EXASCALE COMPUTING PROJECT

STRUMPACK: Rank Structured Solvers for Dense and Sparse Linear Systems

Xiaoye Sherry Li, Pieter Ghysels, Yang Liu
Lawrence Berkeley National Laboratory
Scalable Solvers Group
{xsli, pghysels, liuyangzhuan}@lbl.gov

August 23, 2022

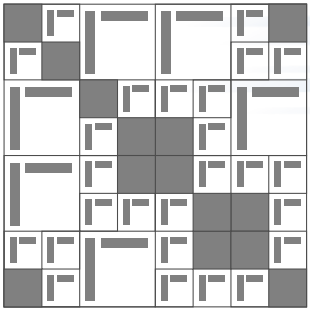
Hierarchical Matrix Approximation

$\mathcal{H}$-matrix representation [1]

- Data-sparse, rank-structured, compressed

Hierarchical/recursive 2×2 matrix blocking, with blocks either:

- Low-rank: $A_{IJ} \approx UV^\top$
- Hierarchical
- Dense (at lowest level)

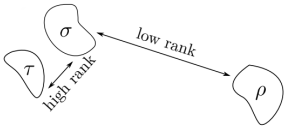


Use cases:

- Boundary element method for integral equations
- Cauchy, Toeplitz, kernel, covariance, . . . matrices
- Fast matrix-vector multiplication
- $\mathcal{H}$-LU decomposition
- Preconditioning

Hackbusch, W., 1999. *A sparse matrix arithmetic based on $\mathcal{H}$-matrices. part I: Introduction to $\mathcal{H}$-matrices.* Computing, 62(2), pp.89-108.

Admissibility Condition

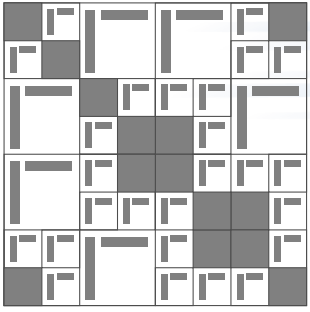


- Row cluster σ
- Column cluster τ
- $\sigma \times \tau$ is compressible $\Leftrightarrow$

$$\frac{\max(\text{diam}(\sigma), \text{diam}(\tau))}{\text{dist}(\tau, \sigma)} \leq \eta$$

  - $\text{diam}(\sigma)$: diameter of physical domain corresponding to σ
  - $\text{dist}(\sigma, \tau)$: distance between σ and τ

- Weaker interaction between clusters leads to smaller ranks
- Intuitively larger distance, greater separation, leads to weaker interaction
- Need to cluster and order degrees of freedom to reduce ranks



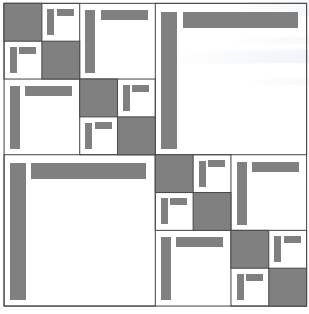
Hackbusch, W., 1999. *A sparse matrix arithmetic based on $\mathcal{H}$-matrices. part i: Introduction to $\mathcal{H}$-matrices.* Computing, 62(2), pp.89-108.

HODLR: Hierarchically Off-Diagonal Low Rank

- Weak admissibility

$$\sigma \times \tau \text{ is compressible } \Leftrightarrow \sigma \neq \tau$$

Every off-diagonal block is compressed as low-rank, even interaction between neighboring clusters (no separation)



Compared to more general $\mathcal{H}$-matrix

- Simpler data-structures: same row and column cluster tree
- More scalable parallel implementation
- Good for 1D geometries, e.g., boundary of a 2D region discretized using BEM or 1D separator
- Larger ranks

HSS: Hierarchically Semi Seperable

- Weak admissibility
- Off-diagonal blocks

$$A_{\sigma,\tau} \approx U_\sigma B_{\sigma,\tau} V_\tau^\top$$

- Nested bases

$$U_\sigma = \begin{bmatrix} U_{\nu_1} & 0 \\ 0 & U_{\nu_2} \end{bmatrix} \hat{U}_\sigma$$

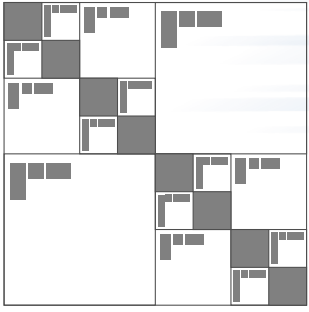
with ν_1 and ν_2 children of σ in the cluster tree.

- At lowest level

$$U_\sigma \equiv \hat{U}_\sigma$$

- Store only $\hat{U}_\sigma$, smaller than U_σ
- Complexity $\mathcal{O}(N) \leftrightarrow \mathcal{O}(N \log N)$ for HODLR
- HSS is special case of $\mathcal{H}^2$: $\mathcal{H}$ with nested bases

$$\begin{bmatrix} D_0 & U_0 B_{0,1} V_1^* & & U_2 B_{2,5} V_5^* \\ U_1 B_{1,0} V_0^* & D_1 & & \\ & & D_3 & U_3 B_{3,4} V_4^* \\ U_5 B_{5,2} V_2^* & & U_4 B_{4,3} V_3^* & D_4 \end{bmatrix}$$



HSS: Hierarchically Semi Seperable

- Weak admissibility
- Off-diagonal blocks

$$A_{\sigma,\tau} \approx U_\sigma B_{\sigma,\tau} V_\tau^\top$$

- Nested bases

$$U_\sigma = \begin{bmatrix} U_{\nu_1} & 0 \\ 0 & U_{\nu_2} \end{bmatrix} \hat{U}_\sigma$$

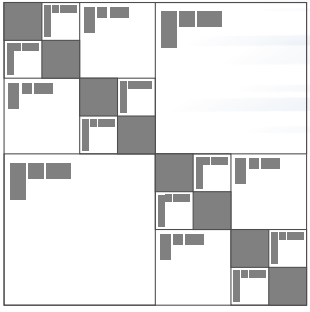
with ν_1 and ν_2 children of σ in the cluster tree.

- At lowest level

$$U_\sigma \equiv \hat{U}_\sigma$$

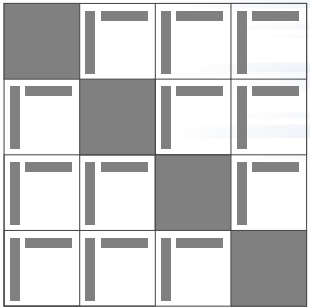
- Store only $\hat{U}_\sigma$, smaller than U_σ
- Complexity $\mathcal{O}(N) \leftrightarrow \mathcal{O}(N \log N)$ for HODLR
- HSS is special case of $\mathcal{H}^2$: $\mathcal{H}$ with nested bases

$$\begin{bmatrix} D_0 & U_0 B_{0,1} V_1^* & \begin{bmatrix} U_0 & 0 \\ 0 & U_1 \end{bmatrix} \hat{U}_2 B_{2,5} \hat{V}_5^* \begin{bmatrix} V_3^* & 0 \\ 0 & V_4^* \end{bmatrix} \\ U_1 B_{1,0} V_0^* & D_1 & \\ \begin{bmatrix} U_3 & 0 \\ 0 & U_4 \end{bmatrix} \hat{U}_5 B_{5,2} \hat{V}_2^* \begin{bmatrix} V_0^* & 0 \\ 0 & V_1^* \end{bmatrix} & U_4 B_{4,3} V_3^* & D_3 \quad U_3 B_{3,4} V_4^* \\ & & D_4 \end{bmatrix}$$



BLR: Block Low Rank [1, 2]

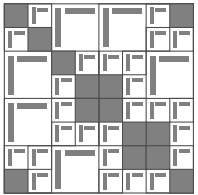
- Flat partitioning (non-hierarchical)
- Weak or strong admissibility
- Larger asymptotic complexity than $\mathcal{H}$, HSS, ...
- Works well in practice



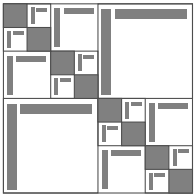
Mary, T. (2017). *Block Low-Rank multifrontal solvers: complexity, performance, and scalability.* (Doctoral dissertation).

Amestoy, Patrick, et al. (2015). *Improving multifrontal methods by means of block low-rank representations.* SISC 37.3 : A1451-A1474.

Data-Sparse Matrix Representation Overview

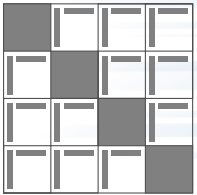


$\mathcal{H}$



HODLR

HSS



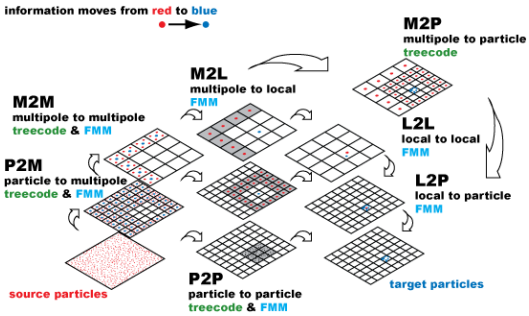
BLR

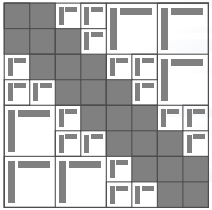
- Partitioning: **hierarchical** ($\mathcal{H}$, HODLR, HSS) or **flat** (BLR)
- Admissibility: **weak** (HODLR, HSS) or **strong** ($\mathcal{H}$, $\mathcal{H}^2$)
- Bases: **nested** (HSS, $\mathcal{H}^2$) or **not nested** ($\mathcal{H}$, HODLR, BLR)

Fast Multipole Method [1]

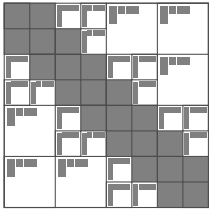
Particle methods like Barnes-Hut and FMM can be interpreted algebraically using hierarchical matrix algebra

- Barnes-Hut $\mathcal{O}(N \log N)$
- Fast Multipole Method $\mathcal{O}(N)$





Barnes-Hut

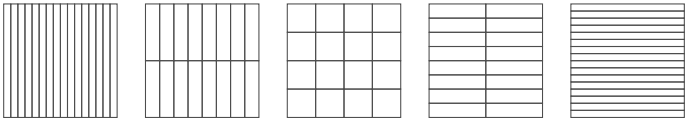


FMM

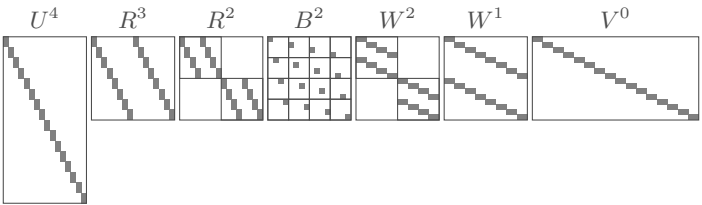
Greengard, L., and Rokhlin, V. *A fast algorithm for particle simulations.* Journal of computational physics 73.2 (1987): 325-348.

Butterfly Decomposition [1]

Complementary low rank property: sub-blocks of size $\mathcal{O}(N)$ are low rank:



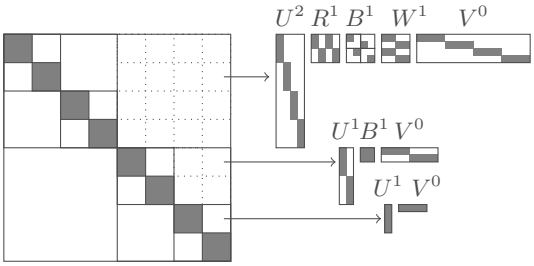
Multiplicative decomposition:



- Multilevel generalization of low rank decomposition
- Based on FFT ideas, motivated by high-frequency problems

Michielssen, E., and Boag, A. *Multilevel evaluation of electromagnetic fields for the rapid solution of scattering problems.* Microwave and Optical Technology Letters 7.17 (1994): 790-795.

HODBF: Hierarchically Off-Diagonal Butterfly



- HODLR but with low rank replaced by Butterfly decomposition
- Reduces ranks of large off-diagonal blocks

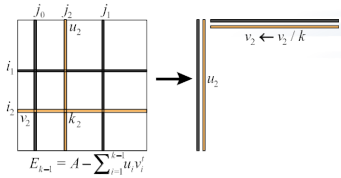
Low Rank Approximation Techniques

Traditional approaches need entire matrix

- Truncated Singular Value Decomposition (TSVD): $A \approx U\Sigma V^T$
  - Optimal, but expensive
- Column Pivoted QR: $AP \approx QR$
  - Less accurate than TSVD, but cheaper

Adaptive Cross Approximation

- No need to compute every element of the matrix
- Requires certain assumptions on input matrix
- Left-looking LU with rook pivoting



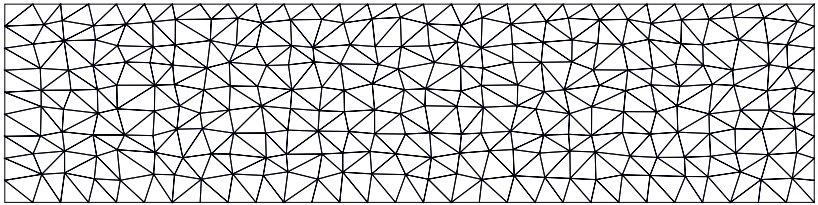
Randomized algorithms [1]

- Fast matrix-vector product: $S = A\Omega$
  Reduce dimension of A by random projection with Ω
- E.g., operator is sparse or rank structured, or the product of sparse and rank structured

Halko, N., Martinsson, P.G., Tropp, J.A. (2011). *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions.* SIAM Review, 53(2), 217-288.

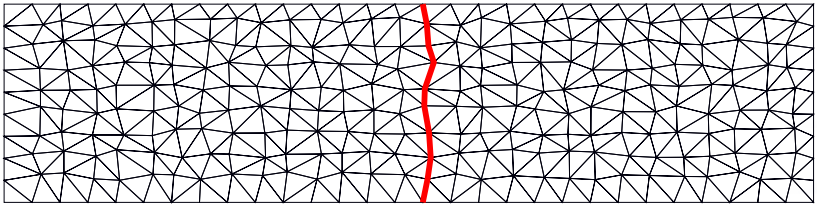
Approximate Multifrontal Factorization

Multifrontal Sparse LU Factorization [Duff & Reid '83]



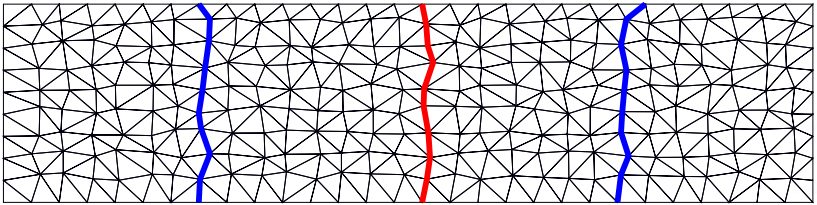
- Nested-dissection reordering
  - Separator/supernodal tree
  - (Par)Metis/(PT)Scotch graph partitioners
- For every separator
  - Dense frontal matrix
  - Partial LU factorization
  - Schur complement update
  - Extend-add: parent nodes "sum" Schur complements from the children
- Multifrontal solve
  - Forward and backward solve
  - Two traversals of the separator tree

Multifrontal Sparse LU Factorization [Duff & Reid '83]

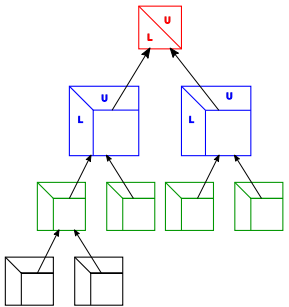


- Nested-dissection reordering
  - Separator/supernodal tree
  - (Par)Metis/(PT)Scotch graph partitioners
- For every separator
  - Dense frontal matrix
  - Partial LU factorization
  - Schur complement update
  - Extend-add: parent nodes "sum" Schur complements from the children
- Multifrontal solve
  - Forward and backward solve
  - Two traversals of the separator tree

Multifrontal Sparse LU Factorization [Duff & Reid '83]

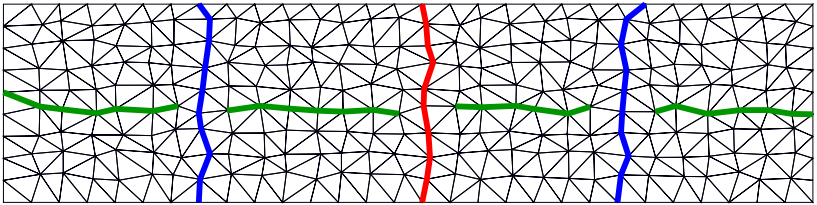


- Nested-dissection reordering
  - Separator/supernodal tree
  - (Par)Metis/(PT)Scotch graph partitioners
- For every separator
  - Dense frontal matrix
  - Partial LU factorization
  - Schur complement update
  - Extend-add: parent nodes "sum" Schur complements from the children
- Multifrontal solve
  - Forward and backward solve
  - Two traversals of the separator tree

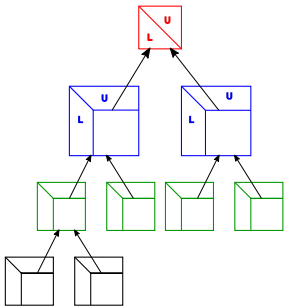


supernodal/separator tree

Multifrontal Sparse LU Factorization [Duff & Reid '83]



- Nested-dissection reordering
  - Separator/supernodal tree
  - (Par)Metis/(PT)Scotch graph partitioners
- For every separator
  - Dense frontal matrix
  - Partial LU factorization
  - Schur complement update
  - Extend-add: parent nodes "sum" Schur complements from the children
- Multifrontal solve
  - Forward and backward solve
  - Two traversals of the separator tree



supernodal/separator tree

Multifrontal LU Factorization Algorithm

Data: A: sparse matrix, b: right-hand side vector
Result: $x = A^{-1}b$

$A \leftarrow PAP^T$ — *# Fill-reducing ordering, nested-dissection*
$I^{\mathsf{s}}_\tau \leftarrow \{\dots\},\ I^{\mathsf{u}}_\tau \leftarrow \{\dots\}$ — *# Symbolic factorization*
foreach *node* τ *in the sep-tree, bottom-up* **do**
  $F_\tau \leftarrow \begin{bmatrix} A(I^{\mathsf{s}}_\tau, I^{\mathsf{s}}_\tau) & A(I^{\mathsf{s}}_\tau, I^{\mathsf{u}}_\tau) \\ A(I^{\mathsf{u}}_\tau, I^{\mathsf{s}}_\tau) & 0 \end{bmatrix} \leftarrow F_{\nu_1;22} \leftarrow F_{\nu_2;22}$ — *# Extend-add*
  $P_\tau L_\tau U_\tau \leftarrow LU(F_{11})$ — *# Partial factorization of* F_τ
  $F_{12} \leftarrow L_\tau^{-1} P^T F_{12}$
  $F_{21} \leftarrow F_{21} U_\tau^{-1}$ — *# Permutation and triangular solve*
  $F_{\tau;22} \leftarrow F_{22} - F_{21} F_{12}$ — *# Compute Schur complement*
end
$b \leftarrow P^T D_r b$
Forward and backward solve
$x \leftarrow D_c P x$

Multifrontal LU Factorization Algorithm

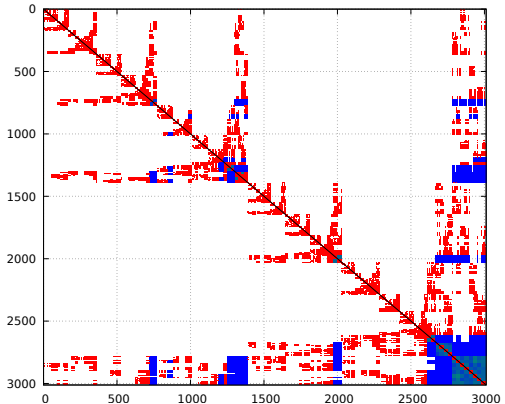
Data: A: sparse matrix, b: right-hand side vector
Result: $x = A^{-1}b$

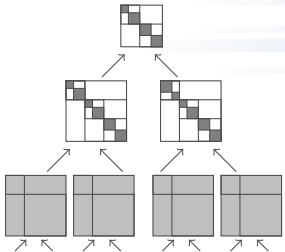
$A \leftarrow PAP^T$ — *# Fill-reducing ordering, nested-dissection*
$I^{\mathsf{s}}_\tau \leftarrow \{\dots\},\ I^{\mathsf{u}}_\tau \leftarrow \{\dots\}$ — *# Symbolic factorization*
foreach *node* τ *in the sep-tree, bottom-up* **do**
  $F_\tau \leftarrow \begin{bmatrix} A(I^{\mathsf{s}}_\tau, I^{\mathsf{s}}_\tau) & A(I^{\mathsf{s}}_\tau, I^{\mathsf{u}}_\tau) \\ A(I^{\mathsf{u}}_\tau, I^{\mathsf{s}}_\tau) & 0 \end{bmatrix} \leftarrow F_{\nu_1;22} \leftarrow F_{\nu_2;22}$ — *# Extend-add*
  $P_\tau L_\tau U_\tau \leftarrow LU(F_{11})$ — *# Partial factorization of* F_τ
  $F_{12} \leftarrow L_\tau^{-1} P^T F_{12}$
  $F_{21} \leftarrow F_{21} U_\tau^{-1}$ — *# Permutation and triangular solve*
  $F_{\tau;22} \leftarrow F_{22} - F_{21} F_{12}$ — *# Compute Schur complement*
end
$b \leftarrow P^T D_r b$
Forward and backward solve
$x \leftarrow D_c P x$

Approximate F by a rank-structured matrix

Sparse Multifrontal Solver/Preconditioner with Rank-Structured Approximations

L and U factors, after nested-dissection ordering,
compressed blocks in blue



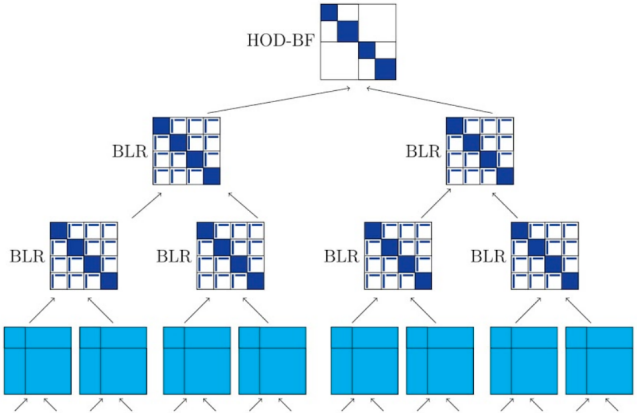


Only apply rank structured compression to largest fronts (dense sub-blocks), keep the rest as regular dense

Combining Block Low Rank and Hierarchically Off-Diagonal Butterfly

Rank-structured compression of largest dense blocks in the multifrontal/assembly tree

- Largest: HOD-BF
- Medium: BLR
- Smaller: dense

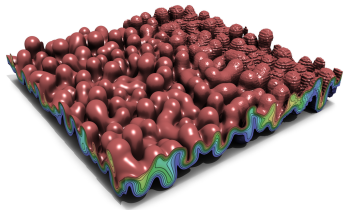


ZFP Compression

- Lossy compression for floating-point data
- Applied to sparse triangular factors
- Drastic memory reduction even for numerically challenging problems, enabling solution of larger problems with manageable errors and with a small overhead during factorization
- Larger overhead during solve needs to be addressed
- (De)compression can be offloaded to GPU

3D 60^3 complex Helhmholtz problem

Precision	Factor (s)	Mem (%)	GMRES (s)	Its
exact	75.09	100	0.70	1
8 bitplanes	79.28	7.97	170.52	20
16	79.75	20.76	54.03	3
24	84.97	33.59	57.13	1
32	86.08	46.41	44.70	1
48	84.21	72.06	61.85	1
64	87.22	97.71	79.88	1

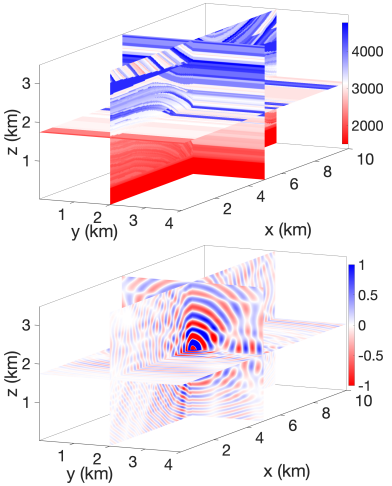


P. Lindstrom, IEEE TVCG, 2014

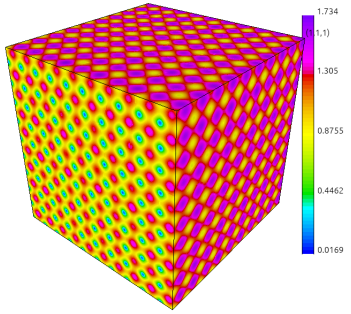
`https://github.com/LLNL/zfp`

High Frequency Helmholtz and Maxwell

Regular $k^3 = N$ grid, fixed number of discretization points per wavelength



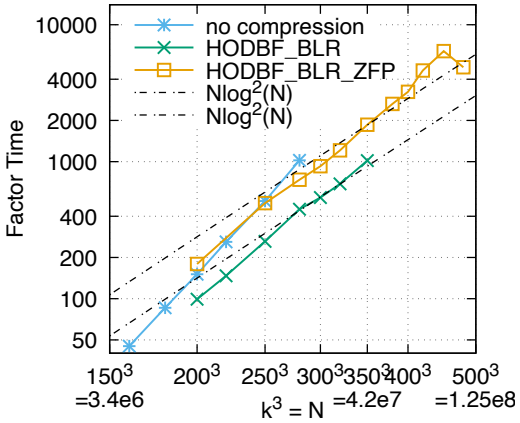
Marmousi2 geophyical elastic dataset

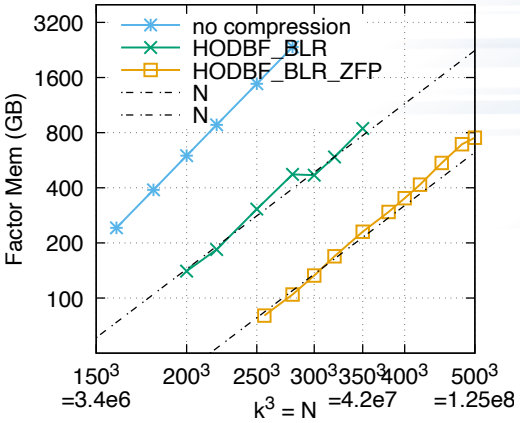


Indefinite Maxwell, using MFEM

STRUMPACK Preconditioners for High Frequency Helmholtz and Maxwell

Sparse multifrontal solver with hybrid ZFP, BLR and HODBF compression





- Highly oscillatory problems are hard for iterative solvers
- Typically solved with sparse direct solvers, but scale as $\mathcal{O}(N^2)$

Rank Structured Preconditioning

```
export OMP_NUM_THREADS=16
./testMMdouble torso3.mtx --sp_compression NONE --sp_disable_gpu
./testMMdouble torso3.mtx --sp_compression BLR  --blr_rel_tol 1e-2
```

matrix	torso3		Geo_1438		nlpkkt80	
compression	NONE	BLR	NONE	BLR	NONE	BLR
factor time (sec)	2.17	1.10	147.81	50.10	140.21	38.19
factor memory (MB)	1,855	1,281	38,523	17,999	30,199	10,128
compression	100%	70.1%	100%	46.7%	100%	33.5%
peak memory (MB)	5,505	3,694	109,741	45,603	89,171	40,775
solve time (sec)	0.09	0.14	1.80	10.69	1.46	5.54
GMRES its	1	2	1	18	1	15

Table: Multifrontal solver with block low rank compression for several SuiteSparse linear systems, using compression tolerance $\varepsilon = 10^{-2}$. GMRES relative tolerance is 10^{-6}.

GPU Performance for SuiteSparse Problems

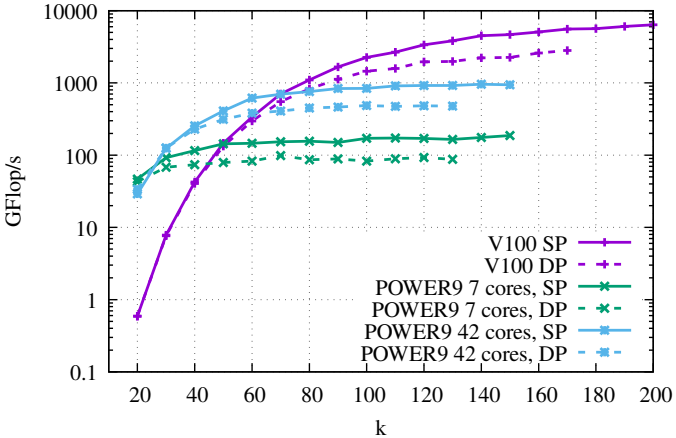
```
OMP_NUM_THREADS=16 ./testMMdouble torso3.mtx
OMP_NUM_THREADS=16 ./testMMdouble torso3.mtx --sp_disable_gpu
```

			time (seconds)		GFlop/s	
matrix	N	nnz	GPU A100	16 CPU	GPU A100	16 CPU
torso3	259,156	4,429,042	0.81	2.18	505.4	174.7
Geo_1438	1,437,960	60,236,322	9.54	147.72	3618.7	233.7
nlpkkt80	1,062,400	28,192,672	7.97	133.58	4056.6	242.0

Table: Numerical factorization time (seconds) and performance in terms of floating point operations per second.

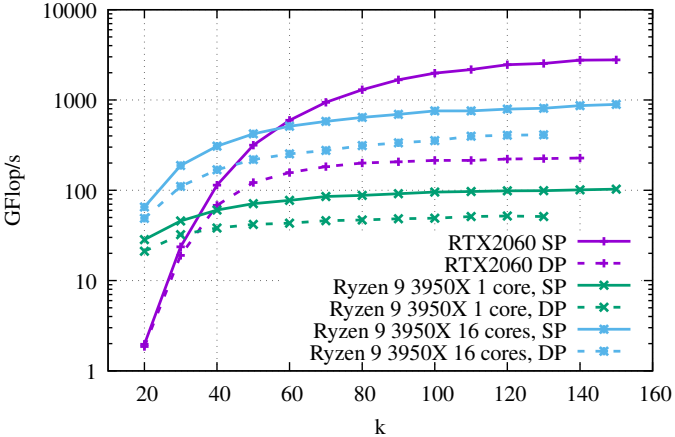
Regular 3D k^3 mesh, 7-point stencil

SUMMIT@ORNL, NVIDIA V100 vs POWER9 CPU (7 or 42 physical cores)



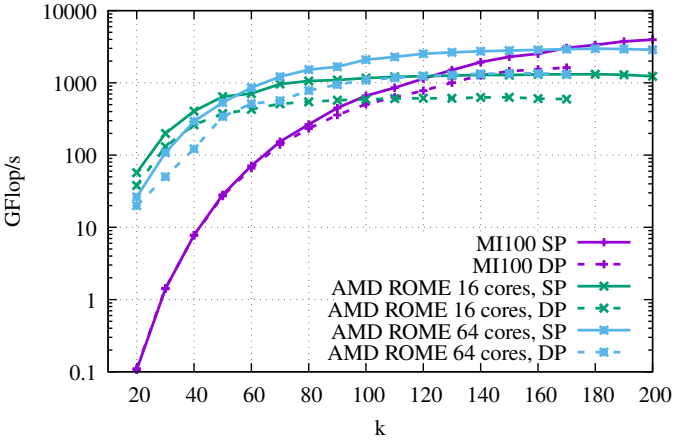
Regular 3D k^3 mesh, 7-point stencil

RTX2060 SUPER vs AMD Ryzen 9 3950X (1 or 16 cores)



Regular 3D k^3 mesh, 7-point stencil

AMD MI100 GPU vs AMD ROME (16 or 64 cores)



HIP/ROCm, rocBLAS, rocSOLVER

Multi-GPU Results

3D visco-acoustic wave propagation

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

	100^3		150^3		200^3		250^3	
	P9	V100	P9	V100	P9	V100	P9	V100
SUMMIT nodes	1		2		4		8	
P9 cores	42		84		168		336	
# of V100s	-	6	-	12	-	24	-	48
MPI ranks	42	6	84	12	168	24	336	48
OpenMP/rank	1	7	1	7	1	7	1	7
Fact. time (sec)	105.58	18.63	581.45	61.29	1612.67	157.67	OOM	266.96
Speedup	$\mathbf{5.7}\times$		$\mathbf{9.5}\times$		$\mathbf{10.2}\times$		-	
TFlop/s	0.51	2.86	1.06	10.09	2.17	22.14	-	50.18
Flops	$5.3 \cdot 10^{13}$		$6.2 \cdot 10^{14}$		$3.5 \cdot 10^{15}$		$1.3 \cdot 10^{16}$	
Fact. mem. (GB)	35.25		185.89		598.77		1482.38	

Software: ButterflyPACK

- Butterfly
- Hierarchically Off-Diagonal Low Rank (HODLR)
- Hierarchically Off-Diagonal Butterfly (HODBF)
- Hierarchical matrix format ($\mathcal{H}$)
  - Limited parallelism
- Fast compression, using randomization
- Fast multiplication, factorization & solve
- Fortran2008, MPI, OpenMP

`https://github.com/liuyangzhuan/ButterflyPACK`

Software: STRUMPACK

STRUctured Matrix PACKage

- Fully algebraic solvers/preconditioners
- Sparse direct solver (multifrontal LU factorization)
- Approximate sparse factorization preconditioner
- Dense
  - HSS: Hierarchically Semi-Separable
  - BLR: Block Low Rank
  - ButterflyPACK integration/interface:
    - Butterfly
    - HODLR
    - HODBF

- C++, MPI + OpenMP + CUDA/HIP/SYCL, real & complex, 32/64 bit integers
- BLAS, LAPACK, Metis
- Optional: MPI, ScaLAPACK, ParMETIS, (PT-)Scotch, cuBLAS/cuSOLVER, hipBLAS, Intel OneMKL, SLATE, ZFP

`https://github.com/pghysels/STRUMPACK`
`https://portal.nersc.gov/project/sparse/strumpack/master/`