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)

2014 CBMS-NSF Conference:
Fast direct solvers for elliptic PDEs
June 23–29, 2014, Dartmouth College

Introduction

Consider solving

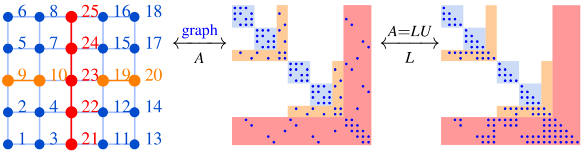
$$Ax = b \qquad \text{or} \qquad M^{-1}A = M^{-1}b$$

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

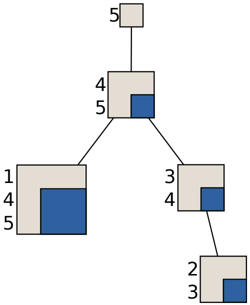
- Fast convergence if good preconditioner $M \approx A$ is available
  1. Cheap to construct, store, apply, parallelize
  2. Good approximation of A

  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)
- Approximation idea: use low-rank approximation

The multifrontal method [Duff & Reid '83]



- Nested dissection reordering defines an elimination tree
  - SCOTCH graph partitioner
- 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: extend-add



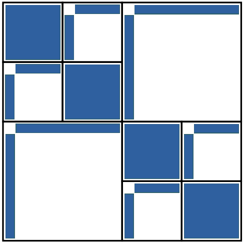
Elimination tree

Low-rank property

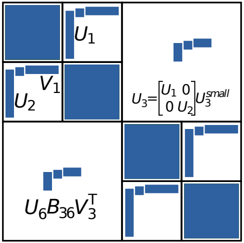
- In many applications, frontal matrices exhibit some low-rank blocks ("data sparsity")
- Compression with SVD, Rank-Revealing QR,...
  - SVD: optimal but too expensive
  - RRQR: QR with column pivoting
- Exact compression or with a compression threshold ε
- Recursively, diagonal blocks have low-rank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

Low-rank representations

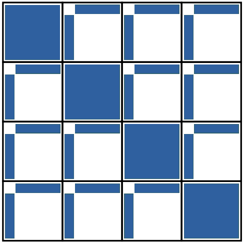
Most low-rank representations belong to the class of $\mathcal{H}$-matrices [Bebendof, Börm, Hackbush, Grasedyck,. . .]. Embedded in both dense and sparse solvers:



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



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

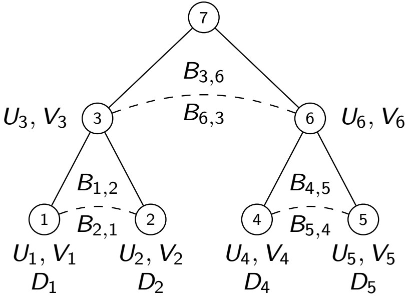


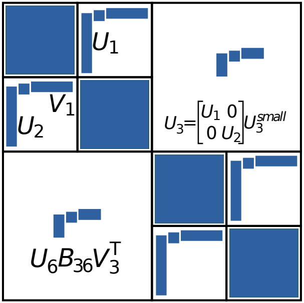
Block-low rank (BLR) [Amestoy et al.] Simple 2D partitioning. Recently in MUMPS.

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.

HSS/HBS representation

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.

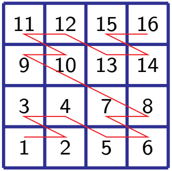
Embedding low-rank techniques in a multifrontal solver

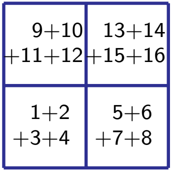
1. Choose which frontal matrices are compressed (size, level...)
2. Low-rankness: weak interactions between "distant" variables

   $\Longrightarrow$ need suitable ordering/clustering of each frontal matrix

   - Geometric setting (3D grid): 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

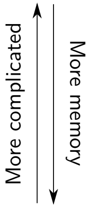




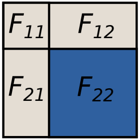
   - Algebraic: add some kind of halo to (complete) graph of separator variables and call a graph partitioner (METIS) [Amestoy et al., Napov]

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
- $S^r = AR^r$ 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 ${S^r}^T = {S^r}^T(:, J^r)X^r$,

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

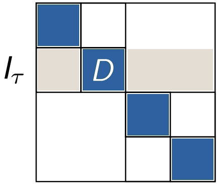
HSS compression via randomized sampling – 2

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: $S_\tau = S_{loc}(J_\tau,:)$

     $R_\tau = U_\tau^T\, R(I_\tau,:)$

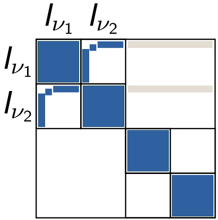
     $I_\tau = I_\tau(J_\tau,:)$



- 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)

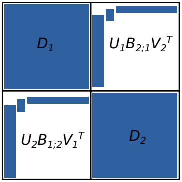
Stages:

- 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

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

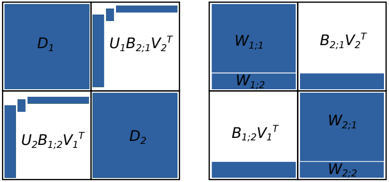


HSS ULV factorization

- 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

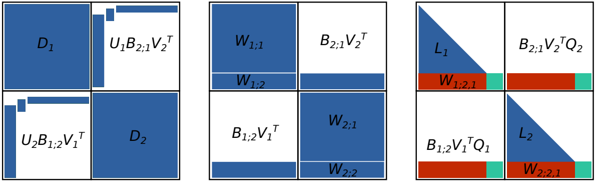
- Exploit structure of U_τ (from ID) to introduce zero's

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

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

- Take (full) LQ decomposition

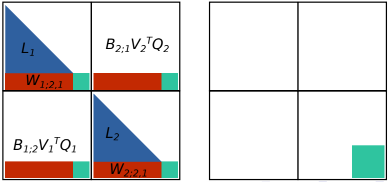
$$W_\tau = \begin{bmatrix} \begin{bmatrix} L_\tau & 0 \end{bmatrix} Q_\tau \\ W_{\tau;2} \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} L_1 & & \\ & [W_{1;2} Q_1^*] & [B_{1,2} V_2^T Q_2^*] \\ & & L_2 \\ [B_{2,1} V_1^T Q_1^*] & & [W_{2;2} Q_2^*] \end{bmatrix} \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix}$$



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

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

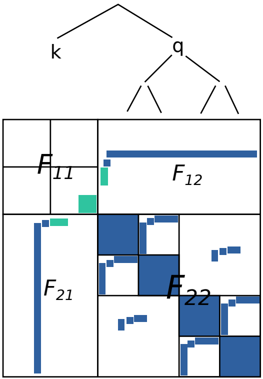
- $\tilde{D}_k$ is reduced HSS matrix $\mathcal{O}(r \times r)$

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

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

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

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



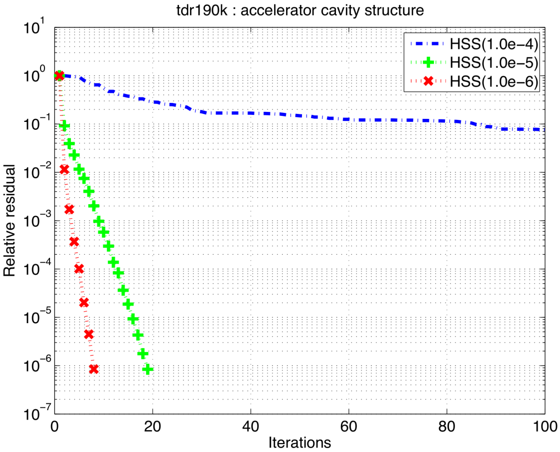
Numerical example – general matrices

- GMRES(30) with right preconditioner
- Multifrontal with HSS vs ILUTP from SuperLU
- $b = (1, 1, \dots)^T$, $x_0 = (0, 0, \dots)^T$
- Stopping criterium: $\|r_k\|_2 / \|b\|_2 \leq 10^{-6}$
- HSS compression tolerance: 10^{-6}

				fill-ratio		factor (s)		its	
Matrix	descr	N	rank	HSS	ILU	HSS	ILU	HSS	ILU
add32	circuit	4, 690	0	2.1	1.3	0.01	0.01	1	2
mchln85ks17	car tire	84, 180	948	13.5	12.3	133.8	216.1	4	39
mhd500	plasma	250, 000	100	11.6	15.6	2.5	7.9	2	8
poli large	economics	15, 575	64	4.8	1.6	0.04	0.02	1	2
stomach	bio eng.	213, 360	92	12.1	2.9	13.8	18.7	2	2
tdr190k	accelerator	1, 100, 242	596	14.1	-	629.2	-	7	-
torso3	bio eng.	259, 156	136	22.6	2.4	86.7	63.7	2	2
utm5940	tokamak	5, 940	123	6.7	8.0	0.1	0.16	3	15
wang4	device	26, 068	385	45.3	23.1	4.4	6.4	3	4

Numerical example – tdr190k

- tdr190k: Maxwell equations in the frequency domain
- GMRES(30) convergence



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 and 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 (s)	8.3	51.5	193.4
Randomized (s)	2.9	16.0	37.2
Dense LU ScaLAPACK (s)	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

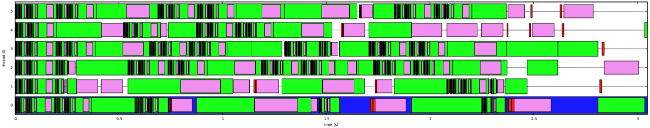
Postorder (leaf to root) tree traversal: handle siblings in parallel, wait for children

```
void traverse(node* p) {
  if (p->left)
    #pragma omp task
       traverse(p->left);
  if (p->right)
    #pragma omp task
       traverse(p->right);
  #pragma omp taskwait
  process(p->data);
}
```

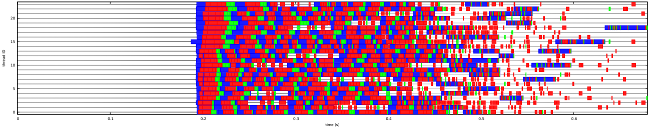
- Run-time system schedules tasks to cores (work-stealing)
- Root node will be handled sequentially: scaling bottleneck
- Nested trees: node of nested dissection tree contains HSS tree

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

- Developing an algebraic preconditioner for general nonsymmetric matrices (from PDEs)
- HSS is restricted format, large gains possible for certain applications, not for all
- Graph partitioning difficulties
  - Separator not always just a plane/line, not always a single piece
  - Bad for rank structure
- Some performance issues need to be addressed
- Separate distributed and shared memory codes
  - How to combine in a hybrid MPI+X code?
- Prepare for next generation NERSC supercomputer
  - Intel MIC based, > 60 cores

Conclusions

- Developing an algebraic preconditioner for general nonsymmetric matrices (from PDEs)
- HSS is restricted format, large gains possible for certain applications, not for all
- Graph partitioning difficulties
  - Separator not always just a plane/line, not always a single piece
  - Bad for rank structure
- Some performance issues need to be addressed
- Separate distributed and shared memory codes
  - How to combine in a hybrid MPI+X code?
- Prepare for next generation NERSC supercomputer
  - Intel MIC based, > 60 cores

Thank you! Questions?