Chapter 2.1b - LU Decomposition

15 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=1FSHG18KWVQ
https://www.youtube.com/watch?v=1fshg18KWVQ

SUMMARY

LU decomposition is a method used to accelerate the solution of the equation \(Ax = B\), particularly when \(A\) is a large matrix and \(B\) varies. By decomposing \(A\) into a lower triangular matrix \(L\) and an upper triangular matrix \(U\), the computational cost of solving the equation is reduced from \(O(n^3)\) to \(O(n^2)\) for each subsequent \(B\).

- *LU decomposition allows for efficient solutions of \(Ax = B\) by transforming it into \(LUx = B\).*
- *The process involves forward substitution to solve \(Ly = B\) and backward substitution to solve \(Ux = y\), both of which are \(O(n^2)\) operations.*
- *The initial LU decomposition requires \(O(n^3)\) operations, but this is only needed once for a given matrix \(A\).*
- *If a zero pivot is encountered during decomposition, row swaps are necessary, tracked by a permutation matrix \(P\).*
- *In practice, LU decomposition is implemented in Python using libraries like SciPy, which provide functions for performing the decomposition and solving triangular systems efficiently.*
- *The method is particularly valuable in scientific computing, where rapid solutions to linear equations are essential for large-scale problems.*

we're now ready to talk about lud decomposition which is a a very classic method for accelerating the solutions to a equal to B specifically if you have a a matrix a that is quite large but that you're going to use solve ax equal B over and over again with different bees then lud DEC composition is one of the classic methods for doing it reducing your computational cost from order n cubed down to order n squar so we

want to exploit that speed Advantage when we can and many of the algorithms for solving linear algebra Tech ax equal B solvers actually use Lu underneath the hood in terms of the algorithm they're actually using again this is part of the datadriven modeling scientific computation lecture series and so let's get into talking about what is an Lu decomposition so the idea behind L DEC composition is pretty simple it's

the idea thinking about a matrix a that you can decompose into a product of L * U where l so here's the Matrix a it's full in general where L is lower triangular and U is upper triangular so one of the questions you can ask immediately is why would I want to do such a thing and it's actually all down to operation count so if I have this L decomposition and we have to talk about what would it take to get the L

decomposition but once I have L andu then I can think about A xal to B as being solved by Lu that's what a is now xal to B but then I can Define ux to be Y and then if that's y I have l y equal to B so the first thing is I can

solve for y right here but notice L itself is in lower triangular form so all I have to do to solve this equation is forward

substitute it's already in a form am meable to substitution forward substitution so this operation is only going to be order n squ once I get y I can throw it to here so now I have y and if I want to solve for x I have to remember that U itself is upper triangular form so in Upper triangular form that means all I have to do is back substitution in fact in the last lecture I showed you what gination TRS

tries to do is make The Matrix a into upper triangular form so that all you have to do is back substitute so this is again an order N squared operation so essentially what you're looking at here is the advantages are if you have an L decomposition I can just do a forward substitution followed by a backward substitution that's is what it results down into once I have the decomposition and both of these operations here are order n squar each

so that's the key piece notice that it once I have this Lu I am now solving this ax equal to B in order n squ two order n squar operations versus Gussy elimination requires me in order n cubed and remember if n is large right even just think of n as a million by million Matrix then this is a million times faster to get the solution because I'm not doing order n cubed I'm doing two order n squs

so the way to think about this the advantages are the Lu is order N squared whereas gination if I do it it's order and Cubed all I'm going for here is speed and this is one thing we have to think about a lot in modern Computing is how do I solve problems as fast as possible because if you're not solving it rapidly you're kind of it's it's it's sort of against the ethic of what the Computing computation is trying to do so

a lot of scientific Computing isn't just about getting the right answer it's also getting the right answer as fast as possible so Lu decomposition is one of the steps in this direction of solving ax equal to B as fast as possible in fact solving it much faster than galum relation or at Le that's what we're trying to do so the question is how do you actually perform this L composition so what I want to do is

quickly walk through what we did did in gination because once we understand that we're going to follow a very similar structure in Lu decomposition so if you remember L in in gination we formed this augmented Matrix we find a pivot and we try to zero out everything in below the pivots so that we can essentially make this upper triangular form so we Define a pivot we try to make everything zero so in this case right you can subtract

equ two from one three from one and now you get this system now when you have this you can take that third equation divide by two and then subtract the third equation from the first equation and then you get this form here now presumably you already saw this in the last lecture so you've already we walked through this a little bit more slowly but this was the gussi elimination process this is what we're trying to do to get ourselves into an

upper triangular form where all I have to do now is back substitute so in some sense this looks a lot like the U Matrix were kind of construct but it cost me order and Cube to get it so let's talk about what Lu tries to do and

how to set up this same kind of Galaxy elimination structure with this augmented Matrix so here's how it starts I take my Matrix a and I multiply it by one which is the same okay so this is

what it looks like this is multiplying The Matrix a by one here's the one here's the Matrix a I'm going to do an L de composition and notice what I'm going to do here it's going to follow a lot of the same structure of Gan elimination but now I have this m one in front of it that I'm going to start to modify so here's what you're going to do you're going to start with this process

and you're going to do the same kind of thing you did before which is you're going to start to decide you're going to try to make the Matrix a itself into some kind of Gyan eliminated form so you going to pick the pivot which is the four and try to make things zero below it and then we're going to do this is you're going to say what would it take what do I have to do to the first

equation so that I can eliminate this minus 2 and this one so from the first equation here I multiply by negative a half right and if I multiply this four byga a half it's -2 now I can subtract these two equations and how do I get this four and one to eliminate I'd multiply the top equation four by 1/4 and then 1/4 * 4 is 1 and I can eliminate this equation so those factors of -2 and 1/4 now show up here was my

identity Matrix but now I get this minus one2 and 1/4 those are what we're eliminating these two positions now come in these two positions and when I do that elimination I now have 0 0 here okay so now I've done one first step which looks a lot like galaxian elimination except for now I am tracking what I had to do in order to make zeros happen below the diagonal now that I have zeros below the diagonal I go to

the next diagonal which is this -2.5 and I asked the question what would it take to eliminate make the -2.5 and 1.25 okay I would need now a factor of negative a half right so if I multiply that -2.5 by negative a half I get 1.25 and now I can subtract these equations so that negative 1 half goes here and now it allows me to subtract the two equations to get this equation here now notice what I have in this I now have an

l and a u i have them in upper and lower triangular form in fact the main thing you have to remember in L decomposition your pattern for elimination has to be consistent throughout the entire process in other words you can't say this time I will subtract the first equation from the second second from the third and then change it around next time whatever you decide to do whatever the pattern is and subtract you have to maintain that

pattern throughout and when you do that you have your L and you have your U okay so now right the L is lower triangular form U is upper triangular form I now have the two matrices so if I want to solve ax equal to B all I have to do is forward substitute and then backward substitute two order n squ operations okay the cost however to get the Lu in the first place is order n cubed so I don't get around this issue

of having to pay order and Cubed but I only have to do it once so in other words once I have a matrix a I only have to do the L Lu decomposition once and if I have different right hand sides B then I already have the Lu

decomposition of a and I can just now solve everything in order n s time so that becomes important because now it's much faster than if I'm trying to do gy elimination every single

time especially when you go to very large scale matrices which we're going to do in in as we progress forward now there are some issues that we have to take care of the first one is regarding pivots when we do x equal to B what happens when we come to a zero pivot well we have to trade we have to trade out some of the rows of the equations to to be able to proceed forward well same thing in lud

composition in other words it could be that when I'm doing the L de composition I have a zero pivot that means I have to tr rows but notice when I do the L decomposition of the Matrix a by itself the B is no longer there right my when we do gination the augmented Matrix has B so when I switch rows the b gets switched but here when we do Lu the a matrix is just if it's switched in the a

I have to tell B about it so the way to think about this is if I have to switch rows there's a permutation Matrix P let's say that switches rows so for instance this permutation Matrix here would switch the first two rows okay so the permutation Matrix is something important which is if you have a zero pivot you might have to switch your the order of the equations and P is going to keep track of that and the idea is that

P * a in other words the perturbed the permuted a is what we're going to actually do the L composition on so always have that in mind in fact that shouldn't be P luu it's just Lu so the PA is Lu and that's what we're doing okay so how would we do this in practice in solving ax equal to B so first of all in Python let's just talk about a regular ax equal B solve so here's a

matrix a and a matrix B then all you have to do is say MP linear algebra. solve a comma B that's it so that takes care of solving ax equal to B and of course the operation count for doing this here is going to be ordering cub you haven't done anything special and so generically if you do this it's going to cost you order n cubed now let's try to do an Lu decomposition first and here's how the

Lu decomposition works so first of all from scipi linear algebra we want to import Lu and solve triangular I'll tell you why we do this in a moment so first you do the L decomposition with the Lu command Lua and it gives you back three matrices p L and U so now you have your permutation Matrix the L and the U now it could be that the permutation Matrix you can always look at it and see if

it's an identity Matrix which means it didn't have to switch any of the rows which is great but if it did switch the rows then your new right hand side B is let's call it PB is essentially P * B in other words what it's going to do p is going to switch the rows of B in order to handle what happened here on the L de composition and now we can solve it now remember there's a two-step process the first

step is y is equal to l comma B so in other words we would typically solve you know L * Y is equal to B but what this does is very important solve triangular says I already have it in triangular form so it immediately does

forward substitution it knows that L is a triangular form if if you don't tell it this it will do the full solution gxy elimination it cost you

order n cubed so this here is guaranteeing you order n squar and solving this which is already in lower triangular form and then you do solve triangular U comma y so now you get X which is ux equal to Y you solved for y cost your order n cubed and now you solve for x cost your order and Cubed again if you don't put solve triangular then what it's going to do is if you just say solve it won't know to look to

see is it in Upper triangular form already now mat lab is interesting in the sense that the first thing mat lab does is actually checks to see are you in upper or lower triangular form or can you be in that form so it would you don't need to do this in mat Lab Mat lab does it automatically it looks for that but not in Python you have to be explicit and tell it to do this so this

is how you would solve it with python code to accelerate your ax equal to B solve and the idea here is that you would actually only do this command Lu one time okay so you would do this once for some Matrix and that's it now from then on with that Matrix you can solve it in order n s for any new be's that you might have so that's the idea so we went from galum ination which is the maximum cost

you're going to pay which is on and Cubed for solving ax equal to B to Lu decomposition which gives you a pathway forward to faster ax equal to B solves you still have to pay the price of order n cubed once but from then on it's order N squared and so you saved yourself especially for large matrices quite a bit of computational time and so the cud decomposition you can imagine has become very important in practice for

accelerating scientific Computing and for accelerating ax equal to B remember ax equal to B is like your most common thing that we're going to do in in basically any kind of numerical computations so so we want fast algorithms especially as we go to larger and larger scale systems