noise, so we'll start with this. Let's just find the initial state with no state or measurement noise. Then you have X T plus one is A X of T plus B U of T, Y of T is C X of T plus D U of T, and what you can do is I can write down the output as – first of all, I can write it down as a function of the initial state. That's what we wanna estimate. That's a matrix OT. OT is this matrix here because the output Y of zero is C X of zero. Y of one is C X of one, and a portion of that due to X of zero is C A X of zero. That's this, and so on. Notice this thing is OT. O is for observer as opposed to C, a script C which is for controller. And it looks very similar to your friend B, A B, A squared B, and so on, except this one is stacked vertically. The C is on the other side, and it's C – and in fact, the components should be totally obvious here. Like for example, C A to the sixth is a block row in this, and it basically says that – it's easy to describe exactly what C A to the sixth is. It's this. It takes the initial state. It propagates it forward six samples, and then it takes it at C, and then operates to map a state to an output measurement. So C A to the sixth is the matrix which maps state now to what the output will see in six samples. I think I got that right. That's what C A to the sixth – so that's what all these are. It's a time propagator followed by a measurement matrix, nothing else. Now T – it may have been on like your first homework or stupid like that. T is basically – this describes the mapping of how the input maps to the output. I think it was – was that on the first homework or something? I think so. Something like this was anyway. That’s easy to work out. Again, you just have to stuff the matrices in the right way. T is for Toeplitz because that’s a Toeplitz matrix, and a block [inaudible] and it looks like this. This is what tells you how Y of zero depends on U of zero, and the fact that this is lower block triangular tells you – it means something. It means causality because this says that basically Y of zero – so I put U of zero down to U of T minus one. By the way, note that I’ve decided now to go back to writing Us in order, not in reverse time like we did before. I can write them any way I like. I just have to be clear about it. So if I do this, then this first row says that Y of zero is D U of zero and has nothing whatsoever to do with U of two, U of one, up to U of T minus one. That makes perfect sense because the output at time zero has nothing to do with what you put into the system at one, two, three and so on. It’s causal, and that’s what this is. And all of these things should make perfect sense. B C A to the T minus two B, that’s B takes you from an input, has an immediate effect. A to the T minus two propagates you forward in time. And C maps the effect on state to the effect on an output, so this makes perfect sense here.

Student:[Inaudible] dynamics [inaudible]?

Instructor (Stephen Boyd):No, it’s actually causal because we fix the initial state.

Student: So you estimate this only if they are looking at the initial state?

Instructor (Stephen Boyd):That’s right. No, sorry. No, it’s not. It’s as if the initial state – the term for the initial state is here, so I’ve separated out – well, no. I guess actually I do accept what you said because we’re looking at the initial state, therefore the initial state appeared here. If we were looking at the final state –

Student: Then we’d be on [inaudible].

Instructor (Stephen Boyd):You’ve got it. Exactly. But in any case, what I can say is this. This is just an exercise in matrix stuffing. You just need to put it in a form that looks like this. A measure that you know is equal to – in this case, it would go something like this. An output which you know is equal to a matrix times the state you’re looking for. I wrote it down for X equals zero – for T equals zero, but it could’ve been T equals five. It doesn’t matter. It could’ve been T plus five, meaning look five states in the future, so that would be that matrix is known times the thing you wanna get. And then plus another term which is something you know. So this thing here has a very strong meaning. This term here is very interesting. This is the effect on the output due to the input. Now you know the input, and you know that, so in fact you know this term. It’s actually your predicted output is what it really is. It’s what you’d predict the output was if the initial state was zero. So that’s what this is. And actually, that’ll just go away in fact, that term. So you can rewrite these equations this way. This way you write OT times X of zero – that’s what we want – is equal to something we know – the output, we measured it – minus – and then this is a beautiful thing. That’s something we know times something we know, so this is whole right-hand side is something we know. But the interpretation of the right-hand side is quite nice. This is what we directly measured. We also know this. Possibly we measured it. When we measure this and propagate it through the matrix, presumably that exists as 15 lines of C somewhere, so this actually goes down – this goes down in a computer. And this basically is your estimate of what you would see if the state were zero. You subtract that from what you did see, and this is – now it may exist only in some processor. It may not have to exist in practice as an analog signal. It may exist only in a processor. This signal over here on the right is beautiful. It's the output kind of cleaned up from what you – from the effects of the input or something like that. Let me try that again. It's the output – I've got it. I'm gonna try it again. It's the output with the effects of the input removed, but it was removed synthetically. It wasn't done in analog. It was done in a processor. That's what this is. Then this says okay what's left is only the component of the output which is due to the initial state, although this would be X of S – it'd be some other state of it, if you're estimating something else. And now you're back up to Week 3, which is to say you have a skinny – you have A X minus B with A skinny. Presumably you have more measurements than you have unknowns, and that's it. And you can do all sorts of things with it. There's no noise at all if you can just solve the equation with your favorite left inverse. If there's noise, you might wanna do least squares. That might be the right thing to do in some cases, but you wouldn't do it always, or you might not do it always. Okay, so with no noise we can answer everything. It goes like this. It says we can determine the initial state of the system only if the null space of this observability matrix is zero. That says the observability matrix, the columns are independent. That's what it says. By the way, note the nice kind of duality to what we looked at early with controllability. So in controllability you have B, A B, A squared B, and so on, and everything came down to – this idea of controllability came down to – in that one it came down to something very simple. It was that this fat matrix should be full rank – should be on to. In this case, it's a skinny matrix and it should be full rank. The meaning in the first case is there's an input which will take you from anywhere to anywhere else, at least in N steps. In this case, it says something like this. If I observe N or more samples of the input and output, I can predict the state exactly if there's no noise. That's what this says. So that's the analog of what it means to be full rank. In this case, you call the thing observable. Notice that the input here has no effect whatsoever on estimating the state. Now there actually are cases when this is not quite right because basically it can just be subtracted off immediately. This is very – now people would say that doesn't make any sense. I would rather get it – it's better if you're not doing anything – someone else would say no, you gather better data. You can estimate the state better if you're wiggling the control stick or something. No, it turns out these are all just wrong. It makes no difference because you can subtract it off. There is one issue. If it turns out that your model is not quite what the thing is, then here the true T is not quite what you think it is, and when you do the subtracting you might get left with some residuals or something. That's a secondary issue. In this case, it doesn't affect it at all. Okay. So again there's a dual to all of this, and I think I'll speed up because it's actually kind of boring, and there's some things I wanna mention at the – there's some stuff I wanna cover at the end. So by the Cayley-Hamilton theorem, you know when you go C, CA, C – when you get up to CA to the N, you've just added a bunch more rows which are linear combinations of rows above. So it says basically if you can figure out – if you can work out what the state is from measurements from U and Y at all, you can do it in N steps. As in the control case, it doesn't mean it's a good idea to do it. You may want to take much more data and then average – I guess so far we have no noise, so that comment doesn't make any sense, but when you add noise it will make sense. So I think we'll go over – I think what I'll do is just go forward to what would be obvious here. I've lost that [inaudible] least squares observers. Thank you. There we go. I got it. So the most obvious thing to do in the case when there's actual measurement noise is to set it up exactly a least squares problem, and you'd have this. You'd estimate your state to be the observer matrix pseudo-inverse here, which in this case if it's actually an observable system is O – I guess it's skinny, so it's O transpose then quantity O transpose inverse times this thing like that times this thing. And this thing has got all sorts of names. I don't know. Let's see. This is something like it's your guess of the output due to the initial state. That's what this signal in here is because this is your guess of the output that would be due to the input that you actually measured if there were no initial state. Then this is what you really did measure. The difference is the output that's due to the non-zero initial state. So that's how you get all of this. I can mention something about this, and I think I'll even – but I won't go on and do the rest. I just say this alone here, if you wrote this in recursive form, you would actually have a very rudimentary form of something called a Kalman filter which you may or may not have heard of. But this thing which is – this is just an exercise from like Week 4 of the class. It's very simple to state this. You get things that will work – these will work unbelievably well. I mean this beats anything, any kind of intuitive scheme you could make up if you wanted to track a vehicle or something like that. Just something this simple which would end up being just a handful of lines would actually make estimates of states and positions that would beat any intuitive method you could possibly imagine, but that's kinda the theme of the class. So actually, I think I'm gonna quit there because I claim all of this is just sort of variations on applications. We could look at one application but I don't know. It was gonna be – it's an application with our friend the range measurements. Maybe I'll just do that very quickly and we'll look at that. So here's the example. It's just to give you a rough flavor of how these things work. By the way, there are entire classes you can take on state estimation, so you can take one in finance, or you can take one in aero-astro, where it's used to track vehicles, or it's used for navigation. So you can take entire classes, probably three or four in wildly different fields, and it's just on this material. Some of them are kind of cool. Actually, a lot of them are kind of cool. So we'll just do a very simple baby example. It's a particle that's moving in R2 with uniform velocity. I have linear noisy range measurements from various directions like that. You can see they're all kinda concentrated from here. By the way, if I have range sensors from here, that's not great. If someone gave you a bunch of range sensors and then arranged them to make your estimation job easy, you'd spread them around. In fact, you might even do something like put – you'd probably spread them around evenly, maybe even at four cross points like this because then you're sort of measuring different things. A bad thing to do would be to put all the range sensors in the same place kinda like this. Right? Because if you put all the range measurements in the same place, are these getting making different measurements? Oh yes, they are, but they're not that different, so you're sort of – you can imagine now that your estimation might not be that good. So that's it. And we'll make the range noises have an error on the order of one. And we'll assume that the RMS value of the velocity is not much more than two. So actually, we can write this out as a linear system. To describe a particle with uniform velocity is just basically this. It says X T plus one is gonna – the bottom block says that – the bottom block of X T which is the velocity is equal to the identity times the velocity. It means it doesn't change. So the bottom two entries of X of T are constant, and the top two change – actually, they're [inaudible]. Then your measurement matrix is gonna look like that. So that's what you have. The true initial position and velocities are these. This makes no difference, but then here would be an example of a track here. What's shown here are that line shows you the true to the particles moving at a constant speed. That would be the range measurement if the range measurements were perfect, but you can see we put a lot of noise on the range measurements, so basically any single range measurement would be – a snapshot of range measurement would be completely useless. You'd make an outrageously bad estimate of the position and velocity, obviously from this. What you can do now is actually just carry out this least squares observer fitting, and we can actually work out the actual RMS position error, and you get something like this. It's hardly surprising what you see, but the velocity error actually goes to zero, and the position error actually will – it won't go to zero, but it will go to a small steady state number. So that's what it is. That's how these things work. And this is just a taste of it. Technically, you could write all the code to make this plot. You could've worked everything out and so on because it's just an application. So that officially finishes the material for the class. The truth is that the last two topics we looked at were just sort of applications, so a little bit more than an application, but they were not real material. The singular value decomposition was the last real material. What I wanna do now though is actually – I wanted to have a bunch of comments about the whole like how does it all fit together and things like that. So that's what I wanted to look at next. The first thing is I wanted to say a few things about linear algebra, not that I haven't been talking about linear algebra for a quarter, but I thought I'd just say a few things about it. So just to organize my thoughts, so it doesn't come out totally randomly, I wanna zoom out a little bit here. The first thing that should be kind of obvious – if you haven't seen it, you will see it or something like that, but the point is it comes up in a lot of practical contexts. It comes up in EE and mechanical engineering, civil engineering, aero-astro, operations research, economics, machine learning, vision – it goes on and on. What's interesting is that actually that wasn't always the case. So signal processing 25 years ago basically went on and on with how you process a scalar signal for example. You would barely – you could go grab an issue of [inaudible] signal processing. You won't find a matrix in it. Now totally the opposite. Everything involves ideas like – it's extremely rare to find something that doesn't. The same is true in economics, and in finance, and in things like that. In aero-astro and control 25 years ago, the idea of sort of a multivariable system with multiple – that was this exotic thing that only PhD level students would study. That's completely wrong. All real systems are designed this way, no exceptions. So that's just the way things work now. You don't design a control system for a modern airplane or something like that, or a process or any – you don't even make an ABS system or anything like that – you don't do it without matrices. And this is a change. This was simply not true 20 years ago. 20 years ago, there were people who talked about matrices, and they were made fun of in fact by the people who were designing systems that worked anyway. I wasn't one of those people. So the point that's – it should also be kind of obvious is that nowadays linear algebra you can actually do it. If you're fiddling around, you can play with MATLAB. MATLAB is a toy and it's not – I have arguments with people about this, but in fact no real implementation uses MATLAB, although there are people who say that's not true. It is a toy in my opinion, and real implementations are not done this way, and that's often true. Now 20 years ago a class like this, there would be zero competition component, none, absolutely zero. By the way, you can imagine how boring it would be. I just – we won't go there. It was kinda boring actually, to tell you the truth. Luckily for you, you were born later than this point. Now of course, you have MATLAB. You've been playing around with that. By the way, there are other variations on MATLAB you can use. There's Octave and things like that. And the truth is that these days if you know what you're doing with some object-oriented system, you can make it look like MATLAB except that it will actually be a real language. So Python could easily be made to look like MATLAB for example. So you could write things like Y equals A X times – you could write Y equals A X plus V and all the right things would happen. There's real codes. LAPACK is something. You should've just heard that name. I don't want you to leave this class without hearing that name. Even though we talked nothing about how to actually carry out these confrontations, it's important that you know that things by the way that MATLAB does, which does nothing but parse what you type in, the actual numerics is done by open source public domain software written by professors and graduate students. It's LAPACK. Then when you get into large systems, there's other schemes like that as well, but this is a good thing to know about. And just to give you a rough idea, I'm sure you kind of know this, but maybe no one made a big deal about it. If you have let's say 1000 variables, and you wanna solve – I don't know. Let's say – let's do 1000 variables – let's do a least squares problem that looks like this, 1000, and let's make this like 2000. It can be anything you like. It can be I have 2000 measurements to estimate 1000 parameters. By the way, this is obviously not a small problem. That's two million real numbers to describe this matrix. Everybody see what I'm talking about here? This is serious business. You could make a big story about how sophisticated this is. You'd be taking in 2000 measurements and doing the optimal – you could make a big long song and dance about how you're doing – I'm blending 2000 measurements to get 1000 estimates of blah blah blah. How fast do you think – how long does it take to do that? You might have some rough idea. Does anyone know? You type A backslash B in MATLAB for this.

Student:[Inaudible].

Instructor (Stephen Boyd):What? You're right. About a second. So the answer's about a second. Might be two seconds, I don't know, something like that. What? 2000 by 1000. Do it twice though so it gets loaded into – okay, sorry. But you know, it's on the order of a second, two, something like that.

By the way, this is just stunning, the fact that you can do this. This is something unheard of. This is what happens when you ignore Moore's Law for let's say 20 years or something, and then come back and check again how fast it is. This is the kind of thing that happens. Now in fact – and this is a beautiful topic. It's something actually some of you might wanna look at. By the way, I don't – I think things like this have not really propagated too far, so there's a lot of times when you might wanna solve – do least squares with 2000 measurements to estimate 1000 things. That's a big thing. That's like a big system. The fact that you can do this in seconds with total reliability I think actually has not diffused into general knowledge. Like I had a professor in my office, and I asked him to estimate how long this would take, and he said half an hour. I won't reveal his name.

Student:[Inaudible].

Instructor (Stephen Boyd):How many? Four. On your laptop.

Student: No, on [inaudible].

Instructor (Stephen Boyd):Okay. Four seconds. There we go. So it's four. That's because it's too warm. Okay, fine. Four seconds gives you a rough idea.

Actually, an interesting thing that we haven't talked about at all – as it turns out, if you wanna do things like least squares, solve linear equations when – you can do this for much bigger, way bigger matrices provided they have special structure. Now, special structure – the typical one would be sparsity. Sparsity means a lot of the entries are zero, in which case there's methods that are just amazing and extend to things like 10,000, 40,000, a million depending on your luck. There's other structures like for example if it's banded or tri-diagonal and all this kind of stuff. And there's just a whole world of knowing the structures and how they imply fast methods. And this would take you to things like how people solve PDEs and all sorts of other stuff. That's worth knowing. Actually, I encourage you in the future to learn about these things. So please don't live your life just in MATLAB. There's something – even if you do, please look at – know what's under the hood. And actually, know then what would happen if – even if you have no intention of ever doing it – of actually figuring out what a real implementation would do as well. That's a good thing. I won't say much more about that. There's whole classes you could take on this at Stanford, entire ones, literally two on numerical linear algebra that I know of. Actually, I know of a couple others, so there's maybe two or three. You can even take one in geophysics and all sorts of other – Now I wanna say something about linear algebra and sort of my view of how it all fits together. I think there's really like several levels of understanding. You could add a fourth one, but the most basic is this. The most basic level of understanding is kind of at the high school level, and it goes like this. What you should know is this. If you have 17 variables and 17 equations, there's usually a solution, and that's when you were tortured by solving whatever three equations and three unknowns for no particular reason whenever you were tortured doing this. And then maybe sometimes 17 equations and 17 variables, there wasn't a solution, or there was more than one, and somebody waved their hand or something like that. And the same high school view would say something like this. It's actually just degree of freedom accounting is what it is. So for example, if you have 80 variables and 60 equations, you just do some accounting and you say, "Well, I've got 20 extra degrees of freedom." You could use those degrees of freedom for anything you'd like, but there's basically a 20 – this is roughly – a 20-dimensional subspace of freedom. You could use it to minimize the two norms or the Euclidian norm, which is what we've been doing, and you get a least norm solution. You could use it to minimize something else. You could use it to do anything, other stuff. There's just degrees of freedom you can absorb. Of course, if that’s an estimation problem, this is probably not good news because it basically means that the only thing you can do is – you don’t have enough measurements to estimate what you want. Now so it’s very important to always start I think when you look at a problem from this point of view, just do the basic accounting. What are we talking here? Do I have more measurements than unknowns? Less? How many degrees of freedom do I have less than all that? Now of course, what you know at the next level, you learn things about singularity, rank, range, null space and so on. And what that would be for example would be here. If someone asks you, you’d say, “Look, I have 17 equations and 17 – but there’s no solution.” Why is that? And you go, “Ah well, you see that 17 by 17 matrix was singular.” And I go, “What does that mean?” And you say, “Well, what it means is this. There’s two ways to view it. The first is you thought you had 17 different equations. Actually, you don’t. You have 16 because equation No. 13 is derivable from the others.” Everybody know what I’m talking about here? So in fact, I’m talking about the left – I’m talking about the rows being dependent. They could say something like this. They’d say you thought you had 17 variables, 17 knobs to wiggle, or whatever you wanna call it. That’s what you thought you had. Actually, you had 16. You know why? Because the action of this fourth variable can be exactly replicated by an action of these other variables. Okay? So you thought you had 17 knobs, 17 design variables, 17 inputs, whatever you wanna call it to mess with, but you didn’t. And that was hidden in this matrix, which was 17 by 17 so it wasn’t obvious to see by the eye. Okay? So that’s the next step. Now, the platonic view – this is really basically called math. It’s very important because – and it basically says something like this. It says that this matrix has Rank 5. It says this is full rank, so it really certifies that there really are 20 extra degrees of freedom here. That’s very important to understand. Now I’ll tell you the good thing about it. Now the good thing about this is everything is precise and unambiguous. There’s no such thing as – there’s no waffling, no nothing, and you don’t – there’s no nonsense. It’s either singular or it’s not, period. Its rank is 17, or it’s 15, but it’s not like anything in between, and it doesn’t depend on who’s looking at it, or anything like that. Things are true or false. What’s nice about this at least – it’s nice because it’s very clear and unambiguous. Unfortunately, it can be misleading in practice, and we’ve already seen at least one example of that. For example, if I have a 17 by 17 matrix and it’s singular, if I pick literally any random entry and perturb it by an arbitrarily small amount, the matrix becomes nonsingular, and I say done. It’s all done. Now you can invert it. And then you’d better leave quickly before they actually – but theoretically they can now invert it, and there’s no problem. In this case, it would be misleading to say for example to make a statement, which you could on the basis of just a platonic mathematical view – you could make the statement that nonsingularity is not problem in practice. Then they’d say, “Why would you say that?” You’d say, “No problem. Any matrix that’s singular, if I perturb its entries by random numbers so small that they couldn’t possibly affect that practical application, I guarantee you with probability one that the resulting matrix is nonsingular. Done. End of story.” At this level, you can’t argue with it. It’s correct. That’s interesting. That’s a mathematical argument for why you don’t need to know the math. Did you notice that? That’s what it was. Now at the next level – this is Level 3. These are sort of levels of – the next level we have the quantitative – now the quantitative level, it’s based not on qualitative things like Rank 3. It’s based on ideas like least squares and SVD. So there you’d say things like – someone would say, "What's the rank of that?" And you'd go, "Ah, it's around six." And they'd say, "What are you talking about? How can the rank be about six?" And you'd go, "It depends on you know how big your – what's your noise level, and what do you consider small and big." So it's a bit wishy-washy, but it's actually more useful in practice. As long as you don't think when you're talking about this that you're actually doing math, which you're not at that point, so you have to keep them very separate. So this is very useful, and in this case no one would be fooled by someone coming along and offering to sprinkle some perturbation dust on your matrix to make it nonsingular. No one would be fooled by it because they'd say, "Yeah. Go ahead. Sprinkle all the dust you like." You sprinkle all the dust on it, and all that happens is one of the singular values, the zero now becomes ten to the minus ten, and you say, "Are you done?" And the person says, "Yes. It's nonsingular. I'm leaving now." And you say, "Well, it may be nonsingular, but the minimum singular value is ten to the minus ten. I can't do anything with it that I would want to do with a nonsingular matrix because for example the inverse is gigantic." The condition number is still ten to the ten or more. Now the interpretation in this analysis, it really depends on the practical context, and these ideas are extremely useful in practice. So I think these ideas by themselves I think are – if you sort of have arrested development here, it's maybe not so good, arrested mathematical development I'm talking about. Actually curiously, arrested development here is probably okay in my opinion. It doesn't lead you to – because you know you don't know a lot of things, and it's no problem because you're not expecting to work, and you type A backslash B, and it says singular [inaudible], you go, "Oh well. I'll try something else. This is not a big deal." Here you can actually be a little bit dangerous because you can walk around and talk about things like controllability and things like this, and singularity, and rank, and so on. This is actually quite – this stuff is actually very useful in practice. And I wanna talk about one more level above this, and that would be the algorithmic and computational sophistication. So that would be – another level beyond this would be to actually know how fast and how well you could actually carry out those calculations in practice. And that would be for example what it would take for example if you were going to let's say start Google. Let's just say because that's nothing but a singular value decomposition, but it's one based on an eigen in a billion by billion matrix. And I can tell you right now you can't type a billion by billion matrix into MATLAB and ask it – and type SVD of it. Okay? So you have that next level where you know how fast you can do things when you do them in real time. You can know things like that's a big matrix, but you know what? It's very sparse. I bet we can do this. We didn't talk about any of these things, but you can do things like calculate the – let me just give one example of that, and then I'll move on to the next topic. Here's the last example. You remember early on, there was this idea of representing a huge matrix as an outer product. If you did this, for example you had a method to speed up simulation of it by some huge amount if you've succeeded – and later, early on you learned that the smallest factorization you could do depends on the rank. It is the rank. And you learn a method for doing it, QR. Fine. Later in the class, when we studied SVD, you learned a method for getting an approximation that's low rank. Then you can take a matrix that's absolutely full rank, giant, and you could say, "This three rank approximation is good enough," and you win big because you can now multiply it so fast, it's amazing. You can do compression. You can do all sorts of crazy stuff now. Now that works if A is up to say let's say 1000 by 2000 or something like that, maybe 2000 by 2000, maybe a bit more.

But the real interesting one is when A is a million by a million. You can't store a million by a million matrix, let alone compute its SVD using the LAPACK routines obviously. And yet that's exactly the case where a low rank approximation would be like super cool because then you can say, "Hey, you know your million by million matrix" – people would laugh at you. They would say, "It's not a million – you can't even store a million by million matrix, so don't tell me about it." And you go, "Yeah, but what I'm telling you is I have a Rank 3 approximation of it." So there you can't just form the matrix and call SVD. That's ridiculous. There are methods that do the following, that will actually calculate the Rank 3 – that will calculate the SVD sequentially, so they'll – and they're extremely efficient and fast. The only thing they need of A is you have to be able to multiply A, and you have to multiply by A transpose. You have to provide routines for doing that. So if A for example is the forward mapping in for example MRI, or PET, or anything like that where you have a fast method for example based on Fourier transforms, you can actually do low rank approximation. I can tell no one has any idea what I'm talking about. That's fine. All I'm saying is there's another level of sophistication where all of the ideas here you can use on very large systems. So let me go on and just answer one more question, or address one more issue. It's like what's next. So there's lots and lots of classes – now of course, after you take the final, you may have a different view of all these things. Oh, I forgot to tell you. Since we give our final, or rather – I do think there's no way we can construct our final to be within the – it's a blatant violation of the registrar's rules, by the way – so at first I thought we could just call it Homework 10, but it counts for a lot, but it turns out you're not allowed to assign homework nonetheless – Anyway, so I should mention when you're doing the final, if you wanna take a break, you can go fill out the teaching evaluations which is fine with me. That's why we have tenure, so you can do anything you like, but if you get really pissed off, you can always do that just to unwind for a little bit. And that's actually I think a – it's possible because of our flagrant and open violation – and now on video violation of the registrar's – sorry. We'll go back to that. Yes. So there's a sequel to this class, but it won't be taught until next year. Now I am teaching this class, and this class 364A and B winter and spring, so I should say a little bit – I mean there's zillions of other classes you could take on all the topics I've been talking about, but let me say a little bit about what this class is about. There's a simple way to do it. This class, a lot of the techniques here relied on sort of least squares – least squares, least norm, and some variations on that, and so there were analytical formulas like A transpose – and so I can tell you what this class is. This class basically says what if you – it extends least squares to other things like the maximum value or the sum of the absolute values. Also, in 263 we don't really have any way to handle things like to say that a vector is positive. You see what I'm talking about? Like if we do Y equals A X plus V, and X is a power or something like that here, if we do least squares and then X would be a vector of powers, Y would be some observed temperatures or something like that, that's measurement error. We have no way of telling least squares that X is positive. We would just calculate a least squares solution, and if we're really lucky, the estimated X will come out to be positive. If not, it'll be negative and we'll have a bit of a problem. So it turns out – by the way, people have dealt with this for years using all sorts of ad hoc methods, in fact using all the ideas you know about like regularization and things like that, and they're completely ad hoc. It turns out you can handle those flawlessly. You lose your analytical formulas, but you can calculate just as fast as you could before with least squares. It’s even more so when you do things like when you design inputs, which we’ve been doing. We just did least squares inputs. The least squares input that takes a system from here to a zero state, or from one state to another, you could do things like add conditions like this, which is actually kinda cool. And these are quite real now. You could add a condition like that. That’s a real condition. You will not find if you go up to an MRI machine a note that says under no circumstances shall the sum of the squares of the RF pulse be more than this. It will not say that. Or when you buy a motor for a disk drive or something like that, there’ll be current limits, there’ll be things like that, and they’re gonna look like this. It turns out you can solve those problems exactly, and that’s what 364 is. When you go to 364, there actually is a lot more fun problems you can solve. You can actually really do all sorts of things like finance, and machine learning, and stuff like that. So that’s what this class is about. I’d be happy to answer any questions if anybody has any. Actually, it’s a bit embarrassing. I believe next week I’m gonna have to give the first lecture for it because I’ll be in India when the actual class starts, so I’ve never done that. I’ve never taped ahead the first lecture before. I’m deeply embarrassed, but it was gonna happen sometime, so it will happen. All right. So with that, I’ll say the class is over. I have to thank Jacob, Yung, and Thanos for an enormous amount of help putting it all together and all that kind of stuff. And unless there’s questions, we’ll just quit here. I mean I’ll be happy to answer questions about these or other classes and stuff like that, but otherwise thanks for slogging through it. Oh, and enjoy the final. I forgot to say I’ll be in Austin while you’re taking most of it, but then I’ll be back Saturday early morning, and I’ll be in touch. I’ll be around Saturday. Good.

[End of Audio]

Duration: 69 minutes