Today

- BVP
- Solving large (sparse) linear systems
- PDE

Announcements

- Final : content
- Finals week Office Hours
- Grade rows

BVPs

$$\vec{u}'(x) = \vec{f}(\vec{u}(x))$$

$$u : [a, b] \to \mathbb{R}^n$$

$$g(\vec{u}(a), \vec{u}(b)) = \vec{0}$$

Shooting Method

Idea: Want to make use of the fact that we can already solve IVPs.

Problem: Don't know *all* left BCs.

$u'' = f(x)$

$u(4) = 15$ $\qquad$ $\cancel{u(5) = 12}$

$u'(4) = ?$

Demo: Shooting method

What about systems?

Yes, just like cannons

What are some downsides of this method?

Efficiency

Stability?

Failures

What's an alternative approach?

Big system of equations

Finite Difference Method

Idea: Replace u' and u'' with finite differences.

For example: second-order centered

$$
\begin{aligned}
u'(x) &= \frac{u(x+h) - u(x-h)}{2h} + O(h^2) \\
u''(x) &= \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} + O(h^2)
\end{aligned}
$$

Demo: Finite differences

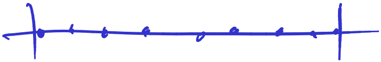
What happens for a nonlinear ODE?

Demo: Sparse matrices

CSR matrix

Collocation Method

$u''(x) + u(x) = 5$



$$(*)\begin{cases} y'(x) = f(y(x)), \\ g(y(a), y(b)) = 0. \end{cases}$$

1. Pick a basis (for example: Chebyshev polynomials)

$$\hat{y}(x) = \sum_{i=1}^{n} \alpha_i T_i(x)$$

   Want $\hat{y}$ to be close to solution y. So: plug into $(*)$.

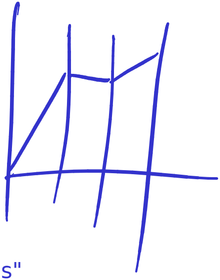
   Problem: $\hat{y}$ won't satisfy the ODE at all points at least.
   We do not have enough unknowns for that.

2. Idea: Pick n points where we would like $(*)$ to be satisfied.
   $\rightarrow$ Get a big (non-)linear system

3. Solve that (LU/Newton)$\rightarrow$ done.

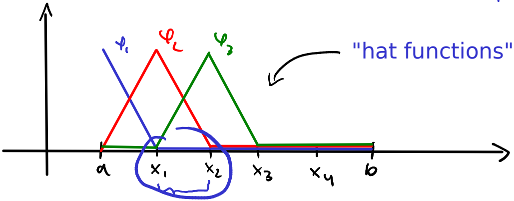
Galerkin/Finite Element Method

$$u''(x) = f(x), \qquad u(a) = u(b) = 0.$$



Problem with collocation: Big dense matrix.

Idea: Use piecewise basis. Maybe it'll be sparse.



What's the problem with that?

no second derivatives

Weak solutions/Weighted Residual Method

Idea: Enforce a 'weaker' version of the ODE.

"test function"

$$u''(x) = f(x)$$

$$\int_a^b u''(x)\,\varphi_i(x)\,dx = \int_a^b f(x)\,\varphi_i(x)\,dx$$

for some n functions $\varphi_1 \dots \varphi_i \dots \varphi_n$

$$\Big[u'(x)\,\varphi_i(x)\Big]_a^b - \int_a^b u'(x)\,\varphi_i'(x)\,dx = \int f(x)\,\varphi_i(x)\,dx$$

"Finite element"

"Galerkin" / "weighted residual" / $\int (u''-f)\,\varphi_i(x) = 0$

Galerkin: Choices in Weak Solutions

Make some choices:

- Solve for $u \in \operatorname{span}\{\text{hat functions } \varphi_i\}$
- Choose $\psi \in W = \operatorname{span}\{\text{hat functions } \varphi_i\}$ with $\psi(a) = \psi(b) = 0$.
  $\rightarrow$ Kills boundary term $[u'(x)\psi(x)]_a^b$.

These choices are called the Galerkin method. Also works with other bases.

Discrete Galerkin

$$u \approx \sum_{i=1}^{n} \alpha_i \varphi_i(x)$$

Assemble a matrix for the Galerkin method.

$$-\int_a^b u'(x)\psi_i'(x)\,dx = \int_a^b f(x)\psi_i(x)\,dx$$

$\psi_i = \varphi_i$

$$-\sum_{j=1}^{n} \alpha_j \underbrace{\int_a^b \varphi_j'(x)\,\varphi_i'(x)\,dx}_{S_{ij}} = \int_a^b f(x)\varphi_i(x)\,dx$$

stiffness

Outline

Advertisement

Remark: Both PDEs and Large Scale Linear Algebra are big topics. Will only scratch the surface here. Want to know more?

- CS555 $\rightarrow$ Numerical Methods for PDEs (spring)
- CS556 $\rightarrow$ Iterative and Multigrid Methods (fall of 20)
- CS554 $\rightarrow$ Parallel Numerical Algorithms

We would love to see you there! :)

- CS598 $\rightarrow$ Fast Algorithms and Int. eq.

Solving Sparse Linear Systems

Solving $A\mathbf{x} = \mathbf{b}$ has been our bread and butter.

Typical approach: Use factorization (like LU or Cholesky)
Why is this problematic?

Idea: Don't factorize, iterate.
Demo: Sparse Matrix Factorizations and "Fill-In"

'Stationary' Iterative Methods

Idea: Invert only part of the matrix in each iteration. Split

$$A = M - N,$$

where M is the part that we are actually inverting. Convergence?

$$\begin{aligned} A\mathbf{x} &= \mathbf{b} \\ M\mathbf{x} &= N\mathbf{x} + \mathbf{b} \\ M\mathbf{x}_{k+1} &= N\mathbf{x}_k + \mathbf{b} \\ \mathbf{x}_{k+1} &= M^{-1}(N\mathbf{x}_k + \mathbf{b}) \end{aligned}$$

- These methods are called *stationary* because they do the same thing in every iteration.
- They carry out fixed point iteration.
  $\to$ Converge if contractive, i.e. $\rho(M^{-1}N) < 1$.
- Choose M so that it's easy to invert.

Choices in Stationary Iterative Methods

What could we choose for M (so that it's easy to invert)?

Name	M	N
Jacobi	D	$-(L+U)$
Gauss-Seidel	$D+L$	$-U$
SOR	$\frac{1}{\omega}D+L$	$\left(\frac{1}{\omega}-1\right)D-U$

where L is the below-diagonal part of A, and U the above-diagonal.

Demo: Stationary Methods

Conjugate Gradient Method

Assume A is symmetric positive definite.

Idea: View solving $A\mathbf{x} = \mathbf{b}$ as an optimization problem.

$$\text{Minimize} \quad \varphi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b} \quad \Leftrightarrow \quad \text{Solve} \quad A\mathbf{x} = \mathbf{b}.$$

Observe $-\nabla\varphi(\mathbf{x}) = \mathbf{b} - A\mathbf{x} = \mathbf{r}$ (residual).

Use an iterative procedure ($\mathbf{s}_k$ is the search direction):

$$\begin{aligned} \mathbf{x}_0 &= \langle\text{starting vector}\rangle \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_k \mathbf{s}_k, \end{aligned}$$

CG: Choosing the Step Size

What should we choose for α_k (assuming we know $\mathbf{s}_k$)?

$$0 \overset{!}{=} \frac{\partial}{\partial \alpha} \varphi(x_k + \alpha_k s_k)$$

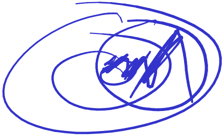
$$= \nabla \varphi'(x_{k+1})^T s_k = r_{k+1} \cdot s_k$$

$$r_{k+1} = r_k + \alpha_k A s_k$$

$$0 \overset{!}{=} s_k^T r_{k+1} = s_k^T r_k + \alpha_k s_k^T A s_k$$

$$\alpha_k = \frac{s_k^T r_k}{s_k^T A s_k} = -\frac{s_k^T A e_k}{s_k^T A s_k}$$

CG: Choosing the Search Direction



What should we choose for $\mathbf{s}_k$?

Steepest desc: bad idea

Better idea: $(\vec{x},\vec{y})_A = x^T A y$ is an IP $\Leftrightarrow$ A spd

$$s_i^T A s_j = 0 \quad \text{if } i \neq j$$

$$e_0 = x_0 - x^* = \underline{\sum_i \delta_i s_i}$$

CG: Further Development

$$s_k^T A e_0 = \sum_i \delta_i s_k^T A s_i = \delta_k s_k^T A s_k$$

$$\delta_k = \frac{s_k^T A e_0}{s_k^T A s_k} = \frac{\overbrace{s_k^T A \left(e_0 + \sum_{i=1}^{k-1} \alpha_i s_i\right)}}{s_k^T A s_k} = \frac{s_k^T A e_k}{s_k^T A s_k} = \alpha_k$$

Use Krylov to pick search direction

↳ orthogonalization stops after 3 (Lanczos), so it's cheap

Demo: Conjugate Gradient Method

Introduction

Notation:

$$\frac{\partial}{\partial x} u \quad = \quad \partial_x u \quad = \quad u_x .$$

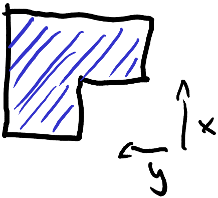
A *PDE* (*partial differential equation*) is an equation with multiple partial derivatives:

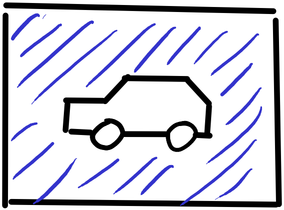
$$u_{xx} + u_{yy} = 0$$

Here: solution is a function $u(x, y)$ of two variables.

Examples: Wave propagation, fluid flow, heat diffusion

- Typical: Solve on domain with complicated geometry.





Initial and Boundary Conditions

- Sometimes one variable is time-like.
  What makes a variable time-like?
  - Causality
  - No geometry

Have:

- PDE
- Boundary conditions
- Initial conditions (in t)