Finding the Inertia of HSS Matrices

Jonas Actor

Lawrence Berkeley National Laboratory
Rice University

5 September 2018

Contact: jonasactor@rice.edu

Trajectory

Table of Contents

Is my matrix...

Is my matrix...

- square?
- symmetric?
- invertible?
- unitary?
- diagonal?
- (semi)definite?
- sparse?
- ...

Sparsity: Shortcut to fast matrix computation

A is **sparse** if it makes sense to keep track of which elements don't matter (Wilkinson)

Sparse $\approx$ few operations to describe A's action

Rank: Matrices with little information

For $A : \mathbb{R}^n \to \mathbb{R}^m$,

$$rank(A) = dim(\, im(A)\,)$$

Rank $\approx$ information to describe linear map A

Sparsity vs. Rank

	Low Rank	Full Rank
Sparse	$\begin{bmatrix} 0 & & & \\ & 0 & & \\ & & 0 & \\ & & & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$
Dense	$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$

Solving equations quickly...

$A \in \mathbb{R}^{n \times n}$

- Goal: solve $Ax = b$ for multiple RHS
- Caveat: Allowed only one factorization $O(n^3)$

Solving equations quickly...

$A \in \mathbb{R}^{n \times n}$

- Goal: solve $Ax = b$ for multiple RHS
- Caveat: Allowed only one factorization $O(n^3)$

$$LU \longrightarrow O(n^2)$$

Exploiting structure

$A = diag(\vec{a}) \in \mathbb{R}^{n \times n}$

- Goal: solve $Ax = b$ for multiple RHS
- Caveat: Allowed only one factorization $O(n^3)$

Exploiting structure

$A = diag(\vec{a}) \in \mathbb{R}^{n\times n}$

- Goal: solve $Ax = b$ for multiple RHS
- Caveat: Allowed only one factorization $O(n^3)$

$$\begin{aligned} LU(A) &\longrightarrow O(n^2) \\ \text{Direct solve} &\longrightarrow O(n) \end{aligned}$$

Other examples: decoupled problems, saddle points, Schur complement, permutations

Exploiting rank deficiency

$A \in \mathbb{R}^{n \times n}$ with rank $r < n$

- Goal: solve $Ax = b$ for multiple RHS
- Caveat: Allowed only one factorization $O(n^3)$

Exploiting rank deficiency

$A \in \mathbb{R}^{n \times n}$ with rank $r < n$

- Goal: solve $Ax = b$ for multiple RHS
- Caveat: Allowed only one factorization $O(n^3)$

$$\begin{aligned} LU &\longrightarrow O(n^2) \\ \text{Rank-revealing factorization} &\longrightarrow O(r^3 + nr) \end{aligned}$$

Question

What if A is full-rank,
but has low-rank blocks?

Table of Contents

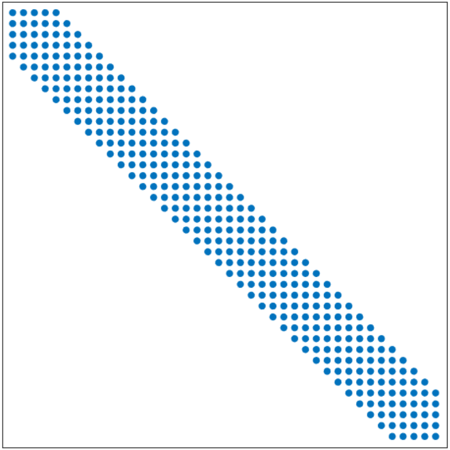
HSS matrices exploit low-rank structures

- HSS = Hierarchically SemiSeparable
- Off-diagonal blocks **recursively** low-rank
- Rank of off-diagonal blocks bounded $\leq r$
- Related to Fast Multipole Method

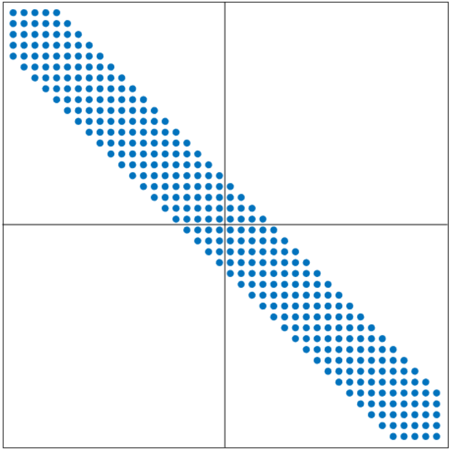
Examples of HSS matrices

- Toeplitz / banded matrices
- Inverse of (many) FD/FEM stencils
- (Many) Boundary integral operators
- (Some) KKT matrices

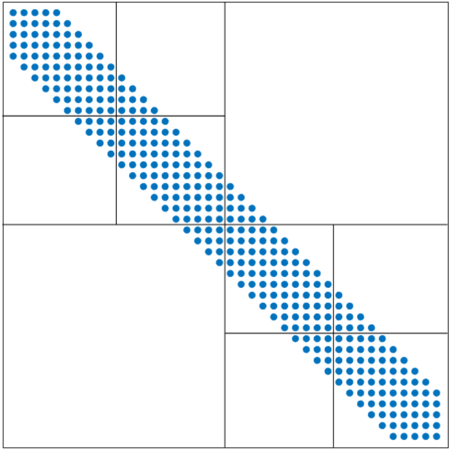
Sketch of basic idea



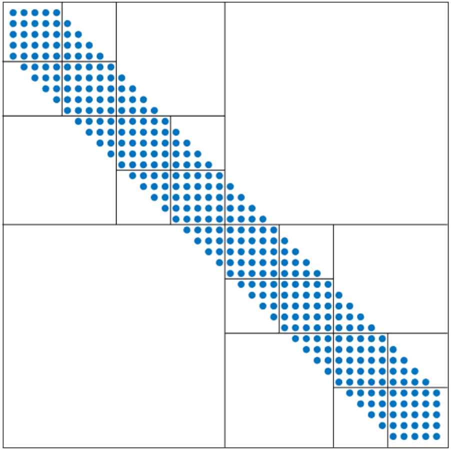
Sketch of basic idea



Sketch of basic idea



Sketch of basic idea



HSS Representation

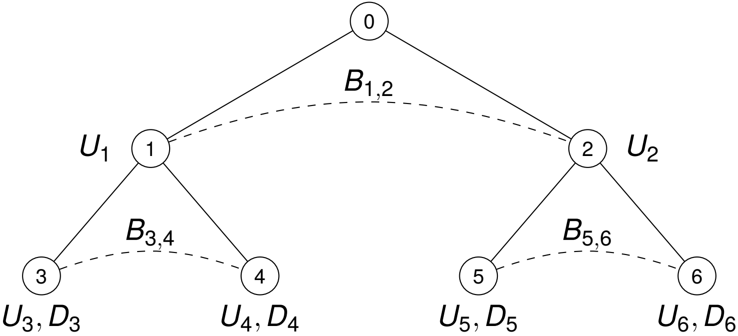
$$A = \begin{bmatrix} D_1 & A_{12} \\ A_{21} & D_2 \end{bmatrix}$$

HSS Representation

$$A = \begin{bmatrix} D_1 & A_{12} \\ A_{21} & D_2 \end{bmatrix}$$

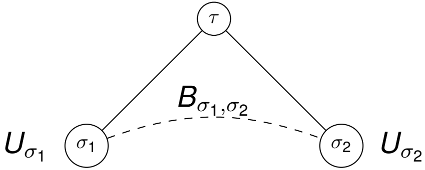
$$= \begin{bmatrix} \begin{bmatrix} D_3 & A_{34} \\ A_{43} & D_4 \end{bmatrix} & A_{12} \\ A_{21} & \begin{bmatrix} D_5 & A_{56} \\ A_{65} & D_6 \end{bmatrix} \end{bmatrix}$$

HSS Tree



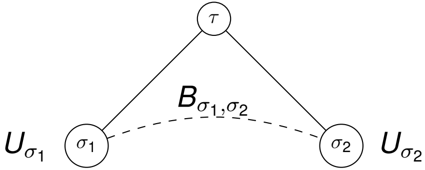
HSS tree, $L = 2$

Recursive Definition



$$D_\tau = \begin{bmatrix} D_{\sigma_1} & U^b_{\sigma_1} B_{\sigma_1,\sigma_2} {U^b_{\sigma_2}}^T \\ U^b_{\sigma_2} B_{\sigma_2,\sigma_1} {U^b_{\sigma_1}}^T & D_{\sigma_2} \end{bmatrix}$$

Hierarchical Basis



$$U_\tau^b = \begin{bmatrix} U_{\sigma_1}^b & \\ & U_{\sigma_2}^b \end{bmatrix} U_\tau$$

HSS Representation

$$A = \begin{bmatrix} D_1 & U_1^b B_{12} {U_2^b}^T \\ U_2^b B_{21} {U_1^b}^T & D_2 \end{bmatrix}$$

HSS Representation

$$A = \begin{bmatrix} D_1 & U_1^b B_{12} U_2^{b\,T} \\ U_2^b B_{21} U_1^{b\,T} & D_2 \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} D_3 & U_3 B_{34} U_4^T \\ U_4 B_{43} U_3^T & D_4 \end{bmatrix} & U_1^b B_{12} U_2^{b\,T} \\ U_2^b B_{21} U_1^{b\,T} & \begin{bmatrix} D_5 & U_5 B_{56} U_6^T \\ U_6 B_{65} U_5^T & D_6 \end{bmatrix} \end{bmatrix}$$

HSS Representation

$$A = \begin{bmatrix} D_1 & U_1^b B_{12} U_2^{b\,T} \\ U_2^b B_{21} U_1^{b\,T} & D_2 \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} D_3 & U_3 B_{34} U_4^T \\ U_4 B_{43} U_3^T & D_4 \end{bmatrix} & U_1^b B_{12} U_2^{b\,T} \\ U_2^b B_{21} U_1^{b\,T} & \begin{bmatrix} D_5 & U_5 B_{56} U_6^T \\ U_6 B_{65} U_5^T & D_6 \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} D_3 & U_3 B_{34} U_4^T \\ U_4 B_{43} U_3^T & D_4 \end{bmatrix} & \begin{bmatrix} U_3 & \\ & U_4 \end{bmatrix} U_1 B_{12} U_2^T \begin{bmatrix} U_5^T & \\ & U_6^T \end{bmatrix} \\ \begin{bmatrix} U_5 & \\ & U_6 \end{bmatrix} U_2 B_{21} U_1^T \begin{bmatrix} U_3^T & \\ & U_4^T \end{bmatrix} & \begin{bmatrix} D_5 & U_5 B_{56} U_6^T \\ U_6 B_{65} U_5^T & D_6 \end{bmatrix} \end{bmatrix}$$

A problem...

How do we find B_{σ_1,σ_2}, U_σ for low-rank off-diagonal subblocks?

Interpolative Decomposition

Express A via spanning set of r columns J

$$A = A(:, J)X$$

Obtain via rank-revealing QR:

$$\begin{aligned} A\Pi &= Q \begin{bmatrix} R_1 & R_2 \end{bmatrix} \\ A &= QR_1 \begin{bmatrix} I & R_1^{-1}R_2 \end{bmatrix} \Pi^{-1} \\ &= A(:, J)X. \end{aligned}$$

How to Use ID

Perform ID on A_{σ_1,σ_2}:

$$A_{\sigma_1,\sigma_2} = A_{\sigma_1,\sigma_2}(:,J)X_2$$

Perform ID on $(A_{\sigma_1,\sigma_2}(:,J))^T$:

$$A_{\sigma_1,\sigma_2} = X_1^T A_{\sigma_1,\sigma_2}(J,J)X_2$$

Structure of HSS Factors

$$B_{\sigma_1,\sigma_2} = A_{\sigma_1,\sigma_2}(J,J) \quad \longrightarrow \quad \text{submatrix of } A_{\sigma_1,\sigma_2}$$

$$\begin{aligned} U_{\sigma_1} &= X_1^T \\ &= \Pi \begin{bmatrix} I \\ R_1^{-1} R_2 \end{bmatrix} \quad \longrightarrow \quad \text{from ID factors} \end{aligned}$$

HSS Factorization: Cost and Benefits

- Forms (near)-ULV factorization
- Costs $O(n^3)$ to factorize
- Costs $O(r^2 n)$ to solve linear systems
- Reduce costs via random projections

Table of Contents

Inertia of matrices

$$\nu(A) = (n_+, n_-, n_0)$$

Triple of integers, counting signs of eigenvalues

Sylvester's Theorem

For any square matrix A, any full rank matrix B,

$$\nu(A) = \nu(BAB^T).$$

If $A = LDL^T$ factorization,

$$\nu(A) = \nu(LDL^T) = \nu(D).$$

Haynsworth's Theorem

For a Hermitian matrix $M = \begin{bmatrix} A & B \\ B^* & C \end{bmatrix}$ where A is nonsingular and Hermitian,

$$\begin{aligned} \nu(M) &= \nu(A) + \nu(M \backslash A) \\ &= \nu(A) + \nu(C - B^* A B). \end{aligned}$$

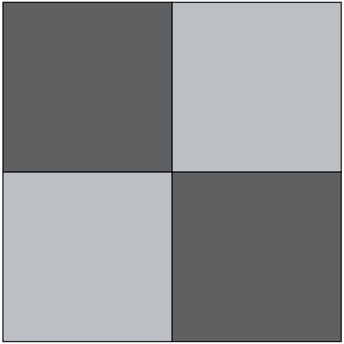
Haynsworth's Theorem: Proof

$$
\begin{aligned}
M &= \begin{bmatrix} A & B \\ B^* & C \end{bmatrix} \\
&= \begin{bmatrix} I & \\ -B^*A^{-1} & I \end{bmatrix} \begin{bmatrix} A & \\ & C - B^*AB \end{bmatrix} \begin{bmatrix} I & -A^{-1}B \\ & I \end{bmatrix}.
\end{aligned}
$$

Inertia for HSS matrices

1. Introduce Zeros
2. Partial LDL^T Factorization
3. Schur Complement
4. Permute and Repeat

Step 0: Factorization



$$D_\tau = \begin{bmatrix} D_{\sigma_1} & U_{\sigma_1} B_{\sigma_1,\sigma_2} U_{\sigma_2}^T \\ U_{\sigma_2} B_{\sigma_2,\sigma_1} U_{\sigma_1}^T & D_{\sigma_2} \end{bmatrix}$$

low-rank blocks LDL^T factors full-rank blocks **diagonal blocks**

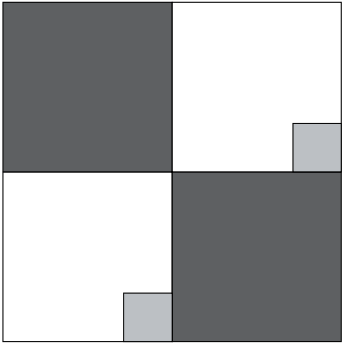
Step 1: Introduce Zeros

Recall from ID that $U_\sigma = \Pi \begin{bmatrix} I \\ R_1^{-1} R_2 \end{bmatrix}$.

Define $\Omega_\sigma = \begin{bmatrix} -R_1^{-1} R_2 & I \\ I & 0 \end{bmatrix} \Pi^T$.

Then $\Omega_\sigma U_\sigma = \begin{bmatrix} 0 \\ I \end{bmatrix}$.

Step 1: Introduce Zeros

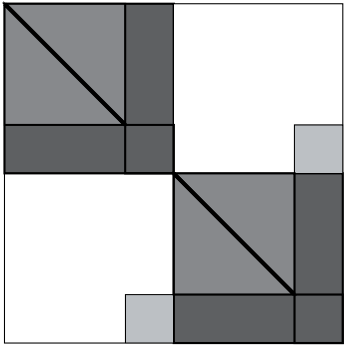


$$\begin{bmatrix} \Omega_{\sigma_1} & \\ & \Omega_{\sigma_2} \end{bmatrix} D_\tau \begin{bmatrix} \Omega_{\sigma_1} & \\ & \Omega_{\sigma_2} \end{bmatrix}^T$$

$$= \begin{bmatrix} \Omega_{\sigma_1} D_{\sigma_1} \Omega_{\sigma_1}^T & \begin{bmatrix} 0 \\ I \end{bmatrix} B_{\sigma_1,\sigma_2} \begin{bmatrix} 0 \\ I \end{bmatrix}^T \\ \begin{bmatrix} 0 \\ I \end{bmatrix} B_{\sigma_2,\sigma_1} \begin{bmatrix} 0 \\ I \end{bmatrix}^T & \Omega_{\sigma_2} D_{\sigma_2} \Omega_{\sigma_2}^T \end{bmatrix}$$

low-rank blocks LDL^T factors full-rank blocks **diagonal blocks**

Step 2: Partial LDL^T Factorization



$$\Omega_\sigma D_\sigma \Omega_\sigma^T = \bar{D}_\sigma = \begin{bmatrix} \bar{D}_{\sigma,11} & \bar{D}_{\sigma,12} \\ \bar{D}_{\sigma,21} & \bar{D}_{\sigma,22} \end{bmatrix}$$

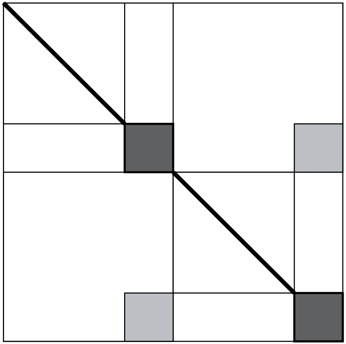
$$= \begin{bmatrix} \widehat{L}_{11}\widehat{D}_\sigma \widehat{L}_{11}^T & \bar{D}_{\sigma,12} \\ \bar{D}_{\sigma,21} & \bar{D}_{\sigma,22} \end{bmatrix}$$

$$\widehat{L}_{21} = \bar{D}_{\sigma,21}(\widehat{D}_\sigma \widehat{L}_{11}^T)^{-1}$$

$$L_\sigma = \begin{bmatrix} \widehat{L}_{11} & \\ \widehat{L}_{21} & I \end{bmatrix}$$

low-rank blocks LDL^T factors full-rank blocks **diagonal blocks**

Step 3: Schur Complement

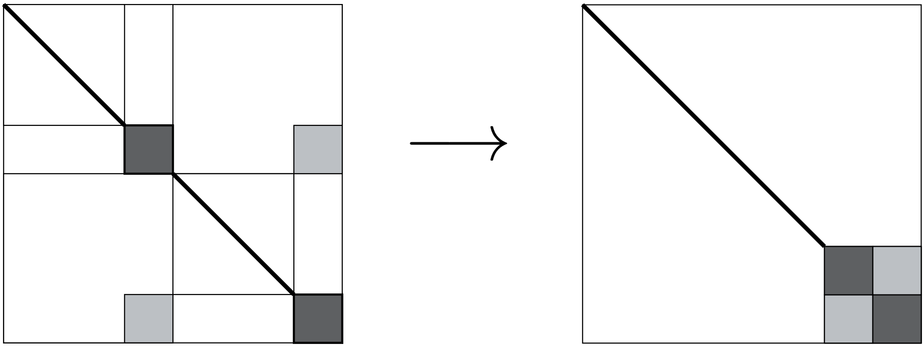


$$S_\sigma = \bar{D}_{\sigma,22} - \bar{D}_{\sigma,21}\bar{D}_{\sigma,11}^{-1}\bar{D}_{\sigma,12}$$

$$L_\sigma^{-1}\bar{D}_\sigma L_\sigma^{-T} = \begin{bmatrix} \widehat{D}_\sigma & \\ & S_\sigma \end{bmatrix}$$

low-rank blocks LDL^T factors **full-rank blocks** **diagonal blocks**

Step 4: Permute and Repeat



low-rank blocks $\quad$ LDL^T factors $\quad$ full-rank blocks $\quad$ **diagonal blocks**

Algorithm 1: Inertia of HSS Matrix

Data: Symmetric HSS matrix factors $U_\tau,\ B_{\sigma_1,\sigma_2},\ D_\tau$

Result: Inertia $\nu = [\lambda_+, \lambda_-, \lambda_0]$

foreach *Node τ (postordering)* **do**

  if τ *is a leaf* **then**

    $\widetilde{D}_\tau \leftarrow D_\tau$

  else

    $\widetilde{D}_\tau \leftarrow \begin{bmatrix} S_{\sigma_1} & B_{\sigma_1,\sigma_2} \\ B^T_{\sigma_1,\sigma_2} & S_{\sigma_2} \end{bmatrix}$

  end

  if τ *is the root* **then**

    $[L, \widehat{D}_\tau, P] \leftarrow LDL(\widetilde{D}_\tau)$

  else

    $\bar{D} \leftarrow \Omega_\tau \widetilde{D}_\tau \Omega_\tau^T \qquad \bar{D} = \begin{bmatrix} \bar{D}_{11} & \bar{D}_{12} \\ \bar{D}_{21} & \bar{D}_{22} \end{bmatrix}$

    $[L, \widehat{D}_\tau, P] \leftarrow LDL(\bar{D}_{11})$

    $S_\tau \leftarrow \bar{D}_{22} - \bar{D}_{21}\bar{D}_{11}^{-1}\bar{D}_{12}$

  end

  $\nu \leftarrow \nu + \nu(\widehat{D}_\tau)$

end

Cost analysis

- $A : \mathbb{R}^{n \times n}$ is HSS with rank r
- HSS tree has $L = \log_2 \left(\frac{n}{r}\right)$ levels
- Level ℓ has 2^ℓ nodes
- Cost is $c \cdot r^3$ for each node

Cost analysis

$$
\begin{aligned}
\text{Total Cost} &= \sum_{\ell=0}^{L-1} \sum_{i=1}^{2^\ell} (\text{Cost at node } i \text{ at level } \ell) \\
&= \sum_{\ell=0}^{L-1} \sum_{i=1}^{2^\ell} c \cdot r^3 \\
&= \sum_{\ell=0}^{L-1} c \cdot 2^\ell \cdot r^3 \\
&= O(r^3 \cdot 2^L) \qquad = O(r^2 n).
\end{aligned}
$$

Table of Contents

Fast NLA can be achieved by...

- Exploiting sparsity
- Exploiting low-rank structures
- Exploiting hierarchy in problem size

If you know the structure of your problem, exploit it!

Inertia of HSS Matrices

- Linear-time algorithm for inertia
- Extension for sparse matrices

- Caveat : requires successful Schur complement
- Caveat : input needed for block-sparse systems

HSS Factorization: Software

STRUMPACK = STRUctured Matrix PACKage

- Uses multithreaded, distributed memory
- Works with sparse, dense matrices
- Exploits random matrix factorizations for speedup

Inertia via STRUMPACK

- Dense, sparse HSS inertia in serial
- Multithreaded coming soon
- Distributed memory not coming soon
  - limited by ScaLaPACK
  - switch to SLATE

Acknowledgements

- Lawrence Berkeley National Laboratory
- Ken Kennedy Institue for Information Technology
- National Library of Medicine
- Keck Gulf Coast Consortium