Class web page

Office hrs: ~~12:30~~ ~~-~~ 1:30pm (4318 Siebel)

today: in this room, right after class

https://bit.ly/fastalg-s24

contains:

- Class outline ✓
- Notes ✓
- Demos ✓
- Assignments ✓
- Discussion forum ✓
- Grading ✓
- Video ✓

$$\int k(x,y)\,\sigma(y)\,dy$$

Why study this at all?

- Finite difference/element methods are inherently
  - ill-conditioned
  - tricky to get high accuracy with
- Build up a toolset that does *not* have these flaws
- Plus: An interesting/different analytical and computational point of view
  - If you're not going to use it to solve PDEs, it (or the ideas behind it) will still help you gain insight.

FD/FEM: Issues

Idea of these methods:

1. Take differential equations
2. Discretize derivatives
3. Make linear system
4. Solve

So what's wrong with doing that?

Discretizing Derivatives: Issues?

- bad conditioning

$\kappa(A) = \|A\| \|A^{-1}\|$

$\|A\|_\infty = \sup_{f \in S} \frac{\|Af\|_\infty}{\|f\|_\infty}$

$A = \partial_x$

$(e^{i\alpha x})' = (i\alpha) e^{i\alpha x} \quad \alpha \in \mathbb{R}$

$\|e^{i\alpha x}\|_{L^\infty(0,2\pi)} = 1$

$\|i\alpha e^{i\alpha x}\|_\infty = |\alpha|$

$\|\partial_x f\|_\infty \le \underset{\|\partial_x\|_\infty}{?} \, \|f\|_\infty$

$\|\partial_x\|_\infty = \infty$

discrete ?

Discretizing Derivatives: Issues?

Result: The better we discretize (the more points we use), the worse the condition number gets.

Demo: Conditioning of Derivative Matrices

To be fair: Multigrid works around that (by judiciously using fewer points!)
But there's another issue that's not fixable.

As $h \to 0$

$$\frac{f(x+h) - f(x-h)}{2h}$$

cancellation

Q: Are these problems real?

Poisson $\Delta u = f$ ← need MG or other preconditioner

So this class is about starting fresh with methods that (rigorously!) don't have these flaws!

Bonus Advertising Goodie

Both multigrid and fast/IE schemes ultimately are $O(N)$ in the number of degrees of freedom N.

Open Source <3

These notes (and the accompanying demos) are open-source!

Bug reports and pull requests welcome:
https://github.com/inducer/fast-alg-ie-notes

Copyright (C) 2013 – 24 Andreas Kloeckner

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Outline

Matvec: A Slow Algorithm

Matrix-vector multiplication: our first 'slow' algorithm.
$O(N^2)$ complexity.

$$\beta_i = \sum_{j=1}^{N} A_{ij}\alpha_j$$

Assume A dense.

columns : sources
rows : targets

Matrices and Point Interactions

$$A_{ij} = G(x_i, y_j)$$

Does that actually change anything?

$$\psi(x_i) = \sum G(x_i, y_j)\,\varphi(y_j)$$

x_i y_j points ($\in \mathbb{R}^2$? $\in \mathbb{R}^3$?)

- x_i "targets" / "observation points"
- y_j "sources"
- G kernel

"where you look"

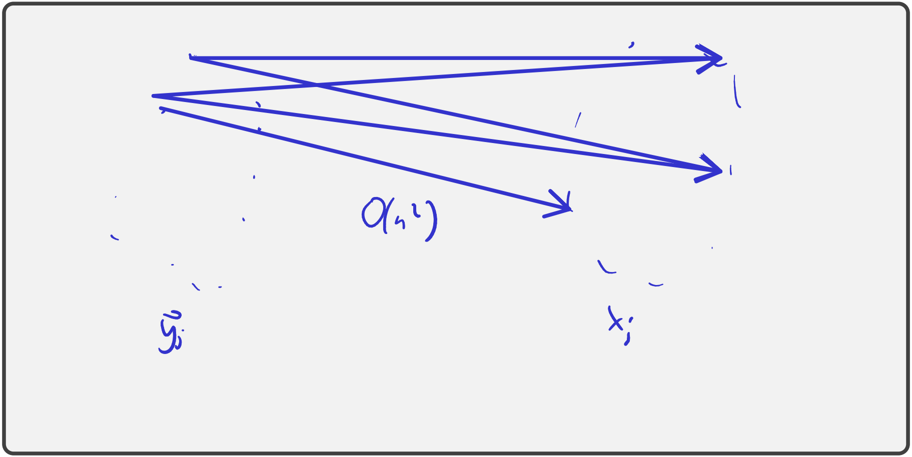
"electrons"

$$G(x,y) = \frac{1}{\|x-y\|_2}$$

Matrices and Point Interactions

$$A_{ij} = G(x_i, y_j)$$

Graphically, too:



Matrices and point Interactions

$$\psi(x_i) = \sum_{j=1}^{N} G(x_i, y_j)\varphi(y_j)$$

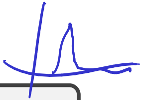
This *feels* different.

make x continuous: $\psi(x) = \sum_{j=1}^{N} G(x, y_j)\,\varphi(y_j)$ ← mat inf. tall

make y continuous: $\psi(x) = \int G(x,y)\,\varphi(y)\,dy$ ← mat. inf. wide

Q: Are there enough matrices that come from globally defined G to make this worth studying?

Point Interaction Matrices: Examples (I)



- Interpolation

$$\Psi(x) = \sum_{j=1}^{n} \underbrace{l_j(x)}\,\varphi(y_j) \qquad L$$

- Differentiation

$$\Psi(x) = \sum_{j=1}^{n} \underbrace{l_j'(x)}\,\varphi(y_j)$$

- Integration

$$\Psi(x) = \Psi(x) = \sum_{j=1}^{n} \int_0^x l_j(\xi)\,d\xi\,(y_j)$$

Point Interaction Matrices: Examples (II)

- Potential evaluation (3D)

$$U(x) = \frac{q_{el}}{4\pi} \cdot \frac{1}{|x|} \leftarrow \text{electrostatics}$$

$$u(x) = \sum_{j=1}^{N} \frac{q_{el}}{4\pi} \frac{1}{|x-y_j|} \varphi(y_j)$$

waves: $$U(x) = C \cdot \frac{e^{ik|x|}}{|x|}$$

Point Interaction Matrices: Examples (III)

- Convolutions

$$\psi(x) = \sum_{j=1}^{N} G(x - y_j)\, \varphi(y_j)$$

sub-$O(n^2)$ algorithm? FFT

↳ grid periodic equispaced

"Butterfly transform"

So yes, there are indeed lots of these things.

Integral Operators

Why did we go through the trouble of rephrasing matvecs as

$$\psi(x_i) = \sum_{j=1}^{N} G(x_i, y_j)\varphi(y_j)?$$

Cheaper Matvecs

$$\psi(x_i) = \sum_{j=1}^{N} G(x_i, y_j)\varphi(y_j)$$

So what can we do to make evaluating this cheaper?

- sparse (FD / FEM)
- special structure
  - Toeplitz
  - circulant

  } FFT-based trickery
- low rank

Fast Dense Matvecs

Consider

$$A_{ij} = u_i v_j,$$

outer product

$A = \vec{u}\vec{v}^T$

let $\mathrm{u} = (u_i)$ and $\mathrm{v} = (v_j)$.

Can we compute $A\mathrm{x}$ quickly? (for a vector x)

$$A\vec{x} = \vec{u}\vec{v}^T\vec{x}$$

$$= (\vec{u}\vec{v}^T)\vec{x} \quad ① \quad \leftarrow O(n^2)$$

$$= \vec{u}(\underbrace{\vec{v}^T\vec{x}}_{O(n)}) \quad ② \quad \leftarrow O(n) \checkmark$$

Fast Dense Matvecs (II)

$$A = \mathbf{u}_1 \mathbf{v}_1^T + \cdots + \mathbf{u}_K \mathbf{v}_K^T$$

$A \in \mathbb{R}^{N \times N}$

Does this generalize? What is K here?

$\operatorname{rank}(A) = K$

Cost: $O(NK)$

Low-Rank Point Interaction Matrices

Usable with low-rank complexity reduction?

$$\psi(x_i) = \sum_{j=1}^{N} G(x_i, y_j)\varphi(y_j)$$

$\vec{a}_1$ $\vec{a}_2$

$(\vec{a}_1 \ \vec{a}_2)$

$$\psi(x_i) = \sum_{j=1}^{N} \underbrace{G_1(x_i)\, G_2(y_j)}_{G(x_i, y_j)}\, \varphi(y_j)$$

"separation of variables"

$$G(x,y) = \frac{1}{\|x-y\|_2}$$ does not look like this.

rank hard to define numerically because of rounding

Numerical Rank

What would a *numerical* generalization of 'rank' look like?

Eckart-Young-Mirsky Theorem

Theorem (Eckart-Young-Mirsky)

SVD $A = U\Sigma V^T$. *If* $k < r = \operatorname{rank}(A)$ *and*

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T,$$

then

$$\min_{\operatorname{rank}(B)=k} |A - B|_2 = |A - A_k|_2 = \sigma_{k+1}.$$

Q: What's that error in the Frobenius norm?

So in principle that's good news:

- We can find the numerical rank.
- We can also find a factorization that reveals that rank (!)

Demo: Rank of a Potential Evaluation Matrix (Attempt 2)

Constructing a tool

There is still a slight downside, though.