PY410 / 505
Computational Physics 1

Salvatore Rappoccio
Partial Differential Equations

• Start looking at PDE’s
  – http://en.wikipedia.org/wiki/Partial_differential_equation
• Just like ODE’s, only harder! (Kidding)

• You should be familiar with the mathematics of PDE’s
  – Poisson equation
  – Diffusion equation
  – Wave equation

• The general strategy is to look at finite derivatives (just like we did in ODE’s), but now we have to look in multiple dimensions at once!
Partial Differential Equations

• First example: Elliptic PDEs

• Given an electric charge distribution \( \rho(r) \), Poisson’s equation is:

\[
\nabla^2 V(r) = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = -\frac{\rho(r)}{\varepsilon_0}
\]

• This determines the potential \( V(r) \) at each point \( r \), provided boundary values are specified
  
  – Dirichlet: \( V(r) \) specified on boundary
  
  – Neumann: normal component \( \hat{n} \cdot \nabla V \) specified on boundary

  • For electrostatics, this specifies normal component of \( E \)-field in a conductor

  – Periodic: \( V(r) = V(r + \text{dr}) \) for some \( \text{dr} \)
• Why “elliptic”? 
• Consider 2-d and let 

\[ V(x, y) \sim e^{ik_x x + ik_y y}. \]

• Then: 

\[ -\nabla^2 V(x, y) = (k_x^2 + k_y^2) V(x, y). \]

• The \( k_x, k_y \) values in k-space of a given eigenvalue satisfy 

\[ (k_x^2 + k_y^2) = \text{constant} \]

• This is (of course) a circle, which is an ellipse

• We’ll continue this “conic section” terminology, as you probably have done in your other courses
Partial Differential Equations

• Second case: parabolic PDEs

• Given a source $S(r,t)$ and a diffusion coefficient $D(r)$, the diffusion equation is:

\[
\frac{\partial n(r,t)}{\partial t} - \nabla \cdot \left( D(r) \nabla n(r,t) \right) = S(r,t)
\]

• This determines the concentration “n” in a closed space
  – Now need both initial conditions ($t=t_0$) AND boundary conditions (Dirichlet, Neumann, periodic)
Partial Differential Equations

• Why “parabolic”?

• Consider one spatial dimension, and a constant $D$, with

$$n(x, t) \sim e^{-\omega t + ikx},$$

• The differential operator on the LHS has the eigenvalue

$$-\omega + Dk^2 = \text{constant}$$

• which is a parabola in omega-k space
The time-dependent Schroedinger equation is also a parabolic PDE:

\[
 i\hbar \frac{\partial \Psi(\mathbf{r}, t)}{\partial t} = -\frac{\hbar^2}{2m} \nabla^2 \Psi(\mathbf{r}, t) + V(\mathbf{r}) \Psi(\mathbf{r}, t) = \mathcal{H} \Psi(\mathbf{r}, t)
\]

This can be viewed as a diffusion equation with imaginary diffusion constant \( D = i\hbar/(2m) \), or mathematically as a diffusion equation in imaginary time with real diffusion constant \( D = \hbar/(2m) \).
• Third case: hyperbolic PDE’s

• The wave equation is:

\[
\frac{1}{c^2} \frac{\partial^2 u(r, t)}{\partial t^2} - \nabla^2 u(r, t) = R(r, t)
\]

• this is hyperbolic because the eigenvalues of the differential operator are:

\[
-\frac{1}{c^2} \omega^2 + k^2 = \text{constant}
\]

• These are hyperboloid surfaces in omega-k space

• Again need initial conditions (t=t0) and boundary conditions (Dirichlet, Neumann, Periodic)
Elliptic PDES
Let's first take a look at the solution to the elliptic equation for Poisson's equation (solving Gauss's law for electrostatics)

We have Gauss's law:

\[ \nabla \cdot E = \frac{\rho(x, y, z)}{\varepsilon_0}, \]

The static electric field can be written as:

\[ E = -\nabla V, \]

And \( V(r) \) satisfies Poisson's equation:

\[ \nabla^2 V = \left( \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \right) V = -\frac{\rho(x, y, z)}{\varepsilon_0}. \]
Partial Differential Equations

• Now, we need to discretize the entire space

• Consider a 2-d space and discretize in 10x10 blocks:
• The 2-d Poisson’s equation is:

\[
\left( \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) V(x, y) = -\rho(x, y),
\]

• Let’s work in units with \( \epsilon_0 = 1 \), and solve in the region of a square with length \( A = 1.0 \)

• The grid is:

\[
x_i = ih, \quad i = 0, 1, \ldots L, L+1, \quad y_j = jh, \quad j = 0, 1, \ldots L, L+1.
\]

• The lattice spacing is \( h = 1/(L+1) \)

• Let

\[
V(x_i, y_j) = V_{ij}, \quad \rho(x_i, y_j) = \rho_{ij}
\]

• Now we need to discretize this
The discretization is to look at an equivalent of Euler’s formula, but now we have to do it in two dimensions:

\[
\left( \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) V(x_i, y_j) \approx \frac{1}{h^2} [V_{i+1,j} + V_{i-1,j} + V_{i,j+1} + V_{i,j-1} - 4V_{i,j}] \\
= -\rho_{i,j}.
\]

Note the following:

– The lattice is only connected to its four nearest neighbors
– We will define “odd” and “even” sites depending on whether \( i+j \) is odd or even (red/black)
– The boundaries are indicated with open circles
First attempt: Jacobi’s iterative method

Suppose we have a solution of the discretized equation

At each lattice site:

\[ V_{i,j} = \frac{1}{4} \left[ V_{i+1,j} + V_{i-1,j} + V_{i,j+1} + V_{i,j-1} + h^2 \rho_{i,j} \right] . \]

If we knew the RHS, then we could compute the LHS

But, the RHS pieces all have their own equations similar to this one!

They all need to be solved simultaneously

Instead of that, we try for a guess at each point, and then iteratively solve:

\[ V_{i,j}^{n+1} = \frac{1}{4} \left[ V_{i+1,j}^n + V_{i-1,j}^n + V_{i,j+1}^n + V_{i,j-1}^n + h^2 \rho_{i,j} \right] , \quad n = 0, 1, 2, \ldots \]
This should remind you a bit of the relaxation method for our ODE’s

– We guess, then iterate until our boundary is solved and the equations are satisfied at the points

But, all we know for sure are the boundary points

Can instead iterate until our solution stops changing very much

Usually “relaxes” to the right solution, but there are of course pathologies that can occur
Partial Differential Equations

• Next example: use the Gauss-Seidel method
• This is almost the same as the Jacobi method, but uses the updated neighbor sites
  – Remember the red/black? Red only talks to black, and vice versa
• Then we have:

\[ V_{i,j}^{n+1} = \frac{1}{4} \left[ V_{i+1,j}^n + V_{i-1,j}^n + V_{i,j+1}^n + V_{i,j-1}^n + h^2 \rho_{i,j} \right] \]

• This converges faster than the Jacobi method
Finally, consider the Successive Over-Relaxation (SOR) method

Jacobi and Gauss-Seidel do not use $V_{ij}$ at the same lattice point in updating $V_{ij}$

If we use a linear combination of the old and new solutions, we can get better convergence:

$$V_{i,j}^{n+1} = (1 - \omega)V_{i,j}^n + \frac{\omega}{4} \left[ V_{i+1,j}^n + V_{i-1,j}^n + V_{i,j+1}^n + V_{i,j-1}^n + h^2 \rho_{i,j} \right]$$

Omega is called the “over-relaxation” parameter
– Can be tuned for performance
• A few notes:
  – Converges only if $0 < \omega < 2$
  – Faster than Gauss-Seidel only if $1 < \omega < 2$
  – It converges fastest on a square lattice if

\[
\omega \approx \frac{2}{1 + \frac{\pi}{L}},
\]

• Here, $L$ is the number of lattice points
For our strategy, we will use the red/black splitting to solve the equations faster:

– First update the even sites, then update the odd sites
– Can use the SOR method (or the others) with faster convergence in this case

In Numerical Recipes 19.5, the iterations required to reduce the overall error by a factor of $10^{-p}$ for Laplace’s equation is:

\[ r \approx \begin{cases} \frac{1}{2}pL^2 & \text{for Jacobi’s method} \\ \frac{1}{4}pL^2 & \text{for the Gauss-Seidel method} \\ \frac{1}{3}pL & \text{for SOR with } \omega \approx 2/(1 + \pi/L) \end{cases} \]

\[
\begin{pmatrix}
\frac{1}{2} \times 3 \times 50^2 = 3,750 \\
\frac{1}{4} \times 3 \times 50^2 = 1,875 \\
\frac{1}{3} \times 3 \times 50 = 50
\end{pmatrix}
\]
To solve for the convergence rates, let’s look at the Poisson equation again:

\[
\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = -\frac{1}{\varepsilon_0} \rho ,
\]

In matrix form, this is:

\[Ax = b ,\]

Can break \(A\) into lower triangular, diagonal and upper triangular bits:

\[A = L + D + U ,\]
Then, at each step, the Jacobi iteration is

\[ D x^{(n)} = -(L + U)x^{(n-1)} + b, \]
\[ x^{(n)} = -D^{-1}(L + U)x^{(n-1)} + D^{-1}b. \]

The matrix:

\[ -D^{-1}(L + U) \]

This is the “iteration matrix”, and the magnitude of the largest eigenvalue is the “spectral radius” for the relaxation problem.
• Spectral radius \( \rho_s \) should satisfy:
  – \( 0 < \rho_s < 1 \) for the method to be stable
  – depends on the boundary conditions and the lattice spacing
  – approaches 1.0 as the number of lattice points increases

• For \( L \times L \) square lattice with Dirichlet boundary conditions:

\[
\rho_s \simeq 1 - \frac{\pi^2}{2L^2}.
\]
• How to derive spectral radius $\rho_s$ ?
• Let’s just do it in 1-d
• The 1-d Laplace equation is:

$$\frac{d^2 V}{dx^2} = 0 .$$

• This can be discretized as:

$$\frac{V_{i+1} + V_{i-1} - 2V_i}{h^2} = 0 .$$

• The Jacobi iteration is:

$$V_i^{n+1} = \frac{1}{2} \left( V_{i+1}^n + V_{i-1}^n \right) .$$

• With Dirichlet BC’s $V(0)=V(L+1)=0$, we see the eigenvectors are:

$$u_i^{(k)} = \sin \left( \frac{\pi k i}{L+1} \right) , \quad k = 1, 2, \ldots, L .$$
• Eigenvalues are determined by plugging in:

\[
\frac{1}{2} \left( u_{i+1}^{(k)} + u_{i-1}^{(k)} \right) = \frac{1}{2} \left[ \sin \left( \frac{\pi k (i + 1)}{L + 1} \right) + \sin \left( \frac{\pi k (i - 1)}{L + 1} \right) \right] \\
= \cos \left( \frac{\pi k}{L + 1} \right) u_i^{(k)} .
\]

• The spectral radius is given by the largest eigenvalue:

\[
\rho_s = \cos \left( \frac{\pi}{L + 1} \right) \approx 1 - \frac{\pi^2}{2L^2} ,
\]

(for large L)

• Similar analysis in 2-D gets the Numerical Recipes version for 2D:

\[
\rho_s = \frac{h_y^2 \cos \left( \frac{\pi}{L_x + 1} \right) + h_x^2 \cos \left( \frac{\pi}{L_y + 1} \right)}{h_x^2 + h_y^2}
\]
• How many iterations does it take for the solution to be damped by a factor of $10^{-p}$?

• Determined by the spectral radius!

$$10^{-p} = \rho_s^n \quad \Rightarrow \quad n = \frac{p \ln 10}{(-\ln \rho_s)} \approx \frac{2pL^2 \ln 10}{\pi^2} \approx \frac{1}{2} pL^2.$$  

• Jacobi method is not very efficient!

• If $L = 1000$, then $n = 1M$ to improve to 1% of current value
• Gauss-Seidel does a little better

• Iteration matrix is

\[-(L + D)^{-1}U,\]

• Then the spectral radius for the LxL Dirichlet lattice is:

\[\rho_s \approx 1 - \frac{\pi^2}{L^2} \quad \Rightarrow \quad n \approx \frac{1}{4} pL^2.\]

• Only about twice as fast as Jacobi!
What about SOR?

Much better here, we have:

\[ \rho_s \approx 1 - \frac{2\pi}{L} \quad \Rightarrow \quad n \approx \frac{1}{3} p L . \]

So, if L=1000, need only n=667 iterations to improve to 1% of current value.
• What about computational complexity?

• Jacobi and Gauss-Seidel update all interior lattice points per iteration
• So, for L×L 2-D lattice, we would have $O(L^4)$

• For SOR, we would have $O(L^3)$

• Neither of these are wonderful for very large L
Can also use spectral analysis to solve our PDE’s, just like you do in your math classes.

Here, “spectral analysis” is the FFT.

In 1D:

\[ \frac{d^2 V}{dx^2} = \rho(x) . \]

Then we express \( f \) and \( \rho \) in terms of their Fourier transforms:

\[ f(x) = \frac{1}{\sqrt{2\pi}} \int g(k)e^{ikx} \, dk, \quad \rho(x) = \frac{1}{\sqrt{2\pi}} \int \sigma(k)e^{ikx} \, dk. \]

This is diagonalized in k-space:

\[ -k^2 g(k) = \sigma(k) \quad \Rightarrow \quad g(k) = -\frac{\sigma(k)}{k^2}. \]

The solution is then the inverse FFT:

\[ f(x) = -\frac{1}{\sqrt{2\pi}} \int \frac{\sigma(k)}{k^2}e^{ikx} \, dk. \]

Two problems: 1. boundary conditions, 2. singularity at \( k=0 \).
Partial Differential Equations

• Boundary conditions dictate the type of Fourier transform you want to use
  –Sometimes sine transforms are best, sometimes cosine, sometimes exponential
• Consider 1-D lattice $0 < x < L$ with N points
  
  $$x_n = nL/N, \quad n = 0, \ldots, N - 1$$

• The complex FFT coefficients of $f(x)$ are
  
  $$g_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} W^{kn} f_n, \quad W = e^{2i\pi/N}.$$ 

• The inverse will be periodic in $x_n$ with period L:
  
  $$f_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} W^{-nk} g_k,$$

• So, if periodic conditions : use the complex FFT
For Dirichlet conditions \( f(0) = f(L) = 0 \), then sine Fourier transform is best:

\[
f_n = \sqrt{\frac{2}{N}} \sum_{k=1}^{N-1} \sin \left( \frac{\pi nk}{N} \right) g_k.
\]

For Neumann conditions use cosine Fourier transform:

\[
f_n = \frac{1}{\sqrt{2N}} [g_0 + (-1)^n g_N] + \sqrt{\frac{2}{N}} \sum_{k=1}^{N-1} \cos \left( \frac{\pi nk}{N} \right) g_k.
\]

Note: These are not just the real and imaginary parts of the complex exponential transform!

- Sine, Cosine, and \( \exp(ikx) \) are all complete sets with different boundary conditions
- Sine/Cosine are real, so also require 2x as many points
Let’s go back to Poisson’s equation in 2d:

\[
\left( \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) V(x, y) \\
\approx \frac{1}{h^2} [V_{j+1,k} + V_{j-1,k} + V_{j,k+1} + V_{j,k-1} - 4V_{j,k}] \\
= -\rho_{j,k}
\]

Let’s take an N\times N grid in region 0 < x,y < 1

Presume there is a point charge at the center

Impose periodic BCs so we use the exponential FFT

Since the FFT is linear, we can do it separately in the x and y directions, and it doesn’t matter which order!
Partial Differential Equations

- The 2-D FFT coefficients are

\[
\tilde{V}_{m,n} = \frac{1}{N} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} W^{mj+nk} V_{j,k} , \\
\tilde{\rho}_{m,n} = \frac{1}{N} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} W^{mj+nk} \rho_{j,k} .
\]

- The inverse transforms are:

\[
V_{j,k} = \frac{1}{N} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} W^{-jm-kn} \tilde{V}_{m,n} , \\
\rho_{j,k} = \frac{1}{N} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} W^{-jm-kn} \tilde{\rho}_{m,n} .
\]

- So, if we plug these into our discretized equation and equating coefficients of \( W^{-mj-nk} \) we get:

\[
\frac{1}{h^2} [W^m + W^{-m} + W^n + W^{-n} - 4] \tilde{V}_{m,n} = -\tilde{\rho}_{m,n} ,
\]

- IFFT gives the potential:

\[
\tilde{V}_{m,n} = \frac{h^2 \tilde{\rho}_{m,n}}{4 - W^m - W^{-m} - W^n - W^{-n}} .
\]
• In some sense, this is even easier than relaxation methods
• Take FFT of rows of rho
• Take FFT of columns of rho

\[ \tilde{V}_{m,n} = \frac{h^2 \tilde{\rho}_{m,n}}{4 - Wm - W-m - Wn - W-n} \]

• Solve equation in Fourier domain

```python
# Solve equation in Fourier space
V = cpt.Matrix(N, N)
W =cmath.exp(1.0j * 2 * math.pi / N)
Wm = Wn = 1.0 + 0.0j
for m in range(N):
    for n in range(N):
        denom = 4.0 - Wm - 1 / Wm - Wn - 1 / Wn
        if abs(denom) != 0.0:
            V[m][n] = rho[m][n] * h**2 / denom
        Wm *= W
        Wn *= W
```

• Take IFFT of rows of rho
• Take IFFT of columns of rho
Partial Differential Equations

• Since PDE’s are done in higher dimensions, it is oftentimes beneficial to use “multigrid methods”

• General gist: start at a coarse scale, get close to the answer, then go to a finer scale
  – Similar to adaptive RK4 in philosophy

• For this, need an estimate of the error at each stage

• Described in Chapter 19 Section 6 of Numerical Recipes
So let’s again consider Poisson’s equation in 2D:

\[
\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -f(x, y),
\]

Again let’s impose this on a grid with units 0-1 and impose Dirichlet boundary conditions.

As before, the solution obeys:

\[
u_{i,j} = \frac{1}{4} \left[u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} + h^2 f_{i,j}\right].
\]
Then here is where things get different

This uses a succession of $\ell$ lattices / grids

This is the “multigrid”!

Here’s the trick: define the interior lattice points as a power of 2 so that:

$$L = 2^\ell + 2$$

Thus the lattice spacing is

$$h = \frac{1}{(L - 1)}$$

There are then sequentially coarser lattices with number of interior points as:

$$2^{\ell-1} \rightarrow 2^{\ell-2} \rightarrow \ldots \rightarrow 2^0 = 1$$
Partial Differential Equations

• Now to compute the error, we define the solution at any stage in the calculation as \( u(x, y) \).

• Also define the exact solution \( u_{\text{exact}}(x, y) \).

• The correction is

\[
v = u_{\text{exact}} - u
\]

• The “residual” or “defect” is defined as

\[
r = \nabla^2 u + f.
\]

• The correction and the residual are related by:

\[
\nabla^2 v = [\nabla^2 u_{\text{exact}} + f] - [\nabla^2 u + f] = -r.
\]

• So interestingly, this has the same form as Poisson’s equation with \( v \) as the function \( u \), and \( r \) being a known source function!
Partial Differential Equations

• Now define the “Simple V-Cycle Algorithm”
• Define two grids (coarse and fine) with points:
  
  $$ L = 2^L + 2 \quad L = 2^{L-1} + 2 $$

• Need to move from one grid to another
• Given any function on the lattice, we need to:
  – restrict the function from fine to coarse
  – interpolate the function from coarse to fine
Partial Differential Equations

- If we have those, the multigrid V-cycle can be defined recursively:
  - \( \ell = 0 \), there is only one interior point, so solve exactly:
    \[
    u_{1,1} = \frac{u_{0,1} + u_{2,1} + u_{1,0} + u_{1,2} + h^2 f_{1,1}}{4}.
    \]
  - Otherwise, calculate current
    \[ L = 2^\ell + 2 \]
  - Perform pre-smoothing iterations with a local algorithm (Gauss-Seidel, etc). This will damp out the short wavelength errors in the solution
  - Estimate correction as:
    \[
    v = u_{\text{exact}} - u
    \]
    - Compute residual
      \[
      r_{i,j} = \frac{1}{h^2} \left[ u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} \right] + f_{i,j}.
      \]
      - Restrict residual \( r \rightarrow R \) to the coarser grid
      - Set the coarser grid correction \( V = 0 \) and improve it recursively
      - Prolongate the correction \( V \rightarrow v \) onto the finer grid
    - Correct \( u \rightarrow u + v \)
    - Perform post-smoothing Gauss-Seidel iterations and return improved \( u \)
• Is this worth it? What’s the scaling with $L$?

• Recall that Jacobi / Gauss-Seidel iterations are the most time-consuming parts of the calculation.
  – Single step: $\mathcal{O}(L^2)$

• Now this gets performed on the sequence of grids with:

  $$2^\ell \to 2^{\ell-1} \to 2^{\ell-2} \to \ldots \to 2^0 = 1$$

• So the total number is of order:

  $$L^2 \sum_{n=0}^{\ell} \frac{1}{2^{2n}} \leq L^2 \frac{1}{1 - \frac{1}{4}}.$$  

• So in this, the TOTAL is $\mathcal{O}(L^2)$ !!!!
Details of restricting residual to coarser lattice:
- Define the coarser lattice $H = 2h$
- Set the value to the average of the values on the four corners:

$$R_{I,J} = \frac{1}{4} \left[ r_{i,j} + r_{i+1,j} + r_{i,j+1} + r_{i+1,j+1} \right], \quad i = 2I - 1, \quad j = 2J - 1.$$
Details to prolong the correction to the finer lattice:

Need to solve the equation

\[ \nabla^2 V = -R(x, y) , \]

In the code this will be called “twoGrid”

Then we copy the value of \( V(I,J) \) into the four neighboring points on the finer lattice \( v(i,j) \):

\[ v_{i,j} = v_{i+1,j} = v_{i,j+1} = v_{i+1,j+1} = V_{I,J} , \quad i = 2I - 1 , \quad j = 2J - 1 . \]
Two possibilities:

- Cell centered: \(2^3 = 8 \rightarrow 2^2 = 4 \rightarrow 2^1 = 2 \rightarrow 2^0 = 1\).

- Grid centered: \(2^3 + 1 = 9 \rightarrow 2^2 + 1 = 5 \rightarrow 2^1 + 1 = 3\).

Note: grid-centered needs to one more point in each dimension.
The boundary points are specified as follows:

- Cell-centered: Boundary points move in space toward the center of the region at each coarsening (so care must be taken here)
- Vertex-centered: Boundary points do not move when lattice is coarsened

A little more convenient to use vertex-centered
Partial Differential Equations

• What about restriction (fine->coarse) and prolongation (coarse->fine) operations?

• Cell-centered:
  – Prolongation: Set the values on the fine to the value from the coarse
  – Restriction: Average fine points to get coarse points

• Vertex-centered:
  – Prolongation: use bilinear interpolation at which value at F at a coarse grid point is copied to 9 neighboring fine-grid points with weights:

\[
\begin{bmatrix}
\frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\
\frac{1}{2} & 1 & \frac{1}{2} \\
\frac{1}{4} & \frac{1}{2} & \frac{1}{4}
\end{bmatrix}.
\]

  – Restriction: Adjoint of the prolongation

\[
\begin{bmatrix}
\frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\
\frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\
\frac{1}{16} & \frac{1}{8} & \frac{1}{16}
\end{bmatrix}.
\]
Partial Differential Equations

• Improvements are to use more than one cycle
  – Repeat the two-grid iteration more than once
  – Full multigrid starts with coarse grid, then proceeds to finer grids

– Numerical Recipes Chapter 19 Section 6 goes over this

– Can look into them at your leisure
Parabolic PDES
Let’s now turn to parabolic differential equations

– Includes diffusion and time-dependent Schroedinger equation

\[ i\hbar \frac{\partial}{\partial t} \psi(x, t) = -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} \psi + V(x)\psi. \]

– Formal solution is:

\[ \psi(x, t) = e^{-\frac{i}{\hbar} \mathcal{H}t} \psi(x, 0), \quad \mathcal{H} \equiv -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} + V(x) = \mathcal{H}^\dagger, \]

– where \( \mathcal{H} \) is the hermitian Hamiltonian operator
Partial Differential Equations

• Two separate strategies:
  – “Marching” in time  
    • Similar to ODE technology, but now must account for derivatives in spatial dimension too!

  – Spectral analysis 
    • Just like in your classes, we can also solve the PDE in the Fourier domain, and it is often more convenient

Will examine both solutions
Partial Differential Equations

• First: Marching
The time-evolution is unitary, so the total probability is conserved:

\[
\left( e^{-\frac{i}{\hbar} \hat{H} t} \right)^\dagger = \left( e^{-\frac{i}{\hbar} \hat{H} t} \right)^{-1}, \quad \int |\psi(x, t)|^2 dx = \int |\psi(x, 0)|^2 dx.
\]

Diffusion equations, on the other hand, are NOT unitary:

\[
\frac{\partial}{\partial t} n(x, t) = D \frac{\partial^2}{\partial x^2} n(x, t) + C n(x, t).
\]

This leads to the characteristic damping.

Schroedinger’s equation is mathematically equivalent to diffusion with an imaginary diffusion constant (or a real one, in imaginary time):

\[
\frac{\partial \psi}{\partial (it)} = \frac{\hbar}{2m} \frac{\partial^2 \psi}{\partial x^2} - \frac{1}{\hbar} V(x) \psi.
\]
Partial Differential Equations

• We will look at a free particle as an instructive case:

\[ \psi(x, t) \sim e^{i(px - E t)/\hbar} \]

• where the momentum is

\[ p = \pm \sqrt{2mE} \]

• Of course, the plane wave is not localized in space
  – Probability is not =1 over all space, so not a “real” particle solution

• Can instead construct a Gaussian state:

\[ \phi(x) = \left( \frac{1}{\pi \sigma^2} \right)^{\frac{1}{4}} e^{-(x-x_0)^2/(2\sigma^2)} \]

  – But, this is stationary :

\[ \langle p \rangle = \int_{-\infty}^{\infty} dx \, \phi^*(x) \left( \frac{\hbar}{i} \frac{d}{dx} \right) \phi(x) = 0. \]
To get this to move, multiply by a phase factor:

$$\psi(x) = \phi(x) e^{ikx}$$

then we have:

$$\langle \psi | p | \psi \rangle = \int_{-\infty}^{\infty} dx \, \phi^*(x) e^{-ikx} \left( \frac{\hbar}{i} \frac{d}{dx} \right) e^{ikx} \phi(x)$$

$$= \int_{-\infty}^{\infty} dx \left[ \hbar k |\phi(x)|^2 - i\hbar \phi(x) \phi'(x) \right]$$

$$= \hbar k .$$

Expectation value of the energy is:

$$\langle \psi \left| \frac{p^2}{2m} \right| \psi \rangle = \frac{\hbar^2}{2m} \left( k^2 + \frac{1}{2\sigma^2} \right) ,$$

This is close to the classical result if the packet isn’t too narrow.
Our wavepacket is:

\[
\psi(x, 0) = \left( \frac{1}{\sqrt{2\pi\sigma^2}} \right)^{\frac{1}{4}} e^{ik_0x - \frac{(x-x_0)^2}{4\sigma^2}}
\]

- Moves to the right with speed \( \hbar k_0 / m \)
- \( \Psi \) is approximated on a lattice by an N-component complex vector
- If potential is a function of space alone, can precompute the quantity

\[
e^{-iV(\xi)} \delta_t / (2\hbar)
\]

- which can be used to speed up computational times
Partial Differential Equations

• Also examine finite difference methods
• Start with a forward time-centered scheme (FTCS):
  – Discretized equation:
    \[
    i\hbar\frac{\psi_{j+1}^{n+1} - \psi_j^n}{\delta_t} = -\frac{\hbar^2}{2m} \frac{\psi_{j+1}^n + \psi_{j-1}^n - 2\psi_j^n}{\delta_x^2} + V_j \psi_j^n,
    \]
  – This can be solved explicitly for the solution at the next time step:
    \[
    \psi_j^{n+1} = \psi_j^n - \frac{i\delta_t}{\hbar} \left[ -\frac{\hbar^2}{2m} \frac{\psi_{j+1}^n + \psi_{j-1}^n - 2\psi_j^n}{\delta_x^2} + V_j \psi_j^n \right].
    \]
  – If we introduce the column vector of values:
    \[
    \Psi^n \equiv \begin{pmatrix} \psi_1^n \\ \psi_2^n \\ \vdots \\ \psi_N^n \end{pmatrix},
    \]
  – Then the equation is (in matrix form):
    \[
    \Psi^{n+1} = \left( I - \frac{i\delta_t}{\hbar} H \right) \Psi^n.
    \]
Problem with this simplest scheme: always unstable

For instance, for an eigenvector we have:

\[ H \Psi^1 = E \Psi^1, \]

Then we’d compute:

\[ \Psi^{n+1} = \left(1 - \frac{i\delta_t E}{\hbar}\right) \Psi^n = \left(1 - \frac{i\delta_t E}{\hbar}\right)^2 \Psi^{n-1} = \ldots = \left(1 - \frac{i\delta_t E}{\hbar}\right)^n \Psi^1, \]

The magnitude of this is:

\[ |\Psi^{n+1}| = \left(\sqrt{1 + \frac{\delta_t^2 E^2}{\hbar^2}}\right)^n |\Psi^1| \rightarrow \infty, \quad \text{as} \quad n \rightarrow \infty. \]

Booooooo.
What about backward time space centered (BTCS) implicit differencing?

\[ i \hbar \frac{\psi_{j+1}^{n+1} - \psi_{j-1}^{n+1}}{\delta t} = -\frac{\hbar^2}{2m} \frac{\psi_{j+1}^{n+1} + \psi_{j-1}^{n+1} - 2\psi_j^{n+1}}{\delta x^2} + V_j \psi_{j+1}^{n+1}, \]

Can’t be solved exactly.

Three unknown quantities on the LHS of

\[ \psi_j^{n+1} + \frac{i \delta t}{\hbar} \left[ -\frac{\hbar^2}{2m} \frac{\psi_{j+1}^{n+1} + \psi_{j-1}^{n+1} - 2\psi_j^{n+1}}{\delta x^2} + V_j \psi_{j+1}^{n+1} \right] = \psi_j^n. \]

If we solve all N equations at the same time, we get a matrix form:

\[ \left( I + \frac{i \delta t}{\hbar} H \right) \Psi^{n+1} = \Psi^n, \]

with steps:

\[ \Psi^{n+1} = \left( I + \frac{i \delta t}{\hbar} H \right)^{-1} \Psi^n. \]
This one, on the other hand, is “stable”, but still wrong:

\[
\Psi^{n+1} = \left(1 + \frac{i\delta_t E}{\hbar}\right)^{-1} \Psi^n = \left(1 + \frac{i\delta_t E}{\hbar}\right)^{-2} \Psi^{n-1}
\]

\[
= \ldots = \left(1 + \frac{i\delta_t E}{\hbar}\right)^{-n} \Psi^1,
\]

Magnitude will be:

\[
|\Psi^{n+1}| = \left(\sqrt{1 + \frac{\delta_t^2 E^2}{\hbar^2}}\right)^{-n} |\Psi^1| \longrightarrow 0, \quad \text{as} \quad n \rightarrow \infty.
\]

No probability conservation, still boooooo.
Symmetric time space centered (STCS) differencing does the trick (Crank-Nicolson):

\[ \Psi^{n+1} = \Psi^n - \frac{i\delta_t}{2\hbar} \mathbf{H} (\Psi^n + \Psi^{n+1}) , \]

Matrix solution:

\[ \Psi^{n+1} = \left( I + \frac{i\delta_t}{2\hbar} \mathbf{H} \right)^{-1} \left( I - \frac{i\delta_t}{2\hbar} \mathbf{H} \right) \Psi^n . \]

– This is unitary:

\[ \Psi^{n+1} = \left[ 1 - \frac{i\delta_tE}{2\hbar} \right]^n \frac{1}{1 + \frac{i\delta_tE}{2\hbar}} \Psi^1 , \]

– And conserves probability at each step:

\[ |\Psi^{n+1}| = |\Psi^1| . \]
As you’d naively guess, this is also more accurate than the forward and backward only versions (by an order of magnitude).

To show explicitly, write the exact evolution operator for one time step:

\[ e^{-\frac{i}{\hbar} \mathcal{H} \delta t} \equiv e^{-z} = 1 - z + \frac{z^2}{2} - \frac{z^3}{6} + \ldots, \]

Here, we have

\[ z = \mathcal{O}(\delta_t) \]

Backward scheme:

\[ \frac{1}{1 + z} = 1 - z + z^2 - z^3 + \ldots = e^{-z} + \mathcal{O}(\delta_t^2), \]

Crank-Nicolson scheme:

\[ \frac{1}{1 + \frac{z}{2}} \left(1 - \frac{z}{2}\right) = \left(1 - \frac{z}{2} + \frac{z^2}{4} - \frac{z^3}{8} + \ldots\right) \left(1 - \frac{z}{2}\right) = 1 - z + \frac{z^2}{2} - \frac{z^3}{4} + \ldots = e^{-z} + \mathcal{O}(\delta_t^3). \]
We have the Schrödinger equation:

\[
i\hbar \frac{\partial}{\partial t} \psi(x, t) = -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} \psi + V(x)\psi.
\]

Solved using Crank-Nicolson algorithm:

\[
\Psi^{n+1} = \left( I + \frac{i\delta t}{2\hbar} H \right)^{-1} \left( I - \frac{i\delta t}{2\hbar} H \right) \Psi^n.
\]

And this is basically a matrix inversion problem!

Is it tractable?

– Incidentally, yes! It’s a sparse matrix!
Partial Differential Equations

- For instance, impose Dirichlet BC’s, and we get:

\[
\left( \frac{\partial^2 \psi}{\partial x^2} \right)_j^n = \frac{1}{\delta^2_x} \begin{cases} 
\psi_2^n - 2\psi_1^n, & \text{for } j = 1 \\
\psi_{j-1}^n + \psi_{j+1}^n - 2\psi_j^n, & \text{for } 1 < j < N \\
\psi_{N-1}^n - 2\psi_N^n, & \text{for } j = N
\end{cases}
\]

- If \( N = 5 \) then we get:

\[
H_{\text{Dirichlet}} = -\frac{\hbar^2}{2m\delta^2_x} \begin{pmatrix}
-2 & 1 & 0 & 0 & 0 \\
1 & -2 & 1 & 0 & 0 \\
0 & 1 & -2 & 1 & 0 \\
0 & 0 & 1 & -2 & 1 \\
0 & 0 & 0 & 1 & -2
\end{pmatrix}
\]

\[
+ \begin{pmatrix}
V_1 & 0 & 0 & 0 & 0 \\
0 & V_2 & 0 & 0 & 0 \\
0 & 0 & V_3 & 0 & 0 \\
0 & 0 & 0 & V_4 & 0 \\
0 & 0 & 0 & 0 & V_5
\end{pmatrix}
\]
• Imposing periodic BC’s we get:

\[
\left( \frac{\partial^2 \psi}{\partial x^2} \right)_j^n = \frac{1}{\delta^2_x} \begin{cases} 
\psi_n^N + \psi_2^n - 2\psi_1^n, & \text{for } j = 1 \\
\psi_{j-1}^n + \psi_{j+1}^n - 2\psi_j^n, & \text{for } 1 < j < N \\
\psi_{N-1}^n + \psi_1^n - 2\psi_N^n, & \text{for } j = N
\end{cases}
\]

• if N=5 then we get:

\[
H_{\text{Periodic}} = -\frac{\hbar^2}{2m\delta_x^2} \begin{pmatrix} -2 & 1 & 0 & 0 & 1 \\
1 & -2 & 1 & 0 & 0 \\
0 & 1 & -2 & 1 & 0 \\
0 & 0 & 1 & -2 & 1 \\
1 & 0 & 0 & 1 & -2 \end{pmatrix}
\]

\[
\begin{pmatrix} V_1 & 0 & 0 & 0 & 0 \\
0 & V_2 & 0 & 0 & 0 \\
0 & 0 & V_3 & 0 & 0 \\
0 & 0 & 0 & V_4 & 0 \\
0 & 0 & 0 & 0 & V_5 \end{pmatrix}.
\]
Partial Differential Equations

• So both of these are tridiagonal, so we can use our Matrix Methods from earlier in the semester to solve this very quickly

• Explicitly:
  – Note that
    \[
    \left( I + \frac{i\delta t}{2\hbar} H \right)^{-1} \left( I - \frac{i\delta t}{2\hbar} H \right) = Q^{-1} - I,
    \]
  – where:
    \[
    Q = \frac{1}{2} \left( I + \frac{i\delta t}{2\hbar} H \right)
    \]
  – So, we solve the linear equation:
    \[
    Q\chi = \Psi^n, \quad \chi = Q^{-1}\Psi^n,
    \]
  – We get an intermediate “chi”, which we can use to solve:
    \[
    \Psi^{n+1} = \chi - \Psi^n.
    \]
Partial Differential Equations

- Second: spectral analysis
To solve this ‘exactly’, can look at the exact solution in the Fourier domain (and keep in mind that we’re going to do the FFT later).

Write the S.E. as

$$i\hbar \frac{\partial \psi(x, t)}{\partial t} = -\frac{\hbar^2}{2m} \frac{\partial^2 \psi(x, t)}{\partial^2 x^2} + V(x) \psi(x, t) \equiv (\mathcal{T} + \mathcal{V}) \psi(x, t),$$

Here, $\mathcal{T}$ is a differential operator and $\mathcal{V}$ is a multiplicative operator in position space.

In Fourier domain:

$$\tilde{\psi}(p, t) = \frac{1}{\sqrt{2\pi \hbar}} \int_{-\infty}^{\infty} dx \ e^{-ipx/\hbar} \psi(x, t),$$

then we’d have:

$$i\hbar \frac{\partial \tilde{\psi}(p, t)}{\partial t} = \frac{p^2}{2m} \tilde{\psi}(p, t) + \frac{1}{\sqrt{2\pi \hbar}} \int_{-\infty}^{\infty} dq \ \tilde{\mathcal{V}}(p - q) \tilde{\psi}(q, t).$$
Here, the kinetic operator $T$ is multiplicative, while the potential operator $V$ is a convolution.

- So, this is an integral equation in the Fourier domain.

Formal solution:

$$\psi(x, t) = e^{-i(T+V)(t-t_0)/\hbar} \psi(x, t_0),$$

Where:

$$e^A = 1 + A + \frac{1}{2!} AA + \frac{1}{3!} AAA + \cdots$$

$T$ and $V$ do not commute here, so exponential is not amenable to numerical evaluation.
To make the discrete time approximation, we use a small time step $\delta t$:

$$\psi(t + \delta t) = e^{-i(T+V)\delta t/\hbar}\psi(x, t)$$

In this case, $T$ and $V$ can be disentangled (linear approximation $\implies$ they commute)

Can use Baker-Campell-Hausdorff formula:

- [http://en.wikipedia.org/wiki/Baker-Campbell-Hausdorff_formula](http://en.wikipedia.org/wiki/Baker-Campbell-Hausdorff_formula)
- This states that:

$$e^A e^B = e^C$$

- if and only if:

$$C = A + B + \frac{1}{2}[A, B] + \cdots$$
Partial Differential Equations

• Commutator is:

\[
[T, V] = -\frac{\hbar^2}{2m} \left[ \frac{d^2}{dx^2}, V(x) \right] = -\frac{\hbar^2}{2m} V''(x) - \frac{\hbar^2}{m} V'(x) \frac{d}{dx} \neq 0.
\]

• So, the simplest factorization has an error of \( O(\delta_t^2) \):

\[
e^{-i(T+V)\delta_t/\hbar} \approx e^{-iT\delta_t/\hbar} e^{-iV\delta_t/\hbar},
\]

• The symmetric factorization, however, has an error \( O(\delta_t^3) \):

\[
e^{-i(T+V)\delta_t/\hbar} \approx e^{-iV\delta_t/(2\hbar)} e^{-iT\delta_t/\hbar} e^{-iV\delta_t/(2\hbar)}
\]

– In addition, this is unitary so preserves the normalization of the wavefunction
Partial Differential Equations

- Split the time evolution operator into a symmetric factorization

- Evolve by:
  - Multiply by first half-step: \( \psi(x, t) \rightarrow \psi_1(x) = e^{-iV(x)\delta_t/(2\hbar)}\psi(x, t) \).
    (diagonal in position space)
  
  - Fourier transform to p-space:
    \[ \tilde{\psi}_1(p) = \frac{1}{\sqrt{2\pi\hbar}} \int_{-\infty}^{\infty} dx \ e^{-ipx/\hbar} \psi_1(x). \]

  - Multiply by kinetic evolution
    (diagonal in momentum space)
    \[ \tilde{\psi}_1(p) \rightarrow \tilde{\psi}_2(p) = e^{-ip^2\delta_t/(2m\hbar)}\tilde{\psi}_1(p). \]

  - Fourier transform back to x-space:
    \[ \psi_2(x) = \frac{1}{\sqrt{2\pi\hbar}} \int_{-\infty}^{\infty} dx \ e^{ipx/\hbar}\tilde{\psi}_2(p). \]

  - Multiply by the second half step evolution operator
    (diagonal in position space)
    \[ \psi(x, t + \delta_t) = e^{-iV(x)\delta_t/(2\hbar)}\psi_2(x) . \]
Hyperbolic PDES
We now turn to the final chapter in our investigation of PDE’s: hyperbolic waves.

This class covers a wide range of physical phenomena:
- Light waves
- Sound waves
- Water waves
- etc

The wave equation is:

\[ \frac{1}{c^2} \frac{\partial^2 u(r, t)}{\partial t^2} - \nabla^2 u(r, t) = R(r, t) , \]

- Wave speed
- Hyperbolic (+dt^2 - dx^2)
- Source term
Partial differential equations

• There is a unique solution if
  – the initial values of \( u(x, t_0) \) and \( \frac{\partial u(x, t)}{\partial t} \bigg|_{t=t_0} \) are specified
  – the boundary values are specified on a closed region

• So examine the 1-d equation with no source term:

\[
\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2},
\]

• This factorizes into simpler first-order equations:

\[
\frac{\partial^2}{\partial t^2} - c^2 \frac{\partial^2}{\partial x^2} = \left( \frac{\partial}{\partial t} + c \frac{\partial}{\partial x} \right) \left( \frac{\partial}{\partial t} - c \frac{\partial}{\partial x} \right).
\]
Partial differential equations

- Solutions to this equation are given by a superposition of left- and right-moving waves:

\[ u(x, t) = g(x + ct) + f(x - ct) , \]

\[ \left( \frac{\partial}{\partial t} - c \frac{\partial}{\partial x} \right) g(x + ct) = 0 , \quad \left( \frac{\partial}{\partial t} + c \frac{\partial}{\partial x} \right) f(x - ct) = 0 , \]

- Here, \( g \) and \( f \) are determined from initial conditions
Partial differential equations

• Examine one of the equations ("right-moving" one):

\[
\frac{\partial u(x, t)}{\partial t} = -c \frac{\partial u(x, t)}{\partial x},
\]

• The analytical solution here is:

\[
u(x, t) = f_0(x - ct),
\]

• where \(f_0(x)\) is the initial condition at \(t=0\)
• This basically means the initial shape simply propagates with a velocity \(c\)
  – This is called "advection"
• Contrast with cases where the wave shape depends on position
  – This is "convection" (hot fluid rising, colder fluid sinking, for instance)
Partial differential equations

• In the advective case, the flux is conserved:
\[
\frac{\partial \vec{u}}{\partial t} = -\frac{\partial \vec{F}(\vec{u})}{\partial x},
\]

Here, \( u(x,t) \) is a vector of functions, and the vector \( F \) is the conserved flux of \( u \).

• Now, suppose that \( u(x,t) \) is the density at point \( x \) and time \( t \).

• Total amount (mass) of fluid in a boundary is:
\[
M(t) = \int_{x_L}^{x_R} u(x, t) \, dx.
\]

• The rate of change of fluid in the region is:
\[
\frac{d}{dt} M(t) = \frac{d}{dt} \int_{x_L}^{x_R} u(x, t) \, dx = \int_{x_L}^{x_R} \frac{\partial u(x,t)}{\partial t} \, dx
\]
\[
= -\int_{x_L}^{x_R} \frac{\partial F(u(x, t))}{\partial x} \, dx = F(u(x_L, t)) - F(u(x_R, t)),
\]
Partial differential equations

• This should remind you of your vector calculus (Stoke’s theorem, etc)
  – http://en.wikipedia.org/wiki/Flux
  – http://en.wikipedia.org/wiki/Stokes'_theorem
Partial differential equations

• In 1-d, should be clear how we may discretize this
• Again can try the forward time-centered solution as we did last lecture (generalized Euler’s method!)

\[ u_{j}^{n+1} = u_{j}^{n} - \frac{c\delta_{t}}{2\delta_{x}} (u_{j+1}^{n} - u_{j-1}^{n}) . \]

• So we try the FTCS :

\[ u_{j}^{n+1} = u_{j}^{n} - \frac{c\delta_{t}}{2\delta_{x}} (u_{j+1}^{n} - u_{j-1}^{n}) . \]

• The spatial derivative was approximated by a symmetric difference :

\[ \frac{\partial u(x, t)}{\partial x} \approx \frac{u_{j+1}^{n} - u_{j-1}^{n}}{2\delta_{x}} . \]

• As we saw last class, the “bare bones” Euler-step-like solution is unconditionally unstable

\[ e^{ikj\delta_{x}} - \frac{c\delta_{t}}{2\delta_{x}} \left( e^{ik(j+1)\delta_{x}} - e^{ik(j-1)\delta_{x}} \right) = \left( 1 - i \frac{c\delta_{t}}{\delta_{x}} \sin(k\delta_{x}) \right) e^{ikj\delta_{x}} \equiv \xi e^{ikj\delta_{x}} . \]

• If \( u_{j}^{n} \sim e^{ikj\delta_{x}} \) : modes amplified by:
Partial differential equations

• Instead, try the "Lax" method:

\[ u_{j}^{n+1} = \frac{1}{2} (u_{j+1}^{n} + u_{j-1}^{n}) - \frac{c\delta_{t}}{2\delta_{x}} (u_{j+1}^{n} - u_{j-1}^{n}) \]

• The mode amplification factor in this case is:

\[ \xi = \frac{1}{2} (e^{ik\delta_{x}} + e^{-ik\delta_{x}}) - \frac{c\delta_{t}}{2\delta_{x}} (e^{ik\delta_{x}} - e^{-ik\delta_{x}}) \]

\[ |\xi|^{2} = \cos^{2}(k\delta_{x}) + \left( \frac{c\delta_{t}}{\delta_{x}} \right)^{2} \sin^{2}(k\delta_{x}) \]

• If we choose \( \delta_{t} = \frac{\delta_{x}}{c} \) then flux is exactly conserved

• Any other choice of \( \delta_{t} \) will make this either decay or grow without bound
Partial differential equations

• This is the Courant-Freidrichs-Lewy condition:
  \[
  \frac{c\delta t}{\delta x} \leq 1
  \]
  (CFL number)

• Consider the domain of dependency
• For any differencing scheme, the domain consists of the set of points in the “past cone”
• If the differencing domain is wider in x than the domain of dependency, then this is stable
• If the differencing domain is narrower, then unstable
Partial differential equations

- Can also add terms of order $\frac{\delta^2_t}{t}$ in the discretization
- Using $\frac{\partial u}{\partial t} = -c\frac{\partial u}{\partial x}$ then we get:

$$u(x, t + \delta_t) = u(x, t) + \delta_t \frac{\partial u}{\partial t} + \frac{\delta^2_t}{2} \frac{\partial^2 u}{\partial t^2} + \ldots$$

$$\approx u(x, t) - c\delta_t \frac{\partial u}{\partial x} + \frac{c^2 \delta^2_t}{2} \frac{\partial^2 u}{\partial x^2},$$

$$u_j^{n+1} = u_j^n - \frac{c\delta_t}{2\delta_x} (u_{j+1}^n - u_{j-1}^n) + \frac{c^2 \delta^2_t}{2\delta_x^2} (u_{j+1}^n + u_{j-1}^n - 2u_j^n).$$

- This is the “Lax-Wendroff” method
- The stability is the same CFL condition as before in the Lax method
- Note that the added term is a discretized diffusive term

$$\frac{\partial n(x, t)}{\partial t} = D \frac{\partial^2 n(x, t)}{\partial x^2}, \quad n_i^{n+1} = n_i^n + \frac{D\delta_t}{\delta_x^2} (n_{i+1}^n + n_{i-1}^n - 2n_i^n).$$

- General feature: diffusive terms in recurrence formulae have damping effects on the amplitude
Partial differential equations

• Can also consider nonlinear wave equations
  – Don’t preserve shape in general
  – Linear wave equation has linear dispersion!

• Dispersion is the relation between wave number and frequency.
  – Plane wave: \( u(x, t) \sim e^{i(kx-\omega t)} \)  \( \Rightarrow \) \((-i\omega - ick)(-i\omega + ick) = 0 \)
    \( \Rightarrow \omega = \pm ck \).

  • Here, all the modes move with the same velocity \( c \)
  • Wave velocity is \( \omega / k \)

• What if the velocity depends on the wave number?
  – Example: \( \frac{\partial u(x, t)}{\partial t} = -c \frac{\partial u(x, t)}{\partial x} - d \frac{\partial^3 u(x, t)}{\partial x^3} \).
  – Plugging in (plane wave), we get a dispersion:
    \( \omega = ck - dk^3 \). Wave velocity depends on \( k \)!
Partial differential equations

• Now let’s go back to advection equation and add a diffusive term

\[ \frac{\partial u(x, t)}{\partial t} = -c \frac{\partial u(x, t)}{\partial x} + D \frac{\partial^2 u(x, t)}{\partial x^2}, \]

• From plane wave, we get the dispersion relation:

\[ \omega = ck - iDk^2 \Rightarrow e^{i(k(x-ct)-Dk^2t)}, \]
Partial differential equations

- Some nonlinear equations can have traveling waves
- Example is Burgers’ equation:
  \[ \frac{\partial u}{\partial t} = -\alpha \frac{\partial u}{\partial x} - \beta u \frac{\partial u}{\partial x}, \]
  - [http://en.wikipedia.org/wiki/Burgers'_equation](http://en.wikipedia.org/wiki/Burgers'_equation)
- The last term is nonlinear in the wave amplitude
- Can solve by calculating partial derivatives:
  \[ \frac{\partial u}{\partial t} = -(\alpha + \beta u)f' - \beta f't \frac{\partial u}{\partial x} \quad \Rightarrow \quad \frac{\partial u}{\partial t} = -(\alpha + \beta u)f'/(1 + \beta f't), \]
  \[ \frac{\partial u}{\partial x} = f' - \beta f't \frac{\partial u}{\partial x} \quad \Rightarrow \quad \frac{\partial u}{\partial t} = f'/(1 + \beta f't). \]
- This is solved if we have a right-moving wave with function
  \[ u(x, t) = f(x - (\alpha + \beta u)t), \]
- This wave moves with velocity \( c = \alpha + \beta u(x, t) \)
Partial differential equations

• Here, the velocity depends on the density of the wave!
• This leads to breaking and shock fronts:
The Burgers’ equation was introduced in 1948 as a simple model of shock propagation

\[
\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2},
\]

First, set \( \nu = 0 \) and we get

\[
\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = 0.
\]

Compare to the linear wave equation:

\[
\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} = 0,
\]

Schematically the speed is equal to “\( u \)”!

Peaks travel faster than troughs in the wave

Eventually we get breaking, which we cannot represent as a function since it is multi-valued

Passes through a shock front (solution is discontinuous)
Partial differential equations

• This kind of PDE was studied by Godunov in 1959
  S.K. Godunov, Mat. Sb. 47, 271 (1959)

• This is a class of “Riemann problem”
  – IVP for a PDE which has a piecewise constant initial value function, with a discontinuity (like a step function)

• Need to find an exact or approximate algorithm for this
  – called a “Riemann solver”

\[
\begin{align*}
    u_{j+1}^{n+1} &= u_j^n - \frac{\tau}{h} \left[ F_{j+\frac{1}{2}} - F_{j-\frac{1}{2}} \right] + \frac{\nu \tau}{h^2} \left[ u_{j+1} + u_{j-1} - 2u_j \right], \\
    F_{j±\frac{1}{2}} &= \text{the average flux on the cells to the left and right of the lattice point } j, \text{ respectively}
\end{align*}
\]

• Solve these from Riemann problems in the cells to the right and left of j using “upwind” initial data:

\[
\begin{align*}
    u_j^{(+)} &= \begin{cases} 
    u_j & \text{if } u_j > 0 \\
    0 & \text{otherwise} 
    \end{cases} \quad \begin{cases} 
    u_j & \text{if } u_j < 0 \\
    0 & \text{otherwise} 
    \end{cases}
\end{align*}
\]
Partial differential equations

• The solution in the left cell is:

\[
F_{j-\frac{1}{2}} = \max \left\{ \frac{1}{2} (u_{j-1}^{(+)})^2, \frac{1}{2} (u_j^{(-)})^2 \right\},
\]

• and on the right it is:

\[
F_{j+\frac{1}{2}} = \max \left\{ \frac{1}{2} (u_j^{(+)})^2, \frac{1}{2} (u_{j+1}^{(-)})^2 \right\}.
\]