LECTURE 19

Numerical Methods: Finite Differences

Thus far, we really only considered three particular (yet fundamental) partial differential equations (the heat equation, the wave equation and Laplace's equation) and developed general formula for their solutions satisfying particular boundary conditions. It should not be surprising that there are many more PDEs out there for which the methods we have developed so far are insufficient to yield good formulas, or even that the formulas that we do have might be very difficult to apply in some situtations (e.g. on an irregularly shaped domain).

In the next couple of lectures we'll explore numerical methods of solving partial differential equations. These methods will have a certain advantage of "universal applicability", yet they will also certain inherent disadvantages as well.

As a means of familiarizing you with the numerical setting, I'll begin by reviewing of the Euler method for first order ordinary differential equations. Thus, suppose you have a first order ODE of the form

$$y' = f(x, y) \tag{1a}$$

$$y(x_0) = y_0 \tag{1b}$$

The Euler method begins with the idea that you can approximate the derivative y'

$$y'(x) \approx \frac{y(x+\Delta x) - y(x)}{\Delta x} \tag{2}$$

(the approximation becoming exact in the limit $\Delta x \to 0$) and that that derivative, in turn, gives you the best straight-line fit to the graph of the solution near x. A little more explicitly, if we set

$$\frac{y(x+\Delta x) - y(x)}{\Delta x} \approx y'(x) = f(x, y)$$

and solve for $y(x+\Delta x)$ we get

$$y(x+\Delta x) \approx y(x) + f(x, y)\,\Delta x \tag{3}$$

Equation (3) and the initial condition (1b) now tell us that we can find an approximate value y_1 for the solution at $x = x_1$, where

$$x_1 := x_0 + \Delta x$$

as

$$y(x_1) \approx y_1 = y(x_0) + f(x_0, y(x_0))\,\Delta x = y_0 + f(x_0, y_0)\,\Delta x$$

But then once we have an (approximate) value for the solution at x_1, we can use the same idea to compute an approximate value y_2 for the solution at $x_2 = x_1 + \Delta x$

$$y_2 = y_1 + f(x_1, y_1)\,\Delta x$$

and indeed by iterating the recursive formula

$$x_{i+1} = x_i + \Delta x$$
$$y_{i+1} = y_i + f(x_i, y_i)\,\Delta x$$

we can compute a whole table of approximate values of the solution $y(x)$.

We shall do something similar for partial differential equations; but before doing so let me point certain problems with this numerical methodology:

- This method does not produce an explicit formula for the solution, but rather a table of approximate values. As such the method is satisfactory only when all you need to understand are the values of the solution for a certain finite range of x.
- Formula (3) is only an approximate expression. When we apply it over and over not only to repeat the same (possibly bad) approximation, but the errors from proceding steps are carried through in each successive step.
- It might seem that by simply letting Δx be very, very small you can make the cummulative error small. In fact, this is true theoretically. However, there are also floating point errors that arise when this algorithm is implemented on a computer. The problem is that if the

  gap between $f(x+\Delta x) - f(x)$ becomes too small, then a computer is going to interprete the difference as 0, and now matter how small

  Δx is, $0/\Delta x = 0$. Other floating point errors occur when you try to add $y_i + f(x_i, y_i)\Delta x$, because the contribution of $f(x_i, y_i)\Delta x$ may be so small that it gets lost in the round-off error of y_i.

Despite these problems, numerical methods are quite indepensible; and so considerable effort has been directed at rendering numerical methods less susceptible to these pitfalls.

1. Finite Difference Expressions for Derivatives

Implicit in the Euler method described above is a partitioning of an interval $[x_0, x_N]$ into N subintervals of equal length

$$\Delta x = \frac{x_N - x_0}{N}$$

This is how one sets up the sequence

$$\begin{aligned} x_1 &= x_0 + \Delta x \\ x_2 &= x_1 + \Delta x = x_0 + 2\Delta x \\ x_3 &= x_2 + \Delta x = x_0 + 3\Delta x \\ &\vdots \\ x_N &= x_{N-1} + \Delta x = x_0 + N\Delta x \end{aligned}$$

The Euler method described above uses the approximation

$$\frac{y(x_{i+1}) - y(x_i)}{\Delta x} = y'(x_i) = f(x_i, y(x_i))$$

to compute sucessive values of $y_i \approx y(x_i)$. Using the approximation

$$\frac{dy}{dx}(x_i) \approx \frac{y_{i+1} - y_i}{\Delta x}$$

however, is not the only possibility. One could also use

$$\frac{dy}{dx}(x_i) \approx \frac{y_i - y_{i-1}}{\Delta x}$$

or even

$$\frac{dy}{dx}(x_i) \approx \frac{y_{i+1} - y_{i-1}}{2\Delta x}$$

That the first two expressions of are reasonable approximations for $y'(x_i)$ follows from the Taylor expansions of $y(x+\Delta x)$, $y(x-\Delta x)$

$$y(x_i + \Delta x) = y(x_i) + y'(x_i)\Delta(x) + \frac{1}{2}y''(x_i)(\Delta x)^2 + \mathcal{O}\left((\Delta x)^3\right) \quad \Longrightarrow \quad y'(x_i) = \frac{y(x_{i+1}) - y(x_i)}{\Delta x} + \mathcal{O}((\Delta x))$$

$$y(x_i - \Delta x) = y(x_i) - y'(x_i)\Delta(x) + \frac{1}{2}y''(x_i)(\Delta x)^2 + \mathcal{O}\left((\Delta x)^3\right) \quad \Longrightarrow \quad y'(x_i) = \frac{y(x_{i-1}) - y(x_i)}{\Delta x} + \mathcal{O}((\Delta x))$$

Taking the differences of these two Taylor expansions one obtains

$$y(x_i+\Delta x)-y(x-\Delta x)=2y'(x_i)\,\Delta x+\mathcal{O}\left((\Delta x)^3\right)\quad\Longrightarrow\quad\frac{y(x_{i+1})-y(x_{i-1})}{2\Delta x}+\mathcal{O}\left((\Delta x)^2\right)$$

is surprisingly a lot more accurate.

Higher order derivatives can also be approximated by certain finite-difference expressions. For the second derivative, the simplest approximation is

$$y''(x_i)=\frac{y_{i+1}-2y_i+y_{i-1}}{(\Delta x)^2}+\mathcal{O}\left((\Delta x)^2\right)$$

2. Finite Differences and the Heat Equation

I'll now describe how such finite-difference approximations for derivatives lead to a numerical method for solving a Heat Equation problem. We'll consider the following simple situation:

$$\begin{aligned}u_t-u_{xx}&=0\qquad 0\le x\le L\quad,\quad 0\le t\le T\\ u(x,0)&=f(x)\qquad 0\le x\le L\\ u(0,t)&=0\\ u(L,t)&=0\end{aligned}$$

The first step is set up a suitable *grid* of points (x_i,t_j) in the solution domain and the easiest way to do this is to fix integers N and M and set

$$\begin{aligned}\Delta x&=\frac{L-0}{N}\quad,\qquad x_i=i\Delta x\quad,\quad i=0,1,\ldots,N\\ \Delta t&=\frac{T-0}{M}\quad,\qquad t_j=j\Delta t\quad,\quad j=0,1,\ldots,M\end{aligned}$$

Next, let $u_{i,j}$ denote the approximate value of the solution at a point (x_i,t_j). We'll have

$$\begin{aligned}\left.\frac{\partial u}{\partial t}\right|_{(x_i,t_j)}&\approx\frac{u_{i,j+1}-u_{i,j}}{\Delta t}\\ \left.\frac{\partial^2 u}{\partial t^2}\right|_{(x_i,t_j)}&\approx\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{(\Delta x)^2}\end{aligned}$$

Replacing the partial derivatives in the Heat Equation with these approximations we obtain

$$\frac{u_{i,j+1}-u_{i,j}}{\Delta t}-\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{(\Delta x)^2}=0$$

or

$$u_{i,j+1}=u_{i,j}+\frac{\Delta t}{(\Delta x)^2}\left(u_{i+1,j}-2u_{i,j}+u_{i-1,j}\right)$$

Notice that on the right hand side, we utilize only the u with same (temporal) index j. So if we knew, for any fixed j all the values $u_{0,j},u_{1,j},\ldots,u_{N,j}$ we could compute all the values $u_{0,j+1},u_{1,j+1},\ldots,u_{N,j+1}$.

Note also the that initial condition

$$u(x,0)=0\quad\Longrightarrow\quad u_{i,0}=u(x_i)=f(x)$$

gives us an initial set of $u_{i,0}$'s. Meanwhile the boundary conditions at $x=0$ and $x=L$ fix all the $u_{0,j}$ and $u_{N,j}$ to be 0. And we can compute all the $u_{i,1}$'s between $i=1$ and $i=N-1$ via the formula

$$u_{i,1}=u_{i,0}+\frac{\Delta t}{(\Delta x)^2}\left(u_{i+1,0}-2u_{i,0}+u_{i-1,0}\right)\qquad,\qquad i=1,\ldots,N-1$$

Then once we have all the $u_{i,1}$'s we can procede to compute the $u_{i,2}$'s, then the $u_{i,3}$'s, etc.. Here is a simple bit of Maple code that would carry out an explicit computation (assuming the constants L, T, N and M and the function f have already been inititialized) :

```
dt := T/M:
dx := L/N:
for i from 0 to N do
  u[i,0] := f(0.0+i*dx)
od:
for j from 0 to M do
  u[0,j] := 0:
  u[N,j] := 0:
od:
for j from 1 to M-1 do
  for i from 1 to N-1 do
     u[i,j+1] := u[i,j] + (dt/dx^2)*(u[i+1,j] - 2*u[i,j] + u[i-1,j]):
  od:
od:
```

REMARK 19.1. Notice that the size of the contribution to $u_{i+1,j}$ from its *predecessors* $u_{i,j}, u_{i\pm1,j}$ is proportional to $\frac{\Delta t}{(\Delta x)^2}$. Accordingly, one should expect that the accuracy of the solution to suffer if this ratio gets to big. In fact, a closer analysis reveals that one needs

$$\frac{\Delta t}{(\Delta x)^2} \leq \frac{1}{2}$$

just for stability (meaning that small changes in initital conditions don't cause radical changes in the numerical solution).