

ConvexOptimizationI-Lecture01

Instructor (Stephen Boyd): Okay. Welcome to 364a, which is convex optimization one. Unfortunately, I have to start class by apologizing. If you're watching this, I guess you'd say – if you're watching this at the scheduled time of the class, you'll notice that I'm not there. That's because I'm in India. You can actually get very confused doing this.

There's actually one more confusing thing than this, and that's when you tape ahead of class before it actually – it goes after a class you haven't given yet. That gets very complicated. I'm sorry to miss the first day of the class. As I said, I'm in India or will be, depending on where the time reference is. That's very tacky. I've never done it before, so this is a first.

Actually, we're going to make a lot of firsts today. I believe we may be making SCPD history because we're taping ahead this lecture not only the quarter before but the year before. A lot of people are very excited about this. If this works out well, who knows where this is going to go. I could teach all of my spring quarter class the last few weeks of winter quarter. We'll see how this works. We're possibly making tape ahead history here.

A couple things I should say about the class, other than apologizing for not being present or, for that matter, in the country for the first day of it. I'll say a couple of things, and then we'll get started. For those of you watching this, since you're not going to get to see the audience and know that I sent out a pitiful plea to come if you're around for the tape ahead, you should know the room is not empty.

Let me say a little bit about the class. I think some of it is kind of obvious. We barely got the website up and running. We change it every day. Let me say a few things about it. The first is the requirements – there is going to be weekly homework, which if you are interpreting this relative to January 8, I think the correct statement would be “has already been assigned.” If you're interpreting it in real time, it's actually fall, so we haven't even thought about homework one, but we will. It'll all be ready to go.

There'll be weekly assignments, and there will be a take home final exam, which will be the standard 24-hour take home exam – the usual format. You'll do serious work. The other requirement for the class – it's not really a requirement, but there will be minimal Mat-Lab programming. It's not a big deal, but there will be Mat-Lab programming, and that will be done using a system called CVX, which if you go to the website, you can find out about.

You're welcome to install it and read the user guide now. There's no reason not to. You'll be using it throughout the quarter. You might as well look at the user guide, install it, see if it works. Not everything in the user guide will make sense until you get to week four of the class, but there's no reason you can't start poking around with it now. When we get to it, you should be extremely grateful for the existence of this stuff, because it

trivializes the “programming” that we’ll do in the class. Programming is in quotes, and the reason is we’re talking things like ten lines. It’s not a big deal.

By the way, on that topic, I would say this – if there’s anyone that – CVX is based on Mat-Lab. That’s the way it is. If there’s anyone who wants to try to use a real language, like Python, I will be very happy to support you in that. The TAs are Yung Lung, who I guess you can’t see, and Jacob Mattingly, who’s in New Zealand, but will not be in New Zealand when this is played back. Jacob would be very happy to – if there’s one person who does all the numerical stuff in Python. I’ll just say that, and feel free to come forward and let us know.

The only other thing I would say is something about the prerequisites. The prerequisites are basically a working knowledge of linear algebra. 263 absolutely – it’s clearly the perfect background. If not, what we mean is a working knowledge, so not the ability to know about these things but actually how you use them in modeling and things like that. The other one is the mathematical sophistication level is going to be a jump up from 263. This is a 300 level class, so the mathematical sophistication is going to jump up.

The cool part is the mathematical sophistication jumps up but actually is still in some sense a very applied course in the sense that we’re not just going to talk about optimality conditions. You will do portfolio optimization. You will do optimal control. You will do machine learning. I mean actually do numerical stuff. It won’t just be stupid programming. It’s going to be figuring out what’s going on. It won’t be obvious. It’ll be good stuff.

I’m trying to think what else to say before we start in on the class. I’ll just set it on context how it’s different from 263. 263, which is a linear dynamical systems class – I would call that basic material. It’s useful in tons of fields. It’s very widely used. A lot of people know it. It’s used in economics, navigation – all of these areas. Control and signal processing is basically all done this way. Communications – a lot of stuff is done this way. It’s a great first look at how these mathematically sophisticated methods actually end up getting used.

364 revisits it. It’s quite different in a couple ways. The first way is you might even just say this is 263 beyond least squares. In other courses similar to that, so in your first look at statistics and your first course on statistical signal processing or whatever – it’s the same sort. Everything is Gaussian. All distributions are Gaussian. All objective and constraints are quadratic. You do this because there’s analytical formulas for it. Behind the analytical formulas, we have very good computational tools to give computational teeth to the theory.

Here, it is going to be way beyond that. We’re going to have inequalities. We’re going to have all sorts of other interesting norms and other functions. It’s a much richer class of problems. These are problems much closer to a lot of real problems, ones where – you don’t have the concept of an inequality in 263. You have no way to deal with noise with a

[inaudible] distribution in your first class on statistical estimation. We're going to look at things like that.

I should also say that the number of people who know this material is relative to 263 very small. It's probably a couple thousand. It's growing rapidly, but it's just a couple thousand. That means you're going to move from material – a lot of the 263 material is kind of from the 60s and 70s. It's not stuff that's new. This class, in contrast – you're at the boundary of knowledge, which makes it fun. Maybe you'll even push the boundary. I hope you do, since that's the point of the class to train you to find your corner of the boundary and start prospecting.

I can't think of anything generic to say, so maybe we'll actually start the class, unless there are any questions.

Student: If you haven't taken 263, is it a big problem if you have a working knowledge of linear algebra?

Instructor (Stephen Boyd): No, no problem. What'd you take instead? No problem. I should also say this – something about the background. The class is listed in electrical engineering. I'm not actually sure why. The last I checked, that's the department I'm in, so it seemed like a good idea all around. You don't need to know anything about electrical engineering – in fact, we'll talk about lots of applications throughout the quarter, and honestly, probably for half of them, I don't even know what I'm talking about. That will happen. You'll be in some field. I'll super oversimplify it. You're welcome to point it out.

The point is, I'll talk about circuit design. We'll talk about machine learning. On all of these topics, there will be people in the class who know more than I do about it, and there will be a lot of people who don't know anything about it. If you're one of those latter, don't worry about it. They're just examples. There is one prerequisite, although I don't know what would happen if you didn't have it. That is that I will assume that you know about basic probability estimation and things like that.

Do I have to say anything about the textbook? Go to the website. That's all I have to say. There's a textbook. You'll find it at the website. I'll start by covering broad basics. It's not in any way representative of what the class is going to be like, what I'm going to talk about today. Don't think it is. We're going to cover the basics, and maybe I'll get into something that's real. This is going to be the very highest level. We'll talk about mathematical optimization. We'll talk about least squares and linear programming, convex optimization. We'll look at a very simple example, and I'll say a little bit about the course goals and style.

Let's start with this. Optimization. Depending on the company you're in, you may have to put a qualifier – mathematical. If you say just optimization, it goes to something on compiler optimization or something in business practices. If you're in a broad crowd like that, you need to put mathematical in front to make it clear.

Here's what it is. You want to minimize an objective function. You have a variable X with N components. These are loosely called the optimization variable. Colloquially, you would say that X_1 through X_N are the variables. By the way, there are other names for these. People would call them decision variables. That's another name for it. In a different context, they would have other names. Decision variables is a nice thing because it tells you these are things you have to decide on.

When you make a choice X , you'll be scored on an objective. That's F_0 of X . We'll choose to minimize that. In fact, you ask what if you wanted to maximize it? Then you would just minimize the negative of that function, for example. It's convenient to have a canonical form where you minimize instead of maximize. This will be subject to some constraints. In the simplest case, we'll just take a bunch of inequalities – that's F_1 of F_X is less than B_1 . Here, without any loss of generality, the B_1 s could be zero, because I could subtract B_1 from F_1 and call that a new F_1 . There'd be no harm.

This is a nice way to write it because it suggests what really these are. In a lot of cases, these are budgets. It's a budget on power. It often has an interpretation as a budget. There are other types of constraints we'll look at. What does it mean to be optimal for this problem? By the way, this is redundant. You should really just say an optimal point or the solution. In fact, you shouldn't trust anyone who says optimal solution, because that would be like people who say the following is a true theorem. It's kind of weird. It's redundant. It doesn't hurt logically, but it kind of – what about the types when you forgot to add the attribute true? What were those theorems?

I'd say an optimal point or a solution is a point that satisfies these inequalities here and has smallest objective value among all vectors that satisfy the constraints. That's what you mean by a solution. Of course, you can have everything possible. You can have a problem where there is no solution. You can have a problem where there's one, where there's multiple, and you can have other things, which we'll get to more formally. This is just our first pass.

Let's quickly look at examples. Here are some absolutely classic examples from all over. The first is portfolio optimization. In portfolio optimization, the decision variables X_1 through X_N – these might represent the amount you invest in N assets. For example, if X_1 is negative, it means you're shorting that asset, unless that asset is cash, in which case you're borrowing money. The X_1 through X_N is actually a portfolio allocation.

You'll have constraints. Obviously, there will be a budget constraint. That tells you the amount that you have to invest. There might be maximum and minimum investment per asset. For example, the minimum could be zero. You're not allowed to actually purchase a negative quantity of an asset, which would be shorting that asset. You can have no shorting.

You might have a maximum amount you're allowed to invest in any asset. You might have a minimum return that you expect from this portfolio. Your objective might be something like the overall risk or variance in the portfolio. You'd say here's 500 stocks

you can invest in. My mean return absolutely must exceed 11 percent. Among all portfolios that meet – I'm going to have no shorting, and the sum of the XI is going to equal one because I'm going to invest one million dollars.

The question is among all portfolios, they guarantee a mean return of 11 percent. I want the one with the smallest variance in return. That would be the one with the least risk. We'll look at these in detail later. This is one where when you solved the optimization problem, it would probably be advisory in the sense that you'd look at what came back and say well, how interesting. Maybe I'll execute the trades to get that portfolio or maybe not. This could also be real time if you're a hedge fund and you're doing fast stuff. This could be programmed [inaudible]. There are lots of different things this could be used for.

Let's look at [inaudible] electron circuit. Here, you have a circuit. The variables, for example, might be the device widths and lengths. It could be wire widths and lengths, something like that. You have constraints. Some of these are manufacturing limits. For example, depending on what fabrication facility is making your circuit, you have a limit on how small the length of a gate and a transistor can be. It can't be any smaller than 65 nanometers. That's the smallest it can be.

That's a manufacturing limit. You would also have performance requirements. Performance requirements would be things like timing requirements. You could say this circuit has to clock at 400 megahertz, for example. That tells you that the delay in the circuit has to be less than some number because it has to be done doing whatever it's doing before the next clock cycle comes up. That's a timing requirement.

You might have an area requirement. You could say if I size it big, I'm going to take up more area. This portion of the circuit can't be any more than one millimeter squared. The objective in this case might be power consumption. You'd say among all designs that can be manufactured in the sense that they don't reject your design instantly and meet the timing requirements among all of those, you want the one or a one with least power. That would be minimized power consumption.

Our last one is just from statistics. It's a generic estimation model. Generic estimation looks like this. What's interesting here is these are from engineering and finance and stuff like that. In these cases, the Xs are things you're going to do. In this case, they're actually portfolios you're going to hold, and in this case, they will translate into polygons that get sent off to the fabrication facility. The [inaudible] are actions. Here, it's very interesting, because the XIs are not actions. They're actually parameters that you're estimating.

Here, you have a model. You can take any kind of data or something like that and you'll have parameters in your model. You want to estimate those parameters. These XI are not things you're going to do. These are numbers you want to estimate. That's what it is.

You have constraints. For example, let's say that you're going to estimate a mean and a covariance of something. It can be much more sophisticated, but let's suppose that's the case. We have a constraint here. There might be some constraints. Here's a constraint.

The covariance matrix has to be positive semi definite. That's a constraint, because if you created a covariance matrix that wasn't, you'd look pretty silly. That's a nonnegotiable constraint.

You could also have things like this. You could say these Xs must be between this limit and that limit. For example, suppose you're estimating some diffusion coefficient or some parameters known to be positive. Then you should make it positive. These are parameter limits. The objective is, generally speaking, dependent on how you want to describe the problem. If you describe the problem in an informal way, it's a measure of misfit with the observed data. For example, if I choose a bunch of parameters, I then propagate it through my forward model and find out what I would have had, had this been the real parameter.

The question is, does it fit the observations well? Another way to say it – it's a measure of implausibility. That's really what it is. It's a measure of implausibility. In this case, we're minimizing it. In many contexts, you'll see it turned around so it's maximized. If you're a statistician, you would reject the idea of a prior distribution on your parameters, and your objective would be to maximize the likelihood function or the log likelihood function. That's what you'd be maximizing. That's essentially a statistical measure of plausibility. You'd minimize the negative log likelihood function, which I think they call loss in some of these things.

Implausibility in a [inaudible] framework, a measure of implausibility would be something like the negative log posterior probability. That would be a measure of implausibility. If you minimize that, you're actually doing MAP, which is maximum a posteriori probability estimation. By the way, we'll cover all those things again, so this is just a very broad brush. If you don't know what these things are, you will, if you take the class.

These are examples of optimization. Everyone in their intellectual life goes through a stage. It can happen in early graduate school, mid graduate school. It can also happen in later life, which is bad. It's not good to have it when you're an adult. Let me describe this stage of intellectual development. You read a couple of books and you wake up at 3:00 in the morning and say oh my god, everything is an optimization problem. Actually, a lot of books start this way. My answer to that is – you have to go through this stage, so that's fine. But get over it quickly, please. Of course, everything is an optimization problem.

What you'll find out quickly is it doesn't mean anything to say that. It says nothing. What matters is what optimization problem it is, because most optimization problems you can't solve. They don't tell you that. Typically, they don't tell you this. Or, what they do is they do tell you, but they distribute it through the whole quarter. It turns out if you just say a little bit of that message every day, at the end of the quarter, no one can accuse you of not having admitted that we don't know how to solve most optimization problems.

However, because it was below the threshold of hearing in each lecture, as a result, all these students went through and the big picture didn't come out, which is basically – the

way you cover it up is by going over 57 different types of algorithm for solving things, which basically is a cover up for the fact that you can't solve anything. We'll get to that.

This is related to the following – solving optimization problems. To say that everything is an optimization problem is a stupid tautology. It all comes down to this. How do you solve them? It turns out it's really hard, and basically in general, I think it's fair to say it can't be done. I can write down shockingly simple optimization problems and you can't solve it, and it's very likely no one else can. I can write down very innocent looking optimization problems and they'll be NP hard.

What do people do about it? Well, there are a couple of ways to do it. What do I mean by NP hard? Nondeterministic polynomial time. You take at least quarters on this in a computer science class. I'm going to give you the 15-second version. I'll tell you about it from a working perspective. In the 70s, people identified a bunch of problems that so far no one had figured out a good polynomial time algorithm for solving.

Polynomial time means there's an algorithm where the problem comes in, you measure the size by the number of bits required to describe – you measure it by the number of bits required to describe the data, and then you look at the number of operations you have to do. If you can bound that by a polynomial, then you'd say that's polynomial time. Bear in mind, I'm compressing a ten week course to about 14 seconds. I'm going to gloss over some of the details.

There were a bunch of problems where people – the famous one would be traveling salesmen problem. No one had found a polynomial time method. A guy named Cook, maybe, did the following. He catalogued a bunch of problems and said if you can solve any of these, then if you make a method for solving that, I can make a reduction. I can take your problem and map it to this, and with that, I can solve the traveling salesmen problem. Then you had an intelligent way of saying of two problems, one is just as hard as the other.

NP hard means, and this is really – people are going to cringe if they have a background in this – it means it's at least as hard as a catalog of problems thought to be very hard. That's your prototype. Basically, what I'm saying is not true, but as a working definition of what it means – for your purposes, this is going to be fine. It means the problem is at least as hard as traveling salesman.

Let me tell you what that means. It is not known that these things cannot be solved from the polynomial time. That's not known. It is thought that that's the case, and it may be at some point, somebody will show that you can't solve these. Right now, they're thought to be harder. I think there's an insurance policy. Let me tell you why it's an insurance policy. A ton of super smart people have worked on all these problems, and now all of these things are banded together as you would do in insurance. You band a whole bunch of things together.

What would happen is if tomorrow, somebody, probably a Russian, proves P equals NP , meaning you can solve all of these problems, it would indeed be embarrassing for a lot of people. However, the embarrassment is amortized across – people could say you went around and made a big deal about convex problems and polynomial time. I say look at all the other people. The embarrassment is spread across a great swath of people. It's an insurance policy. It is thought that they're really hard. If they're not, you're in very good company with the people who also thought they were hard.

Student: It's just in its own field. It's not proven to be certain. It's not mathematically proven to be non-polynomial.

Instructor (Stephen Boyd): That's right. That's why it's still an open conjecture. If, in fact, it turns out that these are theoretically not hard, the [inaudible] could end up being huge, and that would also blunt the embarrassment. In any case, the embarrassment is spread across a lot of people. We'll come back to that problem several times in the class.

Methods to solve a general optimization problem – they always involve a compromise. The two main categories of compromise are as follows. The first one is to not back down on the meaning of solve. Solve means solve. It means find the point that has least objective and satisfies the constraints. That's the definition of solve. You leave that intact, but you leave open the possibility that the computation time might involve thousands of years or something like that. People call that global optimization, and it's a big field. It is almost entirely based on convex optimization.

The other one, which is the one most widely done by people who do optimization – they do the following. It's very clever what they do. They go and solve – they put a footnote, and then way down at the bottom, they write this – they write and now I'll have to ask you to zoom in on that. There it is. It says not really. What that means is they make a big story about this, and they say it's a local solution. What happens is they don't – it gets tiresome to say solve locally, plus, it doesn't look good.

What happens is you drop it after awhile, and then you say I solved that problem, and people give talks and say I minimized that. The point is they didn't. What they did was they got a local solution. We'll talk more about that as well. There are exceptions. If you were going to do that, I would maintain, although this is not widely held – you'll find that many of my opinions are not widely held. If you were going to do that, I would do it via convex optimization.

Student: But you will be doing it in the spring?

Instructor (Stephen Boyd): Yes. It's scheduled to be in the spring. Who knows when I'll do it. What are the exceptions? There are cases where you can solve these problems with no asterisk. The most famous one in the world by far is least squares, least norm. No asterisk there. It's not like oh, yeah, well, I transpose A inverse A transpose B . Yeah, that often works super well in practice. The status of that is that is the global solution. There are a couple others that you might not know about, and that would be things like linear

programming problems. We'll get to those quite quickly. Those are ones where you get the answer.

There are asterisks in these, but they're much smaller fine print. I'll get to them later in the class. When there's an admission that has to be made, I will make it. The parent of these, and a considerable generation, is this convex optimization problem. This is really the exception. The rough idea, to put it all together, is something like this. Most optimization problems – let's review what we've learned so far. A – everything is an optimization problem. B – we can't really solve most optimization problems. The good news is here, that there are huge exceptions. This entire class is about one of the exceptions.

When you do convex optimization, there are no asterisks. You solve the problem and there are no apologies. It's the global solution. Is life convex? I would have to say no, it's not. I hope it's not. It's not sad. If we get to that later today, you'll know why it's not. To check convexity, you have to take two instances and then form weighted averages in between. What would the average of yours and your life look like? The other thing that has to happen is that life has to be better than the average of the qualities of your lives. Let's keep that as a running topic that we'll revisit periodically. For the moment, my position is it's not.

Student: Are there other exceptions besides convex optimization?

Instructor (Stephen Boyd): Yes, there are. Singular value decomposition. That's one where we can compute the singular value – I can write it out as an optimization problem pretty easily. I could say find me the best rank two approximation of this matrix. That's way non-convex. Yet, we can solve it and we get the exact global solution. That's an example. There are some combinatorial problems. So if you've taken things on combinatorial algorithms in computer science – combinatorial algorithms on their face would appear to be non-convex. It turns out a lot of them are convex. It's quite strange.

You take something that's a combinatorial optimization problem that on its face would not be. It turns out if you let the variables vary between zero and one, you can prove that there's always a solution, which is on the vertices, and so there's actually a lot of problems that we can solve but are not convex. Some of them can be secretly turned into convex problems. Getting a rank two approximation of a matrix is an excellent example. We can definitely do that and it is definitely not convex.

We have least squares or, if you're in statistics, regression. It may have other names. I don't know. Here's the problem. You want to choose X to minimize the [inaudible] norm squared of AX minus B . If A is full rank or skinny, you get an analytical solution, which you know if you know about linear algebra. It's just $A^T A^{-1} A^T B$. In this case, it's a unique solution. In fact, we have a formula for it, which is overrated. In this class, we're going to look at tons of problems. There will be analytical formulas to almost none of them. You'll have to wean yourself away from analytical formulas.

The sociology of that is very interesting. You've been trained for 12, 14 years on 19th century math, which was all based on analytical formulas, but we're going to move beyond that. If you look most deeply into what it means to have an analytical formula, it turns out – we're going to solve AX minus B with an infinity norm there or a one norm. There's no analytical formula for it now. But it turns out we can calculate that in the same time it takes for you to calculate this. Having a formula is something that will mean less to you by the end of this class.

Student: So not all optimization problems have that subject.

Instructor (Stephen Boyd): When we go over this in hideous detail later, that's the case. I should mention you should be reading chapter one and chapter two. Chapter one will have a lot of this vague stuff, and chapter two will start in on the real stuff. You're absolutely right. An optimization problem – you do not have to have any constraints, in which case it's called unconstrained, and you don't even have to have an objective. If you have neither, that's called a stupid problem. It's minimized. The universal solution – X equals zero.

If you don't have an objective, it's called a feasibility problem. In some fields, they call it a satisfaction problem. You have a bunch of inequalities and you want to find a point that satisfies all of them. That's what that is. Back to least squares. Much more important than the formula – it turns out you can write down a formula for a lot of stuff, and it doesn't do you any good if you actually want to calculate it. Here, there are reliable and efficient algorithms that will actually carry out this computation for you.

By the way, that's why least squares is so widely used because it has computational teeth. Instead of just talking about it, which is what a formula would allow you to do, you can actually use it. You can actually compute. We're not going to go into too much detail in the numerics. At one point in the class, we will. We'll talk about how you exploit structure and things like that. Here, just so you know, the computation time to solve a least squares problem goes $N^2 K$. N is the number of variables, and that's the small dimension. K is the number of rows in A . It's a good thing to know. It's the small squared times the big.

You can do this as I said not long ago in 263 with modern computing. It's amazing what you can solve. Then, we did a couple thousand row least squares problem. You can call it a regression problem in 2,000 dimensions with 1,000 regressors. It was three seconds. By the way, if A is sparse or has special structure – suppose part of A has an [inaudible] embedded in it. That would come up in medical imaging. You can do that faster. In image processing, you'd have transformations. You can solve least squares that are way bigger.

I would say that least squares is a mature technology. When I do this, people who worked on all of this – it's a huge, active field in lots of areas – they get extremely angry when I use the word technology. I said by the way, I mean technology here. This is the highest praise. This is not an insult. What it means is that other people know enough about the

theory, the algorithms, and the implementations are so good and so reliable that the rest of us can just type A backslash B.

Of course, if you're building some real time system or the sizes get to a million or ten million, you're not going to be using backslash. But that's it. That's a boundary that's growing with time. That's the wonderful thing about anything that has to do with computers. Just take a vacation for a year. My colleagues at the other end of the [inaudible] who actually do real things, they and all their friends around the world will make stuff twice as fast. You just go away. You go to the Greek Islands for three weeks. You come back and computers are faster. It's great.

Of course, I'm not telling people to just use A backslash B. Everyone here has done A backslash B. Probably only a few know what it actually did. Nothing terrible has happened. I'll come back to them and when they're yelling at me, I'll say back off. Do you use TCP/IP? Sometimes, they won't even know what I'm asking. Then I'll say are you using TCP/IP as a black box? You don't even know what's inside it?

Some of them will say yeah, I do. Even communications on your laptop between different components – I'll say do you know what it does? Most of the time, they'll say no. Here's what you need to know. It is not trivial by any means to make a reliable bit pipe to transfer bits from one place to another with unreliable medium insight. It's no more trivial than it is to solve this problem numerically. It is not trivial. All you need to know is this. Very intelligent people have thought about this problem very deeply. They've thought about what can go wrong, deadlocks, all sorts of crazy stuff, and they have come up with algorithms about which they know an incredible amount.

Part two – other people have implemented them, and these are very reliable implementations. The result is for most of us, we can just consider certain things to be reliable bit pipes. We don't have to care about how many packets were retransmitted or how the flow control went and all that kind of stuff. Like least squares – if you're doing real time control or something like that or if you're doing some computing that requires extreme reliability, then you can't treat TCP/IP as a black box.

You might ask does this calm them down? The answer is no. This makes them more angry. I mean technology here in praise of this. When you use least squares – if you've just come from 263, you've used least squares. That's just the beginning. The way you use least squares is this. It's easy to recognize a least squares problem. Sometimes, it comes to you on a platter. Somebody walks up to you and says I took these measurements. It's a linear measurement model. Help me guess these parameters. I received a signal and went to this channel – this kind of thing.

If you learn a few tricks in least squares, you can do well. If you learn how to do weight twiddling and you learn about regularization, those two alone – you are now trained and ready to do battle with using least squares as a tool. You will do really well. Weights is basically you go into a problem and someone says there's no limit on the sum of the squares. The limit is on this. You say no problem. I have these weights here. You look at

the least squares solution. You don't like what you see. You change the weights and you do it again.

In engineering, we do this all the time. It's called weight twiddling. It's a slightly derogatory term, but not bad. I'm sure that you do it in statistics, but I don't know that they admit it. Good. When I'm making fun of a department, I like to have a face to look at. They like to stick to – if you're doing real statistics, you go back in and change the prior distribution. I have to warn you, all of these lectures are being put on the web. It's weird and fun. We'll fuzz out your voice, and if the camera focused on you, we'll put the squares on it. Don't worry about it. If you'd like, we can obscure your department. We can beep it out.

I'm just going to guess that in statistics, they make a big story about the prior distribution. I bet if they don't like the way it comes out, they go back and change that prior distribution. They cover up their tracks. We do it in engineering, and we're not embarrassed. I bet they do it. Next topic is linear programming. Some of you have seen this. How many people have seen linear programming somewhere? A bunch. Okay.

Linear programming – in my opinion, it's what people should learn immediately after least squares, singular value decomposition, linear programming. If you're going to stop somewhere, that's a very good place. I'm taking about if you really want to go out and field algorithms, do real stuff – that's a great place to stop. Everybody should know about it. If you take this class, you're going to know a lot about it. Linear programming is a problem that looks like this. Minimize a linear function subject to a bunch of linear inequalities.

We're going to talk about it in horrible detail later in the class, so I'm not going to go into too much detail now. I want to talk about the attributes of linear programming. The first is in general, except for very special cases, there's no analytical formula for the solution. There is no $A^T A^{-1} A^T B$. By the way, don't confuse that with our inability – you can say a huge amount of qualitative, intelligent things about a linear program. What you can't do is write down a formula for the solution.

There are reliable and efficient algorithms and software. In fact, as a homework exercise, you will write something in the top class linear – it will be 50 lines of Mat-Lab or Python, and you'll write something that will solve huge linear programs quite well. It's no asterisk on solve – you get the solution. The computation, by the time, is proportional to $M^2 N$. That's exactly the same as least squares. If you equate rows of the least squares objective with constraints, it's identical. That's not an accident.

M is a number of constraints here, and K is the height of A .

Student: So this M – there's still that many rows in A .

Instructor (Stephen Boyd): Yes, if I write that as a matrix inequality, $AX \leq B$, yes, that would be M – this M would be that K . We spent hours discussing whether this should be M or K , and we finally – it probably should be K .

Student: What about C ?

Instructor (Stephen Boyd): X is an RN, so C is an RN, too. Actually, linear programming – it's a very old topic. Fourier wrote a paper about it. The modern era starts in the 30s, where else but Moscow. The modern era traces back to Stanford and George Dantzig. LP was something that you just talked about until you had computers, at which point LP looked a lot more interesting. That was 1948. I think a lot of people knew about linear programming. Something like this coupled with computers – that's a mature technology.

Linear programming – it looks like it would be easy to recognize, and in some cases, a problem really comes to you on a platter like this. Someone comes to you and says help me solve this problem. I want to minimize this weighted sum of the variables and I have some budget constraints. There are three problems that have exactly this form. Here's the really cool part about linear programming. You will be stunned later in the class when you see the kind of problems that we can reduce to linear programming. Things that do not look like linear programming at all we will reduce to linear programming.

What that means is they're solved. Unless your problem is huge or you have some super real time thing like in communications, then there's a sense in which you're kind of done at that point. If you do medical imaging, that's a mistake, because the problems are too big. You can't say it's a linear program and walk away. You can't do communications. It depends on the time scale. You had to adapt to things. You can't detect the bits transmitted in a packet, because that's coming at you way too fast. I would recommend that you go into fields that are in between in size.

We're going to see a bunch of tricks. Finally, I'll say what convex optimization is, because it would seem in a class with that title, one should say what it is in the first lecture. Here's what it is. It's an optimization problem – minimize an objective subject in constraints. Here's what has to happen. The function F_0 and the F_i s have to be convex. You know what convex is. It means that the graph looks like that. That's a convex function. The graph should have positive curvature.

Least squares problem has that form because if I look at the least squares objective and I look at the plot of it, it's a quadratic function squared, and basically, it looks like a bowl. If you take a slice at a level set, you get an ellipsoid. It's convex. Linear programming also is a convex problem because all of the objectives are linear. Linear functions are convex. Linear functions are right on the boundary. They have zero curvature.

One way to say convex is just positive curvature. This includes least squares, and kind of the central idea at the highest level of this class is this. If you want to solve a convex optimization problem, there are no analytical solutions. There are in special cases. We'll

see tons of cases in communications and various other places where they have special analytical solutions. You've seen one in least squares already.

In general, there isn't an analytical solution. However, there are going to be reliable and efficient algorithms for solving these with no asterisk. You will get the solution. In fact, if someone came from another planet and landed here and asked you what you're doing, there would be – it would be very difficult to make an argument that solving a convex optimization problem compared to a least squares problem was, for example, that you'd been reduced to a numerical solution, which is what a lot of people might say with a hint of – they don't like the idea. They say numerical method in a way that makes you want to go wash your hands.

The computation time for solving convex problems is roughly proportional to something like N^3 – the number of variables cubed – $N^2 M$ and F , where F is the cost of evaluating the functions and their first and second derivatives. We don't have to get into that, but the point is it's like least squares. You can basically solve these. You will solve them in this class. You'll know how to write the algorithms to solve them and stuff like that. It is an exaggeration, in fact, to say it's a technology. It's almost a technology, and every time I give this class, it's getting closer and closer.

I should say something about – we'll get to it later, but this is a very profound statement. It basically identifies a huge class – let's review what we know so far. A – everything is an optimization problem. B – we can't solve any optimization problems despite ten or twenty weeks of lectures on it. Now what I'm saying is on the positive side. It's really cool, and it's not obvious at all. It says if the objective and the constraints all have positive curvature, then this thing is just like least squares. You can solve it exactly. You can solve it quickly. Although I'm not going to focus on it, you can say a whole lot about it theoretically.

We will say some stuff about it theoretically, but that's not going to be the focus of the class. This is a pedantic way. You might prefer it if I wrote it this way. Four Theta N^0 one. You might prefer it that way.

Student: I don't know why it has to be zero one, though. Why can't it be 99 and 100.5 or negative five?

Instructor (Stephen Boyd): There are versions where there are different constraints. If I just say – it said F of αX plus βY is less than or equal to – if it's αF of X for all α and β , the function would be called sub linear. It's a different thing. This is just a definition of convex.

Student: It doesn't have any physical basis or anything. It's not gonna turn concave if you suddenly make it more than one.

Instructor (Stephen Boyd): We'll answer your question. This is not an accidental. For now, that's the definition.

Student:[Inaudible] just a statement of the [inaudible] of the line between –

Instructor (Stephen Boyd):That's what it is. It's this picture. By the way, we're going to get to this later, so you're not supposed to be understanding everything. First of all, I'm not saying that much. The stuff I am saying is kind of vague. You're not supposed to be parsing this. You're supposed to have your relaxed parsing mode on and let me get away with stuff. There'll be a time for is that really a minus sign. That's coming later.

You already know something you didn't know. Optimization problems where the objectives and constraints have positive curvature, roughly speaking, we can solve. The theoretical implication, I think, is extremely interesting. The practical ramifications of this are absolutely immense. It means you're messing on some problem in your own field, and if you get it to a problem of this form – it won't be obvious.

It's much cooler when you get to it and you turn things around and you switch some variables around. All the smoke clears. There are some functions there that it's not remotely obvious they're convex. You show they are, and then you're a hero, and it's fun. It means you can now solve those problems.

I give a talk about these things lots of places. People say that's really cool. How do you learn about this? How do you know if a problem is convex or not? I go no problem. You just have to come take my class. You do ten homeworks, each of which takes 20 hours and then you do this 24-hour final. After that, for sure you'll be quite well trained. Then they're less enthusiastic about it.

It's actually often difficult to recognize convex problems. That's going to be a theme of the class. Let me distinguish a few things there. I would distinguish problems that come on a platter convex, the ones where you have to do some work and transform them and stuff like that. Let me move on.

I want to do an example just to give a rough idea of what this looks like. For people who did 263, this will kind of tie into that. Here's our problem. I have a surface here with patches, and I have some lamps here. We're going to choose the illuminating powers on the lamps. That's going to be our variable.

The illumination on a patch here is going to be the sum of the illumination from all the lamps here. It's going to be proportional to the lamp power and then the proportionality constant is going to be an inverse square law. R is the distance between the patch and the lamp. There'll be something which is a shading effect, because obviously, if you're coming in straight on here, then you're going to get the full power.

If this lamp, for example, puts very little illumination on here, and this were below its horizon, if there were a lamp here, it would not illuminate this patch at all. This is a simple thing. The problem is to achieve a desired illumination which is given. You want to do this with bounded lamp powers. I have a maximum power. Powers cannot be

negative. I care on a log scale, because what I care about is missing the lamp power by two percent or twelve percent. I don't care about absolute. It's ratio compared to this one.

The question is how would our solve it? Let's take a look and see how you might solve it. If your question is do I shamelessly reuse material from one place in another, I can confirm that, yes. Are you asking have you seen that figure before? The answer could well be yes, you have.

Let's talk about how to solve it. The first thing you might do, and I would recommend this – the first thing you do is you say let's try some simple suboptimal schemes. One is just this – you set all the lamps at the same power. You vary it and you plot this objective. You do a search over it. There's no dishonor in it. You do that. That might work, but it might not. That's a good baseline design. You could say, well, I just learned least squares from 263. You're going to use least squares. I'm going to do this.

The objective here is not the sum of the squares. This is real life. This is the illumination engineers. Everyone uses the absolute value – the sums of the absolute values of the law of percentage error. I'm making it up, of course. You say well, good for you. We use the sum of the squares. You solve this problem. When you solve this problem, I guarantee you some of the Ps are going to come out negative. By the way, you're going to do super well. You're going to get a nice, flat illumination pattern.

What will happen is you'll look at the powers and a whole bunch of them will be negative. It turns out you can do really well in illumination with lamps that can spray negative power out. That's not good. Then the heuristic. What do you do? Here's what you do. You say if a P is bigger than the maximum power, I just set it equal to P max. If it's a negative lamp, I turn that lamp off.

Once again, you see how well you did, and you might do better than uniform – maybe worse. I don't know. Now, this is what someone trained in least squares would do. They'd say not a problem. They'd go over here and say I want PJ to be in the interval zero P max. Therefore, I'm going to add a regularization term which charges P for being away from the center of that interval. Everybody see what this is?

You start with all the Ws one, and you solve this problem. Then, you just start weight twiddling. You'll do quite well here. You'll come up with some algorithm that twiddles the weights, and how you twiddle them is totally obvious. If P comes out outside that interval, that weight needs to go up in the next cycle. If P is timid because your weight is so high, you want to decrease that weight. A couple of cycles of this, and you're going to get a very, very good solution.

Unfortunately, you might also go on to then write an impossible to read 12-page paper with pages and pages describing your iteratively reweighted illumination thing. Hopefully, you won't. You could also use linear programming. If you know about linear programming, you could actually solve this L one problem where you minimize an L one

norm. This is closer than that. Linear programming – it handles the inequalities. There's no twiddling there. This would probably do the best of all of them.

The problem is convex, so this objective function – it just looks like this. It's linear over here, and then it's an inverse over here. You look at that function and you realize a couple of things. The important part – that's what you're going to be trained on in the next ten weeks is looking for convexity. This is what we like to see.

By the way, you will see that that function is not differentiable. In a lot of other treatments of optimization, differentiability has this very high role, because a lot of things are based on gradient and derivatives, and there's no derivative there. So in convex optimization, differentiability is going to actually play a much lower role. In fact, it's a non-role.

This problem, even though it is non-differentiable here, it can be solved as fast as least squares if you know what you're doing. We might even have you write from scratch a solver for it. We could also assign it. It would be four lines in a higher-level language or something like that to solve this. That's it. This is just an example of a problem where it's not obvious exactly how to solve this, and you can imagine a lot of people not knowing.

Let's look at a couple additional constraints.

Student: Where did that curve come from?

Instructor (Stephen Boyd): I plotted it.

Student: I mean the equation. How did you know to do [inaudible]?

Instructor (Stephen Boyd): If you go back and exponentiate my other objective, you'll find this.

Student: So you just did it on both sides.

Instructor (Stephen Boyd): Yes. If you minimize something, it's the same as minimizing the X of that thing because X is monotone increasing. Let's add some additional constraints here. Here's one – no more than half the total power is in any collection of ten lamps. That would be one. Another one is no more than half the lamps are on, but you get to choose which half. I won't go into all the gory details, because for one thing, I'm running out of time, but it's not – both of these sound complicated or easy or something like that. If you don't know about convexity, you wouldn't know the second one is an unbelievably difficult problem.

In fact, you'd have to check all N [inaudible] N over two sets of half the lamps and for each one, solve the problem. Basically, everything would come down to that to actually get the real answer. You could get very good heuristics that would guess an answer, but they would not be certified as the actual correct one. This one – no more than half the

total power is in any ten lamps – that actually is convex, and it's not obvious. By week three of this class, you will know things like that.

This is actually cool, because these are things that look very similar. If I said them the right way, I could probably make you think that they are kind of the same question. I'll often do that in talks, except I don't give the answer, and then I pick some poor person. The point is these are just not obvious at all, these things.

Why is it famous? It's not famous.

Student: I mean, people have [inaudible] papers and have investigated [inaudible].

Instructor (Stephen Boyd): You mean the illumination problem? I think I probably made it up one day and then actually [inaudible] – I can say this because he's a colleague of mine at MIT. The next thing I knew, it was in his book, the famous illumination problem. You've been subjected to it in 263. That would be the 263 version of the illumination problem.

Notice that we didn't ask you to find any powers, because you would have had this annoying problem of negative powers coming out. That's selective, though. I don't think the problem is any more famous than it's been here. I don't know.

No, because this would subsume all – you have to choose which half of them are actually going to be possibly on, not which are actually on. It's exponential in the end. Let me just say a little bit and then I'll quit. The course goals and topics – the goal is to allow you to recognize and formulate problems. You should know enough to develop code for these things. You'll see it's very simple. You might not think that about seven weeks into the quarter, but you'd be shocked at how simple some of these things are.

We'll focus a bit on the theory of this, and do want to skip forward to something at the very end here. I know I'm over, so I'll just take a minute or two to do this. I've already mentioned this. First of all, the theory of convex optimization is more than 100 years old. The first paper on this is from 1890 or something like that.

In fact, the idea is traced earlier. By 1970, it was very well known, the theory. There are some points here that I think I don't have time to go into, but you can read about this in the first chapter. What I do want to say is something about why it's interesting now, if this is 100 years old as a mathematical subfield. What makes it interesting now?

What I can say is since about 15 years ago, people have been finding applications in lots and lots of areas, and it just sort of starts coming up. It may not particularly help you in some area to know that a problem is convex, but it sure doesn't hurt. It might in some cases allow you to solve it. This was sort of not the case, with the exception of least squares and linear programming.

These have been widely used since the 50s and widely applied since the 1950s. Basically since the 1990s, there were a lot of things found in areas like machine learning and statistics and control and signal processing and stuff like that. You get a nice, positive feedback loop because once more people start discovering these problems and start writing papers on, for example, the famous illumination problem – once they start appearing, what happens is people who write the algorithms see two of these and say somebody should write a code for the illumination problem.

Then, they write a beautiful code, well tested, numerically stable, hopefully open source and all that, and then everyone can now solve the famous illumination problem. Only then do people realize that there never was such a problem, hopefully. I quit here, and I'll see you, in theory in two days, but more like two months or something like that.

[End of Audio]

Duration: 80 minutes