ConvexOptimizationI-Lecture18

Instructor (Stephen Boyd):Right this far – we're on? So we didn't get very far. We just started talking about the basic ideas of barrier methods. Now, but we'll really do it today, so we'll look at how it all works.

Okay, so the idea actually is very simple. If you can go down to the pad here. You start by rewriting the problem. You take the constraints, and you simply put them this way. It's f0 plus – there we go. Plus, and then this. I minus is the indicator set of negative reels.

And so i is this function that's zero, and then goes to plus 8 if you're above zero. And this is, of course, this encodes the semantics of a hard constrain. It basically says you have no irritation if it's less or equal to zero, and if it's positive, you're infinitely irritated, meaning, that that x is completely unacceptable.

So this – I mean, we can then say look it's just an equality constrained problem here, but if, of course, this is just rewriting things. And it doesn't really help anything, because this function, it is true the only constraint now is an equality constraint, but it's function is, of course, is highly non-differentiable and because we've put this here. That's exactly the point.

Okay, so what we're going to do is we're basically going to replace this indicator function with an approximation of the indicator function that's smooth. So and this is being many ways to talk – a lot of people talk about this. It's something like this. When I'm making fun of people who propose things like this I usually say this is the same as getting out some sandpaper and saying you don't like this corner here so you sand off the corner or something like that.

So we talked about this at the end of last lecture. So a very common approximation is the log barrier. So the log barrier goes like this. You have the sum of the logs of the negatives f. Obviously, if f becomes zero or positive this goes to minus 8, and with a minus sign, it goes to plus 8. It's a barrier. And you put a one over t in front. T is going to be a parameter that controls, you know, how close this thing approximates this indicator function.

And it's important to understand the tradeoff here. The tradeoff is this. The larger t is, the closer this function looks to that. That's clear. But that's both good and bad. It means that by solving this problem you get – well, we hope it means. We'll see soon if it's true that you're getting a closer and closer approximation of this problem. It certainly means that. But on the other hand, the larger t is and the more this function sort of looks like this one, where it's just basically this indicator function with the corner sanded down. That's going to mean this is a harder problem solve using Newton's method.

Okay? So that's going to be the trade off here. And we'll see how that works around it. So that's the idea – is that we're going to solve this using Newton's method. By the way,

there's nothing wrong with solving this with Newton's method. This we know how to solve that. Any value of t, you have a feasible point. We can actually, a strictly feasible point. Newton's method will work perfectly on this. It's just no problem.

Okay. So this function here, which is the sum of the minus log of the minus constraint functions is called the logarithmic barrier function associated with the constraints. So that's the log barrier. And this minus log minus f. That's convex function by one of these composition rules, because minus log minus something is an increasing function and so on. Increasing in convex. And then f is convex as well, of course, so that's convex. It's a log barrier, and you can get a rough idea of what it is. For example, another way to say it is this. It's the log; it's minus the log of the product of the minus fi's.

So these are slacks, because that's literally how much. That's how much extra you have in the inequality fi of $f(x) < 0$. So minus fi is the slack. And this is negative log, the product of the slacks. So that'll come up a bunch of times.

Okay. For the record, we'll work out what the derivatives are. So the gradient is simply this – because that's just a composition rule for the log. And the Hessian looks something like this. It's got these rank one terms here, and then it's got these things with the scaled version of the Hessians of the individual functions, okay? So that's the – these are the gradient Hessian. We'll come back to this. We'll actually need the gradient very soon.

All right. Now, when you minimize f0 + 1/t fi(x). It's the same as minimizing t(f) 0 + fi(x). It's the same thing. And when you minimize this, we'll actually refer to the minimum of this as x *, the argent of x *(t). So that's going to be the minimizer of this function, subject to that. And we'll call that x(r)(t).

We're just going to assume here that there's a unique minimum, and we'll call it x *(t). There – well, we'll get into that later. Actually the sublevel sets are bound to this unique minimum, but we'll just leave, we'll just say this is the minimzer is x * (t).

This is a function of t, of course, and it traces out a curve. A path. And that's called the central path of this optimization problem. So you call that the central path, and the intuition is something like this. As t goes to infinity, you would sort of guess that x*(t) is going to go to an optimal point of the original optimization problem, because as x*(t) goes, as t goes to 8, the difference between this thing and this. I have to include the minus sign there. The difference between this and this basically goes away. Because as t goes, certainly element wise, point wise – well except if you're at zero. But element wise it's fair to say that this thing converges to that as t goes to 8.

So you sort of guess that this is the case. And here's an example for an LP. This is an LP and r2, of course, so it's kind of silly. But here's you've got some inequalities. I guess one, two, three, four, five, six inequalities like that. And then these dash curves show you the sublevel sets of the barrier function. That's this thing.

Like that. That's the sublevels of the barrier function. And you can see the one in the middle, that's with t = 0. That's, you minimize the log barrier. By the way, that's the same as maximizing the slacks. The products of the slacks, or the geometric mean of the slacks. So that's actually called the analytic center of this polyhedron. And that's right there. And then as you increase t now, what happens in you're incentivized to – well, let's talk about what phi does. Phi is a barrier and what this does, it wants to keep you away from the boundaries. Because if you get near a boundary, one of the fi's is going to be negative, but near zero.

That means one of the minus fi's is going to be small. One of the log of that is going to be big and negative. And with the minus sign, it's going to be big and positive. So this barrier function, if you can visualize it goes something like this. It's got its minimum there. And then I guess it curves smoothly up. It's an analytic function. It curves smoothly up. And then when it hits these walls it goes to infinity. So you should visualize that here.

Now when you crank up t, what happens is – by the way, you'll never find a point that is outside the interior of the feasible set. Because this thing, the domain of this is the interior of the feasible set. So no matter how high t goes, you'll always have a point that is interior. And by the way, that's why one of the names for these things is interior point methods.

We'll see later actually that modern interior point methods don't work this way anymore, but still the name has stuck and that's why they are called interior point methods. Because every point you produce is in the interior. That used to be the case, anyway.

Okay, so what happens is you crank up t, and you're going to move more and more towards. If t is this c, you're going to move. This is the curve, this is the central path, and you can see that as t goes to 8 you're going to approach this optimal point. Okay? So that's the picture. That's how this works. By the way, so far you should be deeply skeptical of all this, because well, for the reasons I talked about last time, but I'll talk about them again.

Let's see, the other thing I should mention is this is a sort of a famous method. There's a general class of methods that are worked this way and they're called homotopy methods – like this. And I can sort of explain a little bit about how these work.

A homotopy method works like this. You have a problem that you want to solve but it's hard. And you don't know how to solve it or something like that. What you do is you make a parameter. You then introduce a parameterized family of problems. Parameterized by, let's say, a parameter t – let's say. And you arrange for the problem to be easily solved when t has some value like zero. In this case, it's just – you've already written code to do that like two homework's ago or something like that. So it's simple to find this analytic center.

Then what happens as you crank the parameter – the problem you're solving is going to transform into the one you want to solve. So in this case that happens, roughly speaking, as t goes to 8. So the homotopy method works like this. You have a parameter value. You solve the problem. Then you crank up the parameter a little bit and solve it again, starting from where you started before. Hopefully that works or whatever. If it doesn't work, maybe you were too aggressive in your t update or something like that and you back off. I mean, you can think of all sorts of methods for this.

By the way, these methods are very widely used. I mean, all over the place – so that's what this is. I'll come back to that. Okay.

Now let's actually do some. Let's see what happens on the central path. Well, if you minimize – f you solve this problem, then it means that the gradient of this plus a transpose nu is zero. That's the dual residual being zero, and of course Ax = b. So it means that the optimality condition for minimizing this barrier-augmented problem is this. You have to have the gradient t grad f0 +, and that's the gradient of the barrier, plus a transposed w = 0 and Ax= b. So that's the optimality condition.

Now if you stare at this long enough you will look at it and realize you've stuff like this before. And in fact, what you saw before is this. We can take this thing and we can divide by t. That's the first thing we'll do. Divide by t like that. And then you stare at this long enough and you realize these are just like positive numbers. So I'm going to call those just lambda i. And then you're going to look at this and say wait a minute, this is grad f0 plus sum lambda I grad f I plus a transposed something equals zero. And you realize like wow, look at this. If you define lambda i this way, this equality means that if you are on the central path you actually minimize the Langrangian with this value of lambda. This value here. Okay? So that's the picture.

Okay, that's great. If you minimize the Langrangian, then it means you actually have a point that's dual feasible. That's what it means. And that means immediately you're going to get a lower bound on your original optimization problem. So let's see what happens. It's actually kind of interesting. It says if you minimize this, which, by the way, you can do very easily for any value of t. This is Newton's methods to minimize this. But if you minimize this, then actually whether you like it or not, you will actually get a dual feasible point for the original problem. That's what's going to happen.

So you're going to minimize that, you're going to get. When you finish you're going to get this. You're going to define lambda i as minute 1/t, fi(x), like this. That will be lambda i. And you'll get a dual feasible point. And then of course you have a dual feasible point and you want to evaluate g. That's the dual function at that point and you work that out. So I'm going to plug this lambda into g.

Now g is nothing but this. Its f0. And I don't have it written here, but I'll write it out. Its going t be f0(x)+= lambda i, fi(x). Well, and then its plus nu transposed Ax- b, but that's going to be zero. So I'm just going to plug in the lambda i I found. This is going to be g of lambda. It's going to be that. But the lambda i, this thing is - 1/t, fi(x), okay? This f

cancels that. That's a sum over the constraints, so this is equal to f0(x) +. Here that's all the same number, so it's actually – let's see, of the lambda i is that. So I get – actually, it looks like to me, - m/t period. Okay?

Now, that's very interesting. It says that if x minimizes this thing, that's something you can do by Newton's method. Then, I also inadvertently calculate dual feasible point, and that gave me the following lower bound.

And that tells me the following, that the optimal value of this problem is clearly as the following property, certainly it's less than f – I have a feasible point x, so the optimal value is clearly less than that. But it's bigger than g of lambda, but g of lambda is this, f0(x) – m/t. Well, it's beautiful. You know what that says? It says that if you minimize this function here – which is a completely smooth function. So that's Newton's method here. If you minimize this, the solution will be no more than m/t suboptimal for the original problem, okay?

So now, you have the hard proof that, in fact, as t goes to 8 you actually compute an optimal point. In fact, it's much more. The truth is that as you solve this problem and vary t you can say much more. You can say the following. When you solve this problem, you will produce not just a primal feasible point, you will produce a dual feasible point. And you'll really produce a primal dual feasible pair. And the gap associated with that pair, the duality gap, the difference between the primal value and the dual value will be exactly m/t. So that's what this means. You have a question?

Student: Should lambda be the minimum of the right insight instead of f0(x)?

Instructor (Stephen Boyd):What's that? You had the right hand? Should the?

Student: Should the lambda be the minimum of right insight over x, so that you?

Instructor (Stephen Boyd):Right, sorry, but the reason x, my x minimizes the Langrangian. You know how I know that? I know that because this is the optimality condition for minimizing the Langrangian for this choice of lambdas. So I do know that it minimizes. This is the, if I plug in my x*, it does give the emphemom over x of the Langrangian. I don't know if that made sense - it was a.

Student: Where did m come from?

Instructor (Stephen Boyd):M is the number of constraints. Right, so this is very – this is pretty cool. And by the way, there's a method for solving the original. We already have our first method, by the way, for using Newton's method to solve a constrained problem, and in fact, it's got a name. We'll see why it's called that later. It's called – so the name of this method is called Bount. We'll see later, because that's a takeoff on a name that we're going to see. Sount, which is sequential, so I'll tell you what this is. This is sequential, unconstrained minimization technique.

Oh, and I like the name. I'll tell you why. It's kind of a retro name, because it points out that everything we're doing so far was actually known in the 60s. And so when you mention this name it's sort of a reminder because you can type this into Google. Well actually, if you type this into Google you'll probably get a lecture I wrote, but below that you'll find whole books on this in the 60s.

That irritates the people who imaged that all of this was done in the last 15 years so – and that's why I like to use the term. So this is the unconstrained minimization technique, and it works like this. This is really dumb. Watch this. You want to solve this constrained problem. Minimize f0(x), subject to fi(x) < 0 and x = b. The Bount method says this. Pick t and solve the following problem. I'm assuming here – by the way, you have a strictly feasible starting point. We'll talk about how to get one if you don't. It says minimize this. F0(x) + 1/t + times minus sum minus, whoops, log minus fi, okay? That is a smooth function. Newton's method will minimize it – well I should say certainly in theory. I mean, so will the gradient method by the way, in theory, minimize it.

But the point is you can minimize this. When you minimize this you will have a point that is feasible here, and actually it will be strictly feasible. And you will – you will absolutely guarantee that you are no more than m/t suboptimal. So you might say done. That's it. We're just done. Pick your tolerance. You want one e minus six tolerance. Pick t, whatever it is. Ten to the sixth times m or something like that. End of story. In theory that's actually correct, so Newton's method just one shot does it.

We'll actually see. It's more of when you combine this with the homotopy idea that you actually start getting methods that are a, both practical and b, pass muster with our complexity theory friends, okay? So that describes the s.

By the way, you shouldn't make fun of the Bount, because there's plenty of practical applications where it's entirely appropriate just to pick a value of t, minimize that, end of story.

Student: On the previous yellow sheet where you rote the dual function, I was wondering how you got Ax – b(0)?

Instructor (Stephen Boyd):Well because, look. Ax = b is part of the optimality condition.

Student: Right, but that is going to zero, right?

Instructor (Stephen Boyd):No, it's not. No, it's zero. No, it's zero. Yes, I know. I solved here. I minimized. Here it is, there we go. I solved that problem, and the constraint is Ax = b, so I assure you that Ax = b. Ax = b is not negotiable here. We have a point that satisfies Ax = b, so. Okay.

Everybody got this? So this is – it's also pretty simple. We're up to about 1967 here. Yes?

Student: Ax = b there also?

Instructor (Stephen Boyd):To where? Oh, over here?

Student: Yeah.

Instructor (Stephen Boyd):Well - I should yes. You mean if I want it to be true?

Student: Yeah.

Instructor (Stephen Boyd):Yeah, sure. Whatever, okay. Sure there. Sure. These are details. You know you don't have to do that in a 300 level class, right? You can look, it's an official rule. I mean, I can switch a minus sign. There's some number above which you're not supposed to have, but it's expected. In fact, you wouldn't even want everything to be right here, would you? It would be insulting. This is a 300 level class where, you know?

So – but thanks for pointing it out. Okay, all right. So let's actually interpret this stuff via KKT condition because you can, right? You can actually interpret it lots of other ways. Here's another one. And in fact, this one is also like quite cool. It's this. Let's take – this is the point on the central path. This is xr(t) minimizes wherever it is. I've lost it. Here it is. It solves this problem here. Lambda * is the lambda defined by that. Let me point one thing out. When you fix t, you don't know lambda * until you calculated x(t). So you don't know what dual point you're going to end up with until you do this.

And curiously, though, you do know exactly what duality gap you're going to have. You're duality cap, your gap is going to be exactly m/t period. So that's – in fact, one way to say it is when you solve this problem – which appears to be a primal problem whether you like it or not, you're actually going to calculate a primal, dual pair for the original inequality constrained problem with a duality gap exactly equal to m/t. That's what you're going to do.

In fact, you can even think of t as a parameter, just, well, m/t is clearly simply the duality gap. So you can literally dial in whatever duality gap you want be done.

Okay, and that would be the Bount method. Okay. So let's look again at what it means to be central. Oh, I should say if you solve this problem you would refer to a point like that as central, or on the central path, or centered would be another one. These are the names you would use for that.

Okay. Well, here's what these points satisfy. Well, in fact, this one I can make stronger. You're actually strictly feasible obviously because, in fact, the objective that defines x *(t) has log minus fi. So this is strictly par.

And in fact, these are strictly positive. Okay, because those are 1/t time s - fi, which is strictly, so these are positive. Okay?

Now here's the cool part. The optimality condition. Well, this is our definition of what lambda i is. We actually get - lambda i fi(x). This is if you're on the central path is equal to 1/ t. Now what's cool is the following. So we write down these conditions, and the optimality conditions for the original problem inequality constrained are really, really simple. And I will draw them right now – ready? It's just that.

Now that's true complimentary slackness is this, right? It says that lambda i, fi is zero. So if, in fact, you have x and lambda that satisfy. Well, now I do have to change this back like that, okay? These one through four with this zero here. That is the exact KKT conditions for the original inequality constrain problem. Okay?

If you're on the central path you satisfy all four of these, except three is modified form exact complimentary slackness to approximate, and this is just 1/t. So that's another way to say it. To say that when you have a point on the central path, you've actually found a set of points that satisfy three of the four KKT conditions and the one where they come up short is complimentary slackness. And in that one, instead on lambda i, fi being like minus lambda i, fi being small. I mean, zero, sorry, they're all equal to 1/t. They're all equal and they're just 1/t.

So that's another interpretation. This is actually the most common interpretation these days of these methods. So if you look at books and things on these methods they'll just start from KKT.

Okay, now you can also think of this in terms of there's a nice force field interpretation – and that's this. Forget the equality constraints to make it easy. What we do is we have a force field here – well, we have a potential. So if you think of this as a potential, and I'm going to think of each of these with the minus sign as the potential contributed by each constraint. So each constraint right now has a barrier function, but I'll think of that as a potential, which means a force field and the gradient of the force field is going to be - I'm sorry, the potential is obviously the force. So that's the way I think of it.

Now, what happens is this. The forces balance at – I mean, when you minimize the potential the forces balance. And that means here that the force contributed by our objective here. So this is a force contributed by this is the potential. This is the thing you want to minimize, is actually balanced by the forces of the contributed by the barriers.

Now, the barrier forces have actually a beautiful interpretation. I mean, especially when you have linear constraints. That's the easiest thing, so I'm going to do that. So let me describe how it works there. So here, for linear constraints – all right, that. Here's the way it works. My force field, my force field on top of this is simply in the direction c.

And by the way, you can make this gravity. So this can be g times whatever down, okay? So that's what this is. So it's simply a constant force field in the direction minus c. That's what this force field is, that's the objective. So here's our point were c – c is apparently this direction, so the force contributed by the objective is simply a constant force in this

direction. It's pushing you towards values of low c transposed s, which is obviously what you want to do.

Now, the force field associated with the each log barrier term. You take the derivative of the log and you get one over something, and it turns out it's exactly this. It's an inverse distance, and so 1/r force field.

So this gives you a beautiful way to actually picture this, understand exactly how this works. Here's what you do. Take each constraint and spray it with some repellant, okay? And the repellant has the following property. Wherever you are here, this plane will put a force on you. It will point away from the plane. That's the direction it will point in, and its magnitude will be actually exactly one over the distance to that plane. Everybody got that?

So by the way, there are no such repellants or whatever. I tried thinking of like some case and some number of dimensions where you could spray positive charge. No – there are none. But anyway, imagine there were such a thing.

So you spray the – and this means, for example, if you're here. Let's actually work out what the force field is there. You feel a very strong force from this plane, and you feel a very strong one from this one. You're about equidistant – oh by the way, what kind of force do you feel from these three? Much less. So basically the force that you feel at this point is – in fact, someone tell me which way to draw it. Like, this way? This way? This way? Well, it's like that, right? So basically it would be forced like that.

And by the way – that's actually good. That's where the barrier force is not fighting the objective. It's actually pointing you away, because basically you're at the completely the wrong point, okay?

So and you can actually imagine all sorts of things. Like what if you get right up to one of these constraints? Which way is the force pointing? Well, it's basically pointing you away from the constraint, okay? All right.

Now, imagine – now, put no force on it. So in fact, what we'll do is we'll just make this in a plane. So this is actually this sitting level, and what happens now is the particle will settle to the analytic center. That's where all these forces balance. It will be like – you know, it will be a point there or there. I don't know. Somewhere in the center, right?

Now actually all I have to do is this. That's c. Let me see if I can get it right. Yeah – okay, so here's what you're going to do. Take this page and start tilting it, because that's exactly what you're doing, right? Because you're putting more and more gravity force on it, and when you tilt it all the way, you have to be able to tilt it like, I guess. Gravity has to be way, way high here.

But as you rotate this thing, what happens is there is a gravitational pull, pulling this in this direction minus c. And it will move over a bit for each angle it will hang off down a

little bit until the repulsive forces from these three counterbalance the gravity force. Everybody see this?

Now as I crank the gravity force up super high you're going to settle over to a point here. You will – of course, you won't be touching these, but you'll be very, very near, and you'll be pinned right up against the active constraints. Near, but not actually touching the active constraints. And there, there repulsive forces will balance the attractive force generated by the objective, so.

This, it goes without saying, none of this actually matters. This is just for you to have a feeling for how this works, okay? Actually, let me just ask you a quick question to make you do get it. Let me ask you, suppose there was, how many constraints I did here. One, two, three, four, five. All right, you know what? I'm going to put some more constraints, ready? Like this, ready? Tons of constraints like that. Like, let's say ten of them. Okay?

Now let me point something out. The constraints I just added, they do not change the feasible set. So what's the optimal point? So of course, it's the same. It's right here. What does the central path look like now? Does it move? We haven't changed the feasible set. Has the central path moved? Where is it? Okay, what happens is these constraints, though they have no effect, they do not in any way change the feasible set. In any way. They are felt very strongly by this particle, because 1/r is a very slowly decaying force field, and what happens is the central path, the central point is probably. And this thing probably sneaks up on the optimal point that way.

Why? Because it's feeling a repulsive force from all these things, even though they're outside the feasible set. Everybody see this? I mean, this doesn't matter, but it's sort of just to give you a feeling for this.

Student: So as you tilt the plane, what is telling you to tilt it in the direction that actually follows the part?

Instructor (Stephen Boyd):I want to put a gravitational force in the friction minus. I'm going to put a force in the direction minus c.

Student: So c is going to determine how that?

Instructor (Stephen Boyd):C will tell you how to tilt the table, yeah. So you tilt it along c. Okay? So that's. Okay.

Now we're ready to tell you about the barrier methods – so the barrier method is this. By the way, the other name for this – there's lots of names for this. Sount, I already told you one. Definitely you should try to use this, because it's irritating to people who work on these things. And that's always good, so you should try to use this term. And you should also drop things like saying, oh wasn't all that known in 1969? And I'll tell you a story about this shortly.

Okay, so all right. So here it is. It's the barrier method, otherwise known as the Sount method. It's got other names too. It works like this – ready? Now, you start with a strictly feasible x. We'll talk about how to get one later. And some additional value of t, which is positive and some parameter mu, and some tolerance. And here's how it works.

You simply do centering steps so you can compute x*(t) by minimizing this. And this could be using Newton's method. I mean, it doesn't really matter, but if you want a complexity method, and if you want this method to work, well, you're going to have to use something like Newton's method here, okay?

So you do this and you will start from the previous value. The previous x* found. Then you update x. Now, when you've x r(t). Of course you're duality gap is exactly m/t. If m/t is less than your tolerance you quit – and by the way, you would return. When you quit, you could actually return actually both x *(t) and lambda * of t. In effect, it's considered polite if you write a solver in convex optimization – at least, it's considered polite to not just return a nearly primal optimal point, but to also return a dual, nearly dual optimal point with a duality gap that's less than some number, right?

Because that way, in fact, it's anyone can check. They don't have to trust you because – in fact if you've asked for a tolerance of epsilon, they return a primal dual point. None of your business how they calculated x lambda nu. Totally irrelevant. You just check the duality gap. And so they're actually returning essentially the suboptimal point and the certificate proving its epsilon suboptimal, okay? So you could do that here.

And then you increase t. And you would multiply t by mu. Now, let me just say a little bit about this. So the stopping criterion of course is completely noneuristic here, right? Because, in fact, it returns not just a suboptimal point with some vague idea that it might be optimal if some unknown parameter was small enough for something like that, which is how most of these method works. Instead, it returns not just the a suboptimal point, but a certificate proving certifying its epsilon sub-optimality.

And here you should imagine a tradeoff in the choice of mu. So the tradeoff would be something like this. Mu is how much you increase that parameter t in each step. You should imagine that things like 1.05. If you have 10.05, here's what should happen. Let's imagine you use Newton's method. Because important to get the tuition money. If you use Newton's method, if mu was 1.05 it should take maybe only two Newton steps to recalculate the new center, because basically you're at some point – well, let me draw a picture here. Basically here's what happens in – here's your central path. You're right here. You've calculated that point. You crank up t by 5 percent, and you want to calculate this, but you've just minimized the problem that was very, very similar. The truth is, where you're starting is nearly optimal anyway. And so basically in two steps of Newton's method you should be right back here, okay?

Now the problem is you only made 5 percent reduction in duality gap, because the duality gap is exactly m/t. If t went up by 1.05, your duality gap went down by 1.05. Of course,

that means if you do it, I don't know, a hundred times it will work pretty well, or something like that.

So these are little mousy steps, and this is what you'd imagine. This is what a normal homotopy method is. A normal homotopy method you should imagine little mouse steps, because homotopy method says I have a parameter, I can solve this problem. I can't solve that problem, and what you do is you crank your parameter just a little tiny bit and you make each step. Not too hard. And you hope, you pray to the god of Newton, whoever controls Newton, whichever god's controlled Newton's method – things like that. [Inaudible] conversion, a couple of steps. That's a standard homotopy method.

However, it turns out here. It's going to turn out that the actual updates are very aggressive that can be used – oh by the way, this model of little mouse steps is exactly what the complexity theorists will tell you to do, so if you want. Later we're actually going to bound the total number of Newton's steps required and, in fact, the complexity theory will require us to take mouse steps there on the order of something like, you know, mu is going to be something like one plus and then one over d square root of the problem size, okay? That's what our complexity. And that choice of mu will optimize the overall bound on the number of Newton steps, okay?

In practice it turns out the values of mu can be very aggressive. They can be things like two, five, ten, fifty, depending on the problem. And these things will work very, very well. Shocking. That's really no homotopy step. These are just huge aggressive steps. So if you crank it up by ten, it means that you're going from this point. You're next point is going to be here. And you're next point will be even closer

So what happens then it might take you a bunch of Newton steps, so you're going to go way – you're going to have a whole bunch of Newton fun and you're going to land up there. Okay? So that's – this is not, you know, a normal homotopy, right, where normal homotopy is you just.

Okay, where I'll give you an example of a homotopy. How many people here use Spice? Well, so that's great. That just, for the record that was like no one here. Anyway, no it's a couple people.

Okay, to those people, so you know in Spice one of the things you have to do, you have a non-linear circuit and you want to calculate the bias, the equilibrium point – the bias thing. We have a set of horrible non-linear equations. I mean nobody, you know – you can't solve it. And you know already you can already have big trouble if you have, like, i-stable things like if it's got memory or stuff like that. Its multiple equilibrium points. But you want to calculate an equilibrium point. Here's the way you do it. So addressed to those people. Let's take an amplifier or something like that, and you want to know what the homotopy parameter is?

This is fantastic. It's totally cool. Anyone guess? Actually, I know a bunch – a lot of you people have been tortured with it, because a bunch of you people from EE. Now don't

hide, don't think you can hide. I know a lot of you are in EE. You're in stat, you're safe. That group over there, but the rest of you there's no way. You know you've seen this stuff.

All right, so the homotopy parameter is the supply voltage. Because take an amplifier and let's hit the supply voltage to zero. Anyone want to tell me what the voltage and currents in the circuit are? You know, thank you. That was easy. Okay.

And then I take the supply voltage, I'm ramping it up to – what's your favorite supply voltage?

Student: Fifteen volts.

Instructor (Stephen Boyd):Fifteen volts – see that? No, no, no, no. I know what he's talking about. That dates him. I mean, you're way too young for that. That's crazy. That's like I might say that, plus or minus 15 volts. I can say that, but why can you say that? You can't say that?

Student:[Inaudible] Smith.

Instructor (Stephen Boyd):Oh, Cendren Smith, because you read old stuff. Okay, fine. All right, no. Your choices are five – depends on your age, okay? Or I guess the age of people whose stuff you read. Okay, so it could be five, 3.3, two, down. Now, if you're really cool you could say like 1.1 or .8 if you're super cool now, right? See, I know people knew and just holding out on. All right. It doesn't matter. So what happens is this. We take a big circuit. We set the supply voltage to zero. Everyone agrees.

All voltages and currents are zero, right? Okay, now you set the supply voltage to 100 millivolts. All right. What happens if you take like a big, horrible digital logic circuit and sent the supply volts to 100 millivolts? By the way, how ell does it work as a logic circuit? It doesn't work at all. Okay? However, can I calculate the – can I calculate the bias point? You know this. You know this, right? Come on, you held back on me on the satellite stuff, even through you're from Harben and should know this, right?

You've had this circuit stuff? A little bit, okay, all right. So take a logic circuit. DDD's 100 millivolts. What does the circuit look like? Oh very, very sub-threshold, right? It looks like a resistor circuit. So we can solve it in one Newton step or two, right?

No problem. See, you get thee things because the transistors, they're not even on. I mean, it's ridiculous. They're not even transistors yet. They're resistors, okay?

Then you go to 200 millivolts, but you start from the one you just found and then you go to 300, 400, whatever and so on. And if you go – by the way, from one volt to 1.3 volts and it fails to converge, then you go, oh. I'll try 1.15. Everybody got the idea? Anyway, so that's a – there you go. That's a homotopy method. That's how Spice works, okay?

Student: Is there any downside to using a large mu, like an aggressive [inaudible] or something?

Instructor (Stephen Boyd):Here? Student;

Yeah.

Instructor (Stephen Boyd):Yeah. I'll tell you what the tradeoff is. In a circuit problem you're trying to find the equilibrium. The downside is real simple. If you too aggressively update VDD, Newton's method, which is what is used to calculate the equilibrium position will fail. It'll fail. It'll just fail to converge, okay? So that's the tradeoff there. For us, we cannot fail because we solve a convex problem with every step. We cannot fail.

You can crank mu up, t up to the ten to the nine, and at least the theory says we can't fail because we can minimize any smooth convex function with equality constraints, right? So we can't fail, but what can happen if you update t super aggressively. Remember our theory, we can tot out all our theorems and talk about numbers of steps and blah, blah, blah. What'll happen is the number of steps will be just absolutely huge. Actually both in theory and in practice. So that's the problem.

Student: We can [inaudible], so.

Instructor (Stephen Boyd):We as a – yeah, Mantle doesn't do anything. Let's remember that. Okay. They do nothing. Remember, LA Pack is doing the work. Don't forget that, okay. Yes.

Student: The fact that the Newton method rolls over at some point in the transition?

Instructor (Stephen Boyd): Yeah, exactly.

Student: Do we use that to choose mu?

Instructor (Stephen Boyd):That's very interesting. There are methods. There's whole methods that actually do exactly this barrier method. And they choose mu so small that the can prove that when you update t by that amount you're still in the region of quadratic convergence. So that's called a short step method. There are lots of them, and that's a lot of the proofs go that way. And basically it says you can prove that by cranking up t by the small amount, if you were in the region of quadratic, if you were in the region of quadratic convergence before, you will remain in it for the new value of t and so on. So that's one way.

No, I can tell you how we pick the update. Well, I can tell you the reasonable way to pick it. Actually how you will pick it. Ready? Mu equals two. There. Here, I'll pick another one. Mu equals ten. How about mu equals 30. These are sort of the numbers that work. There are much more complicated ways if you get into ultra high-end ones. By the way,

they don't do a whole lot better, honestly, than these simple things. That's the embarrassing part. So this whole list – you can get 50 papers written on how to update this parameter, right?

I mean, some of these are quite fancy and effective, at least for some problems. So this empirically – this appears to be the right, you know, anywhere between two and 50 seems to be the right number.

Okay. And there's lots of eristics for the t(0). By the way, this is something that's discussed in the book. You should actually read about that. T(0) actually implicitly is the original duality, your original duality gap. So that's what t(0). M/t(0) is your original duality gap. So if you had some reason to believe that your current point was suboptimal by five, ten, it says a reasonable value of t(0) is on the order of one over five m or something like that. That that would be a reasonable one. That would give you the same duality gap and you'd just center. Okay.

So let's do the convergence analysis. To show that it works is actually kind of silly. I mean, we know that, but we can say a lot of things. Now, the number of outer steps. By the way, the outer steps here are called – these are called centering steps, or outer iterations, or I don't know. Whatever it is, right?

So by the way, it should be clear that you don't need that this is, there's lots of stuff that you could save here. For example, when you do a centering step, you don't need to, like, polish off. You don't have to compute x*(t) to 16 significant figures. That's totally clear because, in fact, this whole thing is just a method to guide you toward a solution, right? So you don't have to do full centering.

But, I mean, with Newton's method it's cheap. It's a couple of Newton's steps, and the difference between getting a okay solution and an unbelievably accurate solution, an amazingly accurate solution is two or three steps in Newton's methods so you know, there's no reason to worry about it too much, but okay.

So there's a lot of stuff here you can imagine is not, can be shaped up. But for an algorithm that's very short to write it's amazing how well it works, okay. So how does this work? Well, the number of steps is – I mean, you can say the exact number. It's very simple, it's just log of – it's log of m over epsilon zero. Every time the duality gap, after ht first step will be exactly m/t(0).

Then each time you do an outer step that shrinks by a factor of mu. So you can immediately just work out the exact number. It's this. That's the number of steps it's going to take, and that's ceiling of these things.

Okay? So that's the number of outer steps period. Notice this has nothing to do with anything but t(0) and epsilon and m. Nothing else, okay. Now, the centering problem. Oh, and let's look at how this varies with mu. If mu is big, this number is kind of small. If mu is like a little mouse step, like 1.01, this number is big. Right? I mean, mu is 1.01, log mu

is .01, so basically that's 100 in the numerator here. Right? That's basically what this is. So what's that?

Student: Mu is [inaudible].

Instructor (Stephen Boyd):No. Log 1.01 is .01, kind of. Very close, right, so. Am I wrong? No. Okay, all right, so okay. Okay, that correctly reflects the variation in the number of outer steps is monotone decreasing in mu. So mu is your update aggression parameter, aggressiveness parameter. And that says that the more aggressive you are, the more duality gap reduction you get per step, so you take fewer outer steps.

Now the problem is, the more aggressive you are in your update, the harder the problems you're solving by Newton's method. So that's now – you can do this intuitively and just say therefore what we should do for our problem class is just try this out, adjust various, you know, solve giant families of problems.

Oh, by the way, Newton's method cannot fail. So we have a proof that Newton's method will converge and so on and so forth. And if you use the classical analysis, it would involve all sorts of constants that you don't know. But it doesn't matter. You would at least have, at that point, a conceptual proof of convergence, right? Which would be to say this method can't fail for something.

So that's fine. But, if you want to get – I mean, the modern results come by using self-concordance. So if f(0) and the log barrier are self-concordant, then tf(0) is self-concordant for t bigger than one. And tf(0) plus phi is self-concordant for t bigger than one.

And we'll get to that a bit later, but let's just look at some examples and see how this works. This is the dual of the problem you're solving on your homework, but it's an LP, with, you know, 100 inequalities and 50 variable. And this shows you the number of Newton steps. By the way, the computational effort is simply measured. The unit is the number of Newton steps period, so that's the cost is Newton steps here.

So the number of Newton steps here is plotted here, and this gives you your duality gap. Now – and so it's plotted with one of these staircase things, and the staircase thing. It says that the width of a stair is exactly the number of Newton steps required to carry out that centering.

The height of a stair is exactly the reduction in duality gap obtained with you finish re-centering. Now, in this method, the duality gap simply goes down by a factor mu, so since that's a log scale, the height of the steps for each of these plots is identical. And in fact it's exactly equal to log mu or something like that, or whatever it is. It's something. It's a factor of mu down.

So here's a little mouse steps here. They're not even that mousy. If you're doubling, but you can see everything. You can guess all sorts of things here. You asked about the

Newton quadratic convergence and stuff like that. You can sort of see down here. You're taking like one or two, probably one or two steps. I don't. Maybe these are two steps each, right? So basically you double mu, you down two Newton's steps, double mu, double t, sorry. Two Newton steps, double t. Do two Newton steps. That's this. Now, you're making pretty good progress down here, and you can see down here is what you get.

The weird part is here. This is mu equals 50. For mu equals 50, you see what's happening is the width of the treads are much wider, because you're taking more Newton steps to actually re-center.

But, of course, then the measure of progress actually is duality gap reduction per Newton step. That's really what you want, because when you start with a duality gap of 100 and you want to reduce your duality gap, which is your ignorance by a factor ten to the eight, then somehow you have to make progress. You have to reduce your ignorance by a factor of ten to the eight, and you have to do that in some kind of Newton steps or whatever.

So and you can see here a mu was 50. It's actually down. By the way, that number is actually like 30 something steps. That actually probably takes a little. This is probably a good time to stop before we get into all the horrible details of mu and t and self-concordance and all that. It's actually time to just take a moment and realize what this says.

By the way, you see this picture? It looks the same if there is a million variables. It looks the same if it's maximum entropy problem. It looks the same if this is a problem from finance. Signal processing, image processing, machine learning. They all look like that. They look just like this. Essentially no difference, okay? And it's completely independent the size of the problem. It just looks like this, okay?

Now this is really quite ridiculous because if mu equals 50, it means that with a total number on the order of 35 Newton steps you've actually solved an LP to extremely high accuracy using Newton steps. Actually before – let's step back and think about what that means. First of all, it's quite ridiculous. The feasible set is a polyhedron defined by 100 inequalities and 50 variables. How many vertices does such a polyhedron have? How many?

Student: Two to the fifty.

Instructor (Stephen Boyd):Two the fifty. Okay, sure, yeah, sure. That's fine. The answer is – by that you meant a lot, then I'll accept your answer. And that's a good answer. Two to the fifty. Okay. The answer is there's a lot of vertices on such a polyhedron. This is a baby, baby, baby problem. There are a lot of vertices on that polyhedron, right? Okay.

One of them generically is our optimal solution, okay? So what is insane is this method has done the following. Thirty-five times you stopped in the interior of the polyhedron,

asked for directions and were told to go in some direction. You then tried a step of one. If you didn't like that you stepped off to a half and then to a quarter.

So somehow after 35 times asking directions you went from somewhere either middle of this polyhedron in our fifty to a point, which is essentially optimal. Everybody see what I'm saying? So the whole thing is highly implausible. Thirty-five steps, that's smaller than the dimension of the problem if you think about it. I mean, that's 50, there's 50 orthodial directions you can go into and you ask for directions 35 times. Since this big figure looks the same when there's a million variables, then it gets just totally beyond anything you can possibly imagine.

And it looks the same, which is ridiculous. It means basically to solve a convex problem with a million variables like 30-odd times you stop and ask for directions. And after 30 such steps you have an extremely accurate solution. It's really quite ridiculous when you about it from that point of view, so.

Student: How would this compare to using [inaudible] method?

Instructor (Stephen Boyd):Simplex method would also be a very, very fast on this. So simplex method is – I mean, it's nothing we're going to talk about, but simplex method is when you actually go along the outside of the polyhedron and you take an edge greenly. That actually works also stunningly well, so.

Student: So for each [inaudible] mu of, or mu squared? Instruction:

Oh, the cost per Newton step? Cost per Newton step depends on the problem, right? So in this case what's the cost per Newton step for this one? You have 50 variables, 100 inequalities so you have to. Yeah, it's something like n cubed. But in fact what you'll actually find, believe it or not, is forming the Hessian. It's more expensive. So it would be 100 times 50 squared.

Yeah, so that's what it would be. So the cost of a Newton step is 100 times 50 squared. I don't want – does someone want to calculate that? I guess – is Jung here or someone? Does someone want to calculate what the cost is? How fast can you do? Let's figure out how long this actually would have gone down. Let's say written in c solving LA pack directly.

So it's like 50, it's 100 times 50 squared. Let's just say that's what the complexity is. Should we just round it up and make it like 100 cubed? Sure, let's make it 100 cubed. That's four times mu d. That's okay. Let's make it 100 cubed. We'll round up, so 100 cubed. How long does it tae to solve an equation with 100 linear equation, 100 variables. I keep coming back to this, and I'm going to keep coming back until people start answering the right way.

Student:[Inaudible] millisecond.

Instructor (Stephen Boyd):Thank you. Millisecond. Okay. Sub-millisecond, okay? So how long did it take to solve this LP? Thirty milliseconds, okay? So by the way, it's not a lot of people who appreciate this fact because, well, why not? I'll go off on a weird tangent and why not?

Okay, so basically, you know, linear program has been around since 1948 or whatever, you know, and it's usually associated. People imagine – I mean, if you close you eyes you picture like a Cray computer and some guys wearing, like, white lab coats – and you know, its machine that costs a million. It's an expensive thing, and you imagine you would use it for, I don't know, scheduling the airlines tomorrow, pricing a derivative, designing a nuclear weapon. You know, you think of these big iron – you know? You know what I'm talking about?

That's kind of what you think. That's how a lot people think if linear programming, especially like if you're an older professor or something like that, and that's when you learn about all this stuff, right? So that's your picture of what solving an LP is, right?

And it's cool. You know, that's how United schedules their airlines tomorrow. In fact, that's how they do it. I mean, roughly, but okay. And, in fact, yes, that's how you optimize portfolios and things like that.

But I think what happened since then is, well, let's say Moore's law computers, I don't know, ten to the eighth times faster or something, and there's also been algorhythmic improvements. So the idea that you can solve an LP of this size in milliseconds I think really hasn't hit people yet. People don't know it. People who do control don't know it, for example. They sure don't know it. They still think you do, like, you can carry out four floating-point operations like per control calculation. They don't know it.

And people do this stuff like over in our own department here, they don't think about doing things like this. So this is important for embedded applications. So all right, enough of that. We'll move on. That was just a big aside. Okay, yeah.

Student: So what caused the mu to 150 in case it takes longer. Was that just because of all the haphazard?

Instructor (Stephen Boyd):You got it. No, no, no, no. The tradeoff – it makes perfect sense. If mu is too small, what's happening is this. You're making not enough progress each centering step. The centering steps are easy, but you're not decreasing your gap fast enough.

When mu is too large, you've bitten off more than Newton can chew. And so what's happening now is when you finish the centering step you've made a lot of reduction in gap, but what's happened is you're taking a whole lot of Newton steps.

Now the cool part is that this parameter. These two trade off each other, and you get this huge flat thing like this. It's not common to be something like this here. This flat – but

the point is, there's a big choice of – this is not a parameter that needs to be tuned, let's put it that way. It's usually set and then forgotten. Another question?

Student:[Inaudible] to changing it over the problems?

Instructor (Stephen Boyd):Oh yeah, absolutely. Yeah, I know, you keep coming back. No, the truth is I'm showing and you will, in fact, code before next Tuesday and amateur version of this. It will be a very small number of lines.

No, fancy ones don't update by fixed parameter mu. You're exactly right. They actually have a very complicated way that, by the way, for LPs and QPs worked amazingly well. It's called like a marrow trip predictor corrector thing, but the shock is that something this stupid works like really quite decently.

So yes, sophisticated methods will actually adjust, will crank up t by some adaptive scheme. So yes, that's correct. There was another question. Do you have one?

Student:I was just going to ask [inaudible].

Instructor (Stephen Boyd):About what?

Student: About a pick? Like a small processor.

Instructor (Stephen Boyd):On a pick. I don't know – but, I mean, it wouldn't. I think it would be pretty. I don't see why we couldn't do it. I mean all you have to do is multiply the entire thing is that you have to multiply a matrix which is 100 by 50 by its transpose.

Student: Right.

Instructor (Stephen Boyd):Or all you really have to do is a QR on it, that's all, so yeah. You could put this on a , you know, arm or a pick or I don't know. I guess. I don't know that people have done that. That would be cool. Someone could do that next quarter as a project.

But some interior point thing on an arm processor or something weird like that for no reason, just because it would be cool. I don't know, okay. All right. So here's a geometric program, a GP. So it looks the same. Same story. And it's just the same. And then here's sort of what happens if you have a family of standard LPs, and then what's happening here is you're changing the size. So this -ism, what is this? Oh hey, this is your homework problem. What do you know?

Oh that's interesting, because you know what? Since all the code – it means Google will find the solution to your homework, but it doesn't matter. We all know that anyway.

Anyway, isn't this what you're doing? I think so, so yeah – fine, so. If you did it right here's what will happen. You'll make something that looks like this, and then this is the

most amazing thing that happens. It's this. Here's the number of, what is m? M is the – what is m? Oh, a is a, r, m, by two m. So there's two, this is the number of constraints, the number of variables is twice this number. Okay, so that's 20 variables, ten constraints. One hundred variables, two hundred constraints.

Did I say that right? Yeah, I reversed those. Anyway, you get the idea. That's 2,000 variables, 1,000 constraints, and here you generate a whole bunch of random problems and then solve them. And this is the number of steps required to reduce the duality gap probably by ten to the minus six or something, or reduce it by a factor of ten to the eight.

When you look at this, you see something amazing, and that is that it's basically it's not growing. It just doesn't grow so and I can tell you a little bit. I'll [inaudible] story about this. I guess when these methods were first looked at, we'll see very soon. I mean, next lecture well see that the theoretical computational complexity says that the total number of Newton steps is less than the square root of m, where that's the number of constraints.

Okay? And by the way, with a number out front, that's just like a number. It's not one of these mysterious, useless convergence proofs that has, like, Lipchitz constant and all sorts of crap that you don't know. It's just a number, so it's going to be some number in front time square root m. That's the number of steps it takes, period, for these things.

Now, if you're a complexity theorists that you'd celebrate this because you'd say well, there you go. You see, it grows like the square root the problem size. Each step takes, you know, in the simplest case, you know, n cubed or something. And so you'd have an order n to the 3.5 , and you'd have a polynomial time party or something. A big celebration or something, right? And walk away.

Now what happened with that? And by the way, that's where the theorists – that's where they are right now. The complexity theorists will tell you it's the number of iterations grows like the square root, but after people see this on and on, and over and over again, actually somebody finally, timidly said no, it clearly grows less than square root m. And somebody finally said, I don't know, the fourth root. And then somebody said the log and finally. And what's become completely clear is that it grows like this, o of one.

Okay, that's become as an empirical fact that would appear to be the case. And in fact, to be much more specific, it appears to be between 20 and 80 steps for all problems. So let that sink in for a second because it's quite ridiculous, and it's actually the essence – it's why these interior point methods. And we'll look into them, and we'll look into centering and duality gap and complexity and all that, but I don't want the main message to be lost in all of that.

The main message is this. They take between 20 and 80 steps, the problem in finance, machine learning – doesn't make any difference. The size doesn't make any difference. It's about between 20 and 80 steps, and I think I may have told you this earlier. Maybe on the first day I told you this, but now you know more about it. We had a talk last year by a guy named Jack Ganzio from Kenbrook, and he came and solved some giant finance

problem with one billion variables, okay? It was dense, by the way. No, seriously. It was some big tree. It had some structure, but this was not a small problem, let me say that. It wasn't super sparse or anything like that. It had a billion variables.

So he solved it and we were like, how do you do that? He had some room where they had a blue jean – you know, they have the whole room full of like thousands of like processors and all this kind of stuff. And I said what algorithm do you use? And he goes, the same one everyone uses. And by the way, not quite this. He used an adaptive steps size, but it's not a whole lot different. It's basically this.

Basically what you're going to write. Its a billion variable problem, and I said well, how did you pick the parameter value? He said, just the same everyone uses. You know, it doesn't matter anyway.

And I said how many steps did it take? He said 21. So basically, that's a single point. Oh no, I think I mentioned this earlier. It took 21 iterations. Now, by the way, each iteration took like several hours and the lights in Edinburgh dimmed when each iteration was carried out.

Well, I mean, it's solving a billion variables with, you know, a dense system blah, blah, blah. But the point is, in some giant room full of computers, right, with a big, you know, 12 kilovolt three-phase thing going in for awhile – but the point was this thing, at least we have one point layout here at ten to the nine provided by our friend, and – oh, well, this is using our unsophisticated methods. The sophisticated methods are down here at the 20 level, although that hardly seems to mater, okay? So question?

Student: This technique, like he's running in a roomful of computers, did this scale in [inaudible].

Instructor (Stephen Boyd):No, his was not a fully dense. His had structure – in fact, it was an event tree. He was doing stochastic programming for some finance problem or something like that. Expanded the event tree out by like 15 steps or something. That'll give you a billion variables.

So the question is how parallel. Let's talk about if you profile one of these what – in fact, if you run an interior point method. In fact, the one you're writing right now. What actually will it be doing? I mean, yours will actually be spending a lot more time doing interpreted overhead than actually doing any actual computing. But if you scaled yours up big enough, you do 20 steps, what are you actually doing each step? What is it?

Student: Linear systems.

Instructor (Stephen Boyd):You're solving linear systems, you're solving a Newton. You're computing a Newton step. Line search and stuff like that. It doesn't cost anything so, in fact, the correct thing to say is if you want to solve a convex optimization problem

you're going to solve 20 liner equations. All right, 80, whatever, you know. Something between 20 and 80 liner equations, period. That's all you're doing.

And you can use any method. I mean, we've just looked at the most simple thing, right? Where you use a Chilesky or something. I mean, if you see something, if there's some structure right under our nose like banded or something like that, of course you should exploit it, but then there's this whole world of solving like huge linear equations that we haven't even touched on, but exist and can be applied, so.

Okay anymore questions about that? That was just like a weird, I don't know, detour. So okay. Let's talk about feasibility in phase one methods. So in fact, you'll have to do this. So the feasibly problem is you want to find an x for which fi(x) is less than zero. Well, I tell you the truth, is you really want to find something like that. If you want to initialize it in a simple interior point method like the one we have. So this is what you want.

So there's lots of ways to do this. They're actually all kind of cool. The standard tricks, these go back to 1948 or something like that. I mean, a standard trick is to solve phase one by solving a similar problem. So here's what you do. You solve this problem. This is the most classic one. You have a new variable s, and you say fi(x) is less than s here. Ax = b, and you minimize s here in this problem. That's a convex problem, too, however, for this problem I can always find a strictly feasible point.

Well, sorry, provided I know any x that's in the domain of f and satisfies Ax = b. I choose whatever x I like, okay? Not feasible for the original problem. I mean, a lot of the fi's are way bigger than zero. And then I just find the max of the si's and I take s equal to that max plus one, and that's now strictly feasible. Strictly feasible point. I applied a barrier method or Sount to this thing, and then I can put in a logic that says this, if s ever gets negative, just quit because that's good enough. You've actually produced.

And then you would drop to phase two. Phase two would start from there and would actually then optimize the original objective, which would be f(0). F(0) is not in here. Everybody see this? That's the idea.

So this is the standard. Now, if you're minimizing this problems and you minimize it, and s* is positive, you actually now have a proof that the original problem is infeasible, and in fact, the dual, by taking the dual point found from s* here by solving this problem actually says certificate establishing that the original system of inequality is infeasible.

Okay? And there's one thing where the results are ambiguous, and that's when p* is zero, okay? So that's when you're right on the boundary. It says that the problem is, let's see, it means it's feasible. Well, it could be all sorts of sick things could happen. It means fum could be feasible, but not strictly feasible, and so on and so forth. Okay.

Now, by the way, let's talk about how you imagine this is going to work. So let's do that. If the problem is feasible this will work perfectly. S will keep going down. At one point, s will become negative and you terminate. You put logic in there to terminate, so you call

your barrier method with a strain that says phase one, and it knows then stop, not when your duality gap is low, but stop whenever s is negative and quit right there.

However, suppose it's infeasible and let's se what happens if something is infeasible. These are fi(x) here, and let's say I start with a problem like this. I don't know why I'm drawing them up like that. I don't know what that was, sorry. Do it again, all right.

So here's zero, our critical point. Here's fi(x), here's a bunch of inequalities that are satisfied like that, and here's a bunch that are violated, okay? Now, this phase one method puts s over here. It puts a bound above these guys. That's s, and it starts pushing this thing to the left, right? Okay, yeah. I was checking if that was the left, yeah. That's pathetic, isn't it?

Oh well, all right, so you start pushing this to the left, and of course if you ever push to zero, then they're all on the left. You exit and you jump to phase two, okay. Now what do you suppose is meant from infeasible. You're going to push and push and push and you're going to stop because you can't find a pint that has all the fi's negative.

When you stop, what do you imagine is going to happen? Suppose when you stop, s* is one. What do you think this plot is going to look like?

Student:A bunch of functions [inaudible].

Instructor (Stephen Boyd):You got it. A whole bunch of constraints are going to be violated by one. Okay? So a lot of constraints are going to be violated, because when you start pushing points like this, right, they'll pile up and an bunch of them will pile up and you'll keep pushing and it'll get harder. And then you'll come up against a wall and you stop.

Oh, by the way, what's the force that you have to push with? The Lagrange multipliers, right? So some of the Lagrange multipliers. Mechanically that's exactly what it is. So you push and push and push and push then you hit the wall. You hit the wall and a whole bunch of things are violated, okay?

Well, you answered your question. You're problem is not feasible, but there are other very interesting variations on this, and I'll talk about one and then we'll move – we'll quit, actually. Let's see. So the sum of infeasibilities method works like this - I introduce a private. Oh by the way, you can think of s as sort of like a little, the interpretation of s here is, you're looking for a point where fi(x) is less than zero. That's what you're looking for. So you say well look, I'll give you a little extra slack, a little extra bonus, you know. You don't have to do zero, you can make it less than seven, and then seven gets pushed down, okay?

So here though, everyone gets their own private little slack thing. By the way, if we've seen this before. This is the beginning of the story on the support vector machine, okay? Actually, it's the beginning of the story on a lot of things.

So what you do is this, and everybody gets this own slack. If this slack is zero, it means that inequality is happy. It's satisfied, and you want to minimize the sum of these infeasibilities. Now, already you should have all the neuroconnections before you even do this or anything like that you should have a feeling for what's going to happen when this is infeasible.

What's going to happen in this case is this is an eristic for solving the problem. Please find me a point where that violates as few of the inequalities as possible. Okay? This is an eristic for it. This is how the support vector machine works. That's how all these things work. Okay?

When you solve this problem, here's what happens. When you solve this, and if you have 100 linear inequalities and 50 variables infeasible, and this is what happens when you. I just plotted backwards I guess, but these are supposed to be positive. When you push these up here, here's what happens. You get to the point and basically most inequalities are violated. So this is the basic phase one. We use some of the infeasibility, there's only 21 in the end that are violated. Is 21 the minimum number of constraints that can be violated? Who knows? No one knows, but if the eristic for violating just a few.

So this would have lots and lots of applications. If you're doing engineering design, for example, and you have a whole bunch of inequalities and it's infeasible of course the main message to tell somebody which is useful is sorry, but your specs can't be met. That's the most important thing.

But after that, someone says gee, thanks for the bad news, but can you tell me which ones I should relax, you know, which constraints do I have to go back to somebody, the specificer and actually beg and grovel for more from, right? That's what basically happens.

So in a normal phase one, you know, everything is violated. You don't really have an idea of which were the ones that were more violating or something like that. In this case you would actually, in this case you'd actually go back and zoom in on one step. I mean, you don't know this is the minimum sets of violations, but it gives you much, much more information, so okay. We will quit here.

[End of Audio]

Duration: 77 minutes