A sparse multifrontal solver using hierarchically semi-separable frontal matrices

Pieter Ghysels

Lawrence Berkeley National Laboratory

Joint work with: Xiaoye S. Li (LBNL), Artem Napov (ULB), François-Henry Rouet (LBNL)

Sparse Days Meeting 2014 at CERFACS, Toulouse, June 5–6

Introduction

Consider solving

$$Ax = b \qquad \text{or} \qquad M^{-1}A = M^{-1}b$$

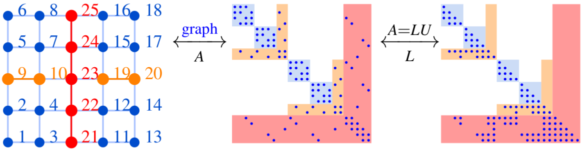
with a preconditioned iterative method (CG, GMRES, etc.)

- Fast convergence if good preconditioner $M \approx A$ is available
  1. Cheap to construct, store, apply, parallelize
  2. Good approximation of A

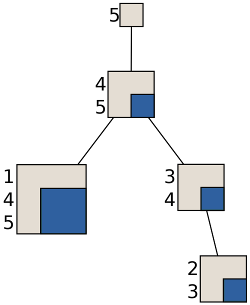
  Contradictory goals $\rightarrow$ trade-off

- Standard design strategy
  - Start with a direct factorization (like multifrontal LU)
  - Add approximations to make it cheaper (cf. 1)
    while (hopefully/provably) affecting little (2)
- Nonstandard approximation idea: use low-rank approximation

The multifrontal method [Duff & Reid '83]



- Nested dissection reordering defines an elimination tree.
- Bottom-up traversal of the e-tree.
- At each frontal matrix, partial factorization and computation of a contribution block (Schur complement).
- Parent nodes "sum" the contribution blocks of their children.



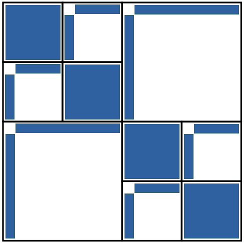
Elimination tree

Low-rank property

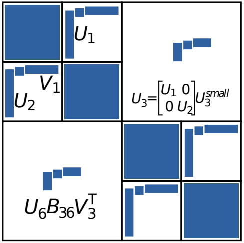
- In many applications, frontal matrices exhibit some low-rank blocks ("data sparsity").
  - What about the inverse or a factorization?
- Compression with SVD, Rank-Revealing QR,...
  - SVD: optimal but too expensive.
  - RRQR: QR with column pivoting.
- Exact compression or with a compression threshold ε.
- Recursively, diagonal blocks have low-rank subblocks too.
  - What to do with off-diagonal blocks?

Low-rank representations

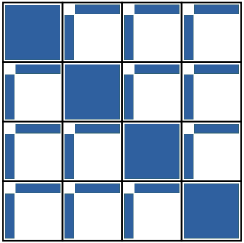
Most low-rank representations belong to the class of $\mathcal{H}$-matrices [Bebendof, Börm, Hackbush, Grasedyck,. . .]. Embedded in both dense and sparse solvers:



Hierarchically Off-Diag. Low Rank (HODLR) [Ambikasaran, Cecka, Darve. . .]



Hierarchically Semiseparable (HSS) [Chandrasekaran, Dewilde, Gu, Li, Xia,. . .] Nested basis.



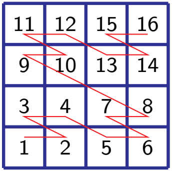
Block-low rank (BLR) [Amestoy et al.] Simple 2D partitioning. Recently in MUMPS.

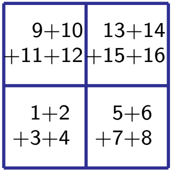
Also: $\mathcal{H}^2$ (includes HSS and FMM), SSS. . .
Choice: simple representations apply to broad classes of problems but provide less gains in memory/operations than specialized/complex ones.

Embedding low-rank techniques in a multifrontal solver

1. Choose which frontal matrices are compressed (size, level. . .).
2. Low-rankness: weak interactions between "distant" variables.
   $\Longrightarrow$ need suitable ordering/clustering of each frontal matrix.
   Fully-summed variables:
   - Geometric setting (3D grid): FS variables = 2D plane separator.
     - Need clusters with small diameters.
     - Hierarchical formats, merged clusters need small diameter too.

     Split domain into squares and order with Morton ordering.



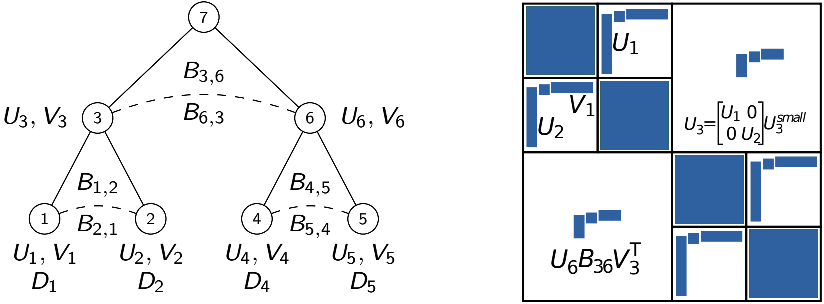


   - Algebraic: add some kind of halo to (complete) graph of FS variables and call a graph partitioner [Amestoy et al., Napov].

   Contribution blocks: same, or inherit from FS of ancestors.
3. Compress: part or whole front? Interleaving with factorization?

HSS representations

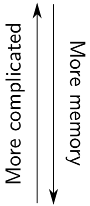
The structure is represented by a tree:



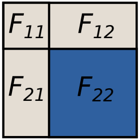
- Number of leaves depends on the problem (geometry) and number of processors to be used.
- Building the HSS structure and all the usual operations (multiplying. . .) consist of traversals of the tree.

Embedding HSS kernels in a multifrontal solver

HSS for frontal matrices:



Fully structured: HSS on the whole frontal matrix. No dense matrix.
Partial+: HSS on the whole frontal matrix. Dense frontal matrix.
Partially structured: HSS on the F_{11}, F_{12} and F_{21} parts only. Dense frontal matrix, dense $CB = F_{22} - F_{21}F_{11}^{-1}F_{12}$ in stack.



- Partially structured can do regular extend-add.
- In partially structured, HSS compression of dense matrix.
- After HSS compression, *ULV* factorization of F_{11} block.
  - Compared to classical *LU* in dense case.
- Low rank Schur complement update.

HSS compression via randomized sampling

[Martinsson '11, Xia '13]

HSS compression of a matrix A.
Ingredients:

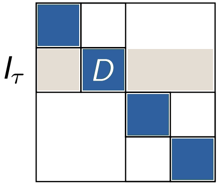
- R^r and R^c random matrices with d columns.
- $d = r + p$ with r estimated max rank; $p = 10$ in practice.
  p: probability of "good" approximation $\simeq (1 - p^{-p})$.
- $S^r = AR^r$ and $S^c = A^T R^c$ samples of matrix A.
  Can benefit from a fast matvec.
- Interpolative Decomposition: $A = A(:, J)\, X$.
  A is linear combination of selected columns of A.
- Two sided ID: ${S^c}^T = {S^c}^T(:, J^c)X^c$ and ${S^r}^T = {S^r}^T(:, J^r)X^r$,

$$A = X^c A(I^c, I^r){X^r}^T$$

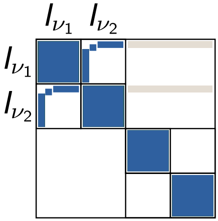
HSS compression via randomized sampling – 2

Algorithm (symmetric): from fine to coarse do

- Leaf node τ:
  1. Sample: $S_{loc} = S(I_\tau,:) - D\,R(I_\tau,:)$
  2. ID: $S_{loc} = U_\tau\, S_{loc}(J_\tau,:)$
  3. Update: $S_\tau = S_{loc}(J_\tau,:)$
     $R_\tau = U_\tau^T\, R(I_\tau,:)$
     $I_\tau = I_\tau(J_\tau,:)$



- Inner node τ with children ν_1, ν_2:
  1. Sample: $S_{loc} = \begin{bmatrix} S_{\nu_1} - A(I_{\nu_1}, I_{\nu_2})\, R_{\nu_2} \\ S_{\nu_2} - A(I_{\nu_2}, I_{\nu_1})\, R_{\nu_1} \end{bmatrix}$
  2. ID: $S_{loc} = U_\tau\, S_{loc}(J_\tau,:)$
  3. Update: $S_\tau = S_{loc}(J_\tau,:)$
     $R_\tau = U_\tau^T\, [R_{\nu_1}; R_{\nu_2}]$
     $I_\tau = [I_{\nu_1}\, I_{\nu_2}](J_\tau,:)$



HSS compression via randomized sampling – 3

- If $A \neq A^T$, do this for columns as well (simultaneously)
- Bases have special structure: $U_\tau = \Pi_\tau \begin{bmatrix} I \\ E_\tau \end{bmatrix}$
- Extract elements from frontal matrix:
  $D_\tau = A(I_\tau, I_\tau)$ and $B_{\nu_1,\nu_2} = A(I_{\nu_1}, I_{\nu_2})$
- Frontal matrix is combination of separator and HSS children
- Extracting element from HSS matrix requires traversing the HSS tree and multiplying basis matrices.
- Limiting number of tree traversals is crucial for performance.

Benefits:

- Extend-add operation is simplified: only on random vectors.
- Gains in complexity: $\mathcal{O}(r^2 N \log N)$ iso $\mathcal{O}(rN^2)$ for non-randomized algorithm. $\log N$ due to extracting elements from HSS matrix

Randomized sampling – extend-add

Assembly in regular multifrontal: $F_p = A_p \mathbin{\updownarrow\!\!\!\!\leftrightarrow} CB_{c_1} \mathbin{\updownarrow\!\!\!\!\leftrightarrow} CB_{c_2}$.
Sample:
$S_p = F_p R_p = (A_p \mathbin{\updownarrow\!\!\!\!\leftrightarrow} CB_{c_1} \mathbin{\updownarrow\!\!\!\!\leftrightarrow} CB_{c_2}) R_p = A_p R_p \updownarrow Y_{c_1} \updownarrow Y_{c_2}$

- $\updownarrow$ 1D extend-add (only along rows); much simpler
- Y_{c_1} and Y_{c_2} samples of CB of children.
- $R_p = R_{c_1} \updownarrow R_{c_2}$ (+random rows for missing indices).

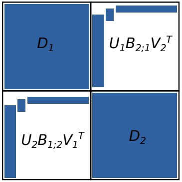
Stages:

- Build random vectors from random vectors of children.
- Build sample from samples of CB of children.
- Multiply separator part of frontal matrix with random vectors: $A_p R_p$
- Compression of F_p using S_p and R_p

HSS ULV factorization

- Exploit structure of U_τ (from ID) to introduce zero's

$$U_\tau = \Pi_\tau \begin{bmatrix} I \\ E_\tau \end{bmatrix}, \quad \Omega_\tau = \begin{bmatrix} -E_\tau & I \\ I & 0 \end{bmatrix} \Pi_\tau^T \quad \rightarrow \quad \Omega_\tau U_\tau = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

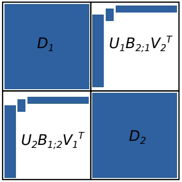


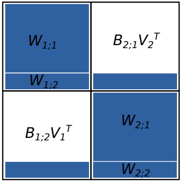
HSS ULV factorization

- Exploit structure of U_τ (from ID) to introduce zero's

$$U_\tau = \Pi_\tau \begin{bmatrix} I \\ E_\tau \end{bmatrix}, \quad \Omega_\tau = \begin{bmatrix} -E_\tau & I \\ I & 0 \end{bmatrix} \Pi_\tau^T \quad \rightarrow \quad \Omega_\tau U_\tau = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

- $$\begin{bmatrix} \Omega_1 & \\ & \Omega_2 \end{bmatrix} \begin{bmatrix} D_1 & U_1 B_{1,2} V_2^T \\ U_2 B_{2,1} V_1^T & D_2 \end{bmatrix} = \begin{bmatrix} W_1 & B_{1,2} V_2^T \\ B_{2,1} V_1^T & W_2 \end{bmatrix}$$





HSS ULV factorization

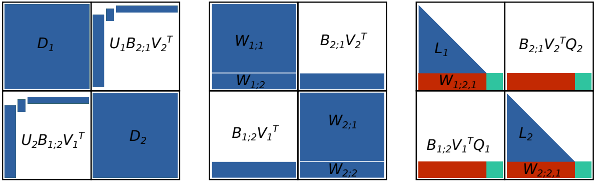
- Exploit structure of U_τ (from ID) to introduce zero's

$$U_\tau = \Pi_\tau \begin{bmatrix} I \\ E_\tau \end{bmatrix}, \quad \Omega_\tau = \begin{bmatrix} -E_\tau & I \\ I & 0 \end{bmatrix} \Pi_\tau^T \quad \to \quad \Omega_\tau U_\tau = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

- $$\begin{bmatrix} \Omega_1 & \\ & \Omega_2 \end{bmatrix} \begin{bmatrix} D_1 & U_1 B_{1,2} V_2^T \\ U_2 B_{2,1} V_1^T & D_2 \end{bmatrix} = \begin{bmatrix} W_1 & B_{1,2} V_2^T \\ B_{2,1} V_1^T & W_2 \end{bmatrix}$$

- Take (full) LQ decomposition

$$W_\tau = \begin{bmatrix} \begin{bmatrix} L_\tau & 0 \end{bmatrix} Q_\tau \\ W_{\tau;2} \end{bmatrix} \quad \to \quad \begin{bmatrix} L_1 & \\ [W_{1;2} Q_1^*] & [B_{1,2} V_2^T Q_2^*] \\ & L_2 \\ [B_{2,1} V_1^T Q_1^*] & [W_{2;2} Q_2^*] \end{bmatrix} \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix}$$



HSS ULV factorization

- Exploit structure of U_τ (from ID) to introduce zero's

$$U_\tau = \Pi_\tau \begin{bmatrix} I \\ E_\tau \end{bmatrix}, \quad \Omega_\tau = \begin{bmatrix} -E_\tau & I \\ I & 0 \end{bmatrix} \Pi_\tau^T \quad \rightarrow \quad \Omega_\tau U_\tau = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

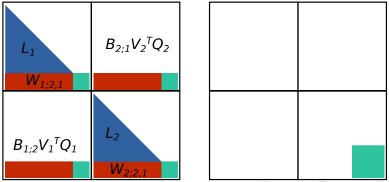
- $$\begin{bmatrix} \Omega_1 & \\ & \Omega_2 \end{bmatrix} \begin{bmatrix} D_1 & U_1 B_{1,2} V_2^T \\ U_2 B_{2,1} V_1^T & D_2 \end{bmatrix} = \begin{bmatrix} W_1 & B_{1,2} V_2^T \\ B_{2,1} V_1^T & W_2 \end{bmatrix}$$

- Take (full) LQ decomposition

$$W_\tau = \begin{bmatrix} \begin{bmatrix} L_\tau & 0 \end{bmatrix} Q_\tau \\ W_{\tau;2} \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} L_1 & \\ [W_{1;2} Q_1^*] & [B_{1,2} V_2^T Q_2^*] \\ & L_2 \\ [B_{2,1} V_1^T Q_1^*] & [W_{2;2} Q_2^*] \end{bmatrix} \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix}$$

- Rows for L_τ can be eliminated, others are passed to parent
- At root node:

  LU solve of reduced $\tilde{D}$



HSS ULV factorization

- Exploit structure of U_τ (from ID) to introduce zero's

$$U_\tau = \Pi_\tau \begin{bmatrix} I \\ E_\tau \end{bmatrix}, \quad \Omega_\tau = \begin{bmatrix} -E_\tau & I \\ I & 0 \end{bmatrix} \Pi_\tau^T \quad \rightarrow \quad \Omega_\tau U_\tau = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

- $$\begin{bmatrix} \Omega_1 & \\ & \Omega_2 \end{bmatrix} \begin{bmatrix} D_1 & U_1 B_{1,2} V_2^T \\ U_2 B_{2,1} V_1^T & D_2 \end{bmatrix} = \begin{bmatrix} W_1 & B_{1,2} V_2^T \\ B_{2,1} V_1^T & W_2 \end{bmatrix}$$

- Take (full) LQ decomposition

$$W_\tau = \begin{bmatrix} \begin{bmatrix} L_\tau & 0 \end{bmatrix} Q_\tau \\ W_{\tau;2} \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} L_1 & & \\ & [W_{1;2} Q_1^*] & [B_{1,2} V_2^T Q_2^*] \\ & & L_2 \\ [B_{2,1} V_1^T Q_1^*] & & [W_{2;2} Q_2^*] \end{bmatrix} \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix}$$

- Rows for L_τ can be eliminated, others are passed to parent
- At root node:
  LU solve of reduced $\tilde{D}$
- ULV-like: Ω_τ not orthonormal,
  forward/backward solve phases

Low rank Schur complement update

Schur Complement update

$$F_{22} - F_{21} F_{11}^{-1} F_{12} = F_{22} - \overbrace{U_q B_{qk} V_k^T}^{F_{21}} F_{11}^{-1} \overbrace{U_k B_{kq} V_q^T}^{F_{12}}$$

- F_{11}^{-1} via ULV solve

$$V_k^T F_{11}^{-1} U_k \rightarrow \mathcal{O}(rN^2)$$

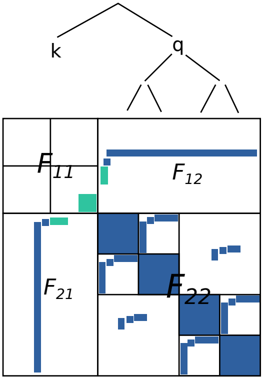
- $\tilde{D}_k$ is reduced HSS matrix $\mathcal{O}(r \times r)$

$$F_{22} - U_q B_{qk} (\tilde{V}_k^T \tilde{D}^{-1} \tilde{U}_k) B_{kq} V_q^T$$

$$\tilde{V}_k^T \tilde{D}^{-1} \tilde{U}_k \rightarrow \mathcal{O}(r^3)$$

$$F_{22} - \Psi \Phi^T \qquad \Psi, \Phi \sim \mathcal{O}(rN)$$

- U_q and V_q: traverse q subtree
- Cheap multiply with random vectors



Rank pattern and solver complexity

With r the maximum rank of a block:

- HSS construction: $\Theta(r^2 N \log N)$
  iso $\Theta(rN^2)$ for non-randomized.
- ULV factorization: $\Theta(r^2 N)$.
- HSS solution: $\Theta(rN)$.

Typical r: [Chandrasekaran et al.] ($^{(*)}$: with some very strong assumptions.)

2D Poisson	$\Theta(1)$ $^{(*)}$
2D Helmholtz	$\Theta(\log k)$ $^{(*)}$
3D Poisson	$\Theta(k)$
3D Helmholtz	$\Theta(k)$

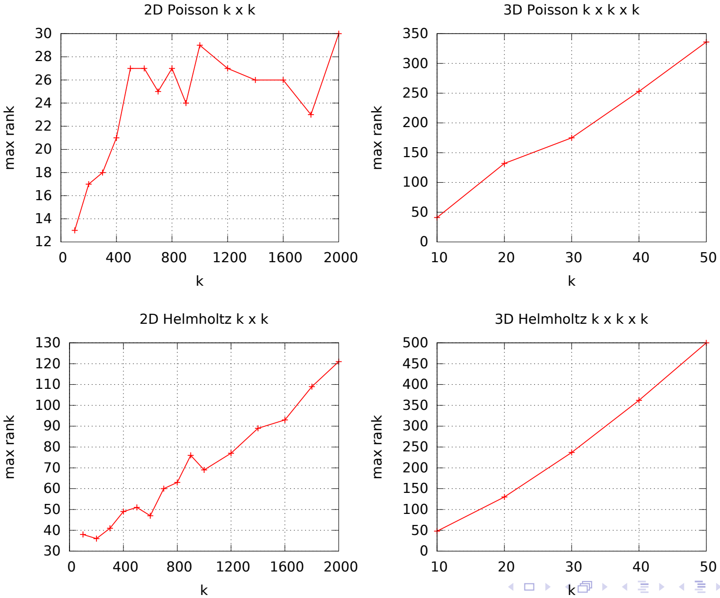
3D Helmholtz:

	Mem	Flops
MF-HSS	$\Theta(N \log N)$	$\Theta(N^{4/3} \log N)$
MF-HSS + RS	$\Theta(N)$	$\Theta(N^{10/9} \log N)$

with regular dense MF for $\ell < \ell_s$ and HSS for $\ell \geq \ell_s$.

Numerical example

Maximum rank over all frontal matrices and all HSS blocks



Parallel implementation

We have a serial code (StruMF [Napov 11'-12'])

- Some performance issues
  - Currently generates random vectors for all nodes in e-tree
  - How to estimate the rank? Currently guess and start over when too small

Parallel implementation is a work in progress

- Distributed memory HSS compression of dense matrix
  - MPI, BLACS, PBLAS, BLAS, LAPACK
- Shared memory multifrontal code
  - OpenMP task parallelism for tree traversal for both elimination tree ans HSS tree
  - Next step is parallel dense algebra

Parallel HSS compression – MPI code

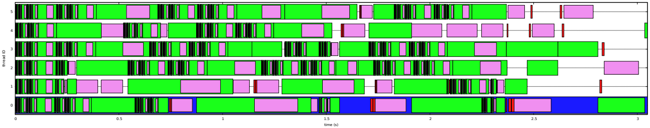
- Topmost separator of a 3D problem, generated with exact multifrontal method

k	100	200	300
N	10,000	40,000	90,000
MPI processes / cores	64	256	1024
Nodes	4	16	64
Levels	6	7	8
Tolerance	1e-3	1e-3	1-e3
Non-randomized	8.3	51.5	193.4
Randomized	2.9	16.0	37.2
Dense LU ScaLAPACK	4.2	57.6	175.9

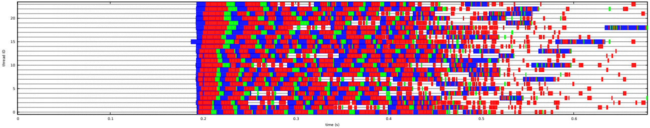
- On 1024 cores
  - achieved 5.3TFlops/s
  - very good flop balance: min / max = 0.93
  - 17% communication overhead

Task based parallel tree traversal – OpenMP

- HSS Compression of dense frontal matrix



- Multifrontal



Blue: E-tree node, Red: HSS compression, Green: ULV-fact

- Extraction of elements from HSS matrix forms bottleneck
- Considering other runtime task schedulers
  - Intel TBB, StarPU, Quark
  - The Quark scheduler from PLASMA could allow integration of PLASMA parallel (tiled) BLAS/LAPACK

Conclusions

- HSS is restricted format, large gains possible for certain applications, not for all
- Some performance issues need to be addressed
- Separate distributed and shared memory codes
  - Needs to be combined in hybrid MPI+X code
- Prepare for next generation NERSC supercomputer
  - Intel MIC based, $>$ 60 cores

Conclusions

- HSS is restricted format, large gains possible for certain applications, not for all
- Some performance issues need to be addressed
- Separate distributed and shared memory codes
  - Needs to be combined in hybrid MPI+X code
- Prepare for next generation NERSC supercomputer
  - Intel MIC based, $>$ 60 cores

Thank you! Questions?