CS 450: Numerical Anlaysis[1]

Partial Differential Equations

University of Illinois at Urbana-Champaign

[1] *These slides have been drafted by Edgar Solomonik as lecture templates and supplementary material for the book "Scientific Computing: An Introductory Survey" by Michael T. Heath (slides).*

Partial Differential Equations

- *Partial differential equations (PDEs)* describe physical laws and other continuous phenomena:
- The *advection PDE* describes basic phenomena in fluid flow,

  $$u_t = -a(t,x)u_x$$

  where $u_t = \partial u/\partial t$ and $u_x = \partial u/\partial x$.

Types of PDEs

- Some of the most important PDEs are *second order*:
- The *discriminant* determines the canonical form of second-order PDEs:

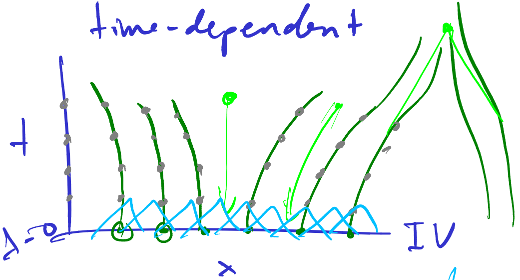
Characteristic Curves

- A *characteristic* of a PDE is a level curve in the solution:

- More generally, characteristic curves describe curves in the solution field $u(t, x)$ that correspond to solutions of ODEs, e.g. for $u_t = -a(t, x)u_x$ with $u(0, x) = u_0(x)$,

PDE Discretization Overview

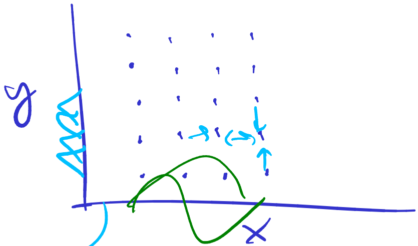
time-dependent



· reduce to ODEs (semi-discrete)

fully discrete

time-independent



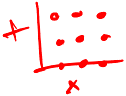
discretization methods

- sparse linear equations
  - sparse direct methods (LU)
  - sparse iterative methods
- FFT / multigrid

Method of Lines

- *Semidiscrete methods* obtain an approximation to the PDE by solving a system of ODEs. Consider the heat equation,

$\Delta x = x_{i+1} - x_i$



$$u_t = cu_{xx} \text{ on } 0 \le x \le 1, \quad u(0,x) = f(x), u(t,0) = u(t,1) = 0.$$

$t=0$

$$u_{xx}(t,x_i) \approx \frac{u(t,x_{i-1}) - 2u(t,x_i) + u(t,x_{i+1})}{\Delta x^2}$$

$y_{i-1}(t)$ $y_i(t)$ $y_{i+1}(t)$

$$\begin{bmatrix} y_1(t) \\ \vdots \\ y_n(t) \end{bmatrix} = y'(t) = A\,y(t) \qquad A = \Delta x^{2}\begin{bmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & & \ddots & 1 \\ & & 1 & -2 \end{bmatrix}$$

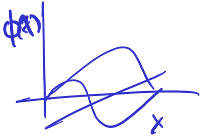
linear constant-coeff system of ODEs

- This *method of lines* often yields a stiff ODE:

may need small Δt — time discretization

Semidiscrete Collocation

- Instead of finite-differences, we can express $u(t,x)$ in a spatial basis $\phi_1(x), \dots, \phi_n(x)$ with time-dependent coefficients $\alpha_1(t), \dots, \alpha_n(t)$:



$$u(t,x) \approx v_{\varphi,\alpha}(t,x) = \sum_{i=1}^{n} \alpha_i(t)\varphi_i(x)$$

collocation method → system of ODEs,

$x_1, \dots, x_n$

$v_{\varphi,\alpha}(t, x_i)$ satisfies PDE

- For the heat equation $u_t = cu_{xx}$, we obtain a linear constant-coefficient vector ODE:

$$\frac{dv}{dt}(t, x_j) = \frac{d^2 v}{dx^2}(t, x_j)$$

precompute

vector-valued

$$\sum_{i=1}^{n} \alpha_i'(t)\underbrace{\varphi_i(x_j)}_{m_{ji}} = c\sum_{i=1}^{n} \alpha_i(t)\underbrace{\varphi_i''(x_j)}_{N_{ji}} \Rightarrow M\alpha'(t) = cN\alpha(t)$$

$$\alpha'(t) = cM^{-1}N\alpha(t)$$

Fully Discrete Methods

- Generally, both time and space dimensions are discretized, either by applying an ODE solver to a semidiscrete method or using finite differences.
  - *Again consider the heat equation* $u_t = cu_{xx}$ *and discretize so* $u_i^{(k)} \approx u(t_k, x_i)$,

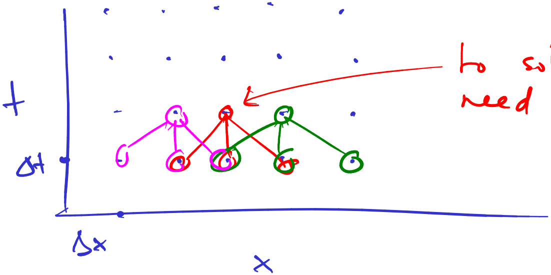
$u(t_{k+1}, x_i)$

$$\frac{u_i^{(k+1)} - u_i^{(k)}}{\Delta t} = \frac{u_{i+1}^{(k)} - 2u_i^{(k)} + u_{i-1}^{(k)}}{(\Delta x)^2}$$

equation for each $u_i^{(k)}$, $i \in \{1, \dots n\}$, $k \in \{1, \dots n\}$

for boundary, enforce other equations

  - *This iterative scheme corresponds to a 3-point stencil*,



to solve for $u_i^{(k+1)}$ need $u_i^{(k)}$, $u_{i+1}^{(k)}$, $u_{i-1}^{(k)}$

Implicit Fully Discrete Methods

- Using Forward Euler's method for the heat equation, stability requirement is

$$\Delta t \le c(\Delta x)^2$$

to improve accuracy e.g. by reducing Δx by 2, have to reduce Δt by 4

- This step-size restriction on stability can be circumvented by use of implicit time-stepper, such as backward Euler,

$$\frac{u_i^{(k)} - u_i^{(k-1)}}{\Delta t} = \frac{u_{i+1}^{(k)} - 2u_i^{(k)} + u_{i-1}^{(k)}}{\Delta x^2} \Rightarrow$$

unknown

obtain a system of equations to solve for each Δt

- Using the trapezoid method to solve the ODE we obtain the second-order *Crank-Nicolson method*,

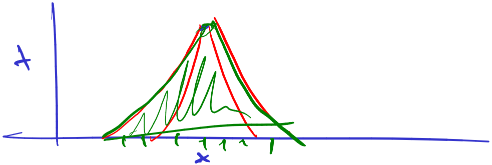
Convergence and Stability

- *Lax Equivalence Theorem*: consistency + stability = convergence
  - *Consistency means that the local truncation error goes to zero, and is easy to verify by Taylor expansions.*
  - *Stability implies that the approximate solution at any time t must remain bounded.*
  - *Together these conditions are necessary and sufficient for convergence.*
- Stability can be ascertained by spectral or Fourier analysis:
  - *In the method of lines, we saw that the eigenvalues of the resulting ODE define the stability region.*
  - *Fourier analysis decomposes the solution into a sum of harmonic functions and bounds their amplitudes.*

CFL Condition

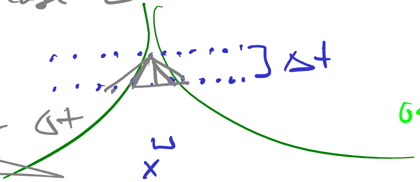
- The domain of dependence of a PDE for a given point (t, x) is the portion of the problem domain influencing this point through the PDE:

characteristic curves

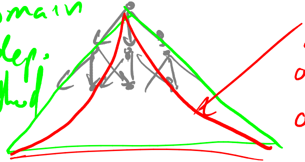


- *The Courant, Friedrichs, and Lewy (CFL) condition states that* a *necessary* condition for an explicit finite-differencing scheme to be stable for a hyperbolic PDE is that the domain of the dependence of the PDE be contained in the domain of dependence of the scheme:

if we decrease Δx



domain of dep. of method



domain of dep. of PDE

Time-Independent PDEs

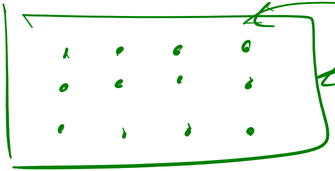
- We now turn our focus to time-independent PDEs as exemplified by the *Helmholtz equation*:

$$u_{xx} + u_{yy} + \lambda u = f(x, y)$$

$\lambda = 0 \Rightarrow$ Poisson

also $f(x,y) = 0 \Rightarrow$ Laplace $\qquad u_{xx} + u_{yy} = 0$

- We discretize as before, but no longer perform time stepping:



Boundary conditions on surface

Finite-Differencing for Poisson

- Consider the Poisson equation with equispaced mesh-points on $[0, 1]$:

$$\boxed{u_{xx} + u_{yy} = f(x,y)}$$

$$u_{ij} \approx u(x_i, y_j)$$

$x_1 \; \cdots \; x_n$

$y_1 \; \cdots \; y_n$

$$\begin{matrix} (x_1, y_1) & \cdots & (x_n, y_1) \\ \vdots & & \vdots \\ (x_1, y_n) & & (x_n, y_n) \end{matrix}$$

$$\frac{u_{i+1,j} - 2u_{ij} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{ij} + u_{i,j-1}}{\Delta y^2} = f(x_i, y_j)$$

$\Delta x = \Delta y = h$

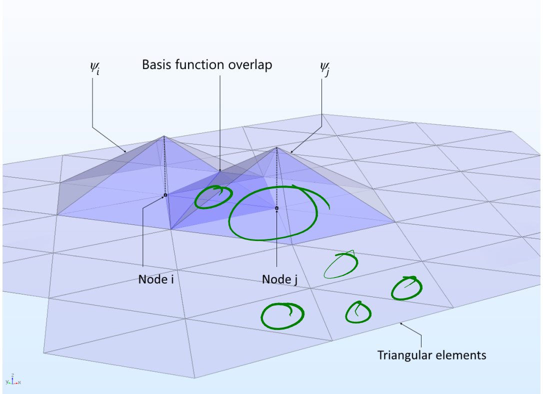
tridiagonal matrix $\frac{1}{h^2}\begin{bmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{bmatrix} = D$

$$(\underset{n \times n}{I} \otimes D + \underset{n \times n}{D} \otimes I)u = F \qquad A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & & \vdots \\ a_{n1}B & \cdots & a_{nn}B \end{bmatrix}$$

$$A\cancel{u} = F$$

Multidimensional Finite Elements

- There are many ways to define localized basis functions, for example in the 2D FEM method[2]:



Sparse Linear Systems

- Finite-difference and finite-element methods for time-independent PDEs give rise to sparse linear systems:
  - *typified by the 2D Laplace equation, where for both finite differences and FEM,*

$$\underbrace{(I \otimes D + D \otimes I)}_{A} x = b$$

$A \to \mathbb{R}^{n \times n}$ has $O(n)$ nonzeros

$O(1)$ nonzeros/row

- *Direct methods* apply LU or other factorization to $\boldsymbol{A}$, while *iterative methods* refine $\boldsymbol{x}$ by minimizing $\boldsymbol{r} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$, e.g. via Krylov subspace methods.

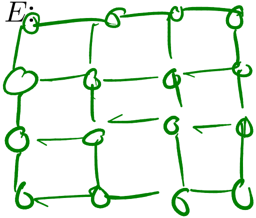
exact solution, reliable, but expensive

Direct Methods for Sparse Linear Systems

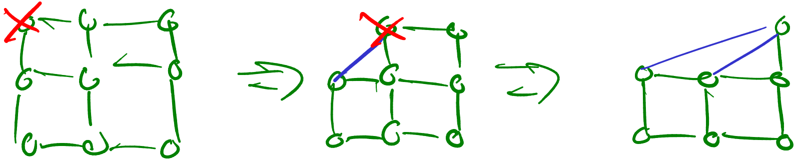
- It helps to think of $\boldsymbol{A}$ as the adjacency matrix of graph $G = (V, E)$ where $V = \{1, \dots n\}$ and $a_{ij} \neq 0$ if and only if $(i, j) \in E$:

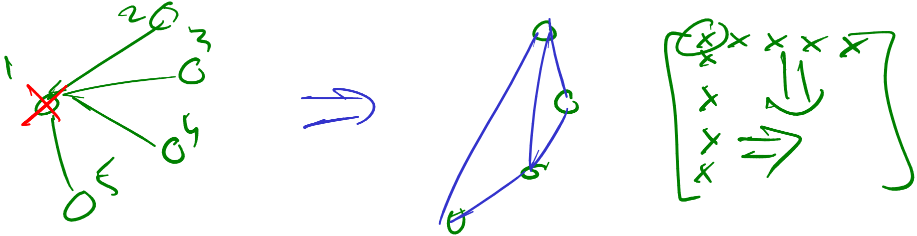
$$\begin{array}{cc} & \text{vertices} \\ \text{vertices} & \begin{bmatrix} \quad \text{edges} \quad \end{bmatrix} \end{array}$$

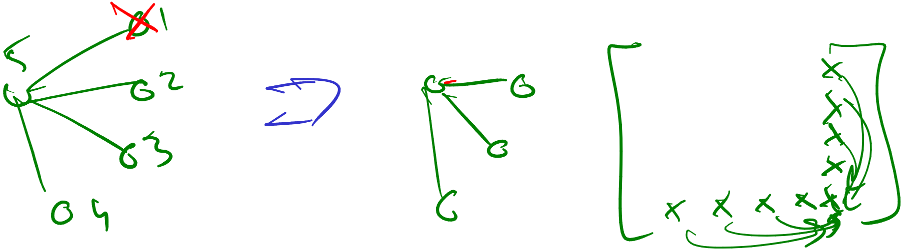
$A_{ij} \neq 0$



- Factorizing the lth row/column in Gaussian elimination corresponds to removing node i, with nonzeros (new edges) introduces for each k, l such that (i, k) and (i, l) are in the graph.




1
2
3
4
5


1
2
3
4
5

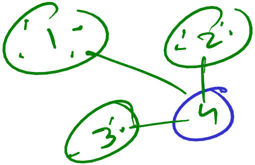
Demo: *Sparse Matrix Factorizations and Fill-In*

Vertex Orderings for Direct Methods

- Select the node of minimum degree at each step of factorization:

  bounded amount of fill at each step

- Graph partitioning also serves to bound fill, remove vertex separator $S \subset V$ so that $V \setminus S = V_1 \cup \cdots \cup V_k$ become disconnected, then order $V_1, \ldots, V_k, S$:



$$\begin{bmatrix} A_{11} & 0 & 0 & A_{14} \\ 0 & A_{22} & 0 & A_{24} \\ 0 & 0 & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix}$$

- *Nested dissection* ordering partitions graph into halves recursively, ordering each separator last.

Sparse Iterative Methods

- Sparse iterative methods avoid overhead of fill in sparse direct factorization. *Matrix splitting* methods provide the most basic iterative methods:

$$\underline{M x_{k+1} = N x_k + b}$$

$$A = M + N$$

fixed point scheme

$$g(x) = \underline{M^{-1} N x} + M^{-1} b$$

so need that

$$g(x^*) = x^*$$

iff $A x^* = b$

$$\rho(g(x)) < 1$$

Sparse Iterative Methods

- The *Jacobi method* is the simplest iterative solver:

$$A = D + L + U \qquad M = D$$

$$N = L + U$$

matrix-vec product, roughly Ag

$$D x_{k+1} = (L+U) x_k + b \Rightarrow x_{k+1} = D^{-1}(L+U) x_k + D^{-1} b$$

- The Jacobi method converges if $\boldsymbol{A}$ is strictly row-diagonally-dominant:

$$\| D^{-1}(L+U) \|_1 < 1$$

even when this is the case, Jacobi converges linearly with a constant that $\to x$ as $h \to 0$

Gauss-Seidel Method

- The Jacobi method takes weighted sums of $\boldsymbol{x}^{(k)}$ to produce each entry of $\boldsymbol{x}^{(k+1)}$, while Gauss-Seidel uses the latest available values, i.e. to compute $x_i^{(k+1)}$ it uses a weighted sum of

$$x_1^{(k+1)}, \ldots x_{i-1}^{(k+1)}, x_i^{(k)}, \ldots, x_n^{(k)}.$$

$M = D + L \qquad N = U$

at each step, solve a triangular system of eqs

- Gauss-Seidel provides somewhat better convergence than Jacobi:

$M^{-1}N$ has smaller largest eigval

Successive Over-Relaxation

- The *successive over-relaxation* (SOR) method seeks to improve the spectral radius achieved by Gauss-Seidel, by choosing

$$\boldsymbol{M} = \frac{1}{\omega}\boldsymbol{D} + \boldsymbol{L}, \quad \boldsymbol{N} = \left(\frac{1}{\omega} - 1\right)\boldsymbol{D} - \boldsymbol{U}$$

- The parameter ω in SOR controls the 'step-size' of the iterative method:

$\omega > 1$ over-relaxation

$\omega < 1$ under-relaxation

$\omega = 1$ Gauss-Seidel

Conjugate Gradient

- The solution to $\boldsymbol{Ax} = \boldsymbol{b}$ is a minima of the quadratic optimization problem,

$$\min_{\boldsymbol{x}} ||\boldsymbol{Ax} - \boldsymbol{b}||_2^2$$

$\min_x \frac{1}{2}x^TAx - x^Tb$

critical point

optimality condition

$Ax=b$

need A symmetric

& positive definite

- Conjugate gradient works by picking $\boldsymbol{A}$-orthogonal descent directions

converges in n steps

- The convergence rate of CG is linear with coefficient $\frac{\sqrt{\kappa(\boldsymbol{A})}-1}{\sqrt{\kappa(\boldsymbol{A})}+1}$:

in general, sparse iterative methods are fast for well-conditioned A

Preconditioning

- Preconditioning techniques choose matrix $\boldsymbol{M} \approx \boldsymbol{A}$ that is easy to invert and solve a modified linear system with an equivalent solution to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$,

$$\boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{b}$$

$\kappa(M^{-1}A) < \kappa(A)$

$(M^{-1}(A x_k))$,

- $\boldsymbol{M}$ is chosen to be an effective approximation to $\boldsymbol{A}$ with a simple structure:

pick M based on LU/Cholesky of A where we skip all fill, so $A \approx LU$

pick $M = LU$