

Fast and Stable Algorithms for Hierarchically Semi-separable Representations

S. Chandrasekaran M. Gu T. Pals

September 30, 2002

1 Introduction

In this paper we consider a new class of structured dense matrices that we term **hierarchically semi-separable** (HSS). This class is a direct generalization of the class of matrices presented in the paper *Fast and Stable Algorithms for Sequentially Semi-separable Matrices*.

This class of matrices is characterized by a hierarchical low-rank structure in the off-diagonal blocks. Examples of such matrices are shown in Figure 1. The matrix in Figure 1.(a) is obtained for example for the matrix $[\log |x_i - x_j|]$, where $0 \leq x_i < x_{i+1} \leq 1$. Similarly the matrix in Figure 1.(b) is obtained for the matrix $[\log |\sin \pi(x_i - x_j)|]$. This class of matrices arise frequently in the numerical solution of partial differential and integral equations. Examples will be presented in a later section.

We begin by displaying the first block 4×4 leading principal sub-matrix of a block 8×8 example:

$$A = \begin{pmatrix} D_1 & U_{3;1}B_{3;1,2}V_{3;2}^H & U_{3;1}R_{3;1}B_{2;1,2}W_{3;3}^H V_{3;3}^H & U_{3;1}R_{3;1}B_{2;1,2}W_{3;4}^H V_{3;4}^H & \cdots \\ U_{3;2}B_{3;2,1}V_{3;1}^H & D_2 & U_{3;2}R_{3;2}B_{2;1,2}W_{3;3}^H V_{3;3}^H & U_{3;2}R_{3;2}B_{2;1,2}W_{3;4}^H V_{3;4}^H & \cdots \\ U_{3;3}R_{3;3}B_{2;2,1}W_{3;1}^H V_{3;1}^H & U_{3;3}R_{3;3}B_{2;2,1}W_{3;2}^H V_{3;2}^H & D_3 & U_{3;3}B_{3;3,4}V_{3;4}^H & \cdots \\ U_{3;4}R_{3;4}B_{2;2,1}W_{3;1}^H V_{3;1}^H & U_{3;4}R_{3;4}B_{2;2,1}W_{3;2}^H V_{3;2}^H & U_{3;4}B_{3;4,3}V_{3;3}^H & D_4 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}.$$

The fifth block column looks like this:

$$\begin{pmatrix} U_{3;1}R_{3;1}R_{2;1}B_{1;1,2}W_{2;3}^H W_{3;5}^H V_{3;5}^H \\ U_{3;2}R_{3;2}R_{2;1}B_{1;1,2}W_{2;3}^H W_{3;5}^H V_{3;5}^H \\ U_{3;3}R_{3;3}R_{2;2}B_{1;1,2}W_{2;3}^H W_{3;5}^H V_{3;5}^H \\ U_{3;4}R_{3;4}R_{2;2}B_{1;1,2}W_{2;3}^H W_{3;5}^H V_{3;5}^H \\ D_5 \\ \vdots \end{pmatrix}.$$

Here $U_{3;1}$, $R_{3;1}$, $B_{1;1,2}$, $W_{3;3}$, $V_{3;3}$, etc., are all matrices of potentially different sizes such that all necessary multiplications are defined. To describe this hierarchical structure it is convenient to use a binary tree. The tree describes in essence how the rows and columns of the matrix have been partitioned. To make

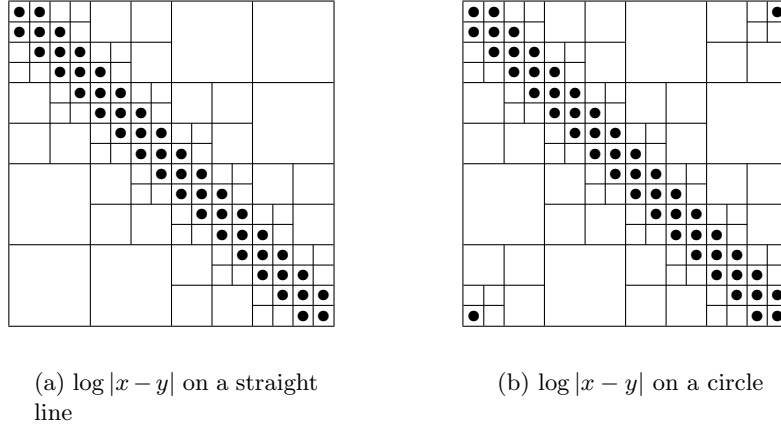


Figure 1: Submatrices labeled with a “•” are full-rank. All other submatrices are low-rank.

things concrete consider the block 8×8 HSS matrix, that we denote by A . Let m_i for $i = 1$ to 8 denote the partition sizes, and let $A_{3;i,j}$ denote the (i, j) -th block of the partition. (The 3 will be explained in a little while.) Note that $A_{3;i,j}$ is an $m_i \times m_j$ matrix. We will denote the square diagonal blocks by $A_{3;i,i} = D_i$. The off-diagonal blocks are described by a formula that is defined by an associated binary tree. We call this tree the **merge tree**. For this example we use the merge tree depicted in Figure 2. The tree is constructed as follows. There are eight leaves, one for each partition. More specifically, the first node is associated with the numbers $1, 2, \dots, m_1$, the second leaf is associated with the numbers $m_1 + 1, m_1 + 2, \dots, m_1 + m_2$, and so on. In general the i -th leaf (counting left to right) is associated with the m_i numbers $m_1 + \dots + m_{i-1} + 1, m_1 + \dots + m_{i-1} + 2, \dots, m_1 + \dots + m_{i-1} + m_i$. For convenience we use the notation $\nu_i = \sum_{j=1}^i m_j$. In this notation the i^{th} leaf is associated with numbers from $\nu_{i-1} + 1$ up to ν_i , with the understanding that $\nu_0 = 0$. Non-leaf nodes are associated with the numbers corresponding to their children. In particular the root node is associated with all the integers from 1 to $m_1 + m_2 + \dots + m_8$ (inclusive). To make it easier to refer to a particular node of the tree, we use a simple convention. We assume that the tree is organized in levels, with the root node in level 0, its children in level 1, and so on, with all the leaves, in this case, in level 3. Further we assume that all the nodes in a level are labeled with consecutive integers beginning from 1, from left to right. See Figure 2. We will use the notation $\text{Node}(k, i)$ to denote node i at level k .

Now we can explain the notation $A_{3;1,2}$ (the $(1, 2)$ block in the 8×8 partitioning of A). Note that this sub-matrix has rows whose indices belong to node 1 on level 3, and it has columns whose indices belong to node 2 on level 3. Similarly $A_{3;i,j}$ has rows whose indices belong to the node i on level 3, and has columns whose indices belong to node j on level 3.

Now, more generally, we can define $A_{k;i,j}$ as being the (i, j) block of the $2^k \times 2^k$ partitioning of the matrix A , where the rows that belong to the i^{th} row partition have indices that belong to node i at level k , and the columns that belong to the j^{th} column partition have indices that belong to node j at level k . In particular observe that

$$A_{1;1,2} = \begin{pmatrix} A_{2;1,3} & A_{2;1,4} \\ A_{2;2,3} & A_{2;2,4} \end{pmatrix}.$$

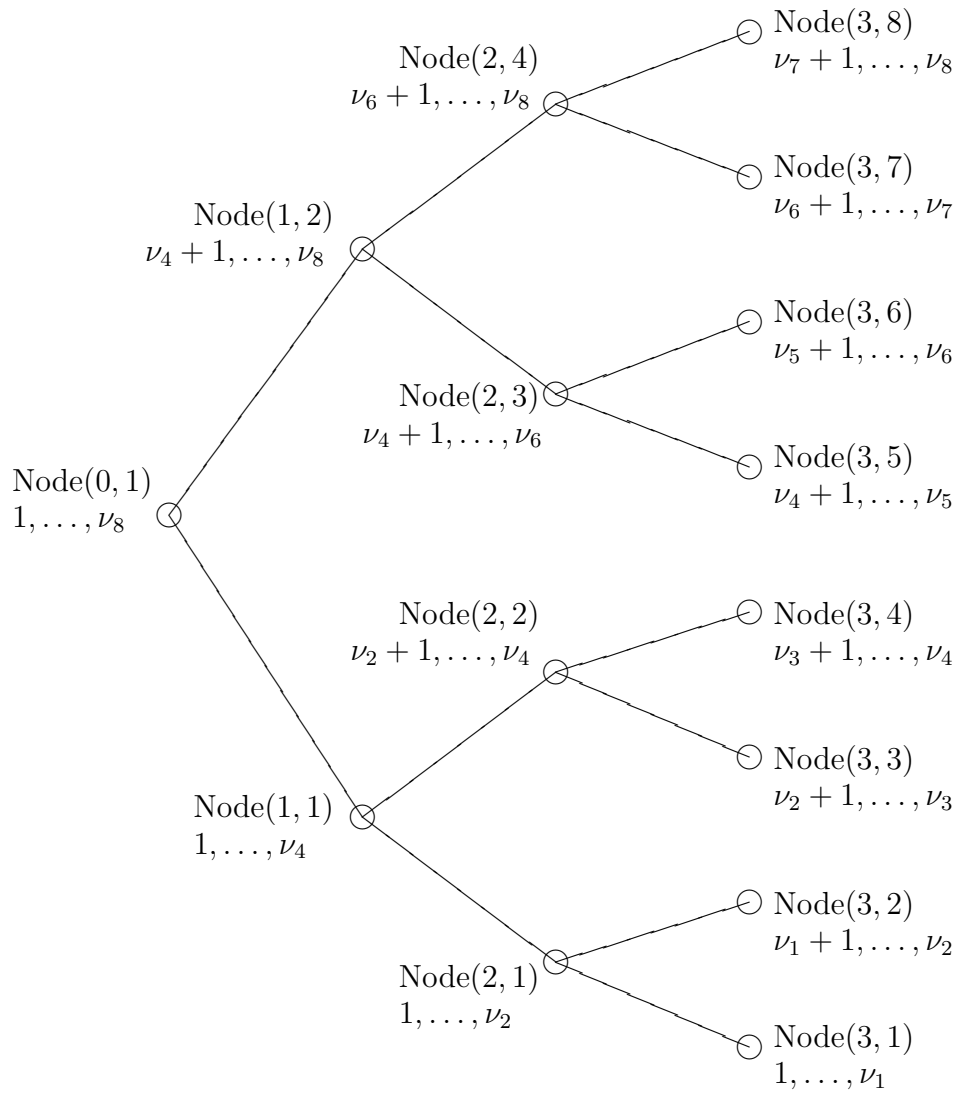


Figure 2: Merge tree showing the pairs of numbers, $\text{Node}(k, i)$, used to uniquely identify each node. Also shown are the row and column indices that are associated with each node.

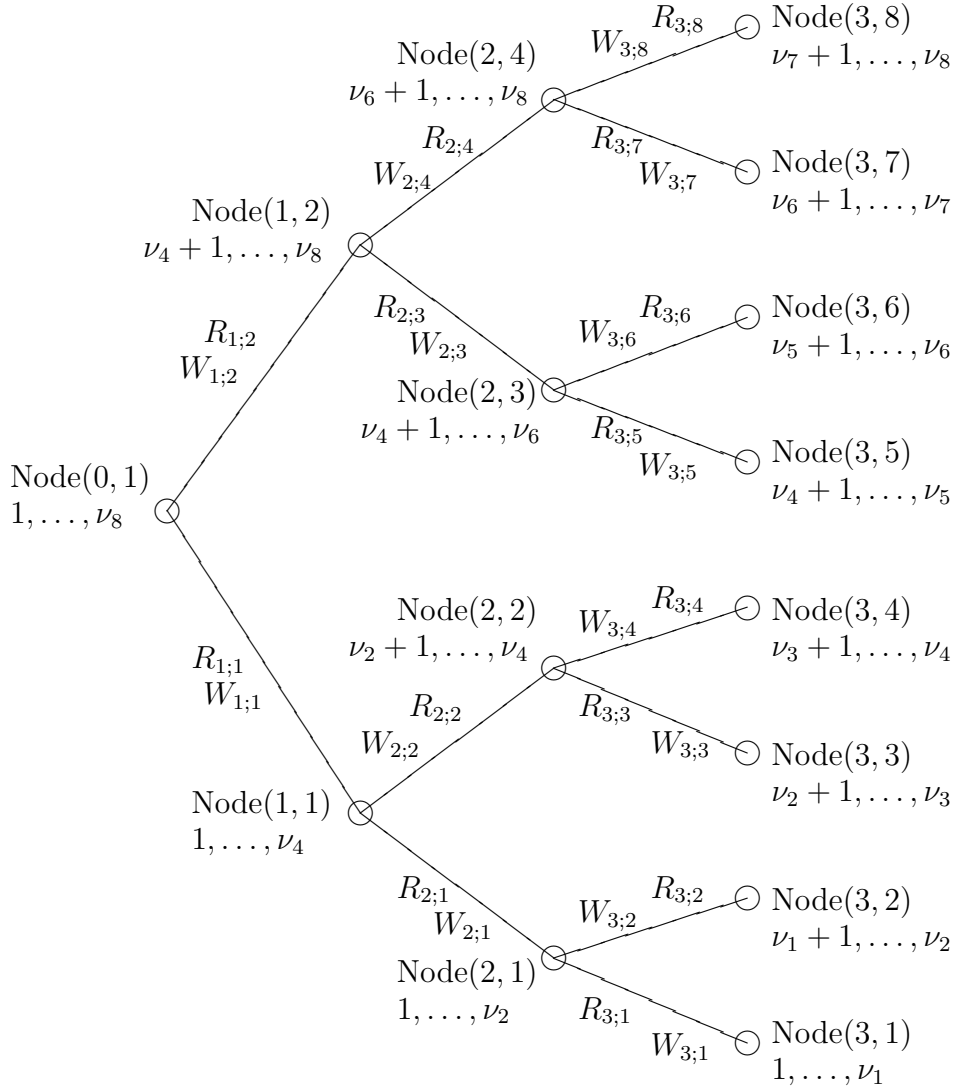


Figure 3: Merge tree of Figure 2 showing the matrices $R_{k;i}$ and $W_{k;i}$ associated with each edge of the tree.

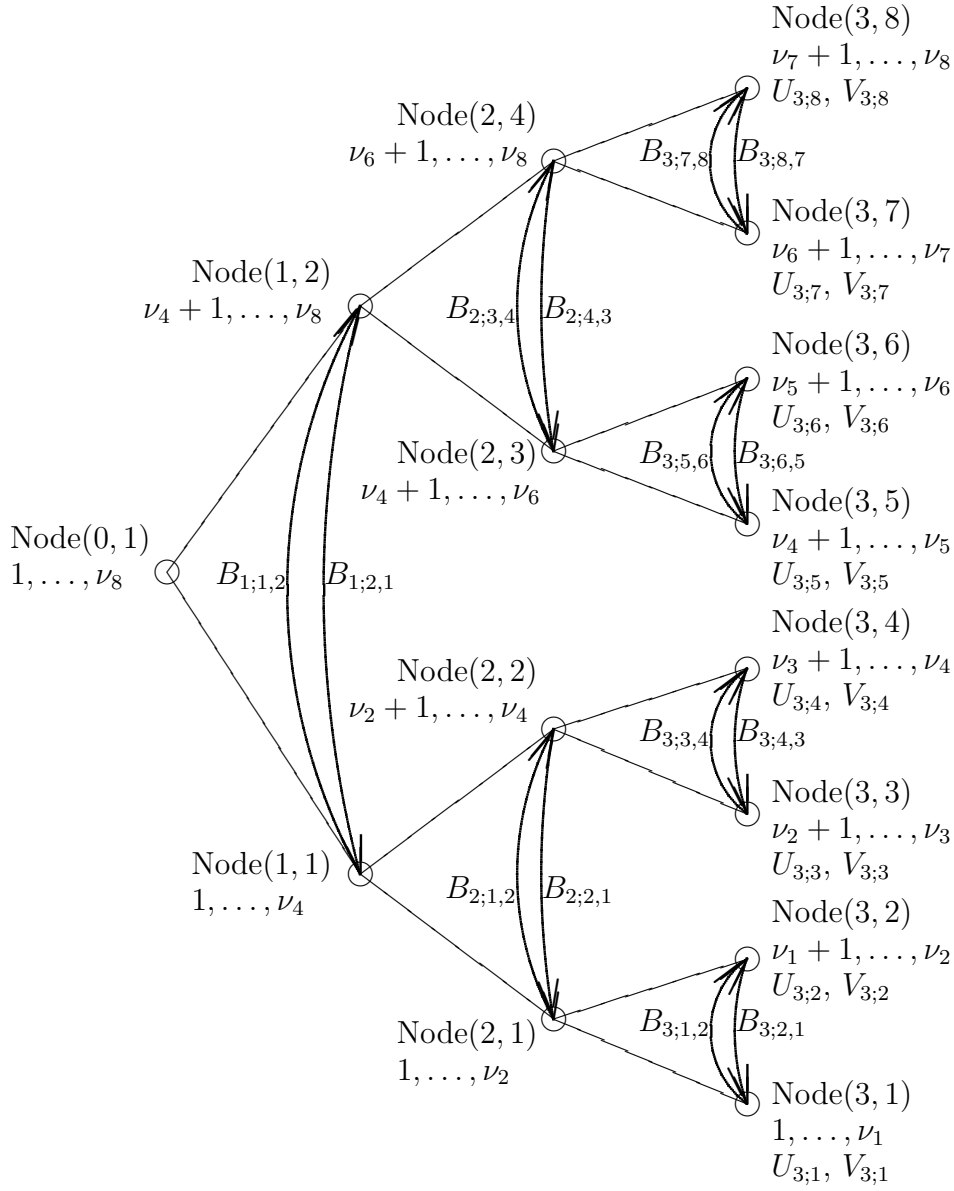


Figure 4: Merge tree of Figure 2 showing the directed edges between siblings and associated matrices $B_{k;i,j}$.

More generally we have that

$$A_{k;i,j} = \begin{pmatrix} A_{k+1;2i-1,2j-1} & A_{k+1;2i-1,2j} \\ A_{k+1;2i,2j-1} & A_{k+1;2i,2j} \end{pmatrix}.$$

That is why we refer to this tree as a **merge** tree.

Now for the matrix A to have an HSS representation we need to specify some matrices that are associated (or indexed if you will) with the merge tree. In particular, let us denote the edge between node i on level k and node $\lceil i/2 \rceil$ on level $k-1$ by $E(k, i)$. Then to each edge $E(k, i)$, of the merge tree we associate two matrices, $R_{k;i}$ and $W_{k;i}$ (see Figure 3). Also to each leaf i on level K (where K is the maximum number of levels in our tree) we associate two matrices, $U_{K;i}$ and $V_{K;i}$ (see Figure 2). Next we add some directed edges between certain nodes of the tree. In particular we add an edge $E_{\mathbb{D}}(k, 2i-1, 2i)$, from node $2i-1$ at level k to node $2i$ at level k , and an edge $E_{\mathbb{D}}(k, 2i, 2i-1)$ from node $2i$ to node $2i-1$ at level k . To each of these directed edge[s] $E_{\mathbb{D}}(k, i, j)$ we associate the matrix $B_{k;i,j}$. See Figure 4. It is important to observe that directed edges only exist between siblings.

Looking back at our example 8×8 matrix we see that we have specified all necessary matrices.

Now we can specify how the formulas for the off-diagonal blocks are determined. Let us start with an example. Consider the sub-matrix $A_{3;4,7}$. To construct the expression for this sub-block we proceed as follows. We first find the ancestors of node 4 and node 7 (both at level 3) who are siblings. In this case it is node 1 and 2 at level 1. Note, that this pair of ancestors must lie on the same level and is unique. Now look at the path from node 4 at level 3 to its ancestor node 1 at level 1. It can be described by the sequence of nodes, Node(3, 4), Node(2, 2), Node(1, 1). Similarly the path from Node(3, 7) to Node(1, 2) is Node(3, 7), Node(2, 4), Node(1, 2). Looking at these two paths we can write down the formula for $A_{3;4,7}$ as follows:

$$A_{3;4,7} = U_{3;4} R_{3;4} R_{2;2} B_{1;1,2} W_{2;4}^H W_{3;7}^H V_{3;4}^H.$$

Note that the sequence of $R_{*;*}$'s is determined by following the path from Node(3, 4) to Node(1, 1), while the sequence of $W_{*;*}$'s is determined by following the path from Node(1, 2) to Node(3, 7). The matrix $B_{*;*}$ is determined by looking at the directed edge from the ancestor of the row node Node(3, 4) to the ancestor of the column node Node(3, 7).

We now state the more general rule by first noting that the ancestor of Node(k, i) is Node($k-1, \lceil i/2 \rceil$). We will also use the notation

$$\begin{aligned} f(m) &= \left\lceil \frac{m}{2} \right\rceil, \\ f^2(m) &= \left\lceil \frac{\lceil \frac{m}{2} \rceil}{2} \right\rceil, \\ &\vdots = \ddots \end{aligned}$$

Then we see that the chain of ancestor nodes of Node(k, i) is given by Node($k-1, f(i)$), Node($k-2, f^2(i)$), Node($k-3, f^3(i)$), \dots , Node(0, 1).

Now consider $A_{K;i,j}$. Let k be the smallest integer such that $f^k(i) + 1 = f^k(j) = 2m$ if $i < j$, or $f^k(j) + 1 = f^k(i) = 2m$ if $i > j$. Then it follows that nodes Node($K-k, 2m-1$) and Node($K-k, 2m$) are

siblings that are ancestors of $\text{Node}(K, \min(i, j))$ and $\text{Node}(K, \max(i, j))$ respectively. (Note, for example, that nodes $\text{Node}(3, 4)$ and $\text{Node}(3, 5)$ are not siblings. That is we need $f^k(\max(i, j))$ to be an even number.) Then we have that the HSS representation for $A_{K;i,j}$ with $i < j$ is given by

$$A_{K;i,j} = U_{K;i} R_{K;i} R_{K-1;f(i)} \cdots R_{K-k+1;f^{k-1}(i)} B_{K-k;f^k(i),f^k(j)} W_{K-k+1;f^{k-1}(j)}^H W_{K-k+2;f^k(j)}^H \cdots W_{K;j}^H V_{K;j}^H.$$

Or, more compactly as

$$A_{K;i,j} = U_{K;i} \left(\prod_{l=0}^{k-1} R_{K-l;f^l(i)} \right) B_{K-k;f^k(i),f^k(j)} \left(\prod_{l=k-1}^0 W_{K-l;f^l(j)}^H \right) V_{K;j}^H,$$

with the understanding that empty products are replaced by the identity matrix, and $f^0(m) = m$.

More specifically we will call the sequences $\{D_i\}_{i=1}^n$, $\{U_{K;i}\}_{i=1}^{2^K}$, $\{V_{K;i}\}_{i=1}^{2^K}$, $\{\{R_{k;i}\}_{i=1}^{2^k}\}_{k=0}^K$, $\{\{W_{k;i}\}_{i=1}^{2^k}\}_{k=0}^K$, $\{\{B_{k;2i-1,2i}\}_{i=1}^{2^{k-1}}\}_{k=0}^K$, $\{\{B_{k;2i,2i-1}\}_{i=1}^{2^{k-1}}\}_{k=0}^K$ as the HSS representation of the matrix A .

In this paper we present a fast algorithm for multiplying a matrix in HSS form with a vector, and a fast and stable algorithm for solving linear systems of equations where the coefficient matrix is given in HSS form. We also present an efficient algorithm to convert a general matrix into HSS form. We then present specialized fast algorithms to convert matrices specified by Green's functions into HSS form. We also discuss applications of these ideas to the numerical solution of differential and integral equations. We also contrast the HSS representation with the SSS representation.

In this paper we restrict ourselves to merge trees that are complete binary trees. In a future paper we will generalize our methods to incomplete binary trees. Then as a special case we will recover the fast algorithms for matrices with SSS representations.

2 Low-rank blocks

Before we introduce the fast algorithms we try to reveal the mystery behind the HSS representation. Observe that the $A_{1;1,2}$ can be written in the factored form:

$$A_{1;1,2} = \begin{pmatrix} U_{3;1} R_{3;1} R_{2;1} \\ U_{3;2} R_{3;2} R_{2;1} \\ U_{3;3} R_{3;3} R_{2;2} \\ U_{3;4} R_{3;4} R_{2;2} \end{pmatrix} B_{1;1,2} \begin{pmatrix} V_{3;5} W_{3;5} W_{2;3} \\ V_{3;6} W_{3;6} W_{2;3} \\ V_{3;7} W_{3;7} W_{2;4} \\ V_{3;8} W_{3;8} W_{2;4} \end{pmatrix}^H.$$

We can make this more intelligible by defining the quantities

$$U_{k;i} = \begin{pmatrix} U_{k+1;2i-1} R_{k+1;2i-1} \\ U_{k+1;2i} R_{k+1;2i} \end{pmatrix}, \quad (1)$$

and

$$V_{k;i} = \begin{pmatrix} V_{k+1;2i-1} W_{k+1;2i-1} \\ V_{k+1;2i} W_{k+1;2i} \end{pmatrix}, \quad (2)$$

where k varies from 0 up to $K - 1$. Then we can re-write the expression for $A_{1;1,2}$ as

$$A_{1;1,2} = U_{1;1} B_{1;1,2} V_{1;2}^H.$$

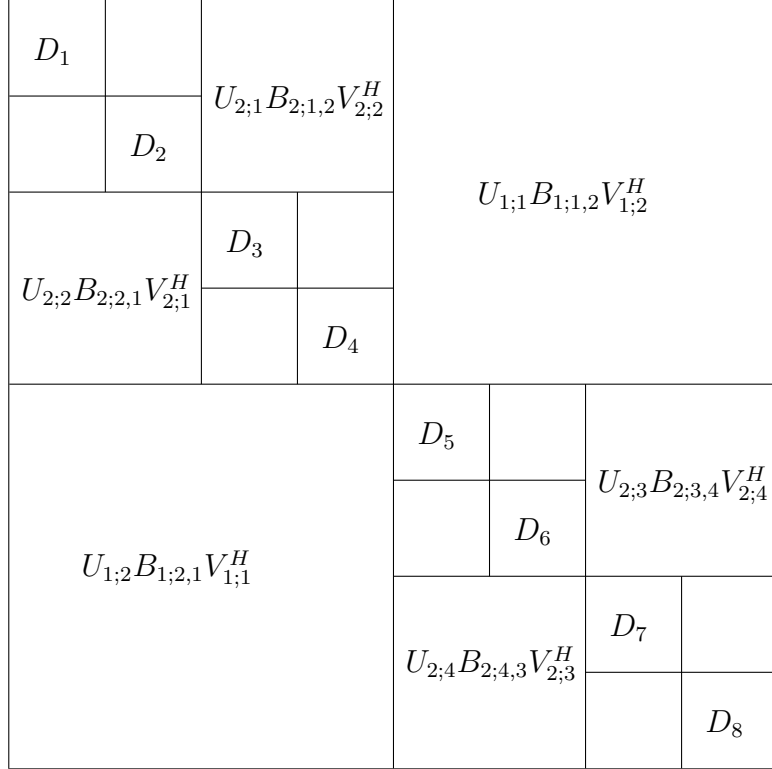


Figure 5: A different view of the HSS representation using the auxiliary matrices $U_{k;i}$ and $V_{k;i}$. These matrices are related to $U_{K;i}$ and $V_{K;i}$ via equations (1) and (2).

More generally every off-diagonal block $A_{k;2i-1,2i}$ can be written in the form

$$A_{k;2i-1,2i} = U_{k;2i-1}B_{k;2i-1,2i}V_{k;2i-1}^H, \quad (3)$$

and every off-diagonal block of the form $A_{k;2i,2i-1}$ can be written in the form

$$A_{k;2i,2i-1} = U_{k;2i-1}B_{k;2i,2i-1}V_{k;2i-1}^H. \quad (4)$$

These equations clearly reveal the meaning of $U_{k;i}$, $V_{k;j}$ and $B_{k;i,j}$. See Figure 5. The matrices $R_{k;i}$ and $W_{k;j}$ help construct the column and row bases respectively of off-diagonal blocks at level $k - 1$ from the column and rows bases of off-diagonal blocks at level k .

Since we place no restriction on the size of these matrices, it turns out that every matrix has an HSS representation. These representations are useful primarily when most of the $B_{k;i,j}$ matrices are of small size when compared to the size of the matrix A . We then say that A has a **short** HSS representation.

3 Fast multiplication

In this section we present a fast algorithm for multiplying a matrix in HSS form with a regular vector (or unstructured dense matrix). In particular consider the matrix-vector product $z = Ab$, where A is in HSS

form, with $K + 1$ levels in its merge tree, and m_i indices in $\text{Node}(K, i)$. Let $(b_{k;i})$ denote a block row partitioning of b such that $b_{k;i}$ has the rows whose indices belong to $\text{Node}(k, i)$. Similarly we partition z .

We begin by observing that we need to do the multiplication $A_{1;1,2}b_{1;2}$ at some point. Looking at this in expanded form using equation (3) we have that

$$A_{1;1,2}b_{1;2} = U_{1;1}B_{1;1,2}V_{1;2}^H b_{1;2}. \quad (5)$$

We then observe that we also need to do the multiplication

$$A_{2;3,4}b_{2;4} = U_{2;3}B_{2;3,4}V_{2;4}^H b_{2;4},$$

and that since

$$V_{1;2}^H b_{1;2} = \begin{pmatrix} V_{2;3}W_{2;3} \\ V_{2;4}W_{2;4} \end{pmatrix}^H \begin{pmatrix} b_{2;3} \\ b_{2;4} \end{pmatrix} = W_{2;3}^H V_{2;3}^H b_{2;3} + W_{2;4}^H V_{2;4}^H b_{2;4},$$

we can reduce the number of flops required to compute $V_{1;2}^H b_{1;2}$ in equation (5) if the number of columns in $W_{2;3}$ and $W_{2;4}$ are small compared to the number of rows in $V_{1;2}$. (This is the basis of all the super-fast algorithms and was first used by Greengard and Rokhlin in the design of fast multipole methods.)

To formalize this idea we define the intermediate quantities

$$G_{k;i} = V_{k;i}^H b_{k;i},$$

and observe that the following recursions, as deduced from the preceding discussion, are available to compute them

$$G_{K;i} = V_{K;i}^H b_{K;i}, \quad (6)$$

$$G_{k;i} = W_{k+1;2i-1}^H G_{k+1;2i-1} + W_{k+1;2i}^H G_{k+1;2i}. \quad (7)$$

With this notation we have that

$$A_{1;1,2}b_{1;2} = U_{1;1}B_{1;1,2}G_{1;2}.$$

We now observe that we need to perform the multiplication $U_{1;1}B_{1;1,2}G_{1;2}$. We also observe that we need to do the multiplication

$$A_{2;1,2}b_{2;2} = U_{2;1}B_{2;1,2}G_{2;2}.$$

Using equation (1) we see that

$$U_{1;1}B_{1;1,2}G_{1;2} = \begin{pmatrix} U_{2;1}R_{2;1}B_{1;1,2}G_{1;2} \\ U_{2;2}R_{2;2}B_{1;1,2}G_{1;2} \end{pmatrix}.$$

And therefore the computation of $A_{1;1,2}b_{1;2} = U_{1;1}B_{1;1,2}G_{1;2}$ can be merged with the computations of $A_{2;1,2}b_{2;2} = U_{2;1}B_{2;1,2}G_{2;2}$ when computing

$$z_{2;1} = \cdots + U_{2;1}(B_{2;1,2}G_{2;2} + R_{2;1}B_{1;1,2}G_{1;2}) + \cdots$$

where \cdots denotes other terms that have to be added to produce the correct $z_{2;1}$. Similarly the term $U_{2;2}R_{2;2}B_{2;1,2}G_{2;2}$ from the computation of $A_{1;1,2}b_{1;2}$ can be merged into other terms involving $U_{2;2}$ in the

computation of $z_{2,2}$. Clearly there is a recursive process here. This motivates us to define the following intermediate quantities recursively

$$F_{0,1} = 0, \tag{8}$$

$$F_{k,2i-1} = B_{k;2i-1,2i}G_{k;2i} + R_{k;2i-1}F_{k-1,i}, \tag{9}$$

$$F_{k,2i} = B_{k;2i,2i-1}G_{k;2i-1} + R_{k;2i}F_{k-1,i}. \tag{10}$$

We then observe that

$$z_{K;i} = D_i b_{K;i} + U_{K;i} F_{K;i}.$$

With that we have described how to compute $z = Ab$ rapidly when A has an HSS representation. Equations (6) and (7) are called the up-sweep recursions, and equations (8), (9), and (10) are called the down-sweep recursions for multiplication.

4 Fast backward stable solver

In this section we present our fast solver. The algorithm we describe computes a ULV^H decomposition implicitly, where U and V are unitary matrices, and L is a lower-triangular matrix. By implicit, we mean that the factors are not computed and stored explicitly. However, the algorithm and techniques can be modified to compute the factors explicitly if so desired. That will be the subject of a future paper. The algorithm can also be easily modified to permit U and V to be represented as a product of elementary Gauss transforms and permutation matrices. This would lead to a more efficient algorithm with a small probability of numerical instability.

The basic idea of the algorithm is akin to that for the SSS representation. The one major difference is that we operate on all block rows at the same time, whereas in the SSS representation, each block row was operated on in a sequential fashion.

The algorithm is recursive in nature, and the recursion takes one of three possible forms.

4.1 Compressible off-diagonal blocks

This is the first possible way in which the recursion can proceed.

We begin by observing that block row i , excluding the diagonal block D_i , has its column space spanned by the columns of $U_{K;i}$. Hence if the number of columns of $U_{K;i}$, denoted by $n_{K;i}$ is strictly smaller than m_i , the number of rows in that block, we can find a unitary transformation $q_{K;i}$ such that

$$\bar{U}_{K;i} = q_{K;i}^H U_{K;i} = \begin{matrix} m_i - n_{K;i} \\ n_{K;i} \end{matrix} \begin{pmatrix} 0 \\ \hat{U}_{K;i} \end{pmatrix}.$$

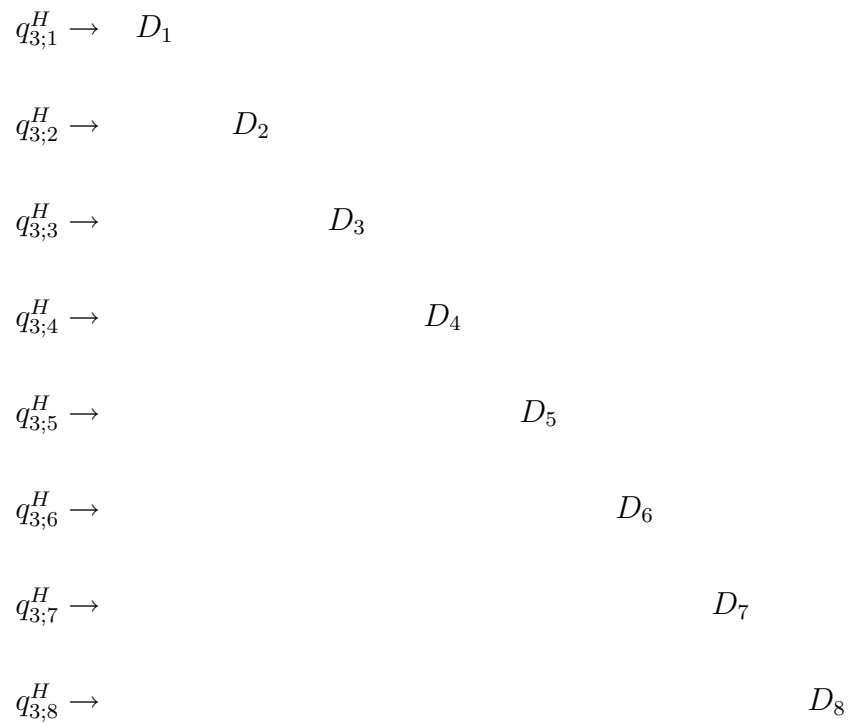


Figure 6: A pictorial representation showing the $q_{K;i}$'s compressing the off-diagonal portions of each block row. The black rectangles and triangles show the non-zero positions in the column bases of each off-diagonal block after compression by the $q_{K;i}$'s.

$$\begin{array}{cccccccc}
w_{3;1}^H & w_{3;2}^H & w_{3;3}^H & w_{3;4}^H & w_{3;5}^H & w_{3;6}^H & w_{3;7}^H & w_{3;8}^H \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow
\end{array}$$

Figure 7: A pictorial representation showing the $w_{K;i}$'s lower triangularizing the diagonal blocks after the compression of the off-diagonal blocks by the $q_{K;i}$'s (see figure 6). The off-diagonal portions of each block row. The black rectangles and triangles show the non-zero positions in the column bases and the diagonal blocks.

We now multiply block row i by q_i^H . (See figure 6.) The change in the off-diagonal blocks is represented by the above equation since all of them have $U_{K;i}$ as the leading term. The i -th block of the right-hand side changes to become

$$q_{K;i}^H b_{K;i} = \begin{pmatrix} m_i - n_{K;i} & \beta_{K;i} \\ n_{K;i} & \gamma_{K;i} \end{pmatrix}.$$

We also observe that D_i , the diagonal block has become $q_{K;i}^H D_i$. Now we pick a unitary transformation $w_{K;i}$ such that

$$\bar{D}_{K;i} = (q_{K;i}^H D_i) w_{K;i}^H = \begin{pmatrix} m_i - n_{K;i} & n_{K;i} \\ n_{K;i} & 0 \end{pmatrix} \begin{pmatrix} D_{i;1,1} & 0 \\ D_{i;2,1} & D_{i;2,2} \end{pmatrix}.$$

We then multiply the block column i from the right by $w_{K;i}^H$. (See figure 7.) The change in the diagonal block is represented by the above equation. The off-diagonal blocks in block column i have $V_{K;i}^H$ as the

common last term. Hence we just need to multiply $V_{K;i}$ to obtain

$$\bar{V}_{K;i} = w_{K;i} V_{K;i} = \begin{matrix} m_i - n_{K;i} \\ n_{K;i} \end{matrix} \begin{pmatrix} \check{V}_{K;i} \\ \hat{V}_{K;i} \end{pmatrix}.$$

Since we multiplied block column i from the right by $w_{K;i}^H$, we need to replace the unknowns $x_{K;i}$ by $w_{K;i} x_{K;i}$:

$$w_{K;i} x_{K;i} = \begin{matrix} m_i - n_{K;i} \\ n_{K;i} \end{matrix} \begin{pmatrix} z_{K;i} \\ \hat{x}_{K;i} \end{pmatrix}. \quad (11)$$

At this stage the first $m_i - n_{K;i}$ equations in block row i read as follows

$$D_{i;1,1} z_{K;i} = \beta_{K;i},$$

which can be solved for $z_{K;i}$ to obtain $z_{K;i} = D_{i;1,1}^{-1} \beta_{K;i}$. We now need to multiply the first $m_i - n_{K;i}$ columns in the block column i by $z_{K;i}$ and subtract it from the right-hand side. To do this efficiently we observe that the system of equations has been transformed as follows

$$(\text{diag}(q_{K;i}^H) A \text{diag}(w_{K;i}^H)) (\text{diag}(w_{K;i}) x) = \text{diag}(q_{K;i}^H) b.$$

If we define the vector

$$\bar{z}_{K;i} = \begin{matrix} m_i - n_{K;i} \\ n_{K;i} \end{matrix} \begin{pmatrix} z_{K;i} \\ 0 \end{pmatrix}$$

we then observe that the stated subtraction can be re-written as follows

$$\bar{b} = \text{diag}(q_{K;i}^H) b - (\text{diag}(q_{K;i}^H) A \text{diag}(w_{K;i}^H)) \bar{z}.$$

We can do this operation rapidly by observing that

$$(\text{diag}(q_{K;i}^H) A \text{diag}(w_{K;i}^H))$$

has the HSS representation $\{\bar{D}_i\}_{i=1}^{2^K}$, $\{\bar{U}_{K;i}\}_{i=1}^{2^K}$, $\{\bar{V}_{K;i}\}_{i=1}^{2^K}$, $\{\{R_{k;i}\}_{i=1}^{2^k}\}_{k=0}^K$, $\{\{W_{k;i}\}_{i=1}^{2^k}\}_{k=0}^K$, $\{\{B_{k;2i-1,2i}\}_{i=1}^{2^{k-1}}\}_{k=0}^K$, $\{\{B_{k;2i,2i-1}\}_{i=1}^{2^{k-1}}\}_{k=0}^K$, and using the fast multiplication algorithm in section 3. Of course, the algorithm can (and should) be modified to take advantage of the zeros in $\bar{D}_{K;i}$, $\bar{U}_{K;i}$, and $\bar{z}_{K;i}$.

Once the subtraction has been done, we discard the first $m_i - n_{K;i}$ columns of block column i and the first $m_i - n_{K;i}$ rows of block row i . We observe that this leads to a new system of equations of the form

$$\hat{A} \hat{x} = \hat{b},$$

where

$$\bar{b}_{K;i} = \begin{matrix} m_i - n_{K;i} \\ n_{K;i} \end{matrix} \begin{pmatrix} * \\ \hat{b}_{K;i} \end{pmatrix},$$

and \hat{A} has the HSS representation $\{D_{i;2,2}\}_{i=1}^{2^K}$, $\{\hat{U}_{K;i}\}_{i=1}^{2^K}$, $\{\hat{V}_{K;i}\}_{i=1}^{2^K}$, $\{\{R_{k;i}\}_{i=1}^{2^k}\}_{k=0}^K$, $\{\{W_{k;i}\}_{i=1}^{2^k}\}_{k=0}^K$, $\{\{B_{k;2i-1,2i}\}_{i=1}^{2^{k-1}}\}_{k=0}^K$, $\{\{B_{k;2i,2i-1}\}_{i=1}^{2^{k-1}}\}_{k=0}^K$.

Therefore we are left with a system of equations identical to the one we started with and we can proceed to solve it recursively. Once we have done that we can recover the unknowns x from z and \hat{x} using the formulas

$$x_{K;i} = w_{K;i}^H \begin{pmatrix} z_{K;i} \\ \hat{x}_{K;i} \end{pmatrix}.$$

Note, that we have tacitly assumed that all block rows are such that $m_i > n_{K;i}$. However, it is easy to modify the equations so that only those block rows that satisfy $m_i > n_{K;i}$ have their off-diagonal blocks compressed.

4.2 Incompressible off-diagonal blocks

This is the second possibility for the recursion. It occurs when *all* block rows for the system cannot be compressed any further by invertible transformations from the left. In this case we proceed to merge block rows and columns that correspond to siblings in the merge tree. In particular consider the first two block rows:

$$\begin{pmatrix} D_{1;2,2} & U_{3;1}B_{3;1,2}V_{3;2}^H & U_{3;1}R_{3;1}B_{2;1,2}W_{3;3}^H V_{3;3}^H & U_{3;1}R_{3;1}B_{2;1,2}W_{3;4}^H V_{3;4}^H & \cdots \\ U_{3;2}B_{3;2,1}V_{3;1}^H & D_{2;2,2} & U_{3;2}R_{3;2}B_{2;1,2}W_{3;3}^H V_{3;3}^H & U_{3;2}R_{3;2}B_{2;1,2}W_{3;4}^H V_{3;4}^H & \cdots \end{pmatrix}.$$

We observe that these two block rows can be re-written as

$$\left(\begin{pmatrix} D_{1;2,2} & U_{3;1}B_{3;1,2}V_{3;2}^H \\ U_{3;2}B_{3;2,1}V_{3;1}^H & D_{2;2,2} \end{pmatrix} \begin{pmatrix} (U_{3;1}R_{3;1}) \\ (U_{3;2}R_{3;2}) \end{pmatrix} B_{2;1,2} \begin{pmatrix} (V_{3;3}W_{3;3})^H \\ (V_{3;4}W_{3;4})^H \end{pmatrix} \right) \cdots$$

This immediately suggests that we merge as follows:

$$\begin{aligned} \hat{D}_i &= \begin{pmatrix} D_{2i-1;2,2} & U_{K;2i-1}B_{K;2i-1,2i}V_{K;2i}^H \\ U_{K;2i}B_{K;2i,2i-1}V_{K;2i-1}^H & D_{2i;2,2} \end{pmatrix}, \\ \hat{U}_{K-1;i} &= \begin{pmatrix} U_{K;2i-1}R_{K;2i-1} \\ U_{K;2i}R_{K;2i} \end{pmatrix}, \\ \hat{V}_{K-1;i} &= \begin{pmatrix} V_{K;2i-1}W_{K;2i-1} \\ V_{K;2i}W_{K;2i} \end{pmatrix}. \end{aligned}$$

We then see that A has an HSS representation (with a merge tree with K , as opposed to $K+1$, levels) given by the sequences: $\{\hat{D}_i\}_{i=1}^{2^{K-1}}$, $\{\hat{U}\}_{i=1}^{2^{K-1}}$, $\{\hat{V}\}_{i=1}^{2^{K-1}}$, $\{\{R_{k;i}\}_{i=1}^{2^k}\}_{k=0}^{K-1}$, $\{\{W_{k;i}\}_{i=1}^{2^k}\}_{k=0}^{K-1}$, $\{\{B_{k;2i-1,2i}\}_{i=1}^{2^{k-1}}\}_{k=0}^{K-1}$, $\{\{B_{k;2i,2i-1}\}_{i=1}^{2^{k-1}}\}_{k=0}^{K-1}$. Let us denote by \hat{A} the matrix with this HSS representation (of course, $A = \hat{A}$). We then observe that the system of equation is now in the form

$$\hat{A}x = b \tag{12}$$

which is exactly in the form we started with, except that the new HSS representation has only K levels in the merge tree. Hence we can solve this system of equations recursively for x .

4.3 No off-diagonal blocks

Observe that if $K = 0$ the equations read $D_1x = b$, which can be solve by traditional means for x . This case terminates the recursion.

With this we have given a complete account of the algorithm.

4.4 Flop count

We use the flop counts in table 1 of the basic matrix operations that can be found for example in [?].

Operation	Flops
QL factorization of skinny $m \times n$ matrix	$2n^2(m - n/3)$
Q times $m \times k$ matrix	$2kn(2m - n)$
Forward substitution of $n \times n$ matrix with k right-hand sides	n^2k
$m \times n$ times $n \times k$ matrix	$2mnk$

Table 1: Flop counts of basic matrix operations.

We first begin by estimating the flop count for the fast multiplication algorithm as that is an integral part of the solver. For simplicity we will assume that the ranks $n_{k;i}$ are independent of i , and that there are l indices in each of the leaves of the merge tree.

Computing $G_{K;i}$ will cost us $2ln_Kr2^K$ flops, where r is the number of columns in the right-hand side. Computing $G_{k;i}$ from $G_{k+1;i}$ costs $4n_kn_{k+1}r2^k$ flops. Computing $F_{k+1;i}$ from $F_{k;i}$ costs $42^{k+1}n_kn_{k+1}r$ flops. Finally computing $z_{K;i}$ costs $2lr(l + n_K)2^K$ flops. Summing these costs over k we obtain

$$2lr(l + n_K)2^K + 2ln_Kr2^K + 8r \sum_{k=1}^{K-1} n_kn_{k+1}2^k,$$

as the total cost. Letting $N = 2^Kl$ be the order of the matrix we can simplify this to obtain

$$2Nr(l + 2n_K) + 8r \sum_{k=1}^{K-1} n_kn_{k+1}2^k,$$

as the number flops for the fast multiplication algorithm

We now proceed to estimate the flops for the fast backward stable solver. To keep the calculations simple we will assume that each level of the tree undergoes a compression step before going through a merge step. We will also assume that $m_{k;i}$, the size of the block-rows at the k -th stage is independent of i .

Let us start with the compression step. We first need to compute the QL factorizations of $U_{k;i}$. This will cost us $2n_k^2(m_k - n_k/3)2^k$ flops. Then we need to apply $q_{k;i}$ to the right-hand side. This costs us $2rm_k(2m_k - n_k)2^k$ flops. We also need to apply $q_{k;i}$ to D_i (at the k -th level), which costs us $2m_kn_k(2m_k - n_k)2^k$ flops. Next the LQ factorization of the diagonal blocks costs us $4m_k^32^k/3$ flops. Applying $w_{k;i}$ to $V_{k;i}$ costs $2n_km_k^22^k$ flops. The partial forward-substitution at level k costs $(m_k - n_k)^2r2^k$ flops. Subtracting the computed unknowns from the right-hand side costs

$$2^k2m_kr(m_k + 2n_k) + 8r \sum_{s=1}^{k-1} n_s n_{s+1} 2^s,$$

flops. Recovering $x_{k;i}$ from $z_{k;i}$ will cost $2rm_k^2 2^k$ flops. That completes the compression stage.

For the merge step, forming the new diagonal blocks costs $8m_k n_k^2 2^k$ flops. Merging $U_{k;i}$ and $V_{k;i}$ costs $8m_k n_k^2 2^k$ flops.

Therefore the total cost of the fast backward stable solver is

$$\sum_{k=1}^K 2n_k^2(m_k - n_k/3)2^k + 2rm_k(2m_k - n_k)2^k + 2m_k n_k(2m_k - n_k)2^k + 4m_k^3 2^k/3 + 2n_k m_k^2 2^k + (m_k - n_k)^2 r 2^k + 2^k 2m_k r(m_k - n_k)$$

which can be simplified to

$$\sum_{k=1}^K 2^k \left(\frac{4}{3}m_k^3 + 6m_k^2 n_k - \frac{2}{3}n_k^3 + r \left(7m_k^2 + n_k^2 + 8 \sum_{s=1}^{k-1} n_s n_{s+1} 2^s \right) \right).$$

The terms not involving r can be thought of as the cost of factorization.

We now observe that under our assumptions $m_k = 2n_{k+1}$ for $k < K$. Making this substitution we can simplify the count to

$$2^K m_K^2 \left(\frac{4}{3}m_K + 6n_K \right) + 24 \sum_{k=1}^{K-1} 2^k n_{k+1}^2 n_k + \frac{14}{3} \sum_{k=2}^K 2^k n_k^3 - \frac{4}{3}n_1^3 + r \left(7m_K^2 2^K + 15 \sum_{k=2}^K 2^k n_k^2 + n_1^2 + 8 \sum_{k=1}^K \sum_{s=1}^{k-1} 2^s n_s n_{s+1} \right)$$

To simplify further we assume that $n_k \geq n_{k+1}$. Then we can get an upper bound on the flop count

$$2^K m_K^2 \left(\frac{4}{3}m_K + 6n_K \right) + \frac{86}{3} \sum_{k=1}^K 2^k n_k^3 - \frac{4}{3}n_1^3 + r \left(7m_K^2 2^K + 15 \sum_{k=2}^K 2^k n_k^2 + n_1^2 + 8 \sum_{k=1}^K \sum_{s=1}^{k-1} 2^s n_s^2 \right).$$

We now compute the flop counts for some canonical examples. First we consider the case when $n_k = p$, a constant. In this case the upper-bound on the flop count simplifies to

$$2^K m_K^2 \left(\frac{4}{3}m_K + 6p \right) + \frac{86}{3}p^3 2^{K+1} + r (7m_K^2 2^K + 23p^2 2^{K+1}).$$

Using $N = 2^K m_K$, and assuming that $m_K = 2p$, we get,

$$46Np^2 + 37Npr.$$

As can be seen, the constants are modest. By switching to Gauss transforms rather than Householder transforms we can reduce the constants even further.

In many cases this flop count is sufficient to give an indication of the performance of the algorithm. However, for theoretical purposes we also provide an upper bound on the flop count under the assumption that $n_k \leq \gamma^k n_0$. This model is useful when applying the algorithm to matrices of the form $A_{ij} = f(x_i, x_j)$, when the points x_i lie in high dimensional spaces.

For example, when $f(x_i, x_j) = \log|x_i - x_j|$, and x_i is a point in the two-dimensional plane, we can take $n_0 = \alpha N^{\frac{1}{2}}$ and $\gamma = \frac{1}{\sqrt{2}}$. For there to be any speed-up possible at all we must have that

$$\alpha \leq \frac{N^{\frac{1}{2}}}{\sqrt{2}}.$$

For simplicity, and since it is common in practice, we assume that $\alpha \geq 1$. We then observe that $m_k = N2^{-k} \geq 2\alpha N^{1/4}2^{-k/2}$, provided $k < \log_2 N - 2(\log_2 \alpha + 1)$. Hence we take the depth of the tree to be

$$K = \lfloor \log_2 N - 2\log_2 \alpha - 1 \rfloor.$$

Note that m_K is approximately $4\alpha^2$ in this scenario. Under these assumptions the flop count for the fast solver is not more than

$$98N^{\frac{3}{2}}\alpha^3 + 70N\alpha^4 + N\alpha^2 r(4\log_2^2 N + 11\log_2 N + 28).$$

As can be seen the constant is quite sensitive to the size of α . **What are typical values of α ?**

Next we consider three dimensional problems. For example, when $f(x_i, x_j) = \|x_i - x_j\|^{-1}$, and x_i is a point in three-dimensional space, we can take $n_0 = \alpha N^{\frac{2}{3}}$ and $\gamma = \frac{1}{\sqrt[3]{4}}$. To obtain any speed-up at all, we must ensure that $\alpha < (N/2)^{1/3}$. For the sake of simplicity we will also assume that $\alpha \geq 1$. We can determine the maximum depth of the tree from the constraint $m_k \geq 2n_k$, which yields

$$K \leq \lfloor \log_2 N - 3\log_2 \alpha - 1 \rfloor.$$

Under this scenario m_K is approximately $2\alpha^3$. With these assumptions the flop count for the fast solver is less than

$$58N^2\alpha^3 + 18N\alpha^6 + r(39N^{\frac{4}{3}}\log_2 N\alpha^2 + 74N^{\frac{4}{3}}\alpha^2 + 14N\alpha^3).$$

As can be seen the constant is modest.

Observe that in both cases the fast dense solver matches the asymptotic complexity of the corresponding sparse direct finite-element and finite-difference solvers of the same dimension. Of course, many times the integral equations corresponding to a particular PDE will be one dimension smaller, frequently yielding the advantage to the integral equation method. However, the quadratic dependence on N for three-dimensional problems makes this algorithm suitable only when the linear system is highly ill-conditioned and a suitable pre-conditioner is lacking. In fact, this solver can serve as an ideal pre-conditioner in this and other situations. Another situation where this method is suitable vene for three-dimensional problems is when there are a large number of right-hand sides.

We remark that if many of the leaves at level K are empty, then the algorithm we have specified will become inefficient. A more complicated algorithm that does not suffer from this deficiency will be presented in a future paper.

5 Computing the HSS representation

In this section we describe an $O(n^2)$ algorithm to compute the HSS representation of an arbitrary matrix to a given tolerance.

The key idea is to compute the singular value decomposition (SVD) of the matrices

$$H_{k;i} = (A_{k;i,1} \ A_{k;i,2} \ \cdots \ A_{k;i,i-1} \ A_{k;i,i+1} \ A_{k;i,i+2} \ \cdots \ A_{k;i,2^k}). \quad (13)$$

Notice that $H_{k;i}$ is essentially block row i of the matrix when partitioned according to level k of the merge tree, except that the diagonal block corresponding to that level $A_{k;i,i}$, is missing.

Similarly we also need to compute the SVD of the matrices

$$G_{k;i} = (A_{k;1,i}^H \ A_{k;2,i}^H \ \cdots \ A_{k;i-1,i}^H \ A_{k;i+1,i}^H \ A_{k;i+2,i}^H \ \cdots \ A_{k;2^k,i}^H)^H. \quad (14)$$

Suppose we have the SVD of $H_{k;i}$ and $G_{k;i}$ for $k = 1$ to K and for $i = 1$ to 2^k

$$\begin{aligned} H_{k;i} &= U_{k;i} C_{k;i} J_{k;i}^H, \\ G_{k;i} &= L_{k;i} M_{k;i} V_{k;i}^H. \end{aligned}$$

Observe that these equations directly define the auxiliary quantities $U_{k;i}$ and $V_{k;i}$ that appear in equations (1) and (2). In particular we obtain $U_{K;i}$ and $V_{K;i}$. Using equations (1) and (2) we can also compute

$$\begin{aligned} R_{k+1;2i-1} &= U_{k+1;2i-1}^H (U_{k;i})_1, \\ R_{k+1;2i} &= U_{k+1;2i}^H (U_{k;i})_2, \\ W_{k+1;2i-1} &= V_{k+1;2i-1}^H (V_{k;i})_1, \\ W_{k+1;2i} &= V_{k+1;2i}^H (V_{k;i})_2, \end{aligned}$$

where we have the conforming partitions

$$\begin{aligned} U_{k;i} &= \begin{pmatrix} (U_{k;i})_1 \\ (U_{k;i})_2 \end{pmatrix} = \begin{pmatrix} U_{k+1;2i-1} R_{k+1;2i-1} \\ U_{k+1;2i} R_{k+1;2i} \end{pmatrix}, \\ V_{k;i} &= \begin{pmatrix} (V_{k;i})_1 \\ (V_{k;i})_2 \end{pmatrix} = \begin{pmatrix} V_{k+1;2i-1} W_{k+1;2i-1} \\ V_{k+1;2i} W_{k+1;2i} \end{pmatrix}. \end{aligned}$$

This only leaves us with determining formulas for $B_{k;2i-1,2i}$ and $B_{k;2i,2i-1}$. Observe that $A_{k;2i-1,2i}$ is the $2i-1$ sub-matrix of $H_{k;2i-1}$ and $G_{k;2i}$, in the partitioning in equations (13) and (14). Therefore assuming that $(J_{k;2i-1})_{2i-1}$ and $(L_{k;2i})_{2i-1}$ denote the appropriate submatrices, we have that

$$A_{k;2i-1,2i} = U_{k;2i-1} B_{k;2i-1,2i} V_{k;2i}^H = U_{k;2i-1} C_{k;2i-1} (J_{k;2i-1})_{2i-1}^H = (L_{k;2i})_{2i-1} M_{k;2i} V_{k;2i}^H.$$

This immediately gives us the formulas

$$B_{k;2i-1,2i} = C_{k;2i-1} (J_{k;2i-1})_{2i-1}^H V_{k;2i} = U_{k;2i-1}^H (L_{k;2i})_{2i-1} M_{k;2i}.$$

Similarly $A_{k;2i,2i-1}$ is the $2i-1$ submatrix of $H_{k;2i}$ and $G_{k;2i-1}$ in the partitioning in equations (13) and (14). Therefore assuming that $(J_{k;2i})_{2i-1}$ and $(L_{k;2i-1})_{2i-1}$ denote the appropriate submatrices, we have that

$$A_{k;2i,2i-1} = U_{k;2i} B_{k;2i,2i-1} V_{k;2i-1}^H = U_{k;2i} C_{k;2i} (J_{k;2i})_{2i-1}^H = (L_{k;2i-1})_{2i-1} M_{k;2i-1} V_{k;2i-1}^H.$$

This immediately gives us the formulas

$$B_{k;2i,2i-1} = C_{k;2i} (J_{k;2i})_{2i-1}^H V_{k;2i-1} = U_{k;2i}^H (L_{k;2i-1})_{2i-1} M_{k;2i-1}.$$

All we need now is an efficient way to compute the needed singular value decompositions. To this end we observe that $H_{k;i}$ is closely related to $H_{k+1;2i-1}$ and $H_{k+1;2i}$. In fact, by dropping the $2i - 1$ block column from

$$\begin{pmatrix} H_{k+1;2i-1} \\ H_{k+1;2i} \end{pmatrix},$$

we obtain $H_{k;i}$. Similarly, by dropping the $2i - 1$ block row from

$$(G_{k+1;2i-1} \quad G_{k+1;2i}),$$

we obtain $G_{k;i}$. Hence we can obtain the SVD of $H_{k;i}$ efficiently from the SVDs of $H_{k+1;2i-1}$ and $H_{k+1;2i}$. Similarly, for $G_{k;i}$.

Assuming that $B_{k;2i-1,2i}$ is an $n_{k;i} \times n_{k;i}$ matrix and $B_{k;2i,2i-1}$ is an $p_{k;i} \times p_{k;i}$ matrix, the complexity of the above algorithm is $O(N(N + \sum_{k;i}(n_{k;i}^2 + p_{k;i}^2)))$.

5.1 Smooth Matrices

When the matrix entry A_{ij} is specified by a function, $f(x_i, x_j)$ that is smooth away from the diagonal, the HSS representation can be computed more rapidly than in the general case. In this section we consider the special case when the points x_i lie on the real line. The more general case is beyond the scope of this paper. Important examples of the function $f(x, y)$ include $\log \|g(x) - g(y)\|$ and $\|g(x) - g(y)\|^\alpha$, where $g : \mathcal{R} \rightarrow \mathcal{R}^d$, represents a simple closed or non-closed curve in d -dimensional space.

Since we are restricting ourselves to uniform HSS representations in this paper, we will assume that the points x_i are distributed uniformly in the interval $[0, 1]$. Note, this does not mean that the points x_i are equi-spaced. Furthermore, for simplicity, we will assume the function f has at most singularities at 0 and 1, and that it is analytic away from these singularities. A good example to keep in mind is $f(x, y) = \log |x - y|$.

From the basic theory of polynomial approximation of such functions it follows that if $T_k(x)$ denotes the k -th Chebyshev polynomial

$$T_k(x) = \cos(k \arccos x), \quad -1 \leq x \leq 1,$$

and

$$\phi_{a,b} : [a, b] \rightarrow [-1, 1], \quad \phi_{a,b}(x) = -1 + 2 \frac{x - a}{b - a}.$$

denotes the affine-linear function that maps the interval $[a, b]$ to $[-1, 1]$, then on any rectangle $[a, b] \times [c, d]$ such that $a < b < c < d$, and $\min(d - c, b - a) > c - b$, we can find a short two-sided Chebyshev expansion of $f(x, y)$ to a given accuracy:

$$f(x, y) \approx \sum_{p,q} \beta_{p,q} T_p(\phi_{a,b}(x)) T_q(\phi_{c,d}(y)).$$

More specifically, the (i, j) -th entry of the matrix can be represented to a prescribed accuracy by a short expansion of the form

$$f(x_i, x_j) \approx \sum_{p,q} \beta_{p,q} T_p(\phi_{a,b}(x_i)) T_q(\phi_{c,d}(y_j)).$$

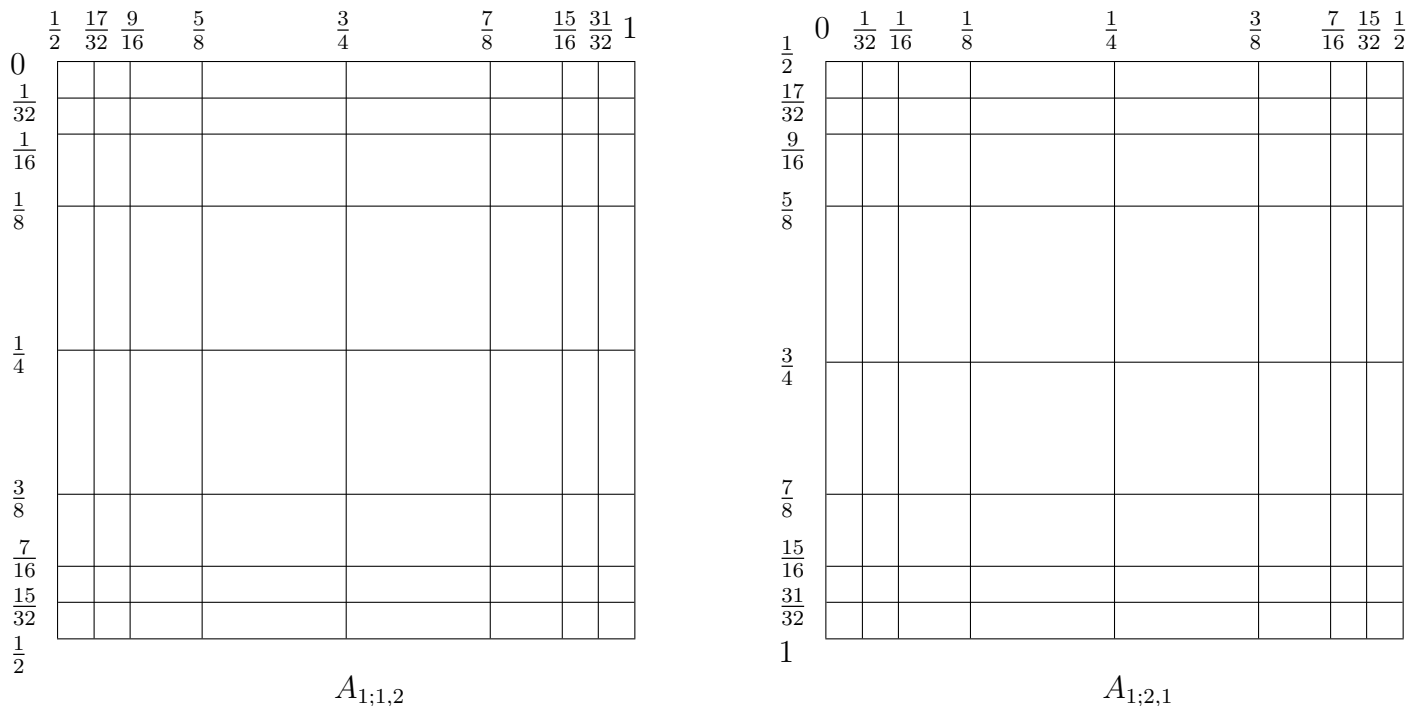


Figure 8: Block partitioning of $A_{1;1,2}$ and $A_{1;2,1}$ suitable for Chebyshev expansions. The vertical and horizontal lines are labelled according to the interval boundaries.

We shall now show how these expansion coefficients can be used to compute an HSS representation for the matrix quickly.

We first need to specify the merge tree we are going to use. We do so as follows. We will assume that all the points x_i lie in the interval $[0, 1]$. Hence we will associate the interval $[0, 1]$ (and hence all the points x_i , and hence all indices) with the root node. To the left child of the root we associate the interval $[0, 0.5]$ and with the right child the interval $[0.5, 1]$. This means, that we associate all points x_i in the interval $[0, 0.5]$ with the left child, and hence, all the corresponding indices with the left child. Similarly for the right child. To the left child of the left child of the root node, namely Node(2, 1) we associate the interval $[0, 0.25]$, to Node(2, 2) we associate the interval $[0.25, 0.5]$, and so on. In this way we assign the indices to the merge tree.

Note, that the number of indices in two different nodes at the same level can be different. Also note that we do not assume that the points x_i are equi-spaced.

Let us denote the set of points x_i that belong to Node(k, i) by $x_{k;i}$. Let us denote $\Gamma_{k;i}$ the Chebyshev-Vandermonde matrix evaluated at the points $x_{k;i}$. We will assume that the number of columns in $\Gamma_{k;i}$ is fixed at p to ease the exposition.

Each node of the merge tree is associated with a particular interval of the real line. In particular Node(k, i) is associated with the interval $2^{-k}[i - 1, i]$. Therefore it follows that $A_{k;i,j}$ is associated with the *rectangle* $2^{-k}[i - 1, i] \times 2^{-k}[j - 1, j]$.

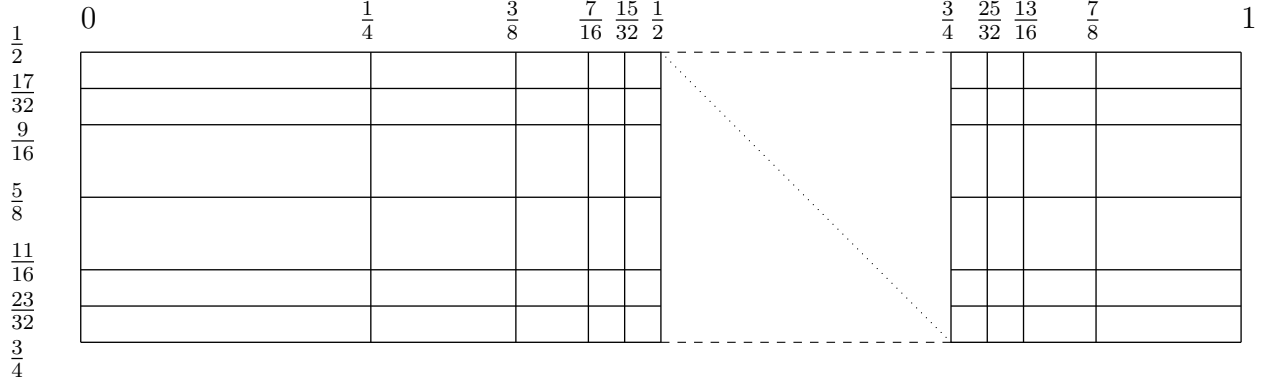


Figure 9: Block partitioning of $H_{2,3}$ suitable for Chebyshev expansions. The vertical and horizontal lines are labelled according to the associated interval boundaries. The missing block in the middle is the diagonal block. The dotted diagonal line shows the position of the diagonal.

Figure 8 displays a partitioning of $A_{1,1,2}$ and $A_{1,2,1}$ that will prove useful. We observe that each off-diagonal block, except possibly the bottom-left and upper-right blocks, in the displayed partition is associated with a rectangle on which the function f has a short two-sided Chebyshev expansion. However, note that the blocks are sometimes specified by intervals on two different levels of the merge tree. Hence we will use the notation $A_{(k;i),(r;j)}$ to denote the sub-matrix whose row indices come from Node(k, i) and column indices come from Node(r, j). We shall also use the notation

$$A_{(k;i),(r;j)} = \Gamma_{k;i} C_{(k;i),(r;j)} \Gamma_{r;j}^H, \quad (15)$$

for the corresponding two-sided Chebyshev expansion. Observe that $C_{(k;i),(r;j)}$ can be computed in time independent of the size of $A_{(k;i),(r;j)}$. Since the block $A_{K;i,i+1}$ does not necessarily have a short two-sided Chebyshev expansion, we will assume instead that it has fewer than p rows and columns, in which case it trivially has an expansion of the form (15).

To construct the HSS representation we remind the reader that it is the low-rank expansions of $H_{k;i}$ and $G_{k;i}$ that are crucial. Hence in figure 9 we show the partitioning of $H_{2,3}$ that we will use. Now observe that we can construct a low rank expansion for $A_{k;i,i+1}$ and $A_{k;i+1,i}$, for odd i , as follows. Let

$$\Delta_{k;i} = \text{diag} \begin{pmatrix} \Gamma_{K;2^{K-k}(i-1)+1} \\ \Gamma_{K;2^{K-k}(i-1)+2} \\ \vdots \\ \Gamma_{k+2;2(2i-1)} \\ \Gamma_{k+2;2(2i)-1} \\ \vdots \\ \Gamma_{K;2^{K-k}i-1} \\ \Gamma_{K;2^{K-k}i} \end{pmatrix}, \quad (16)$$

be a block-diagonal matrix. In the above notation we assume that $\Delta_{K;i} = \Gamma_{K;i}$, and

$$\Delta_{K-1;i} = \begin{pmatrix} \Gamma_{K;2i-1} \\ \Gamma_{K;2i} \end{pmatrix}.$$

Note that these formulas are consistent with (16). Then let

$$U_{k;i} = \Delta_{k;i},$$

$$V_{k;i} = \Delta_{k;i},$$

and let $B_{k;i,i+1}$ be the block matrix with block entries

$$(B_{k;i,i+1})_{r,s} = C_{(k_r;i_r),(k_s;(i+1)_s)},$$

where $\text{Node}(k_r, i_r)$ is the node corresponding to the r -th diagonal block of $\Delta_{k;i}$. Similarly we define

$$(B_{k;i+1,i})_{r,s} = C_{(k_r;(i+1)_r),(k_s;i_s)}.$$

All we have to specify now is $R_{k;i}$ and $W_{k;i}$. First observe that $R_{k;i} = W_{k;i}$ since $U_{k;i} = V_{k;i}$. From the definition of $\Delta_{k;i}$ observe that

$$\Delta_{k;i} = \begin{pmatrix} \Delta_{k+1;2i-1}\Omega_{k+1;2i-1} & 0 \\ 0 & \Delta_{k+1;2i}\Omega_{k+1;2i} \end{pmatrix},$$

where

$$\begin{aligned} R_{k;2i-1} &= (\Omega_{k;2i-1} \ 0), \\ R_{k;2i} &= (0 \ \Omega_{k;2i}). \end{aligned}$$

Hence it is sufficient to specify the $\Omega_{k;i}$'s. To do that we first specify the two sets of auxiliary matrix-valued functions

$$\begin{aligned} \sigma_u(0) &= I, \\ \sigma_u(i+1) &= \begin{pmatrix} \sigma_u(i)C_L \\ C_R \end{pmatrix}, \end{aligned}$$

and

$$\begin{aligned} \sigma_l(0) &= I, \\ \sigma_l(i+1) &= \begin{pmatrix} C_L \\ \sigma_l(i)C_R \end{pmatrix}. \end{aligned}$$

Then

$$\begin{aligned} \Omega_{k;2i} &= \begin{pmatrix} \sigma_u(K-k-1) & 0 \\ 0 & I \end{pmatrix}, \\ \Omega_{k;2i-1} &= \begin{pmatrix} I & 0 \\ 0 & \sigma_l(K-k-1) \end{pmatrix}, \end{aligned}$$

with the understanding that $\sigma_u(-1)$ and $\sigma_l(-1)$ denote the empty matrices.

With this we have given a complete specification for computing the HSS representation (assuming a uniform tree) of a smooth matrix with a one-dimensional kernel function.

However given the sparse structure of $U_{k;i}$ and $R_{k;i}$, the fast solvers and multipliers presented in this paper can, and should, be modified to exploit the extra structure. This is important as the Chebyshev expansions are not optimal low-rank expansions.