

Course Name: Optimization Theory and Algorithms
Professor Name: Dr. Uday K. Khankhoje
Department Name: Electrical Engineering
Institute Name: Indian Institute of Technology Madras
Week - 07
Lecture - 51

Checks for positive definite matrices

So as far as the Newton method goes, there is, it's pretty straightforward. You get the, so if you were to summarize the Newton method, what would you do? You have, let's just summarize the Newton method. So supposing you were writing an algorithm for Newton method, what would you do as step one? How do you start? How do you start? pick a starting point. This could be randomly generated or it could be from some intuition about the problem ok. So, next what do we do? I need to know in which direction to walk. Therefore, I need my p_k , p_k is going to be equal to

Summarize NM:

- 1) Pick x_0
- 2) $p_k = -B_k^{-1} \nabla f_k$ ← dir.
check B_k for P.D, then
- 3) Compute $\alpha_k \rightarrow$ B.T. L.S.
- 4) Go to x_{k+1}
- 5) Check $\|\nabla f_k\|$

OPTIMIZATION THEORY AND ALGORITHMS

$$p_k = -B_k^{-1} \nabla f_k$$

I need to compute this may be expensive, but if I am doing Newton method means I am computing this. So, I got my vector. So, this was my direction. What do I do next? One answer is find out α .

Is there a more precise answer or let me put it differently, are not you forgetting something? Not exact line search, exact line search may not be possible. In exact, before you get on to the bus you need to buy the ticket, what is the ticket? Check if it is a descent direction, in other words how do I check if it is a descent direction? Is it positive definite? If B_k , if B_k is not positive definite, p_k is not going to be descent direction right. So, then check B_k for PD and actually first

you check for this and then you compute this, otherwise it is a waste of time. So, I got my positive definite check, supposing it is done. Next what do I do? α .

So, I should not say pick, I say compute. Most popular way of computing α is for example backtracking line search. It is an example of inexact line search, right. So, I got this. What do I do next? I have α , I have p .

Good. What do I do? Go to x_{k+1} , right. and I can keep doing this. What do I need to check to stop my iterations? How do I know whether I am done? Well, the simplest thing to do is we check this right, that is my signature. You could check $\|x_k - x_{k+1}\|$.

That will, that is not exactly the signature. That is telling you, supposing you enter some very shallow valley, then it may end prematurely. You need to check this and if it is good, you end. If it is not good, which step do you go to? Step 2, correct? So, end is one option. Step 2 is another option.

So, everyone has a very clear picture of Newton method. In terms of coding, supposing I give you like 10 minutes to code this, you have already seen the code for steepest descent, you have seen it for conjugate gradient. Do you think this is less difficult or more difficult? It is about the same difficulty. There are no new steps over here. The only thing different is how you calculate p_k .

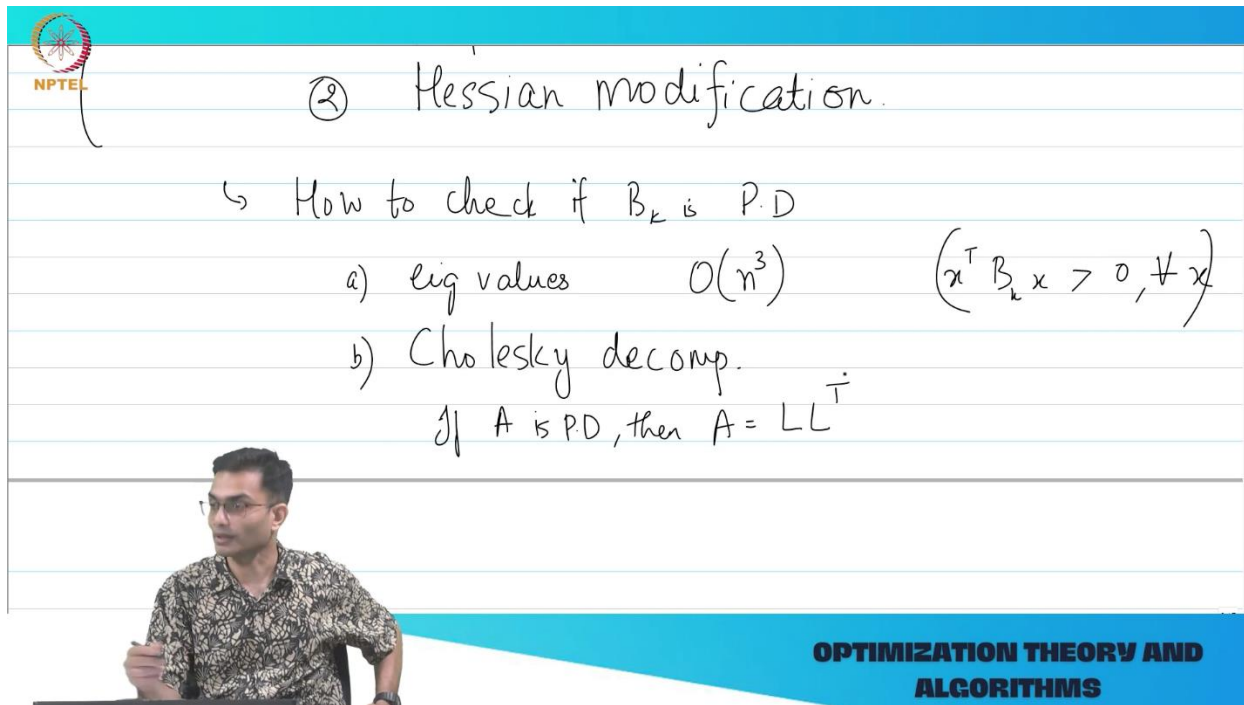
You have to do, you have to spend some, do something clever to calculate B_k , right. Most likely if you do not know the function analytically, you cannot calculate B_k analytically, in which case what would you do? Finite differences, ok. Everyone remembers finite differences, how do I calculate second derivative of a function? Taking differences, right. So, that is how you do it. So, what do you think is the weakest link in this? You know, when you have a relay race, you need to identify who is the weakest runner, right? Because that determines your outcome.

So who is the weakest runner in this? Step two, right? This guy. Because I only wrote one, I wrote an incomplete if condition. I wrote if B_k is positive definite, then do this. If it is not, what do I do? This is the next topic that we want to talk about. If B_k is not positive definite, then what do I do? Based on what you already have learned so far, if you are writing this code, what would you do? The most common sense thing to do, I should not say common sense, I would say the one of the easiest solution is to ditch the Newton method and switch to something you already know which is steepest descent or steepest non-linear CG as depending on what the function is.

I could just switch into that because there I do not need to check whether it is positive definite or not. If I take $-\nabla f$, it is always going to be a descent direction. So I could do that and after a few steps I could switch back into Newton method, right. So people do that. So they will have a code that switches between this.

A disadvantage of course is, what would be the disadvantage of doing this? You will lose the quadratic rate of convergence, but in this case you have no, it seems like you have no choice. So, one possibility is drop down into steepest descent or CG or non-linear CG ok, depending on the problem this is. Now turns out there is another, there is entire you know family of tricks that people do. So, the book has a long discussion of different, different tricks to do. So, what I am going to do is I am going to write down tell you about one of them ok and one of them is Hessian modification ok. So, this is if you think in terms of surgery Step one is like non-invasive surgery.

Step two is like invasive surgery. We're gonna cut open the patient and do something a little bit nasty, okay. Now as the word says Hessian modification means I want to modify it so that what happens, so that it becomes positive definite. On the other hand I do not want to modify it in such a way that I am now solving a different problem altogether, right. So I want to introduce some small changes to the Hessian so that I can do this Newton method, ok.



NPTEL

② Hessian modification.

↳ How to check if B_k is P.D

a) eig values $O(n^3)$ $(x^T B_k x > 0, \forall x)$

b) Cholesky decomp.

If A is P.D, then $A = LL^T$

OPTIMIZATION THEORY AND ALGORITHMS

So there are some tricks by which of all you can figure out, how do you know before we get to Hessian modification, I forgot to answer one, I mean you should have asked me, how do I check whether or not a matrix is positive definite? Ok, so before, so how to check if B_k is, we should have actually done this before the recipe for what to do after the check fails. okay. So one is eigenvalues. Does that look like a good idea? It is an expensive idea. It is because eigenvalue decomposition, remember I need to calculate all the eigenvalues.

It is not enough for me to just calculate one or two eigenvalues. I have to calculate all of them and what property should all of them satisfy? Should be strictly greater than 0, right. So there is no inexpensive eigenvalue decomposition over here. I am stuck. Eigenvalue decomposition, order n^3 , n^3 , if n is a large problem, this is infeasible, right.

I mean, what else can I do? $x^T B_k x$. I should check whether this is greater than or equal to 0. But what was the remaining part of the statement? Good luck with that. So, here is where again linear algebra comes to our help and you had seen this technique last time, this is we make use of this Cholesky decomposition, ok. So, what does the Cholesky decomposition say? That if A is positive definite, then it can be written as LL^T or CC^T , whatever notation you want.

Where L was what? Lower triangular, right. L is lower triangular, ok. It is not, there is actually one more condition on it, it is lower triangular and the diagonals are positive, there is one more condition. And this Cholesky decomposition theorem actually goes both ways, says if A is positive definite, then I can write it as LL^T and the other way is that if A can be written as LL^T

where L has this property, that means A is positive definite. Both ways it works, right. So, if $A = LL^T$, then A is positive definite, right. So, option B for checking whether or not B_k is positive definite is simply attempt a Cholesky decomposition of your matrix B_k . If you succeed, then you can conclude that your matrix is positive definite, ok.

Now that is again, the computational cost of doing a full Cholesky decomposition is non-trivial, right. So, this is again not one of the, not a very nice solution to try, ok. So, now with this in mind, there is going to be a third method that I will talk about now. This is expensive, I mean sorry this is expensive. The third method is actually one of the most inexpensive and elegant methods which I have come across.

In fact, this method is not mentioned in the book. So, I am going to spend some time over here talking about it in detail, ok. Not surprisingly it is named after some Russian. So, it is called the Gershgorin. Anyone has heard of this? One person. Nobody else? Okay, you have heard of it, right.

So this, those of you who have done linear algebra may have heard of this theorem, okay. Again, most of the tricks of optimization come from linear algebra, right. So this is one of those things. And it is, so let me just tell you what this theorem states, okay. So, assume A is and it is a very general theorem.

If $A=LL^T$, then A is P.D.

c) Gershgorin disk thm. (L.A.)

Assume A is a complex matrix.

Disk $i \rightarrow$ Centre A_{ii}
 Radius $R_i = \sum_{\substack{j=1 \\ j \neq i}}^n |A_{ij}|$

Thm: Every eig value of A li

NPTEL

OPTIMIZATION THEORY AND ALGORITHMS

So, take a complex matrix, ok. Now we need to make disks represent this matrix as a collection of disks. So, disk i and imagine you have to imagine the Argon plane, right. So, this is the real axis and this is the imaginary axis. Everyone is familiar with the complex plane over here: real and y is the imaginary part. So, the center of the disk is A_{ii} , that means the diagonal element of that row or column, whatever is the center.

So, you look, this is a complex number. So, it will be somewhere in the complex plane, ok. What is the radius? I take this, this summation of, what is this in English? Is the row sum of absolute values of the i -th row. i is fixed, j is what I am iterating over, but what am I skipping? I am skipping the center point, right.

So, I have got this. So, basically what you have is a disk whose center here is this, is A_{ii} , and this radius is r_i . How computationally expensive do you think this is to evaluate all the disks? It is no cost, it is just adding numbers, right? There is no order n^3 anything over here. And here is the lovely theorem. The theorem says that every eigenvalue of A lies within at least one disk, it is really very, very powerful, right? Because it is not, it is exactly what we want in the sense that I do not want to know the exact eigenvalues, right? I am not going to do any particular calculation with the eigenvalue. I just need to be assured what? That they are positive.

So this theorem is going to help me because it is telling me the bounding box or the bounding disk within which an eigenvalue is, right. So, let us sort of particularize it to our case. We are dealing with, let us say we are dealing with a real value problem, right.

So, our matrix B_k is real value. So, that implies that where are all of these centers of the disks? They are all on the real line. So, if I can draw one real line over here, this is A_{11} , this is A_{12} , 2, whatever, these are all the centers over here. Now, this row sum obviously without the centers is also a real number. So, what am I going to get? I am going to get these disks, right. So, once I compute all of these disks, what do I have to make sure? That there is no disk which is kind of creeping into the negative x-axis, right.

If this happens, then there is a possibility that there is an eigenvalue here, but it is not a guarantee. It is saying that an eigenvalue is contained in this disk, right. So, this Gershgorin Disk Theorem is telling us, giving us a very clear picture without any computational cost. Look at these disks. If all of these disks lie to the right of the origin, then there is no chance that you will have a negative eigenvalue.

If you do not have a negative eigenvalue, you are in business because then all eigenvalues are positive and your matrix is positive definite. Okay, so do I need to check every disk or should I check only the leftmost disk? The radius of the disks can be moved, right. So I can have a disk on the right with a larger radius, right. So I have to check that, okay. But this is like two lines of code to check it, right.

So it is very easy to check all the disks, right. So now, so what are we sort of, let me just quickly summarize what we did is that the Newton method works if the B_k is positive definite. So how do I check whether B_k is positive definite? There are three ways, expensive to begin with, either do eigenvalue decomposition, expensive. Do Cholesky decomposition, also somewhat expensive. Third is look at the Gershgorin disks and make sure that all of them are to the right of the origin.

So this tells us whether we are in business or not, right. Now if let us say some eigenvalues are negative, right. If eigenvalues are negative, we have two choices: either switch to steepest descent, conjugate gradient, whatever, or do a Hessian modification, right. So now let us have a look at Hessian modification.