IntroToLinearDynamicalSystems-Lecture06

Instructor (Stephen Boyd): Are we on? We're on. Let me go down the pad here, and I'll make a couple of announcements. The first one, is that Homework 3 is now posted and Homework 2 is due this afternoon. I want to make one other comment, this is just in case I forget to announce this at the end of the lecture, you should read the notes on – there's several notes in the notes section on the course website. One is on notes on least squares, least norm, and solving linear equations. I think that's two different sets of notes. We won't have covered all of the material, we will not have covered all of the material by today, but there's no reason that you can't read these. These are each two pages or something like that, and you need to read these.

I'm trying to think if there's any other announcements. I should probably remind you, I'm sure you know, that the midterm is coming up actually in two weeks. And, yes, I'm as shocked as you are at this. We've scheduled a day, or days, there are two days. You can take it on either Friday to Saturday or Saturday to Sunday. If that's not gonna work for your schedule, like you're in Antarctica or something like that on those days, you need to contact us, and we'll work something out. Generally, it means you take it early, and that makes you a beta tester. Whatever you do, don’t just not show up those day, because we'll come looking for you, and we know where you live or we can find out where you live.

Okay. Any questions about last time? If not, we'll start in on – actually, the first part of the course, I think I hinted at this last time, which is actually quite useful. Everything else, so far, has just been sort of background, really leading up to this and its infrastructure. We're actually now at the payoff point.

So we're gonna talk about least squares. It's something you've probably seen in a couple of different contexts, and it concerns overdetermined linear equations. So we have a set of over determined linear equations. Now, here we have y=ax, where a is we'll make strictly skinny. It's overdetermined because you have more equations than unknowns. And, of course, unless y is in the range of a, which if you pick y randomly, and rm is an event of probability zero, you can't solve y=ax. So one method to approximately solve y=ax, and it's very important to emphasize here we're not actually solving y=ax, is to choose x to minimize the norm of this residual. So ax - y, by the way, some people call the residual y – ax, it doesn't make any difference because mostly you measure yourself by the norm, and the norm is the same. So the residual is ax - y. It's the amount by which ax misses y. If you take the norm of that, that's the distance between the point ax and the point y. If you minimize that, that's called the least square's approximate solution. Now, here, I'm going to insist on using approximate. I hate to tell you that, on the streets, you would call this the least squares solution of y=ax. This drives me insane. Because whenever somebody says the blah, blah, blah solution of y=ax, I presume, in particular, x solves y=ax. And it's very important to understand in this case, it does not solve it, that's the whole point, it's an approximate solution. Of course, it can solve it, that's possible. And that, indeed, would be the least squares solution, because residuals don't get any smaller than zero, norms of residuals. So I'll call it least squares approximate solution, but you will here people, if you go to Wikipedia, or whatever, and type it in, you'll hear people talk about the least squares solution of y=ax, which is not a solution of y=ax. So the geometric interpretation is something like this. This is very childish, that I can only draw this in r2. In this case a , of course, is 2 by 1, it's kind of silly. The range of a is a line, so here's the range of a, and here's a point not on the line. And so the question is to find the point closest to this target point y that's on the line. And there's a name for that. When you find a point in a set closest to the point in the set, closest to a given point, that's called the projection of the original point on the set. So here ax, ls is the projection of y on the range of a. And you'll see all sorts of notation for that, but maybe the most standard would be this. So ax, ls would be a projection of this point on that set. So that would be probably, if you looked at ten books, you'd probably see this in eight. So that's the geometric interpretation. Let's work out what it is. It's not too complicated to actually just work out a formula for it. So a is full rank and skinny, that's our assumption here, and what we need to do is minimize the norm of the residual. Well, that's the same as minimizing the square of the norm of the residual. The square of the norm of the residual is r transpose r, and you simply plug in what that is and you get the following function here. Now, that's a constant, that's the norm of y squared, that's a linear function of x, and that's a quadratic function of x, a pure quadratic function of x. The whole thing all together is just a quadratic function of x. This is actually called the quadratic form, the pure one. How do you minimize this? Well, we'll set the gradient with respect to x to be zero. And what you'll find is the gradient of this – is that on the current homework? No, it's on the one we just assigned. Of course, you can work out the gradient yourself by taking the partial derivative of this with respect to xi. But what you'll get is the gradient of this function is 2 a transpose ax, it should look very, very much like the formula in the scaler case, minus 2 a transpose y, and that's equal to zero at a stationary point, in this case it's the minimum. This only has one solution. There's only possible solution of this equation because a transpose a is invertible. It's invertible because a is full rank and skinny. That's one of these basic properties. So if I reduce the two and put it on one side, these are very famous equations, they're called the normal equations. You'll see why in a minute. Actually, the picture before explained that. Since a transpose a is invertible you have this: You have x least squares is a transpose a inverse a transpose y. So I don't how many ASCII characters that is, let's say it's ten or something like that, watch out because the whole point of overloading notation is that a very small number of characters can pack a lot of meaning. So this looks very simple, looks very innocent, it means a lot. By the way, it's also unbelievably useful. So it's a very famous formula, you probably shouldn't forget it. You probably won't forget it because if you look for it you'll find it in lots of other contexts in classes and things like that. So here it is, it's a transpose a inverse a transpose y. You can do all sorts of checks here just to scan this and make sure it's okay. You can check things like, well, if you're gonna invert something it should be square. And, indeed, a transpose a is square. That's fine. You should check, in fact, that this is full rank. For example, what can you tell me about that? Why not? You're shaking your head.

Student:

[Inaudible.]

Instructor (Stephen Boyd):Exactly. This is square but not full rank. So that's a semantic error not a syntax error. How about this one? That's a syntax error. Right. That was vigorous and violent headshaking, which is the correct reaction to this.

So there's the formula for it, it's quite simple. Actually, one very interesting thing here is this is actually a matrix. Let's call it bls times y; it's linear. So calculating this approximate solution is actually a linear operation on y. So that's the first thing you note.

Now, if a happens to be square, then it's invertible, a transpose a inverse is a inverse times a transpose inverse. By the way there's some slang for a transpose inverse, it's not really slang, people write this as a – 1 and then a - t, that's standard. That's what this is. And then if I multiply on the left by a transpose, let me do that, a transpose, I get this, and you get a inverse. So you can see that when x is square, this is just a pedantic and silly complicated formula for the inverse again. So you can think of this as, actually, a generalization of the inverse, and you'll want to think of this as a generalization of the inverse of a. Now, watch out, because in this formula a is not square. But if a were square, this would reduce to the inverse of a. Of course, this calculation here is completely wrong when a is not square. In fact, you have syntax errors already here, right here.

Now, suppose if y is in the range of a, that means there is an x that solves ax=y. In this case, it turns out x least squares is it. How do you know that? It's actually by the definition, x least squares minimizes the norm of ax - y over all possible x. But if there exists an x for which ax=y, then this thing can be as small as zero. You're not ever gonna do any better than that, so that x has got to be x least squares.

That's a good property of an approximate inverse to have, or a method that gives an approximate solution, is when the equations actually have a solution, it will return the solution.

Now, there's a name for this matrix, it comes up all the time, and it's called a dagger, that's a very common, that's universal notation. So it's a with a superscript dagger. And that's called the pseudo-inverse of a, and it's got other names, too. If you're British you'll here it called the Moore-Penrose inverse, and there's probably some other names for it, too. In fact, some people, who like long names, call it the Moore-Penrose pseudo-inverse. And there's other names, but they don't matter.

Here's what's interesting about it, it is a left inverse of a full rank skinny matrix. So it's this, if you work out a dagger a, that's a transpose a inverse times a transpose, that's a dagger times a. And I re-associate this this way, and I see a transpose a inverse times a transpose a, that's the identity. So you have the following: It is a left inverse of a. By the way, the dimensions of this pseudo-inverse here are the transpose of the dimensions of a. So if a is skinny, it is in this context, I shouldn't say if a is skinny, in this context a is skinny, and a dagger, or a or pseudo-inverse, is gonna be fat here, and it's a left inverse. By the way, I could have said this early on. One of the statements on this linear algebra review was that, if a is skinny and full rank, then it has a left inverse. And I could just have said, here it is, a transpose a inverse a transpose, and you could have just plugged in the formula. But, of course, you'd be scratching your head thinking where on earth did that come from. So that's why I deferred it. So you now know, here's a very explicit way to construct a left inverse of a matrix, it's just the pseudo-inverse. It is not the only left inverse, we'll find others, and, in fact, we'll find all left inverses soon. Now, if you want to look at projection on range of a, if you multiply a by xls, by definition it's the point in the range of a that's closest to y, and it's the projection of y onto the range of a, that's denoted this way, and that's ax, ls. And the projection function is actually linear, and it's given by this matrix here, it just multiplies by that matrix. And that is an m by y m matrix; that's what it does. It's very interesting matrix. By the way, it's pretty, it's got a nice a symmetry, you've got some a's and some a transpose appearing, and an a transpose a inverse, and all that sort of stuff. You're actually gonna see a lot of formulas like this; you're gonna have to get used to checking these very carefully to make sure that they're, first, past syntax, and then whether they're up to the semantics as well. And if you look at it, it has the look of a matrix that sort of would want to be the identity. It is not the identity, unless, by the way, a is squared. In which case, when all the smoke clears, this thing just goes away and it's just the identity. But it's not the identity, it's projection, like that. By the way, this already has lots of uses. It basically says, you can actually imagine something like this, you can say you receive a signal with noise, a vector signal, a bunch of, say, multiple antennae communication system, but you happen to know that the signal should lie in a certain subspace to the range of a. You can simply then project onto the range of a, and what you've got now is the closest signal to the one you received, which is sort of consistent, that's in this known range, and this is used.

Student:

[Inaudible]. I don't get why a transpose a is full rank.

Instructor (Stephen Boyd):Oh, why is a transpose a. So that goes back to my slide, which said that if a is full rank and skinny then a transpose a is invertible. So I'm referring back there. I'm not showing it. Actually, you could now, with the qr factorization, we could actually establish that connection.

Student: Excuse me, question.

Instructor (Stephen Boyd):So this is a matrix you will also see quiet often. It's a, a transpose a inverse a transpose, it's actually a projection onto the range of a, that's what this matrix is.

Student: Excuse me.

Instructor (Stephen Boyd):Let's look at a couple of things here. There's a so-called Orthogonality Principle, and that says this: That the residual –

Student: Question. Hello.

Instructor (Stephen Boyd):Yes.

Student: All these thing that you said is –

Instructor (Stephen Boyd):This is so cool.

Student: Sorry?

Instructor (Stephen Boyd):This stuff hasn't been used since 1969. Oh, my God.

Student: No, actually, the next class here is using it so often.

Instructor (Stephen Boyd):Really?

Student: Yeah.

Instructor (Stephen Boyd):You mean more recently than 1969?

Student: Yes.

Instructor (Stephen Boyd):Cool.

Student: Much more recently, like last Tuesday.

Instructor (Stephen Boyd):Really?

Student: Yeah, really.

Instructor (Stephen Boyd):That's so cool.

Student: Yes, it is.

Instructor (Stephen Boyd):Okay. I didn't know where it was coming from. Anyway. You have a question I bet.

Student: Yeah, I still do. So all these things that we are saying is about a skinny full rank matrix, right?

Instructor (Stephen Boyd):Right.

Student: But what if we have a, like, sensory and it's not full rank? That's really bugging me. If we have a matrix and it's not full rank what should we do?

Instructor (Stephen Boyd):It should bug you. We're gonna get to it. So our discussion right now, here's our discussion, is skinny full-rank matrices. Why? Because I said so, for now. I promise you we're gonna talk first from here, will talk within next week, we'll be talking about fat full-rank matrices, then a week or two later we will talk about what happens when you have non full-rank matrices. But I promise you we're gonna get back to this. By the way, just to warn you, a lot of times when I say that we're gonna get back to it later, of course we don't. But in this case we really are.

Student:I'll remember that.

Instructor (Stephen Boyd):Okay. Good, you should. And it's a good question. And, by the way, the answer's it's a bit complicated, and that's why we're doing the simple thing first.

So the Orthogonality Principle says that the optimal residual, that's ax, ls - y is this thing, it's a transpose a inverse a transpose - i times y, is actually orthogonal to the range of a. It's right here. That if this is the range of a, and here is ax, ls, and here's y, the residual, like this, is orthogonal to this. Now, to really make this more interesting, you might imagine at least r3, and in r3 the range of a you could make a plane. And then you have a point, and then you have ax, ls; ax, ls is the projection of that point on the plane, and if you go from that point, the point on the plane, up to the original point, you're gonna get something that's normal to that plane. So that's the picture. And it's quick arithmetic to just calculate r transpose az, and you'll see that it all just works out here. I'll let you do it because this is just arithmetic to work out.

Now, by the way, there's also a way to understand this. In this case, suppose someone said, "Well, that usually happens. Sure that happens in the university, but when you get out in real life, that's not right, that doesn't really happen." And you can have things, "Oh, we found things where the angle can be 92 degrees." Something like this. There's actually a pretty simple way to know that this can't happen. And I can turn this into a formal mathematical argument, but I don't want to. If someone produced a point that was alleged to be the projection of y on a subspace, and this residual weren't normal to the subspace, how would you prove him or her wrong quickly? I just want a quick geometric argument, which, by the way, would easily be turned into a mathematical argument. What would you do?

Student:

[Inaudible.]

Instructor (Stephen Boyd):I can see your fingers are – I'm going to interpret your hand motions. What you do is you take this point and simply move it in the direction where your residual's leaning. If you move a small distance in that direction your angle goes up and your distance to that point goes down, and you've just proved the other person a liar. Because you've just moved closer, and you're still in the range. The original point was alleged to be the closest point and you just moved closer than the point they had, so it's impossible.

This is a very good [inaudible]; for example, Fourier series and probably other things as well. Okay. Let's make a connection with QR factorization. If a, is again, full rank and skinny, we can factor it as a=QR. Now, in this case, here R is square because it's full rank and skinny. So we write it, a, which looks like this, as Q, and then R, where this upper triangular, like so. So that's the picture. And the columns of Q are orthonormal. Well, the pseudo-inverse is just this, it's a transpose a inverse, and I'll just plug in for a, QR. So this becomes a transpose is of course R transpose Q transpose, that's here, QR, and now you have to be very, very careful with these things. So when the columns of Q are orthonormal, then Q transpose Q becomes i. So this just goes away then, and you have R transpose R inverse. Now, R is upper triangular and invertible, because its elements on the diagonal are positive, so it's invertible. In fact, its determinant is the product of the elements on the diagonal, which are positive, so it's invertible. So R transpose R is therefore invertible. In fact, the determinant of R transpose R is the determinant of R squared, because the determinant of R transpose is the determinant of R. So this is invertible, and of course it becomes R inverse R - t, that's slang for R inverse transpose, which is the same as R transpose inverse, these two operators commute, times R transpose Q transpose, that goes away, and in the end you get R inverse, Q transpose. So a very simple formula in terms of the QR factorization for the least squares approximate solution of y=ax. And it's kind of weird. And let me give you kind of a complete hand waving, and utterly wrong argument, but kind of just as a rough idea, it works okay. You say, well, you have a=QR. If a were square, and Q were square, and R were square, then you would have a inverse is of course R inverse, Q inverse. Ah, but Q is, in this case, orthogonal, so I can replace Q inverse with Q transpose. This is if a is square. But now I look at this for a minute, and I say, "Hey, wait a minute, what if a is not square?" Of course a doesn't have an inverse, so I want some kind of approximate inverse, and I just use the exact same formula, which is R inverse, Q transpose. So roughly speaking, the transpose of a matrix with orthogonal columns is something like an inverse. I can tell you're not buying it. But anyway. Well, it was just I just threw that out there in case that made any sense to you. But I guess it didn't. Now, projection on the range of a, that's a, a transpose a inverse, but we just worked out what all this stuff is, this is just that. So if I multiply a, which is QR, the R's go away and you've got Q, Q transpose, and now you know what Q, Q transpose is, it's actually projection on the range of a. Now you know what it is. So let me remind you, that last lecture we talked about Q skinny matrix, it actually has to be, because I'm gonna assert that the columns are orthonormal. So the columns are orthonormal, that's Q. And in this case, that is exactly the same as saying Q, Q transpose = i. And let's make it M high, and My, it doesn't matter, something like that. And I mentioned at the time that Q, Q transpose is not i. And I said I'd tell you later what it is. And this is one of those times where I wasn't lying then, because I really am telling you what it is later, which is now. Q, Q transpose is not the identity, it's actually the projection, it projects onto the range of these columns, that's what it is. And it's sort of like an inverse. Projection sort of gets you as close as you can in the subspace. By the way, if you're in the subspace it's the same as the identity. So now you know what happens when you take, for example, three orthonormal columns in R10, and form, call that Q, form Q, Q transpose. You get a 10 by 10 matrix, and what it is, is it has rank three, and what it does is it projects onto the range of those three columns. That's what it does, it takes an argument x, and it returns a y, which is the closest point in the range of those three columns, that's what it is. I should mention one thing here, least squares is not – actually, you can, and I would even encourage you to do that, in math lab, although, I mean, this is not supposed to be a class about math lab, but I'll say just a little bit about that. You are more than welcome to type something like this, in fact, I would encourage you do this, this times a prime times, let's say, y. You're more than welcome to type that in, and you will get something up to numerical precision, which is basically a transpose a inverse a transpose y. Now, in fact, generally, this is not how this is calculated. It's actually calculated using the QR factorization. And I'm not talking about math lab, I'm talking about how people do numerical calculations, which is not what this class is about, but I thought I'd mention it. So you have to distinguish between the formula and then actually how it's calculated. In fact, it's computed, typically, using the QR factorization. So what would really happen is you would actually do this, you'd form Q transpose and then R inverse here. R inverse because it's upper triangular and it's very easy to carry out. I should mention one thing. This is so common, that in math lab this is just this, it's backslash. And that, I believe, actually, is a math works innovation. It might be just about the only one, but there it is, as far as I know. They were the one to popularize it basically to say, a transpose a inverse a transpose is so important it should have one ASCII character assigned to it, that's how important it is. And I'm pretty sure that was, in fact, these guys. You could write this, but I guess that would be at least two if you type it in in text. So that's it. Now, I also have to warn you about this, this does other stuff. So you asked what happens when a is fat, when it's skinny, when it's skinny but not full ranked; a backslash y will do something. So be very careful, that's a symbol that is overloaded, and it does not just work out this. It will not warn you or you'll get very strange warnings, and things like that. And it could very happily return an answer, and it won't be what you think it is. So I just wanted to make mention of this. There's some notes on the website about this and you have to read those. So let's look at full QR factorization. So here, you simply extend out the columns, Q, you extend out, you complete an orthonormal basis, and you append on the bottom of R a bunch of zeros. So this now has the same size as a, and this matrix Q1, Q2, although it looks fat, because I've separated the columns into two parts, is actually square, this is actually a square matrix, and it's orthogonal; it's columns are orthonormal. Now, if I take ax - y norm squared, this is a, that's the full QR factorization x - y, and what I'm gonna do is take this expression here, that's a vector in RM, and I'm gonna multiply in the left by the M by M matrix Q1, Q2, that's orthogonal. Now, when you multiply by an orthogonal matrix, you do not change the norm. So I can shove a Q1, Q2 transpose on the left here, I can shove that into both sides, it doesn't affect the norm. So over here I get this. This is the identity, because Q1, Q2 is orthogonal, so you get this. So this is the identity, and I get R1x - Q1 transpose y, and the second block entry is zero x, it's minus Q2 transpose y. So this thing is equal to that norm squared of that vector. But the norm squared of a vector, if you've blocked it out, is the sum of the norm squares of the blocks. That's what it means. So this is equal to this. By the way, we haven't solved the problem yet, this is simply a formula for norm ax - y norm squared. That's what it is. And so the question is, if you look at this, now we can ask the question, how should you choose x to make this as small as possible? Well, if you look at the right-hand side, it has nothing to do with x; there's nothing you can do about it. So the right-hand side is irrelevant if you're minimizing. Now, you want to choose x to minimize this. Well, let's see, how small could this be? Well, it's a norm squared, it can't be any smaller than zero, but it can be made zero, and it can be made zero by taking x=R1 inverse Q1 transpose y; that's our same old formula before. So that's the picture. And the residual, with the optimal x, is Q2, Q2 transpose, which you can work out directly from this because that's the residual. So what that means is that Q1, Q1 transpose gives you projection on the range of a, and Q2, Q2 gives you projection of the orthogonal compliment of the range of a. This stuff is all very abstract for now, but when we put in the context of actual applications, I think it'll become clear, and possibly even interesting or something like that. So that's the connection to full QR factorization. Now, we can talk about applications. So, basically, we already had the conversation about y=ax. Frankly, it's a short conversation, we already had it, and it's good you have it. So the short conversation goes something like this: If you have inversion problem, you have y=ax, then if the columns of a are independent, which is to say that a is full rank, I'm assuming it's skinny here, then it says, yes, in that case you can reconstruct x exactly from y. What would do it? Well, a left inverse would do it, but you already know a left inverse now, which is, in fact, this a dagger, or a transpose a inverse a transpose. So you know a left inverse, that's this. Now, generally speaking, there's gonna be some, in all cases where you actually make measurements, there's gonna be some small noise. Now, v might be very, very small, v could, in fact, be the error in simply calculating a times x with double precision numbers. So v could be as small as round off error in a numerical calculation. But generally speaking, this actually refers to some inversion estimation or reconstruction problem, it involves received signals or things like that, and v is simply – well, if these a sensors, these are sensor noises, errors, things like that, they could be errors because a sensor is only 14 bits. All that is lumped into this y. So the real picture is y=ax + v. Now, you want to reconstruct x given y. If there were no v, you can reconstruct it exactly if a is full rank and skinny. But, in fact, now there's an error, and, in fact, now I'll tell you the sad news, you cannot reconstruct x exactly if there's noise. That's kind of obvious, right? So you have to make some prior assumptions about what v is, and then actually you're not gonna simply get an x that – well, you don't know what v is. If you knew what v is you could subtract it from y, and you're back in the other case again. So if you don't know what v is, then you have no choice but to do something like best effort here. And the best effort is gonna be, well, we'll see, is least squares estimation. So least squares estimation says this: You choose x hat, that's your guess as to what x is, by choosing x hat to minimize norm of ax - y. And ax has a specific meaning. It is what you would observe if x were really x hat and there was no noise; y is what you really observed. So you're basically getting the best match between what you would observe if the input were x hat and what you did observe. So that's the picture. So you minimize the mismatch between what you actually observed and what you predicted – oh, by the way, in a lot of fields a times x is sometimes referred to as the forward model, that's a good term. So it's the forward model, and basically, it's something that maps x into ideally what sensors you would see if the sensors were flawless and perfect. That's y=ax. Some people call that the forward model. By the way, it's a big deal. Somebody could work a couple months to come up with some code that computes the forward model. It simply maps x's into what you would observe. And it's a big task, depending on the particular problem. So it would be highly non-trivial, for example, to work out the forward model if the problem is a mechanical structure, and the x's are loads that you put on it, and the y is a vector of 1,000 displacements. It would involve all sorts of finding an element, modeling, you'd have to read in some standard description of the structure and all this sort of stuff. But for us it's just ax, and that's called a forward model. So what this is doing is, it says you get something which actually doesn't match anything in the forward model. And what you're doing is you're minimizing the mismatch between something that the forward model can give you and what you actually observed. So that's the picture. Now, what is it? It's nothing more than this. It says, you take a transpose a inverse a transpose, and that gives you your least squares estimate. Very simple. You'll see in a minute this works extremely well, does very, very clever stuff, all rather automatically. So let me mention here, the BLUE property. You also see it with BLU. It's the Best Linear Unbiased Estimator. So let me explain that. B are measurements of noise, so y=ax + b; a is full rank and skinny. And let's suppose you're gonna look at a linear estimator. A linear estimator takes y, multiplies it by, in this case, a fat matrix B, and returns x hat. That's a linear estimator. Well, you just multiply this out, this is x hat=B times ax + v here. And it's called unbiased if, without noise, it's infallible, it always reconstructs x. So an estimator that works perfectly in the absence of noise is called unbiased. Now, what that means is this: If v is zero, it says x hat is Bax, and the only way for x hat to equal x always is for Ba to be i. Well, that means it's a left inverse. So, by the way, when someone says left inverse to me, I think only of an unbiased estimator, that's what it is, it is nothing more. They are identical. If you see Ba=i, in that case a has to be skinny, if you see Ba=i, it means if a is sort of a measurement system, B is a perfect equalizer or a perfect reconstructor, that's what it is, or an unbiased estimator. Now, if you have an unbiased estimator then your error, that's x - x hat, that's the error in your estimate, is x - Bax + B. But Ba is i, so this x - x goes away and it's just this. So what happens, if you have an unbiased linear estimator, is that your error is now completely independent of what x is, and it depends only on B. And, in fact, it's minus B multiplied by this error. Now, what you want now is the following: Now you can see that not all left inverses are the same. Right. Because, in fact, when one person has one left inverse, and another has another, they both give you unbiased estimators. And, by the way, if there's no noise they will both reconstruct x perfectly. Now, you introduce noise. When you put in noise, the two estimators differ if the two left inverses are different. And they differ because the estimation error is actually that matrix multiplied by v. So now you can see what you'd like, you really want a small left inverse. Now, obviously, B can't be [inaudible] zero because then you could not possibly have Ba=i. So now you can see, if you want a small left inverse, and by the way, in this context small means it's error is not sensitive to the noise. Well, it turns out our friend, a transpose a inverse a transpose is the smallest left inverse of a, in the following sense. If you take simply the entries of a matrix, and think of them as a vector, then its norm squared would be sort of the sum of the squares of the entries. And so it says that our friend, the pseudo-inverse, is the smallest in the sense of sum of squares left inverse of a. And this is not too hard to show, but I'm not gonna do it. And so people say, then, that least squares provides the best linear unbiased estimator. I should mention one other thing. So far everything we're gonna talk about, when we talk about estimation we don't do it in a statistical framework because, for some reason, long ago, it was determined that probability was not a prerequisite for this class. I'm not sure why, but that's the way it is, and it seems okay. I should mention that, of course, estimation, obviously, should be looked at from a statistical point of view. In which case, you know, after weeks and weeks of discussion of probability and so on, in statistics you'll come up with things like least squares. Someone will say least squares, and you'll say, "What's that? Oh, I'm really doing maximum likelihood estimation with a Gaussian noise model," for example, that's what least squares would be. But that's always the things are, something that's a good idea can usually be very well justified by several completely different methods. So you'll also see this in statistics, but with completely different discussion all around it. And it's not bad to know that ideas like least squares, although they are deeply statistical, you can also understand them completely, and be absolutely effective in using them without knowing statistics. That was not a recommendation to not learn statistics, by the way. So let's look at a simple example, and let's just see what this does. So this is navigation from range measurements and it's distant beacon, so we're gonna use the linearized measurements. Now, what that means is, each measurement is an inner product along this normal vector, pointing from the unknown position to the beacon. So this guy simply measures not circles, but in fact planes, like this, lines, I guess in R2, that's what it measures, you get this distance. Now, the linearization is quiet good. And so what you're gonna get is this, you're gonna get four measurements of x, linear measurements. So that's a 4 by 2 matrix, and there'll be a noise here. Now, I chose a specific noise here, and the noise I chose from a random distribution. Again, this is totally irrelevant, but let's just give a rough idea of what it is. The noises are independent Gaussian; they have a standard deviation of 2. This is totally unimportant. What it means is that these v's, the entries, are off typically by ± 2. They can easily be off by ± 4, but it's very unlikely they're off by ± 6, and essentially impossible that they're off by ± 8 or something like that. Just to give you a rough idea, that's noise. And if you like to think of that, what it really means is that each of these measurements is something like a ± 4 meter accuracy measurement; if you want to get a rough idea of what these are. So what you'd say here is that what you're gonna get is you're gonna get four ranges, measurements of the range, but each range has an error of about ± 3, ± 4 meters, just roughly, that's the idea. So, for example, if the real range is 10,000-blah, blah, blah, you could easily have 10,000 + 2 or 10,000 - 3. Given those four noise corrupted measurements, what you want to do is estimate the position. Now, here you've got four measurements, you have two unknowns to do. So the quick arithmetic tells you right away, just a quick accounting, says you've got about a 2 to 1 measurement to parameter redundancy ratio. That's good. It's better than the other way around, which is a 1 to 2. So you have 2 to 1 measurement redundancy, just roughly. Now, the actual position happens to be this, we just generated it randomly, there it is. And the measurement you get is this. Now, by the way, there is no x that actually will give you, if you plug in two numbers here, multiply it by these four vectors, you will not get these numbers. These are inconsistent measurements. Why? Because there's noise. In fact, we can probably get a rough idea of how inconsistent they are. They're probably inconsistent in each entry by about two or three or something like that because that's how the noise was generated. Now, we're gonna do two things. The first one is this, we're gonna take someone who took a linear algebra class. They didn't talk about non square matrices, they only got to inverses and things like that, but they didn't talk about left inverses or least squares. And so they say, "Look, I don't know what you're talking about here, I never learned about tall matrices. We did square matrices. And we learned that if you have two things you want to find, two measurements does the trick, and you get a 2 by 2 matrix, and I even know the formula for the 2 by 2 inverse or something like that." So one method to do here is to simply ignore the third and fourth range estimates. So this is the just-enough-measurements method. You've got two things to measure, you take two measurements. And without noise, that certainly is gonna do the trick, assuming your two measurements are independent. Well, what that corresponds to is taking the top half of this matrix and inverting it, that's how you get x. And that would actually exactly reconstruct x if there's no noise. So, in fact, this method of sort of inverting the top half, and padding it out with zeros, this creates a left inverse you can check. Well, I didn't give you a. But had I given you a, you could multiply this thing by a and you would get the identity, it's a left inverse. And a left inverse means this, it will reconstruct perfectly if there's no noise. That's what this does. By the way, when you see a matrix like that, let's call that B just enough measurements, I should maybe call that B just enough measurements, like that. If we write this out, if you see this 2 by 4 matrix, and I say this matrix maps range measurements into position estimates, when you see this, and you see these zeros, you should immediately come up with an explanation. And in this case, what is the meaning of these four zeros?

Student:

[Inaudible.]

Instructor (Stephen Boyd):It means the last two measurements are irrelevant. This zero is sort of just an accident, and it just means to the precision I'm printing these, which is not exactly high precision, it says that, basically, our x1 hat doesn't depend on y1. Did I say that correctly? I think I did. And, by the way, you can probably look back at the geometry and figure out why that's the case.

So here the norm of the error is 3.07. We should give the rms error, so you should divide by square root 2. So 3 divided by 1.4, I don't know, what's that? So you get an error of about 2 in each – I'm getting the rms error here. So you get an error of about 2. And, by the way, if someone says, "Hey, you're off by two." They'd say, "Pardon me, your range measurements are off by about two." Does that sound reasonable? That would be the response if you use this method. They're off by about two in both x and y, but the measurements were off by about two, so who can blame the person.

Let's look at the least squares method. When you work out a transpose a inverse a transpose, you get this matrix here. And you see a lot of things about it. Now, let's talk about some of the things you see about it. Oh, by the way, the estimate it produces a 4.95 and 10.26, the norm of the error is .72, you divide by 1.4, and you have a norm of error, which is about .5. So the first thing you know, is at least for this one random sample I generated, this thing did a lot better, it did a lot better. So that's the first thing you notice.

Now, this is another left inverse. And, in fact, it's interesting to compare the two left inverses. Let's compare these two. So let's just look at these two left inverses. This one we already said this estimator doesn't use the second two measurements, it doesn't use the third and fourth range measurement. What you see here is if all these numbers are kind of the same – they're all there, they're smaller than these, you don't get numbers as high one and minus one, but you're using everything. By the way, there's a name for this, it's called sensor blending, that's what's actually happening, or fusion, that's a beautiful name. People talk about sensor fusion. And so if someone said what have you done here, and you don't want to say, "Oh, it's just least squares." If you want to impress the person, you say, "I've implemented sophisticated sensor fusion algorithm." So that would be the correct way to say this. And they'd say, "What does that mean?" And you'd go, "Well, as you can see, I'm blending all four measurements to get each estimate in just the right proportions." And you'd say, "Well, how did you get these?" And you'd go, "It was tough. It was really tough." But the idea remains, this is one of the ideas you see in – you laugh, but there's whole fields where people don't even have a clue about these ideas, where you get a lot of data and you want to make some estimate of something, and there's totally straightforward ways to do it, like this, but that's simple. There are three ASCII characters. This is like a backslash y or something like that. But the point is, it's actually doing something quiet sophisticated, it's kind of blending all these things the right way. So that's the first thing you say. I want to ask you a question, though. Obviously, for this one instance, this estimate was a whole lot better than this one. But I have a question for you, could you please tell me a situation in which this estimator will outperform this one. And, in fact, let's make it simple. I want it to outperform it by a lot. Having just made fun of this estimator, I would like you to explain a situation in which this estimator outperforms this one. What is it?

Student: Do you just randomly [inaudible] noise [inaudible] sensor and [inaudible]?

Instructor (Stephen Boyd):There you go, perfect description, I'll repeat it. It goes like this. Suppose, I mean, the one thing you can say about this one compared to this one is it doesn't use the last two measurements here. So if those last two measurements were way off, in fact, suppose that you were being spoofed, for example, or suppose the range sensors got totally whacky, dropped a bit somewhere and they were way off, way off. This one is completely insensitive to the errors in the last two measurements. This one, however, if you're way off, and v3 or v4 is really big, you're in big trouble because of this. So it's not that simple to say that this is always a better estimator than this one. And, indeed, there are cases where it might not be. But the point is, generally speaking, if what you can say about the v's is that they're all small or all about on the same order of magnitude, this is very likely to be a lot better than that. So that's the picture. Now, we can go back and we can actually do our example from the first lecture, which is this: You had an input that was piecewise constant for a period of a second or a nanosecond or microsecond, if you like to make a different time scale, but it doesn't matter. It goes into a filter, and that simply means that this signal here is a convolution of that input here, with a convolution, kernel or impulse response. Just sample that at 10 Hz, that means you sample it every 100 milliseconds, you get this. And, then, now comes the part that's tricky, we actually subject that to three-bit quantization. So the quantizer characteristic is this, and this basically selects the nearest of eight levels spread over the interval minus 1, 1. So that's what this formula does. And so here's the input signal that we don't know. It might have been, for example, a coded communication signal being sent to us. Here's the step response. This convolved with the derivative of this, that's the impulse response give you this. And you can see that it's kind of smooth, but it kind of tracks it. Here you can see sort of where this dips down, you see a dip here. It's delayed on the order of one second. And you can see sort of where this goes up, there's a bump. It smoothed out, though, like that, so it's delayed a half a second or something like that, and smooth. And then this is what it looks like when you severely quantize it, three bits. And the idea is, given this, estimate that. Now, by the way, looking at it by your eye, I think the answer would be forget it, it's not gonna work. It's already hard enough here, because this thing is this thing kind of all smeared around. In other words, this point here is produced not just by what's in this interval, but what's here, here, here, and so on. So you just say forget it, it's all mixed together, can't do it. And then when you quantize it, it gets even sillier. It seems sillier. So we'll just write it as y=ax + v, where a is an R 100 by 10, and aij is this, and you can just work out the details. This is how you map x into y. Then we'll take v here, in our model y=ax + v, is gonna be the quantization error. It's the difference between y tildy and the quantized version. And, by the way, it's actually very complicated, it's a deterministic function of y tildy, but I can tell you one thing about it, all of its entries are between ± .125. Because the width of an lsv here is .25, a least significant bit. And so the error's no more than .125. And the least squares estimate is this. And you can see that it pretty much nails it, or at least it gets very close. In fact, oddly, it gets twice as accurate as the quality of your a to d. I tried to shock you with this the first day, and I remember you were like, that's cool. So you should be impressed by it. So that's the idea. By the way, this example is a perfect example of how all of this actually is used and how it goes down in practice. No one will bump into you later tonight and say, "Quick, I need help." "What is it?" "I've got this matrix a, and I've got this vector y, and I need to find x so that norm ax minus y is minimized, can you help me?" No one is gonna do that. What will happen is, you will go back to some lab or something like that or go off at some company, and someone will give you, except it won't be clean like I made it, they'll throw some huge thing down, and if it's communications there'll be acronyms flying all over the place, and they'll say, "Yeah, were using four quam, and this," and it'll go on, and on, and on. "Oh, and you should use this such and such model, and there's this." And there'll be some finite elements, and PDE's thrown in. Horrible thing. Nowhere will they say least squares, they will certainly not say linear algebra, they probably don't even know linear algebra, they won't say it. They'll have all sorts of things, they'll talk about who knows what. There'll be spherical harmonics, and cosigns, and signs, and different coordinate systems. And your job will be to just sit there, very calmly, read the pile of stuff, go read a couple of books on the field, and sit there, and realize I will write all of this as y=ax + v. And, then, you will go to your computer and you will type a backslash y, very calmly. So the last part will be very fast. But the first part is all the work. So I'm just saying this example is a good one, this is exactly how it happens. There's block diagrams, and all sorts of transfer functions, and all sorts of nonsense all around. And it's your job to get rid of all the field specific details and abstract it away in something that fits this way. So that's it. Actually, I want to show you one thing here. In this case, I just selected a couple of rows of a transpose a inverse a transpose. Now, what this does is this thing maps y, that is this smeared quantized received signal. This maps the smeared, quantized received signal into our estimate for these ten real numbers, the ten real numbers in the original signal, x1 through x10. That is a 10 by 100 matrix. It maps quantized, smeared received signals into estimate of x1 through x 10. I have plotted here Row 2, Row 5 and Row 8; these have a meaning. Row 2 of that matrix actually shows you how those 100 received signals get multiplied to create your estimate of x2. So when you take an inner product of your 100 received signals, with this vector, that's forming this unusually good estimate of what x2 is. Look at it and you'll see some really cool stuff. If you look at this, this says that to get the estimate of x2 here – in fact, let me ask you. It says that the quantized signal from six seconds to ten is pretty much not used. Why am I saying that? Because these entries of b are small. Do you see what I'm saying? That makes perfect sense, if you think about it. Because you know that the smearing is kind of local in time. It says that to get a good estimate of x2, it says the most important thing is actually y right at time 2. By the way, x2, for the record, is actually not there, x2, this value, sort of the middle, is 1.5 seconds. So this thing was smart enough to realize, very roughly, that you should sample one-half second late. Why? Because that's roughly what that smearing did, it added a one-half second delay. So that's what this shows. This little negative bit here, that's cool, that's actually what, in communications, is called equalization. And it basically says, you should sort of subtract from what you see here, this thing, because this thing was kind of corrupting your measurement. And so if you subtract for it you're actually kind of compensating for it. This is equalization. By the way, this is the third part of how this goes down in practice. We already talked about the first part. The first part is decoding all these books and the silly field specific notation, which finishes when you put it in the form something like y=ax + v. The second part is a backslash y, that goes right away. Now, the third part is to go back and explain your estimate to people who have never heard of pseudo-inverses. If you say it's just least squares, and they're, ""Okay, that's fine." But sometimes they'll just look at you and they'll say, "What's that?" It just means you have to explain it. And then you'd plot these pictures, and you'd give this story until they go away. And you'd say, "This is how I estimate the second one. This is the how I estimate Row –" truly, what's really estimating, of course, is a transpose a inverse a transpose. But at least this make the story clear. By they way, you could even do things like now truncate this, and they'll have sort of a finite equalizer if they were uncomfortable about your estimating using stuff from the future and stuff like that. So that's the idea. Any questions about this? And you're still not impressed with this, I can tell. We'll look at some more applications. And the first one is data fitting. It's very famous. You're given some functions on a set s, s could be anything, it actually could be finite, it could be r, it could be an interval, it could multi-dimensional, like it could be r3 into r, and these are called regressors or basis functions. It's called basis functions even if they're not independent, but we'll leave that alone. So you have a bunch of functions here from s into r. And you have some measurements. Now, a measurement looks like this, by the way, some people call them samples. So the other one is you have measurements or samples. And a sample is a pair si and gi. So that's a sample. And so, for example, let's make this specific, let's let s be r2, let's make it simple. In fact, let's make s zero 1 cross zero 1, that's the unit square, it's something that looks like that. And so one of these functions, a basis function here, is a function from this unit square to r. And so, for example, I'll tell you f1 is just the constant 1, f2 is a, let's put a physical interpretation here, that's the temperature at a position on a square plate, that's what it is, that's the meaning of this. So each of these basis functions is now a temperature distribution. So the first one is just flat, uniform distribution 1. The next one is like, let's say, an x gradient. That means it's something where as you move across here, the temperature grows linearly; f3 is gonna be a horizontal gradient; f4 is some weird thing, it depends on the field. In fact, these things come from the field. So you go to the person, let's say it's Bob, who's been doing this for a long time. And Bob says, "No, you should use the spherical la Gare harmonics of the 12th order. We've been doing this for years, trust me." You're like, "Okay, whatever." So that's usually where these come from or they might come from some other place. For example, you might say, "Oh, no, no, it's heat equation, that's Poisson equation – "you know what, these should be the Igen function of the heat kernel, and so these should be the cosigns and signs, and this is actually a Fourier series. So it just depends. So you're given measurements. That means that you sample the temperature at some of these points. And what you want to do is you're gonna form a temperature distribution, which is a linear combination of these temperatures in such a way that it matches approximately your measurements. Now, here we'll talk about the other regime, but in this case we're gonna assume you have a lot of measurements and not that many bases. So I have six bases, in fact that's kind of the point of having a well chosen basis, is that if you have a well chosen basis, you can actually describe all the temperature distributions that you're likely to actually see in practice quite well as a linear combination of let's say eight basis elements. Now, you sample the measurement a 1,000 places. Well, you've only got six numbers to choose, and you have a 1,000 measurements. So a least squares fit does this, it says let's choose these coefficients, the x1 through xn. By the way, in this context these are often called, not that it makes any difference, something like a1 through an, because – well, I'm using x because it's gonna end up as x when we write norm y – ax, but, in fact, normally it would be called alpha. By the way, to make it really complicated for you, these things are actually very often indexed not simply from 1 to n, but by multiple complex indices. So, for example, if you're doing image processing, these might be the such and such wavelets or the edgelet, these might be the edgelet series, and they might have like three indices. This is just to mess you up. But we just enumerate them as 1 through n. And your job is to choose those coefficients like this to make this thing small, to get a good fit. Once again, I'm not gonna do it, but you look at this, and you just take a deep breath. And, remember, they won't do this, they'll index these horribly. They'll give you 40 papers on edgelets and the wavelet transforms. They'll say, "Oh, you don't know about wavelets? Wow, where were you?" And they'll throw a bunch of papers at you and walk away. And so these x's will be called alphas, and they'll have, let's say, have two indices in the superscript and two as subscripts, and it'll be hideous. You're job is to just convert this very calmly to this, to just ax - g. And aij here is nothing but fj of si. So that's it. And you just work that out because you just realize that this is axi, and that means that that's actually giving the ith row, and so, in fact, it's nothing but this. So it's extremely straightforward, it's just this thing. So the least squares fit here is a transpose a inverse a transpose g, you just work it out. And the corresponding fit is this. So this would give you your fit. And this has a lot of uses, a lot of uses already. And let me just say what some of them are. But mostly what it is is it's a very good way to develop a simple approximate model of data. And let me make up some examples and you'll get the idea. When you have a simple model, you can do things like interpolation. You can say I'm measuring the temperature on this core at 75 places, but I'm really interested in what the temperature is, very likely at these 22 other places, I couldn't get a sensor in there. What is it? You study the thermodynamics, and somebody tells you, "'Well, actually, here's a good basis, the prolate spherical something harmonics." And you go, "Whatever." So you get 30 of these or 20 of these or whatever the appropriate number is, and you now fit the data you have to those. Now, you have a model, that's your least squares fit. You can now make a guess as to what the temperature would be anywhere on that plate. Are you right? Who knows. It depends on whether, in fact, the real temperature distribution is in the range of the basis. It probably isn't. So as to whether you get a good estimate, a good interpolation or extrapolation, for that matter, here it depends on the quality of the basis chosen. But the point is, it give you a very good method to do it. By the way, these methods will work very well if they're done responsibly. You can also do smoothing of data. So that's one of the other things that happens, if if the data has lots of noise, but you have a lot of it, and you fit it, you're actually gonna get a nice smooth thing that will actually – so you can use it as a method to de-noise measurements that way. You can even go back, and, actually, if you think of, let's say, gi as an extremely noisy measurement of what some underlying f of si should be, you can actually first do the fit, then come back and look at fls fit of si. And you say, you know what this is? I believe that's really what this is, this is just the horrible noisy measurement. And now you've got a beautiful way to de-noise the data. I'm just making up applications because there's lots of them. I'll give you one more. So here's a real application. Actually, all of the ones I talked about were real, but here's one. You're developing an arrow model of some vehicle. Finally, you do all the computational fluid dynamics, all this stuff, finally you take the thing down to NASA, or NASA Aims, and you stick it in a wind tunnel. This is very, very expensive. You fire the whole thing up, and you take a bunch of data, and you tilt it up, and you do all sorts of things, and you change the speed of the thing, and you take thousands of measurements of the moment and the force put on that structure, on that vehicle, as a function of, for example, it's angle of attack, it's various other angles, so you've got three or four dimensions there, and the speed of the fluid it's moving in. Everybody see what I'm saying? It's unbelievably expensive. You take a million measurements because you just take it continuously, and so on, that's what you do. Now, you want to come back and you want to just make a simple simulator for this vehicle. Well, you need, at some point – basically, it looks like this, it looks like x [inaudible] f of x, there's some other stuff in there, but this is this function you just took samples of. So what you do now is this, you don't have f, what you have is one million samples of f from your wind tunnel tests. Here's what you do, you do have a very good idea of what f should look because otherwise you shouldn't have put the thing in the wind tunnel, so you clearly had some idea, and everyone knows roughly what this looks like. I mean, the first derivatives, you know what those are, those have well known names and arrow or the stability derivatives or whatever they call it, you know roughly what those are. So, actually, you can figure out a very good basis for these. You know, for example, about the phenomenon of stall, you know that if the angle of attack gets too high you're lift is gonna precipitously drop. So you know all these things because you've done those. So you can come up with a very appropriate basis. You now take that one million data points, and you fit an approximately f. By the way, that wouldn't take so long, if you're curious how long it takes to do a least squares with a million variables, it doesn't take that long. But in any case, let's say it takes 10 seconds or it doesn't really matter what it takes. You do that, you know what you have now, you now have a very small piece of code, like one screen of c or something like that, that will evaluate an approximation of f in well under a microsecond. So you just took 10 million pieces of data, and you scrunched it down, you fit some data, and you now have an approximation, it's an approximation of f but a good one, and it's just 50 lines of c. You can evaluate it way under a microsecond. You know what that means? It means you can simulate this thing like crazy now. Because in a simulator you're gonna call that every time step, and you're gonna call it tens of thousands of times. I get the feeling that no one has any idea what I'm talking about. Did any of that make any sense? A few people. So that's the basic idea. And we'll look at a quick example, and then we'll quit for today. So least squares polynomial fitting, this is very famous. You want to fit a polynomial of degree less than n. So your coefficients are a zero up to an - 1. The basis functions are, of course, the powers of t. By the way, that's a very poor choice of basis function, unless the data happens to be between, for example, remarkably the t's happen to be between minus 1 and 1. Even then it's a poor choice. And this matrix a has this form, which is a Vandermonde matrix. So the rows are increasing powers. And I won't go through this, but we'll just look at a quick example. Here's a function, which is by no means a polynomial, I just made it up, it's some rational function. And we take a 100 points and we'll do least squares fit for degrees one, two, three, and four, and we'll just take a look at what happens. Now, a degree 1 polynomial, that's a line. This thing has almost zero slope, but not quite, and this is the fit to this function of degree 1. And you can see it's trading off the underestimate here with the overestimate here. Because it's a line, it can't do very much. Here's the quadratic fit. And you can see it got a bunch of the curvature here a bit, the dashed one is the curvature, so it kind of got some curvature. Here's the cubic fit. And notice you're already getting a pretty good estimate here. The cubic allowed you to put this little tail over here, and here's your quartic. And these are all just done least squares, something like that. So we'll quit here.

[End of Audio]

Duration: 78 minutes