Fast Construction of Hierarchically Low Rank Matrices Using Randomized Sketching

Xiaoye Sherry Li
xsli@lbl.gov
Lawrence Berkeley National Laboratory

Randomized Numerical Linear Algebra, ICERM

Collaborators

Henry Boateng, San Francisco State University

Wajih Boukaram, KAUST

Lisa Claus, LBNL

Pieter Ghysels, AMD

David Keyes, KAUST

Yang Liu, LBNL

Osman Malik, Encubez

Yotam Yaniv, LBNL

George Turkiyyah, KAUST

Funding:

- Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration
- Scientific Discovery through Advanced Computing (SciDAC) program through the FASTMath Institute

Outline

- Background of hierarchical matrices
- Construction using randomization techniques
- Applications

Exploit STRUCTURES

1. Sparsity structure: defined by $\{0,1\}$ structure (Graphs).
   - Factors are denser (fill-in)
   - sparse LU: $O(N^2)$ flops and $O(N^{4/3})$ memory, for typical 3D PDEs

2. Low rank structure: in addition to structural sparsity, can find data-sparse structure in dense (sub)matrices (approximation)
   - Goal is to achieve $O(N)$ or $O(N\ polylog(N))$ memory & flops for compression, factorization ...
   - Hierarchical matrices: $\mathcal{H}$- & $\mathcal{H}^2$-matrix [Hackbusch et al. 1999] and their subclasses.

     [Bebendorf, Bini, Börm, Chandrasekaran, Chow, Darve, Dewilde, Grasedyck, Gu, Le Borne, Martinsson, Tygert, Van Barel, van der Veen, Vandebril, Xia, ..., MANY MORE]

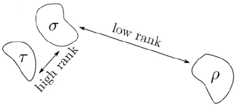
Application 1: Applying integral operator (N-body comput.)

$$(Kg)(x) = \int_Y K(x,y)g(y)dy, \quad \texttt{Kernel} \ K : X \times Y \to C$$

- Low rank property:

$$\texttt{admissible}(\tau, \sigma) = 1,$$

$$\textit{if} \ \frac{\texttt{Diam}(\tau) + \texttt{Diam}(\sigma)}{2} \le \eta \cdot \texttt{Dist}(\tau, \sigma)$$



- Many kernels have this property:
  - Gaussian kernel: $K(i,j) = \exp\left(-\frac{\|x_i - x_j\|_2^2}{2h^2}\right)$
  - Green's functions for Laplace equations:

$$\texttt{2D:} \ \log(||x_i - x_j|); \qquad \texttt{3D:} \ \frac{1}{(||x_i - x_j||)}$$

  - Green's functions for Helmholtz equations:

$$\texttt{2D:} \ H_0(k|x_i - x_j|); \qquad \texttt{3D:} \ \frac{e^{ik|x_i - x_j|}}{|x_i - x_j|}$$

  - Others: Fourier-type of kernel $e^{i\Phi(x,y)}$ where $\Phi(x,y)$ is smooth; e^{ixy}; Bessel function $J_0(xy)$

Hierarchical matrix approximation

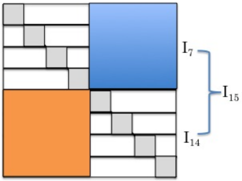
- Same mathematical foundation as FMM [Greengard/Rokhlin 1987], put in matrix form:
  - Diagonal block ("near field") represented exactly
  - Off-diagonal block ("far field") approximated via low-rank format

FMM separability of Green's function	Algebraic low rank off-diagonal
$G(x,y) \approx \sum_{\ell=1}^{r} f_\ell(x) g_\ell(y)$ $x \in X, y \in Y$	$A = \left[\begin{array}{c\|c} D_1 & U_1 B_1 V_2^T \\ \hline U_2 B_2 V_1^T & D_2 \end{array}\right]$

- Algebraic power: factorization, inversion, tensors, ...

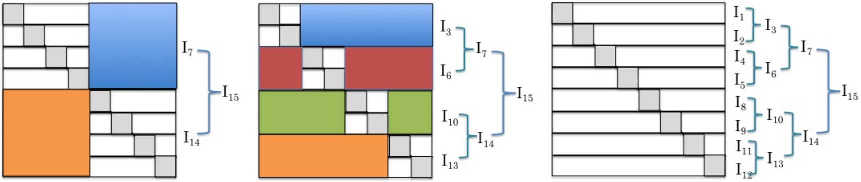
Hierarchical organization is key to optimal algorithm complexity

Example: Hierarchical Off-Diagonal Low Rank (HODLR)



Hierarchical organization is key to optimal algorithm complexity

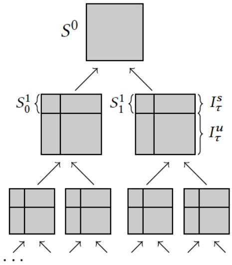
Example: Hierarchical Off-Diagonal Low Rank (HODLR)

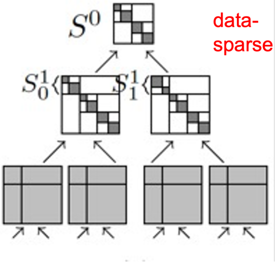


Algorithm can be top-down or bottom-up

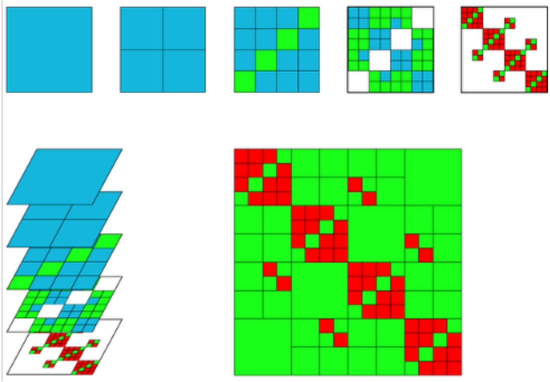
Application 2: discretized PDE

- Globally sparse, locally dense
  - Can embed LR data-sparse in sparse multifrontal algorithm
- In addition to structural sparsity, further apply LR data-sparsity to dense frontal matrices



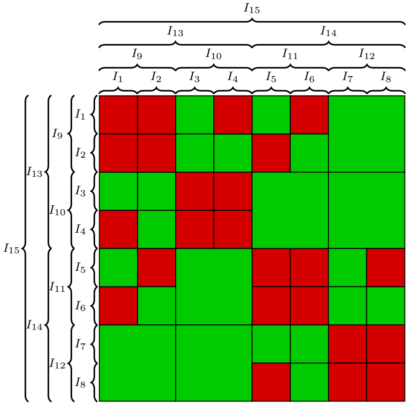


Hierrarchical matrix structure



1. Cluster the variables into a cluster tree
2. Define **admissibility condition** to determine whether a pair of clusters admits a low rank block
3. Hierarchically determine **admissible blocks** using a dual tree traversal on the cluster tree

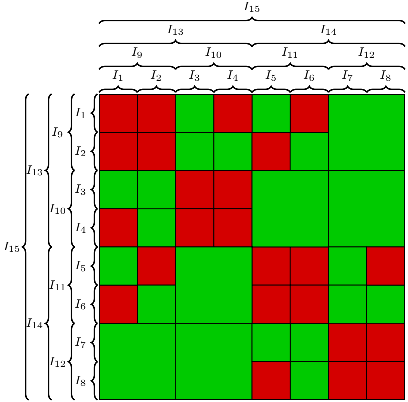
Cluster tree, Matrix tree

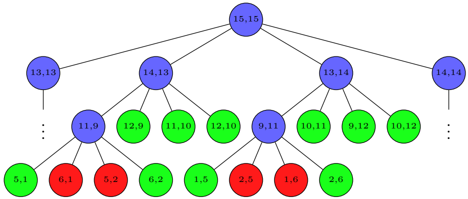


Cluster tree I: pairs of clusters define blocks in the matrix.

- Leaves form a block partitioning
- Red blocks = inadmissible leaves
- Green blocks = admissible blocks

Cluster tree, Matrix tree





Cluster tree I: pairs of clusters define blocks in the matrix.

- Leaves form a block partitioning
- Red blocks = inadmissible leaves
- Green blocks = admissible blocks

Matrix tree: generally not a complete tree

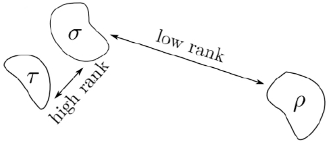
- Blue nodes = inadmissible blocks
- Red nodes = inadmissible leaves
- Green nodes = admissible leaves

Admissibility condition

$$\texttt{admissible}(\tau,\sigma) = 1,\ \textit{if}\ \frac{\texttt{Diam}(\tau)+\texttt{Diam}(\sigma)}{2} \le \eta \cdot \texttt{Dist}(\tau,\sigma)$$

Typically

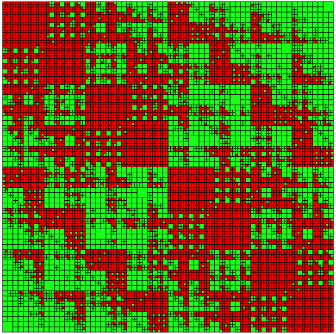
- weak admissibility: $\eta \ge 1$
  e.g., HODLR, HSS
- strong admissibility: $\eta \le 0.5$
  e.g., $\mathcal{H}$,$\mathcal{H}^2$



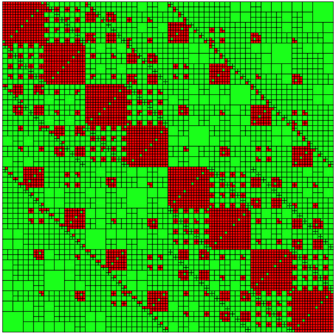
Admissibility condition

Example: block partitioning of a matrix associated with a set of $N = 2^{15}$ points in 3D geometry

Smaller η leads to more refined partitioning of off-diagonal blocks



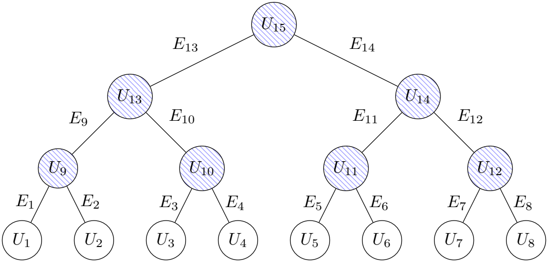
$\eta = 0.5$



$\eta = 0.7$

Nested basis leads to optimal complexity

- Represent parents in terms of their children using **transfer matrices**
  - $U_i^{l-1} = \begin{bmatrix} U_{i_1}^l & \\ & U_{i_2}^l \end{bmatrix} \begin{bmatrix} E_{i_1}^l \\ E_{i_1}^l \end{bmatrix}$
- Every low rank block has the form $A_{ts}^l = U_t^l S_{ts}^l {V_s^l}^T$
  - S_{ts}^l is the **coupling matrix**
- Assume we have an orthogonal basis: ${U_t^l}^T U_t^l = I$



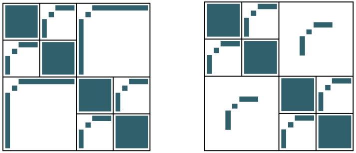
Classes of hierarchical low rank matrices

	Strong admissibility	Weak admissibility
Indep. bases	$\mathcal{H}$	HODLR
Nested based	$\mathcal{H}^2$ Inverse FMM	HSS, HIF Recursive Skeletonization

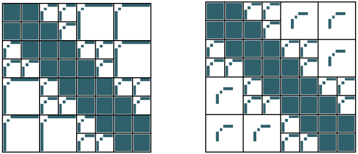
Classes of hierarchical low rank matrices

	Strong admissibility	Weak admissibility
Indep. bases	$\mathcal{H}$	HODLR
Nested based	$\mathcal{H}^2$ Inverse FMM	HSS, HIF Recursive Skeletonization

weak admissibility



strong admissibility



Deep dive: construction of hierarchical matrices

Low rank compression tool: adaptive randomized sketching

- Understanding probabilistic error

Low rank compression via randomized sketch

Approximate range of A:

1. Pick random matrix $\Omega_{n\times(k+p)}$, k target rank, p small, e.g. 10
2. Sample matrix $S = A\Omega$, with slight oversampling p
3. Compute $Q =$ ON-basis(S)

Benefits:

- Matrix-free, only need matvec
- When embedded in sparse frontal solver, simplifies "extend-add"

How to sketch only admissible blocks?

1. Preprocessing: Draw a big sketch using global A, $S = A\Omega$
   Sketching cost:
   - Dense matvec: $O(rN^2)$
   - FFT: $O(rN \log N)$ (e.g., Toeplitz)
   - FMM: $O(rN)$

   Can it be faster using structured random Ω?

[1] [Martinsson 2011; Xia 2013; Boukaram/Liu/Ghysels/L. 2025]

How to sketch only admissible blocks?

1. Preprocessing: Draw a big sketch using global A, $S = A\Omega$
   Sketching cost:
   - Dense matvec: $O(rN^2)$
   - FFT: $O(rN \log N)$ (e.g., Toeplitz)
   - FMM: $O(rN)$

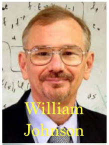
   Can it be faster using structured random Ω?

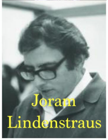
2. At each level, subtract small sketches corresponding to the inadmissible blocks [1]
   o Example: in HSS construction:
   block diagonal matrix at level ℓ: $D^{(\ell)} = diag(D_{\tau_1}, D_{\tau_2}, \ldots, D_{\tau_q})$

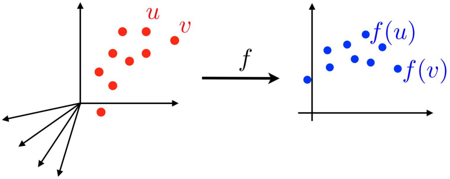
$$\text{Off-diagonal sketch: } S^{(\ell)} = \left(A - D^{(\ell)}\right)\Omega = S^r - D^{(\ell)}\Omega$$

[1] [Martinsson 2011; Xia 2013; Boukaram/Liu/Ghysels/L. 2025]

Unified view of sketching: Johnson-Lindenstrauss Lemma







Existence of good subspace embedding

JL-Lemma [Johnson-Lindenstrauss 1984]
Given $\varepsilon \in (0,1)$, let m and d be positive integers such that $d \geq 4(\varepsilon^2/2 - \varepsilon^3/3)^{-1} \log m$. For any set P of m points in $\mathbb{R}^n$ there exists $f : \mathbb{R}^n \to \mathbb{R}^d$ such that for all $u, v \in P$

$$(1-\varepsilon)\|u-v\|^2 \leq \|f(u)-f(v)\|^2 \leq (1+\varepsilon)\|u-v\|^2.$$

JL sketching operator: length preservation of a single vector

Suppose $\mathcal{D}$ is a distribution over matrices of size $d \times n$. We say that a matrix $R \sim \mathcal{D}$ is a $(n, d, \delta, \varepsilon)$-JL sketching operator if for any vector $x \in \mathbb{R}^n$ it satisfies

$$\Pr_{R \sim \mathcal{D}} \left[\frac{\left| \|Rx\|^2 - \|x\|^2 \right|}{\|x\|^2} > \varepsilon \right] < \delta.$$

- JL implies subspace embedding [Woodruff 2014]

Distributional JL family: drawn from certain distribution

Provided d sufficiently large, the following all satisfy JL sketching operator requirement:

- $R \sim Gaussian(n, d)$, i.i.d. normal distribution with mean 0, variance $1/d$ [Dasgupta/Gupta 2003]
- $R \sim SRHT(n, d), R = PHD$ [Halko/Martinsson/Tropp 2011]
  - P: sample d coordinates from n uniformly at random
  - H: Hadamard matrix
  - D: diagonal matrix, entries drawn from $\{+1, -1\}$ uniformly at random
- $R \sim SJLT(n, d, \alpha)$, Sparse JL Transform [Cohen/Jayrram/Nelson 2018]
  - α nonzero entries per row

Theory: Range-finder bound for general JL sketching operator

[Y. Yaniv, O.A. Malik, P. Ghysels, X.L.; CAMCoS 2025]

Theorem (Distributional JL implies Range-finder Bound)

Suppose $A \in \mathbb{C}^{m\times n}$ *is a matrix and let* $0 < r < \min(m, n)$ *be the target rank. If* R *is a* $(n, d, \frac{\delta}{2\max(5^{2r},n)}, \frac{\varepsilon}{12})$*-JL sketching operator with* $\varepsilon/12, \delta \in (0, 1)$ *and* $d = r + p$ *with* $p \geq 0$*, then the following holds with probability at least* $1 - \delta$*:*

$$||(I - P_Y)A|| \leq \left(\sqrt{1 + \frac{n(1+\varepsilon)}{(1-\varepsilon)}}\right)\sigma_{r+1}(A), \qquad (1)$$

where $Y = AR = Q\Omega$ *with* $P_Y = QQ^{\dagger}$*.*

Theory: Range-finder bound for special JL sketching operator: $R \sim Gaussian(n, d)$

Theorem ([Halko/Martinsson/Tropp 2011])

Choose oversampling parameter $p \geq 4$ *and target rank* $r \geq 2$, *where* $r + p \leq \min(m, n)$. *Draw* $R \in \mathbb{R}^{n\times(r+p)}$, $Y = AR = Q\Omega$ *and* $P_Y = QQ^*$, *t then the norm squared approximation error is*

$$||(I - P_Y)A|| \leq \left(1+16\sqrt{1 + \frac{r}{p+1}}\right)\sigma_{r+1}(A)+\frac{8\sqrt{r+p}}{p+1}\left(\sum_{j>r}\sigma_j^2(A)\right)^{1/2},$$

with probability at least $1 - 3e^{-p}$.

Theory: Range-finder bound for special JL sketching operator: $R \sim SJLT(n, d, \alpha)$

Theorem ([Yaniv/Malik/Ghysele/L. 2025])

A target rank $r < \min(m, n)$. Fix $\varepsilon, \delta \in (0, 1)$. If $\alpha = \Theta(\log^3(r/\delta)/\varepsilon)$, $d = \Omega(r \log^6(r/\delta)/\varepsilon^2)$, $Y = AR = Q\Omega$ and $P_Y = QQ^*$ then

$$||(I - P_Y)A|| \le \sigma_{r+1}(A)\sqrt{1 + \frac{1}{(1-\varepsilon)} \max\left(\frac{e^2 n\alpha}{d}, \log\left(\frac{2d}{\delta}\right) - \frac{n\alpha}{d}\right)}. \tag{2}$$

with probability $1 - \delta$.

Frobenius norm concentration bound for any JL sketching operator

- **Theorem** [Yaniv/Malik/Ghysele/L. 2025]
  Let $A \in \mathbb{R}^{m \times n}$ and $\varepsilon, \delta \in (0,1)$. If $R \in \mathbb{R}^{n \times d}$ is a $(n, d, \delta', \varepsilon)$-JL matrix where $\delta' = \delta/m$, then the following holds with probability at least $1 - \delta$:

$$(1-\varepsilon)\|A\|_F^2 \leq \|AR\|_F^2 \leq (1+\varepsilon)\|A\|_F^2.$$

- Furthermore, we established the following stochastic relationship between $||A||_F$ and $||S = AR||_F$: $\mathbb{E}\left[\frac{1}{\sqrt{d}}||S||_F^2\right] = ||A||_F^2$

- In practice, leads to robust stopping criteria in adaptive sketching
  Absolute: $||(I - QQ^*)A||_F \approx ||(I - QQ^*)S||_F \leq \varepsilon_a$
  Relative: $\frac{||((I-QQ^*)A||_F}{||A||_F} \approx \frac{||(I-QQ^*)S||_F}{||S||_F} \leq \varepsilon_r$

Sparse JLT is a highly structured random sparse matrix

Mitigate dense sampling cost

- SJLT [Kane/Nelson 2014; Cohen/Jayram/Nelson 2018]
  An SJLT matrix R of size $n \times d$ has a fixed number $\alpha \in [d]$ nonzero entries per row. The nonzero entries are drawn independently from two values $\{1/\sqrt{\alpha}, -1/\sqrt{\alpha}\}$ with equal probability. Example:

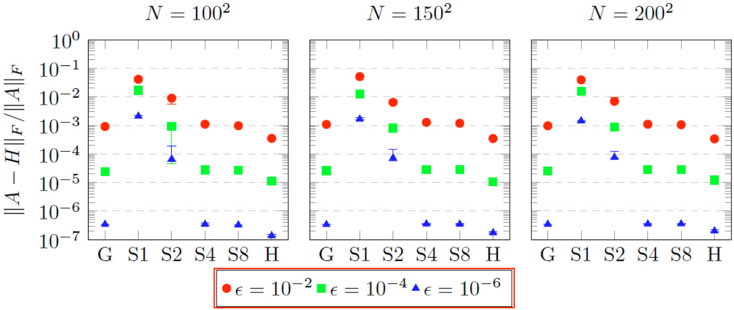
  $R \sim SJLT(4, 3, 2)$

$$R = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & -1 \\ 0 & -1 & -1 \\ 1 & -1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right)$$

- Sketching AR only involves add/subtract; no multiplication
- Implementation is no harder than sparse matrix-vector multiplication

SJLT-sketching in HSS construction: accuracy & runtime

- Compared to Gaussian sketching; with varying sparsity level, varying compression tolerance
  Example: the top frontal matrix of 3D Poisson



$\alpha = 4$ **is sufficient!**

- Seral runtime up to 2.5X faster
- Parallel runtime with 32 MPI prrocesses:

  Sketching time up to 17X faster

  Total time up to 14X faster

Applications

- Iterative linear solvers:
  Hierarchical matrix-vector multiplication

- Direct linear solvers:
  Factoization/inversion: ULV, $\mathcal{H}$-LU, $\mathcal{H}^2$-LU [2], ...

[2] on GPU: [Boukaram/Keyes/L./Liu/Turkiyyah 2026]

STRUMPACK – STRUctured Matrices PACKage
http://portal.nersc.gov/project/sparse/strumpack/

- Fully algebraic solvers/preconditioners
- Dense: Can take user input of cluster tree & block partitioning
  - HSS, BLR, $\mathcal{H}^2$ (soon)
  - ButterflyPACK integration/interface: Butterfly, HODLR, HODBF
- Sparse multifrontal direct solver
- Approximate sparse factorization preconditioner
- C++, MPI + OpenMP + CUDA, real & complex, 32/64 bit integers
- BLAS, LAPACK, Metis
- Optional: MPI, ScaLAPACK, ParMETIS, (PT-)Scotch, cuBLAS/cuSOLVER, SLATE, ZFP

Kernel Ridge Regression: Ridge Regression + Kernel Trick

1. Training stage to compute model parameters:
Need to minimize the cost function:

$$argmin_w \, C(\mathbf{w}) = \sum_i (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|^2$$

- x_i's are data points (rows of the data matrix $X^{n \times d}$)
- y_i's are their labels
- **w** is the normal vector to the target hyperplane

Can derive the optimal weights for the prediction model:

$$\mathbf{w} = X^T (\lambda I + X X^T)^{-1} \mathbf{y}$$

2. Prediction stage: given a new vector $\mathbf{x}_1$ from the test set, compute:

$$\begin{aligned} y_1 := \mathbf{w}^T \mathbf{x}_1 &= [(\lambda I + X X^T)^{-1} \mathbf{y}]^T X \mathbf{x}_1 \\ &\approx [(\lambda I + \mathcal{K}(X, X))^{-1} \mathbf{y}]^T \cdot \mathcal{K}(X, \mathbf{x}_1) \leftarrow \text{kernel trick} \end{aligned}$$

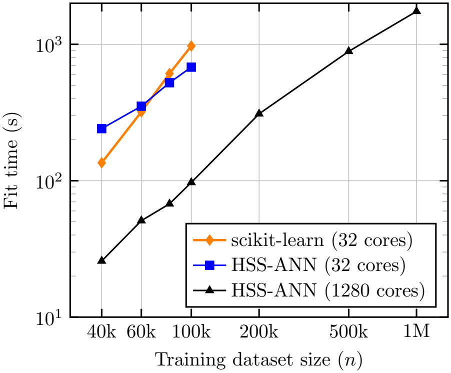
$\Longrightarrow$ Binary classification: class label predicted by the sign of y_1

STRUMPACK Python interface to Scikit-learn

- Scikit-learn: machine Learning in Python, http://scikit-learn.org/stable/
  - contains classifiers and regressors
  - but only provides shared-memory parallelism
- STRUMPACKKernel Python class:
  - derived from scikit-learn base classes BaseEstimator and ClassifierMixin
  - implements member functions: fit, predict and decision_function
  - can leverege all the other functions in scikit-learn

Time comparison between scikit-learn and HSS

Use Gaussian kernel $K(i,j) = \exp\left(-\frac{\|x_i - x_j\|_2^2}{2h^2}\right)$

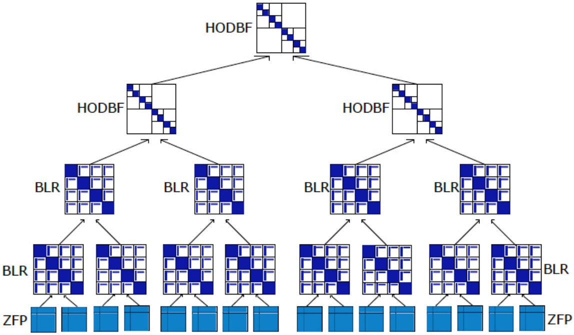


SUSY dataset (particle physics): classification to distinguish between a signal process which produces supersymmetric particles and a background process which does not

Rank structured multifrontal sparse factorization

Compressing large dense blocks in the multifrontal tree
Combining Hierarchically Off-Diagonal Butterfly (HODBF) and Block Low Rank (BLR)

- Largest: HODBF
- Medium: BLR
- Smaller: dense or lossy compression (ZFP)

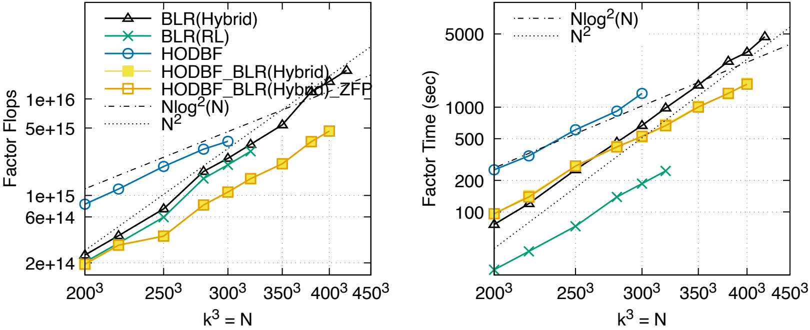


3D visco-acoustic wave propagation

Governed by the Helmholtz equation: $\mathbf{x} = (x_1, x_2, x_3)$

$$\left(\sum_i \rho(\mathbf{x}) \frac{\partial}{\partial x_i} \frac{1}{\rho(\mathbf{x})} \frac{\partial}{\partial x_i} \right) p(\mathbf{x}) + \frac{\omega^2}{\kappa^2(\mathbf{x})} p(\mathbf{x}) = -f(\mathbf{x})$$

- Solution method: FD on staggered grids using a 27-point stencil, 8 PML absorbing boundary layers



Summary: Stages of operation

- Data clustering, matrix reordering
- Compression – usually dominating cost
  - Complexity depends on: black-box Av and $A^T v$, black-box entry evaluation $A(i,j)$
  - Goal: $O(N \, log^{\alpha} N)$
- Building solvers
  - Iterative solver: matrix-vector multiplication
  - Direct solvers: factorization (e.g., ULV, H-LU), solve, inversion
- Principal tool for efficient implementation and parallelization
  Sweeping through "trees" upward / downward: cluster tree, separator tree, ...

Open Problems

- How to choose α in $SJLT(n, d, \alpha)$?
- Subsampled randomized trig transform (SRTT), or Fourier transform (SRFT)
- Does it make sense to do $\mathcal{H}$, $\mathcal{H}^2$-QR? How?
- Spectral analysis for matrices preconditioned by low-rank factorization
- Data-sparse rank analysis for matrix inverse
- Data-sparse for tensor computations