Hierarchically semi-separable (HSS) matrix, space padded around each matrix block, which are uniquely identified by dimensions and color.
HSS matrix, three levels
Hierarchically semi-separable matrices

Structure and definition

HSS matrix formal definition

Consider matrix $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$

- the $l$-level HSS factorization is
  \[
  \mathcal{H}_l(A) = \begin{cases}
  \{ U, V, T_{12}, T_{21}, A_{11}, A_{22} \} & : l = 1 \\
  \{ U, V, T_{12}, T_{21}, \mathcal{H}_{l-1}(A_{11}), \mathcal{H}_{l-1}(A_{22}) \} & : l > 1
  \end{cases}
  \]

- the low-rank representation of the diagonal blocks is given by
  $A_{21} = \tilde{U}_2 T_{21} \tilde{V}_1^T$, $A_{12} = \tilde{U}_1 T_{12} \tilde{V}_2^T$ where for $a \in \{1, 2\}$,

  $\tilde{U}_a = U_a(\mathcal{H}_l(A)) = \begin{cases}
  U_a & : l = 1 \\
  \begin{bmatrix} U_1(\mathcal{H}_{l-1}(A_{aa})) & 0 \\ 0 & U_2(\mathcal{H}_{l-1}(A_{aa})) \end{bmatrix} U_a & : l > 1
  \end{cases}$

  $\tilde{V}_a = V_a(\mathcal{H}_l(A)) = \begin{cases}
  V_a & : l = 1 \\
  \begin{bmatrix} V_1(\mathcal{H}_{l-1}(A_{aa})) & 0 \\ 0 & V_2(\mathcal{H}_{l-1}(A_{aa})) \end{bmatrix} V_a & : l > 1
  \end{cases}$
HSS matrix–vector multiplication

We now consider computing $y = Ax$

- with $H_1(A)$ we would just compute $y_1 = A_{11}x_1 + U_1(T_{12}(V_2^T x_2))$ and $y_2 = A_{22}x_2 + U_2(T_{21}(V_1^T x_1))$
- for general $H_l(A)$ we will perform an up-sweep and a down-sweep
  - up-sweep computes $w = \begin{bmatrix} \bar{V}_1^T x_1 \\ \bar{V}_2^T x_2 \end{bmatrix}$ at every tree node
  - down-sweep computes a tree sum of $\begin{bmatrix} \bar{U}_1 T_{12} w_2 \\ \bar{U}_2 T_{21} w_1 \end{bmatrix}$
- the up-sweep is performed by using the nested structure of $\bar{V}$

$$w = \mathcal{W}(H_l(A), x) = \begin{cases} 
\begin{bmatrix} V_1^T & 0 \\ 0 & V_2^T \end{bmatrix} x & : l = 1 \\
\begin{bmatrix} V_1^T & 0 \\ 0 & V_2^T \end{bmatrix} \begin{bmatrix} \mathcal{W}(H_{l-1}(A_{11}), x_1) \\ \mathcal{W}(H_{l-1}(A_{22}), x_2) \end{bmatrix} & : l > 1 
\end{cases}$$
HSS matrix–vector multiplication, down-sweep

We now employ each \( w = \mathcal{W}(\mathcal{H}_l(A), x) \) from the root to the leaves to get

\[
y = Ax = \begin{bmatrix} U_1 T_{12} w_2 \\ U_2 T_{21} w_1 \end{bmatrix} + \begin{bmatrix} A_{11} x_1 \\ A_{22} x_2 \end{bmatrix} = \begin{bmatrix} \tilde{U}_1 & 0 \\ 0 & \tilde{U}_2 \end{bmatrix} \begin{bmatrix} 0 & T_{12} \\ T_{21} & 0 \end{bmatrix} w + \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix} x
\]

- using the nested structure of \( \tilde{U}_a \) and \( v = \begin{bmatrix} U_1 & 0 \\ 0 & U_2 \end{bmatrix} \begin{bmatrix} 0 & T_{12} \\ T_{21} & 0 \end{bmatrix} w, \)

\[
y_a = \begin{bmatrix} \mathcal{U}_1(\mathcal{H}_{l-1}(A_{aa})) & 0 \\ 0 & \mathcal{U}_2(\mathcal{H}_{l-1}(A_{aa})) \end{bmatrix} v_a + A_{aa} x_a \quad \text{for } a \in \{1, 2\}
\]

- which gives the down-sweep recurrence

\[
y = Ax + z = \mathcal{Y}(\mathcal{H}_l(A), x, z) = \begin{cases} \\
\begin{bmatrix} U_1 q_1 \\ U_2 q_2 \end{bmatrix} + \begin{bmatrix} A_{11} x_1 \\ A_{22} x_2 \end{bmatrix} & : l = 1 \\
\mathcal{Y}(\mathcal{H}_{l-1}(A_{11}), x_1, U_1 q_1) + \mathcal{Y}(\mathcal{H}_{l-1}(A_{22}), x_2, U_2 q_2) & : l > 1 \\
\end{cases}
\]

where \( q = \begin{bmatrix} 0 & T_{12} \\ T_{21} & 0 \end{bmatrix} \mathcal{W}(\mathcal{H}_l(A), x) + z \)
Prefix sum as HSS matrix–vector multiplication

We can express the \( n \)-element prefix sum \( y(i) = \sum_{j=1}^{i-1} x(j) \) as

\[
\mathbf{y} = \mathbf{L} \mathbf{x} \quad \text{where} \quad \mathbf{L} = \begin{bmatrix}
\mathbf{L}_{11} & 0 \\
\mathbf{L}_{21} & \mathbf{L}_{22}
\end{bmatrix} = \begin{bmatrix}
0 & 0 & \cdots & 0 \\
1 & 0 & \cdots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
1 & \cdots & 1 & 0
\end{bmatrix}
\]

- \( \mathbf{L} \) is an \( \mathcal{H} \)-matrix since \( \mathbf{L}_{21} = \mathbf{1}_n \mathbf{1}_n^T = [1 \ \cdots \ 1]^T [1 \ \cdots \ 1] \)
- \( \mathbf{L} \) also has rank-1 HSS structure, in particular

\[
\mathcal{H}_l(\mathbf{L}) = \begin{cases} \{ \mathbf{1}_2, \mathbf{1}_2, [0], [1], [0], [0] \} & : l = 1 \\
\{ \mathbf{1}_4, \mathbf{1}_4, [0], [1], \mathcal{H}_{l-1}(\mathbf{L}_{11}), \mathcal{H}_{l-1}(\mathbf{L}_{22}) \} & : l > 1
\end{cases}
\]

so each \( \mathbf{U}, \mathbf{V}, \tilde{\mathbf{U}}, \tilde{\mathbf{V}} \) is a vector of 1s, \( \mathbf{T}_{12} = [0] \) and \( \mathbf{T}_{21} = [1] \).
Prefix sum HSS up-sweep

We can use the HSS structure of $L$ to compute the prefix sum of $x$

- recall that the up-sweep recurrence has the general form

$$w = \mathcal{W}(\mathcal{H}_l(A), x) = \left\{ \begin{array}{ll} \begin{bmatrix} V_1^T & 0 \\ 0 & V_2^T \end{bmatrix} x & : l = 1 \\ [V_1^T & 0] [\mathcal{W}(\mathcal{H}_{l-1}(A_{11}), x_1)] & : l > 1 \end{array} \right.$$

- for the prefix sum this becomes

$$w = \mathcal{W}(\mathcal{H}_l(L), x) = \left\{ \begin{array}{ll} x & : l = 1 \\ [1 & 1 & 0 & 0] [\mathcal{W}(\mathcal{H}_{l-1}(L_{11}), x_1)] & : l > 1 \end{array} \right.$$

- so the up-sweep computes $w = \begin{bmatrix} S(x_1) \\ S(x_2) \end{bmatrix}$ where $S(a) = \sum_i a(i)$
Prefix sum HSS down-sweep

The down-sweep has the general structure

\[ y = \mathcal{V}(\mathcal{H}_l(A), x, z) = \begin{cases} \begin{bmatrix} U_1 & 0 \\ 0 & U_2 \end{bmatrix} q + \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix} x & : l = 1 \\ \mathcal{V}(\mathcal{H}_{l-1}(A_{11}), x_1, U_1 q_1) & : l > 1 \\ \mathcal{V}(\mathcal{H}_{l-1}(A_{22}), x_2, U_2 q_2) \end{cases} \]

where \( q = \begin{bmatrix} 0 & T_{12} \\ T_{21} & 0 \end{bmatrix} \mathcal{V}(\mathcal{H}_l(A), x) + z \), for the prefix sum:

\[ \begin{bmatrix} 0 & T_{12} \\ T_{21} & 0 \end{bmatrix} \mathcal{V}(\mathcal{H}_l(L), x) = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} S(x_1) \\ S(x_2) \end{bmatrix} = \begin{bmatrix} 0 \\ S(x_1) \end{bmatrix} = q - z \]

\[ y = \mathcal{V}(\mathcal{H}_l(L), x, z) = \begin{cases} \begin{bmatrix} z(1) \\ x(1) + z(2) \end{bmatrix} & : l = 1 \\ \mathcal{V}(\mathcal{H}_{l-1}(L_{11}), x_1, 1_2 z(1)) & : l > 1 \\ \mathcal{V}(\mathcal{H}_{l-1}(L_{22}), x_2, 1_2 (S(x_1) + z(2))) \end{cases} \]

- initially the prefix \( z = 0 \) and it will always be the case that \( z(1) = z(2) \)
Short pause
Cost of HSS matrix–vector multiplication

The down-sweep and the up-sweep perform small dense matrix–vector multiplications at each recursive step

- Let's assume $k$ is the dimension of the leaf blocks and the rank at each level (number of columns in each $U_a, V_a$)
- The computation cost for both the down-sweep and up-sweep is
  \[
  T(n, k) = 2T(n/2, k) + O(k^2 \cdot \gamma), \quad T(k, k) = O(k^2 \cdot \gamma)
  \]
  \[
  T(n, k) = O(nk \cdot \gamma)
  \]

- Q: What parallelization approach would be sensible for small $k$?
- If we assign each tree node to a single processor for the first $\log_2(P)$ levels, and execute a different leaf subtree with a different processor
  \[
  T(n, k, P) = 2T(n, k, P/2) + O(k^2 \cdot \gamma + k \cdot \beta + \alpha)
  \]
  \[
  = O((nk/P + k^2 \log(P)) \cdot \gamma + k \log(P) \cdot \beta + \log(P) \cdot \alpha)
  \]
**Synchronization-efficient HSS multiplication**

Our first algorithm would require $O(\log(P))$ BSP supersteps
- Q: what would we need to do to reduce this to $O(1)$?
- we can observe that the leaf subtrees can be computed independently

\[ T_{\text{leaf-subtrees}}(n, k, P) = O(nk/P \cdot \gamma + k \cdot \beta + \alpha) \]

thus we can focus on doing the up-sweep and down-sweep on a binary tree with $\log_2(P)$ levels
- executing the root subtree sequentially would yield a cost of

\[ T_{\text{root-subtree}}(n, k, P) = O(Pk^2 \cdot \gamma + Pk \cdot \beta + \alpha) \]

this could be prohibitive on a large number of processors
- instead have $P^r$ ($r < 1$) processors compute subtrees with $P^{1-r}$ leaves, then recurse on the $P^r$ roots

\[ T_{\text{rec-tree}}(k, P) = T_{\text{rec-tree}}(k, P^r) + O(P^{1-r}k^2 \cdot \gamma + P^{1-r}k \cdot \beta + \alpha) \]

the algorithm has BSP complexity

\[ T_{\text{rec-tree}}(k, P) = O(P^{1-r}k^2 \cdot \gamma + P^{1-r}k \cdot \beta + \log_{1/r}(\log(P)) \cdot \alpha) \]
Synchronization-efficient HSS multiplication

We can do better by exploiting what the HSS tree nodes are doing

- again lets focus on the top tree with $P$ leaves (leaf subtrees)
- lets try to assign each processor a unique path from a leaf to the root
- given $w = \mathcal{W}(H_i(A), x)$ at every node its clear each processor can compute a down-sweep path in the subtree independently
- for the up-sweep, we can realize that the tree applies a linear transformation, so we can sum the results computed in each path
- for each tree node, there is a contribution from every processor assigned a leaf of the subtree of the node
- therefore, there are $P - 1$ sums of a total of $O(P \log(P))$ contributions, for a total of $O(kP \log(P))$ elements
- we can do these with $\min(P, k \log_2(P))$ processors, each obtaining $\max(P, k \log_2(P))$ contributions, so

$$T_{\text{root-paths}}(k, P) = O(k^2 \log(P) \cdot \gamma + (k \log(P) + P) \cdot \beta + \alpha)$$

- we have not improved the asymptotic number of messages, but only the number of synchronizations, and can leverage efficient reductions
HSS multiplication by multiple vectors

Consider multiplication $\mathbf{C} = \mathbf{A} \mathbf{B}$ where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is HSS and $\mathbf{B} \in \mathbb{R}^{n \times b}$

- lets consider the case that $P \leq b \ll n$
- if we assign each processor all of $\mathbf{A}$, each can compute a column of $\mathbf{C}$ simultaneously
- this requires a prohibitive amount of memory usage
- Q: could you propose a good BSP algorithm for this problem?
- A: use transpose like in the $b$ scans problem
  - perform leaf-level multiplications, processing $n/P$ rows of $\mathbf{B}$ with each processor (call intermediate $\bar{\mathbf{C}}$)
  - transpose $\bar{\mathbf{C}}$ and apply $\log_2(P)$ root levels of HSS tree to columns of $\bar{\mathbf{C}}$ independently
- this algorithm requires replication only of the root $O(\log(P))$ levels of the HSS tree, $O(Pb)$ data
- for large $k$ or larger $P$ different algorithms may be desirable