ConvexOptimizationI-Lecture06

Instructor (Stephen Boyd):Great. Okay. Can you turn off all amplification in here? That'd be great. So let me start with a couple of announcements. The first is – let's see. Oh, Homework 3 is posted. I know you'll be very happy about that. Actually, Homework 3 start early. Do not start next Tuesday. Just a hint. Start early. In Homework 3, there's numerical stuff, so although it should fit you. You'll see the X. Although, you're absolutely welcome to go the Python route. Actually, anybody here gonna go the Python route? There was at least one person. Typically, that would correlate with people who wake up at let's say 1:00 p.m., so it's not surprising that they're not here. They are in the class but sleeping right now. So you will use CBX. It should install in about 25 seconds or something like that. But just in case it does not, that's why I recommend you start immediately. And the other thing is like anything else, you just wanna start early and get all the stupid questions out of the way and all that early. So please start Homework 3 early.

The other thing is that the Monday section, Jacob will give, and it's gonna be on using CBX, so you're strongly urged to go to that for this. And of course, we'll post both the video and we'll also post the slides for Monday's section. Let's see. I wanna make one other very general announcement about the homework is we had a lot of questions about what do we mean sort of in the homework, like when it says which of the following functions is convex. And we got people asking things like do I have to have a full proof and all that sort of stuff. So I thought I'd say something about that. The main thing actually is that you should sort of understand it, and work through everything, and try to figure out which things are quasi-convex or quasi-concave or whatever they happen to be. Some of the problems we asked of course, these are not by any means easy, so if you have a headache trying to figure out if the quartile as a function of the probability distribution is whatever it is – if you have a headache from that? That's normal. That's as it should be.

Most of the applications won't involve stuff that tricky, but some actually will, so that's kind of the idea behind that. As for how much detail you have to give to justify this, that's kind of up to you. You're welcome – we don't need to see some long proof or something like that, but actually some justification would be good unless you have developed some kind – well, I guess you all know about the concept of like an idiot savant. These are people who sit in institutions, drool most of the time, and then about once a week look up and spit out a 32-digit prime number. Everyone knows about these people. Apparently not. So I haven't encountered it yet, but if in fact – maybe not such a severe case, but if it turns out you can look at a function, just think about it, make no calculations, just look at it, smile and say quasi-concave, that's fine. Actually, if that's the case – if you find that you have that ability, I'd like to speak to you. Barring that however, the important part is that you should have some kind of a justification. If you happen to have had a class on analysis, or if your background is strong in math, feel free to give us a nice proof. If it's not, don't worry. We just want some kind of a justification. You wanna look at the sublevel sets or something like that. So any question about last time? If not, we'll continue right along. You should be reading of course Chapter 4 by

now, and in fact should probably be finished by around today or something like that. So between the homework and the lectures, you should be okay.

If you can go down to the pad, last time we looked at linear fractional program. That's a very famous – and maybe the most famous quasi-convex problem. There are lots of variations on it, and a variation – it's a generalized linear fractional problem is this. You minimize a linear fractional function here. Now always in a linear fractional function, you have to decide which sign the denominator has, and by convention this would be positive. That technically has to be specified. You have to specify the domain. So here's a linear fractional function. You minimize a linear fractional function subject to linear inequalities and linear equality constraints. Now that's a quasi-convex optimization problem, and you can solve it by bisection. That's easy to see because if you wanna know is F0 of X less than T, that's the question. So you're really asking is the optimal value of this problem less than equal or T. Can it even be achieved, the objective value T?

All you do is you take this thing which is positive, and you multiply it out, and it turns out that's equivalent to the linear inequality like this. Now that's a linear inequality, and therefore you'd call it an LP feasibility problem or something like that. To check if the optimal value – if there's an X that satisfies all this and has F0 of X less than T, to solve that – to determine that feasibility problem, you would solve an LP feasibility problem. And then you could bisect on T and solve this. Okay? So that's one way to solve this. It turns out though, some quasi-convex problems can actually be transformed to convex ones. It hardly matters frankly, but in this case it's good to see how this works. And let me point something out. If you were to take this approach, you'd have to solve an LP for each bisection step. Now the number of bisection steps you'd probably have to solve would be somewhere between 10 and 20. 20 would give you two to the minus 20 in bisection, and you would reduce your ignorance – that is the difference between a known feasible point and a known bound – you'd reduce that by a factor of a million. So you'd have to solve by 20 LPs.

By the way, for most things it wouldn't make any difference at all. And it's even more than that. Later in the class, you'll understand that you can actually do something called warm start, which means you actually solve one LP, which is very close to the LP you just solved, and it can be done very fast, so the effort is not even a factor of 20. It's probably a factor of four. Nevertheless, it's interesting I think theoretically, and it's actually interesting in practice that this quasi-convex problem can be turned into actually just an LP. Yeah?

Student: That second line, should that C transpose X plus be on our right side, should that be a [inaudible]?

Instructor (Stephen Boyd):Should it be C transpose – in fact, what on earth should it be? Should it be E transpose like that? Are you happy now? Good. You're happy because what I had was wrong, and this is right. That's good. Thank you. It was supposed to be the denominator. It just didn't work out that way. So let's see how to do this. It's very interesting, actually. It turns out this is nothing but the perspective transformation. Or

some people call this homogenization, so let's see how that works. What you do is you write down this problem. And let's actually see what happens here. What you do is you write down something – well, here. It's equivalent to this LP, and let me go through the argument, or at least I'll give you some of the argument. The full argument you can find in the book.

Here what you do is you introduce a new – I start with the original problem. Let me just start with the original problem up here, and I'm gonna introduce a new variable called Z scalar. I hope this'll work. So I wanna minimize C transpose X plus D divided by – this time I'm gonna get it right – this. I wanna minimize that. Note the period. Everything's cool here. Subject to – that's st, but it means subject to of course – H and AX equals B. Now you know when you do things like you have a quadratic function and you add a new variable that's equal to one to convert it to a quadratic form, I guess that general principle is called homogenization. You make the problem homogeneous by introducing a new variable. So the way we're gonna do that is this. I'm gonna introduce a new variable called Z, and I'm gonna say it's one. So I haven't hurt anybody, and I can most certainly do this. I guess the Z should've gone in front of the H, but you know it's okay. In loose parsing, you can put a scalar after a vector. Everybody cool on this? I mean how could you not be? I just multiplied a bunch of things by one.

However, if you stare at this problem long enough, without the Z equals one you realize the following: it's completely homogeneous. In other words, scaling X and Z up here by a positive number has no effect whatsoever on the inequalities, none, has no effect on the objective. That means in fact I can normalize – my current normalization just equals one, but I can normalize it any way I want and simply divide out later. It will make no difference. So I'm gonna renormalize this problem. And I'm gonna renormalize it – now I am cheating. I'm looking down here. I'm gonna normalize it so that this – I'm gonna comment this out, and I'm gonna add this constraint in. That's just a different normalization of the same thing. It's just fine. Now – what's that? Z has to positive. That's correct. Sorry. Thank you. Actually, technically Z has to be strictly positive, so in fact I'll even write it that way like that. Now if this is one, the denominator goes away. That goes away. And I am left with an LP here that I have to solve. There's a question.

Student: Sorry. What's Y?

Instructor (Stephen Boyd):What is Y? It's nothing.

Student:[Inaudible] even shows X?

Instructor (Stephen Boyd):No. It's just nothing. It's supposed to be X. It's just one of those days, I think. I have a feeling where yeah, fine, good – keep me on my toes here, not apparently that it's gonna do any good today, but we'll see what happens. I should've had that second espresso. I think that was the problem. I didn't. I meant to, but I didn't. All right. This is now an LP here. And in fact, if you replace this with this, you get all sorts of information out of this. I won't go into the details, but for example if you were to solve this and Z comes out positive, then by simply renormalizing here, you get this

solution. That's the first part. If Z turns out to be zero, that's like this one case where this thing is unbounded below or something like that, but that's the idea. By the way, this is not a simple trick, but it's actually one that comes up a couple of times, and it's pretty cool that you can do this. By the way, for most quasi-convex problems this trick won't work, but for some it does.

Now we get to the idea of a generalized linear fractional problem, and that's this. You wanna minimize the maximum of a bunch of linear fractional functions. And that's on the polyhedron where all the linear – well, it's at the intersection of the domains of the linear fractional functions. That's quasi-convex. There's no way to convert that to an LP in general, so that you're gonna solve by bisection. And here's an example that's actually quite interesting. It's the von Neumann growth model, and it works like this. We're gonna optimize over X and X plus the following. So X is gonna be a set of activity levels. I guess we have to have X positive here too, or something like that. X is a bunch of activity levels at one period, and X plus are the activity levels in an economy at the next period, so that's what these are. Then we have matrices AX and AX produces – this gives you the amount of – AX tells you the amounts actually produced by an activity level X, so this could be for example in our M, if the N activity levels produce say M different classes of goods. So if you have an activity level X, AX is a vector telling you how much goods – the vector of goods produced by that activity level here. That's AX. Now when you operate the economy at the next step, that's X plus – are the activity levels. BX plus – that's a vector telling you how much is consumed. And then you simply say that the amount consumed at the next stage here, it uses some – it has to use the goods produced in the first stage, so you'd have an inequality that looks like that. Okay?

These describe – it's a polyhedron, of course. Right? The set of activity levels current and next that satisfy these inequalities is a polyhedron. And now what we want to do is we wanna maximize over these the minimum activity growth level. That's XI plus over XI, so that's what we want. I guess we don't need this because when you write this thing, X positive is implicit here. To maximize this is the same as minimizing the negative of this, which is minimizing the max of XI over XI plus, each of those, so that's the same thing. And that's exactly of this form. That's a quasi-convex optimization problem, and you can solve it directly, so that's one. What you wanna do is you wanna set the activities in order to maximize the growth rate of the slowest growing sector, so that's the problem here. And there's lots of variations on this.

So that ends up sort of problems that are linear and related to linear. Now we move on to quadratic. By the way, I can tell you what the year is on some of these. The linear programs as I mentioned earlier go back to Fourier at least, but the year maybe generalized linear fractional stuff is maybe the '50s or something like that, already widely used in the '50s. Quadratic programming is the same. This would be mid-'50s. So in a quadratic program, you have a convex quadratic objective. That's here. So P is positive semi-definite, and you minimize this over a polyhedron, so a set of linear inequalities and a set of linear equalities. By the way, if you just said P equals zero, you recover an LP, so this is a strict extension of LP. And the picture now is this. Here's your polyhedron which is your feasible set shaded, and your objective function now instead of being affine or

linear is in fact quadratic and convex quadratic. So the sublevel sets are ellipsoids, and so this is showing you the sublevel sets of a quadratic function. Of course, the gradient points down the slope, and is the normal to the tangent hyperplane at the level surface. So for this particular problem instance, that's the solution there. One thing you see immediately that you didn't have for an LP – it's a basic fact, although not of particular importance for us, that in a linear program you can always take a solution that's at a vertex. This is very basic. It's kind of obvious geometrically. It's also completely obvious analytically, but it doesn't seem actually to have too much use for us and in most cases, although it's made a big deal of I think mostly for historical reasons.

But you can see clearly here that the solution does not have to be at a vertex. It can be inside a face. Again, it doesn't make any difference to us or whatever, but that's – so this is a quadratic program. We're about mid-'50s here in where we are historically. So let's look at some examples. The most basic one would be just least squares. There's no constraints. That's basic linear algebra. The solution is just given by – or sorry, a solution I should say in the general case is given by this pseudo-inverse or Moore-Penrose inverse. That's just A dagger B. But immediately for example, you can take a least squares problem, and a very common thing – I mean this is really dumb, but you just add some linear inequalities. In fact, [inaudible] balance. So that's a quadratic program. By the way, what you're going to see later in the class is that for example a least squares problem with bounds on the Xs, that can be solved basically as fast and as reliably as just a least squares problem with a very small modest factor, totally reliable.

Let me actually say something sort of culturally about how the course goes. There's a lot of people that know about least squares, okay? A lot. I guess not a huge number, but you know there's a lot of fields where least squares is widely – a lot of people know about least squares in signal processing, statistics, communications, machine – it goes on and on. A lot of people know about that, okay? You just throw in something as dumb as this, ranges on the Xs, and that number of people who recognize that as basically a problem that is completely solved is – just technology. We can just solve that, period – drops by at least a factor of ten. What you will find instead – do people have things like this? Sure, they do. But of course, what they do is they do heuristics. They either invent their own algorithm that's impossibly long, might actually work, but it's fantastically complicated, or they make some iterative thing with regularization. You wouldn't believe what happens when people encounter things like this. And people make a big deal out of it. There's no reason for it. So I think the zeroth order thing to do is to look at a problem like that and say QP, therefore we're done. It's just done. There's almost nothing interesting to say about it. You'll write code that solves this. It'll be 30 lines. It'll be totally reliable. So it's not a big deal, but at the same time it has huge practical uses.

So if you wanna do parameter estimation but you have bounds on the Xs – here's a stupid one. Suppose you're estimating powers or intensities. Well, these are generally speaking non-negative, so that's just like non-negative least squares. So for us, that's just a QP. I'm telling you now, that's already – you've moved into a much smaller group than the people who know about least squares. For example, a whole huge thing in statistics – this is just amazing actually to me, but whatever. So that's the constraint that the Xs be

ordered. So of course that's a set of N minus one linear inequalities. I think you had it on some homework, like Homework 1 or something. I don't know. Write us a set of linear inequalities. So believe it or not, there's a whole subfield called isotonic regression or something. There's books written on this, literally on minimizing – on doing least squares with that. Why is there a book on it? Because there's no analytical solution like there is for just ordinary least squares. It comes up enough anyway. For us, monotonic regression? That's below the level I would even assign as a homework problem. No, I would go that low. But it would be one of a – when we're counting up the amount you have to do in a week. That would count for like none, very little anyway, and whole books on it.

So I'm just mentioning these things are simple, but actually so far this has not diffused out to a lot of application areas because people haven't gotten the word yet, which is very strange to me, that you can do this and it is completely straightforward. Okay. We're back on track here. Here's an interesting one is linear program with random cost. So that's another – let's do a specific example. If you wanna make a specific example, let's do diet – the infamous diet problem. So you are – X represents the amounts of a bunch of foodstuffs you're gonna buy, something like that. And there are constraints on it. There are things that have to balance out exactly, and there's inequalities. Of course, this can include lower bounds and upper bounds on for example nutrient levels and things like that. So our constraints become a polyhedron in X here. Now what we're going to – the problem now is we're gonna formulate this diet now, but we're gonna implement it next month, and we don't know the prices of the foodstuffs, so it's not as simple as saying just minimize C transpose X which would give you the cheapest diet that would meet the requirements because you don't know C. You should know something about C. C for example has a mean, C bar, but it also has a covariance sigma, something like that. This doesn't matter here. There are many things you could do. In practice, I'll tell you what would probably be done would be that. You would simply ignore the variation and just work with the expected values.

Now it's not hard to imagine that that could be a seriously bad choice because it's very easy to do. All you do is take one foodstuff, or one component, or whatever and you make it slightly cheaper than another one that has similar nutrients, but you make it have much more price volatility than the other one. Everybody see what I'm saying? The LP will – I mean if you ignore the risk – this is price risk – if you ignore the price risk, the LP of course will use the cheap stuff based on this expected value. So that's an extreme example, but that's the kind of thing you could do. So what you really wanna do is trade off mean price and price risk. This would be done – you'll see we'll do more of this later – by simply minimizing a weighted sum of the expected cost and the variance here. Gamma is a risk aversion parameter, and there's lots of ways to figure out what it is. For that matter, you would probably solve this problem for a whole family of gammas and get a tradeoff of expected price versus the variance in the price. And then you select based on your risk aversion where you should operate on that curve. So that's the point. Okay. But if you work this, take a look at it, it's nothing but a – this is a QP. There's one condition. Gamma has to be positive here. Otherwise, this is no longer convex. So what's interesting about that is that you can do risk averse design by convex QP, but you can't do risk

seeking. If this were negative, it would be risk seeking. You'd actually prefer if there's two diets that have about the same mean cost, meet all the constraints, in risk seeking you'd actually prefer the one that has higher variance in cost. Fortunately, that's not something you wanna do. So it turns out once again what you wanna do lines up with what you can do.

I should make one other comment here, and that's this that in general – by the way, not entirely always – if P is not positive semi-definite, so if it's got just literally one negative eigenvalue or something like that, this problem goes from being extremely straightforward to NP-hard. So if you know what I'm talking about, that's fine. If you don't, I will just say this. It goes from a problem that is in theories – actually multiple theories would assert that that problem is now very hard to solve, and also in practice. You'd go from a problem where you can solve the exact solution if X can be huge like 10,000 by – that's just no problem. The 30-line code you'll write to solve this for example will easily solve that. If this becomes indefinite, oh, you can solve these problems, but for example with X in R20, you'd have to have a lot of luck and a big old cluster, and you might get the answer later that day. You're just in a completely different regime. If X is in R100 when P is indefinite, this problem essentially becomes impossible. So convexity makes all the difference with quadratic programs. Okay.

Now another generalization – I mean it's kinda silly but – is a QCQP, that's a quadratically constrained quadratic program. So here the constraints can also be convex quadratics. Now here, if these are zero, you recover a QP. And of course, if all the Ps are zero, you recover a general LP. Here, these things – if P is positive definite, of course this represents an ellipsoid, so here the feasible set is an intersection of – well, ellipsoids and degenerate ellipsoids. Now, a degenerate ellipsoid is one that can have a flat face and be infinite. So a half space is actually degenerate ellipsoid, for example. But for example, if P is positive semi-definite but only has Rank 3, this is sort of a degenerate ellipsoid. It means that in N minus three dimensions, it's sort of infinite. In three dimensions, it's got curvature. For example, it would be a cylinder in R2 for example – R3, sorry. A cylinder in R3 would be a degenerate ellipsoid. That would be degenerated by a Rank 2 – a three by three matrix P that's Rank 2, and this inequality would then be not a cylinder, but it would be an ellipsoid like this, and then it would have an infinite extent in another direction. Okay? So that's it. It's a convex problem and can do this.

Now we get to the first thing. Again, this is maybe – we're still in the mid-'50s. Now we're jumping ahead. Now we get to – well, it depends how you wanna count the history, but you're into the '90s now, so this is new roughly. Although, actually there's papers written in the '50s that refer to this. They knew how to do it when there was one inequality like this, but not in general. So let's take a look at this. A second order cone program – this is SOCP. There was a confusing while by the way in the '90s when these were referred to as – some people were trying to call these QPs and it never stuck. But there was a brief period with a small group of where they tried to get this to be called QP, but it just didn't – fail. And it seems like SOCP has won out as the term. So here it is. You minimize the linear function subject to – and the constraints are quite interesting. It's a norm of an affine thing is less than or equal to an affine thing. Notice – very important

here, it is not norm squared. If this was norm squared, this problem would be a QCQP. It is not norm squared. This sort of the square root of a convex quadratic. That's what this thing is. Okay?

And it's called second-order cone programming because each constraint is nothing but this. You have an affine mapping that maps X into AIX plus BI, that's a vector, comma, then a scalar – CI transpose X plus DI – and that that should be in the unit second-order cone, the Lorentz-cone in RN plus one. Okay? So that's each of these constraints. Now this is more general than everything we've seen so far. It includes – well, not the linear fraction. It includes linear programming. That's obvious because you just make that zero here. It includes linear programming. You can rewrite quadratic programming and you can rewrite QCQP in this form. But it's more. There are problems you can write as an SOCP you cannot write as any of these, so this is a real extension. There's also something very interesting to point out here. If you write the – if you rewrite this in sort of our standard form, that's AIX plus BI minus – I guess you'd write it this way, right? Minus CI transpose – minus DI is less than zero. So this would be FI of X. Okay? That would be our standard form to write this. Here's a very interesting fact about FI of X. It's not differential. This is the first problem we've seen that has that feature. It looks differential. I mean – well, it actually – this function – of course, that's affine. That's differentiable everywhere. By the way, when is the norm differentiable? The two norm? Where is differentiable? Where is it not differentiable?

Student:[Inaudible] is the origin.

Instructor (Stephen Boyd):It's not differentiable at the origin. How about everywhere else? It's analytic everywhere else. It's got derivatives of all orders, right? And in fact, just think because of the graph. The graph is like this ice cream cone. It's perfectly smooth, no problem except for that tip.

So by the way, you might say then, "Oh, come on. That one point makes that much difference?" This is like what are we in the math department? Only a mathematician would worry about that – these are the kind of things you would think but not say. You would say what kind of a human being would worry about this one point where it's non-differentiable out of all RN. What do you think of that? Let's not worry about it. Well, you know obviously that argument is totally wrong. It turns out what is interesting about second-order cone programs is that of course the solution is very often at the point. Obviously, if you have pointed things and you optimize things like linear functions, of course you end up at the point. So yes, these functions are non-differentiable at only one point. Well, okay at the whenever AI transpose X plus BI is zero, it's non-differentiable. However, it turns out that's where the solutions lie. That's what makes this useful in applications and so on. So it's not just something – well, people in math are correctly interested in the non-differentiability. So this is a non-differentiable problem.

By the way, it would be completely unknown how to solve this efficiently let's say in 1988 or something like that. There were some people who knew about it by 1988 in Moscow. That was about it. So now it's completely routine, so it's as if it's always been

here, so people do this all the time. By the way, I think it's in Excel now, so to give you a rough idea of what's happened. Well, a lot of these things actually are. Okay. Let's look at an example of second-order cone programming. So very interesting is this – let's start with a linear program. Let's minimize C transpose X subject to AI transpose X less than BI, and what we're gonna do is we're gonna add the idea that there's uncertainty in C, AI, and BI. And in fact, you know what? We'll worry about uncertainty in the A and B. By the way, earlier in the diet problem, we're worrying about cost variation. I'll just make C fixed, and I wanna worry about variations in AI. By the way, if you wanna go back to the diet problem and figure out what that would mean, what would it mean that there would be variations in AI? I mean in terms of the diet problem. What's the implication? What would it mean? What does it mean.

Student: Maybe there's variation in how much nutrition you absorb from certain food.

Instructor (Stephen Boyd):Exactly. That's exactly right. So the point is when you get some foodstuff and you analyze it on Monday, and Tuesday, and Wednesday, it's close but it's not the same. And therefore, if you compose a diet actually, and you ask for so many calories or whatever, and you form it this way and you just use the mean, you might not actually on a sample by sample basis actually have the number of calories you want, in fact the number of calories in distribution. So that's exactly what it is, so it might be nutrient variation or something in the foodstuffs. Okay. Actually, pretty much any LP that's ever solved, here's the way it works. The coefficients in AI, the actual numbers in it – by the way, often they are zero. And by the way, when they're zero, they probably really are zero. It basically means if the third entry of A1 is zero, it means that X3, the third variable, doesn't enter into the first constraint. And the ones by the way often also have the flavor of being actually one because you're sort of adding something. And typically the minus ones, too. Actually, when you see a minus one, it typically really means minus one.

Any other number you see is probably – probably traces back to – if you trace the provenance of that number, it will go back to something which is not known. It will go back to some analyst or some estimate of a price. It'll trace back to some geometry and some Young's modulus or whatever. It'll trace back to all sorts of crazy things. But for sure, any number like that has some variation. You might know it within 1 percent. You might even know it within 0.1 percent. It depends on the field and application, but it's not at all uncommon that you would know it to like 10 percent for example. So that's just to point out that this is quite realistic to have this – to model variation in data in a problem. Now you could do this lots of ways, and later in the class we're gonna look at this in much more detail, but this is just for now just an example of SOCP. Two common ways are this, and it's strange people argue about which is better, and the answer of course is it depends on the application. So in a deterministic model, here's what you would do is you would say something like this. You would say that each of the AIs lies in an ellipsoid. That would be something like an uncertainty ellipsoid. And you would insist that the constraint hold for any AI in the ellipsoid. And so the other names for that would be like a worst case or model or something like that.

By the way, the people who would do this would go and give you long philosophical arguments. They'd say the greatest advantage of their approach is that they don't assume a probability distribution because there aren't any and blah blah blah. They go on and on with this. Or it's a worst case, and it doesn't matter. Detractors would say that's way too conservative because you're assuming the worst thing that when you buy all these foodstuffs and put them together and blah blah blah – anyway, it all makes no sense. And then the other model of course is stochastic. So here you'd say AI is a random variable, and this is called a chance constraint. You'd require that the constraint be satisfied with some reliability. So [inaudible] might be 0.9, more commonly 0.95, 0.99. You might even see 0.999. And by the way, the embarrassing thing is that these two are not that far apart because for example if that's 0.999 and you have a three sigma ellipsoid and you put that over here, you're very close to doing the same things, but it doesn't keep these people from – they still fight with each other. So that's the – and it's very strange because it doesn't make any sense at all. It depends on the application, what's the cost of violating the constraint, and so on and so forth. It all depends.

So [inaudible], both of these turn out to be SOCPs, second-order cone programs, and the way you do that is this. Let's parameterize the ellipsoids as a center plus the image of the unit ball under a matrix PI. So this is one parameterization of an ellipsoid. And then the robust LP looks like this. It says minimize C transpose X subject to AI transpose X is less than BI. That's for all A in this ellipsoid. By the way, some people make a huge big deal out of this. They call this constraint here a semi-infinite constraint. Got it? Why is it semi-infinite? Because you see this represents a single linear inequality for each element of an ellipsoid, and there's an infinite number points in an ellipsoid. Did you know that? So that's why this is a semi-infinite thing. Big deal. Right? Although it doesn't keep people from writing whole books on it. I'll go on. But this is easy to – when is it true that AI transpose X is less than BI. Let's fix X. You fix B. A varies over this ellipsoid. When would this hole for every element in the ellipsoid – you have to check whether AI bar plus PIU transpose X – whether that's less than BI for all U of norm less than one. Okay? But that's easy to do because this is nothing but AI bar transpose X plus – and then I'll make it this way. I'll write it as U transpose times PI transpose X like that. Now if I wanna maximize this over all U with norm less than one, that's a constant. This is an inner product of U with a vector. And so obviously by the Cauchy-Schwarz inequality that this thing, the largest that number could be is the norm of PI transpose X. And by the way, the worst U would be PI transpose X divided by norm PI transpose X. That's the worst U. Okay?

So I plug in the worst value and I get this, and I insist that that should be less than BI. And that is right here. Now if you stare at this closely, you'll realize that's an SOCP because you have just what you – you have a norm, two norm, of an affine function, and a linear, and a constant. It's SOCP. By the way, let me tell you a little bit about what the implications of this are. People can solve SOCPs now basically as fast as LPs, so almost as fast. Certainly for all reasonable – for huge numbers of situations, it's the same speed, same reliability, everything. That means people can solve robust LPs basically at no additional cost – little additional cost over an LP. That has lots of implications. By the way, it is – that fact or observation is propagating out, so it has now hit finance. So that's

why SOCP solvers are in Excel now because whenever you solve an – basically, whenever you solve an LP, then you ask the person how well do you know the data. If they don't say perfectly – and if they do, they're probably a liar – but if they don't say perfectly, then you should say then you should be solving SOCP. And actually, a lot more people these days will say that's exactly what we're doing.

So this actually has important – lots of practical consequences. It's not fully diffused, but it's getting there. Let's look at the stochastic approach. So in the stochastic approach, we'll assume that the constraint vector is Gaussian with mean AI bar and covariance sigma I. Now when you form AI transpose X, that's just a scalar Gaussian random variable. It depends on X of course, but we're fixing X. It's got mean AI bar transpose X and variance sigma X transpose sigma I sigma. And so the probability that you meet your constraint is equal to the CDF. That's a normalized random variable of course. Right? That's BI minus AI – this is the probability. That's the normalized random variable or something like that. This one. And then you put BI here, and this gives you the probability that you meet this constraint. Okay? By the way, these are called chance constraints, and problems that generally involve constraints that involve probabilities of something are – that's a whole field called stochastic programming. This one actually is [inaudible] enough to actually in my opinion deserve a field name. That's not true of some of the others, but this is actually complicated stuff, and I'm okay that there's whole books written on this.

But in this case, in this simple case we can write this out analytically as this, and it's an SOCP. Actually, it's interesting. It's an SOCP provided the inverse CDF of a Gaussian evaluated at eta is positive. That happens for eta bigger than 50 percent. So actually, it's extremely interesting. We can do chance-constrained linear programming. In other words, I can have a linear program with Gaussian random constraint vectors, and I can impose the constraint that each constraint is satisfied with a certain probability. The important part is that probability has to be more than 50 percent. By the way, which corresponds exactly to risk aversion versus risk seeking. So the one we're really interested in probably is eta equals 0.95, 0.99. These are the ones we're really interested in. We're probably not interested in 10 percent. That's risk seeking. Anyway, that doesn't matter because we can't solve a problem for 10 percent risk aversion anyway. Okay. Our next topic is geometric programming. By the way, for all of these there's no reason – you should have read the chapter, so this should not be the first time you're seeing these things. You should not be following everything I'm saying because I'm going fast, but I assure you all of these problems you will encounter multiple times during the rest of the quarter, so you should just be kinda getting a rough idea here. You'll see all of these again. So don't be concerned if – I'm going fast. That's all I'm saying.

Geometric programming – this is an interesting special class of – we'll see they transform to convex problems. This also by the way has a long history. It goes back into '70s. Actually, this goes to the '60s, including the first book written on it. It's super cool. It's this book where they – it's a whole book on geometric programming. I'll say what it is in a minute, but in the last paragraph in the book it actually says – literally like in the last paragraph it says, "You know, it's entirely possible that some day these problems could

be solved using a computer." No, I mean it. It says that. The whole book is about these cases where you would solve it by hand by various ways, but it's very cool. By the way, they had it exactly right. So this has a long history. Actually, probably elements of it were known in statistics in the '60s. And it comes up in a lot of other fields. It's a weird thing. When you first see it, you'll think, "I have never ever heard of such a thing," but once you start looking for these, they're like everywhere, or in certain fields they're everywhere.

All right. Here's what it is. So unfortunately, there's some deeply unfortunate nomenclature here, but it came from the '60s and nothing we can do about it now. Just for the record, right – a monomial – people in math have been using the word monomial for like I don't know 400 years or something like that? And it always means the same thing. It's a product of a bunch of – it's a single term polynomial. It has always meant the same thing. There's absolutely no idea as to what monomial would mean, but they decided they would take this term which everyone agrees on has a meaning and extend it. So in the context of geometric programming or GP – you have to be very careful with GP, though. If you type GP into Google, you're going to get several things. You're going to get geometric programming, but you're also going to get something called genetic programming, very different. I'd better not say anything about it. We'll just move on here. So in the context of GP as in geometric programming, you have something called a monomial function. It's a local definition, so don't ever say this outside in public or something like that. Don't ever say that like X to the one half is a monomial because if there's anyone – I mean unless you're only around people known to be talking at that moment about geometric programming because you'll sound like an idiot. It'd be like changing the word polynomial or something like that, and saying, "Well, that's what you call a polynomial, but I call a polynomial this."

So here's what's a monomial. It's a positive coefficient times a product of variables, each one to a power, and the power can be anything. It can be an integer, it can be irrational, and it can be negative. So that's a monomial. This does come up in a bunch – I mean obviously it's clear this comes up in a lot of engineering. People call this a scaling law or something like this. It depends on the field you're in. It's a scaling law or something. Now again, don't look at me. This was not my idea just for the record. They came up with this thing called a posynomial, which first of all sounds stupid, No. 1, and it also is stupid because it's supposed to combine positive and polynomial. That's okay fine, but the point is these aren't even polynomials because the exponents here can be like minus 0.1, and they can be 1.1, so anyway I guess languages are living, and that's a stupid name and it stuck, so posynomial. There it is. It's a sum of these things. So you know an example would be something like this, okay? There. That's a monomial. Okay? That's a monomial, and here's a posynomial. X1 X2 plus square root X2. There you go. And there's a posynomial.

Here's a GP, a geometric program. You minimize a posynomial subject to some posynomial inequalities less than one, and a bunch of monomials are equal to one. Okay? Now you might ask why the one as zero had been our standard before. There's a real good reason. The theory of a GP with right-hand side zero is a very short theory because

monomials and posynomials are always positive. So the branch of GP where the right-hand side was a zero here didn't go very far because all such problems are infeasible. So that's a GP. Now by the way, this is not remotely a convex problem. For example, you could say minimize square root X and finish with some stuff down here. That would be a GP. Obviously, it's not convex. Square root X is concave. Okay? So these are not convex problems. However, these can be changed by a change of variables. These can be converted to convex. And this is it. It's actually quite interesting, the conversion, and it's not at all unexpected. So if the variables are positive, you should have at least a slight urge to take the log. That would be normal. So if the variables are positive – we'll just take these variables YI so that XI is E to the YI. And then let's see what happens. We're also gonna do the same thing – we're gonna minimize. Well, you can put log in front of anything you minimize because log is monotone, and I could put log FI less than or equal to zero, so now our friend zero is back on the right hand side. Okay?

But let's see what happens. What is [inaudible] zero? And I'm gonna write this in kind of a – I'm overloading some notation. That's the vector whose entries are E to the YI. So the question is what is this thing. Well, let's take a monomial, and let's take the log of this monomial. Well, you get log C. That's B. Then you get plus, and then you get the exponents. You get A1 log X1, but log X1 is Y1, so when you take – when you change variables by this logarithmic transform, and you take the log of a monomial, you get an affine function. That's good. Now take a posynomial. So here I take the log of a sum of these things, but that's the same as log sum X of an affine function. Okay? Because each of the Xs I replace with E to the YI like this. Then I take a sum of those things, and I take the log. That's log sum X. That's this thing. And if you look at this, this function is convex in Y because it's log sum X of an affine function of Y, so it is definitely convex. That's not remotely obvious by the way here, not even remotely obvious. I mean it is once you've seen it and all that, but – in fact, articles on geometric programming that didn't understand that it converts to convex programming by a change of variables persisted until like the early '80s. So this is not so obvious. So what that means is this GP over here which is not remotely a convex problem – I mean also look at the equality constraints. The equality constraints would be things like X1 to the 0.3, X2 to the minus 0.7, X3 to the 1.2 is equal to one. Well, that has no place in a convex problem. The only equality constraints you can have in a convex problem are affine – are linear. But it transforms to a convex problems like that, so it just transforms right to a convex problem, and is actually it turns out – by the way GP comes up in lots of fields. It comes up in statistics and machine learning all the time. It has to do with exponential families and maximum likelihood of estimation. We'll also see that it's intimately connected to entropy type problems. We'll get to that. It comes up in information theory, but it also comes up in circuit design and seems to be completely ubiquitous. And it comes up in power control and wireless. I'm just mentioning this, and we'll see a lot of these later.

Student: How did you get the monomial F to transform into the log is equal to [inaudible] in between?

Instructor (Stephen Boyd):Can you go over to the pad here, and I'll go over that real quickly down here? So the question was how did I get – if you go down to the pad here,

I'll just work this out. Okay. I'll just describe it. You take the log of a monomial, and the first term – oh, you can see it. I see, but maybe it's not on – okay. That's fine. If you can see it, that's fine. But for those of you who are sleeping right now, this is what happens when you sleep through the class. There we go. All right. What we're gonna do is we're simply gonna take the log of this. That's log C plus A1 log X1 plus dot dot dot, plus A2 log XN, but log X1 is Y1, so I get exactly that. That's how that transforms. And what I was saying was that it comes up in a lot of applications. Once you're sensitized to GP, you'll start seeing it all over the place. In fact, kind of in any problem where the variables are positive, they represent powers, intensities, or something like that, densities, you can start thinking how GP might come into play, and we'll definitely see lots of examples of these, of GPs. Actually, it's quite interesting because they're quite natural. For example, in wireless – and we'll see examples of this, but for example in wireless power allocation or something like that, the Xs are gonna be the powers of transmission, or they could be like the powers allocated to channels in some big multichannel system or something like that, or different frequency bands. And what this transformation says – I mean if someone's doing optimization, they say you know what it's better to work with the log of the powers, but that's what engineers have been doing for like decades and decades because these are called decibels. So you work with – so it says you should work with decibels. No one would be surprised by that.

And then the constraints would be things like if signal to interference ratios should exceed like some factor. When you take the log of the constraint, you're basically saying that you should write your constraint this way, that the signal to interference ratio should exceed eight decibels. Again, that's how they would do it. People don't the SIR has to be bigger than like 2.8. They say it's gotta be bigger than you know 8 dB or something like that. So it's actually interesting that exactly the way people who would actually work in these fields would use these things turns out to be – corresponds precisely to the convexifying transformation of variables. We'll see lots of examples of this. Let's look at a mechanical example, and it's design of a cantilever beam. So here you have a bunch of segments. This one just has four, but you know obviously you'd have lots more. And what we're gonna do is this. It's gonna be – so it's gonna stick straight out, and a force will be applied to the tip, and two things will happen. Of course, this thing will deflect. We're not gonna do the huge horrible deflection. We're gonna do the small deflection so that we can use linearized approximations. So it'll deflect substantially, but we'll assume we'll use linear models and things like that. It will deflect, and there'll actually be a strain – a stress – well, both on each segment. Okay?

And what we wanna do here is obviously if I design – the thicker I make the cross-sections, obviously the stiffer the beam. Yeah. Stiffer meaning if I apply a force here it deflects less and the stress is less. So if I make it a big thick thing all the way out – and you can easily – and you probably have a pretty good idea intuitively of what you want here. You certainly wouldn't want a beam that looks like this, that tapers out like that because now you're putting a whole bunch of weight over here. Actually, you're just making it much worse. You'd probably want one that tapers like that. That's just completely intuitive. So the question is gonna be to get the optimal tapering of a beam. That's a cross-section. We're gonna design the shape of a beam, cantilever beam. Okay.

So we'll minimize the total weight subject to – and what we'll do is we'll – and we're gonna actually – these are not – these have a width and height, so there's a height. I know very little about mechanics except I do know that sort of the strength of – well, from carpentry that the strength of thing goes up like the cube of the height, for example. Beyond that, I'm sure there are people here who know a lot more about this than I do. So I've got lower bounds on the width and the height of the segments. We'll limit the aspect ratio so you don't get like an infinitely thin thing like this. We'll upper bound the stress, and we'll upper bound the vertical deflection at the end of the beam, which is actually where the maximum deflection occurs. So that's the same as – Now, this is actually quite a complicated problem. It's highly nonlinear and so on and so forth. So but let's say it's actually a GP. To see how that works, we look at the total weight. Now, these are variables. By the way, what kind of function of W and H is that? Again, outside the context of GP, if someone walked up to you on the street, what kind of function of W and H is that? It is a function of W and H, and I wanna know what kind it is. Linear? No. It's linear in W if H is fixed. And it's linear in H, so this is not – well, maybe I wasn't clear. There wasn't two questions, so I'll say it again correctly. What kind of function is this of left paren W comma H right paren? What's the answer?

Student:[Inaudible].

Instructor (Stephen Boyd):It's quadratic. Okay. Is it convex quadratic? Hey, wait a minute here. You should not hesitate.

Student: Yeah.

Instructor (Stephen Boyd):No, it's not. It couldn't possibly be. What kind of quadratic form does this have?

Student:[Inaudible].

Instructor (Stephen Boyd):There's nothing on the diagonal. Diagonals would be things like WI squared, HI squared. There's nothing on the diagonal. This is block like zero diag diag zero. Now, do you really have to think whether a matrix like that is positive semi-definite? I don't think so. Positive semi-definite things need non-negative on the diagonals. Of course, this has zero on the diagonal, but there's a rule. If you have a positive semi-definite matrix and it's zero on the diagonal, that row and column is zero. So this has split eigenvalues. So this is a quadratic function of W and H, but it is not remotely convex. It's got split eigenvalues – or concave. However, it's a posynomial because each of these is a monomial, and it's a sum with positively weighted blah blah – it's a posynomial. So that part is cool. Now the aspect ratio and inverse aspect ratio are monomials, so if I set them less than a number, I could divide by it, and I'd get a monomial – well, they're posynomials, so I can make an inequality of them. That's fine.

And the maximum stress is given by this, and that's a monomial here. Now the vertical deflection and slope are very – it's quite complicated to actually work out how much this thing – the tip deflects as a function of the width and height of each of these beam

segments. It's very complicated. Okay? But it's given by a recursion, and the recursion looks like this. It says you take VI, and that's 12 times I minus a half here. I think this – in this case – this is for I equals one to four – no, it says for I equals one to N. Sorry. Everything's cool here. The VI is this thing. It's F over – that's a constant – E is Young's modulus, WI – these are the variables, so this term – hopefully, I can get this right. Yes. Okay. What is that term in GP terminology? What's that?

Student: Monomial [inaudible].

Instructor (Stephen Boyd):Monomial. That's a monomial, right? It's got a positive coefficient in front because I is one is the smallest thing here, right? F is positive. It's a positive force on the tip. Young's modulus, positive, and then it's actually W to the minus one, H to the minus three – HI to the minus three. Okay? So that's a monomial. And now you add this one that's gonna go from the tip back plus this. So actually recursively here, the first one is a monomial that – sorry, the last one is a monomial, and going backwards you take a monomial and you add it to the previous one. So by recursion, the VIs are monomials – sorry, posynomials. They're all posynomials. They're sums of posynomials all the way to the front. Then you get the YIs this way. Well, each YI is the next one – well, in the first one it's gonna be this plus that. That's in fact in the last case it's actually a monomial, so the last YI's a monomial. And the second one is posynomial, and all the others are posynomial. So what that says is that this is quite – these are very complicated expressions. You wouldn't want to see them written out, but they are indeed posynomials, all of them. So that's actually quite interesting. It's not remotely obvious. And let me say some of the things that it implies. That's not obvious, and I really doubt – actually, I think I can say with extremely high probability there's absolutely no one in mechanical engineering including someone with super good lots long design history stuff like that would know this. And let me just say one of the things you can conclude right away.

It says that if you have two designs, two cantilever beam designs – so imagine two. One is quite wide and high, and it makes no difference what they are. I'm gonna form a compromised design from the two beams. However, what I know to do is the following. I know I should take a log of everything. So the compromised design, instead of taking like WI and WI bar, and taking the average of the two – actually, my compromised design is gonna be the geometric mean because I know I really should be working – the problem mathematically is more natural in the log of the variables. So if you have one log width, you have another, I should take the average of the log widths and then take the X of that. That's the geometric mean. So the first thing it says is the way you should blend two cantilever beam designs is by geometric mean. By the way, that fact alone is not obvious at all.

And then there's a stunning conclusion about it. It would be this. If we're solving this problem – if you have two designs that meet all these specifications – these are fantastically complicated. When you start talking about the tip deflection and stuff like that – two designs that do it. If you take those two designs and form a compromised, a blended design given by the geometric means of the other ones, I can say something

about the total weight of that blended design. I can say that it's less than or equal to the geometric means of the weights of the two designs. Okay? So for example, if those were two different designs that satisfied the constraints and had the same weight, that blend could only be better than either of them. Everybody see what I'm saying here? By the way, in other areas I've always tried this. I go to a field. I find someone who alleges to be a good designer with lots of experience and stuff, and I talk them through what the implication of geometric programming is in their field. And in only one case – it was Mark Horowitz who actually said yeah, that makes sense. He didn't say it right away either, by the way. Actually, I believe him. This is in the context of circuit design. I really believe him. He says now that makes sense. That's how you blend two circuit designs and so on.

I think there might be someone who does enough mechanical engineering that this would at least not sound stupid. Actually, it never sounds stupid. The question is whether or not – because you could make very strong statements about what happens when you make these blended designs. That's just one of the implications of what it means to be a GP. So how do you write this as a GP? Well, that's a posynomial directly, and you write out all the inequalities, which I won't write out. I won't go through this. I think you can go through this yourself. Let me look at one more example of a GP. It's a more generic one, also very interesting example. I don't know any practical uses of it, but there might be some. Actually, I've been itching to find one for years, never have. Here it is. Let's take a matrix that's element-wise positive, so a square matrix element-wise positive. So you might know about something called Perron-Frobenius theory if you've had a class on Markov chains or something like that, but I'll tell you what it is basically.

It's this. It says that the eigenvalue of A – A's not symmetric, so it can have complex eigenvalues. But it says that the complex value of largest magnitude – that it's spectral radius is that magnitude – is positive, No. 1, and also the associated left and right eigenvectors can be chosen to be positive. So that's Perron-Frobenius theory. I wanna see how many people have seen this in some context. I'm just sort of curious. That's all? Markov chains. Somebody took a class on Markov chains, right? What'd they say about that? Really? They didn't mention this? Okay. How do you know that the steady state equilibrium distribution is positive? Or in this case positive. This is the strong form. Where did you hear about it?

Student:A Markov chain class.

Instructor (Stephen Boyd):A Markov chain class? Here? No. It's not looking good for Stanford, is it? Where was that?

Student:[Inaudible].

Instructor (Stephen Boyd):University of Washington. Oh, okay. All right. And this is called the Perron-Frobenius eigenvalue. It's positive. And of course, what it does is it tells you the asymptotic growth rate of A to the K. So if you take powers of the matrix, if it's a dynamical system with positive entries, it says it'll tell you the growth rate. So for

example, I mean this would come up in lots of dynamical cases. This would be for example an economy. It would tell you how an economy – or it might tell you how a multispecies system would work. Actually, the condition that A should be element-wise non-negative means that everything is I guess you would call it excitatory and nothing is inhibitory. I don't know if those are words, but you know what I mean, right?

So it means you start with a positive X. These are [inaudible] concentrations of something, and the concentration of anything does not inhibit the growth of anything else, so it would just lead to it. It would increase it. It would be excitatory. So in that case, A to the K tells you how this dynamical system would propagate, and the asymptotic growth rate, which could be negative by the way, is simply completely determined by this number. If this number is 1.05, it says that by the time you do this 100 steps, you've grown like 1.05 to the 100. If it's 0.96, it asymptotically decays as 0.96 to the K. So that's what this is. Fantastically complicated function of A, right? Because you calculate the eigenvalues of A, which is to mean you form the characteristic polynomial, which as a function of the entries of A you don't wanna see written out because it has something like N factorial terms. That's No. 1. You're gonna have to write – you're gonna have to find the roots of that polynomial, which again you can't do if N is 5 or bigger because there is no formula for the roots of a polynomial of degree 5 or bigger.

And then, once you get these N roots, you wanna take the absolute value and take the maximum of them. And I think by now you should be convinced, this is a fantastically complicated function of the matrix A. Okay. Well, another way to characterize this is that the Perron-Frobenius eigenvalue of a matrix is it's the smallest lambda for which AV is less than or equal to lambda. By the way, for some positive V – it turns out here the first thing you actually end up showing is it's that. It turns out you never get inequality here. It's always equality here, which of course makes sense because now lambda's an eigenvalue like that. But what's amazing is this – is you can now minimize the Perron-Frobenius eigenvalue of A where the entries of A are themselves posynomials by making a problem that looks like this. You minimize lambda subject to A of X sum – and these are all monomials in the variables, which are X, V, and lambda, and these are posynomial inequalities. So basically, log sum X comes out of this. That's how that works.

And you can make up some fake applications of this, I suppose. I think we attempted one in the book or something like that, but you can – it'd be nice to actually find a real one somewhere. That's a challenge, by the way. So I think what we'll do is we'll quit here, but let me say just a few things about where we are. We're sort of marching through all these problems. You will get to solve in a theoretical and in a numerical and practical sense every type of problem we're talking about multiple times. So we'll be doing all this for the next seven weeks actually, we'll be doing this stuff. So this is just your first introduction.

[End of Audio]

Duration: 73 minutes