Overview of slides 07

Recap + extensions

- Many boundary-value odes can be discretised $\longrightarrow$ matrix equations
- General methods for solving matrix equation $Ax = b$:
  - Direct elimination (Gaussian elimin, Gauss-Jordan elimin)
  - Iterative techniques for sparse matrices:
    Conjugate gradient and variants, GMRES, Arnoldi,....
    Collectively known as Krylov subspace techniques
  - Iteration (Jacobi, Gauss-Seidel,...) for diagonally dominant matrices
- Sparse matrices appear in many physical problems
  - Savings in storage and computation:
    Sizes $\gg 10^4$ become possible on desktop
  - Many numerical methods are tailor-made for sparse matrices

Iterative methods: Jacobi & Gauss-Seidel

Want to solve $Ax = b$, i.e., find elements of x in terms of elements of A and b.
i.e. want all x_i in terms of the a_{ij} and the b_i.

'Solve' for diagonal elements:

$$a_{ii}x_i = b_i - \sum_{k \neq i} a_{ik}x_k \qquad \Longrightarrow \qquad x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{k \neq i} a_{ik}x_k\right)$$

This is not a solution; right side has x_i's.

BUT: could use as an iterative scheme.

Reminder: Iterative method for 1 variable

Fixed point iteration

Solving $f(x) = 0 \quad \longrightarrow \quad$ Rewrite as $x = g(x)$

Iterate $x^{(n+1)} = g\left(x^{(n)}\right)$. Solution is the fixed point of this iteration

Generally, infinite ways of rewriting as iteration

Can converge if $g'(x) \in (-1, 1)$ in region containing solution.

Iterative methods require a termination condition.

While loops are often appropriate for iterations

Reminder: Iterative method for 1 variable

Example(s)

Solving $x^2 = 2 \rightarrow$ Rewrite: $2x^2 = x^2 + 2 \implies x = \frac{x}{2} + \frac{1}{x}$

Iterate $\boxed{x^{(n+1)} = \frac{1}{2}x^{(n)} + 1/x^{(n)}}$

This is Newton-Raphson for $f(x) = x^2 - 2$ SHOW!

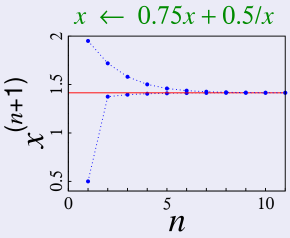
Infinite number of other ways to rewrite $x^2 = 2 \rightarrow$

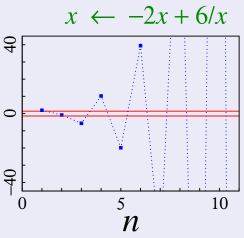
$$x^{(n+1)} = \frac{3}{4}x^{(n)} + \frac{1}{2x^{(n)}}$$

Converges

$$x^{(n+1)} = -2x^{(n)} + 6/x^{(n)}$$

Doesn't converge





Iterative methods: Jacobi & Gauss-Seidel

'Solve' $A\mathbf{x} = \mathbf{b}$ for diagonal elements:

$$a_{ii} x_i = b_i - \sum_{k \neq i} a_{ik} x_k \qquad \Longrightarrow \qquad x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{k \neq i} a_{ik} x_k \right) \tag{1}$$

Using Eq.(1) as basis for iterative methods $\longrightarrow$

- Jacobi
- Gauss-Seidel
- Successive over-relaxation (SOR)

Jacobi iteration

Jacobi iteration

Use Eq.(1) directly as iterative algorithm, starting with guess $x^{(0)}$:

$$x_i^{(m+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{k \neq i} a_{ik} x_k^{(m)} \right)$$

Example

For a 3×3 system:

$$\begin{aligned}
x_1^{(m+1)} &= \frac{1}{a_{11}} \left(b_1 - a_{12} x_2^{(m)} - a_{13} x_k^{(m)} \right) \\
x_2^{(m+1)} &= \frac{1}{a_{22}} \left(b_2 - a_{21} x_1^{(m)} - a_{23} x_3^{(m)} \right) \\
x_3^{(m+1)} &= \frac{1}{a_{33}} \left(b_3 - a_{31} x_1^{(m)} - a_{32} x_2^{(m)} \right)
\end{aligned}$$

Gauss-Seidel iteration

For calcluating $x_1^{(m+1)}$, use the m-th estimates for all x_i, i.e., $x_i^{(m)}$.

For calcluating $x_2^{(m+1)}$, could use improved estimate for $x_1 \longrightarrow$

$x_1^{(m+1)}$ is already available.

Gauss-Seidel method for calculating $x_i^{(m+1)} \longrightarrow$

Use current estimate, $x_k^{(m+1)}$, for $k = 1, \ldots, i-1$.

Use previous estimate, $x_k^{(m)}$, for $k \geq i$.

Example

Gauss-Seidel for a 3×3 system:

$$x_1^{(m+1)} = \frac{1}{a_{11}} \left(b_1 - a_{12} x_2^{(m)} - a_{13} x_k^{(m)} \right)$$
$$x_2^{(m+1)} = \frac{1}{a_{22}} \left(b_2 - a_{21} x_1^{(m+1)} - a_{23} x_3^{(m)} \right)$$
$$x_3^{(m+1)} = \frac{1}{a_{33}} \left(b_3 - a_{31} x_1^{(m+1)} - a_{32} x_2^{(m+1)} \right)$$

Gauss-Seidel iteration

Gauss-Seidel iteration

Modify iteration to use values from ongoing iteration step:

$$x_i^{(m+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{k<i} a_{ik} x_k^{(m+1)} - \sum_{k\geq i} a_{ik} x_k^{(m)} \right)$$

Successive over-relaxation (SOR)

Performing Gauss-Seidel once on $\vec{x}^{(0)}$, we obtain $\vec{x}_{GS}^{(1)}$.
In Gauss-Seidel, we took as the next approximation:

$$\vec{x}_{GS}^{(1)} = \vec{x}^{(0)} + \left(\vec{x}_{GS}^{(1)} - \vec{x}^{(0)}\right).$$

The part in brackets 'relaxes' our estimate toward correct solution.

Over-relaxation

Multiply the relaxation effect by a factor ω:

$$\vec{x}^{(m+1)} = \vec{x}^{(m)} + \omega\left(\vec{x}_{GS}^{(m+1)} - \vec{x}^{(m)}\right)$$

Picking optimal ω is an important and tricky topic.

Jacobi & Gauss-Seidel in matrix form

General scheme

$$A\vec{x} = \vec{b} \quad \implies \quad B\vec{x} = B\vec{x} - A\vec{x} + \vec{b} = (B-A)\vec{x} + \vec{b}$$

Here B is any matrix. 'Solve': $\qquad \vec{x} = B^{-1}(B-A)\vec{x} + B^{-1}\vec{b}$

Can use this as an iterative scheme:

$$\boxed{\vec{x}^{(m+1)} = B^{-1}(B-A)\vec{x}^{(m)} + B^{-1}\vec{b}}$$

Jacobi, Gauss-Seidel $\quad\longrightarrow\quad$ different choices of B.

Lower-triangular part, diagonal part, upper-triangular part

$A = L + D + U$. Example:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ a_{21} & 0 & 0 \\ a_{31} & a_{32} & 0 \end{pmatrix} + \begin{pmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{pmatrix} + \begin{pmatrix} 0 & a_{12} & a_{13} \\ 0 & 0 & a_{23} \\ 0 & 0 & 0 \end{pmatrix}$$

Jacobi & Gauss-Seidel in matrix form

Jacobi

Choosing $B = D$: $\boxed{\vec{x}^{(m+1)} = -D^{-1}(L+U)\vec{x}^{(m)} + D^{-1}\vec{b}}$

Show: this is identical to per-element Jacobi itertion (D is easy to invert)

Gauss-Seidel

Choose $B = D + L$: $\boxed{\vec{x}^{(m+1)} = -(D+L)^{-1}U\vec{x}^{(m)} + (D+L)^{-1}\vec{b}}$

Show!

SOR

Corresponds to $B = D/\omega + L$.
Iteration formula can be written in terms of ω and matrices D, L, U.

Convergence of iterative algorithms

The linear iteration $\vec{x}^{(m+1)} = C\vec{x}^{(m)} + \vec{d}$ converges if the largest eigenvalue of C has magnitude < 1

$$\Longrightarrow \quad \text{spectral radius, } \rho(C) < 1.$$

Jacobi & Gauss-Seidel

$$C = -D^{-1}(L+U) \quad \text{or} \quad C = -(D+L)^{-1}U$$

Diagonal dominance of A is sufficient:

$$\boxed{|a_{ii}|^2 > \sum_{k \neq i} |a_{ik}|^2}$$

Stringent!
Not likely for random matrix.
But common for ODE/PDE

SOR

Necessary: $\omega \in [0, 2]$. Sufficient conditions difficult.

Done with Linear Algebra

.... for now.

Next: Partial Differential Equations

Partial differential equations

A very large number of physical problems are formulated as pdes:

Schrödinger equation for 1 particle

$$i\hbar\frac{\partial\Psi}{\partial t} = \left[-\frac{\hbar^2}{2m}\nabla^2 + V(\vec{x})\right]\Psi$$

Poisson equation

$$\nabla^2 u = \rho(\vec{x})$$

Navier–Stokes equation

$$\rho(\frac{\partial\vec{u}}{\partial t} + \vec{u}\nabla\cdot\vec{u}) = -\nabla p + \eta\nabla^2\vec{u} + (\frac{1}{3}\eta + \zeta)\nabla(\nabla\cdot\vec{u})$$

Korteweg–de Vries equation

$$\frac{\partial\phi}{\partial t} + \frac{\partial^3\phi}{\partial x^3} + 6\phi\frac{\partial\phi}{\partial x} = 0$$

and many more!!

Classification

Traditional classification

Second-order pde, 2 variables:

$$a\frac{\partial^2 u}{\partial x^2} + 2b\frac{\partial^2 u}{\partial x \partial y} + c\frac{\partial^2 u}{\partial y^2} + d\frac{\partial u}{\partial x} + e\frac{\partial u}{\partial y} + fu + g = 0$$

elliptic	$b^2 - ac > 0$	Poisson
parabolic	$b^2 - ac = 0$	Diffusion
hyperbolic	$b^2 - ac < 0$	Wave

Classification

Computational classification

- Time evolution (initial value problem)
  - Mostly hyperbolic, parabolic
  - Most important issue: stability
- Static solution (boundary value problem)
  - Mostly elliptic
  - Most important issue: efficiency

Many pdes are a mix of parabolic, hyperbolic, elliptic

Methods

- Finite differences
- Spectral methods
- Finite elements
- Monte Carlo
- Variational
- ...

Finite differences

For 2 spatial variables:

1. Replace x and y with a discrete grid (i, j):

$$x_i = x_0 + i\delta_x\,, \quad y_j = y_0 + j\delta_y$$

   Function is given by values on lattice points: $u(x, y) \to u_{ij}$

2. Replace derivatives by finite differences

3. Result can be written as matrix equation $Au = b$
   - u, b are $N = n_x \times n_y$ vectors
   - A is a sparse $N \times N$ matrix

4. Use direct/iteration/Fourier methods to solve for u

In isotropic systems (with static solutions) we usually choose $\delta_x = \delta_y = a$

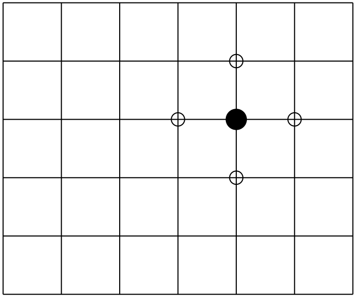
Contrast: for time evolution we need $\delta_t \ll \delta_x$

Poisson equation (elliptic)

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

Discrete derivative: $\dfrac{\partial^2\Phi}{\partial x^2} \rightarrow \dfrac{\Phi_{i+1,j} - 2\Phi_{i,j} + \Phi_{i-1,j}}{a^2}$

$$\Longrightarrow \quad \Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j-1} + \Phi_{i,j+1} - 4\Phi_{i,j} = a^2\rho_{ij} \equiv \hat{\rho}_{ij}$$



To write as matrix equation, relabel coordinates: $k = i + N_x j$

We get $A\mathbf{\Phi} = \hat{\rho}$

$$A_{mn} = \delta_{m,n+1} + \delta_{m,n-1} + \delta_{m,n+N_x} + \delta_{m,n-N_x} - 4\delta_{mn}$$

A is sparse: most $A_{mn} = 0$!

Boundary conditions

Important classes of boundary conditions:

- Dirichlet $\Phi(x) = b(x)$ on boundary
- Neumann $\partial_n \Phi(x) = b'(x)$ on boundary
  $\partial_n \Phi$ is normal derivative, orthogonal to boundary
- Periodic $\Phi(x + L) = \Phi(x)$
  This only works for regular (rectangular) boundaries!
  Used mostly when we are interested in bulk behaviour
  $\rightarrow$ take large volume limit
  Topology changes: line $\rightarrow$ circle, rectangle $\rightarrow$ torus, ...

Implementation

Dirichlet

If one of the neighbours is on the boundary, it gets replaced by the boundary value b_j in the finite difference:

$$\nabla^2\Phi_{1,j} \quad \rightarrow \quad b_{0j} + \Phi_{2,j} + \Phi_{1,j-1} + \Phi_{1,j+1} - 4\Phi_{1,j}$$

As in ODE boundary value problem:

Subtract boundary values from source term

Implementation

Neumann

Several possibilities, as in ODE.
E.g., use forward/backward derivative on the boundary:

$$b'_{0j} = (\partial\Phi)_{0j} = \Phi_{1j} - \Phi_{0j}$$

So we replace $\Phi_{0j} \rightarrow \Phi_{1j} - b'_{0j}$ in finite diff operator

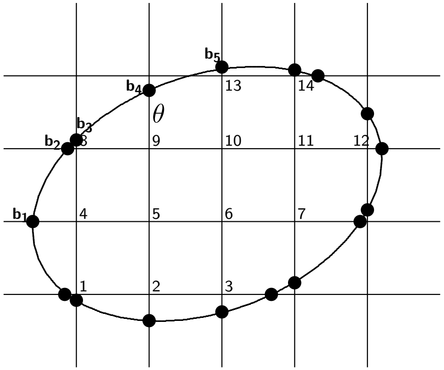
Disadvantage: becomes spatially 1st order.
2nd order? $\longrightarrow$ similar methods as in ODE boundary value problem.

Periodic

If we are at the edge, the nearest neighbour is at opposite edge!
So $\Phi_{0j} \rightarrow \Phi_{Lj}$; $\Phi_{L+1,j} \rightarrow \Phi_{1j}$ etc in all finite differences

Irregular boundaries

Map boundary onto the regular grid. Number interior grid points and boundary points. Find distances from interior points to boundary. Taylor expand differential operators near boundary.



Taylor expansion around Φ_9:

$$b_4 \approx \Phi_9 + \theta\partial_y\Phi_9 + \frac{1}{2}\theta^2\partial_y^2\Phi_9$$

$$\Phi_5 \approx \Phi_9 - \partial_y\Phi_9 + \frac{1}{2}\partial_y^2\Phi_9$$

$$\partial_y^2\Phi_9 \approx \frac{\theta\Phi_5 + b_4 - (1+\theta)\Phi_9}{\theta(1+\theta)}$$

$\rightarrow$ Difference approx for $\partial_y^2\Phi_9$

Repeat for all boundary points!

Some Python tricks

`X.reshape((M,N))`	rearranges `X` into a `M`×`N` matrix
`X.reshape((M,N,P))`	rearranges X into a `M`×`N`×`P` matrix
	etc.
`X,Y = np.meshgrid(x,y)`	creates 2d arrays `X,Y` from vectors x,y
`scipy.sparse.eye(N)`	creates the sparse `N`×`N` unit matrix

Summary

- Direct Iteration for Linear Systems of Eqs
  - Jacobi, Gauss-Seidel, SOR
  - Matrix formulation
- Classification of pdes:
  - time evolution, initial value (mostly parabolic, hyperbolic)
  - static solution, boundary value (mostly elliptic)
- Finite differences transform PDEs into matrix equations
  - Similar to BVP's for ODEs
- Boundary conditions
  - Dirichlet: subtract boundary terms from source term
  - Neumann: modify finite diff on boundary, add boundary term to source
  - Periodic: wrap finite differences around boundary