Lab / Assignment 3: Differential Equations

Alex Matheson, Austin Nhung

April 4, 2018

1 Introduction

2 Methods

2.1 Warm-Up

Finite differences allows one to set up an equation relating the value of a variable at a point to the values in the surrounding discrete locations. For instance, a two dimensional Laplace equation of the form $u_{xx} + u_{yy} = 0$ would have the form $u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} = 0$. By re-writing this equation in terms of $u_{i,j}$, we can have a series of equations that define the value of each point relative to that of its neighbors. If boundary values are supplied, this yields $n \times m$ equations for a region divided into $n + 2$ x-points and $m + 2$ y-points. The **direct method** is used to solve this system of equations exactly, by solving the series of equations using matrix methods such as Gaussian methods or L-U decomposition. Direct methods however may not always be desirable. This is because the matrix solving algorithms used may be slow and inefficient. Additionally, some data sets may not behave well with certain matrix methods computationally. As a result of this, iterative methods have been developed to address these shortcomings.

Iterative methods involve "guessing" a solution to the equation, and then continually refining the results until they converge to a solution (note that convergence tests may be required to ensure that the system converges to the correct solution). Given a series of boundary values, the inner values may then all be guessed. Next, finite differences may be used to determine an equation that defines the value of a cell based on its neighbors. Whereas the direct method tried to simultaneously determine all cells, the iterative starts with a guess, then uses the finite difference equation to create a new "generation" of grid based on the previous grid's values. In the earlier example, this equation would be $u^{k+1}_{i,j} = (u^k_{i+1,j} + u^k_{i-1,j} + u^k_{i,j+1} + u^k_{i,j-1})/4$ where k signifies the current iteration of the system. This method continues until certain criteria are met, though the final result will have some associated error. The method benefits, however, by generally performing faster than direct methods, especially when the guess is close to the actual answer.

There are different ways to implement iterative methods. The Jacobi method is closest to what has been described so far. In the Jacobi method, the grid is solved at some iteration k. Once the iteration is complete, the system is compared to exit conditions, and then the next iteration is generated. The Gauss-Seidel method considers a different approach. We know that the more iterations take place, the more likely a value at a particular point is to be correct. For the first point in a grid we consider, we use all the neighbor values from the previous iteration. In the Jacobi method, the next point also uses all the neighbor points from the previous iteration. The Gauss-Seidel method realizes that for the second point, we already have one "updated" point that should be closer than the previous iteration, so we use the updated neighbor when available instead of the previous iteration to approach the solution more quickly. In practice, the Gauss-Seigel method uses a checkerboard scheme, where points alternate between usng only previous iteration values and using a mixture of previous - and current iteration values.

This scheme maximizes the number of updated points used in the calculation, speeding up the algorithm. The Jacobi method is more intuitive to implement, since it completes one iteration at a time. The Jacobi method can also be split apart into different intersecting independent grids for parallelization, whereas the Gauss-Seigel method cannot be parallelized easily.

The aforementioned methods can be used to solve numerous physical systems. Many differential equaitons have general forms that have been given names over time.

Differential equations may be classified according to the descriminant of the generalized form $au_{xx} + bu_{xy} + cu_{yy} + du_x + eu_y + fu + g = 0$. A zero discriminant is classified as parabolic, whereas positive and negative are hyperbolic and elliptic respectively. Based on this, three sample PDEs may be considered. The equation $u_{tt} = 2u_{xx}$ is hyperbolic with a discriminant of 8, $u_{xx} + u_{yy} = 0$ is elliptic with a discriminat of -4, and $u_t - u_{xx}$ is parabolic with a discriminant of 0.

There are three different first order FD operators that may be used on a function. For the function $u(x) = x^2$, the forward FD is as follows:

$$\begin{aligned}\partial_x^+ u =& \frac{u(x+h) - u(x)}{h}\\ \partial_x^+ u(3) =& \frac{u(3+0.1) - u(3)}{0.1}\\ \partial_x^+ u(3) =& \frac{9.61 - 9}{0.1}\\ \partial_x^+ u(3) =& 6.10\end{aligned} \tag{1}$$

With a finer grid resolution, the result becomes:

$$\begin{aligned}\partial_x^+ u =& \frac{u(x+h) - u(x)}{h}\\ \partial_x^+ u(3) =& \frac{u(3+0.05) - u(3)}{0.05}\\ \partial_x^+ u(3) =& \frac{9.3025 - 9}{0.05}\\ \partial_x^+ u(3) =& 6.0500\end{aligned} \tag{2}$$

In both cases, the result is near the expected result of 6, but is off by a single factor of h. This matches the error of order $O(h)$ for the forward method.

Another FD method is the central method. The same two examples may be computed using this method.

$$\begin{aligned}\partial_x u =& \frac{u(x+h) - u(x-h)}{2h}\\ \partial_x u(3) =& \frac{u(3+0.1) - u(3-0.1)}{2*0.1}\\ \partial_x u(3) =& \frac{9.61 - 8.41}{0.2}\\ \partial_x u(3) =& 6.00\end{aligned} \tag{3}$$

$$\begin{aligned}\partial_x u =& \frac{u(x+h) - u(x-h)}{2h}\\ \partial_x u(3) =& \frac{u(3+0.05) - u(3-0.05)}{2*0.05}\\ \partial_x u(3) =& \frac{9.3025 - 8.7025}{0.1}\\ \partial_x u(3) =& 6.000\end{aligned}$$

As expected according to the text, the central method was more accurate. The result matched the analytical solution exactly.

References

[1] Ouyed and Dobler, PHYS 581 course notes, Department of Physics and Astrophysics, University of Calgary (2016).

[2] W. Press et al., *Numerical Recipes* (Cambridge University Press, 2010) 2nd. Ed.

[3] T. O'Brian, "Properties of Pulsars", University of Manchester Jordell Bank Observatory, accessed at `http://www.jb.man.ac.uk/distance/frontiers/pulsars/section1.html`.

[4] D. Hathaway, "The Sunspot Cycle", NASA, accessed at `https://solarscience.msfc.nasa.gov/SunspotCycle.shtml`

There are many different finite difference schemes that may be employed.

Name	Form	Order
Poisson's equation	$\nabla^2 u = f(x, y, z)$	2
Laplace's equation	$\nabla^2 u = 0$	2
3-Dimensional Wave equation	$u_{tt} = c^2 \nabla^2 u$	2
1-Dimensional Heat equation	$u_t - \alpha u_{xx} = 0$	2
1-Dimensional Wave equation	$u_{tt} = c^2 u_{xx}$	2
Ginzburg-Landau equation	$A_t = A_{xx} + \sigma A - \|A\|^2 A$	2
Korteweg-de Vries equation	$u_t + u_{xxx} - 6uu_x = 0$	3

Table 1: Different types of differential equations, their general forms, and order.

Name	FD Scheme	Order	CFL
Forward Euler	$u_k^{l+1} = u_k^l + r(u_{k+1}^l - u_k^l)$	1	$r \leq 1$
Upwind	$u_k^{l+1} = u_k^l + r(u_k^l - u_{k-1}^l)$	1	$r \leq 1$
Leap Frog	$u_k^{l+1} = u_k^{l-1} + r(u_{k+1}^l - u_{-1}^l)$	2	$r \leq 1$
Lax-Wendoff	$u_k^{l+1} = u_k^l - \frac{r}{2}(u_{k+1}^l - u_{k-1}^l) + \frac{r^2}{2}(u_{k+1}^l - 2u_k^l + u_{k-1}^l)$	2	$r \leq 1$
Backward Euler	$u_k^{l+1} = (1+r)^{-1}(u_k^l + ru_{k+1}l + 1)$	1	None
Crank-Nicholson	$u_k^{l+1} = u_k^l + \frac{r}{2\Delta x}(u_{k+1}^{l+1} - 2u_k^{l+1} + u_{k-1}^{l+1} + u_{k+1}^l - 2u_k^l + u_{k-1}^l)$	2	None

3 Appendix

For access to the source codes used in this project, please visit `https://github.com/Tsintsuntsini/PHYS_581` for a list of files and times of most recent update.