Matrix-free construction of HSS representations using adaptive randomized sampling

Xiaoye Sherry Li
xsli@lbl.gov

Gustavo Chavez, Pieter Ghysels, Chris Gorman, Yang Liu
Lawrence Berkeley National Laboratory

Chris Gorman, UC Santa Barbara

Randomized Numerical Linear Algebra and Applications

Acknowledgement

This research was supported by the Exascale Computing Project (http://www.exascaleproject.org), a joint project of the U.S. Department of Energys Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nations exascale computing imperative.
Project Number: 17-SC-20-SC

Hierarchical matrix approximation

- Same mathematical foundation as FMM (Greengard-Rokhlin'87), 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

$$G(x, y) \approx \sum_{\ell=1}^{r} f_\ell(x) g_\ell(y)$$

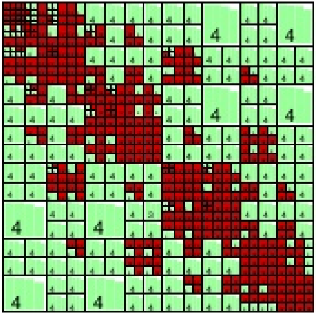
$$x \in X, y \in Y$$

Algebraic
low rank off-diagonal

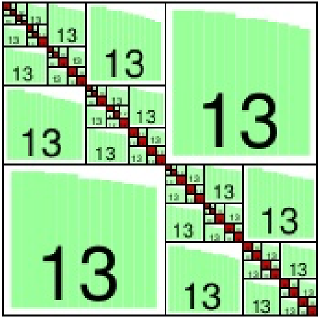
$$A = \left[\begin{array}{c|c} D_1 & U_1 B_1 V_2^T \\ \hline U_2 B_2 V_1^T & D_2 \end{array}\right]$$

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

Hierarchical matrix formats



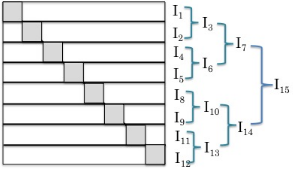
H-matrix (W. Hackbusch et al.)
O(r N log N)



HSS matrix (J Xia et al.)
O(r N)

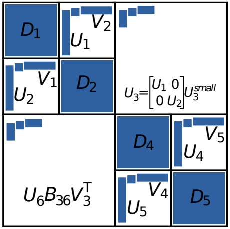
Block cluster tree and nested bases

Example: Hierarchically Semi-Separable matrices (HSS)



- Diagonal blocks are full rank: $D_\tau = A(I_\tau, I_\tau)$
- Off-diagonal blocks as low-rank:

$$A_{\nu_1,\nu_2} = A(I_{\nu_1}, I_{\nu_2}) = U_{\nu_1} B_{\nu_1,\nu_2} V_{\nu_2}^*$$



- **Column bases U and row bases V^* are nested:**

$$U_\tau = \begin{bmatrix} U_{\nu_1} & 0 \\ 0 & U_{\nu_2} \end{bmatrix} U_\tau^{\text{small}}, \; V_\tau = \begin{bmatrix} V_{\nu_1} & 0 \\ 0 & V_{\nu_2} \end{bmatrix} V_\tau^{\text{small}}$$

HSS matrix – ULV factorization

ULV-like factored form (U and V^* unitary, L triangular)

$$\Gamma_{1;b\leftrightarrow 2;t}\begin{bmatrix} I & & & \\ & \Omega_1 & & \\ & & I & \\ & & & \Omega_2 \end{bmatrix}\begin{bmatrix} \Gamma_{3;b\leftrightarrow 4;t} & \\ & \Gamma_{5;b\leftrightarrow 6;t} \end{bmatrix}\begin{bmatrix} \Omega_3 & & & \\ & \Omega_4 & & \\ & & \Omega_5 & \\ & & & \Omega_6 \end{bmatrix} A \begin{bmatrix} Q_3^* & & & \\ & Q_4^* & & \\ & & Q_5^* & \\ & & & Q_6^* \end{bmatrix}\begin{bmatrix} \Gamma^T_{3;b\leftrightarrow 4;t} & \\ & \Gamma^T_{5;b\leftrightarrow 6;t} \end{bmatrix}\begin{bmatrix} I & & & \\ & Q_1^* & & \\ & & I & \\ & & & Q_2^* \end{bmatrix}\Gamma^T_{1;b\leftrightarrow 2;t}$$

$$= \left[\begin{array}{ccc|ccc|c}
L_3 & & & & & & \\
0 & L_4 & & & & & \\
(\Omega_1 L_{4,3})_t & (\Omega_1 L_{3,4})_t & L_1 & & & & \\
\hline
 & & & L_5 & & & \\
 & 0 & & 0 & L_6 & & \\
 & & & (\Omega_2 L_{6,5})_t & (\Omega_2 L_{5,6})_t & L_2 & \\
\hline
(\Omega_1 L_{4,3})_b & (\Omega_1 L_{3,4})_b & W_{1;b} Q^*_{1;t} & \multicolumn{3}{c|}{B_{1,2} V_2^* \begin{bmatrix} V_5^* Q^*_{5;t} & & V_5^* Q^*_{5;b} & \\ & V_6^* Q^*_{6;t} & & V_6^* Q^*_{6;b} \end{bmatrix}\begin{bmatrix} I & \\ & Q_2^* \end{bmatrix}} & \\
\multicolumn{3}{c|}{B_{2,1} V_1^* \begin{bmatrix} V_3^* Q^*_{3;t} & & V_3^* Q^*_{3;b} & \\ & V_4^* Q^*_{4;t} & & V_4^* Q^*_{4;b} \end{bmatrix}\begin{bmatrix} I & \\ & Q_1^* \end{bmatrix}} & (\Omega_2 L_{6,5})_b & (\Omega_2 L_{5,6})_b & W_{2;b} Q^*_{2;t} & D_0
\end{array}\right]$$

Low rank compression via randomized sampling (RS)

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) via RRQR

Accuracy: [Halko, Martinsson, Tropp, '11]

- On average: $E(\|A - QQ^*A\|) = \left(1 + \frac{4\sqrt{k+p}}{p-1}\sqrt{\min\{m,n\}}\right)\sigma_{k+1}$
- Probabilistic bound: with *probability* $\geq 1 - 3\cdot 10^{-p}$,
  $\|A - QQ^*A\| \leq [1 + 9\sqrt{k+p}\sqrt{\min\{m,n\}}]\,\sigma_{k+1}$

(in 2-norm)

Low rank compression via randomized sampling (RS)

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) via RRQR

Accuracy: [Halko, Martinsson, Tropp, '11]

- On average: $E(\|A - QQ^*A\|) = \left(1 + \frac{4\sqrt{k+p}}{p-1}\sqrt{\min\{m,n\}}\right)\sigma_{k+1}$
- Probabilistic bound: with *probability* $\geq 1 - 3 \cdot 10^{-p}$,
  $\|A - QQ^*A\| \leq [1 + 9\sqrt{k+p}\sqrt{\min\{m,n\}}]\ \sigma_{k+1}$

(in 2-norm)

Benefits:

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

HSS compression via RS [Martinsson '11, Xia '13]

- R random matrix with $d = r + p$ columns
  - r is the **estimated maximum rank,** p is oversampling parameter
- Random sampling of matrix A
  - $S^r = AR$, columns of S^r span the column space of A
  - $S^c = A^* R$, columns of S^c span the row space of A
- Only sample off-diagonal blocks at each level (Hankel blocks):
  Block diagonal matrix at level ℓ: $D^{(\ell)} = diag(D_{\tau_1}, D_{\tau_2}, \ldots, D_{\tau_q})$

$$S^{(\ell)} = \left(A - D^{(\ell)}\right) R = S^r - D^{(\ell)} R$$

- Rank-revealing QR on $S^{(\ell)}$

Practical issues

- Need ε-rank: $\|A - QQ^* A\| \leq \varepsilon$
- Non-decay singular spectrum
- Sampling is expensive using traditional dense matvec

Solution:

1. Gradually increase sample size
   - User manually restart from scratch $\rightarrow$ costly!
   - Built-in automatic strategy $\rightarrow$ not to re-do already-compressed blocks.

     $\Rightarrow$ **Need good error estimation!**
2. Faster matvec in sampling: FFT, FMM, Gauss transform, $\mathcal{H}$-matrix, ...

Automatic adaptive sampling is essential for robustness

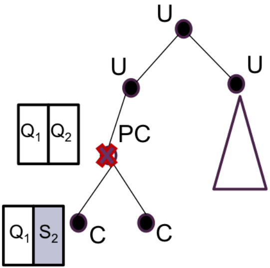
Increase sample size d, build Q incrementally **(block variant)**
$[S_1 \quad S_2 \quad S_3 \quad \dots]$

$Q \leftarrow \emptyset$;
$S_1 \leftarrow A\Omega_1$;
$i \leftarrow 1$;
WHILE (error still large) {
    $Q_i \leftarrow QR(S_i)$;   // Orthogonalize within current block
    $Q \leftarrow [Q \ Q_i]$;
    $S_{i+1} \leftarrow A\Omega_{i+1}$;   // New samples
    $S_{i+1} \leftarrow (I - QQ^*)S_{i+1}$; // Orthogonalize against previous Q
    Compute error;
    $i \leftarrow i + 1$;
}

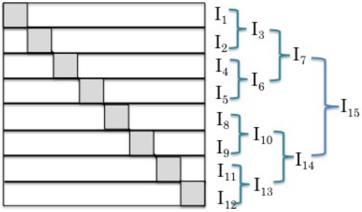
Adaptive sampling in HSS tree

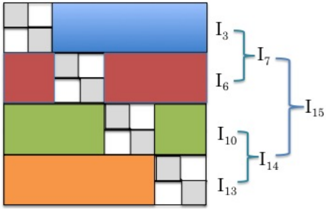
Recall:

- Only have $S = A\Omega$
- At level ℓ:
  $S^{(\ell)} = (A - D^{(\ell)})\Omega = S - D^{(\ell)}\Omega$



Node state:
U – untouched
C – compressed
PC – partially compressed





Adaptive sampling: probabilistic error estimation

- **Goal:** Bound errors for A: $||(I - QQ^*)A||$, but A is not available.
- **Approach:** Use sample S. **Need to establish a stochastic relationship between $||A||$ and $||S||$.**

Let $A \in \mathbb{R}^{m\times n}$, and $x \in \mathbb{R}^n$ with $x_i \sim \mathcal{N}(0,1)$. Consider SVD:

$$A = U\Sigma V^* = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^* \\ V_2^* \end{bmatrix}$$

Define $\xi = V^* x$, ξ is also a Gaussian random vector.

$$||Ax||_2^2 = ||\Sigma\xi||_2^2 = \xi_1^2\sigma_1^2 + \cdots + \xi_r^2\sigma_r^2. \tag{1}$$

Here, $\sigma_1 \geq \cdots \geq \sigma_r > 0$ are positive singular values. Therefore,

$$\mathbb{E}\left(||Ax||_2^2\right) = \sigma_1^2 + \cdots + \sigma_r^2 = ||A||_F^2. \tag{2}$$

For d sample vectors: $\mathbb{E}\left(||S||_F^2\right) = d\,||A||_F^2$.

Adaptive sampling: stopping criterion

Let $[S_1\ S_2] = [AR_1\ AR_2]$, $Q = QR(S_1)$

Absolute criterion:

$$\|(I - QQ^*)A\|_F \approx \frac{1}{\sqrt{d}}\|(I - QQ^*)S_2\|_F \leq \varepsilon_a$$

Relative criterion:

$$\frac{\|(I - QQ^*)A\|_F}{\|A\|_F} \approx \frac{\|(I - QQ^*)S_2\|_F}{\|S_2\|_F} \leq \varepsilon_r$$

Cost: one reduction to compute norms of the sample vectors.

Estimation accuracy: exponential decaying tail probabilities

Define random variables:

$$X = ||Ax||_2^2 \sim \sigma_1^2 \xi_1^2 + \cdots + \sigma_r^2 \xi_r^2, \quad \overline{X}_d \sim \frac{1}{d}\left[X_1 + \cdots + X_d\right],$$

X_i are independent realizations of X, $\mathbb{E}(X) = \mathbb{E}\left(\overline{X}_d\right) = ||A||_F^2$.

Theorem [C. Gorman]

$$\mathbb{P}\left[\overline{X}_d \geq ||A||_F^2\,\tau\right] \leq \exp\left(-\frac{d\tau}{2}\right) ||A||_F^{dr} \prod_{k=1}^{r} \left(A'_k\right)^{-d} \qquad \tau > 1$$

$$\mathbb{P}\left[\overline{X}_d \leq ||A||_F^2\,\tau\right] \leq \exp\left(-\frac{d\tau}{2}\right) ||A||_F^{dr} \prod_{k=1}^{r} \left(A''_k\right)^{-d} \qquad \tau \in [0,1)$$

where, $(A'_k)^2 = ||A||_F^2 - \sigma_k^2, \quad (A''_k)^2 = ||A||_F^2 + \sigma_k^2$.

This shows the probability tails of X and $\overline{X}_d$ decay exponentially away from the mean $||A||_F^2$.

Adaptive sampling example: decay singular value

$A = \alpha I + UDV^*$, U, V $rank = 120$, $D_{k,k} = 2^{-24(k-1)/r}$

$\varepsilon_r \backslash \varepsilon_a$	1e-2	1e-4	1e-6	1e-8	1e-10	1e-12	1e-14
1e-1	24; 64	24; 64	24; 64	24; 64	24; 64	24; 64	24; 64
1e-2	42; 80	42; 80	42; 80	42; 80	42; 80	42; 80	42; 80
1e-3	59; 96	59; 96	59; 96	59; 96	59; 96	59; 96	59; 96
1e-4	77; 112	77; 112	77; 112	77; 112	77; 112	77; 112	77; 112
1e-5	94; 128	94; 128	94; 128	94; 128	94; 128	94; 128	94; 128
1e-6	111; 128	111; 128	111; 128	111; 128	111; 128	111; 128	111; 128
1e-7	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128
1e-8	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128

Table 10: Size: 100000; Alpha: 100000; Rank: 120; Decay-value: 24; d-start: 16; d-add: 16. These numbers are "(Computed HSS Rank); (Random Samples Used)". Here, we are using $\text{fl}\left[\left(I - Q_1 Q_1^*\right) S_2\right] = \left(I - Q_1 Q_1^*\right)^2 S_2$, with the products computed intelligently.

Adaptive sampling: non-decay singular values

$A = \alpha I + UDV^*$, U, V $rank = 120$, $D_{k,k} = 1$

$\varepsilon_r \backslash \varepsilon_a$	1e-2	1e-4	1e-6	1e-8	1e-10	1e-12	1e-14
1e-1	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128
1e-2	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128
1e-3	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128
1e-4	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128
1e-5	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128
1e-6	120; 128	120; 128	120; 128	120; 128	120; 768	120; 768	120; 768
1e-7	120; 128	120; 128	120; 128	120; 128	120; 768	120; 768	120; 768
1e-8	120; 128	120; 128	120; 128	120; 128	120; 768	120; 768	120; 768

Table 8: Size: 100000; Alpha: 100000; Rank: 120; Decay-value: 0; d-start: 16; d-add: 16. These numbers are "(Computed HSS Rank); (Random Samples Used)". Here, we are using $\mathrm{fl}\left[\left(I - Q_1 Q_1^*\right) S_2\right] = \left(I - Q_1 Q_1^*\right)^2 S_2$, with the products computed intelligently.

Adaptivity cost is small

$A = \alpha I + UDV^*$, U, V $rank = 1200$, $D_{k,k} = 2^{-24(k-1)/r}$, $N = 60,000$, $P = 1024$, $\varepsilon_a = \varepsilon_r = 10^{-14}$

		"Known-rank"	Adaptive	"Hard-restart"
$d_0 = 128$ $\Delta d = 64$	Compr. time	36.5	37.2	100.3
	HSS-rank	1162	1267	1165
	Num. adapt.	0	17	4

Mitigate dense sampling cost

- HSS compression cost = sampling cost + $O(r^2 N)$.
- Sampling cost:
  - Traditional matvec: $O(rN^2)$
  - FFT: $O(rN \log N)$ (e.g., Toeplitz)
  - FMM: $O(rN)$

Mitigate dense sampling cost

- Kernel Rdge Regression for classification [IPDPS ParLearning Workshop 2018]
  - Kernel matrix: $K_{ij} = exp(-\frac{1}{2}\frac{||x_i - x_j||^2}{h^2})$
  - Need to solve $w := (K + \lambda I)^{-1} y$; **a few digits suffice** $\rightarrow$ use HSS
- **Use $\mathcal{H}$-matrix to perform sampling for HSS construction.**
  UCI dataset; parallel runtime on Intel Haswell at NERSC.

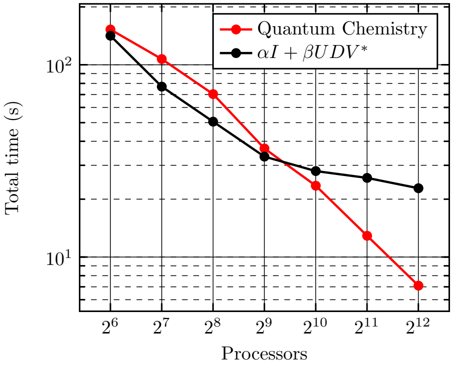
SUSY: 4.5M, dimension=8; COVTYPE: 0.5M, dimension=54

	SUSY		COVTYPE	
Cores	32	512	32	512
$\mathcal{H}$ construction	173.7	18.3	36.5	32.2
HSS construction	3344.4	726.7	432.3	239.7
$\rightarrow$ Sampling	2993.5	662.1	305.2	178.4
$\rightarrow$ Other	350.9	64.6	127.1	61.3
ULV Factorization	14.2	3.3	26.5	4.6
Solve	0.5	0.3	0.5	0.4

Dense scalability

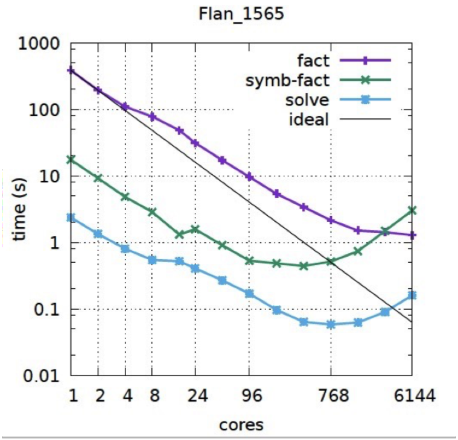
P = 4096, Cray XC40 (Cori at NERSC)

- quantum chemistry (Toeplitz): $a_{i,i} = \frac{\pi^2}{6}$ and $a_{i,j} = \frac{(-1)^{i-j}}{(i-j)^2 d^2}$
- $A = I + UDV^*$, U, V, $r = 500$, $D_{k,k} = 2^{-24(k-1)/r}$, $N = 500,000$



Sparse scalability

Matrix from SuiteSparse collection:
Flan_1565 (N = 1,564,794, NNZ = 114,165,372)



- Flat MPI on nodes with 2 12-core Intel Ivy Bridge (NERSC Edison)

STRUMPACK – STRUctured Matrices PACKage

http://portal.nersc.gov/project/sparse/strumpack/

- Two components:
  - Dense – applicable to Toeplitz, Cauchy, BEM, integral equations, etc.
  - Sparse – aim at matrices discretized from PDEs.
- Open source on Github, BSD license.
- C++, hybrid MPI + OpenMP implementation
- Real & complex datatypes, single & double precision (via template), and 64-bit indexing.
- Input interfaces:
  - Dense matrix in standard format.
  - Matrix-free, with query function to return selected entries.
  - Sparse matrix in CSR format.
- Can take user input: cluster tree & block partitioning.
- Functions:
  - HSS construction, HSS-vector product, ULV factorization, Solution.
- Available from PETSc, MFEM.
- **Extensible to include other data-sparse formats.**

Summary

- Sampling is handy, but still needs more mathematical insight to make it robust and efficient.
- Preconditioner appears to be robust [IPDPS 2017]
  - Works well for problems where AMG has slow convergence, e.g., indefinite problems.
  - More parallelizable than ILU, fewer parameters to tune.
- More research
  - Dynamic load balancing.
  - Communication analysis for sparse solvers.
  - Rank analysis of different application problems.
  - Good ordering and hierarchical clustering / partitioning to reduce off-diagonal rank.
  - Not all problems compress well in HSS, look into other formats.

THANK YOU !

HSS matrix – ULV factorization

ULV-like factored form (U and V^* unitary, L triangular)

$$
\Gamma_{1;b\leftrightarrow 2;t}
\begin{bmatrix} I & & & \\ & \Omega_1 & & \\ & & I & \\ & & & \Omega_2 \end{bmatrix}
\begin{bmatrix} \Gamma_{3;b\leftrightarrow 4;t} & \\ & \Gamma_{5;b\leftrightarrow 6;t} \end{bmatrix}
\begin{bmatrix} \Omega_3 & & & \\ & \Omega_4 & & \\ & & \Omega_5 & \\ & & & \Omega_6 \end{bmatrix}
A
\begin{bmatrix} Q_3^* & & & \\ & Q_4^* & & \\ & & Q_5^* & \\ & & & Q_6^* \end{bmatrix}
\begin{bmatrix} \Gamma^T_{3;b\leftrightarrow 4;t} & \\ & \Gamma^T_{5;b\leftrightarrow 6;t} \end{bmatrix}
\begin{bmatrix} I & & & \\ & Q_1^* & & \\ & & I & \\ & & & Q_2^* \end{bmatrix}
\Gamma^T_{1;b\leftrightarrow 2;t}
$$

$$
= \left[\begin{array}{c|c|c|c|c|c|c}
L_3 & & & & & & \\
0 & L_4 & & & & & \\
(\Omega_1 L_{4,3})_t & (\Omega_1 L_{3,4})_t & L_1 & & & & \\
\multicolumn{3}{c|}{} & L_5 & & & \\
\multicolumn{3}{c|}{0} & 0 & L_6 & & \\
\multicolumn{3}{c|}{} & (\Omega_2 L_{6,5})_t & (\Omega_2 L_{5,6})_t & L_2 & \\
(\Omega_1 L_{4,3})_b & (\Omega_1 L_{3,4})_b & W_{1;b} Q^*_{1;t} & \multicolumn{3}{c|}{B_{1,2} V_2^* \begin{bmatrix} V_5^* Q^*_{5;t} & & V_5^* Q^*_{5;b} & \\ & V_6^* Q^*_{6;t} & & V_6^* Q^*_{6;b} \end{bmatrix} \begin{bmatrix} I & \\ & Q_2^* \end{bmatrix}} & \\
\multicolumn{3}{c|}{B_{2,1} V_1^* \begin{bmatrix} V_3^* Q^*_{3;t} & & V_3^* Q^*_{3;b} & \\ & V_4^* Q^*_{4;t} & & V_4^* Q^*_{4;b} \end{bmatrix} \begin{bmatrix} I & \\ & Q_1^* \end{bmatrix}} & (\Omega_2 L_{6,5})_b & (\Omega_2 L_{5,6})_b & W_{2;b} Q^*_{2;t} & D_0
\end{array}\right]
$$

Themes

Many research areas for exascale computing: **https://exascaleproject.org**

- Algorithms with lower arithmetic & communication complexity. Multilevel algorithms:
  - Multigrid
  - Fast Multipole Method (FMM)
  - **Hierarchical matrices – algebraic generalization of FMM, applicable to broader classes of problems**

Themes

Many research areas for exascale computing: **https://exascaleproject.org**

- Algorithms with lower arithmetic & communication complexity. Multilevel algorithms:
  - Multigrid
  - Fast Multipole Method (FMM)
  - **Hierarchical matrices – algebraic generalization of FMM, applicable to broader classes of problems**
- Parallel algorithms and codes for machines with million-way parallelism, hierarchical organization.
  - Distributed memory
  - **Manycore nodes: 100s of lightweight cores, accelerators, co-processors**

Why factorization?

- Target problems:
  - indefinite, ill-conditioned, nonsymmetric (e.g. those from multiphysics, multiscale simulations)
- Where can be used?
  - Stand-alone solver.
  - Good for multiple right-hand sides.
  - Precondition Krylov solvers.
  - Coarse-grid solver in multigrid. (e.g., Hypre)
  - In nonlinear solver. (e.g., SUNDIALS)
  - Solving interior eigenvalue prolems.
  - ...
- Error analysis:
  - Componentwise error bounds (**Guaranteed solution accuracy**).
  - Condition number estimation.

Arithmetic complexities – dense HSS

Let r = HSS rank, i.e., maximum rank found during the different compression steps.

Compression

- Without RS: $O(r\,N^2)$.
- With RS: sampling cost (dominant) + $O(r^2 N)$

  sampling cost:

  - Classical matvec: $O(r\,N^2)$.
  - FFT (e.g., Toeplitz matrix): $O(r\,N \log N)$.
  - FMM: $O(r\,N)$.

ULV factorization and solve: $O(r\,N)$