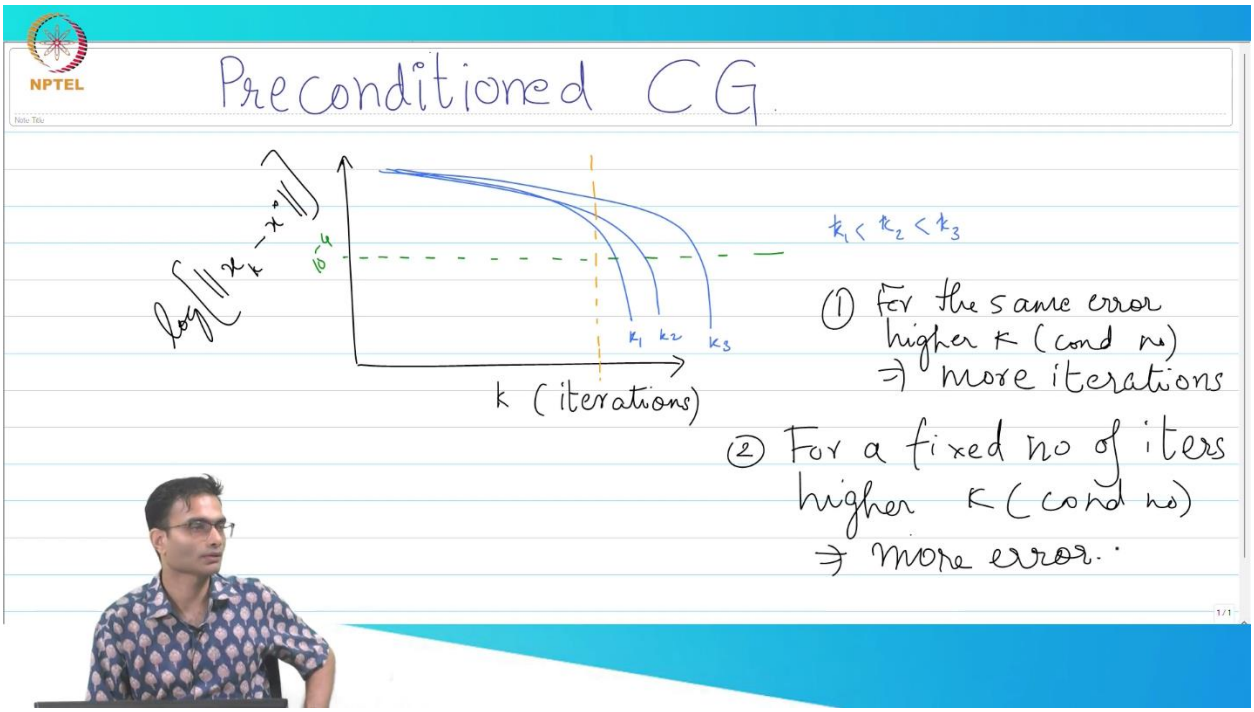
Course Name: Optimization Theory and Algorithms
Professor Name: Dr. Uday K. Khankhoje
Department Name: Electrical Engineering
Institute Name: Indian Institute of Technology Madras
Week - 06
Lecture - 45

Preconditioned Conjugate Gradient - Part 1

So, the last time we looked at the conjugate gradient method, we saw that it blew steepest descent out of the water, right? So, today's class is to show you that all is not hunky-dory; there are places where it runs into trouble, okay? So, let us begin. So, I am going to draw an approximate graph, okay? So, let us say that we are very happy with our CG method and we want to study how it behaves with the condition number. We know that it is going to get worse as the condition number increases, but how much worse, right? So, there is MATLAB code on the website that we discussed last time, which allows you to manually adjust the condition number and make your problem easier or harder. Then, you can plot this graph, which shows you how close I get to the solution as the iterations progress. So, what you will find is something like this: This is, let us say, κ_1; then you will have κ_2, and after that, you will have κ_3. So, κ_1 is something like this: we do not need the exact details, but the condition and behavior of the error are something like this.



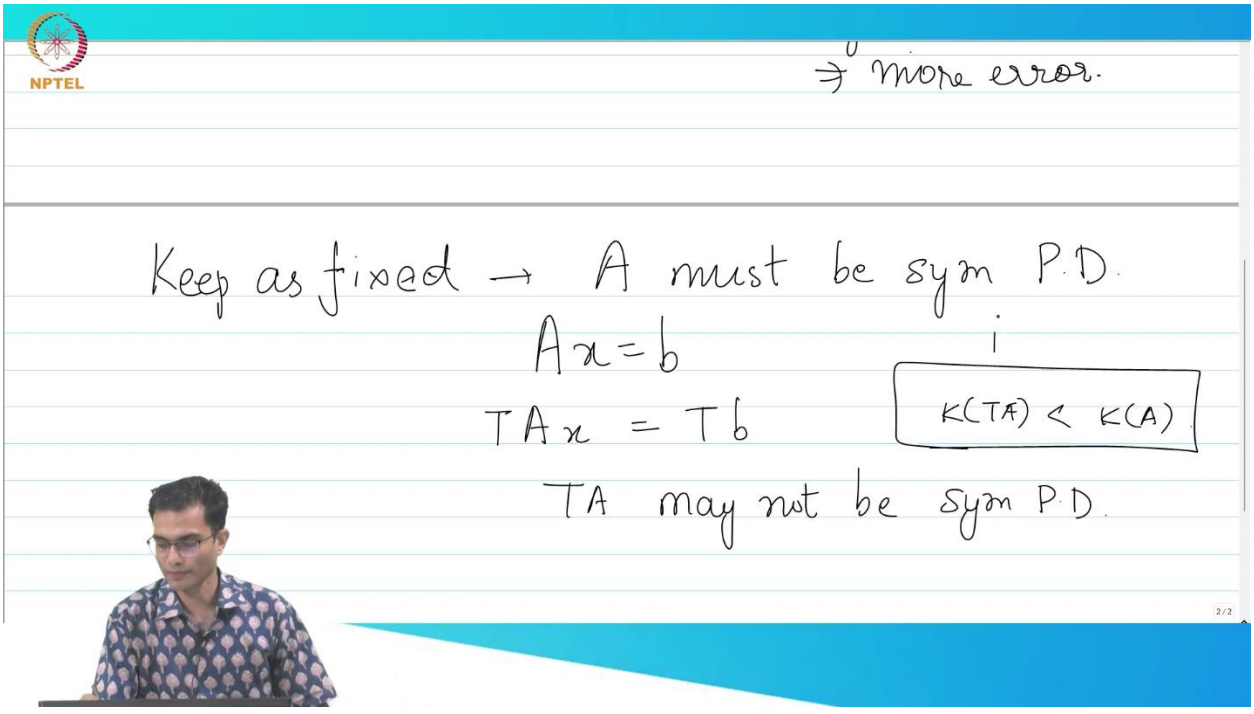
Okay, and this is actually in the log. So, what are the few observations you make from here? For example, if I were interested in a certain threshold to stop my iterations, or a specific accuracy, that would be represented by a horizontal line in the graph or a vertical line on the right. So, let us say that I want to continue as long as I receive an error below this threshold, τ; I am happy with my solution, right? It can happen that I do not want an error of 10^{-16}; I want 10^{-4}, which is

good enough for me, okay? So, this τ could be, for example, 10^{-4}. Now, what is our very obvious observation? To get the same error, I need more iterations if the condition number is worse, right? So, we can say that for the same error, a higher condition number implies more iterations.

This is slightly confusing because it should be κ; this is κ, not k. And if I were to have fixed computational resources and wanted to devote only a limited number of iterations to this problem, that would result in a vertical line on this graph. So, something like this is my vertical line. So, what is our observation when looking at this vertical line? If I have a fixed number of iterations, the one with a higher condition number has a higher error, right? So, for a fixed number of iterations, a higher κ, which is the condition number, implies a greater error. Those are the two very basic observations I can make by looking at this graph.

So, obviously, people looked at this and said, "Hey, is there a way to improve this?" So, that is what leads to the idea of the preconditioned conjugate gradient method. So, let us see what it means. So, we are going to stay within the realm of the conjugate gradient method, which means that I am constrained to use my CG routine. Therefore, what is the limitation on A? What can I not sacrifice for A? It is only when the matrix is symmetric positive definite that I can use the CG method I have studied. So, I am going to keep it fixed as a must to be symmetric positive definite, okay? Remember for CG what I was trying to solve: $Ax = b$.

We wrote that neatly as an objective function in the form of a quadratic expression. Now, let us stare at this a little bit; it has a bad condition number. I still want to use the CG routine to solve this. What could I possibly do? Multiply with another matrix. Multiply with the other matrix, then what will happen? So, that is a good idea.

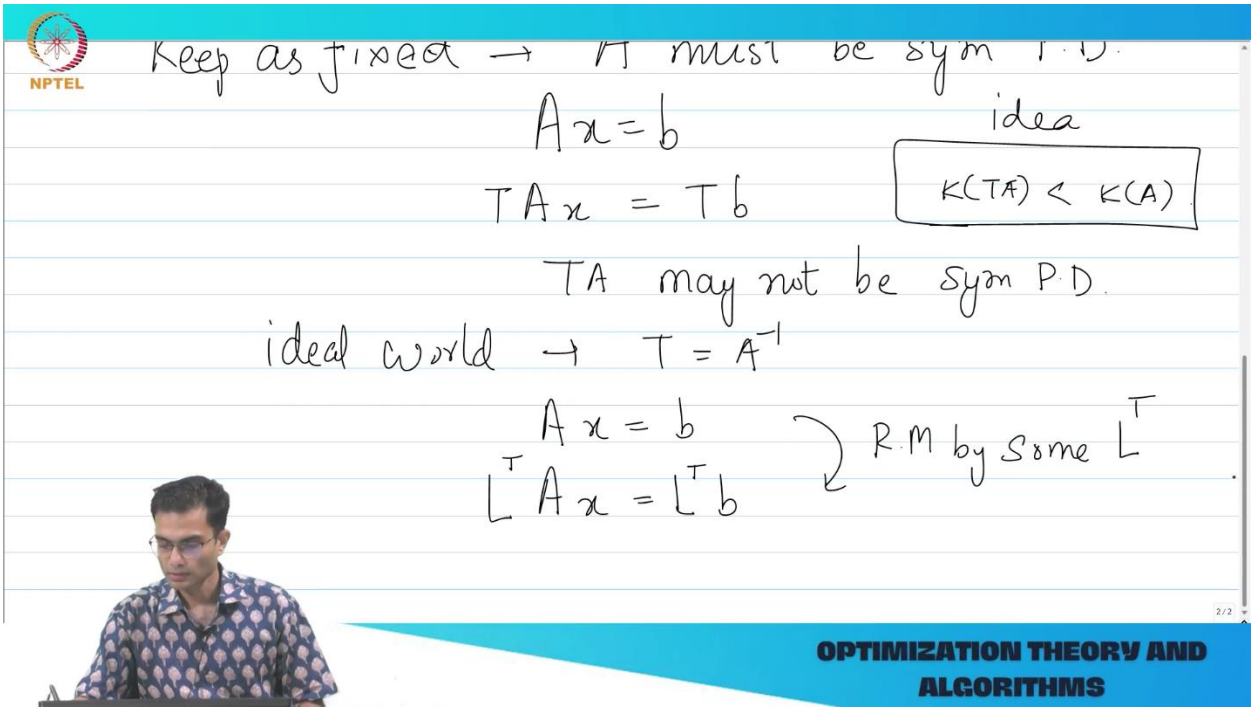


So, supposing I multiply this by some matrix, let us say T, and then multiply both sides by T. Therefore, $Tx = Tb$, okay? The motivation is that TA may have a better condition number than

simply A. So, $\kappa(T)$ is greater than $\kappa(A)$. Is that a good idea? The answer is literally staring you in the face. Is that a good idea?

No. I see some head shaking; why not? I mean, if the condition number is less than that, it's great, right? Why is it a bad idea? The matrix cannot be input into my algorithm unless I ensure that TA is also positive definite, which is not guaranteed, right? So, TA may not be symmetric positive definite, but the motivation you have here—this idea—is essential; it is the only lever we have in our hands. If we can somehow improve the condition number, then there is some hope. So, this looks like a minor roadblock on the way. Now, let us pretend for a minute that the symmetric positive definite conditions allow us to relax.

Let us pretend for a moment. Let us live in an ideal world. In an ideal world, it would seem very impractical, which is why you may have a difficult time guessing it. What is the ideal temperature? An inverse right. What is the best condition, or in other words, what is the best condition number I can obtain? 1 is possible only if $T = A^{-1}$, right?



So, in the ideal la-la land, $T = A^{-1}$, and in this ideal world, this is ridiculous because if I already knew A^{-1}, there would be no problem left, right? (The sentence is grammatically correct as it is.) No changes are needed. However, that gives us an idea of what we can do. The trick I am going to use is to achieve two things: a reduction of the condition number while maintaining positive definiteness. So, supposing I take $Ax = b$, and as suggested, let us multiply by some matrix.

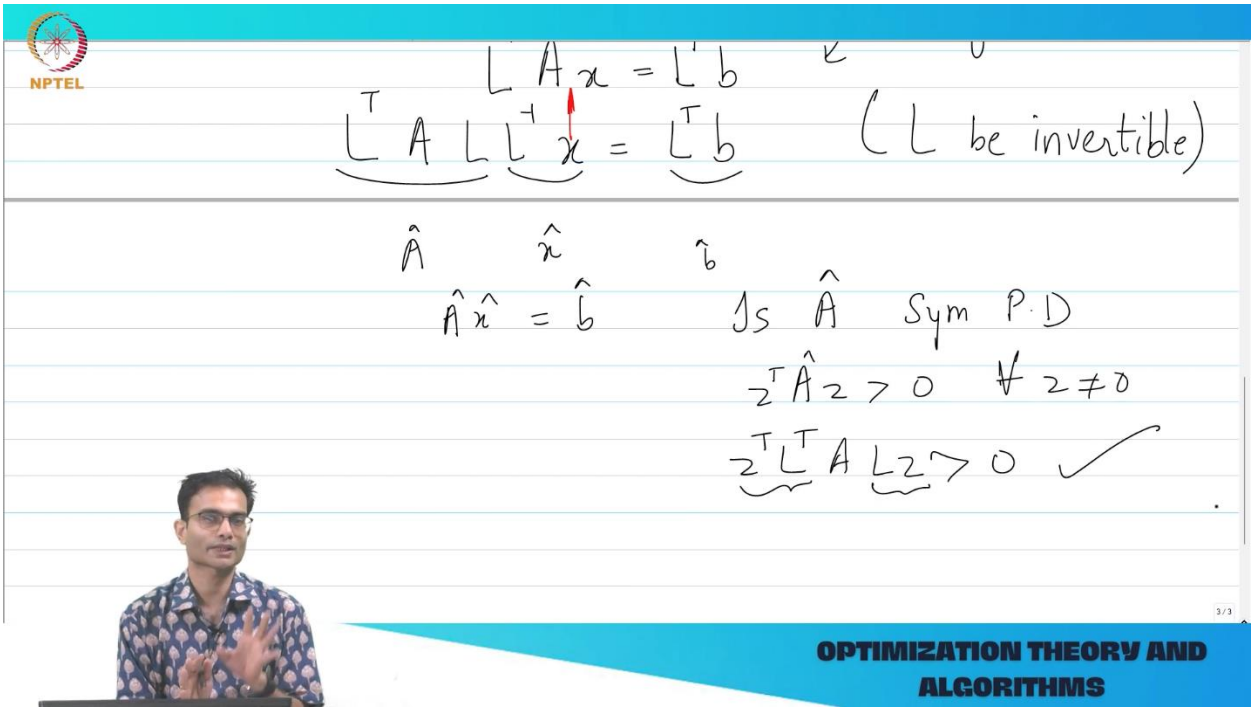
Instead of saying T, I am going to create a new convention: I will multiply by a new matrix, L. Therefore, $Ax = L^T b$. Okay? So, I am performing a right multiplication by a new matrix, where L^T could be my T, okay? I am still facing the same problem as before: I do not know whether it is positive definite or not. Is there a clever trick I can use? Let's assume that L is your favorite invertible matrix. Is there a small trick we could use to shift this or transform it into another

problem where I have a symmetric positive definite matrix? So, let me give you a hint: Can I do something here? It is related to the question that was just asked.

What property of L was inquired about? Is L invertible? So, what can I possibly insert here: L and L^{-1}? So, let us try that again. So, I am going to get L^T (not $LAL^TAL^{-1}x$) is equal to L^Tb, right? So far, there has been no cheating; I just need L to be invertible. Now, what is next? Is it looking closer to what I wanted it to be? I need to group some terms neatly. So, if I group it like this and that. So, I am going to let us call this $\hat{A}$, let us call this $\hat{x}$, and let us call this $\hat{b}$.

Therefore, I get that $\hat{A}\hat{x} = \hat{b}$. Now, is A a symmetric positive definite matrix? To check if a matrix is positive definite, what do I need to do? I need to quickly compute the transpose of $\hat{z}$; z should be greater than 0 for all $z \neq 0$. Is that the case? The sentence "It is." is already grammatically correct. If you need a different form or context, please provide more details! It is correct because you can immediately see that this will be z^TL^TALz, and I can combine this; it is already greater than 0, right? So, this is symmetric positive definite—perfect, right? I do not care what $\hat{b}$ is; it is whatever it is—some new vector $\hat{b}$.

So, I have kind of shifted the problem; I have not solved it. I have just kicked the football a little further down the field. What have I kicked it into now? What is the new problem that I need to address? Give me a good solution; otherwise, how do I proceed? Okay. So, let us see what we can do. So, let us assume this.



Let us defer the choice of how to determine L for now, but we have a way to improve the condition number of $\hat{A}$ and solve the problem. Once I solve for $\hat{x}$, how do I retrieve my x? Multiply by L, right? So, once I get it, I will proceed. So, $L^{-1}x = \hat{x}$. So, if I can solve this problem, I just need to multiply by L, and once I get my x back, I am done, okay? Now, the part of how to get L, we will get to. As I mentioned, a very, what should I say, important question is do I need to re-derive the entire thing?

I could solve supposing I gave you an L; you could take this guy and feed it into a CG routine and get the solution, right? Now, the question and the little more clever thing to do is you already have a piece of code written for the CG method, where the inputs were A and b, right? That is what I had, and later I discovered that the condition number was bad. Now, I am asking if I can be a bit more clever. So that I can use that code as much as possible without having to recreate all the variables in terms of the hatted quantities, okay? So, you will find a surprisingly simple answer: there are actually very few modifications you need to make in order for this to work. So, we will work that out. Having worked that out, we will see a very clever choice for L that will come up—there are lots of.

It's a field in itself. So, this L, for example, is what is called a preconditioner because it is conditioning the preconditioner, right? It is changing the condition number. There are lots of methods we will talk about—one possible way of doing it—but has everyone got the motivation over here? We did some very simple linear algebra tricks to make sure it's symmetric positive definite, that's all.