Chapter 8.6 - Boundary Value Problems: Direct Solve and Relaxation

11 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=EVAR1AKD8MW
https://www.youtube.com/watch?v=EVAr1akD8mw

SUMMARY

The discussion focuses on an alternative strategy for solving boundary value problems (BVPs) using direct solve or relaxation techniques, contrasting it with the shooting method previously employed. The approach involves discretizing the problem using finite differences to create a system of linear or nonlinear equations, which can then be solved simultaneously rather than iteratively.

- *The direct solve method simplifies the process of solving boundary value problems by using finite differences to approximate derivatives.*
- *Instead of shooting from one boundary condition to another, this method constructs a large system of equations that incorporates all boundary conditions at once.*
- *The discretization process involves using Taylor series expansions to derive formulas for first and second derivatives, leading to a system of equations in the form \(Ax = b\).*
- *The boundary conditions are embedded in the system, allowing for the solution of interior points while knowing the values at the boundaries.*
- *For nonlinear systems, relaxation methods are employed, requiring a good initial guess to ensure convergence to the solution.*
- *This method is computationally efficient as it solves the entire system of equations simultaneously rather than iteratively, as seen in the shooting method.*
- *Understanding the transition from differential equations to the matrix form \(Ax = b\) is crucial for implementing this approach effectively.*

We're now going to talk about an alternative strategy for solving for boundary value problems. So this what we did previously is we essentially used the techniques of time stepping and we kind of commandeered that for doing spatial stepping by shooting across in the shooting method from one side to another making sure we start with one boundary condition launch across and enforce the other boundary condition on the other side. And so we used a lot of the same ideas that we did for time stepping from you know things like rungakata to do this but now what we want to do is change the strategy altogether and you do what's called a direct solve or relaxation techniques.

So what this is going to do is going to come back to these ideas of just what is the definition of the derivative and how can we use that directly in here to build something that's a a different architecture for solving for these solutions. Again, we're going to go after an igen value and value igen function problem with this direct method. but let's illustrate it on something like just a standard boundary value problem. So here is a second order boundary value problem nonlinear generically with two boundary conditions on the left and the right at x= a

and x= b or here I used t. So t equals a t equals b. But this is the boundary value problem we want to solve at the two boundary value problems with the differential equation between. And again before what we did is we launched from this boundary condition through that condition and try to land on this boundary condition. And so we modified the launch conditions in order to do that.

Now instead we're going to say well wait a minute let's do something simpler maybe which is I'm going to go ahead for first of all simplify this down a little bit just so we can see how this work out. Let's let's go a and take a second order differential equation, but now it's linear non-constant coefficient with much simpler boundary conditions. At a it's alpha and at b it's beta. So we're going to do this and we're going to ask the questions, how would I go about solving this in a computational way? Well, what we're going to do is rely on this idea of we know how to construct estimates for the derivatives and second derivatives by using finite differences. So if you go back to the lectures on numerical integration, numerical differentiation, we know exactly how to approximate a derivative.

So we used tailaylor series expansions for instance. So f of t plus delta t is you tailaylor expand this here. Here it is all the way up to this is exact by the way for some constant c1 that lives between t and t plus delta t. So what we did is an expansion up to delta t cubed and we can also do an expansion of t minus delta t again up to delta t cubed with some constant c2 and c2. So this is an exact expression on both of these where c1 and c2 are these constants that we we don't know but we're actually going to approximate the derivative by throwing these away anyway. That's going to be the level of approximation. And the way we got a derivative formula is just simply adding these together or subtracting them, excuse me. So if you subtract these, you take the first equation here, subtract the second equation, you get this f of t plus delta t, f of t minus delta t. Here's the expression you get. You have this delta t cubed. Notice the second derivatives cancel out these guys. You get a delta t cub term here, which we're going to actually ignore. So we're hoping if delta t is small, deltat t cubed is very small. In fact, we can solve here for the derivative term to get this slope formula. And that's all this is, right?

This is just a slope. If you want to compute the derivative, it's rise over run. And here, what we're using is our neighbors. The point in front, point behind divided by 2 delta t. This is a slope formula that we have here using our two neighbors. And here is the error in the approximation. It goes like delta t^ squared. So for instance, if delta t is like 10 minus 2, then the error we're going to have is delta t^ squ is 10 minus 4. So that's going to be the error in approximating the derivative with this rise over run formula. And if you remember, we learned formulas for the second derivative, first derivative, and so forth. And so we can use all that now in this problem that I showed you, which was a second derivative, P of T, first derivative, G of T. And so what I did here for the second derivative, I replaced it with the second derivative formula, which is the point in front, point behind minus 2 of the current point over delta t ^2. Same thing here.

Here's the first derivative. Point in front, point behind over 2 delta t. That's just rise over run formula. And so what I've done here is discretize that boundary value problem by using the definition of derivative and our slope formulas. Okay? So there's a certain approximation accuracy. I can even make it more precise by using higher order differentiation formulas, but right now we're using the simplest ones possible. And so this is these are accurate to delta t squared both first derivative and the second derivative.

Once I have this, I can start collecting like terms. So I can collect all the terms that have a y of t plus delta t. So there's a term here. There's also a term there. And so what you end up with the terms that have y of t plus delta t. It's a one minus delta t over 2 p of t. There it is. I can also collect the terms with all the y of t terms. All the y of t minus delta t terms. And then on the right hand side is everything else.

Delta t ^2 r of t which comes from this term over here. And what I did here is I multiplied everything by delta t ^2. So I end up with this expression which is kind of interesting because what this is is it's basically breaking up all the unknowns y of t into points in front the y of t plus delta t points behind y of t minus delta t y of t. And so my claim is all this is going to produce for us is a giant ax= to b solve. That's what we're actually have in this system.

And we also have the boundary conditions to take care of which is the first point and the last point are actually known. We've discretized the domain. The first point y t0 is alpha and the last point y of t event is beta. So really what I'm doing is solving for all the interior points. The first and the last point are known because those are the specified boundary conditions. And so what I want to do is solve for everything else in between. And here's the relationships that I have. And so this is just ax equal to b. And let me show you what the ax equal to b looks like. And by the way, writing this in the ax equal to b is a really important step. You really have to understand how I take this here to this ax= to b because once you have it in ax equal to b, all you have to do is solve the x equal to b. Okay? And for the igen value problem, instead of ax equal to b, you get ax= lambda x. And you can just do iix on that matrix a to get everything. So here's what the matrix A looks like on the diagonal. So by the way, let's go back a term.

yeah, so here's the term I had. Here's the diagonal term. The y of t depends on itself. But here's here it is. This is the point in front. So this is an off diagonal on the on the top side. Here's the off diagonal on the bottom side. So those are what are populating the diagonal itself and then the upper diagonal and the lower diagonal come from these this term here and this term here. So that's how I built this matrix. The right hand side is just the things that's left over the delta t ^2 r of t.

The unknowns a when you do ax= b I'm actually trying to solve for y. So here they are. So once I have this this I I can just do solve for an x equal to b. The only thing that I would point out is the boundary conditions are embedded here on the first and last point. So remember I'm solving for the interior but the interior points when I get to the very edge depends on its two neighbors. When you get to the t1 point it depends on t0 but I know what t0 is.

The value there is alpha. And so it shows up here on the right hand side because I know what it is. I don't have to solve for it. And when I get to the far boundary, the nth point, okay, the or the end point, I know. So the n minus one point talks to the nth point and that shows up right here. And it's the value beta. Once I have that discretized, that's it. But it is really important you understand how to get there from this this equation here with these boundary conditions. It's equivalent. If I write them all together, this is what it looks like.

It's it's just ax equal to b. And then again, if you have an igen value problem where you're solving like this and

then all you're doing then is getting a matrix A that you would just have to find the igen values for. And that's very easy to do computationally by using the i command. Okay, that was a linear system gives you x= b. What happens if you have nonlinear system? Well, in a nonlinear system, here it is. I have a nonlinear differential equation. And this is where these relaxation methods come because once you do this, the same thing happens here. You simply discretize the derivative, first derivative, everything like that. You get a system of equations. But now you get a system of nonlinear equations and you have to solve the system of nonlinear equations.

And so when you go after the solution of the so the system of nonlinear equations, now you have to, you know, get a good guess and iterate towards the solution. And often times there's different relaxation schemes that give you nice iteration schemes for solving the system of non-indifferential equations. This is much harder and it very much relies on you having a good guess to the equations. This is some of this is built in directly into Python and I'll show you that. And so one of the keys to making it work is having a really good guess. So if you have a good guess that's near the solution, this is likely to work. If you do not have a good guess, this is unlikely to work. So those are just some things to know.

So that's how you might go about doing this. Notice it's a very different strategy than the shooting method. You just simply discretize across. You get a large system of equations to solve either linear or nonlinear depending upon the equation or the the what you're trying to solve. but it's a it's a completely different strategy than what we did when shooting where we started from one side and we just use standard steppers to get to the other side. Here you do the whole thing at once. Sort of more of a global solution technique as it were. And the cost of this one and the the cost of the other one is you have to keep solving it over and over and again to get to the other side. The cost of this one is you solve a large system of equations all up front at one time. So that is how you would solve this with you know these direct solve and relaxation techniques.