Scientific Computing: Lecture 22

- General classifications of PDEs
- Boundary and initial conditions
- Explicit solutions
  - FTCS, Lax, Lax-Wendroff
  - Stability
- Example: Wave equation

CLASS NOTES

- HW09 due next Friday (optional for undergrad students).
- Some materials posted on web.
- Proposal Comments back to you electronically.

General Classifications of PDEs

- Partial differential equations mathematically describe a system which depends on multiple variables and their derivatives.
- Examples:
  - Wave equation (acoustics, optics)
  - Laplace equation (electrostatics)
  - Schrodinger equation (quantum mechanics)
  - Navier-Stokes equation (fluid flow)
- Several general classes of PDEs often dictate different numeric approaches

Classes of PDEs

- Consider a generic 2nd order PDE with variables x and y

$$a\frac{\partial^2 A}{\partial x^2} + b\frac{\partial^2 A}{\partial x \partial y} + c\frac{\partial^2 A}{\partial y^2} + d\frac{\partial A}{\partial x} + e\frac{\partial A}{\partial y} + fA(x,y) + g = 0$$

where A is the solution and the rest are constants.

- **hyperbolic** if: $b^2 - 4ac > 0$
- **parabolic** if: $b^2 - 4ac = 0$
- **elliptic** if: $b^2 - 4ac < 0$

Some examples

- 1D Wave equation is hyperbolic:

$$\frac{\partial^2 A}{\partial t^2} = c^2 \frac{\partial^2 A}{\partial x^2}$$

- Diffusion equation is parabolic:

$$\frac{\partial}{\partial t} T(x,t) = \kappa \frac{\partial^2}{\partial x^2} T(x,t)$$

- Poisson's equation is elliptic:

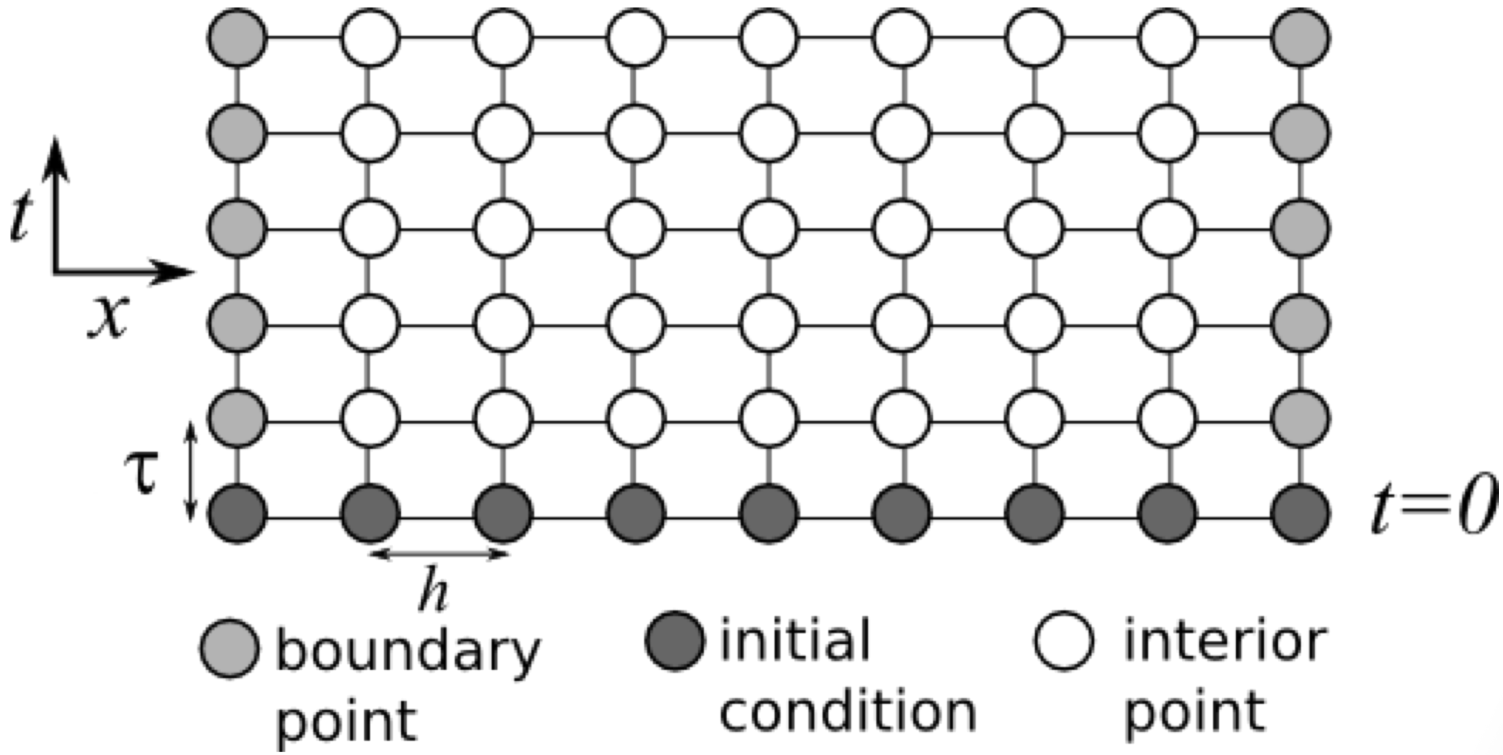
$$\frac{\partial^2 \Phi}{\partial y^2} + \frac{\partial^2 \Phi}{\partial x^2} = -\frac{1}{\epsilon_0} \rho(x,y)$$

Initial and Boundary Conditions

- Initial Conditions
  - Consider one independent variable is time and another is space in 1 dimension (say x).
  - We need an initial value (at t=0) for all positions along x.
- Boundary conditions
  - We also need values at both ends of the space domain which are known for all times.
- Driving terms
  - Known values of the solution at interior points which may change with time.

Typical PDE Grid (1 space and time)

- Space stencil: h and time stencil: τ



Types of Boundary Conditions

- Dirichlet Boundary Conditions
  - Also known as 'fixed'
  - Values of the solution at the end points are known for all times – like for a flexible string which is clamped at both ends.
- Neumann Boundary Conditions
  - Values for the derivative of the solution are known for all times – like heat energy flux at the end of a rod.
- Cauchy Boundary Condition
  - BOTH the above are known – the value of the solution AND the normal derivative – liked a clamped stiff bar.

Discretization of PDEs

- Typical idea is to:
  - 1. Convert all partial derivatives into finite difference equations via FDA, BDA, or CDA
    - Higher order derivatives require more terms
  - 2. Algebraically solve for the values of the solution at the next time (or space) step in terms of values at previous times (or spaces).
- Example: the Advection equation

$$\frac{\partial A}{\partial t} = -c\frac{\partial A}{\partial x}$$

Forward Time-Center Space (FTCS)

- Using forward difference method for time and center difference method for space derivative, this becomes:

$$\frac{A_i^{n+1} - A_i^n}{\tau} = -c\frac{A_{i+1}^n - A_{i-1}^n}{2h}$$

where n is time index and i is space index.

- Now solve for the solution of A at the n+1 time step:

$$A_i^{n+1} = A_i^n - \frac{c\tau}{2h}\left(A_{i+1}^n - A_{i-1}^n\right)$$

- Unfortunately FTCS is unstable for ALL values of the time step! Solution will eventually "blow up".

Lax Method

- We can improve stability by averaging for the value of A at space points before and after:

$$A_i^{n+1} = \frac{1}{2}\left(A_{i+1}^n + A_{i-1}^n\right) - \frac{c\tau}{2h}\left(A_{i+1}^n - A_{i-1}^n\right)$$

- Courant-Friedrichs-Lewy (CFL) stability condition. "c" has units of speed, so this amounts to saying that the numerics must be able to "move" faster than the system.

$$\tau_{max} = \frac{h}{c}$$

- Numeric "speed" is: $\frac{h}{\tau}$

Lax-Wendroff Method

- FTCS and Lax methods are based on dropping 2nd order terms.
- Stability and accuracy are improved by dropping 3rd order terms.
- This makes expressions for finite differences more complicated algebraically (not shown here).
- Schemes are identical IF: $\tau = \tau_{max}$
- For a large time step, Lax method grows (eventually blows up)
- For a smaller time step, Lax method decays to 0!
- Sometimes called numeric damping or viscosity.

Example: Wave Equation

- Recall wave equation for homogeneous media and no damping can be written as
$$\frac{\partial^2 A}{\partial t^2} = c^2 \frac{\partial^2 A}{\partial x^2}$$
- Which involve 2nd order derivatives. Prescription is the same – convert to 2nd order finite difference equations and solve for next time step.
- up: u at time plus 1, u: current time, um: time minus 1

```
while t <= tstop:
    t_old = t; t+=dt
    if method == 's':
        for i in range(1,n):
            up[i] = -um[i] +2*u[i] + C2*(u[i-1] - 2*u[i] + u[i+1]) + dt2*f(x[i],t_old)
```

Looping trick in Python

- Here we are looping over time (**while** loop), then looping over space (**for** loop).
- Since these are arrays, we can leverage the fast underlying C code which handles array slicing.
- Called "vectorizing" the code.
- The **for** loop is <u>replaced</u> by

```
up[1:n] = -um[1:n] +2*u[1:n] +C2*(u[0:n-1] - 2*u[1:n] +u[2:n+1]) + dt2*f(x[1:n],t_old)
```

- Recall `u[1:n] means u[1], u[2], u[3], …, u[n]`
- This provides a HUGE speed up by pushing the looping down to the compiled C code level.
- This is almost as fast as writing the program in C.

Output with Dirichlet BCs