Today:

- matvec $\rightarrow O(n^2)$

↳ integral operators

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.

Matrices and Point Interactions

$\in \mathbb{R}^3$

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

$\in \mathbb{R}^3$

Does that actually change anything?

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

$x_i \rightarrow$ Targets / Observation points

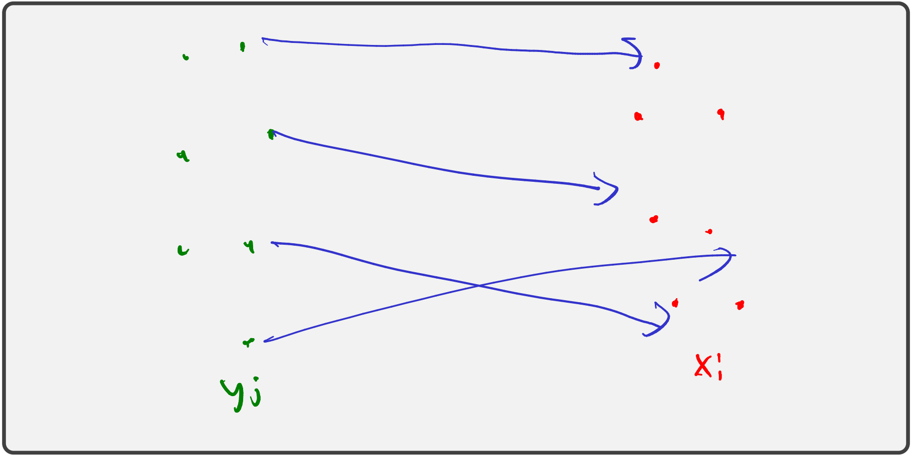
$y_j \rightarrow$ Sources

$G(x, y) \rightarrow$ Kernel

Matrices and Point Interactions

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

Graphically, too:



Matrices and point Interactions

infinitely tall matrix

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

This *feels* different.

$$\psi(x) = \int_{\Omega} G(x, y)\, \varphi(y)$$

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

Point Interaction Matrices: Examples (I)

Interpolation: $\tilde{f}(x) = \sum_{j=1}^{N} \underbrace{l_j(x)}\, \varphi(y_j)$

Interpolation error: $\varphi(x) = f(x) - \sum_{j=1}^{N} \underbrace{l_j(x)}\, \varphi(y_j)$

Num diff.:

$\tilde{f}'(x) = \sum_{j=1}^{N} \underbrace{l_j'(x)}\, \varphi(y_j)$

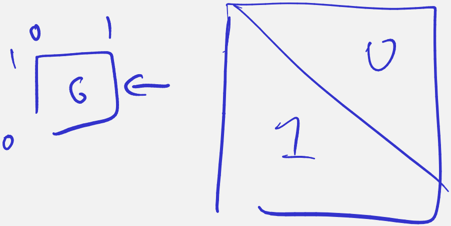
Identity:

$$\varphi(x) = \int_{\mathbb{R}} \underbrace{\delta(x-y)} \varphi(y)\, dy$$

$$z_i = \sum A_{ij} \quad z_j$$

Point Interaction Matrices: Examples (II)

$$x \in [0,1] \mapsto \int_0^x f(y)\,dy = \int_0^1 \underbrace{1}\, f(y)\,dy$$



Point Interaction Matrices: Examples (III)

Convolutions

↳ $\chi(x) = \int G(x-y)\,\varphi(y)\,dy$

↳ circulant matrices

$G(x,y) = C \cdot \frac{1}{|x-y|} \leftarrow \underset{(3D)}{\Delta u = 0}$ fund. solution

$C \cdot \log(|x-y|) \leftarrow$ 2D

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
- circulant / Toeplitz / convolutions
- low rank

Fast Dense Matvecs

Consider

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

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

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

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

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

Fast Dense Matvecs (II)

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

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

Does this generalize? What is K here?

Cost: $O(Nk)$

Low-Rank Point Interaction Matrices

Usable with low-rank complexity reduction?

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

$A = uv^T$

$G(x, y) = u(x) \cdot v(y)$

simple "compact operator"

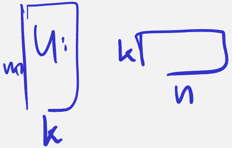
"compact" $\Leftrightarrow$ "approximately finite-dim. range"

Numerical Rank

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

A has rank k if $\quad A \in \mathbb{R}^{m\times n}$

$$A = UV$$



$$\operatorname{numrank}(A, \varepsilon) = \min\{k : \exists\, U \in \mathbb{R}^{m\times k}, V \in \mathbb{R}^{k\times n},\ \|A - UV\|_2 \le \varepsilon\}$$

Eckart-Young-Mirsky Theorem

$\|A\|_2 = \max |\sigma_i|$ / $\|A\|_F = \sqrt{\sum_{i,j} A_{i,j}^2}$

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

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

$\|A\|_F = \|\Sigma\|_F = \sqrt{\sum \sigma_i^2}$

F $\quad F = \sqrt{\sum_{i=k+1}^{\min(m,n)} \sigma_i^2}$

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.

build: $O(N^2)$

factorize: $O(N^3)$

matvec: $O(Nk)$

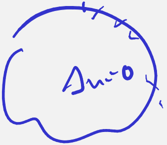
Representation

$Ax=b$ $\qquad M \approx A^{-1}$

What does all this have to do with (right-)preconditioning?

(MA) ← left preconditioning

(AM) ← right preconditioner



Representation (in context)

Outline

Rephrasing Low-Rank Approximations

SVD answers low-rank-approximation ('LRA') question. But: too expensive. First, rephrase the LRA problem:

$$A = UV^T \leftarrow \text{factorization form}$$

$$A \approx \underset{k}{\underline{Q}}\,\underbrace{Q^T A}_{V} \leftarrow \text{projection}$$

$$\underset{Q}{N\,\Big[\ \Big]}$$

Using LRA bases

$A \approx QQ^T A$

If we have an LRA basis Q, can we compute an SVD?

		Complexity
1	$B = Q^T A$	$N^2 k$
2	$B = \tilde{U} \Sigma V^T$	$k^2 N$
3	$A \approx QB = \underbrace{Q \tilde{U}}_{U} \Sigma V^T$	$k^2 N$

Finding an LRA basis

How would we *find* an LRA basis?

Goal: Find Q columns

$$\|A - QQ^T A\|_2 \le \varepsilon$$

- Fixed-rank approximation
- Adaptive LRA

Idea: SVD

Idea 2: Randomized algorithm for rangefinding

Giving up optimality

What problem should we actually solve then?

$$\|A - QQ^T A\|_2 = \min_{\mathrm{rank}(X) \le k} \|A - X\|_2 = \sigma_{k+1}$$

$$\|A - QQ^T A\|_2 = \min_{\mathrm{rank}(X) \le k+p} \|A - X\|_2 = \cancel{\sigma_{k+1}}\ \varepsilon$$

Recap: The Power Method

How did the power method work again?