Fast Algorithms for Hierarchically Semiseparable Matrices

Xia, Chanrasekaran, Gu, Li

Presented by Loren Anderson

April 27, 2017
Motivation

Setup: \( N \times N \) matrix \( H \), assume \( 4 \mid N \)
Partition: \( \left\{ \frac{N}{4}, \frac{N}{4}, \frac{N}{4}, \frac{N}{4} \right\} \)
Setup: $N \times N$ matrix $H$, assume $4|N$
Partition: $\{ \frac{N}{4}, \frac{N}{4}, \frac{N}{4}, \frac{N}{4} \}$
Motivation

Setup: $N \times N$ matrix $H$, assume $4|N$
Partition: $\{\frac{N}{4}, \frac{N}{4}, \frac{N}{4}, \frac{N}{4}\}$
Motivation

Setup: $N \times N$ matrix $H$, assume $4|N$
Partition: $\left\{ \frac{N}{4}, \frac{N}{4}, \frac{N}{4}, \frac{N}{4} \right\}$
Motivation

Setup: $N \times N$ matrix $H$, assume $4|N$
Partition: $\{N/4, N/4, N/4, N/4\}$
Motivation

Setup: $N \times N$ matrix $H$, assume $4|N$
Partition: $\left\{ \frac{N}{4}, \frac{N}{4}, \frac{N}{4}, \frac{N}{4} \right\}$
Motivation (Contd.)

Setup: $N \times N$ matrix $H$, assume $4|N$
Partition: $\{(N/4 + N/4), (N/4 + N/4)\}$
Motivation (Contd.)

Setup: $N \times N$ matrix $H$, assume $4|N$
Partition: $\{\left(\frac{N}{4} + \frac{N}{4}\right), \left(\frac{N}{4} + \frac{N}{4}\right)\}$
Motivation (Contd.)

Setup: $N \times N$ matrix $H$, assume $4|N$
Partition: $\{ (\frac{N}{4} + \frac{N}{4}), (\frac{N}{4} + \frac{N}{4}) \}$
Motivation (Contd.)

Setup: $N \times N$ matrix $H$, assume $4|N$
Partition: $\{(\frac{N}{4} + \frac{N}{4}), (\frac{N}{4} + \frac{N}{4})\}$
Motivation (Contd.)
Motivation (Contd.)
Motivation (Contd.)

The diagram illustrates the structure of an HSS matrix, showing the levels and non-zero patterns. The matrix $H$ is decomposed into four sub-matrices: $m_{1;1}$, $m_{1;2}$, $m_{2;1}$, and $m_{2;2}$. Each sub-matrix represents a block of the HSS matrix, with non-zero entries highlighted in red and zero entries in blue.
Formal Definitions

Let $H$ be of dimension $N \times N$. Form a partition sequence

$$\{m_{k,j}\}_{j=1}^{2^k} \text{ where } \sum_{i=1}^{2^k} m_{k,j} = N.$$ 

Partition rows and columns of $H$ so that block row (column) $j$ has row (column) dimension $m_{k,j}$.

**HSS Block**

Let $D_{k,j} = m_{k,j} \times m_{k,j}$ be the intersection of block row and column $j$. Removing $D_{k,j}$ from the block row (column) $j$ yields the $j$-th HSS block row (column) at the $k$-th level partition.
Formal Definitions

\[ m_{2;1} \]
\[ m_{2;2} \]
\[ m_{2;3} \]
\[ m_{2;4} \]
Formal Definitions

\[
\begin{array}{cccc}
m_{2;1} & D_{2;1} & & \\
m_{2;2} & & D_{2;2} & \\
m_{2;3} & & D_{2;3} & \\
m_{2;4} & & & D_{2;4}
\end{array}
\]
Formal Definitions

$m_{2;1}$
$m_{2;2}$
$m_{2;3}$
$m_{2;4}$
Formal Definitions

$m_{2;1}$

$m_{2;2}$

$m_{2;3}$

$m_{2;4}$
Formal Definitions

$m_{2;1}$
$m_{2;2}$
$m_{2;3}$
$m_{2;4}$
Formal Definitions

$m_{2;1}$

$m_{2;2}$

$m_{2;3}$

$m_{2;4}$
Formal Definitions

\begin{align*}
m_{2;1} & \\
m_{2;2} & \\
m_{2;3} & \\
m_{2;4} & \\
\end{align*}
Upper Level HSS Block

Upper level HSS blocks are formed using \( \{ m_{i,j} \}_{j=1}^{2^i} \) where

\[
m_{i-1,j} = m_{i;2j-1} + m_{i;2j}, \quad j \in \{1, 2, \ldots, 2^{i-1}\}, \quad i \in \{k, k-1, \ldots, 2, 1\}.
\]
Upper Level HSS Block

Upper level HSS blocks are formed using \( \{m_{i,j}\}_{j=1}^{2^i} \) where

\[
m_{i-1,j} = m_{i,2j-1} + m_{i,2j}, j \in \{1, 2, \ldots, 2^{i-1}\}, i \in \{k, k-1, \ldots, 2, 1\}.
\]
Upper level HSS blocks are formed using \( \{m_{i;j}\}_{j=1}^{2^i} \) where

\[
m_{i-1;j} = m_{i;2j-1} + m_{i;2j}, j \in \{1, 2, \ldots, 2^{i-1}\}, i \in \{k, k-1, \ldots, 2, 1\}.
\]
Upper Level HSS Block

Upper level HSS blocks are formed using \( \{m_{ij}\}_{j=1}^{2^i} \) where

\[
m_{i-1,j} = m_{i;2j-1} + m_{i;2j}, j \in \{1, 2, \ldots, 2^{i-1}\}, i \in \{k, k - 1, \ldots, 2, 1\}.
\]
Upper Level HSS Block

Upper level HSS blocks are formed using \( \{m_{i,j}\}_{j=1}^{2^i} \) where

\[
m_{i-1,j} = m_{i,2j-1} + m_{i,2j}, j \in \{1, 2, \ldots, 2^{i-1}\}, i \in \{k, k - 1, \ldots, 2, 1\}.
\]
Formal Definitions

Upper Level HSS Block

Upper level HSS blocks are formed using \( \{m_{i;j}\}_{j=1}^{2^i} \) where

\[
m_{i-1;j} = m_{i;2j-1} + m_{i;2j}, j \in \{1, 2, \ldots, 2^{i-1}\}, i \in \{k, k - 1, \ldots, 2, 1\}.
\]
Upper Level HSS Block

Upper level HSS blocks are formed using \( \{m_{i,j}\}_{j=1}^{2^i} \) where

\[
m_{i-1,j} = m_{i;2j-1} + m_{i;2j}, j \in \{1, 2, \ldots, 2^{i-1}\}, i \in \{k, k-1, \ldots, 2, 1\}.
\]
Formal Definitions

HSS Rank

The HSS Rank of \( H \) is the maximum rank of all the HSS off-diagonal blocks at all levels of HSS partitions of \( H \).
Formal Definitions

HSS Rank

The HSS Rank of $H$ is the maximum rank of all the HSS off-diagonal blocks at all levels of HSS partitions of $H$. 

$$
\begin{align*}
&\begin{array}{c}
m_2;1 \\
m_2;2 \\
m_2;3 \\
m_2;4 \\
\end{array} \\
&\begin{array}{c}
m_1;1 \\
m_1;2 \\
\end{array}
\end{align*}
$$
Formal Definitions

**HSS Tree, HSS Representation**

Let $T$ be a perfect binary tree where each node is associated with an HSS block row. The matrix $H$ has an HSS representation if there exist matrices $D_{i;j}, U_{i;j}, V_{i;j}, R_{i;j}, W_{i;j}, B_{i;j, j \pm 1}$ associated with each tree node $(i;j)$ satisfying a set of recursions such that $D_{0;1} = H$. 
Formal Definitions

HSS Tree, HSS Representation

Let $T$ be a perfect binary tree where each node is associated with an HSS block row. The matrix $H$ has an HSS representation if there exist matrices $D_{i,j}, U_{i,j}, V_{i,j}, R_{i,j}, W_{i,j}, B_{i,j}, j \pm 1$ associated with each tree node $(i,j)$ satisfying a set of recursions such that $D_{0,1} = H$.

\[
D_{i-j;1} = \begin{pmatrix}
D_{i;2j-1} & U_{i;2j-1}B_{i;2j-1,2j}V_{i;2j}^T \\
U_{i;2j}B_{i;2j,2j-1} & D_{i;2j}
\end{pmatrix}
\]

\[
U_{i-1;j} = \begin{pmatrix}
U_{i;2j-1}R_{i;2j-1} & \\
U_{i;2j}R_{i;2j}
\end{pmatrix}, \quad V_{i-1;j} = \begin{pmatrix}
V_{i;2j-1}W_{i;2j-1} & \\
V_{i;2j}W_{i;2j}
\end{pmatrix}
\]

$j \in \{1, 2, \ldots, 2^{i-1} - 1, 2^{i-1}\}, \quad i \in \{k, k-1, \ldots, 2, 1\}$
Formal Definitions
Formal Definitions

(2,3) block: $U_{2;2} R_{2;2} B_{1;1,2} W_{2;3}^T V_{2;3}^T$
Formal Definitions

(2,3) block: $U_{2;2}R_{2;2}B_{1;1,2}W_{2;3}^{T}V_{2;3}^{T}$
Formal Definitions

(2,3) block:  \( U_{2;2} R_{2;2} B_{1;1,2} W_{2;3}^T V_{2;3}^T \)
Outline

1. Motivation
2. Definitions
3. New HSS Tree Construction
4. Algorithms
New HSS Notation

\[ H \]

\[ 1; 1 \]

\[ 2; 1 \]

\[ B_{1;1,2} \]

\[ B_{1;2,1} \]

\[ 2; 2 \]

\[ W_{2;1} \]

\[ W_{2;2} \]

\[ R_{2;1} \]

\[ R_{2;2} \]

\[ U_{2;1} \]

\[ U_{2;2} \]

\[ D_{2;1} \]

\[ D_{2;2} \]

\[ 1; 2 \]

\[ 2; 3 \]

\[ B_{2;1,2} \]

\[ B_{2;2,1} \]

\[ B_{2;3,4} \]

\[ B_{2;4,3} \]

\[ 2; 4 \]

\[ W_{2;3} \]

\[ W_{2;4} \]

\[ R_{2;3} \]

\[ R_{2;4} \]

\[ U_{2;3} \]

\[ U_{2;4} \]

\[ D_{2;3} \]

\[ D_{2;4} \]
New HSS Notation

Problem: unintuitive, tedious notation
New HSS Notation

Solution: use postordering notation
New HSS Notation

Advantages

- less complex
- HSS operations frequently traverse HSS Trees
- structure transformations, data manipulations, parallelization
- good data locality
- extends to arbitrary binary trees
New HSS Notation

We consider HSS trees that are full binary trees.
Construction of HSS Representation

\[
H = \begin{bmatrix}
D_{1} & U_{1} & V_{1} & D_{2} & U_{2} & V_{2} & D_{4} & U_{4} & V_{4} & D_{5} & U_{5} & V_{5}
\end{bmatrix}
\]
Construction of HSS Representation

\[
H = \begin{pmatrix}
D_1 & T_{12} & T_{14} & T_{15} \\
T_{21} & D_2 & T_{24} & T_{25} \\
T_{41} & T_{42} & D_4 & T_{45} \\
T_{51} & T_{52} & T_{54} & D_5
\end{pmatrix}
\]
Construction of HSS Representation

\[ H = \begin{pmatrix} D_1 & T_{12} & T_{14} & T_{15} \\ T_{21} & D_2 & T_{24} & T_{25} \\ T_{41} & T_{42} & D_4 & T_{45} \\ T_{51} & T_{52} & T_{54} & D_5 \end{pmatrix} \]
Algorithm

\[
\begin{pmatrix}
D_1 & T_{12} & T_{14} & T_{15} \\
T_{21} & D_2 & T_{24} & T_{25} \\
T_{41} & T_{42} & D_4 & T_{45} \\
T_{51} & T_{52} & T_{54} & D_5
\end{pmatrix}
\]
Algorithm

\[
\begin{pmatrix}
D_1 & T_{12} & T_{14} & T_{15} \\
T_{21} & D_2 & T_{24} & T_{25} \\
T_{41} & T_{42} & D_4 & T_{45} \\
T_{51} & T_{52} & T_{54} & D_5
\end{pmatrix}
\]

- Compress block row/column for node 1
Use QR factorizations:

\[
\begin{pmatrix}
D_1 & T_{12} & T_{14} & T_{15} \\
T_{21} & D_2 & T_{24} & T_{25} \\
T_{41} & T_{42} & D_4 & T_{45} \\
T_{51} & T_{52} & T_{54} & D_5
\end{pmatrix}
\]

- Compress block row/column for node 1
Algorithm

\[
\begin{pmatrix}
D_1 & U_1 \tilde{T}_{12} & U_1 \tilde{T}_{14} & U_1 \tilde{T}_{15} \\
\tilde{T}_{21} V_1^T & D_2 & T_{24} & T_{25} \\
\tilde{T}_{41} V_1^T & T_{42} & D_4 & T_{45} \\
\tilde{T}_{51} V_1^T & T_{52} & T_{54} & D_5
\end{pmatrix}
\]

- Compress block row/column for node 1
Algorithm

\[
\begin{pmatrix}
D_1 & U_1 \tilde{T}_{12} & U_1 \tilde{T}_{14} & U_1 \tilde{T}_{15} \\
\tilde{T}_{21} V_1^T & D_2 & T_{24} & T_{25} \\
\tilde{T}_{41} V_1^T & T_{42} & D_4 & T_{45} \\
\tilde{T}_{51} V_1^T & T_{52} & T_{54} & D_5
\end{pmatrix}
\]

- Compress block row/column for node 1
- Compress block row/column for node 2
Algorithm

\[
\begin{pmatrix}
D_1 & \tilde{U}_1\tilde{T}_{12} & U_1\tilde{T}_{14} & U_1\tilde{T}_{15} \\
\tilde{T}_{21}V_1^T & D_2 & T_{24} & T_{25} \\
\tilde{T}_{41}V_1^T & T_{42} & D_4 & T_{45} \\
\tilde{T}_{51}V_1^T & T_{52} & T_{54} & D_5
\end{pmatrix}
\]

- Compress block row/column for node 1
- Compress block row/column for node 2

Use QR factorizations:

\[
(\tilde{T}_{21}T_{24}T_{25}) = U_2(B_2\tilde{T}_{24}\tilde{T}_{25})
\]

\[
(\tilde{T}_{12}T_{42}T_{52}) = V_2(B_1^T\tilde{T}_{42}\tilde{T}_{52})
\]

Xia, Chanrasekaran, Gu, Li

Fast Algorithms for HSS Matrices

April 27, 2017 22 / 31
Algorithm

\[
\begin{pmatrix}
D_1 & U_1 B_1 V_2^T & U_1 \tilde{T}_{14} & U_1 \tilde{T}_{15} \\
U_2 B_2 V_1^T & D_2 & U_2 \tilde{T}_{24} & U_2 \tilde{T}_{25} \\
\tilde{T}_{41} V_1^T & \tilde{T}_{42} V_2^T & D_4 & T_{45} \\
\tilde{T}_{51} V_1^T & \tilde{T}_{52} V_2^T & T_{54} & D_5
\end{pmatrix}
\]

- Compress block row/column for node 1
- Compress block row/column for node 2
Algorithm

\[
\begin{pmatrix}
D_1 & U_1B_1V_2^T & U_1\tilde{T}_{14} & U_1\tilde{T}_{15} \\
U_2B_2V_1^T & D_2 & U_2\tilde{T}_{24} & U_2\tilde{T}_{25} \\
\tilde{T}_{41}V_1^T & \tilde{T}_{42}V_2^T & D_4 & T_{45} \\
\tilde{T}_{51}V_1^T & \tilde{T}_{52}V_2^T & T_{54} & D_5
\end{pmatrix}
\]

- Compress block row/column for node 1
- Compress block row/column for node 2
- Identify and compress rows/columns 1 and 2

\[
\begin{pmatrix}
\tilde{T}_{14} & \tilde{T}_{15} \\
\tilde{T}_{24} & \tilde{T}_{25}
\end{pmatrix}
= \begin{pmatrix} R_1 \\ R_2 \end{pmatrix}
\begin{pmatrix}
\tilde{T}_{34} & \tilde{T}_{35}
\end{pmatrix}
\]
Algorithm

\[
\begin{pmatrix}
D_1 & U_1 B_1 V_2^T & U_1 \tilde{T}_{14} & U_1 \tilde{T}_{15} \\
U_2 B_2 V_1^T & D_2 & U_2 \tilde{T}_{24} & U_2 \tilde{T}_{25} \\
\tilde{T}_{41} V_1^T & \tilde{T}_{42} V_2^T & D_4 & T_{45} \\
\tilde{T}_{51} V_1^T & \tilde{T}_{52} V_2^T & T_{54} & D_5
\end{pmatrix}
\]

- Compress block row/column for node 1
- Compress block row/column for node 2
- Identify and compress rowss/columns 1 and 2

\[
\begin{pmatrix}
\tilde{T}_{41}^T & \tilde{T}_{51}^T \\
\tilde{T}_{42}^T & \tilde{T}_{52}^T
\end{pmatrix}
= \begin{pmatrix} W_1 \\ W_2 \end{pmatrix}
\begin{pmatrix} \tilde{T}_{43}^T & \tilde{T}_{53}^T \end{pmatrix}
\]
Algorithm

\[
\begin{pmatrix}
D_1 & U_1 B_1 V_2^T & U_1 \tilde{T}_{14} & U_1 \tilde{T}_{15} \\
U_2 B_2 V_1^T & D_2 & U_2 \tilde{T}_{24} & U_2 \tilde{T}_{25} \\
\tilde{T}_{41} V_1^T & \tilde{T}_{42} V_2^T & D_4 & T_{45} \\
\tilde{T}_{51} V_1^T & \tilde{T}_{52} V_2^T & T_{54} & D_5
\end{pmatrix}
\]

- Compress block row/column for node 1
- Compress block row/column for node 2
- Identify and compress rows/columns 1 and 2
- Continue compression
Summary of Algorithm

- Input: dense matrix, Output: HSS representation
- Compresses off-diagonal blocks
- Follows Postordering
- Ignoring basis matrices reduces cost
- Stable due to orthogonal transformations
- Time: $O(N^2)$ flops with a small HSS rank
- Smaller coefficient than one in [2]
- Other specific algorithms can be more efficient.
Fast Cholesky for SPD HSS Matrices

- Input: HSS form of SPD matrix, Output: HSS form Cholesky Factor
- First major operation: eliminating principal diagonal block
- Second major operation: updating the Schur complement
- Cost: $O(N^2)$ flops, where $H$ is dimension $N \times N$
Input: HSS form of SPD matrix, Output: explicit ULV factorization
First major operation: introduce zeros into off-diagonal blocks
Second major operation: partially factorize diagonal blocks
Third major operation: merge child blocks
Uses the postordering to eliminate nodes
Complexity: $O(r^2N)$
Good data locality
**System Solution**

- **Input:** generalized Cholesky HSS factorization, \( b \) vector,
- **Output:** Solution to \( Hx = b \)
- Forward substitution uses postordering
- Backward substitution uses reverse postordering
- **Cost:** \( O(rN) \)
References


