October 15, 2026

Announcements

- Project proposal

Goals

- Direct solve
- Butterfly transform
- PDE solving
  - ⇝ layer potentials

Review

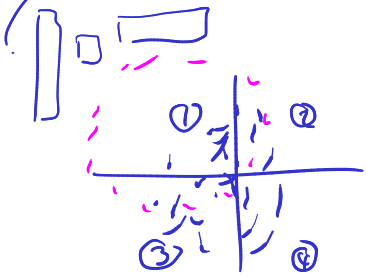
Block-Separable Matrices

A *block-separable matrix* looks like this:

$$A = \begin{bmatrix} D_1 & P_1\tilde{A}_{12}\Pi_2 & P_1\tilde{A}_{13}\Pi_3 & P_1\tilde{A}_{14}\Pi_4 \\ P_2\tilde{A}_{21}\Pi_1 & D_2 & P_2\tilde{A}_{23}\Pi_3 & P_2\tilde{A}_{24}\Pi_4 \\ P_3\tilde{A}_{31}\Pi_1 & P_3\tilde{A}_{32}\Pi_2 & D_3 & P_3\tilde{A}_{34}\Pi_4 \\ P_4\tilde{A}_{41}\Pi_1 & P_4\tilde{A}_{42}\Pi_2 & P_4\tilde{A}_{43}\Pi_3 & D_4 \end{bmatrix}$$

Here:

- $\tilde{A}_{ij}$ smaller than A_{ij}
- D_i has full rank (not necessarily diagonal)
- P_i shared for entire row
- Π_i shared for entire column



Block-Separable Matrix: Questions

Q: Why is it called that?

Q: How expensive is a matvec?

$N^{3/2}$ similar to single-level HSS

Q: How about a solve?

BSS Solve (I)

Separate out 'coarse' unknowns. Use the following notation:

$$B = \begin{bmatrix} 0 & P_1\tilde{A}_{12} & P_1\tilde{A}_{13} & P_1\tilde{A}_{14} \\ P_2\tilde{A}_{21} & 0 & P_2\tilde{A}_{23} & P_2\tilde{A}_{24} \\ P_3\tilde{A}_{31} & P_3\tilde{A}_{32} & 0 & P_3\tilde{A}_{34} \\ P_4\tilde{A}_{41} & P_4\tilde{A}_{42} & P_4\tilde{A}_{43} & 0 \end{bmatrix}$$

and

$$D = \begin{bmatrix} D_1 & & & \\ & D_2 & & \\ & & D_3 & \\ & & & D_4 \end{bmatrix}, \quad \Pi = \begin{bmatrix} \Pi_1 & & & \\ & \Pi_2 & & \\ & & \Pi_3 & \\ & & & \Pi_4 \end{bmatrix}.$$

$A = D + B\Pi$

$$\begin{bmatrix} D & B \\ -\Pi & \mathrm{Id} \end{bmatrix} \begin{bmatrix} x \\ \tilde{x} \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} \rightsquigarrow \begin{array}{l} Dx + B\tilde{x} = b \\ \Pi x = \tilde{x} \end{array}$$

BSS Solve (II)

Q: What are the matrix sizes? The vector lengths of $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$?

Now work towards doing *just* a 'coarse' solve on $\tilde{\boldsymbol{x}}$, using the Schur complement:

$$\begin{bmatrix} D & B \\ 0 & Id + \Pi D^{-1} B \end{bmatrix} \begin{bmatrix} \cdot \\ \tilde{x} \end{bmatrix} = \begin{bmatrix} \cdot \\ \Pi D^{-1} \zeta \end{bmatrix}$$

BSS Solve (III)

Focus in on the second row:

$$(\mathrm{Id} + \Pi D^{-1} B)\widetilde{\boldsymbol{x}} = \Pi D^{-1} \boldsymbol{b}$$

Every non-zero (i.e. off-diagonal) entry in $\Pi D^{-1} B$ looks like

$$\pi_i D_i^{-1} P_i \tilde{A}_{ij}$$

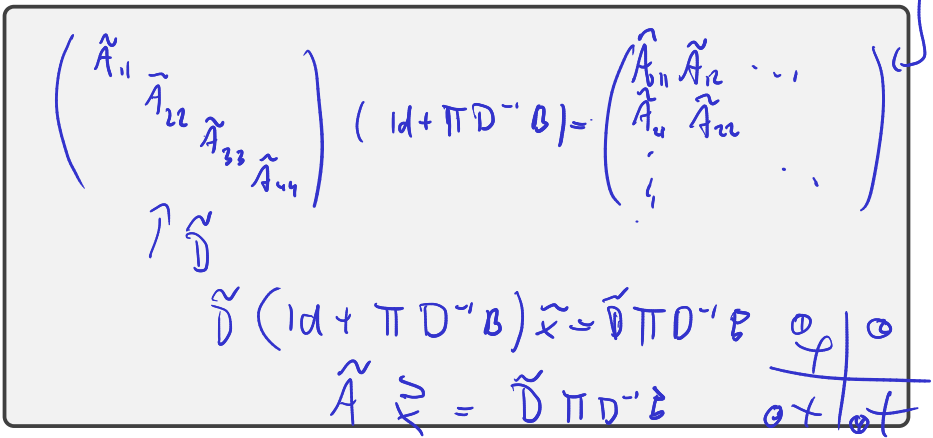
Define a diagonal entry:

$$\tilde{A}_{ii} = (\pi_i D_i^{-1} P_i)^{-1}$$

$$P_i^{-1} D_i \pi_i^{-1}$$

BSS Solve (IV)

Next, left-multiply $(\mathrm{Id} + \Pi D^{-1} B)$ by block-diag$_i(\tilde{A}_{ii})$:



Cost?

Hierarchically Block-Separable

To get to $O(N)$, realize we can *recursively*

- group skeletons
- eliminate more variables.

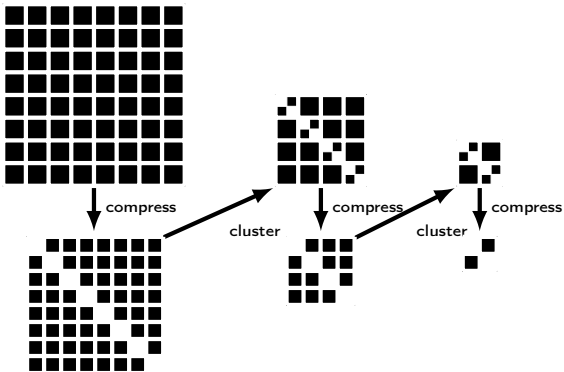
Where does this process start?



Demo: Skeletonization using Proxies (Hierarchical)

Hierarchically Block-Separable

In order to get $O(N)$ complexity, could we apply this procedure recursively?



(Figure following G. Martinsson, drawn by A. Fikl)

Hierarchically Block-Separable

- Using this hierarchical grouping gives us *Hierarchically Block-Separable* (*HBS*) matrices.
- If you have heard the word $\mathcal{H}$-*matrix* and $\mathcal{H}^2$-*matrix*, the ideas are very similar. Differences:
  - $\mathcal{H}$-family matrices don't typically use the ID (instead often use *Adaptive Cross Approximation* or *ACA*)
  - $\mathcal{H}^2$ does target clustering (like FMM), $\mathcal{H}$ does not (like Barnes-Hut)

Outline

Introduction

Dense Matrices and Computation

Tools for Low-Rank Linear Algebra

Rank and Smoothness

Near and Far: Separating out High-Rank Interactions
- Barnes-Hut
- Fast Mutipole
- Ewald Summation and DMK
- Direct Solvers
- The Butterfly Factorization

Outlook: Building a Fast PDE Solver

Going Infinite: Integral Operators and Functional Analysis

Singular Integrals and Potential Theory

Boundary Value Problems

Back from Infinity: Discretization

Computing Integrals: Approaches to Quadrature

Going General: More PDEs

Recap: Fast Fourier Transform

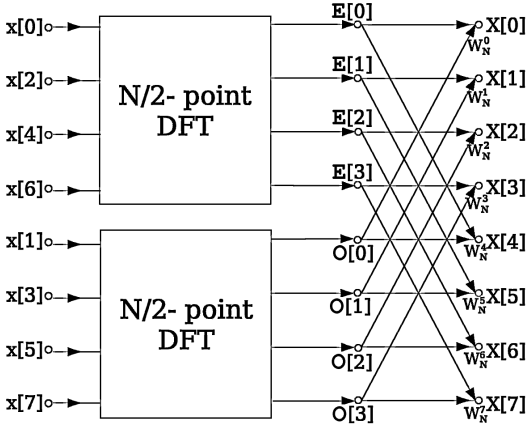
The *Discrete Fourier Transform (DFT)* is given by:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nk} \quad (k = 0, \ldots, N-1)$$

The foundation of the *Fast Fourier Transform (FFT)* is the factorization:

$$X_k = \underbrace{\sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N/2} mk}}_{\text{DFT of even-indexed part of } x_n} + e^{-\frac{2\pi i}{N} k} \underbrace{\sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N/2} mk}}_{\text{DFT of odd-indexed part of } x_n} .$$

FFT: Data Flow



(Figure credit: Wikipedia)
Perhaps a little bit like a butterfly?

Fourier Transforms: A Different View

Claim:

> *The [numerical] rank of the normalized Fourier transform with kernel $e^{i\gamma xt}$ is bounded by a constant times γ, at any fixed precision ϵ.*

(i.e. rank is proportional to the area of the rectangle swept out by x and t) [O'Neil et al. '10]

Demo: Butterfly Factorization (Part I)

Recompression: Making use of Area-Bounded Rank

How do rectangular submatrices get expressed so as to reveal their constant rank?

