IntroToLinearDynamicalSystems-Lecture18

Instructor (Stephen Boyd):Let me start with a couple of announcements. One's half announcement, half apology. The current homework, I think the last problem requires material that we haven't covered yet, but I'm going to cover it right now. I sent an email out about that, so I hope that – but there were a lot of people panicked, asking me yesterday, how can I do this and this sort of stuff. I guess for a lot of you, this didn't bother you.

The other announcement was that someone pointed out that last time, I said something about homework eight being – I think I said it was due Thursday, wrote down Friday or something like that. For full inconsistency, we should've had the web site make it due Wednesday or something. So I didn't achieve full inconsistency, but near. Of course, in that case, the rules are simple. The precedence rules. The web site is right.

Then I think that worked out perfect. Usually what I say is wrong. That would be the least reliable. What I write would be slightly more reliable, but I hope that didn't cause any confusion. The other thing is just to make sure you know, we posted homework nine. Obviously, that's the last homework, so that's posted now. We're going to finish up what is essentially the last topic in the class today. We'll finish it.

We'll do another bunch of material, which is very interesting, but it is nothing but applications of this stuff we've done so far. Any questions about last time? If not, you can go down the pad. I guess I've been trying to motion to you that you're focused way off in the wrong place there. You're going to want to move over here.

There we go. So if you go down to the pad, we'll finish off our coverage of SVD. So we'll go to matrix solving square system of equations. Y equals AX, so you have N equations, N unknowns. That's Y equals AX, and of course, the solution is nothing by A inverse Y. So that's the solution.

Now we're going to look at what would happen if Y were off a little bit. Y might be off a little bit for many reasons. I could be as innocent as a floating point round off, or it could be that Y is measured with finite precision, which, of course, is always the case. Is that twisted at an angle of ten degrees for you? Okay. Just curious.

Getting closer. Well, we'll just not worry about it. It's still twisted, but that's okay. So we'll look at what happens when Y varies. Of course, if Y varies a little bit, then X will vary a little bit, and the change in X will be A inverse delta Y. Last time, I think, I pointed this out, but if you have a matrix, which is invertible, nonsingular, but where the inverse is huge – and of course this is exactly what you'd get if you had a matrix which was, for example, singular, and then you perturbed it slightly to make it nonsingular. You will have a matrix that's now nonsingular, but it’s inverse if going to be huge.

Now, when I matrix has an inverse that's huge, you can solve Y equals AX, no problem. That's what it means to be invertible. Here's the problem right here. It says the tiny errors

in Y can become extremely large errors in delta X. So that means that it may mathematically be invertible, but it is of no particular practical use, unless you're willing to certify that there are no errors in Y out of the 15th digit or something like that. So that's the difference.

If you want to bound that, you can say that the norm of delta X is going to be less or equal to the norm of A inverse times norm of delta Y. So in this sense, it says that when you see that the norm of the inverse matrix is huge, that's a hint that small perturbations in Y can lead. It does not mean it's guaranteed to. It means can lead to large changes in X.

Now, when you're looking at a perturbation, for example, delta Y or delta X, as to whether or not it's big or small, a far better measure than simply the absolute number is the relative error. Relative error is something like this. It's delta X over X. So if this is, for example, 0.01 delta X over X, you can basically just say that means it's off by say, one percent. The same here.

For example, if you have a ten-bit A to D, when you measure Y, assuming Y's a measurement, then this would say over here – this number here would be less than or equal to something on the order of 0.01 percent. That's two to the minus ten, roughly. Okay.

Now by noting that Y in norm is no bigger than the norm of A times norm of X, and then simply dividing this inequality by the previous one, you get a very, very important inequality. We looked at this last time. It's this. It says that the relative error in X is no bigger than the relative error in Y times the norm of A times the norm of A inverse. Now, this comes up all the time, norm A, norm A inverse, and it's got a name. It's called a condition number. It's sometimes written cond A. So that's – it's a very famous quantity. It's often denoted using kappa, and you can write it in terms of singular values easily because the norm of A is, of course, the largest singular largest value of A. The singular values of A inverse are one over the singular values of A. Therefore, the largest singular value of A inverse is one over the smallest singular value of A.

It's the spread in the singular values, the range. By the way, geometrically, this is an N isotropy. That's what this is. If this is two, it says that the gain of the matrix varies no more than a factor of two with direction. That's the meaning of this. It says that the image of the unit ball under A will map into an ellipsoid that is not too skewed. No semi axis is more than a factor of two if two is what this is – just two, period. No semi axis is a factor of two longer than any other semi-axis. That's what this says.

So this, you can think of the condition number as a measure of being nonisotropic. In other words, it's how much the gain varies as a function of direction. That's exactly what the condition number is.

A couple things to note about it. You will have some homework problems on this, but some are obvious. One is that this number is always bigger than one. So it's always bigger than one. The second is the – that's obvious because it's the maximum singular

value divided by the minimum singular value. Another factor is it's homogenous. If I multiply A by ten, the condition number doesn't change. So the condition number in no way changes. That makes sense because we're doing relative error here.

See, if I multiply it by ten, each of these goes up by ten, or if you like, this goes up by ten, that goes down by ten, and it preserves the condition number. So this normally interpreted something like this. The relative error in the solution, X, is less than or equal to the relative error in the data, Y, multiplied by the condition number. That's a [inaudible].

However, in the '60s, maybe early-'70s, things like this weren't appreciated deeply enough. As it spread, it became, in fact, almost a little counterproductive because this sort of became almost a religious thing. In other words, people would simply say, oh, condition number's too high. Forget it. In terms of, this is just an inequality, and in fact, there's lots of cases where, due to special structure in A, this actually works. You can have a condition number of ten to the eight, and actually, the solution's are computed reliably. In other words, there's actually no change in the relative error.

Of course, there you have to look – this is just a bound, and there you have to look more into the actual properties of A. But there are many problems like that. In fact, some that were abandoned, there were methods abandoned in the '70s on the basis of large condition number. So people would say, oh, the condition number of those matrixes are huge. You can't do that. Everyone knows that's bad practice. It turned out, actually, the solutions were being computed quite accurately, but these methods were then out of circulation for 30 years and then rehabilitated five, ten years ago.

So a looser way to say this is you take the log base two of both sides, and you actually get something that's – it's a very good way to put it. It says something like this. It says that the number of bits of accuracy in the solution is about equal to the number of bits of accuracy in the data minus log two of the condition number. Let me say a little bit about what that means. What I'm saying is something like this. If this number is, let's say, one percent, if that's 0.01, it basically says, I know X to one percent. If I know X to one percent, it takes about – oh, somebody help me out. Eight? Seven? Six and a half. Okay. So that's six and a half bits. I should've taken 1,000, should I?

So let's make this 0.1 percent. That one I know. That's two to the minus ten. So if this number is 0.001, then this thing here, it basically says, is about ten bits of accuracy in these. That's not quite right because this is norms of vectors and it's not the same as the norms of the individual entries and the [inaudible] entries, but roughly speaking, we'll say there's about ten bits of accuracy in this at that point. Certainly, the 15th bit in any component of delta X is meaningless. That is certainly the case here.

So if you accept that definition of what it means to have the number of bits of accuracy, then this is actually correct. What this says is that when you solve linear equations, when you invert equations, what happens is the following. I should mention, by the way, the exact same thing works for mapping X into Y because note that the condition number is

independent. You can swap A and A inverse and it's the same. So in fact, multiplication by A, in fact, that same analysis holds. It gives you exactly the same decrease, potentially, the same bound and decrease in accuracy in solving. So A and A inverse are kind of the same from that point of view.

What this says is in terms of this bound, you can only decrease accuracy. The only matrixes that will not decrease accuracy are ones with condition No. 1. Condition No. 1 means sigma max equals sigma min. That means all singular values are equal, and that, in fact, only occurs if a matrix is a multiple of an orthogonal matrix. It's actually kind of obvious because you write U sigma V transpose. If, in fact, all the numbers on the diagonal of sigma are equal, which would happen if the condition number is one, sigma is actually – it's something like sigma max, which is the same as all the sigmas, times I, this thing.

That comes out, and it basically has the form sigma max I times UV transpose. That's going to be an orthogonal matrix. So this is why people who do numerical work and people at one end love orthogonal matrixes. They'll do anything to – so in their algorithms, they're multiplying for orthogonal matrixes. For that matter, also solving equations with orthogonal matrixes.

It's not just because it's easy to invert an orthogonal matrix because it's the same as the transpose. It's not just that. It also has to do with liability. So various transforms you'll hear about DCT, the Discrete Fourier Transform, all sorts of the common transformations you'd see actually have this form. The other people who love it are people in signal processing for the same reason, although the values of these things are off considerably. If you're doing numerical analysis, you're worried about errors in floating point numbers. So you're worried out there at the 14th, 15th digit. If you're doing signal processing, the third digit is probably suspect, but it's the same principle. That's the idea.

So the way you'd say this is if a matrix has a small condition number, you'd say it's well conditioned. It's called poorly condition if kappa's large. Whether it's small or large, that depends entirely on the context. If around you are people who do numerical analysis, and the types of errors they're worried about are things like floating point round off, but that's all they're worried about, then a matrix with a condition number of 1,000 would actually, in many cases, be considered well conditioned, from a numerical analyst point of view. That would be well conditioned because at that point, that's not a big enough number to have you worry about how floating-point errors are going to propagate forward.

If you're doing signal processing, or you're doing anything in an engineering context, then a condition number of 1,000 means that you can potentially lose ten bits of accuracy. It means you can start with 14-bit signals, and you'll end up with something with four bits of accuracy. Or if you start with a 12-bit signal, you'll end up with something with two bits of accuracy, at which point, you're kind of right at the boundary – so you're at the "why bother" boundary because you're just computing stuff that may not make any sense.

So poorly conditioned, if you do numerical analysis, that starts maybe to round ten to the six, goes up to ten to the eight, things like that. Ten to the six and ten to the eight in signal processing or estimation or statistical context, we have a name for a matrix like that. It's called singular, meaning there's just no point. Anything you compute, if the number ever had anything to do with any actual real measurement or something like that, certainly can be complete nonsense. They might be right, but that's because this is just a bound here.

But it says that you cannot get a mathematical argument at what you're computing – it could just be noise, and artifact of your ADD or something like that. By the way, the same analysis holds for lee squares. If you have a nonsquare, you have sigma max over sigma min. Here, this is simply a dagger is put here, and it's exactly the same. So the same story analyses these things.

Like I said, you can take an entire quarter that's on this topic and related topics at Stanford. There's multiple quarters that you can take that does this kind of stuff. This is very important to know.

Our almost last topic in the singular value decomposition has to do with low-rank approximation. Actually, I mentioned this way early in the class. We talked about rank, and I said, well, if a matrix is low-rank, you can write it as B times C. You can make a skinny/fat factorization of the matrix. We talked about applications of this and what it would mean. For example, multiplying by A, if you have this factorization, is way fast now. So if this were a million by a million matrix, and I factored it as – first of all, I couldn't even store a million by a million matrix. Let's start with that. Much less could I multiply a vector by it. But if it happened to have rank ten and I factored it as million by ten, ten by a million, no problem. Not only can I store it, I can multiply by it, and the speed-up factor is just something absolutely unimaginable.

So we talked about the idea of what does it mean if something's low rank. That's just one application. At the time, I said this would become much more powerful when you have a method for actually calculating – by the way, we found a method for calculating B and C this way. QR factorization was one way. So QR factorization did it right off the bat. So QR gave us a skinny factorization.

I mentioned at the time that later, we're going to have a method that does this. Then you have something like this. This is actually really interesting because it turns out there are lots and lots of matrixes whose rank is a million but which are almost rank ten. You already know what that means. It means you have a million by a million matrix, and if you work out singular values, they're all positive. Therefore, its rank is a million.

But if, in fact, the singular values of a million by a million matrix are big, up to ten, and the precipitously drop, but remains positive, it means that matrix is almost rank ten. We're going to see how that works right now. That's this topic. The use of this is – I mean, there are so many uses, it's amazing, and there are many more, I'm sure, still to find. So that's what we're going to talk about now.

So let's take an N by N matrix with rank R, and that's its SVD expansion here. So that's the SVD expansion. What we're going to look for is a matrix of rank P, which is less than R. We want this to be a rank P approximation of A. How are we going to measure approximation? We're going to measure it by the norm of A minus A-half. That's the matrix norm, so that means it's the worst-case gain of the error.

By the way, there are other ways to measure – there's other norms on matrixes. You'll encounter one, I guess, like on upcoming homework. Another one is something called the Frobenius norm. The Frobenius norm is nothing, but it's actually this. I'll write it this way. I think this works. It does. So this says string A out as a vector and treat it as a vector in RMN. This is nothing but this. It's the sum over I and J of the entries, separately. It's that. You'll have a homework problem on that, but basically it says treat the coefficients of the matrix as a vector, and take the ordinary Euclidean norm. So that's another one.

But this is the matrix norm. Here's a solution, actually, for both cases. It's this. The optimal rank, P, approximation of a matrix, A, is given by this. You simply truncate the SVD. You write out the SVD. You gather P diads. If you want rank P, that is, in fact, the optimal rank P approximation. This norm, A minus A hat, what's left over is – since A hat is the first chunk, it's the head of the SVD expansion, what's left is the tail of the SVD expansion. That's this. This matrix here has norm sigma P plus one.

So that's actually quite beautiful because it gives you an interpretation of actually what sigma three means. It's actually quite beautiful, sigma three. This gives you a perfect interpretation. It says it's the error by which you can approximate A by a rank two matrix. That's what sigma three means. We'll get to that in a minute.

Now, this is completely intuitive. It's not obvious, but it is – of course, there are a lot of things you have to remember that are intuitive and false. That's actually why you're here. If everything intuitive were true, then we wouldn't need to do all this stuff. This makes sense. It basically says something like this. You rank the diads in order of importance. The sigma gives you order of importance. You take as many as you can afford in your rank budget, which is P.

This gives you the rank of P, approximately. There's lots of applications of this immediately. It says here that if you have to multiply by a matrix many, many times, let's say in some real-time communication system or in anything else, you need to multiply by a matrix many, many times. What it says – for example, that might be some forward simulation. Who knows why you need to multiply by a matrix. Let's say you do.

It says what you might want to do is look at the SVD of that matrix. That's offline. That could take minutes or a day. It doesn't really matter to compute. You look at the SVD of A, and you decide how many singular values you feel are significant for your application. What kind of error will you accept? Then you form the leading – you choose a rank P, you get a rank P approximation like this, like a rank 10 approximation. Now you have a

method of multiplying by A, approximately. If you truncated anything, you're no longer multiplying by A. By the way, you can bound the error.

You can never be off in a worst-case sense by more than sigma P plus one times the norm of what went in. So this is not just approximation. It's approximation with a guaranteed bound. So you already have lots and lots of applications of this, of getting a low-ranked matrix.

For example, how do you get a rank-one approximation of a matrix? What's the best rank-one approximation of a matrix?

Student:[Inaudible].

Instructor (Stephen Boyd):How'd you get it?

Student:[Inaudible].

Instructor (Stephen Boyd):Rank one means I want to write my matrix A is BC transpose, like that. I want to write it as an outer product. How do I get the best outer product measured?

Student:[Inaudible].

Instructor (Stephen Boyd):You'd take the largest singular – you'd take U one. In fact, it's this. Here, I'll just write it out. It's sigma one, U one, V one transpose. That's it. That's the best rank-one approximate of a matrix. Let me ask a question. What would be the properties of the singular values of A under which it can be well approximated by a rank-one matrix?

Student:[Inaudible]

Instructor (Stephen Boyd):What is it?

Student:[Inaudible].

Instructor (Stephen Boyd):That's it. So if, in fact, sigma one is much bigger than sigma two, sigma three and all those, and those are small. By the way, what is the exactly condition that makes A rank one? Then there's sigma one equals zero, and if you – well, I use the loose interpretation of sigma two. Sigma two, sigma three and so on are zero.

But if sigma two, sigma three are small, it says that will be well approximated by rank one. Actually, I'd encourage you, in anything you poke around in, to look at these things. It's absolutely worth your while. There are shocking things that occur all the time. So when you work out a matrix that means something, some mapping, if you're doing something, look at the singular. In fact, it's just good practice to know them and look at them all the time, just so you know what's kind of going on.

Also, you'd be shocked. A lot of matrixes you think you've measured a lot, or you think you have a lot of degrees of freedom or a lot of control or something like that, you have an airplane and you put all sorts of thrusters and things like that on it and little flaps and vector thrust, you think you are way over actuated. A quick check of an appropriate SVD will tell you, actually, whether or not you really have eight or twelve actuators. You might only have three or something like that, roughly, but this is the kind of thing that would tell you that. Okay.

So let's prove this. We're going to show this in the case where the norm is the matrix norm. I should mention, this is from maybe 1920 or something like that, roughly. I could be wrong, but I know that it can be traced – I think it's associated with a name, Jan or something like that. It's also a mainstay of a lot of semi-modern statistics. Semi-modern statistics are statistics after Fischer in the '20s, but before the most recent one based on a lot of L1 and convex optimization methods. But that's not the story. So semi-modern was in the middle, that'd be '60s through '80s, and a lot of that is based on low-rank approximation, SVD and all this. There, it's called PCA.

I should say, this is actually a highly nontrivial result. There are basically no other examples where any optimization problem involving rank constraints can be solved. Absolutely none. The most minor variation on this problem, and it's impossible to solve. I point this out because some things are close to solvable. Others are not. For example, if you say – if you had any other condition on these, it's just no one knows what the rank approximation is. It's just this one isolated result.

So let's look at the proof of this. It's actually quite interesting how it works. It works like this. Let's let B be any other matrix whose rank is no more than P. Then let's look at the dimension of the null space at B. Well, that's got to be bigger than [inaudible] to N minus B because the rank of B plus advection of the null space is going to be equal to N. That's the preservation of rank. So in other words, if your rank is small, that says that your null space – if you have an upper bound on the rank, you have a lower bound on the dimension of the null space. They add up to a number, N, so it's kind of obvious.

What we're going to do is this. I'm going to take V1 through VP plus one. These are from the SVD. These are the P plus one most sensitive input directions. They're mutually orthogonal. This here is – I'm going to give a name for this. It's actually very important to get the right intuition here. Certainly, span of V1 is simple. That's a line, and it is the input directions which have the highest gain for the matrix, A. Span of V1 and V2 is a plane. It is the plane on which A has the highest gain. Now, that's rough, and you have to say exactly what that means in the right way, but that's sort of the idea.

So this is something like the P plus one dimensional subspace along which A has the biggest gain. That's what this is. I'm just giving you an interpretation. Nothing of what I just said is going to be needed in the proof. It's just to interpret what this is. Now the dimension of this is P plus one.

Very interesting, we're in RN, and we have the following. I have one subspace that's null space in B with dimension N minus P, and another subspace with dimension P plus one. They have to intersect not just at zero. In fact, all subspaces intersect at zero, but they have to intersect at a point that's nonzero. So there has to be a unit vector Z in RN, which is in both this subspace and this one. Actually, if that were not the case, then if I take the vector sum of the two subspaces, then I get a subspace of dimension N plus one in RN, which would substantially decrease my credibility. That's worse, even, than saying Thursday and writing Friday.

Let's see what it means. This says the BZ is zero, but Z is in the span of this thing. You can actually see what's going to happen now. It basically says – remember, what I want to show is that basically, B is not a good approximation of A. So B actually annihilates Z. But Z also happens to be in this high-gain subspace. So because Z is in the span, it's going to get amplified by A by some minimum amount. That's going to give me a bound on how good you could be. So A minus BZ, well, BZ is zero. That's AZ, but AZ is this.

Z actually, here, only comes into the first P plus one of these, like that, but this thing, here, sum VI transpose Z squared from I equals one to P plus one is actually the norm of Z squared, which is one. The reason is Z is in the span of V1 through VP plus one. So that's what happens here. So these numbers add up to one. These are non-negative numbers that add up to one, and this is a combination of sigma Is from one to P plus one squared.

So that's certainly bigger than the smallest one, which is sigma P plus one squared times norm Z.

This says – I found a vector Z whose length is one, but when multiplied by A minus B, it came out with a norm sigma P plus one, at least or bigger. That says that the gain of the matrix, A minus P, is bigger than sigma P plus one. I should mention that probably you've done that on this homework a couple of times, but I'll just mention that. If I have a matrix A and I say, what is the norm of it? If it's bigger than two by two, then you'd say, I can't answer that. I need a computer, or something like that. Then the person says, no, give me a bound or something on it, roughly. Then there's one thing that always works. If you find any nonzero Z and you just calculate this, that's a lower bound on the norm of A. But I think maybe you've done that on the homework that's due today. You've seen arguments like this. That's a lower bound on the norm, so that's the argument I'm doing here.

So this establishes this low-rank approximation idea, and this allows us to say all sorts of stuff. It actually gives a beautiful interpretation of sigma I. It says that the Ith singular value is the minimum matrix norm distance to a rank I minus one matrix. So it's quite beautiful. That's what it says. For example, sigma one – have I got this right? Does that look right? That's right.

So if I ask you for – I'm looking at this. Is anyone else confused, or is it just me? It looks completely wrong. Let's just go through it and see what it really is supposed to be. Is it N

minus I plus one? It's this? I'm just confused? It happens occasionally. Actually, more than occasionally. Okay. Let's just try it here. Let's try sigma two.

So sigma two is the minimum two a rank – this should be N minus one or something matrix, actually. I'm – is this right? What? What I have here is right?

Student:[Inaudible].

Instructor (Stephen Boyd):It is right? Okay. Sorry. My confusion. It happens. We talked about this before. Sigma three, you want to know what is sigma three, the interpretation is it's something like – if sigma three is small, it says you're very close to a rank two matrix. That came out right, didn't it? That's consistent with this. I wonder why I was confused. I was just temporarily confused. All right.

Very important one is if you have an N by N matrix, the minimum singular value is the distance to the nearest singular matrix. So that's what that is. So that's what it means. You can see, again, if you have a very small, singular value for a matrix that's invertible – a square matrix that's invertible, many ways to say it. One way is to understand and say, your inverse is huge. That's one way. Another way to understand it now is you are very close. That matrix may not be – it is singular. It's not singular. It's not singular, but it's very close to a singular matrix.

So if you have a minimum singular value of ten to the minus eight, it says that matrix is not singular. But I can perturb that matrix by another matrix, which is on the order of – it has a norm of no more than ten to the minus eight, and my new matrix will be singular. That's what it says. So that's the interpretation here.

Let's look at an application, and I'm going to do one more application after this to finish up. So the first – that one, you just modeled simplification. This is kind of obvious, but suppose you have something like one equals AX plus V, where A is in 100 by 30, and it has singular values of ten, seven, two, 0.5, 0.01, and they go all the way down here. That's sigma 30 right there, and it's 0.0001.

Now, norm X is on the order of one. Let's say this unknown noise here has norm on the order of 0.1. So that's my model. It's supposed to be rough, so don't worry about the exact numbers or anything like that.

This is actually quite interesting because if – well, we can do some quick analysis here. Geometrically, we can say the following. If norm X is on the order of one, and you don't know anything else about X, let's propagate X through A. Well, you can imagine a unit ball being propagated through A, and you're going to get an ellipsoid. In fact, a very good name of that ellipsoid would be the signal ellipsoid. In communications, that exactly what you'd call it. Is the signal ellipsoid. It shows you kind of what would be the possible received signals if the input varies over a unit ball.

By the way, this might come up through some transmitter power constraint or something like that. So this is quite close to a lot of real problems. So you get this ellipsoid, which is a signal ellipsoid. Now, that ellipsoid looks like this. It's got some big semi-axis, ten, seven, two, 0.5, 0.01. What's interesting about it is this. To that ellipsoid, that's the signal. But it basically says that then the signal is corrupted by a norm of 0.1. What you can see immediately is that basically, all of these signals are sort of lost. They're corrupted by the noise. They're bigger.

In other words, this thing, you can imagine as a ball of a radius 0.1. Then you have this ellipsoid, which actually has positive volume. It's got some really thin dimensions. It's got 26 thin dimensions. In fact, 26 of the dimensions are much thinner than this. What that really tells you is for all practical purposes, this A is rank four. This make sense. You can even talk about the signal to noise ratio. You can talk about the signal to noise ratio here. It's 1,000,

Down here, you're down to a signal-to-noise ration of about five to one. Here, it's the other way around. It's one to ten or minus 20 decibels. Whatever you want to call it over here and so on. So you basically say that for all practical purposes, this simplified model here is going to be absolutely equivalent. This might be depressing because this might have been a measurement system, and you might've though you had a 3.3 to one measurement to parameter you want to measure advantage. Turns out you're wrong. It's the other way around. You actually measured four things, even though you thought you measured 100. This make sense?

Don't read too much into it. This is sort of the typical application. Now, I want to do one more application that I realize this morning should've been in the notes. I don't know why it's not. I guess it will be next year, depending on if I remember to put it in. It's this. I want to go back to the beginning. Let's rewind back to the third week of the class. I'm going to ask you this. Suppose I had a bunch of vectors, A1 through A – let's just make it specific though. So it's A100, and these are in R10, okay? So I have 100 vectors in R10.

By the way, these could be snapshots of something I've measured. Who know what they are? It could be ten stock prices over 100 periods. It could be anything. It's just a block of data. Then I'm going to ask you the following question. I could've asked you this question week three. I could ask you the following. How would you determine if these 100 measurements, these 100 vectors in R10, actually live in a subspace of dimension three? First of all, let's find out what that means. Let me ask you this. Could any human being detect that?

Totally out of the questions. Absolutely out of the question. If I showed you ten numbers, unless it's stupid like, for example, the first number is always the same, like it's always one. But let's assume it's a generic case. I just show you – I say, look. Here's some prices. 100 long, 100 last trading days. Here it is. We need comments. No human being can look at it and go, whoa, those are in a three-dimensional subspace. Can't do it. Out of the question.

How would you do it, computationally?

Student:[Inaudible].

Instructor (Stephen Boyd):QR. Exactly. You simply stake this as a matrix, and you run QR. You know what'd happen? You'd pick a first one, second one, third one. You'd run the QR that doesn't dump core when the matrix is less than full rank. You'd run a modified QR. You'd run it, and I would run through 100 vectors, and it will have generated a rank only three – you know, Q1, Q2, Q3. By the way, the implication of this is very interesting. You would say, you know, that pi statement was really interesting. In terms of it varies in a three-dimensional subspace. By the way, what would that mean? What would be the interpretation? Suppose prices vary in a three-dimensional subspace? What does it mean?

Student:[Inaudible].

Instructor (Stephen Boyd):You can do what?

Student:[Inaudible]

Instructor (Stephen Boyd):There you go. Yeah. That's a perfectly good way to say it. It basically says that instead of just treating these as a bunch of random – it would say, this looks like random ten vectors. No human being can look at ten vectors and say something was fishy about them. What this would say is that this analysis would say these prices depend only on three underlying factors. It would allow you stunning abilities.

For example, given – I guess the right way to say it would be three – seven. If I gave you seven of the prices, you could predict the other three. Did that come out right? Yeah? Okay. That's what it is.

So you want to know what's the practical consequence of knowing this fact? It's amazing. It's basically, given seven prices, you can predict flawlessly the last three. I think that's right, isn't it? Other way around?

Student:[Inaudible].

Instructor (Stephen Boyd):Wow, that's funny. I had a double-expresso this morning. Wow. I'm back to three shots. That's it. Not all the cylinders are firing. Well, it's even more stunning, then. Given three, you can predict the rest of the other seven. But what I said, technically was true, just for the record.

Now, do you imagine that stock prices are exactly in a three-dimensional subspace? That was rhetorical. No, of course not. Do you imagine received signals in some multiple-antenna system are in three-dimensions? Of course not. But now, you now know how to detect if these vectors are almost in a subspace of rank three or if – well, let's see. The

range of these is definitely R10, but you can now detect whether the range of these is almost R3 or something like that. So somebody tell me how do you do it?

You just take the SVD of this matrix. You just take the SVD of this data matrix, and what will you see if you take the stupid data and you type SVD of A? What would tip you off that that data is almost in a three-dimensional subspace?

Student:[Inaudible].

Instructor (Stephen Boyd):You'd see three substantial singular values, and then you'd see a nice strong drop. Sigma four of the sigma ten would be small but positive. By the way, what that would mean is if you took these points and if you think of it geometrically, it says take these points and make a cloud in R10. I don't know how your R10 visualization skills are, but I'll tell you what mine are. Since I appear to not even be able to get – did you notice that both the mistakes I made today were of the same nature? It's an N minus I confusion? If it continues, I'll go to a neurologist, and they'll find out there's some lesion in some part of my brain.

Let's move on. Maybe that was the lesion talking. Let's move on here. If you were to visualize these in R10, if you could visualize vectors in R10, you could say, oh, let me see some more price data, and you could go, hang on just a minute, and you could go like this or whatever. And you'd be like, wow, that's in almost a three-dimensional subspace, but no one can do that. But if you could do that, you would see a cloud of points.

The cloud of points would actually – the SVD tells you something about this. It tells you that in some directions, the cloud would have a big extent. But in seven directions, it would be quite small. That's what you'd know. So that's how you'd be able to say that this thing is roughly in R3. By the way, how would you then actually calculate the so-called factor model in this case? How would you calculate an orthonormal basis of that three-dimensional subspace of the price subspace?

Student:[Inaudible].

Instructor (Stephen Boyd):First three Us. End of story. First three Us, U1, U2, U3, that's an ortho basis for that subspace. I guess it's not in the notes because it's basic and obvious, and this is just the definition of what SVD is, but still, this is the kind of thing you can do and will do using the singular value decomposition.

So that actually finishes our last real topic in the course, so that's it. The next topic we're going to look at has to do with this idea of controllability and solvability. It's interesting stuff. Actually, more than anything, what it shows is that once you understand everything up till today's lecture, everything else is nothing but a trivial application of it. So all other topics and stuff like that just follow from everything we've done up till now, period.

So essentially, the class is over now. You can treat the other topics as just interesting applications. Just cool stuff you can do with these ideas. Yes?

Student:[Inaudible].

Instructor (Stephen Boyd):I did.

Student:[Inaudible].

Instructor (Stephen Boyd):Are you asking what's the difference between the –

Student:[Inaudible].

Instructor (Stephen Boyd):Oh, what's the relationship? Yeah, I can say a little bit more about that. If these price vectors – let's make them price vectors. If these are price vectors or price change vectors or return vectors or whatever, if these are price change vectors like that, and they're not exactly in a three-dimensional subspace here, and you run the QR factorization on it, what do you get?

Student:[Inaudible].

Instructor (Stephen Boyd):Yeah, you get a Q that's N by N. The thing's full [inaudible]. You get a Q that looks like this, and you get and R, which looks like that. That's it. That's your QR factorization. It doesn't really tell you a whole lot in this case. Whereas the SVD, if you write out the full SVD, you get the same thing as this. The same shape, not the same thing. So the SVD is what would allow you, whereas the rank-revealing QR would not.

By the way, if sigma four gets small enough, it will trigger your rank-revealing algorithm to decide that it's rank three. That's a parameter in that method, and it has to be set right. It would normally be set way low. It would not be triggered by anything that came from data.

So that's the difference between a rank-revealing QR. And rank-revealing QR is something we looked at week three. So that wasn't yet quantitative. Week three, when someone said rank to you, it means what rank means. It has a mathematical definition. In that case, this would be rank ten. This real price would be rank ten. Now that you know about things like the singular value decomposition, rank is a fuzzier concept, although I should say something about that.

Rank by itself means rank. It's the mathematical definition of rank, and if anyone says, what's the rank of this data matrix, the answer's ten. If you're going to use a fuzzy definition of rank, you must qualify it. You must say something like – if someone says, what's the rank of that? You could say ten. You can say, but it's awfully close to rank three. It's well-explained by three factors, and it's almost perfectly explained by four factors, but you can't just say the rank of this is three. Everybody see what I'm saying?

Of course, once you're doing this kind of stuff and people on the streets, they say things like that all the time. Singular means singular. Non-singular means singular. Rank means

rank. Null space means null space. However, if you have a couple of singular values that are really tiny, you can call that the almost null space. If you were writing or there were lawyers present, I would not call it the null space. You can say it's the practical null space or for the purposes of this problem, for the purpose of this application, it's the null space or whatever.

But I'm wondering all around, and I'm not answering your question. The question was, what's the difference between A equals QR and Q equals U sigma V transpose? Was that the question?

Student:[Inaudible].

Instructor (Stephen Boyd):Yeah, roughly? Today? I'm going to take that as encouragement. Okay. I will tell you what the relations are. Let's do this case where it's full rank and fat or whatever. That's orthogonal and that's orthogonal. V transpose and R have the same size. That's pretty much it. You can say a few more things. There are some subtleties you can say about the size of the elements about the diagonal of R and sigma, but they're much more fancy and advanced. Basically, there's not much you can say about the two.

There are, but they're more advance topics. They're not simple things. Did I answer your question, which was not actually your question, but you were polite enough to suggest that it was close enough? Yes? Good.

Had that been your question, would I have answered it? Good.

Next topic, unless there's any other questions about SVD? Did you have a question?

Student:[Inaudible].

Instructor (Stephen Boyd):That's an orthonormal – the first three vectors in U is an orthonormal basis for the three-dimensional subspace that the As are approximately in. I think that came out right. Is that the question? You might ask, actually, then what the three Vs are. V1, V2, V3. That's interesting. I'll let you think about it. That's what it is. It's actually quite interesting in this context of, say, if these were price changes or return vectors, let's say. You might ask what V1, V2 and V3 mean. Now, if they're transpose, then they're like this. You would get something that looks like this and then something that looks like that. There's some sigmas in there, but you can shove the sigmas either way you like.

In fact, go ahead. Let's go ahead and answer it. What do you think these are? Let's say this is time, so the index represents time. It's closing-day returns. What do you think these three are? They're very interesting. Do you have a guess?

Student:[Inaudible].

Instructor (Stephen Boyd):Well, this basically says what's driving the returns on the prices. There's just three underlying factors. They're not totally random and unrelated. There's three underlying factors. This is the time history of those three segments of the economy. That's what it is. You can look at the first one, and you can say, oh, that's related to interest rates. I'm totally making this up, by the way. There's probably people in here who know much more about this than I do. I know enough to know that's what that is. So this actually gives you the time history. If this means time, this is history.

In signal processing, these would be the actual recovered signals, and this would be the subspace in which those signals lie. So that would be the example. But these are best understood by just doing problems. Yeah?

Student:[Inaudible].

Instructor (Stephen Boyd):No. It would not be. If you did the singular value analysis of this, and all ten were significant, sigma ten, for example, was not small enough that you could say anything about it, that means, basically, that this cloud of points again, if you were able to do R10 visualization, kind of extends about the same in all directions. That's kind of what it says. It would be like, no – by the way, this might be a good thing, it might be a bad thing. It's bad if your goals it predict the value of seven of these components when given three. That's bad, but for other applications, it might be good.

In that case, it actually depends – no, it wouldn't be the same. I'll just leave it at that. There'd be no connection between them, no. Yeah?

Student:[Inaudible].

Instructor (Stephen Boyd):These? No. These are the ten vectors, and that's an orthonormal basis. Basically, the assertion is this. Every one of these 100 vectors is very well approximated by some linear combination of these three. That's what it says.

Student:[Inaudible].

Instructor (Stephen Boyd):No. It transposes here. So this thing here is V1 transpose, V2 transpose, V3 transpose, like that. The Vs are 100-long vectors. This theta block is ten by 100, so the V is also ten by 100. So V transpose – oh, it came out wrong. V transpose is also ten by 100. Yes?

Student:[Inaudible].

Instructor (Stephen Boyd):Y, and it would be terrible. So if all the singular values of a matrix are the same – actually, we talked about that earlier today. That is a matrix with condition No. 1, but you can say a lot about that. It should be or was or will be a homework problem for you, I think, if we were on the ball. We may have accidentally not assigned that, but basically the homework problem – it may not even exist, but it would go like this. I'm losing it.

It would go like this. I'm making sense, condition No. 1, only if it's a multiple of an orthogonal matrix. Why is that? Because it looks like U sigma V transpose, but all the sigmas are the same. Therefore, sigma is a multiple of the identity, and there for it pops outside. It really look like lower-case sigma, UV transpose. That's a multiple of an orthogonal matrix. So that would be great. Matrixes like that just bring tears of joy to the eyes of people who do numerical analysis.

For signal processing, that's the best channel you could possibly have because you're not going to lose anything. You thought you had ten dimensions. You pay for ten antennas, ten receivers, they're all paying off. If that were data, and you were analyzing it, you would say, so much for that. If you're analyzing data, it just means, well, it's all over the place. It's worse. It's in a balanced way, it kind of goes all over the place. You haven't figured out any underlying structure in there. I don't know if this makes sense.

If A actually is a channel in a communication system, for example, and it has only three significant singular values, that's bad because basically it says that the gain in three directions is significant, but then after that – so, but you think, but I paid for ten transmitters and ten receivers. What this says is you paid for ten. For all practical purposes, you're getting three.

I think I'm answering questions all around yours. I'm sort of surrounding it. What do you think? I think – what was your question?

Student:[Inaudible].

Instructor (Stephen Boyd):You've forgotten now?

Student:[Inaudible].

Instructor (Stephen Boyd):Yes, that would be correct. If all the singular values of a matrix were the same, then I had a long incoherent discussion of what that means. It means it's a multiple of an orthogonal matrix, right? Then it says – this is not a good candidate for low-rank approximation because basically any approximation you make is going to be terrible. Why? Because the thing kind of pokes in all directions, so approximating it by something of lower rank is going to give you a huge 100 percent error, basically, is what it's going to give you. I think we're done.

So, let's look at the next topic, which, like I said, is an application. It's controllability and state transfer, so we'll look and see what the idea is. So we consider a linear dynamical system – the truth is, we've already done these problems. This just kind of formalizes a bunch of it. These could be homework problems, really. So let's take X dot equals AX plus BU for a Discrete time system over some time interval, TITF. That's initial and final.

Now, of course, these are integers in the case of a Discrete time system, and these are real numbers, and that's an actual real number interval in the case of a continuous time system. So you say that an input is a trajectory. It's a function that maps this interval, the

time interval into RN. You say that it steers or transfers the state from the initial state to the final state for that time interval. Then you have immediate questions like, for example, given an initial state, what can you transfer it to at the final time? What can you do? What are your degrees of freedom? If you think of U as a joystick or something like that or a possible input, the question is, what can you pull of over that time interval?

If you can transfer it to some target state, then the question is how quickly can you do it? Can you do it in three seconds? Can you do it in one? What's the minimum time it takes to transfer a given state to a target state. By the way, often, the target state would be zero. Remember, this is all – zero means some equilibrium position like some vehicle, an airplane flying at some trim position.

Another question would be how fast can you bring, after some disturbance like a wind gust – how fast can you bring it back to zero state, zero meaning it's back at the trim condition, for example. It can be an industrial process or something like that or network flows or an economy or something like that. It's all the same, in this class. If you're in those individual classes, it's not all the same.

Another question would be how do you find one that transfers you from some initial state to a final one. Another would be how do you find a smaller, efficient one that transfers you from one state to another. These are the types of questions – you can make up zillions more, but these are the types of questions that we can actually answer. In the Discrete time case, by the way, we can answer it all now, basically. You know all the answers. You don't need to know anything else.

Let's look at reachability. Reachability studies the question of taking an initial state of zero and going to some state, X of T. Then you say X of T is reachable. If it's a Discrete time system, you say in T seconds – sorry, it's continuous time, it's in T seconds. If it is Discrete time, or whatever the time units are, you'd say in T periods or T epochs, is what you would say. We'll let RT be the set of points reachable in T seconds or epochs.

That would be something like this for a continuous time system. It's the set of all N vectors that look like that. This thing here is X of T. When you start it, X equals zero. That's the formula for mapping an input over an interval, zero T, into the final state. You multiply by B, you put the exponential there, and you get that. This is a huge, infinite-dimensional set here. It's the set of all possible inputs that map zero T into RM.

Discrete time system, it's actually much simpler. It's the set of all things of this form where U of T is in RM. So the parameter that parameterizes this description is finite-dimensional. So here, I should say something like this, maybe T equals zero up to T minus one, I think. Is that right? Yeah. So that's what this would be.

It actually has dimension TM, this thing. Okay. These are subspaces, and it's easy to show directly. It's also easy to – basically, it's easy to show directly. It has an interesting implication to say it's a subspace. To say it's a subspace says the following. It says if you can reach a state in three seconds, let's say, and you can reach another state in three

seconds, it says you can reach, for example, the sum state. By the way, what input do you think you'd use?

The sum of the corresponding input. You'd have to check that and all that, but it would work out. That's how that works. The other thing is that if you can reach – you have this. RT is a subset of RS, if S is bigger. This says that the points you can reach in 3.6 seconds is a superset of the set of points you can reach in 1.5. I'll write those down, so I don't forget them.

Actually, let's argue that. Do you not believe it?

Student:[Inaudible].

Instructor (Stephen Boyd):No, it's true in Discrete time, too.

Student:[Inaudible].

Instructor (Stephen Boyd):No, that's my next question, but it's a great question. I'll summarize your question. It wasn't a question. It was a statement, like, that's completely wrong. That's maybe not – put much more politely, but that was the question. So let's figure this out. I claim if you can hit a state in 3.6 seconds, you can hit it in – no, sorry. It's one of those days. You have them.

Let's try it again. This is where I'd really like to be able to edit those videos. It'd buffer up real fast. It'd be about 12 minutes. Let's try it again. If you can reach a state in 1.5 seconds, you can reach it in 3.6. Your intuition was that it might not be the case. Someone, tell me how to do that. How do you do it?

Student:[Inaudible].

Instructor (Stephen Boyd):Okay. So first you reach 1.5, and then take your hands off the control?

Student:[Inaudible].

Instructor (Stephen Boyd):Oh, okay. I'm getting two different – you said you want to get there in 3.6 seconds. Get there in 1.5 and take your hands off the accelerator. U equals zero. You take that back?

Student:[Inaudible].

Instructor (Stephen Boyd):Okay. It's formally retracted. The reason that doesn't work is because you get there at 1.5, great. You're at the target state. Take your hands off the thing, and what's propagating you forward? E to the TA. So you're going to drift. 3.6 seconds, you will not be where you want to be. But there's a minor variation which I

heard in a kind of – actually, I heard ten answers coming at me, but they were kind of in a two-dimensional subspace.

Sorry. They were. Anyway, the correct way to do it is to do nothing for – do you think I can subtract these? 3.6. That's tough. You do nothing for 2.1 seconds. There you go. I got my stride back.

If you do nothing, normally you drift. But here, you started from zero. You get the distinction?

Student:[Inaudible].

Instructor (Stephen Boyd):You got it. It will not work. In fact, if you don't start from zero, you're absolutely right. If you can get from one place to another in 1.5 seconds, there's no reason to believe you can do it in a longer time period. But if you start from zero, that works. Everybody got that? That's it.

You say the reachable set is a set of the points that you can get to at any time. So if you can get there at all in any amount of time, that's called the reachable set. We can actually analyze this completely, about 15 seconds, for Discrete time linear dynamical systems. This uses all the ideas we know and love at this point. The Discrete time system, [inaudible] AX equals BU of T. Then I can write this. The state at time T, I will stack my inputs over zero up to T minus one, I'll stack them. So that's a big vector in RTM, like that.

I'll just write it out in a matrix, and that matrix looks like that. It's got a name. I guess it's called the controllability matrix. Some people call this the reachability matrix or something, but it's got names like that. The matrix looks like this. It's B, AB, up to AT minus one B. By the way, there should be absolutely no mystery as to what these are. Notice that I chose to stack the Us in reverse time. Why? I don't know. Just so this matrix would come with ascending powers of A, not descending powers of A.

So I've stacked my Us in reverse time. So if you do that, this says – how does UT minus one, which is actually the last input that has any affect of X of T, how does it affect X of T. The answer is just by B. The dynamics makes no difference. This says U of T minus two. That's your penultimate action. It's the action before the last one. It's UT minus two. That gets multiplied by AB. That should make perfect sense to you. B tells you the immediate action on X of T minus one A, then propagates that affect forward in time, one step. So that's what this matrix is. It's a very famous matrix.

It looks like this. By the way, the matrix comes up in other contexts. It's a huge topic in large-scale scientific computing and things like that, except they call it a Krylov matrix there. I'd mentioned this just because you'll see this matrix, actually, not with a vector – V will be a vector in that case. Okay. So in terms of that's real simple. We have a name for it. The reachability subspace at time, T, is actually the range of this matrix. The matrix grows, if I increase T, and now I can ask you any question at all. I didn't need any of this

to ask you, but now I can say, I have a Discrete time system. I start at zero. I have you a target state to hit, and I ask you, give me the minimum number of time steps it takes to hit that target.

One possible answer is you'll never hit it. We haven't quite gotten there. How do you solve that?

Student:[Inaudible].

Instructor (Stephen Boyd):Could've done that in week three. How'd you do it?

Student:[Inaudible].

Instructor (Stephen Boyd):Okay. So what you do is this. You take – first I ask you, can you hit the target vector in zero steps? Can you? There is one case you can do that. If the target's state is zero, then the person who asked you this, you go back and say, pardon, you're already there. It's done. Then they'd say, amazing. I'm glad you took that class.

Now, let's say T equals one. Can you hit it in one step? The only things you can get to in one step is of the form BU of zero. That's what X of one can be. So the answer is this. You can hit a target, X sub target can be hit in one step if and only if it's in the range of B. Notice that A plays no role whatsoever. Can you hit it in two steps?

You check if X target is in the range of B and then AB. If it's in the range, done. That's it. You can hit it. If it's not, you can not. So you simply keep increasing these until X target is in the range of them. Make sense? That's it. Everybody – and this is basic, so that's the idea.

That'll give you absolutely the minimum time required to hit a given target state. Now, by the Cayley-Hamilton Theorem, once you get to A to the N, A to the N is a linear combination of I, A, A squared, up to AN minus one. That means, basically, when you get A to the N, B here, you're adding M columns, but for sure they're linear combinations of the previous ones. Actually, Cayley-Hamilton Theorem now has a very interesting implication in a dynamic system.

It says the following. It says that after N steps, the range of RT is equal to RN. So it basically says this. In principle, the set of points you can hit grows with each T. By the way, it need not, but it grows with each T, but once you hit N, it will never get bigger. That's it. So when you take N here, that's called a controllability matrix, and people say that the system is controllable if the controllability matrix is rank N.

That basically says you can hit any point in N steps. That's what it says. Actually, it may not be a good idea to hit it in N steps. You may want to hit it in more, but at least in principle, you can hit any point in N steps. I think we'll stop there, which is a good idea, unless I can think of one last gaff I can get in. But I can't think of any, so we'll quit here.

[End of Audio]

Duration: 77 minutes