Nonsymmetric Algebraic Fast Multipole Method

Severin Reiz[§], Chenhan Yu[†], James Levitt[†], George Biros[†], Hans-Joachim Bungartz[§]
[§] Technical University of Munich, [†] University of Texas at Austin, **reiz@in.tum.de**, chenhan@utexas.edu, jlevitt@ices.utexas.edu, gbiros@ices.utexas.edu, bungartz@in.tum.de

$\mathcal{H}$-Matrix Algebra: Matrix operations with hierarchical Approximations

Applications:

Kernel methods (e.g. KDE), neural-networks, PDEs, graphs and covariance operators etc

Goals:

(1) constructing $\tilde{K} \approx K$ requires $\mathcal{O}(N \log N)$ work;
(2) a matvec with $\tilde{K}$ also requires $\mathcal{O}(N \log N)$ work;
(3) $||\tilde{K} - K|| \leq \epsilon K$, where $0 \ll \epsilon \ll 1$ is a user-defined error tolerance

Challenges:

- Approximate tunable correctness
- Complexity constraints
- Load balancing and communication

Hierarchical Off Diagonal Low Rank (HODLR)[1,2]

$$K \approx \tilde{K} = \begin{bmatrix} \begin{bmatrix} \tilde{K}_{11}^{(2)} & (U\Sigma V^*)_{21}^{(2)} \\ (U\Sigma V^*)_{21}^{(2)} & \tilde{K}_{22}^{(2)} \end{bmatrix} & (U\Sigma V^*)_{12}^{(1)} \\ (U\Sigma V^*)_{21}^{(1)} & \begin{bmatrix} \tilde{K}_{33}^{(2)} & (U\Sigma V^*)_{34}^{(2)} \\ (U\Sigma V^*)_{43}^{(2)} & \tilde{K}_{44}^{(2)} \end{bmatrix} \end{bmatrix}$$

Fast multipole additive splitting:

$$\tilde{K}_{\alpha\alpha} = \begin{bmatrix} \tilde{K}_{11} & 0 \\ 0 & \tilde{K}_{rr} \end{bmatrix} + \begin{bmatrix} 0 & UV_{1r} \\ UV_{r1} & 0 \end{bmatrix} + \begin{bmatrix} 0 & S_{1r} \\ S_{r1} & 0 \end{bmatrix}$$

Compressed matrices offer speed ups in computations!

Matrix-vector product: $u_i = \underbrace{\sum_{p \in \texttt{Near}_i} K_{ip} w_p}_{\text{block diagonal and sparse}} + \underbrace{\sum_{p \in \texttt{Far}_i} K_{ip} w_p}_{\text{low rank}}$

Matrix inversion: $K = D + UV = D(Id + \underbrace{W}_{D^{-1}U} V)$, $Z = (Id + VW)^{-1}$

Sherman-Morrison-Woodbury $K^{-1} = (Id - WZV)D^{-1}$

Geometry-oblivious Fast Multipole Method (GOFMM)

Partitioning schemes $\begin{cases} \text{geometric} \\ \text{geodesic} \\ \text{geometry-oblivious} \end{cases}$

Low-rank decomposition

Nested interpolative decomposition

$$G \approx G_{col} P$$

- G_{col} is a subset of columns of G
- P a projection matrix

Error estimate

$$\|G_{col}P - G\|_2 \leq \sigma_{s+1}\sqrt{s(n-s)+1}$$

Rank-revealing QR factorization (GEQP3)
Triangular solve (TRSM)
Alternative approach using randomization[3]

Randomization: Distance based **sampling**

- Compute nearest neighbors
- Start on a leaf level
  - Sample rows from neighboring indices

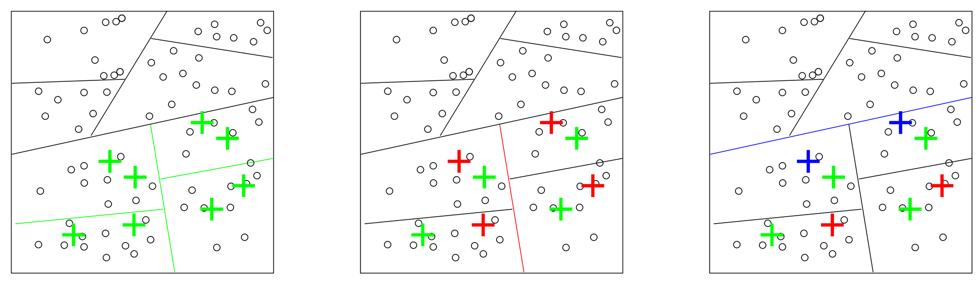
$$\mathcal{N}_\alpha = \cup_i \mathcal{N}_i \text{ where } i \in \alpha$$

  - Select skeletons and save

$$\mathcal{N}_\alpha = \cup_i \mathcal{N}_i \text{ where } i \in \mathcal{S}_\alpha \ .$$

- On interior levels
  - Use columns from children $\mathcal{S}_{\text{left}} \cup \mathcal{S}_{\text{right}}$
  - Sample rows from children skeleton neighbors

$$\mathcal{N}_\alpha^s = (\mathcal{N}_{left} \cup \mathcal{N}_{right}) \backslash (\texttt{Nodes}_{left} \cup \texttt{Nodes}_{right})$$



Kernel distance: Gram vector space

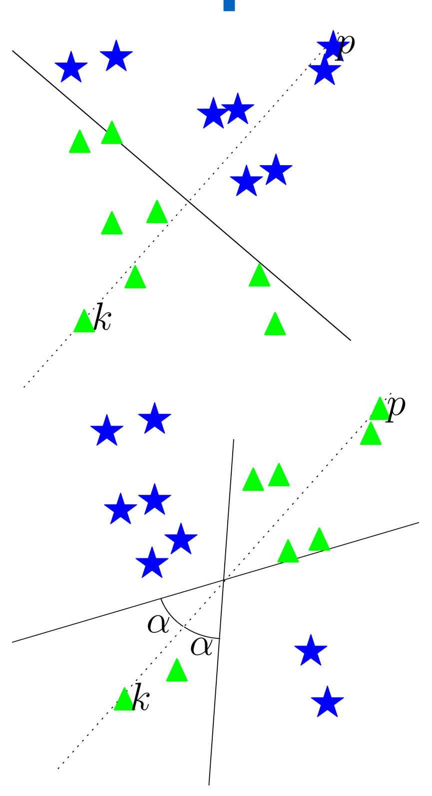
SPD Matrix can be resembled by scalar product of unknown vectors ϕ

$$K_{ij} = \langle \phi_i, \phi_j \rangle$$

- **Two** distances strategies: Gram-ℓ_2 and Gram-angle (introduced in[4])

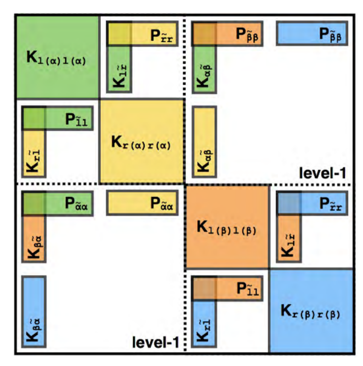
$$||\phi_i - \phi_j||_2^2 = \langle \phi_i - \phi_j, \phi_i - \phi_j \rangle = \underbrace{\langle \phi_i, \phi_i \rangle}_{K_{ii}} - 2\underbrace{\langle \phi_i, \phi_j \rangle}_{K_{ij}} + \underbrace{\langle \phi_j, \phi_j \rangle}_{K_{jj}}$$

$$\cos(\sphericalangle(\phi_i, \phi_j)) = \frac{\langle \phi_i, \phi_j \rangle}{||\phi_i|| \cdot ||\phi_j||} = \frac{\langle \phi_i, \phi_j \rangle}{\sqrt{\langle \phi_i, \phi_i \rangle \langle \phi_j, \phi_j \rangle}}$$



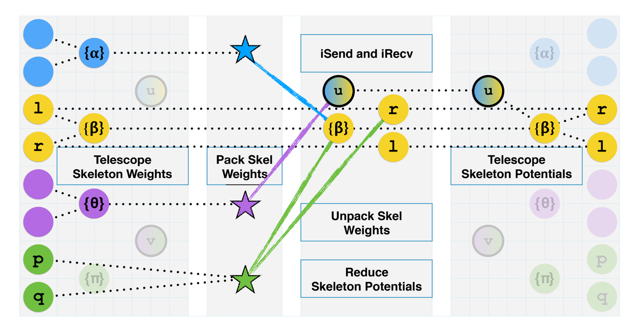
Numerical simulation codes

- `gofmm_python`[a]
  - Suitable for development prototyping
  - Modular framework
  - Suitable to check approximability for new test cases



[a] `https://gitlab.lrz.de/ga36wom/gofmm\_python`

- `MPI-GOFMM`[a]
  - C++ with MPI implementation
  - Load balancing through task model
  - Runtime dependency analysis
  - Asynchronous evaluation scheme



[a] `https://github.com/severin617/hmlp-1`

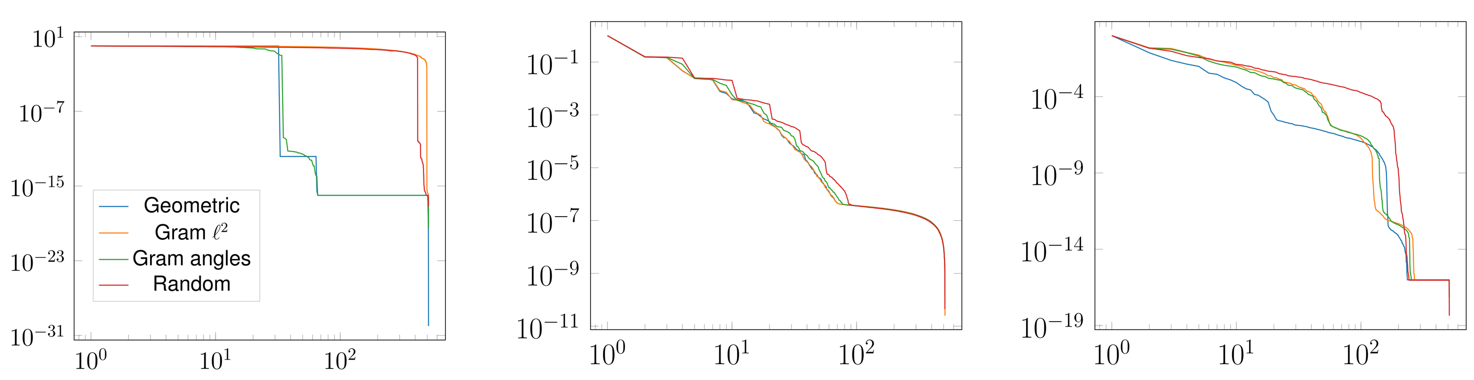
Results

Evaluation of geometry-oblivious scheme

Gram scheme: Partitioning

- For many cases similar spectra (lower plots) compared to geometric schemes
- Allows applicability where no geometry is available (graphs etc)

Spectra of off-diagonal level-1 block (σ_i/σ_1 vs. i)



Gram scheme: Accuracy

`ASKIT`:[5] Approximate Skeletonization Kernel-Independent Treecode in High Dimensions

- Geometric scheme
- Aim: Kernel methods in high dimensional feature space
- Similar neighbor-sampling and approximation scheme

Parameters			`ASKIT`			`GOFMM`		
case	N	τ	ϵ_2	Comp	Eval	ϵ_2	Comp	Eval
K04	36 864	1E−3	2E−4	0.3	2E−2	2E−4	0.6	2E−2
K04	36 864	1E−6	8E−7	1.4	4E−2	7E−7	1.0	3E−2
K04	65 536	1E−3	2E−4	1.0	4E−2	2E−4	1.2	4E−2
K04	65 536	1E−6	7E−7	2.2	8E−2	6E−7	1.7	4E−2
K06	36 864	1E−3	4E−2	6.6	6E−2	3E−2	3.3	4E−2
K06	36 864	1E−6	2E−2	7.4	6E−2	3E−2	4.8	5E−2
K06	65 536	1E−3	4E−2	11.1	1E−1	4E−2	5.7	8E−2
K06	65 536	1E−6	5E−2	12.0	1E−1	4E−2	7.7	9E−2

MPI-GOFMM

	`ScaLAPACK`	`STRUMPACK`[6]			MPI-GOFMM		
case	multiplication	ϵ_2	compression	multiplication	ϵ_2	compression	multiplication
K04	4.1	0.17	224	7.47	1.7E-05	1.69	0.09
K07	4.0	0.02	15.8	3.27248	1.0E-04	1.45	0.04
K11	4.1	0.01	93.6	2.195	7.9E-06	1.52	0.04
K12	4.1	0.11	222	4.28205	6.6E-05	1.64	0.05
G03	2.8	0.10	**33.5**	2.081	7E-05	**1.44**	0.04
H02	3.9	0.09	**19.6**	5.3	7.0E-04	**11.65**	0.57

Table 1: $100,000 \times 100,000$ float32 matrices from kernels, PDEs, graphs, Hessians ($\mathcal{O}(N \log N)$ work) on 4 "Skylake" nodes

Research Focus

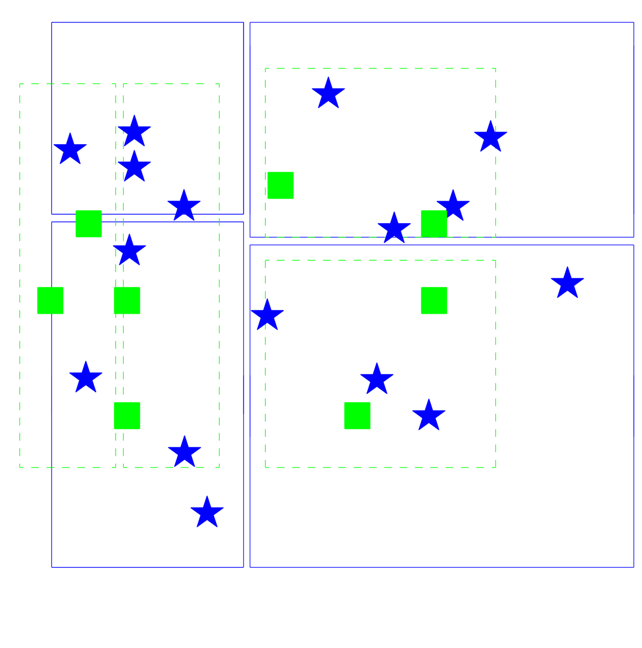
Nonsymmetric case: ($K \in \mathbb{R}^{N \times M}$)

- GOFMM only works for SPD
- Nonsymmetric scheme works currently on points
  - Single tree or dual tree?
  - Aim: matrix-vector multiplication in $(N+M)\log(N+M)$

Applicability studies and acceleration

- Entry look-up requires $\mathcal{O}(1)$
- Gauss-Newton Hessians in Neural Networks, studies on approximate entries

Matrix Inversion



References

[1] W. Hackbusch, *Hierarchical matrices: algorithms and analysis*, vol. 49. Springer, 2015.
[2] S. Ambikasaran, "A fast direct solver for dense linear systems." `https://github.com/sivaramambikasaran/HODLR`, 2013.
[3] P.-G. Martinsson, G. Quintana Orti, N. Heavner, and R. van de Geijn, "Householder QR factorization with randomization for column pivoting (HQRRP)," *SIAM Journal on Scientific Computing*, vol. 39, no. 2, pp. C96–C115, 2017.
[4] C. D. Yu, J. Levitt, S. Reiz, and G. Biros, "Geometry-oblivious FMM for compressing dense SPD matrices," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, p. 53, ACM, 2017.
[5] W. B. March, B. Xiao, C. D. Yu, and G. Biros, "ASKIT: An Efficient, Parallel Library for High-Dimensional Kernel Summations," *SIAM Journal on Scientific Computing*, vol. 38, no. 5, pp. S720–S749, 2016.
[6] F.-H. Rouet, X. S. Li, P. Ghysels, and A. Napov, "A distributed-memory package for dense hierarchically semi-separable matrix computations using randomization," *ACM Transactions in Mathematical Software*, vol. 42, pp. 27:1–27:35, June 2016.