CS667 **Lecture 13: Partial Differential Equations** 10 March 2005

Jose Guerra Lecturer: Steve Marschner

1 Introduction

A partial differential equation (PDE) is an equation that involves an unknown function and at least one of its partial derivatives. Any function that satisfies the relationships defined by this equation is a solution to the problem.

PDEs are found to describe physical phenomena such as fluid flow, gravitational fields and electromagnetic fields. They are practical in fields such as simulation, weather prediction and computer graphics.

2 Classification of PDE Problems

Traditionally, PDEs are classified by their mathematical characteristics into three main categories: *hyperbolic*, *parabolic*, and *elliptic*. This classification is based on the equation's curve of information propagation. Prototypical examples of each of these equation types are respectively: the *wave equation*, the *heat (or diffusion) equation* and the *Poisson equation*.

Another way to classify PDEs is as *initial value problems (IVPs)* or *boundary value problems (BVPs)*. From a computational point of view, this classification has more meaning since it gives insight into finding a solution to the equation.

Initial value problems deal mainly with the evolution over time of dynamic phenomena. Hyperbolic and parabolic equations fall under this category. The number of variables involved in this type of equations is $n + 1$ (spatial information variables plus a time variable). If information at some initial time t_0 is given, a numerical code should be able to track the time evolution with some accuracy. A principal computational concern of initial value problem algorithms is their *stability*.

Boundary value problems define equilibrium of static phenomena. Elliptic equations fall in this category and the number of variables involved in this equations is n (only spatial information). Given information on the boundary of the region of interest of the problem, a solution can be computed using a numerical code. Since stability is relatively easy to achieve in these problems, *efficiency* is the principal concern in boundary value problems.

3 Example: BVP Poisson Equation

The Poisson equation is of the elliptic type and is defined as:

$$u_{xx} = p(x) \tag{1}$$

Setting $p(x)$ to zero, gives us Laplace's equation:

$$u_{xx} = 0 \tag{2}$$

A representation of these equations on multiple dimensions is given by:

$$\nabla^2 u = p(x) \tag{3}$$

And since the definition of the Laplace operator is given by $\nabla^2 = (\nabla \cdot \nabla)$, we have:

$$u_{xx} + u_{yy} = p(x) \tag{4}$$

We can use the *finite difference method (FDM)* to solve this problem computationally. This method is used to approximate differential operators such as $u'(x)$ by a difference operator such as $(u(x+h) - u(x))/h$ for some small but finite h. Doing this for a large number of sample points in the domain gives a system of equations that can be solved algebraically.

Representing the function $u(x, y)$ by its values at a discrete set of points $N \times N$ and applying the computational mask of the Laplace operator yields a matrix equation that can be solved by different numerical methods.

We first obtain the computational mask of the Laplace operator (∇^2) by summing up the second derivative computational masks (as shown in figure 1) for each dimension variable in our domain.

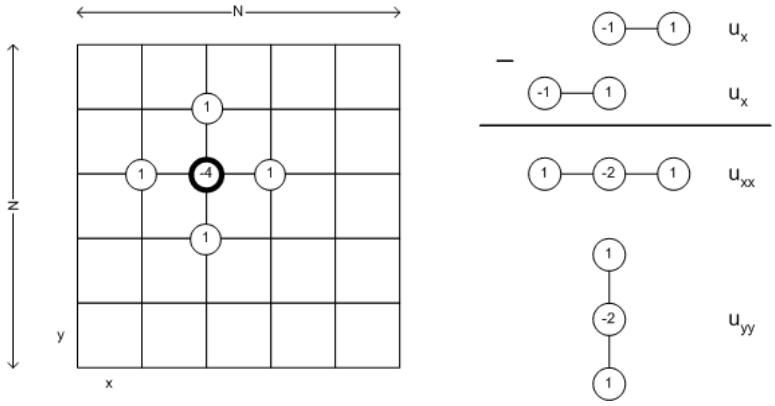

Figure 1: Computational masks for the Laplace operator (∇^2)

Applying the resulting mask at each point at an $N \times N$ grid will then create a matrix which has N^2 rows by N^2 columns in the case of an $N \times N$ two dimensional domain (shown in figure 2). Since this matrix depends on the number of samples N and the dimensionality of the domain, it is usually sparse - that is, it's large and contains many zero elements. For this reason, care should be taken when storing such a matrix in memory.

We now have a matrix equation of the form:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b} \tag{5}$$

We can go ahead and solve this equation using the usual methods for linear systems; in this context we would call those *"direct matrix methods"*. Or, the equation can also be solved by specialized numerical methods like *relaxation methods* or *rapid methods*.

4 Example: IVP Hyperbolic

Examples of initial value problems include the *advection equation* and the *wave equations*.

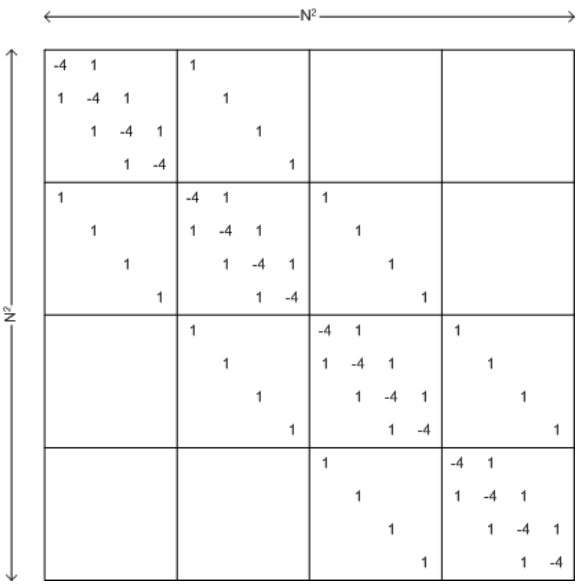

Figure 2: Matrix structure with $N = 4$, elements not shown are zero

The advection equation has the following form:

$$u_t = -Vu_x \tag{6}$$

and it defines translation on the domain over time with an example being the transport of pollution along a water pipe.

The wave equation is an important PDE that describes all kinds of waves such as sound, light and water waves. It has the following form in one dimension (plus time):

$$u_{tt} = V^2 u_{xx} \tag{7}$$

Its form in 2 and 3 dimension is as follows:

$$u_{tt} = V^2(u_{xx} + u_{yy} + u_{zz}) = V^2\nabla^2 u \tag{8}$$

5 Advection Example

We have the following advection equation:

$$u_t = -u_x \tag{9}$$

We use the computational masks from figure 3 to come up with the following equation:

$$\frac{u_{i,n+1} - u_{i,n}}{\Delta t} = -\frac{u_{i+1,n} - u_{i-1,n}}{2\Delta x} \tag{10}$$

Solving for $u_{i,n+1}$ we get:

$$u_{i,n+1} = u_{i,n} - \frac{\Delta t}{2\Delta x}(u_{i+1,n} - u_{i-1,n}) \tag{11}$$

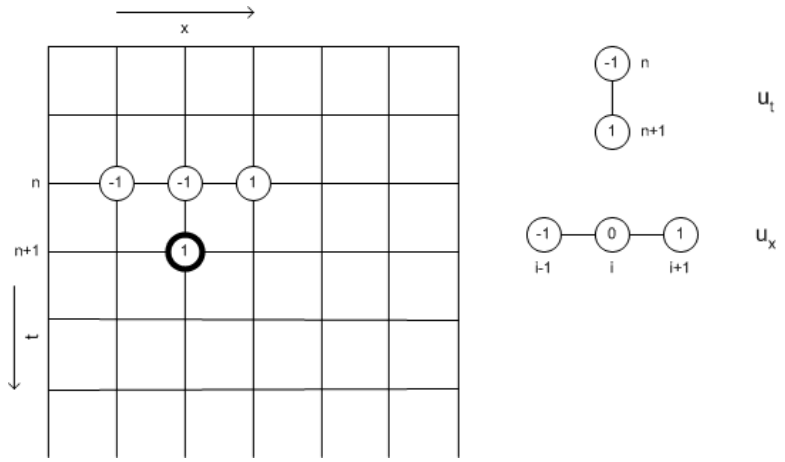

Figure 3: Preliminary computational masks for the advection equation which lead to unstable solutions

A similar analysis procedure to the one used for ordinary differential equations (ODEs) shows that this scheme will always produce solutions with error that grows exponentially, making it unconditionally unstable. A simple fix for this is to replace the mask with the *Lax scheme* (see figure 4).

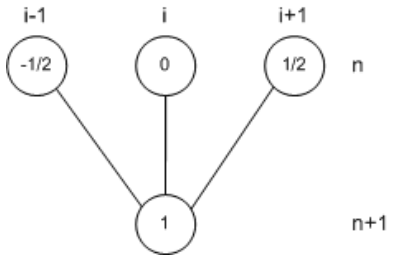

Figure 4: Computational mask for the time derivative in Lax scheme

The Lax scheme yields the following equation:

$$u_{i,n+1} = \frac{1}{2}(u_{i+1,n} + u_{i-1,n}) - \frac{\Delta t}{2\Delta x}(u_{i+1,n} - u_{i-1,n}) \tag{12}$$

The Lax scheme is conditionally stable. This stability criterion is:

$$\frac{|V|\Delta t}{\Delta x} \leq 1 \tag{13}$$

or

$$\Delta t \leq \frac{\Delta x}{|V|} \tag{14}$$

This condition means that the time step has to be smaller than the time it takes information to propagate across one step in space. This condition is called *CFL* after its authors *Courant, Friedrichs and Levy*. Its main principle is that information must come from an area that covers the domain of dependence. As seen

in figure 5.a we have an equation with propagation speed V and our neighbor samples completely cover the region that could influence the answer, making it stable. In contrast, in figure 5.b there are values that can influence the answer that lie outside the area covered by the samples and therefore are not taken into account.

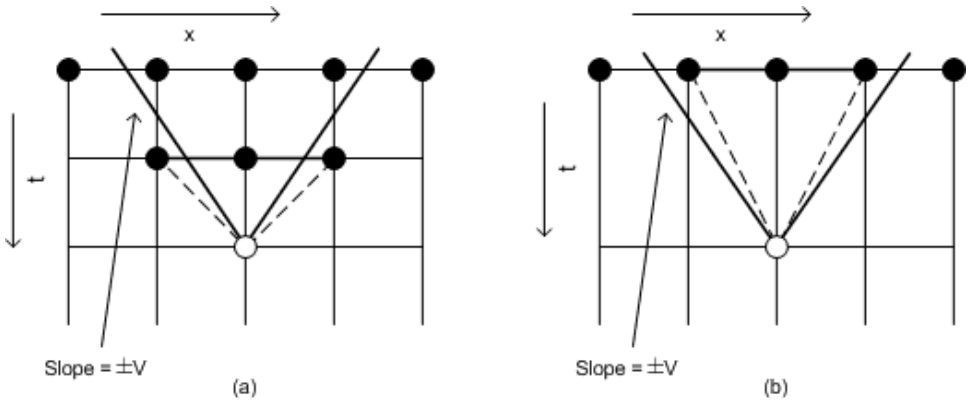

Figure 5: CFL condition

6 Example: IVP Parabolic

The canonical example for parabolic initial value problems is the diffusion or heat equation. This equation describes the variation of temperature for a given region over time. Solutions of the heat equation makes disturbances die away by gradually smoothing the initial temperature distribution by the flow of heat from warmer to colder areas of an object.

The equation is the following:

$$u_t = -Du_{xx} \tag{15}$$

and its multidimensional version is:

$$u_t = -D\nabla^2 u \tag{16}$$

The simple scheme that failed on the advection example works fine in the heat equation (See figure 6). It is conditionally stable for discretization, with the following condition:

$$\frac{2D\Delta t}{\Delta x^2} \leq 1 \tag{17}$$

or

$$\Delta t \leq \frac{\Delta x^2}{2D} \tag{18}$$

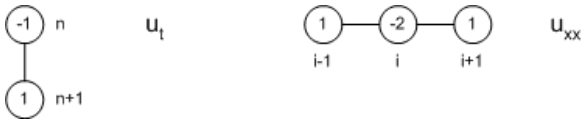

Figure 6: Computational masks for the finite difference approximation to the diffusion equation

This is another CFL condition: for the diffusion case, the time to propagate through a distance Δx is proportional to $(\Delta x)^2$. The parabolic nature of this inequality translates into slow propagation for long distances but fast propagation over short ones. Figure 7 shows a diagram of this behavior. Other schemes (implicit ones) are stable for larger time steps but tend to damp at the smaller scale.

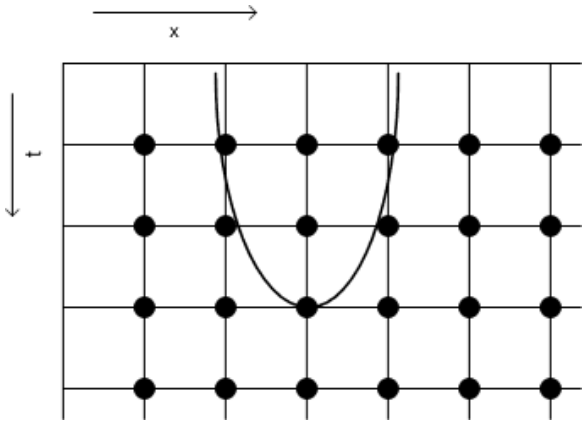

Figure 7: Parabolic nature of the diffusion equation

7 Properties of Finite Differencing Schemes for Initial Value Problems

Some properties to take into account when solving initial value problems with finite differencing schemes are: *the stability criterion*, *the order of accuracy* and *the types of errors*.

As mentioned before, the principal computational concern of initial value problems is stability. And as we saw on the examples the stability criterion is usually specified as an inequality that has to be maintained to keep our solution stable, an example of such criteria is the CFL condition.

While stability measures global error and prevents our solution from growing exponentially, the order of accuracy considers a more localized error. Minimizing the error term of a Taylor approximation of the scheme is commonly used to improve the scheme's accuracy.

The order of accuracy of our solutions is usually of second order although it can be different in the time and space domains. For example, we could have a first order in time solution that is of second order in space.

Two types of errors are most typical in these finite differencing schemes: *dissipation* and *dispersion*. Dissipation, which is also known as *numerical viscosity*, is the tendency to damp out small scale waves. Dispersion is the tendency of waves at different scales to propagate at different speeds.

8 Example: Cloth Modeling

We can write a PDE that relates the deformation of our cloth model to its acceleration (Figure 8). There are a set of ways in which we can simulate this deformation behavior. The first way we could try is by simulating static draping and set the problem as a boundary value problem (elliptic type). Another possibility is to use a mass-spring particle system and solve it as an ODE. We could also set the simulation as an initial value problem and either use finite difference discretization or finite element discretization to solve it.

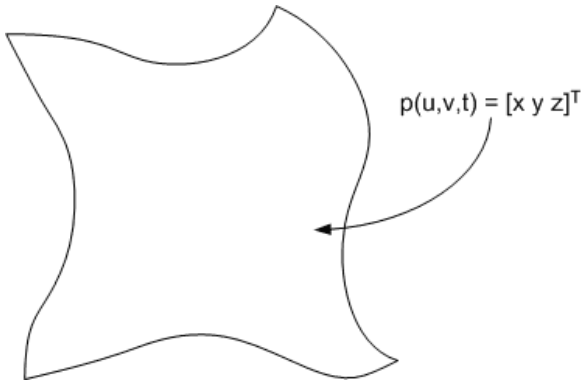

Figure 8: Cloth simulation