

SuperFast Sparse Solvers

Shivkumar Chandrasekaran

ECE Dept., University of California, Santa Barbara

Patrick Dewilde

TU Delft

Ming Gu

Math. Dept., University of California, Berkeley

Bill Lyons Tim Pals

ECE Dept., University of California, Santa Barbara

Problem

- Differential and integral equations are a popular modeling tool.
- Mere mortals need numerical solutions.
- Current algorithms on foreseeable computers run out of **memory** (internal and external) and time, for yesterdays problems.

Problem

- Differential and integral equations are a popular modeling tool.
- Mere mortals need numerical solutions.
- Current algorithms on foreseeable computers run out of **memory** (internal and external) and time, for yesterdays problems.

Solution(s)

- Throw **super-computers** at the problem.
- Find **practical** linear time, linear space solvers.

The Basic Problem

$$Ax = b$$

- A is
 - sparse (differential equations)
 - dense (integro-differential equations)
- A^{-1} is **dense**.

The Basic Problem

$$Ax = b$$

- A is
 - sparse (differential equations)
 - dense (integro-differential equations)
- A^{-1} is **dense**.

Past Hopes

- A has **sparse LU factorization**.
- There exists a “**sparse**” M and a **short** polynomial p such $p(MA) \approx (MA)^{-1}$.

$$MAx = Mb$$

The Basic Problem

$$Ax = b$$

- A is
 - sparse (differential equations)
 - dense (integro-differential equations)
- A^{-1} is **dense**.

Past Hopes

- A has **sparse LU factorization**.
- There exists a “**sparse**” M and a **short** polynomial p such $p(MA) \approx (MA)^{-1}$.

$$MAx = Mb$$

Hopes Dashed

- For 3D problems, A does not have a **sparse factorization**.
- “**Sparse**” M and **short** p 's require live brains.

Green's Function

The way out

- A^{-1} corresponds to the Green's function.
- Green's function has regularity properties.
- What about A^{-1} ?

Green's Function

The way out

- A^{-1} corresponds to the Green's function.
- Green's function has regularity properties.
- What about A^{-1} ?

Many attempts

- Hackbusch's H -matrices. Directly capture low-rank (smoothness) in off-diagonal blocks of A^{-1} .
- ILU pre-conditioners. Capture **rapid decay** in factorization.

Green's Function

The way out

- A^{-1} corresponds to the Green's function.
- Green's function has regularity properties.
- What about A^{-1} ?

Many attempts

- Hackbusch's H -matrices. Directly capture low-rank (smoothness) in off-diagonal blocks of A^{-1} .
- ILU pre-conditioners. Capture **rapid decay** in factorization.

Program

- Look at analysis-based FMM low-rank structure.
- Capture and manipulate it using algebraic ideas from systems theory (Dewilde, et. al.).
- Relate it to sparse matrix algorithms.

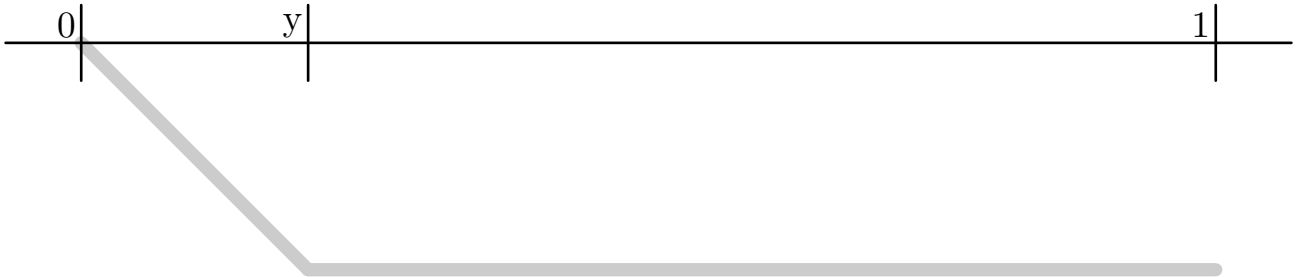
Popular Myth

Green's functions don't **have** to decay.

$$\frac{d^2}{dx^2}G(x, y) = \delta(x - y), \quad 0 \leq x, y \leq 1$$

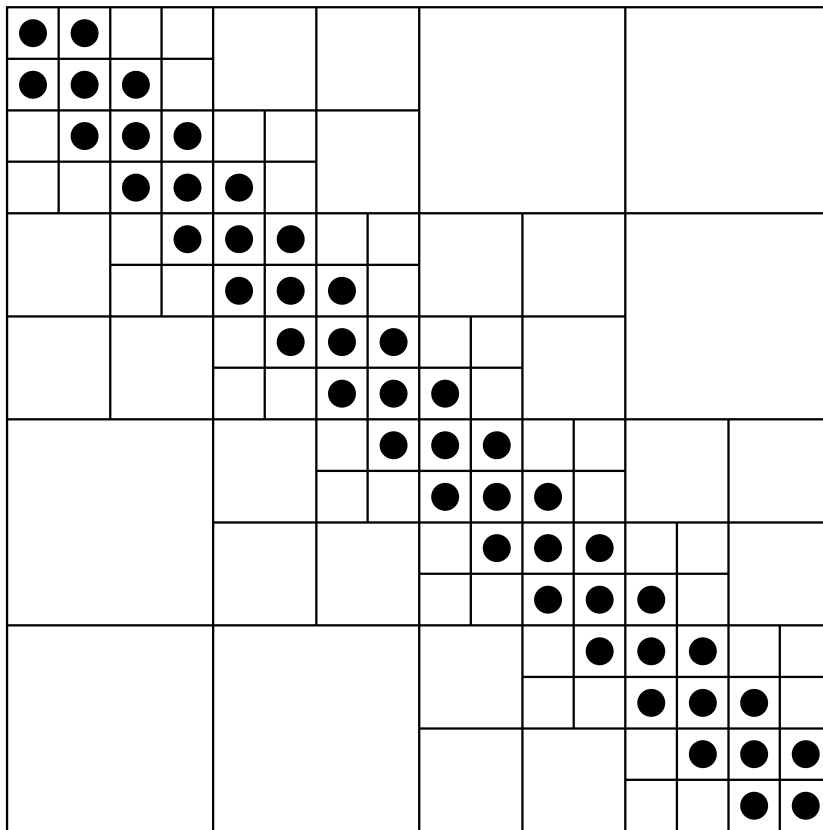
$$G(0, y) = 0$$

$$\left. \frac{d}{dx}G(x, y) \right|_{x=1} = 0$$



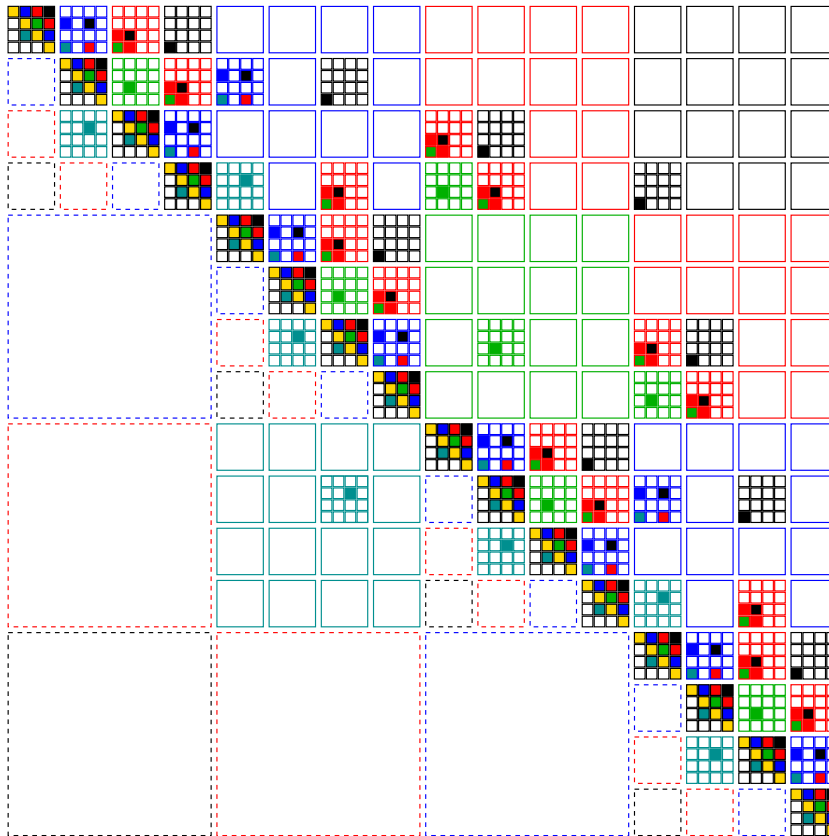
1D FMM Structure

$A_{i,j} = \log \|x_i - x_j\|$, $x_i \in \mathbb{R}$. Bulleted sub-matrices are full-rank. Others have **bounded** rank.



2D FMM Structure

$A_{i,j} = \|z_i - z_j\|^\alpha$, $z_i \in \mathbb{R}^2$. Empty sub-matrices have **bounded** rank. Colored sub-matrices are full-rank.



FMM Work

- FMM literature concentrates on finding optimal low-rank representations for known Green's functions.
- Poincare-Steklov, interface, Schur complement operators have same, but **unknown**, structure.
- We can now compute these FMM representations rapidly on the fly.
- (Implies fast linear-time solvers.)

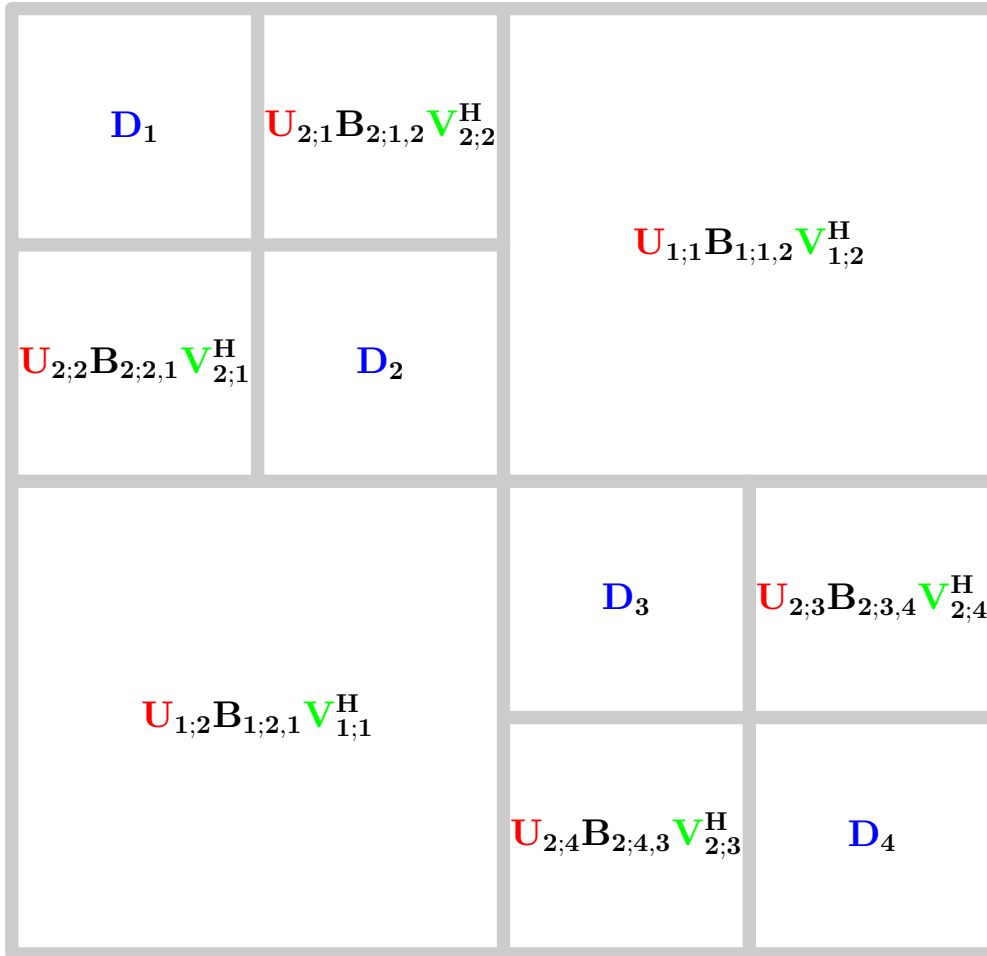
Detailed Program

- **Goal:** develop fast direct solvers for FMM structures.
- **How?**
 - FMM recursions are linear on tree.
 - Invert recursions on the tree using **direct sparse solvers!**
 - Complexities:
 - 1D $46 n p^2$
 - 2D $98 n^{1.5} p^3$
 - 3D $58 n^2 p^3$
 - Matches the complexities for the sparse spine.
 - Sufficient to produce linear-time direct sparse solvers for 3D FEM and FD matrices.
- Can we find linear-time direct solvers for FMM structures? **Yes!** See above!!

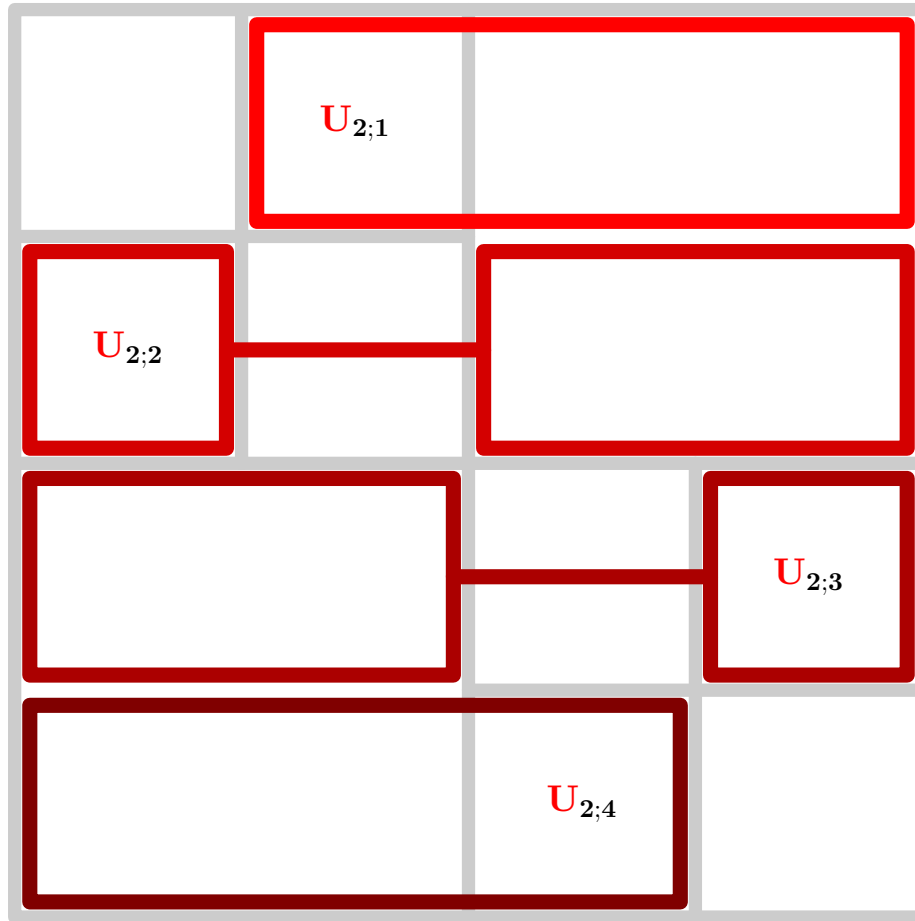
0.5D FMM Structure

- Sufficient to consider the 0.5D FMM structure!
- Hierarchically semi-separable representation (HSS).
- Generalization of the algebraic representations used in time-varying systems theory (Dewilde, van der Veen, Gohberg, Eidelman).
- HSS is closed under $+$, $-$, \times , LU , QR , ULV , pseudo-inversion.
- Formal systems theory for producing fast matrix algorithms.

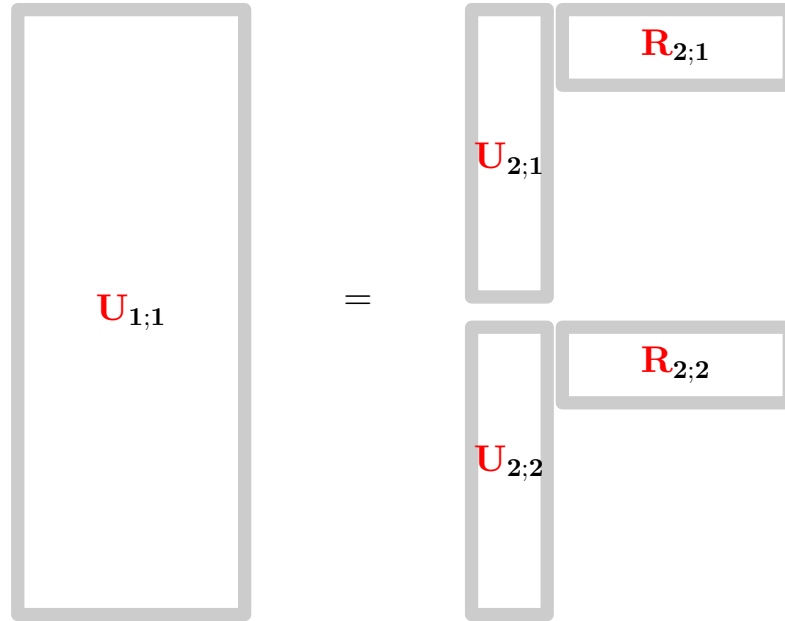
HSS



Column Bases



Column Translation Operators

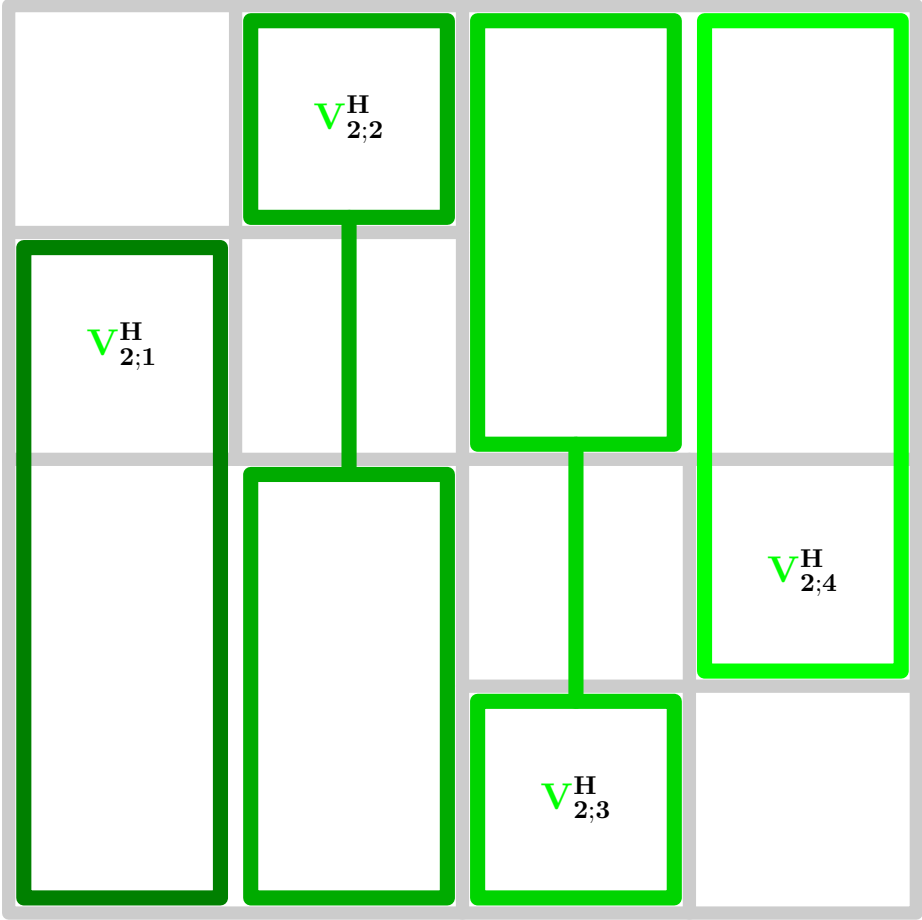


Similarly

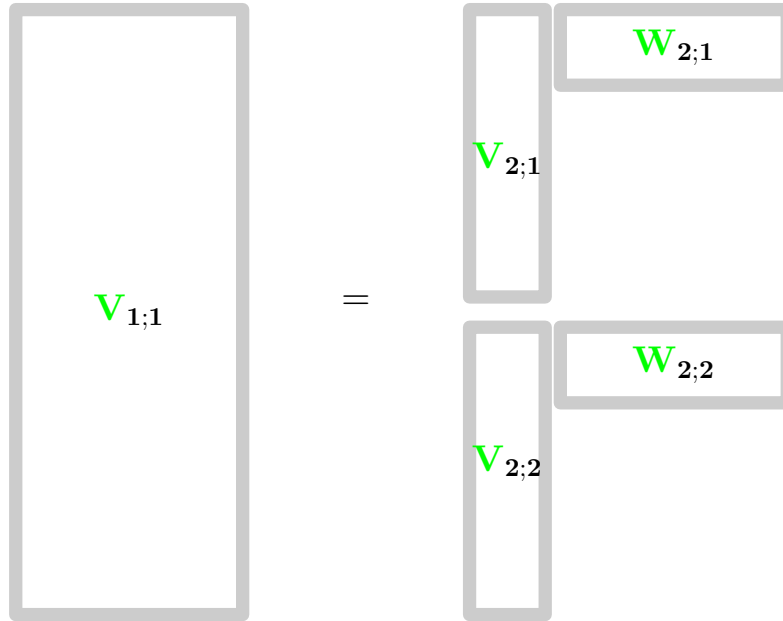
$$U_{1;2} = \begin{pmatrix} U_{2;3} R_{2;3} \\ U_{2;4} R_{2;4} \end{pmatrix}$$

- $U_{1;*}$ not needed and not stored.
- $R_{2;*}$ are smaller and stored instead.

Row Bases



Row Translation Operators

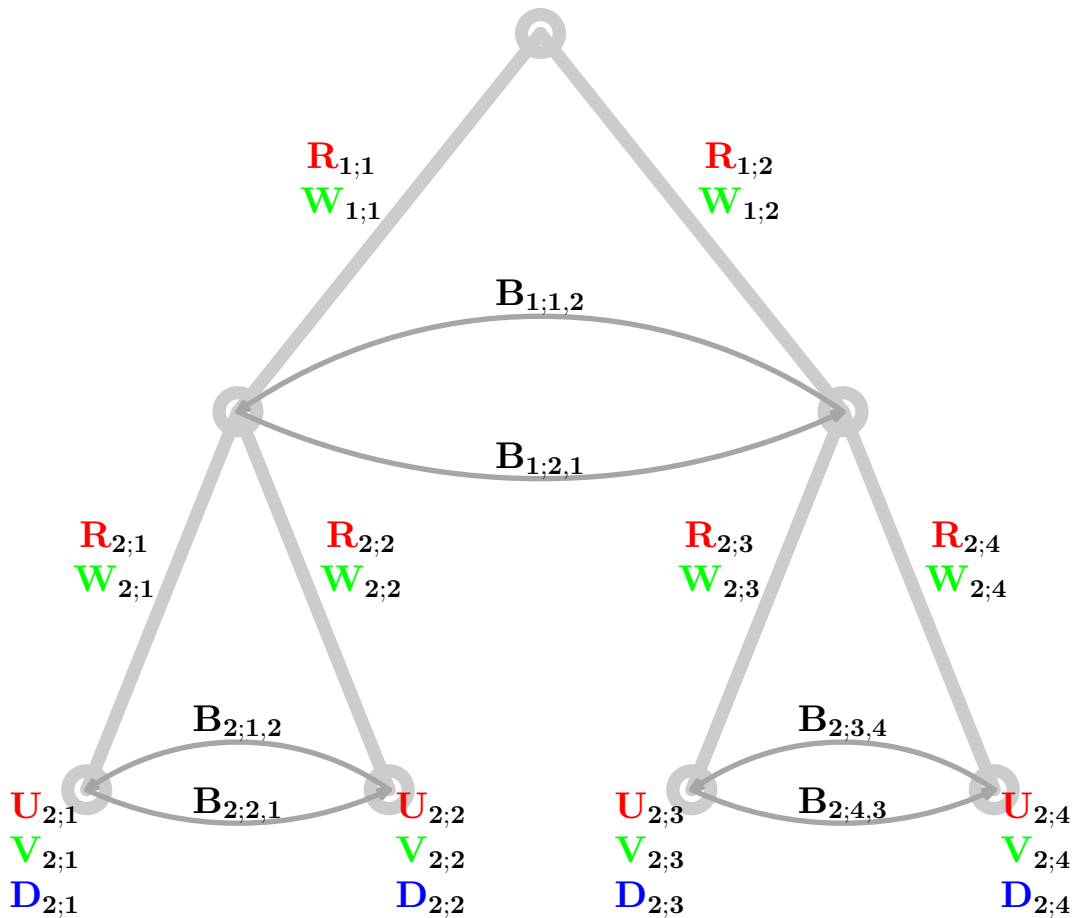


Similarly

$$V_{1;2} = \begin{pmatrix} V_{2;3} & W_{2;3} \\ V_{2;4} & W_{2;4} \end{pmatrix}$$

- $V_{1;*}$ not needed and not stored.
- $W_{2;*}$ are smaller and stored instead.

Binary Tree Representation



FMM (One Level)

$Ax = b$. Consider first level

$$\begin{pmatrix} \mathbf{D}_{1;1} & \mathbf{U}_{1;1}\mathbf{B}_{1;1,2}\mathbf{V}_{1;2}^H \\ \mathbf{U}_{1;2}\mathbf{B}_{1;2,1}\mathbf{V}_{1;1}^H & \mathbf{D}_{1;2} \end{pmatrix} \begin{pmatrix} x_{1;1} \\ x_{1;2} \end{pmatrix} = \begin{pmatrix} b_{1;1} \\ b_{1;2} \end{pmatrix}$$

Signal flow graph representation

$$\begin{matrix} & x_{1;1} & & x_{1;2} \\ & \downarrow & & \downarrow \\ b_{1;1} \leftarrow & \begin{pmatrix} \mathbf{D}_{1;1} & \mathbf{U}_{1;1}\mathbf{B}_{1;1,2}\mathbf{V}_{1;2}^H \\ \mathbf{U}_{1;2}\mathbf{B}_{1;2,1}\mathbf{V}_{1;1}^H & \mathbf{D}_{1;2} \end{pmatrix} & & \end{matrix}$$

Key intermediates

$$g_{1;i} = \mathbf{V}_{1;i}^H x_{1;i}$$

$$f_{1;i} = \mathbf{B}_{1;i,j} g_{1;j}$$

$$b_{1;i} = \mathbf{U}_{1;i} f_{1;i} + \mathbf{D}_{1;i} x_{1;i}$$

FMM (Two Level)

$$\begin{array}{l}
 b_{2;1} \leftarrow \\
 b_{2;2} \leftarrow \\
 b_{2;3} \leftarrow \\
 b_{2;4} \leftarrow
 \end{array}
 \left(
 \begin{array}{cccc}
 & \begin{array}{c} \mathbf{x}_{2;1} \\ \downarrow \end{array} & \begin{array}{c} \mathbf{x}_{2;2} \\ \downarrow \end{array} & & & \begin{array}{c} \mathbf{x}_{2;3} \\ \downarrow \end{array} & \begin{array}{c} \mathbf{x}_{2;4} \\ \downarrow \end{array} \\
 & \mathbf{D}_{2;1} & \mathbf{U}_{2;1} \mathbf{B}_{2;1,2} \mathbf{V}_{2;2}^H & & & \mathbf{U}_{1;1} \mathbf{B}_{1;1,2} \mathbf{V}_{1;2}^H & \\
 \mathbf{U}_{2;2} \mathbf{B}_{2;2,1} \mathbf{V}_{2;1}^H & & \mathbf{D}_{2;2} & & & & \\
 & \mathbf{U}_{1;2} \mathbf{B}_{1;2,1} \mathbf{V}_{1;1}^H & & & \mathbf{D}_{2;3} & \mathbf{U}_{2;3} \mathbf{B}_{2;3,4} \mathbf{V}_{2;4}^H & \\
 & & & & \mathbf{U}_{2;4} \mathbf{B}_{2;4,3} \mathbf{V}_{2;3}^H & & \mathbf{D}_{2;4}
 \end{array}
 \right)$$

We must compute $g_{k;i} = \mathbf{V}_{k;i}^H \mathbf{x}_{k;i}$. However, $\mathbf{V}_{1;1}$ not available (for example)! Hence

$$\begin{aligned}
 g_{1;1} &= \mathbf{V}_{1;1}^H \mathbf{x}_{1;1} \\
 &= \left(\mathbf{W}_{2;1}^H \mathbf{V}_{2;1}^H \quad \mathbf{W}_{2;2}^H \mathbf{V}_{2;2}^H \right) \begin{pmatrix} \mathbf{x}_{2;1} \\ \mathbf{x}_{2;2} \end{pmatrix} \\
 &= \mathbf{W}_{2;1}^H g_{2;1} + \mathbf{W}_{2;2}^H g_{2;2}
 \end{aligned}$$

FMM (Two Level: continued)

At output try the formula

$$b_{k;i} = \mathbf{D}_{k;i} \mathbf{x}_{k;i} + \mathbf{U}_{k;i} f_{k;i}$$

Obviously $f_{1;1} = \mathbf{B}_{1;1,2} g_{1;2}$. However $\mathbf{U}_{1;1}$ not available. What about $f_{2;1}$ (for example)?

$$\begin{aligned} b_{2;1} &= \mathbf{D}_{2;1} \mathbf{x}_{2;1} + \mathbf{U}_{2;1} \mathbf{B}_{2;1,2} g_{2;2} + \mathbf{U}_{2;1} \mathbf{R}_{2;1} \mathbf{B}_{1;1,2} g_{1;2} \\ &= \mathbf{D}_{2;1} \mathbf{x}_{2;1} + \mathbf{U}_{2;1} (\mathbf{B}_{2;1,2} g_{2;2} + \underbrace{\mathbf{R}_{2;1} \mathbf{B}_{1;1,2} g_{1;2}}_{f_{1;1}}) \\ &\quad \underbrace{\hspace{10em}}_{f_{2;1}} \end{aligned}$$

Hence

$$f_{2;1} = \mathbf{B}_{2;1,2} g_{2;2} + \mathbf{R}_{2;1} f_{1;1}$$

FMM

FMM up-sweep recursions

$$\begin{aligned}g_{k;i} &= \mathbf{V}_{k;i}^H \mathbf{x}_{k;i} \\g_{k-1;i} &= \mathbf{W}_{k;2i-1}^H g_{k;2i-1} + \mathbf{W}_{k;2i}^H g_{k;2i}\end{aligned}$$

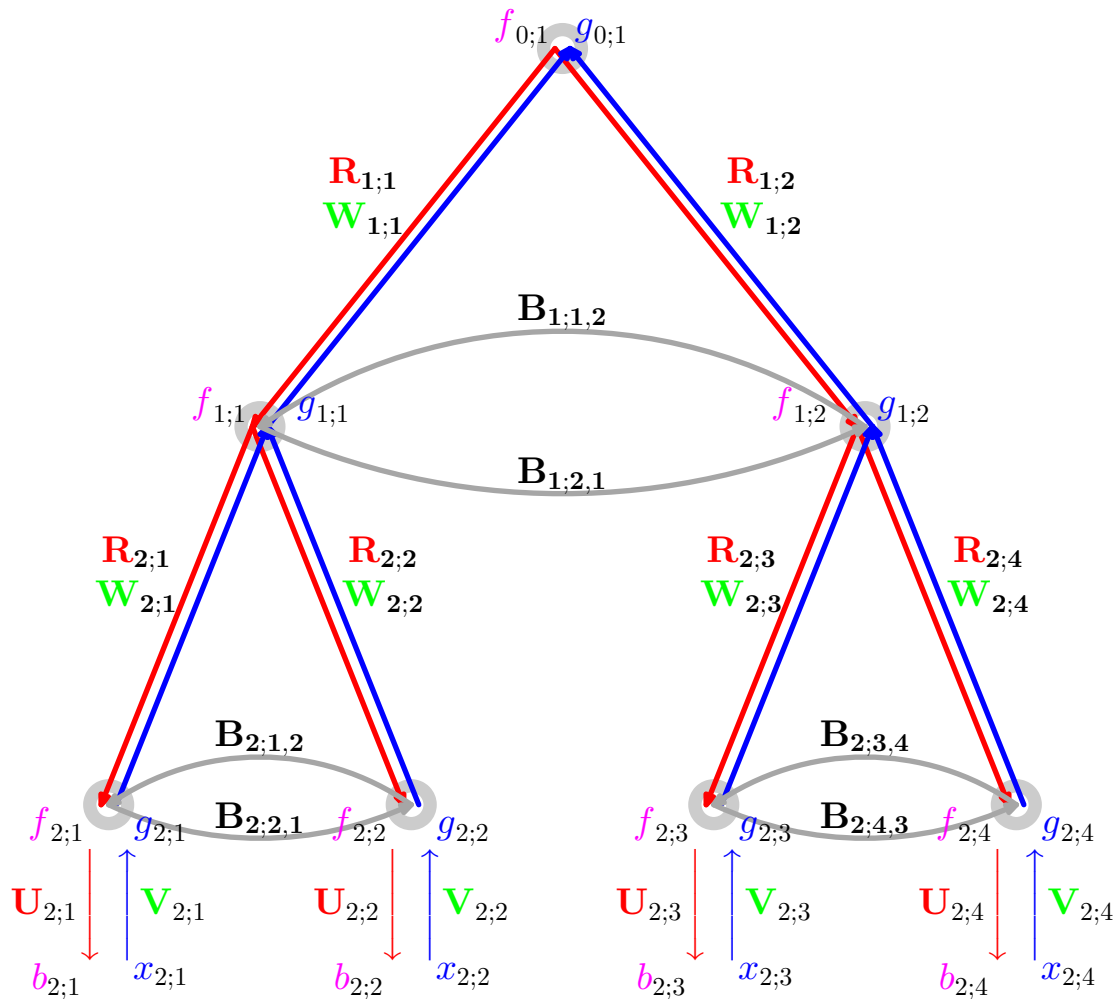
FMM down-sweep recursions

$$\begin{aligned}f_{0;i} &= (\cdot) \\f_{k;2i-1} &= \mathbf{R}_{k;2i-1} f_{k-1;i} + \mathbf{B}_{k;2i-1,2i} g_{k;2i} \\f_{k;2i} &= \mathbf{R}_{k;2i} f_{k-1;i} + \mathbf{B}_{k;2i,2i-1} g_{k;2i-1}\end{aligned}$$

Output formula

$$b_{k;i} = \mathbf{U}_{k;i} f_{k;i} + \mathbf{D}_{k;i} \mathbf{x}_{k;i}$$

FMM Signal Flow Graph



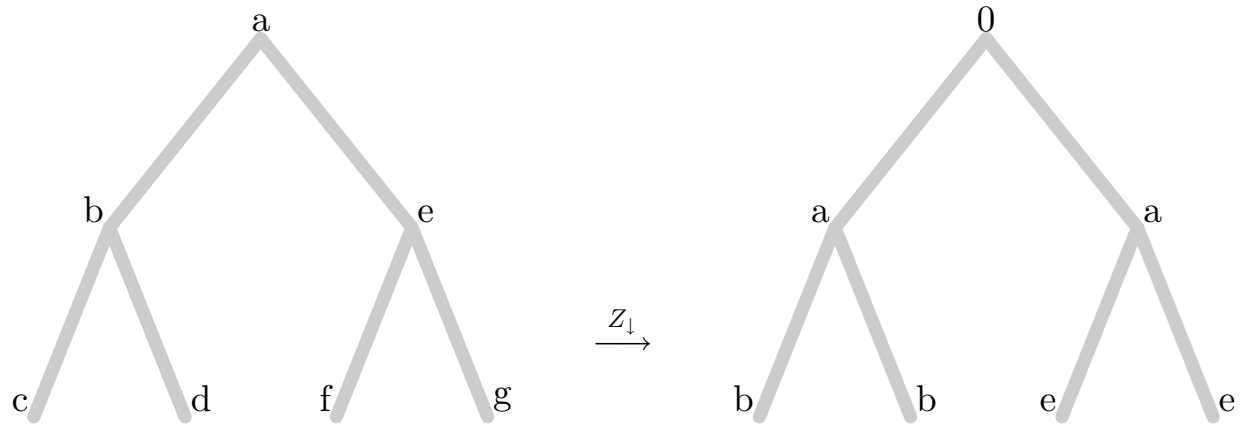
Sparse Representation

The signal flow graph via sparse matrices:

$$\begin{pmatrix} \mathbf{D} & 0 & \mathbf{U}P_{\text{leaf}} \\ 0 & \mathbf{B}Z_{\leftrightarrow} & \mathbf{R}Z_{\downarrow} - I \\ P_{\text{leaf}}^H \mathbf{V}^H & Z_{\downarrow}^H \mathbf{W}^H - I & 0 \end{pmatrix} \begin{pmatrix} x \\ g \\ f \end{pmatrix} = \begin{pmatrix} b \\ 0 \\ 0 \end{pmatrix}$$

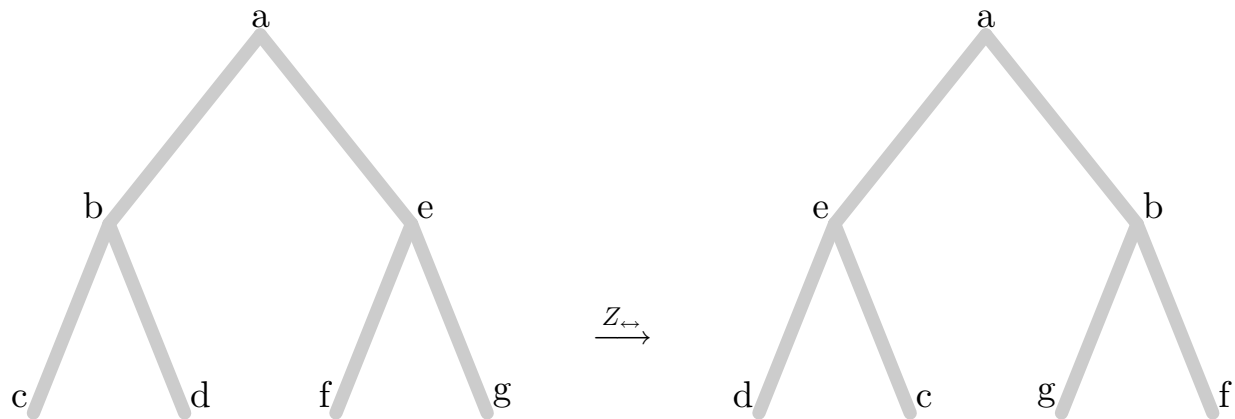
- \mathbf{D} , \mathbf{U} , \mathbf{V} and \mathbf{B} are block diagonal matrices.
- Z_{\downarrow} is a *shift* down matrix acting on trees.
- Z_{\leftrightarrow} exchanges siblings on the tree.
- P_{leaf} projects onto the leaf indices.
- The incidence graph of the block-sparse matrix is the binary tree.
- The binary tree has good separators.
- Hence we have a **fast** (LU or QR or ULV) solver.

Tree Shifts (Z_{\downarrow})



$$\underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ I & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 & 0 & 0 \\ I & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 & I & 0 & 0 \end{pmatrix}}_{Z_{\downarrow}} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{pmatrix} = \begin{pmatrix} 0 \\ a \\ b \\ b \\ a \\ e \\ e \end{pmatrix}$$

Tree Shifts (Z_{\leftrightarrow})



$$\underbrace{\begin{pmatrix} I & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 & 0 & I & 0 \end{pmatrix}}_{Z_{\leftrightarrow}} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{pmatrix} = \begin{pmatrix} a \\ e \\ d \\ c \\ b \\ g \\ f \end{pmatrix}$$

Schur Representation

The (1, 1) Schur complement of sparse form yields

$$(\mathbf{D} - \mathbf{U}P_{\text{leaf}}(I - \mathbf{R}Z_{\downarrow})^{-1}\mathbf{B}Z_{\leftrightarrow}(I - Z_{\downarrow}^H\mathbf{W}^H)^{-1}P_{\text{leaf}}^H\mathbf{V}^H)x = b$$

as the diagonal form of the HSS representation.

- Encodes, and reveals, the FMM recursions!
- Used to systematically find fast algorithms (Dewilde and van der Veen).

Fast Inversion

- HSS matrices can be inverted in $O(nk^2)$ flops.
- 2D IEs and PDEs can be solved in $O(n^{1.5})$ flops.
- 3D IEs and PDEs can be solved in $O(n^2)$ flops.
- A full **fast** HSS algebra can be developed.
 - $+$, \times
 - LU , QR , with each factor in HSS form.
 - Inverse in HSS form.

Fast Solver Timings

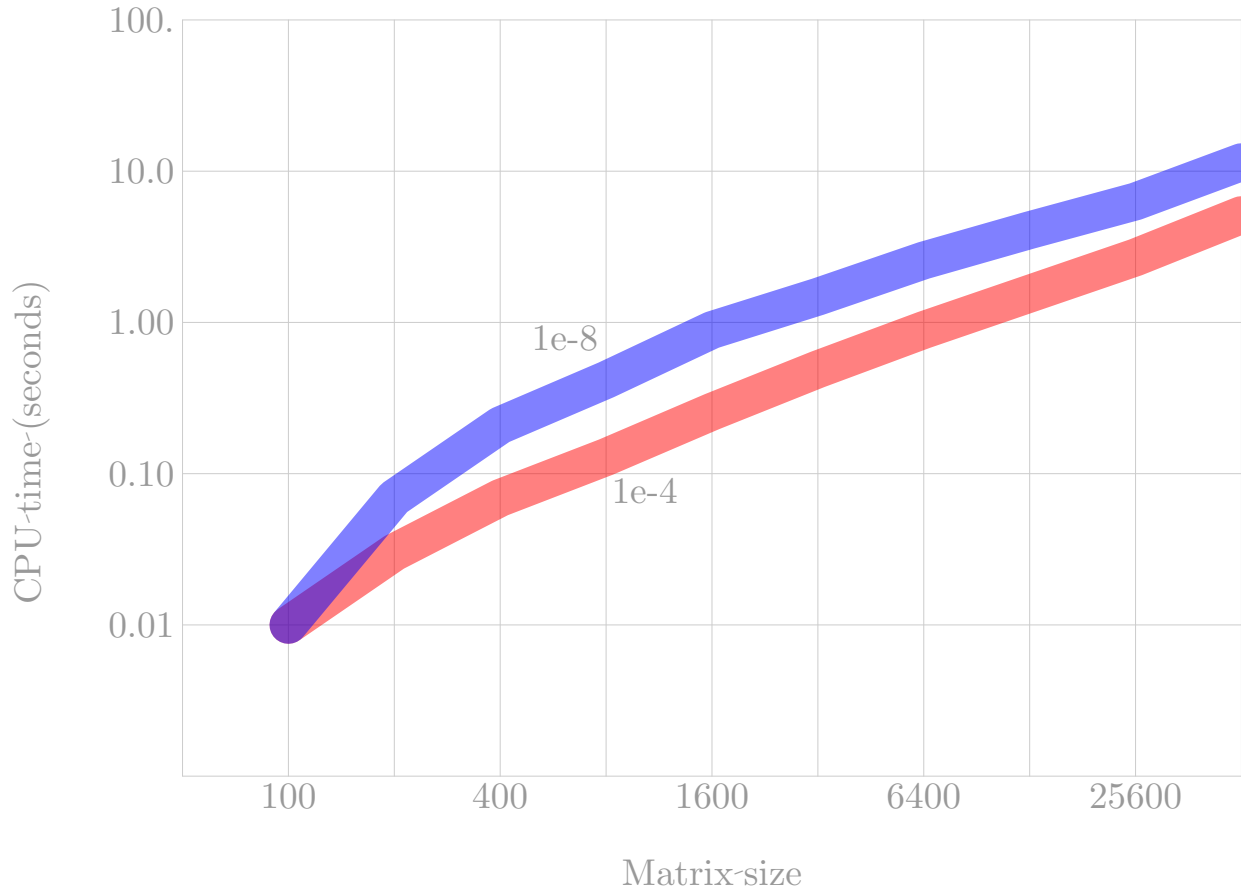
Kernel

$$A_{i,j} = \begin{cases} \frac{\sin(\pi\|x_i-x_j\|_2)}{\|x_i-x_j\|_2}, & i \neq j \\ 1, & i = j \end{cases}$$

x_i — random points on the 3D unit sphere in ND order.

Number of points	Time in seconds	
	tol. = 10^{-4}	tol. = 10^{-8}
100	0.01	0.01
200	0.03	0.07
400	0.07	0.21
800	0.13	0.42
1,600	0.26	0.90
3,200	0.50	1.48
6,400	0.89	2.57
12,800	1.58	4.05
25,600	2.71	6.35
51,200	5.11	11.48

Fast Solver Timings



Fast Pseudo-spectral Solver

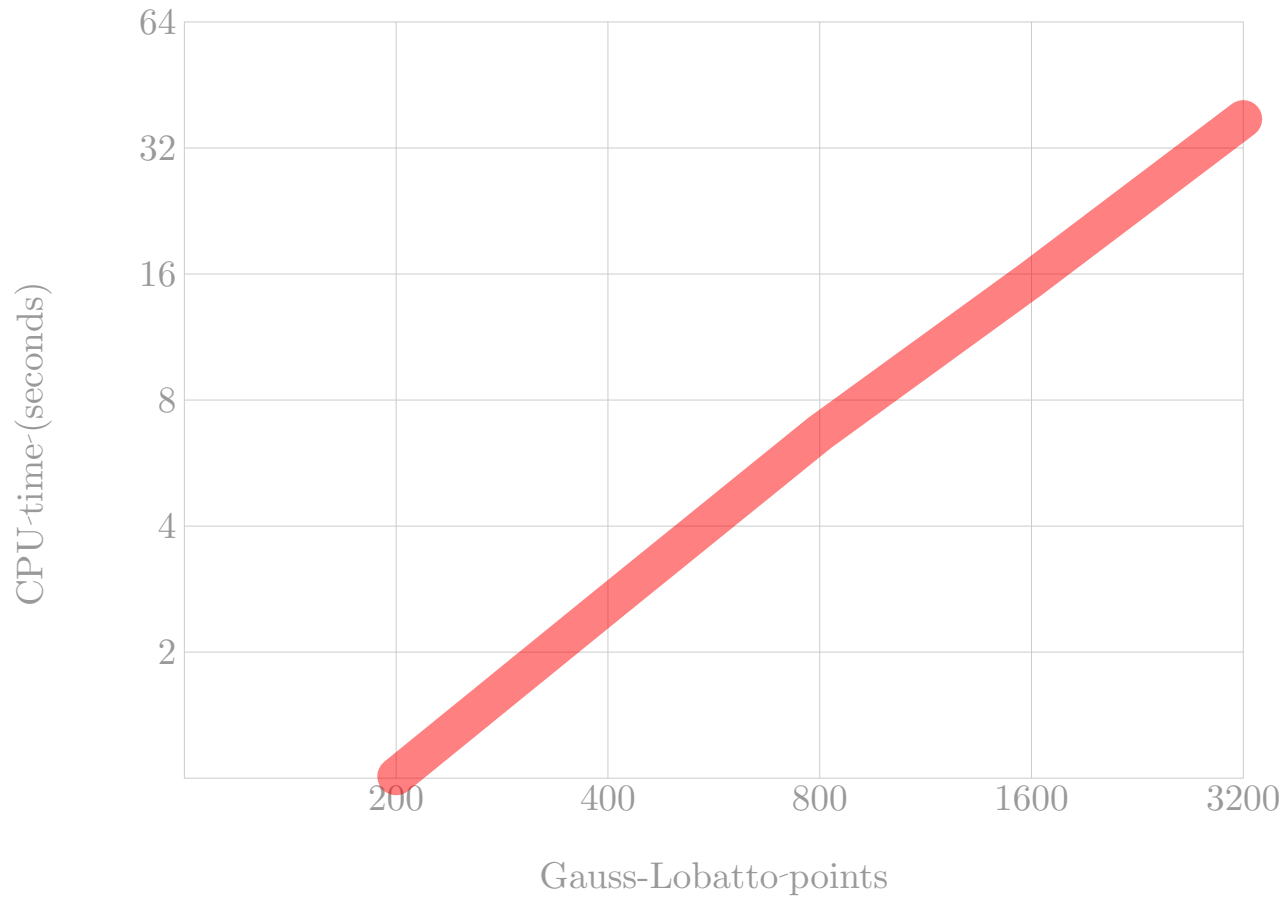
Diffusion equation

$$\begin{aligned}\frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2}, & -1 \leq x \leq 1 \\ u(x, 0) &= \exp x^2 \\ u(\pm 1, t) &= \sqrt{\frac{1}{t+1}} \exp\left(-\frac{1}{4(t+1)}\right), & t > 0\end{aligned}$$

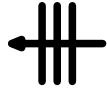
- Pseudo-spectral derivative operator has compact HSS representation.
- 60 time-steps of Crank-Nicholson.

Gauss-Lobatto points	CPU time (seconds)	error ($\times 10^{-6}$)
200	1.01	2.95
400	2.62	4.01
800	6.7	3.34
1600	15.66	3.60
3200	37.55	3.26

Fast Pseudo-spectral Solver Timings



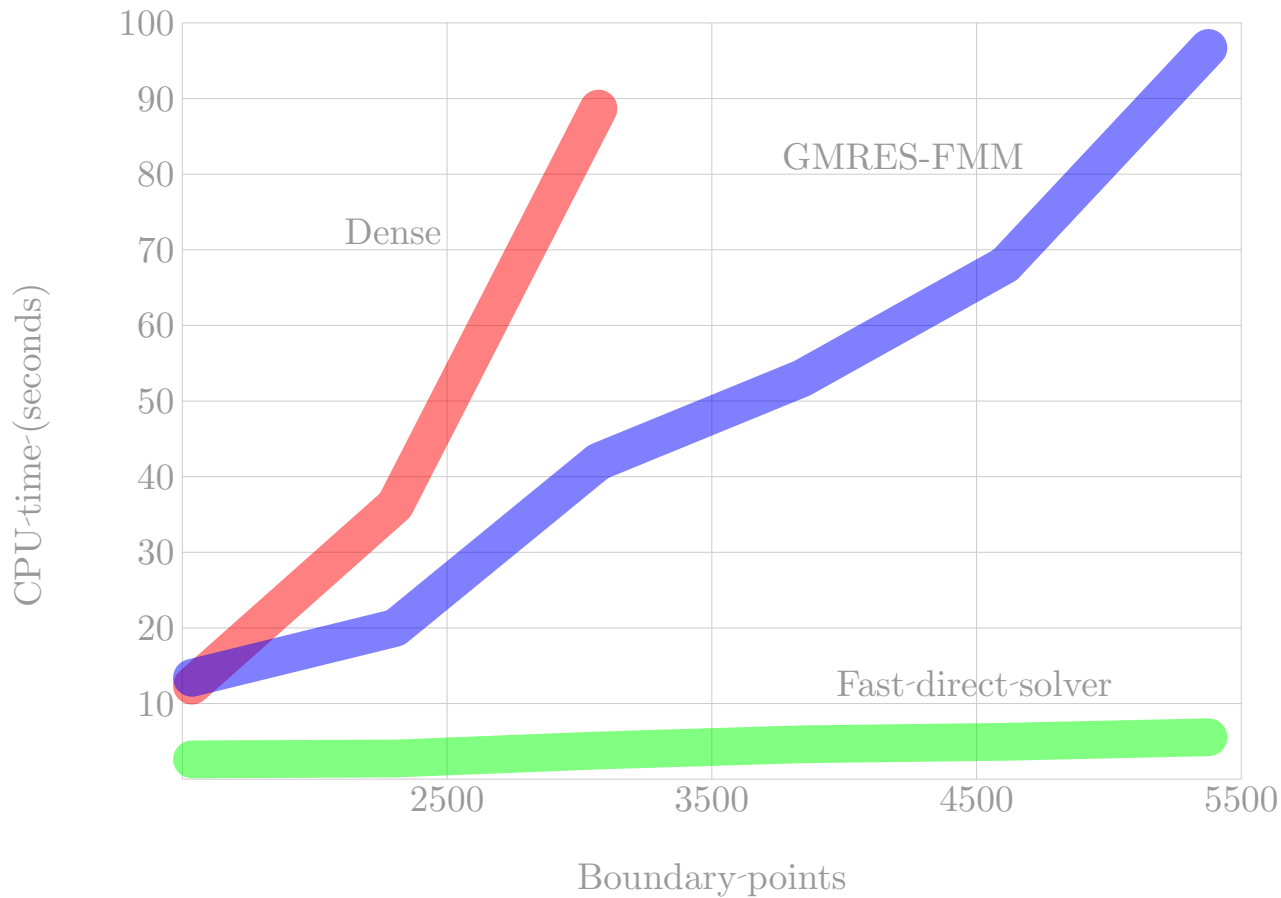
Fast inversion of FMM



- Classical exterior scattering (Helmholtz equation)
- Wavenumber = 100
- Boundary integral equation formulation
- FMM relative accuracy = 10^{-10}
- Run-time in seconds for **full FMM** inversion

n	Dense LU	GMRES-FMM	Sparse LU
1536	12.36	13.40	2.62
2304	36.24	20.03	2.73
3072	88.69	42.13	3.79
3840	(173)	53.13	4.61
4608	(299)	68.56	4.94
5376	(475)	96.65	5.54

Timings of FMM Inversion



Constructing HSS

Can we compute cheap HSS representations?

Yes, via **fast HSS algebra**.

HSS \times HSS

$A_{\text{HSS}} \times B_{\text{HSS}} = C_{\text{HSS}}$. Define $g_{k;i}$ and $f_{k;i}$ via

$$g_{k;i} = \mathbf{V}_{k;i}^H(A) \mathbf{U}_{k;i}(B),$$

$$\mathbf{D}_{k;i}(C) = \mathbf{D}_{k;i}(A) \mathbf{D}_{k;i}(B) + \mathbf{U}_{k;i}(A) f_{k;i} \mathbf{V}_{k;i}^H(B)$$

Up-sweep and down-sweep recursions

$$g_{k-1;i} = \mathbf{W}_{k;2i-1}^H(A) g_{k;2i-1} \mathbf{R}_{k;2i-1}(B) + \mathbf{W}_{k;2i}^H(A) g_{k;2i} \mathbf{R}_{k;2i}(B),$$

$$f_{k;i} = \mathbf{B}_{k;i,j}(A) g_{k;j} \mathbf{B}_{k;j,i}(B) + \mathbf{R}_{k;i}(A) f_{k-1;\lceil \frac{i}{2} \rceil} \mathbf{W}_{k;i}^H(B)$$

Answer

$$\mathbf{U}_{k;i}(C) = \begin{pmatrix} \mathbf{U}_{k;i}(A) & \mathbf{D}_{k;i}(A) \mathbf{U}_{k;i}(B) \end{pmatrix},$$

$$\mathbf{V}_{k;i}(C) = \begin{pmatrix} \mathbf{D}_{k;i}^H(B) \mathbf{V}_{k;i}(A) & \mathbf{V}_{k;i}(B) \end{pmatrix},$$

$$\mathbf{R}_{k;i}(C) = \begin{pmatrix} \mathbf{R}_{k;i}(A) & \mathbf{B}_{k;i,j}(A) g_{k;j} \mathbf{R}_{k;j}(B) \\ & \mathbf{R}_{k;i}(B) \end{pmatrix},$$

$$\mathbf{W}_{k;i}(C) = \begin{pmatrix} \mathbf{W}_{k;i}(A) \\ \mathbf{B}_{k;j,i}^H(B) g_{k;j}^H \mathbf{W}_{k;j}(A) & \mathbf{W}_{k;i}(B) \end{pmatrix},$$

$$\mathbf{B}_{k;i,j}(C) = \begin{pmatrix} \mathbf{B}_{k;i,j}(A) & \mathbf{R}_{k;i}(A) f_{k-1;\lceil \frac{i}{2} \rceil} \mathbf{W}_{k;j}^H(B) \\ & \mathbf{B}_{k;i,j}(B) \end{pmatrix}.$$

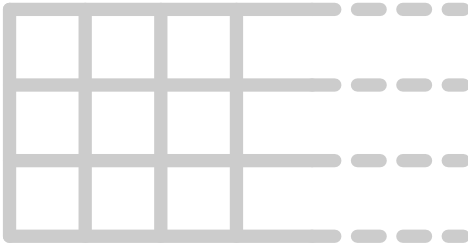
Superfast?

- How to break the graph-theoretic barriers?
- These are based on lower bounds for **exact** Gaussian elimination.
- Approximation is the key.
- Approximation not available in a purely algebraic setting, honoring the lower-bounds.
- Approximation is available for PDEs and IEs.

Superfast 2D PDE Solver

$$F_1 \nabla \cdot (F_2 \nabla u) + F_3 \cdot \nabla u + F_4 u = f_1$$

$$u = f_2, \quad \text{on boundary}$$



FD discretization:

$$\begin{pmatrix} A_1 & B_1 & & & & \\ C_1 & A_2 & B_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & C_{n-2} & A_{n-1} & B_{n-1} & \\ & & & C_{n-1} & A_n & \end{pmatrix}$$

Gaussian elimination:

$$S_{i+1} = A_{i+1} - C_i S_i^{-1} B_i$$

Everything is (and remains approximately) HSS \Rightarrow Superfast!

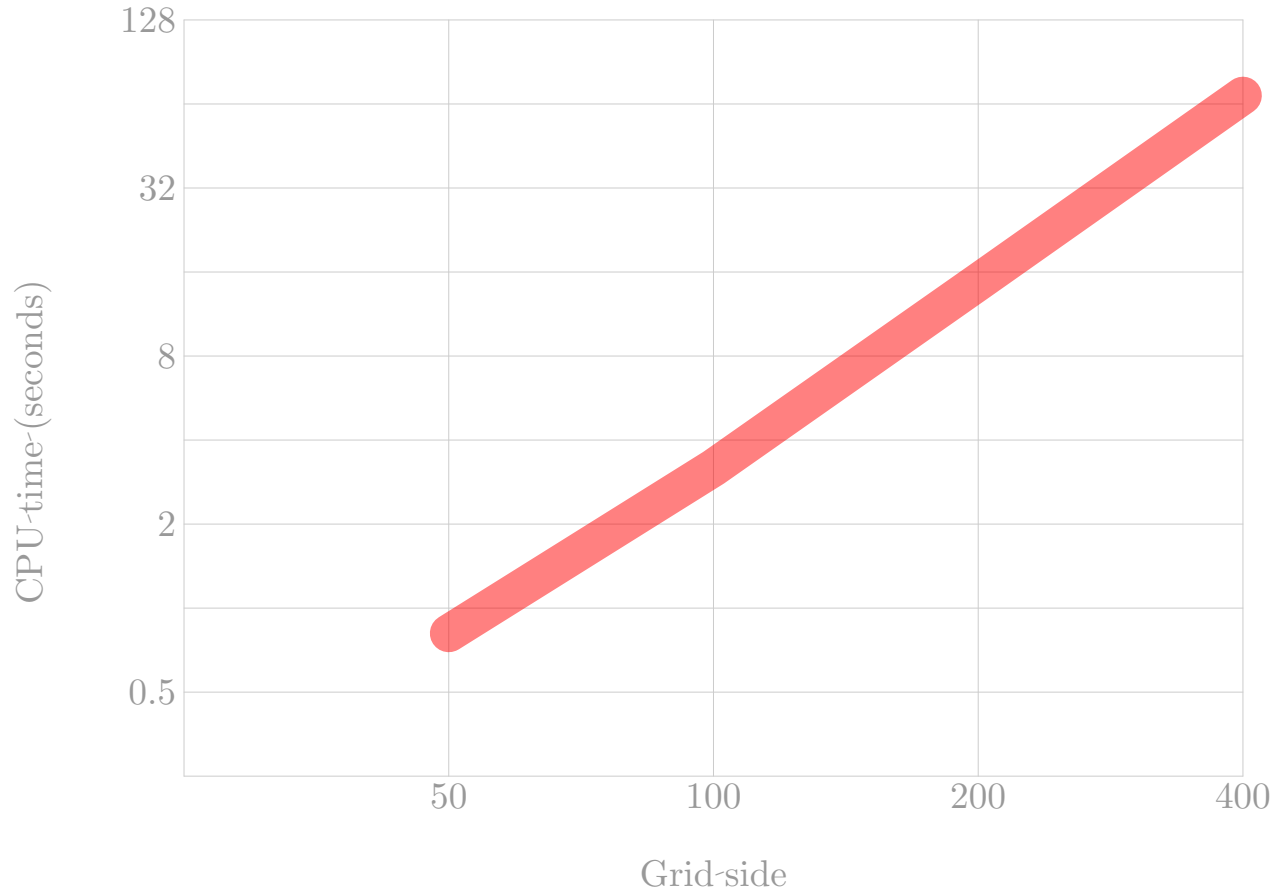
Preliminary Timings

Laplace's equation with Dirichlet conditions on square using 9-point stencil FD scheme.

$n \times n$ grid	Time in seconds
50	0.61
100	3.20
200	14.81
400	68.90

Clearly faster than $O(n^3)$! Can do much better in ND ordering.

Preliminary Superfast Solver Timings



Superfast for ND order

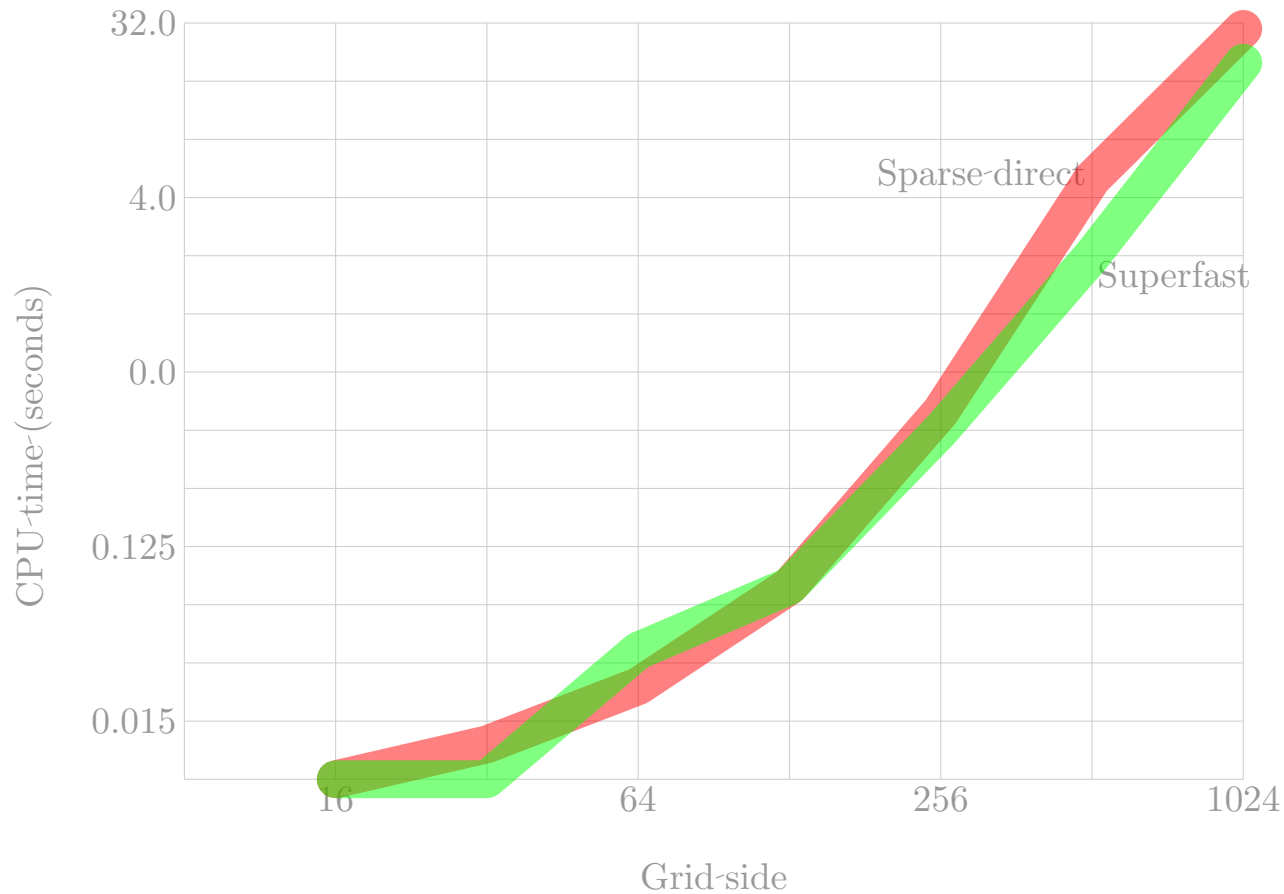
$$\frac{\partial}{\partial x} \left(p \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left(q \frac{\partial u}{\partial y} \right) = f(x, y), \quad (x, y) \in \Omega \equiv [0, 1] \times [0, 1]$$
$$u = 0, \quad (x, y) \in \partial\Omega.$$

- $p = q = 1$.
- p and q uniform random on $[0, 1]$!
- p, q piece-wise constant with $O(n)$ pieces.
- $p = q = 10^{-7} + |\sin(\rho(x^2 + y^2))|$ with very large ρ .

Run-time in seconds for 5-point stencil FD

$n \times n$ grid	15	31	63	127	255	511	1023
direct	0.0	0.01	0.02	0.1	0.8	5.0	30
tol. = 10^{-6}	0.0	0.0	0.04	0.1	0.5	2.0	20

Timings of Superfast Solver for ND Order



Conclusion

- Fill-in for PDEs is structured.
- The structure is identical to that of integral and pseudo-spectral differential operators.
- Structure does not depend on order of discretization.
- Structure can be captured as HSS representation on the fly.
- **Result:**

Direct solvers that scale optimally in grid size