ConvexOptimizationI-Lecture15

Instructor (Stephen Boyd):Great! So today we're going to start in, I guess for real, on this third part of the course, which is on algorithms for solving complex problems. The last topic, which was numerical linear algebra, which is sort of, it's just a sort of background for it. It's actually what – everything's going to be built on top of that.

Okay. So today, we'll start in on unconstrained minimization. I guess this is on your current homework as well. And actually, I'll make a couple comments about your current homework. I don't mentioning that it wouldn't take you very long, a couple of clicks at most, to actually find source code to solve the kind of problems you're looking at. I mean, particularly just, you know, for the book we have essentially all the source code online that we use. So I recommend there, so I mean you probably knew this, but I'd recommend there, actually, the code you're asked to write for the current homework is, you know, not long, ten lines, maybe. I mean conceptually it's not a big deal. It's not exactly like a heavy assignment. You should try it yourself first, just to see, but if you get stuck or something like that you type the right things into Google and two clicks away and there's all the source you want. It does pretty much what you want. Of course, actually, then you have to figure out how to adapt it to your problem and all that, but if you want to see the structure of it. So I just mention this if you get stuck with anything there you can always look there to see how to do this. You shouldn't get stuck, though. Okay. That's just a small comment about the homework. Let's look at unconstrained minimization. So here, the problem is very, very simple. Just want to minimize a non quadratic convex function smooth, that's minimized F of X. Now, of course, minimizing a quadratic is quite simple. Minimizing a quadratic you can do by solving a linear equation, because the gradient of F is F line, and so you set that equal to zero, that's a linear equation. So we're interested in minimizing non quadratic convex functions smooth.

Well, by the way, lots of methods for this, we're going to look at methods that produce a sequence of points, there're all in a domain, and the function values are going to go to this algorithm value P STAR. By the way, in the case when P STAR is minus infinity, that means this is unbounded below, you also call this a minimizing sequence. So here, I mean, in that case, of course, the Xs don't converge. Okay. And of course, to minimize a smooth convex function is the same as solving the equation, which is the degrading sequel to zero. So another way to think of this is you want to solve this set of N, non linear equations and invariables. By the way, these equations would be linear equations if F were quadratic. So this is a method for solving these systems. That's another way to think of it. Okay. Now, a couple of technical things, I will not go into too much detail about the technical stuff because in the book it's pretty, well, what's written there is actually true. But there are far more technical descriptions than that, but I don't believe they're needed. Okay.

So we'll assume the following, that we have a starting point because you have to have a point in the domain of the function. And we're going to assume that the sublevel set is closed. Okay. Now, these are sort of, I mean, mostly technical conditions. There are a

few real problems where you would have things like this wrong unless you're very, very careful about it, one would be empetre, and we'll get to those later. Now, when all sublevel sets are easy to work out, but that's basically the same as saying that the Optigraph is closed and some people just call FN a closed function. That's standard notation in convex analysis to call a closed function if its Optigraph is closed. And this would be true, for example, the domain is everything, so that's one case. And another case is this, if F goes to infinity, as you approach the boundary of the domain, this has another name, it's called a barrier, but this is another condition that does the trick. Okay. So for example, one over X is going to be closed, defined on X positive, that's closed because as you approach the boundary of the domain, which is zero, from, of course, positive X, the function goes to infinity, so that's one.

Okay. So here's some examples. This one is easy, along side X of a bunch of F line functions, that's closed because, in fact, the domain of this is everything. So that's automatically closed. This function is much more interesting to understand why this has a closed sublevel sets. First of all, the domain is not everything. The domain is an open polyhedron here. So if the sets of point X for which AI transpose X is strictly less than BI. So the domain is an open polyhedron here. By the way, the condition that we know in X zero in dome F, is not trivial, in this case. Right? Because, in fact, to find such a point, you'd have to determine if a certain polyhedron has non empty interior and produce a point in it. So some of these assumptions here are not, I mean, they can have real implications, as is the case here. Now, in this case, if you approach the boundary it means you approached the boundary of the polyhedron. It means that one of these terms BI minus transpose X goes to zero. If that happens, the associated log term goes to minus infinity and with a minus sign it goes to up plus infinity. So this satisfies the second condition for sure and, therefore, this has closed sublevel sets. Okay.

By the way, it's interesting because when someone says could you minimize this, please, you could get very confused. And you could say, I can't do that yet, I'll be able to do it next week when we look at how to solve problems with constraints. You say that's not an unconstrained problem. And someone else would say, well, why is not unconstrained. You'd say because you have the constraint AI transpose X has to be less than BI. These are constraints, there's no way around that. And yet, minimizing this we're gonna discuss it now and it's going to work just fine. Okay. And the reason is – well, we'll see why. So in fact, minimizing this is an unconstrained problem even though you might imagine it has constraints. These constraints will be taken care of automatically. And if you want a rough idea now of why this appears to have constraints but, in fact, can be solved, considered as and solved by unconstrained optimization is – the reason is this: A problem like this, the constraints could never possibly be active. So, in a sense, they're really not there. That's the sense in which they're not there. You certainly couldn't minimize this and find out that AI transpose X equals BI was the essence of a constraint, for example, a linear program or anything else, is that when you solve it some of the constraints will be active. So that's the difference between here and there. But this requires actually some care and thinking so you know which is which. Okay.

So we're going to make some very strong assumptions and I should say something about that. In fact, I should say something about how this all works. So a traditional treatment of all this material is you work out various proofs of conversance of things like this. That's fine, just get a rough idea. It turns out these things are not that useful in practice. And in fact, the typical conversance will look like this. It will say if, and it'll have a long list of things, by the way, not one of which is actually generally speaking verifiable. Right. Not only that, it'll involved all sorts of constants that you don't know. We'll get to some of those constants soon, like [inaudible] constants and things like that. It will say then, and it'll have a statement that says, you know, X converges to the optimal blah, blah, blah, within and it will be some number, usually they don't even tell you the number it is, there's no bound number of sets. If there is a bound number of sets it involves all sorts of constants you don't know. Okay. So from one point of view, one could argue that such a theorem is useless in practice, and I believe that's actually a correct statement. I believe that's the case. So when you then confront people who spend, let's say, four weeks of a class, and let's say, quotes in that material, they'll something like this, they'll say, no, no, no we do that because it's a conceptually is full, and that, actually, I agree with. So it's useful to know. It's great to know something converges. Even if, when someone says, how long does it take to converge, the answer involves all sorts of numbers you don't know. So it's kind of like at least it's, I don't know, it's a nice thing to know that if you ran it forever, that it would eventually converge. That's a useful thing.

Now, in view of this being, what I believe, and a lot of other people also believe, is the role of all of these conversion theorems, it means really, since you're not gonna be really using – you might as well use a big hammer and just make the biggest assumptions you can, because these are just conceptual, in fact. Saying that is sort of the idea here. So we're gonna take basically, we gonna make the strongest possible assumptions, basically to shorten the proofs, that's why. Okay. So in that spirit, we're gonna take strong convexity. Now, strong convexity means this, it means that there's a minimal curvature, a positive minimum curvature. So we'll assume that in this case. By the way, for lots of functions that you actually have come up all the time this is false, you know. I mean, one of your answers are perfectly good example that obviously doesn't satisfy this, the curvature goes to zero. But still, we'll take this and we do just to simply the proofs because there's no reason not to, in my opinion. So you take this. This means it's a minimum curvature. Now, there's a couple of implications I'm not going to derive these, although they're quite simple, in this case, this is what it means to be convex. This says, if you look at the function at another point it's bigger or equal to, well look, this thing over here is nothing but the first order Taylor approximation and it says, basically, the function is above that.

If you add this thing here, it's actually very – in fact, it's not hard at all to prove, you write up the second order term here, and use the mean value theorem, or whatever they call it, and plug in some value in this and you'll get this. And this says, you're bigger than or equal to the first-order Taylor series, this has a minimum curvature away from it. This says, when you move away, it's not that there's a linear thing and you're above that linear thing, it says there's actually a quadratic. This is a quadratic here in Y. And it

basically says, there's a quadratic that you're above. So it's a strong assumption. Which that implies all sorts of things. It implies that the sublevel set it bounded. It also implies, although it's a couple lines to derive at the following, it says that if you have a point X then F of X minus P STAR, that's the minimum value of F, it's less than one over two M times the norm of the great of F squared. Okay.

This strengthens all sorts of things. You know the optimality condition is the gradient N value of zero and that, of course, is completely consistent of this. This is now a quantitative one. And this thing here, gives you a stopping criteria. And it basically says, when the norm of the gradient is small you can stop. How small does it have to be? It depends on whatever two M. Then you might ask the question, do you generally know M? And the answer is no, you do not. So the use of this is sort of it's a bit conceptual. It means that when someone pokes into your see source and finds the stopping criterion, which is, you know, basically, if the norm of the gradient is less than some number, they say, how do you know you're close to the optimum? You'd trot out this inequality. And they'd say, yeah, but I looked at your code and I didn't see any M. And how do you know M? And that's when, I don't know, you get even more vague. And you say, well, I just made a number so small that depending on what M – anyway, you try to weasel out of it. You really can't. But the idea is that justifies this. Okay. So we'll look at descent methods after this. Actually, there's lots of descent methods, some of those interesting ones are not descent methods, but maybe, the most widely used ones for smooth problems are all descent methods. And let's look at what they are. It looks like this. And this is following now, maybe, 50 or 100, at least 50, so maybe, 70/80 years of sort of tradition and things like that.

It goes like this, you're next iterate is going to be the current point plus a direction, well, a vector, a displacement times the scale factor. Now, these are universally come to be called, I guess people call this the search direction. Now, normally if someone says a direction, they mean a normalized vector, like a unit vector, because the direction doesn't have a link associated with it. But these are generally not normalized. And then they call P the step lengths, even though it wouldn't be the length unless this thing was normalized in length, but still, that's what people call them. So that's what we'll call them. And it's generally divided into two steps. The general step goes like this. You have a point X, you calculate at that point a direction to move in, and then you calculate, it's called a line search but, in fact, it's really more a ray search, because you only look in positive P. You find a positive step size and then you update like this. Oh, by the way, usually in finding a descent direction, if the gradient of F is zero, then you terminate, right, and you don't generate. So when you determine this end direction part of that usually is something that checks the stopping criteria or something like that. So this is the general method. Now, these search directions have to make a negative enter product with the gradient. So, in another works, if you're function look like this, and the gradient looks like that, then it says that you're search direction has to be sort of in this open as space, over here. Now, by the way, when I draw this picture, there's one thing that your eye goes to immediately, and it just seems like how could you beat it. And that's the most obvious search direction there is, which is the negative gradient. We will get to the search direction in a minute.

I mean, that's sort of like why would you not go if the gradient is the direction in which locally the function is increasing as rapidly as possible, why on earth would you ever not go in the direction of the negative gradient, which is, after all, where the function is. It's the direction in which the function is decreasing as fast as possible. We'll see there's extremely good reasons for not doing this. This is often an exceedingly poor direction. But we'll get to that later. I mean, it's not a big surprise if you conceptualize it the right way. Okay, so that's in descent method. Now, in the line search you have to determine the step size and again, there's lots and lots of types of line searches. You know, hundreds and hundreds of articles, and books, and chapters, and it goes on, and on, and on. And it turns out, actually, what's kind of weird about it, is that for line searches it turns out, and it's something, actually, I frankly don't fully understand, but I accept it as an empirical thing, and that is that the actual line search matters much less than you'd ever imagine. Right. So here's some extremes. On an exact line search actually says, once you point on the direction and you have a ray, you actually sort of plot, you work out what F is along that way. In fact, for all practical purposes, you've reduced your problem to a one dimensional problem. One dimensional problems are easy. You plot it and use your eyeball. I mean, that's clear, right, it just goes like that. If it just keeps going down forever you found a direction of infinite descent.

If it points up, when T's positive it means you need to fire whoever wrote the code that generated the search direction. So basically, this is what an exact search would give you this here. And you'd think, well, look that's the point to minimizing F that's got to work well. But the other one – there are lots, and lots, and lots of line searches. But at the other extreme, there is something that is embarrassingly simple, and it goes like this, it's really, really dumb. You start with a step of one. Okay. And then, what happens is, you repeatedly multiple by a constant Beta. Beta's typically a half or something like that. So you try a step of one. And you try it. Then you check, this condition here is sometimes called the Armeho or the – I think it should be called the Armeho Goldstein, or really the Goldstein, since it is from Moscow, and it's almost certain where this came from. So this condition says this, it says when you evaluate F of X plus T Delta X, and let's actually look at a few things. Let's put Alpha equal to one. This is the derivative, is a directional derivative of the function along, in fact, you see this function right here where I erased the Alpha? This is this. And actually, by convexity, this condition here will never happen. It cannot. That's the point. This thing here is a lower bound on that. So you'll never get a decrease as good as that.

So for example, if this number here is like, you know, .6 and you try a step size of T equals one, this predicts that F will go down by .6. But it cannot go down by .6. IT can go down by .599 or something like that. It cannot go down by .601, that's for sure. Okay. So you have to degrade how much reduction you want. And in fact, what this does, is you multiple by Alpha, which is between zero and a half. We'll see why a half later. But you multiple by this, which says that you're going to accept some fraction of the predicted descent reduction and that's this. And what you do now, is you start at some T. You multiple by half, like this, until you lie below this dash one. Now, this is incredibly crude and with things like Beta equal .1, it's embarrassingly crude. It basically says, try step length of one. If that doesn't work, try .1, try .01, try .001. I mean, you can hardly said to

be examining a variety of step lengths. Now, you'd imagine that, like for example, an exact line search would work way better than a line search where Beta equals .1. You can only do like three or four steps of, you know, five or ten steps of a line search of Beta equals .1, because after ten steps you're looking at ten to the minus ten steps. At that point, you might as well quit. Here's the shocking part, actually. Exact line search works a little bit better, often sometimes, in many cases, it doesn't work any better, at all. And in fact, it seems to be completely almost independent of these parameters Alpha and Beta. You can take, I don't know, Beta equals a half, Beta equals .8, .1, and it, generally speaking, doesn't make a whole lot of difference. So it's a very odd idea. But it's born out by lots and lots of empirical evidence.

So this is called backtracking, this method, and that's the idea. And what happens is, you get the first point, when you multiple by Beta, that lies below this critical value T zero. T zero is the first point where you get equality in this thing. It's where this line intersects the actually function there. Okay. So that's the picture. These are line search types. Okay. I should actually mention one important thing about line searches since you will be implementing one. By the way, I mean, as I said earlier, we're talking like six lines, if you put a lot of comments in and make this as fervors as you can. We're not talking a complicated thing. One thing very important is this, is F can have a domain that looks like that. Okay. Here. And if this is T equals one here, I mean, this is kind of obvious. By the way, this thing works perfectly provided your function returns the right thing. If you evaluate this thing and you're out of domain, if it returns plus infinity everything is cool, because – then this in equality is interpreted as false, because there's no way infinity is going to be less than anything. Right?

Now, on the hand, if you write slightly more sophisticated code, you can write one where if you ask for what F of X plus C Delta X and this is out of domain, it will actually pass back an OOD token, which means out of domain. In which case you'd have to write some logic that fixes that. Now, unfortunately, in many cases that's not going to happen. The worse part is your F function actually will, although you're outside the domain, that has to be added there special – I mean, for example, you take one over X and we're only looking at X positive. Okay. I can check if one over X – if I step out of the domain for one over X, that's this thing right here, what would happen is, it'll make sense, it'll be a negative number. It'll be a negative number, and everything's going to work, and you'll never recover from this. So this is a hint for yourselves, though. That's all right. I said more than enough on that. Okay. So now, we get to gradient descent method. It is the most obvious method, is the first thing anybody would think of. I would hope it's the first thing anyone would think of if you want to minimize a function. And it just says this, here is it, it says take the negative gradient direction. By the way, this is a classic, greedy algorithm in the computer science sense, because basically it says you want to achieve a goal, which is to minimize F, and it says, instead of considering, you know, what's going to happen in the future, you simply look at that iteration and make the maximum progress towards your goal. This is literally the greedy algorithm for minimization.

Now, once you put in that context, you can imagine now, it might not be a good idea because some greedy algorithms work, some work well, in other words, they give you

actual algorithms that work very, very well, as well as any other. And some, of course, don't work at all. So we'll say this is closer to not working at all. It works in a theoretical sense. There's what you do, you chose a negative gradient, you do a line search, and the stopping criterion would typically be that the gradient is small. So the exit criterion would be actually somewhere here checked in step one. So it turns out, I'm not going to go through the details, which is in the book, it's this. It turns out that you can show this converges, I mean, with exact line search backtracking, it just works, and not only that, it converges expedientially. Okay. So that means that your sub optimality, that's the difference between F and the optimal value of F, goes down by a factor less than one at each step. Okay. Now, that's a pretty good convergence, expediential convergence, and this would be – usually you say that's great. Now, it turns out, actually – and you'd think, well, look, it doesn't get any better than that. I have a simple algorithm, it's totally obvious, makes perfect – it's what anyone would do. You have expediential convergence. You would taut that, that would be like a big fancy thing. It turns out in fact, it can work absolutely terribly, but let's look and see how that works.

And this is really sort of, you know, the one non obvious fact. You know, there's a lot of non obvious facts, but not that many in this whole course. This is one of them. That it's not obvious that the first thing that you'd ever do, why it works poorly, or that it should work poorly. But let's take a look and see. So here's just a quadratic problem. Obviously, we know what the minimum is. The minimum is zero, that's clear. But let's just run the algorithm on it. Well, when I minimize this I can work out the gradient, I can actually work out exact line search, and so on, and you can, actually, just analytically work out what the points are, what the iterates are, so they look like this. Okay. And indeed, they converge to zero geometrically, exponentially, as we were told they must.

Now, the problem here is Gamma, which by the way is the condition number of the quadratic form here, that's exactly what Gamma is. So if Gamma's bigger than one it's the condition number, if Gamma's less than one, this is one over the condition number. Okay. One of over Gamma's the condition number. Then, it says that you go like, actually, the Kappa minus one over Kappa plus one. It means something like that. Now, if you're condition number is ten, that's fine. You multiply that .90 step. If you're condition number's 100, it means that you're only making 1 percent progress and if it's 10,000 or a million, it means you're making very, very poor progress at each step. So that's – And this is, by the way, this is not a bound. This was a bound. You can always hope when you have a bound that things can be better than the bound. And in this case, they're not better, they're right on bound. And if you actually look at the iterate you sort of see what's happening. What happens is this, you're here – by the way, this is for Gamma equals ten. That's a very small condition number. You're here, and so the direction, that's the gradient and it gives you the direction of most rapid local increase. It's the most uphill direction. You go in the most downhill direction, and notice that it's pointing kind of in the wrong place. So you go along here and then, of course, you minimize along that line, that when you minimize you're orthogonal to the gradient here, and that's here. So you actually overshot.

And so it's, actually, very useful to visualize this in 3D. So you imagine the function coming up, like this, and you sort of take a ball or whatever and you roll it, it rolls down, and actually, when it hits the minimum, the minimum, by the way, is not in the valley. Everyone here, you'd know how to do this, right. I guess if you're flying an airplane about this and looked down you know exactly what to do. It says, you go down the valley and then go down the valley, because that's what this is, right. Now, instead, that's – actually, when you're at the valley it turns out you're still decreasing going this way, right. So you end up doing this. Now, this is for Gamma equals ten. So when Gamma equals, you know, 100 or 10,000 or something like that you get very, very slow conversions. Okay. Here's a non quadratic example and you can see here – well, in this case, actually, the backtracking lines search worked quite, the exactly line search, I guess, worked quite well, but just to show you a rough idea how this looks. Let's look at some bigger problems to see how this scales. Here's a problem that looks like that, and you can look at it, and these are the two things, and there's just a few things here. Actually, the first thing that's actually interesting, note that like up until about, I don't know, 100 iterations the backtracking line search did better than exact line search. So that's weird.

Actually, the reason is you don't really want to be. You can see here that, actually, being sloppy in your line search here would be an advantage. In fact, if your line search here were foreshortened, then that should be very much in your favor. In other words, if you didn't go all the way to the minimum, it would actually be way, way better. Because you'd end up closer to sort of this access here. And then on your next step, you'd shoot right in. So the first thing here is you see that there's basically not a huge difference between the backtracking and the exact line search, which is weird. The other thing is you can see it's taking, you know, it's taking hundreds of iterations. Although, I guess, it's getting a reasonable accuracy. And you can estimate C from this. And the other thing you see is, you know, roughly, this is a, I mean, just very crudely, that's like a line. And a line here, since that's a log access here, means that you're reducing your error, or you sub optimality, by kind of, roughly, very roughly speaking, a fix fraction each step. And here you can see if you do, I don't know, let's do it this way. Let's say you do 100 steps and you reduce it by a factor of ten to the four, so a ten to the four by, I guess, you have to take logs or something like that to get the right number in 100 steps but I don't know what that is, it's, anyway, can anyone do the log real quick in their head? I guess not. You can figure it out.

But then you'd say something like, C is .999, or something. That's just probably like .999 or something like that. It's easy enough to figure out what it is. And by the way, this is referred to by linear conversions if it – because it means that each iteration you improve your sub optic. You multiple you sub optimality by roughly, you know, .99 or something like that, by some number. Okay. We will get to a little more about that. Now, generalization of that is the steepest descent method. The steepest descent method says, I have a norm and I want to minimize, I want to find the steepest direction or descent. So what you do is you simply, you look and you say I want to minimize the gradient transpose times V. So V is my direction I'm going to go into. That's the directional derivative along X plus TV. That's what this is XT equals zero. But obviously, this has to be normalized so V equals one. So this would be the search direction. Now, it's usually

un normalized, it's un normalized. You scale by the dual norm of the gradient. You don't have to worry too much about this. So the steepest descent is just basically so you know what it is and all that. But it's not going to be key because we're going to focus on something else. Okay.

Now, the interpretation here is that in this norm you get this tells you sort of the direction where for a unit step in that direction the first order prediction of decrease is maximized. And actually, it's a weird thing to think that it actually depends on the norm. And that's actually the key to understanding everything. You would imagine if you're on some sort of surface and you asked someone which way is the fast way to go downhill. Everyone would point in the same direction. Well, it turns out that's – I mean, obviously, if the norm's agreed upon, everybody does point in the same direction. It depends on the norm, that which is very weird. So if someone's using like a L. Because the difference is this, when you say fastest decrease of direction, you really mean, decrease in F divided by say the number of meters you move. But meters is measured in some norm. And people use different norms the denominator's going to be different and they're going to chose different directions to be the steepest descent. So this is actually – once you understand, that's actually the key to understanding everything we're going to do. Now, the convergence is something like the gradient descent method. In fact, gradient descent is steepest descent in the Euclidean norm, which is not surprising. Okay. So let's look at an example here. And once you visualize what steepest descent is you'll realize why it is when you ask two people at the same place what the fastest way to go downhill, they can point in different directions. By the way, they can't point – they actually have to point in directions that – They can only differ by less than 180 degrees, that kind of obvious. If they're pointing completely in opposite directions, there's some trouble. Right. Or it's very good or you have to minimize – well, no, they should have told you that. But they can point different directions.

Let's see what it is. If I take just a quadratic norm then it turns out the steepest descent direction is minus P inverse descent. We can actually work out what this is. You're here, and actually, I guess, first you can do the following. Let's, actually, first workout what the steepest descent is in Euclidean norm. So what you do is you say I'm going to use the linearlized model. So use the lineralized model, that's a negative gradient. And you say I'm going to move in this ball to minimize grad F transpose V, in this ball. And then answer, clearly, is right here. It's says go in that direction., which is the negative gradient. But now, suppose, in fact, that you measure your displacement using this quadratic norm this is the ball. Then it says, go as far as you can in this direction in this ball. And what happens is, it's here, it's been rotated over to this direction. It's been skewed over here. So you can see, these are, obviously, not the same direction, and you've actually twisted the thing. Now, in fact, you can't twist it more than 180 degrees because that's a positive definite matrix and you multiple a gradient of one vector by a positive definite matrix, you don't rotate it more than 180 degrees. In fact, that's 90. Well, whatever, this 180. That is exactly what – actually 90, sorry. That is exactly what a positive definite matrix is. So you've twisted it over like this. It gets interesting, too, if you wan to do steepest descent in the L one norm because you'd draw a ball like this and take the gradient like this. By the way, what would be the steepest descent direction in the L one norm if this

were the gradient, if that were the negative gradient, what would be the steepest descent direction?

Well, what's the farthest – it's weird, it's curved, that's kind of a weird thing. All right. Imagining that I had drawn that arrow straight, what would be the steepest descent direction in L one? What would it be? Well, you go as far as you can in the direction and so it would actually be this point here, and that would be your direction. And you can kind of make a guess about steepest descent in L one. Go ahead and make a guess as to what it –it's always a long unit vector. Or can always be chosen to be a long unit vector. If you're going along a unit vector it actually has a very beautiful interpretation. It basically says you're updating one component of the variable. So the deepest descent in L one at each step you optimize over one component of the variable. Okay. So that's about it. Okay. That's not actually – our main interest is not that. It actually has to do with the choice of the norm for steepest descent and so this will illustrate this here. Now, the first thing we'll take – now, by the way, these are sublevels of some function here, some function we just looked at, like this. And the point is if you look at his function what you see, it's very rough, but this is the idea. The function is sort of elongated, it's not spherical. Right. It's kind of fatter than it is tall, roughly. Okay. That's the main thing here.

If you chose a norm that is sort of aligned with the gross geometry of the sublevel set, an interesting thing happens, you end up, actually, making very fast progress towards the center, okay, because you actually end up going in the right direction. If you take a norm, which actually is whose unit ball is sort of not aligned the same way the gross geometry of the sublevel set is, it actually amplifies this effect of osculation. Okay. So the rough idea is something like this. There's another way, also, to understand this, which I'll get to in a minute. Basically, to understand something like this, if you really want the norm that you use, if you use steepest descent you want the norm to sort of be consistent with the geometry of your sublevel sets. Okay. So if you're problem is the sublevel sets, you know, we're sort of more or less spherical, you know, like that, right. By the way, if the sublevels set were spherical, how would gradient descent work, or nearly spherical? How would gradient method.

Student:[Inaudible].

Instructor (Stephen Boyd):Actually, if it were perfectly spherical it would work flawlessly. It would say that's the negative gradient, it would go in that direction. Negative gradient would point, if not to the center, to the optimizer, it would point very near it. You'd minimize it and you'd end up over here. So actually, the gradient works perfectly when your function is, and I'll try to say this right, when it's isotropic or something like that when it kind of extends, when it looks about the same in all directions. It is kind of spherical or in all directions it looks about the same. Gradient descent works absolutely horribly when the sublevel sets look like this. Okay. That's exactly when you get this sort of story. Okay.

Now, when you do steepest descent in a norm, it's the same as changing concordance and then applying gradient descent. So what you want to do is you want to change concordance to make things that look like this look like that. So the whole key, in fact, is choosing the norm. That's the key to everything. Okay. So the norm you do steepest descent in is very, very important. Okay. Now, by the way, this already, if you just incorporate this amount, you already know a lot. If you really – you actually know a lot more than a lot of people who do optimization. If you just understand just what I just said. So for example, suppose you know something about the function you're going to minimize. Right. Suppose you know you have samples of it or something like that. You can do all sorts of cool stuff. You could like fit a quadratic. Suppose you already evaluated the function like a couple hundred points, you can fit a quadratic to it, or do whatever you like.

You fit a quadratic and a quadratic would sort of tell you which direction the function is sort of, you know, long in, and where it's got steep canyons. You would then use that quadratic to change concordance or if you like, you can use that metric to do steepest descent. Everybody see what I'm saying? That would work extremely well. I mean, that would improve steepest descent tremendously. Okay. Now there's a very obvious choice, though. If you want to get to, let's see, I want to motivate this, so I motivate it this way. Let's suppose you're near the bottom of a function, you're near the minimizer. So here you are. It's smooth. Actually, what does the function look like near the minimizer? What's an approximation of F near the minimizer? So well, it's equal to F of X star, right, plus grad F of X star, transpose X minus X star, right. What's that?

Student:[Inaudible].

Instructor (Stephen Boyd):That's zero. Okay. So we go to our next string, which is one half, X minus X star, transpose, you know, the Hessian, X minus X star. Right? So that's our next string.

Now, near the optimum point, the optimal point, it looks quadratic. And so what does the sublevels of F look like, approximately, near the minimum? They're ellipsoids. And they're ellipsoids determined by the Hessian. So they look like this. You know, they're ellipsoids. Hey, that's funny. That's exactly what we want because it tells you what the shape looks like. And it basically says, if for some reason you knew that number, of course that would be very – I'm sorry, that matrix, that would be very interesting that you knew that because the only way to find that, I guess, would be to find X star first and evaluate it, but suppose anything intelligent you could guess about this Hessian, it's going to tell you something about the shape of the sublevel sets, certainly in the end game when you get close. Okay. So then you can imagine, I'll do steepest descent, but I'll do it in the norm induced by the Hessian. Okay. And that brings us, that's very famous, that's called Newton's method, and that's our next topic. Okay. So let's look at the Newton step. So the Newton step is this. It's actually, in fact, a – it's most way interpreted, it's minus the Hessian inverse times the gradient. That's all. And you can say lots of things about it. So here's one. One way to motivate what the Newton step is this, is you take your function and you develop a second order approximation, that's this. And then it

says, I'll minimize this, instead. If you minimize this, I mean, you just take the gradient with respect to V equal zero and you'll get V equals the Newton step here. So this minimizes that. That's one way to think of it. And, in fact, that's a very good way to think of Newton's method. Newton's method will use the Newton step.

And it basically says, you take a function at each point, you develop, not a first order approximation, but a second order, and you minimize that, and then you update. So that's this. Another way to say it is this, you want to solve, if you're at the point X you would like to find a V for which the gradient of F, evaluated X plus V, that's where you're going to step. That should be zero. But the problem is the gradient nonlinear function, you don't really know what gradient of X plus V is. Okay. All right, so what you do is you say, well look, this nonlinear function is about equal to – I'll replace it with this linearized approximation, which is the gradient where you are plus the derivative of the gradient, but that's nothing but the Hessian. So this is that. And then you set that equal to zero. And if you solve this equation, of course, you get V equals minus Hessian inverse gradient. You get that.

And the two pictures correspond to something like this down here. So in the first case, here's F, our function, and here's our point X. So in the first interpretation it says develop not a first order Taylor series approximation, but a two term Taylor approximation. By the way, when you carry out two term Taylor approximation, convexity is preserved. If your originally function is convex, so is your two term Taylor's expansion, obviously, because the Hessian is positive semi-definite. And it says minimize that and that would give you this one, this point here. Notice that you know, at that point already you can see that F hat and F are not the same. By the way, what will happen in the next step here? If that's our Newton step and I carry out this again, I fit a quadratic to this point and do it again. What's going to happen? It's actually going to be much better this time. And when I start getting near this bottom point, these quadratics are going to extremely good approximations and the improvement you're going to make each step is going to be dramatic. Okay. Another way to see it is this, here's F prime and we want to solve F prime minus zero, that's our optimally. So here's my – and it's convex, which means F prime is increasing. So F prime is increasing. That's F prime. I'm here and evaluate the derivative of prime. That's the second derivative. Okay. And I form this dash blind, which is, in fact, this thing. Okay. Which is here is very simple. And I take the zero of that, that's that. By the way, what happens on the next step?

When this one is extremely easy to see what's going to happen. You're going to put a line through here. This is not linear here or F line, but it's very close and the next iterate is going to be somewhere in here. When you get in there, the iterates are going to be unbelievably accurate and you're going to get very fast conversions. Okay. So this is sort of the picture. So by the way, people call methods, where you update sort of this matrix here, they call these a beautiful term for this, although it means, let me be a little more specific, it's a variable metric method. So metric here refers to how you describe distances and things like that. So a variable metric method is a method that varies. It's essentially, the norm being used in steepest descent at each step and it adjusts it to the geometry of the particular problem instance. That's what a variable method metric is. So

although you wouldn't call a Newton method a variable method, it is one in that broader sense. Okay. So you can think of the third interpretation is this, is that the Newton step is steepest descent direction in the local Hessian norm, something like that. So that's what it is. That's the picture. Okay.

Let's look at the Newton decrement. Now, the Newton decrement is actually, it's – the Newton decrement square, it's defined as this. It's actually the norm, [inaudible] is the norm of the gradient but in the metric induced by the Hessian. So that's what this is. That's that. And it's a measure of the proximity of the X to X star. And it's actually a good one and we'll see some very good properties it in a minute. And it allows you to make a very good approximation of how suboptimal you are. So one half [inaudible] of squared is a very good approximation of how suboptimal you are. And it's basically the norm of the Newton step and the quadratic Hessian. It's also the direction of the derivative in the Newton direction so it turns out you actually have to calculate this whether you like it or not because that's the thing that you used in the line search. That's the RB hole. This is the quantity that appears on the right hand side of the RB hole line search anyway. And here's the really cool part about it. It's F line invariant. And let me say a little bit about that with Newton's method. We haven't talked about Newton's method, but I'll say something about that in a minute, but let's – yeah, okay. Let me get to Newton's method and then I'll come back to this.

So Newton's method, very, very simple, looks like this. You're given a starting point and a tolerance, or something like that. You compute the Newton step in decrement. You know, obviously, here if land to square is small enough you quit. That's fine. Otherwise, you do a line search and then you update. That's Newton's method. All right. Now, what we'll find out is that this works unbelievably well. Shockingly well. We'll take a look at it. We'll see in a minute. But there are many reasons one, I think probably the best zero order description of why it works is F line invariance. So let me explain what that is. If you want to minimize F of X, I'll try to get the notation here right. Okay. I could change concordance and let TY equal X, where T in a nonsingular matrix, and I could, just as well, minimize, you know, F of TY over Y. Obviously, I could do that, a change of concordance. Okay. But there's a little bit of – and then you ask what do I apply if I change concordance and then applied gradient method, for example, you know, do you get the same thing. And the answer is, no you don't. And in fact, what happens is in a gradient method when you take, if you call this thing F tilde of Y, then grad F tilde of Y is actually T transpose time F of TY. Okay. So you get that. What have I done there? So you actually, it changed concordance and that means, actually, if you change concordance and then apply gradient method, these don't commute. It's not the same as applying gradient and changing concordance. They don't commute. Well, in fact, that's the message of everything we just looked at, if you change concordance or change the metric the gradient method changes.

Now, you can look at it as an optimist and say, hey, that's cool. If I only get the right concordance gradient method is going to work really well. In fact, we know what the right concordances are geometrically, we want change concordance so the function looks isotropic, it looks round. That's what we want, we want the function to look about the

same in each direction. We don't' want steep canyons. We don't want stuff like that. If it kind of looks like the same in all directions, gradient descent is going to work really well in that metric. So that's the idea. Okay. Now, the truth is, there's a lot more bad choices of concordance than there are good ones. So with all due respect to the optimism express by saying there is a choice of concordance, which makes gradient method work well, which is a true statement. There's a lot of ones that don't. Another way to say it is this, if by any chance you're problem is solving well with gradient descent, and by the way, there are plenty of problems that solve well with gradient descent. Obviously, these are the ones, which, for various reasons, are round, or they're not too un round. Right. They're the ones that where they kind of about – there's lots of cases where gradient method just works. For super large scaled problems, you know, it's about one of your only choices. So you know, good for you, if a gradient method works. Right. But in fact, they generally don't work for smaller problems and I can always just change concordance. I can even take T diagonal. And I guaranteed you if your gradient method is working and I change concordance, I mean, I can just scale your variables, and I will bring your gradient method to a halt. Of course, in theory, it will convert. But I can make your C .99999999 and you're going to have a lot of iterations. Okay.

Now we get to Newton's method. Newton's method, actually, you get a commutative diagram. So Newton's method and affine changes of concordance they commute. So in other words, if I want to minimize F of X and I change concordance and I do something like I minimize F of TY here. I will get a sequence of Newton iterates. Okay. So this is Newton, this is change of concordance, right. I will apply Newton and I'll get, you know, X one, X two, you know, and it's going to converge. And here, I'll get Y one, Y two, and so on, like that. Okay. Now, I guess, if you have any aesthetic sense at all or even a small training in mathematics you have an overwhelming urge now to – you're hands are shaking because you want to fill this in. Its' just a – anyway, so you want to fill – and that's called a commutative diagram. And it basically means, it's cool. It says, basically, this basically was called a commutative diagram because it's basically Newton's method and changes of concordance commute. And in this case it's correct. By the way, if I replace these with gradient method, it's completely wrong. It does not commute at all. These are totally different sequences. And from a practical point of view, this is very, very important because it means, for example, these X's could converge basically to a good enough solution in ten steps and this could take ten million. Okay. So change of concordance and gradient method do not commute, very important. Changes of concordance and Newton method, what did I just say? It was wrong. That's the problem with these things being on TV. All right, here. I'll say it again and this time I'll say it slower and it will come out right. I think I said – did I say it wrong?

Student:[Inaudible].

Instructor (Stephen Boyd):I did. Oh, I've just confused myself. Okay. That's fine. I'll go slow now. Changes of concordance and Newton's method do commute. I was right. So they do commute. And actually, that means all sorts of cool stuff. It means, I can tell you what it means. It means to first order scaling is a non issue in Newton's method. In fact, it's actually only numerical that it's an issue. And let me explain that. In gradient

method, if the condition number is 1,000 of the sublevels sets or whatever it is, you're in deep trouble because that means you will be making .1 percent progress at each step. That's what it means. That's what it means. So in that sense, you know, a cap of 1,000 is already bad for gradient method. Now, if you take Newton's method and you do everything exact arithmetic, it's just no problem, it's identical. There is a second order effect. The second order effect actually has to do with numerical and your algebra and actually solving HV equals minus G, where H has high condition number. But a condition number of 1,000 for a positive definite thing is a joke. It's basically zero for all practical purposes. That's an extremely well conditioned positive definite matrix from the point of view of the numerical analyst. From the point of view of the gradient method, it's basically slows the gradient method to useless.

I mean, 1,000, I was drawing pictures here with ten. And you could actually visually see the ridiculous osculation. Imagine what it is for 1,000. Okay. And in a lot of dimensions where you can't visualize it. Anyway, it will be very, very slow. There'll be lots of osculation and stuff like that. And it'll basically be unusable but for Newton's method, no difference, absolutely no difference, it will just run perfectly. Okay. So that's the idea. So let me tell you a little bit about the convergence analysis in Newton's method. So Newton's method, first we'll look at the classical one. This is Kontorovich. You'll never guess where he was a professor. I figure your first guess is right, actually. So this is, although I'm not pronouncing it right, but I'm sure there's people here who speak Russian, you know, can tell me how. But that's how I pronounce it and I don't know. It's like Gaous or – anyway. Okay. So here are the assumptions. We'll make F as strongly convex on S with constant M, and the Hessian is going to be [inaudible] continuous. Now, by the way, this is essentially a constraint on the third derivative of F. Right. Because basically we're saying how fast can the derivative change. You can interpret L as a limit on the third derivative of F. Now, let me tell you why we don't do it that way. Because what is the third derivative of a function from RN to R? Here's a hint. Only people in mechanical engineering and physics would not be bothered by what it is.

Student:[Inaudible].

Instructor (Stephen Boyd):Thank you. That's a tensor. It's a tri linear form, if you're in math, or it's a tensor. It's got three N to C's and all that kind of stuff. It's not fun. I guess, actually, if you're fully trained in all these things, I think it doesn't scare you at all. It's no problem. In which case, all I have to say is good for you.

So anyway, it's easy. All we need is a bound on it so we'll just put it this way. By the way, before we get into this let me ask you some questions. How well does Newton's method work on quadratic functions?

Student:[Inaudible].

Instructor (Stephen Boyd):One step! It works super well. Why? Because Newton's method does this, it takes a point and develops a quadratic model of your function. Now, in this case it happens to be your function because your function is quadratic. Okay. It

them minimizes that quadratic function, which unbeknownst to it is your function. And then steps to what's the minimum and asks again, say, hand me the gradient and the Hessian, and the gradient is zero, and then it actually sends a message back and says, don't even bother with the Hessian. So it's one step. Okay.

So basically, when do you think Newton's method's going to work well? Well, it should work well when the Hessian is slowly changing. So if the third derivative is small Newton's method should work really well. Is everybody kind of seeing the idea here? Because, basically, Newton's method is you're basically, let's see, at each step you're actually minimizing a quadratic model. If your quadratic model is really good, even for big displacements, Newton's method is going to work really well. So basically, small third derivative is what makes Newton's method work. Okay.

So I'm just getting you all prepped for the idea that L, the ellipsoid's constant, is going to play a role in Newton's method. And the ellipsoid's method on the Hessian is indeed a bound on the third derivative. And we know, at least in one extreme case, if L is zero then Newton's method works very well. One step is quadratic. Okay. All right.

Now, here's how Newton's method, the classical analysis works. It's going to work like this. There's going to be two regions of sort of convergence and it's going to work like this. If the norm is bigger than some number, and that's some positive number, then what's going to happen is F, you're going to be guaranteed at each Newton step to decrease F by a constant, just an absolute constant. Okay. Now, by the way, that means that this step, if F does have a minimum, this step cannot be executed more than some number for times. Because basically, it says you guarantee a decrease of, you know, whatever it is, .26, at each step. But that means if you make case steps you have reduced F by .26K so you can't go too – if you went infinitely off in doing this you'd have a minimizing sequence going to, it would unbound up. Okay. So that's the picture. Now, so eventually the gradient gets smaller than something. Now, what if the gradient is smaller than some number [inaudible] you have the following. You have to look at this very, very carefully. It's got a constant L over two M square, don't worry about that, but look at this whole thing has measure of error. That's a measure of – well, it is a scaled residual, and this says that the residual is less than the residual squared at each step. Okay.

Now, by the way, once this number – let's suppose, at some point, this number becomes one half, just for example. So this number gets down to a half. I says on the next step it's less than a quarter. And on the next step it's less than a 16th, and then 1/256th, and then 1/256th squared. Everybody see? So and notice the convergence then is extremely fast. You're basically, once this number – it basically says the error is squared. It's all very rough, but it says, basically, that the error gets squared every step. Once that error is small, that's a huge reduction in the error. Compare that to saying the error gets multiplied by .99 at each step, huge difference. Okay. So let's look at how this works. Now, when you're in the so called damped Newton phase, damped Newton means, well, we'll see it, it means you actually require backtracking steps. It means that a full Newton step you actually have to backtrack from T equals one to whatever, something. And the function value decreases, it has to decrease by a least Gamma. Now, as I said, if the

function is bounded below, this has to end after, at most, this number of iterations. That's obvious because at each step – otherwise, if you ran more than that number of iterations, you've really reduced the value of F to blow it's minimum value, which is not possible. So that bounds the number of steps you take in this stage. Once you get into the so called quadratic convergence stage, it basically says the following. It says that this thing, here, is less than one half two to the L minus K. So that's how that works. So once you've gotten to that point. So at this point, you're actually, I mean, actually, many people say this many, many ways, one way is to say the error is squared at each step, not quite true. Another way is to say the number of actuate digits doubles every step, because that's what this says. Well, it says it in bits, actually. And so the number of accurate digits doubles.

Of course, you know, most normal people use doubles and floats, a tripling floating point. So this is amazing, but it basically says that you know, you can't execute this more than about four of five times before you get the double precisions stationery. At which point, normal people say you're done, you're actually done. Okay. So if you work out the global complexity analysis it works like this. It says that the number of iterations until you get less than S one, is less than this, that's the initial phase. And look, it depends on how bad your initial point is. So if your initial point, I mean, basically, actually, it depends on how suboptimal your initial point is here. And then there's the second one. This is the quadratic term. This is log, log, log, F one zero over F one. F one zero's one of the constant. So this is it. Now, I can, let me explain a little bit. Actually, I go around and give talks often about things like this. And what I do, actually, when I do that, is I actually give talks and I replace this with the number five. I'm just bating people. So it's for me to torture complexity theorists. So I wait, I wait, I just keep going with the five, I'll even stare right at somebody I know, just kind of look at them right in the eye and I'll say plus five. Just baiting them, baiting them, and finally, someone takes the bait and they'll say, excuse me. And I'll say, oh, yes, do you have a question?

And they'll say, yeah, you know, you can't solve that problem exactly so what happened to the accuracy Elvin? Huh? I don't see Elvin in there. And I say, oh, sorry, this symbol five, that's how I write log, log one over Elvin. So from now on whenever you see the number five, it actually means log, log Elvin zero over Elvin. Is that okay? Because that's how normal people write this expression, is five. Okay. So it works pretty well, actually. They do fall for it pretty often, actually. All right. So the interesting thing about this, actually, is it says that to a normal person it basically says that, actually – let me ask you this. How does the number of Newton steps required, and I'm talking by the theory, even by the theory, and by the way, in practices it's even stronger, how does the number of steps of Newton iterations required change with the accuracy required? Though, if you're in a court under oath, what would you say? You would say, very, very slowly, log, log one over Elvin would be the answer. But any other place, what would you say? You'd say, not at all. And someone would say, really? The cost of computing a super accurate answer is the same like a crude, sloppy, terrible answer? By the way, for gradient method, those two numbers are grossly different, right. Because you paid for gradient and it's hard work. You keep multiplying by some number that's .99 and you have to do it a lot more times before you drive something to an error that likes one E minus eight or one

E minus 12 or whatever it is, right? It's going to cost a lot. For Newton's method, it's basically costless. Everybody see what I'm saying? Okay.

So that's why, by the way, people don't worry a lot in practice about what's the stopping criterion for Newton's method. Because roughly speaking it's this, once you're in to the quadratic, because the cost of calculating a super accurate solution, is essentially like one or two steps more than calculating an okay solution. So it just doesn't matter. That's one way to say it. Okay. Okay. Now there's two minor problems with this classical conversance analysis and I'll say what they are. The first is that, obviously, this goes back to my comments before, obviously, in any particular problem, you have absolutely no idea what L is. Number one, you have absolutely idea what M is. Now, the other things appearing in here, like in Gamma, you have to have Alpha and Beta, these are the line search parameters. You do know what those are because they're in your code. Okay. They're in your – you just look through your code and you'll find them. But you don't know what M is, you don't know what L is. That is ridiculous. Right? So the point there is this is still useful because you will observe sort of a damped phase and then a quadratic convergent phase. So let's look at some examples. Here's an example of Newton's method for that problem before. And you want to notice two things in this example. You want to notice actually the access here. And you want to notice the access here. And by the way, here you see, let me show you what you see. You can find iterations. So the point is but here you have this region something like that, you would argue maybe here, it's sort of the damped phase.

You have to see this. You have to see this thing roll over for Newton method. You'll implement one this week. Actually, in the next two days, right, it's Tuesday. So if you don't see this thing rolling over, you're not achieving quadratic convergence. That's what it means. Okay. It is possible, actually, that quadratic can have this look kind of flat and then roll over later. But we wouldn't assign something like that to you because it would give you the wrong message. Generally speaking, if you see it flat, by the way, if you're plot looks like this, yeah, then there's something really seriously wrong. Okay. But if you don't see that, you're not doing Newton's method. Also, these are about the right numbers here. And you might ask how does that scale and all that kind of stuff, and you would see something like this. This is an example of our 100. Again, you can note this axis and again, exact backtracking line search saves you one step or something like this. And you want o know how this scale. Here's an example in our 10,000. And it's very close, I think, to what you might be doing, anyway. So here, actually, you can see it quite clearly. So this is our 10,000, you're minimizing some smooth convex function. And, in fact, you might argue that quadratic convergence goes into effect somewhere around here. Something like that. I mean, who knows, but that's roughly what it is. This is what you see.

So you might have like 13 steps and then, once you hit quadratic you'd go six more and it's all over. And you get something like this. This is our 10,000. That's a big place, by the way. You might ask what this look like at our ten million, and the answer is it looks exactly like this.

Student:[Inaudible] computing the inverse [inaudible] compared to the cost of the actual steps you have to do after [inaudible].

Instructor (Stephen Boyd):That's a very good point. So let me actually say something about that. Because that's a great point. Also, I'll short circuit a whole section of what you'd hear in other classes. Okay. So you should notice that this axis, it is completely unfair to compare this axis to the one in the gradient method, for precisely that point. Each step here requires solving HV equals minus G. That's how you calculate a Newton step. Okay. So in a dense case, the cost of that is going to be N cubed. The cost of evaluating the gradient and all that, we could maybe make it N. I mean, it depends on what the problem is and all that. But let's just say it's N or something to keep it simple. It'd probably be more like N squared. But let's make it N. So then you actually have to take that into account. At which point, you'll actually still find the following. If you need high accuracy, nothing will beat it. Nothing will beat Newton. So that's it. Now, by the way, there's a whole family of methods in between gradient and Newton. And these came into being in the '60s and they came into being because solving, for example, there are even very famous people who said things like, no one will ever have to solve, it wouldn't even make any sense, to solve 100 equations with 100 variables. That just doesn't make any sense. There's never a time when 100 things could depend on another 100 things. And besides, there'd be 10,000 co efficiency in there. You see what I'm saying here?

So this would just be inappropriate to solve AX equals B if AX is B. I mean, we're talking iron corps memory days, which means nothing to any of you. And by the way, it means nothing to me, too, just for the record. But still, you know, this is a day when it was a big – N cubed was a very scary thing when N was 100. Okay. For you, N cubed doesn't start getting scary right now this year until N equals 2,000 and it's not even that scary. Right? It's not that big a deal to make it go a bit higher. It'll start getting scary around 5,000, 10,000, that could get scary. But that's what 100 looked like before. So in those days they worked out all sorts of methods where to avoid getting out of solving H here, HV equals minus G, they avoided the N cubed trick. And they did things like they would update H by eristic and stuff like that at each step and so on, and there's lots of methods. By the way, some of these methods are quite good and they actually do have their uses. We'll look at them next quarter. And these are called quasi Newton methods. They have all sorts of names like this.

So I have a couple comments about it. So the first is this, in terms of the cost of Newton's method. If you don't know how to solve linear equations, and I'm sorry to report that many, many people do not know, then they overestimate what it costs to solve HV equals G, this thing, HV equals minus G. Okay. So when you're more sophisticated, like you are now, after last week, and H is sparse, you know banded plus low rank, if it's got Copeland structure, anything like that, you will realize all of those tricks can be brought to the bear here. So I think, one of the things that you have to remember is that in many problems, when you compute the Newton step, there's lots of structure lying around in real problems, lots and lots. And it it's exploited, which is to say that you do your linear algebra intelligently, as opposed to stupidly, you'll find, often, that the cost of Newton, that Newton's method is entirely feasible even in things like signal processing problems,

image processing problems, where the number variables in a million or more, you can find you can do Newton's method. Obviously, you couldn't even, remotely, write down the Hessian and store a Hessian in a single processing, memory-processing problem or a large machine-learning problem, you couldn't do it. But you know what you're doing, you can actually solve these equations, shockingly. So that's a longwinded answer to this.

So it's not 1963, in which case the rest of the quarter would be methods for avoiding using Newton's method at all costs. It's not. So things have changed and now, Newton's method is just not a big, it's just not a big deal. And especially, coupled with if you know how to solve linear equations intelligently. And a lot of times, the cost of Newton method drops right down to the quasi Newton methods from the '50s and '60s. So okay. Well, we'll quit here and have fun, because you're going to do all of this.

[End of Audio]

Duration: 77 minutes