Comparing linear PDE solvers

Computer simulations in physics

Kürti Zoltán

November 19, 2021

Contents

1 Introduction

Based on feedback about the last project, in this project I will be more focused on explaining my thought processes and motivations. I will also use more informal language as it is better suited to describe the journey I took while working with partial differential equations.

1.1 Motivation

The importance of numerically solving partial differential equations can hardly be overstated both in purely academic and engineering fields. At the same time I also like working on both partial and ordinary differential equations, this was one of the reasons my first project involved differential equations too. I dream about obtaining solutions for complicated situations in general relativity, but that's simply not feasible during a couple weeks, in fact I'm not even sure a single person could get meaningful results in this field. So I have to set realistic goals and the obvious first step is to start with a simple linear problem, like the heat equation.

I also like to implement everything on my own, and I also like to come up with my own ideas even if I'm no the first to do so. This method takes a long time and generally isn't optimized for obtaining a result fast and with the less amount of energy. During evaluating a lab measurement to write the report or during a BSc. thesis the focus is on getting results in physics, not programming, and time is critical. In these situations usually people can't afford to begin the project by writing for loops, multiplications and additions in C, I certainly couldn't. This course however does focus on computer programming and writing low level algorithms. All in all this is the perfect opportunity for me to spend time on solving the heat equation on my own.

2 Solution outline

The first problem I thought about was calculating the laplacian on a discretized grid. This seemed to be an important and frequent operation while solving linear partial differential equations. While working on this problem I created the `coefficients.py` file. This file contains functions that help finding coefficients to approximate the laplacian at a point based on neighboring points up to different orders and with different conditions

on the error terms. The functions generally work in N dimensions, although the time for computations often grows exponentially with the number of dimensions. Some of the methods work for irregular grids, which would be very useful in case the discretization of a problem was matched to the geometry of the problem. I did not investigate this aspect, but it is a high priority extension I would make if I continue this project. With these functions it is possible to reproduce all the results about stencils in [1], but my approach applies to irregular and also higher dimensional cases.

The next step was to test some of these stencils I got from `coefficients.py`. I focused on one and two dimensional cases and compared single and double precision calculations too. Solving partial differential equations isn't just about choosing the right algorithms. A huge part of the process is finding optimal hyperparameters for the algorithm, like the resolution of discretization, the time step, relaxation parameters, precision of the floating point operations and so on. The choice of these parameters makes or breaks the success of the calculation. My impression so far is that there is no general rule for finding an optimal value for a parameter directly. Experimentation is needed, and this experimentation process can be sped up dramatically by good intuition. This part of the project certainly helped me to make more informed guesses during these types of experimentation.

The last component to solve the first concrete problem is treating boundary conditions. My goal was to be able to handle arbitrary geometries both with Dirichlet and Neumann type boundary conditions. I did meat this goal in two dimensions and my solution generalizes to higher dimensions easily (adding one or two more nested for cycles depending on the function in question). The basic idea applies to irregular discretization, although in that case additional data structures would be needed to look up nearby points. To demonstrate this capability I solved the heat equation in a circle using a square grid for discretization with different combinations of Dirichlet and Neumann boundary conditions. []

3 Float vs double

4 Higher order in space?

4.1 Designing the kernels

4.2 Testing the kernels

5 Boundary conditions, where numeric methods shine

6 Conclusion

References

[1] Michael Patra and Mikko Karttunen. *Stencils with isotropic discretization error for differential operators*. Numerical Methods for Partial Differential Equations, 22(4):936–953, 2006. https://onlinelibrary.wiley.com/doi/pdf/10.1002/num.20129