Fast Methods for PDE ! IE

Github.com/mitmath/18336 - find everything here!

Gradescope - submit assignments here

Piazza - ask questions here (and announcements)

⇒ Links in 1st Announcement on Canvas

Why solve PDEs ! IEs?

$$\frac{\overset{\text{temp.}}{\partial u}}{\partial t} = \overset{\text{thermal diff.}}{\alpha} \frac{\partial^2 u}{\partial x^2} + \overset{\text{volumetric heat source}}{f(x)}$$

$u(0) = 0$, $u(1) = 1$

Heat flow in a rod

Modeling ⟺ Simulation

⇙ ⇘

Detection/ Inference (inverse problems)

Design (optimization)

Modern applications often involve very large simulations, that require solving complex PDEs many times in an "inner loop."

$\Rightarrow$ Weather / Climate simulations

$\Rightarrow$ Electronic Structure (material design)

$\Rightarrow$ Photonic design

$\Rightarrow$ Fluids / multiphysics

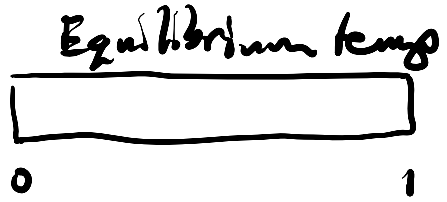
$\Rightarrow$ ... Many more!

Numerical Methods for PDE

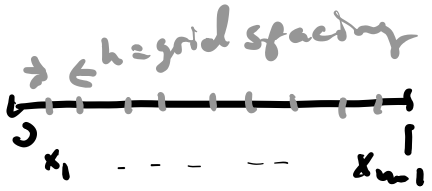
$$\frac{d^2u}{dx^2} = f(x) \qquad (*)$$

Stationary 1D PDE (Laplace)

$u(0) = u(1) = 0$



Step 1: Discretize



"solution vector"

$\underline{u}_n = [u_1, u_2, \ldots, u_{n-1}]$ $\quad$ $\underline{f} = [f_1, \ldots, f_{n-1}]$

matrix $(n-1)\times(n-1)$

PDE becomes $\Rightarrow$ $A_n \underline{u}_n = \underline{f}_n$

Step 2: Solve

numerical linear algebra

$$\underbrace{\begin{bmatrix} \quad & \quad & \quad \\ & & \\ & & \end{bmatrix}}_{A_n}\underbrace{\begin{bmatrix} \\ \\ \\ \end{bmatrix}}_{\underline{u}_n} = \underbrace{\begin{bmatrix} \\ \\ \\ \end{bmatrix}}_{\underline{f}_n}$$

Solve $(n-1)\times(n-1)$ linear sys with NLA routines.

Step 3: Approximate/Analyze

How close is $\underline{u}_n = [u_1, \ldots, u_{n-1})$ to the truth $\underline{u}_n^* = [u(x_1), \ldots, u(x_n)]$?

What is $\|\underline{u}_n - \underline{u}_n^*\|$?

Usually want to understand how

$\|\underline{u}_n - \underline{u}_n^*\|$ scales as $n \to \infty$.

This depends on how we "discretize" the problem in step 1.

How should we think about "cost" of solving this stationary PDE (Laplace Eq.)?

1) How much time to compute to a desired tolerance $\|\underline{u}_n - \underline{u}_n^*\| \le \varepsilon_*$?

2) How many FLOPS, required to compute $\underline{u}_n$ so that $\|\underline{u}_n - \underline{u}_n^*\| \le \varepsilon_*$?

(2) is traditional and often useful, but does not always correlate with (1)!

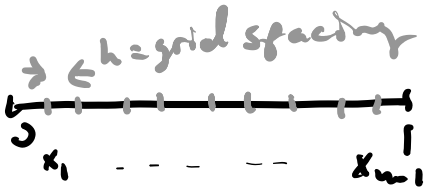
$\Rightarrow$ memory

$\Rightarrow$ communication

$\Rightarrow$ Distributed, asynch, etc.

Finite Difference Approximations

$$\frac{du}{dx} = \lim_{h \to 0} \frac{u(x+h) - u(x)}{h}$$



$$\frac{d^2u}{dx^2} = \lim_{h \to 0} \frac{u(x+h) - 2u(x) + u(x-h)}{h^2}$$

$\Rightarrow$ Pick finite $h > 0$ for "FD" approx

PDE at j^{th} Gridpt:

$$\frac{u(x_{j+1}) - 2u(x_j) + u(x_{j-1})}{h^2} = f_j \qquad \text{for } j=1,\dots,n-1$$

all together ⇒

$$\begin{bmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-1} \end{bmatrix}$$

tridiag! ⇒ $\mathcal{O}(n)$ solve

Gaussian Elim

$$-\begin{bmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{bmatrix} = \begin{bmatrix} 1 & -1 & & \\ & 1 & -1 & \\ & & \ddots & -1 \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}$$

Sparse/banded LU factorization!

What is the accuracy of u_n?

from Taylor series

$$\frac{d^2u}{dx^2} = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} + \mathcal{O}(h^2)$$

↓

term $\leq Ch^2$ for suff. small $h > 0$.

as $h \to 0$ (as $n \to \infty$)

$h \sim 1/n$

Can we improve the scaling with n?

$\Rightarrow$ trade off smoothness!

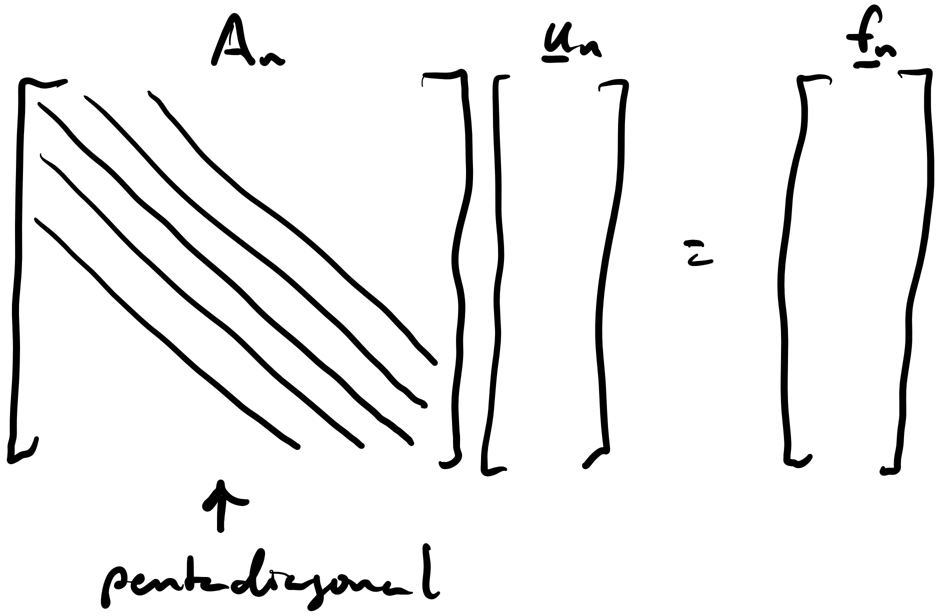
<u>Higher Order FD</u>

$$\frac{d^2u}{dx^2} = \frac{-u(x+2h) + 16u(x+h) - 30u(x) + 16u(x-h) - u(x-2h)}{12h^2}$$

$$+ O(h^4)$$

Error decreases faster with h!

But bandwidth increases



$\Rightarrow$ Trade-off between cost of linear solve and size of linear system required to achieve error tolerance $\varepsilon > 0$?