Adaptive Fast Boundary Element Methods in Industrial Applications

Söllerhaus, October 2nd, 2004.

Efficient Update of Hierarchical Matrices

joint work with L. Grasedyck, W. Hackbusch and S. Le Borne

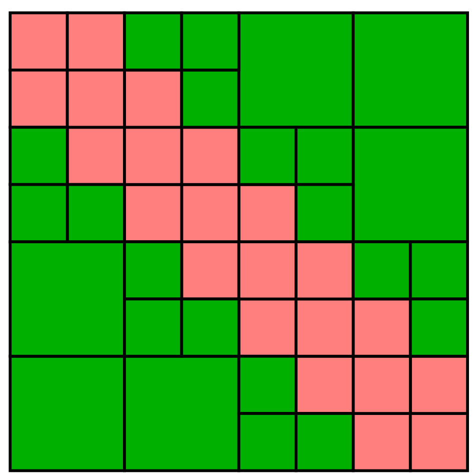
Jelena Djokić

Max Planck Institute for
Mathematics in the Sciences
Leipzig

- Concept of hierarchical (or $\mathcal{H}$-) matrices
- Motivation for update of $\mathcal{H}$-matrices
- Update algorithm
- Numerical results

Properties of $\mathcal{H}$-Matrices

- $\mathcal{H}$-matrix is an approximation of full matrix that e.g. arises from discretisation of integral operator.
- $\mathcal{H}$-matrices have a block structure - each block is either rank k (`Rk`) or dense (`full`) matrix.
- With $\mathcal{H}$-matrices is possible to perform matrix operations (MVM,MM,Inv) with almost linear complexity.



Some construction remarks

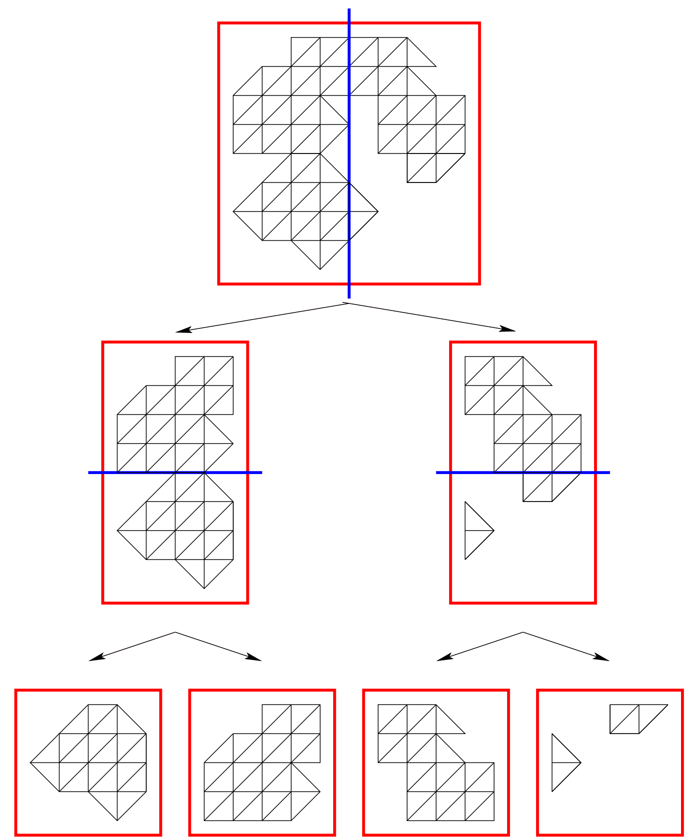
- $\mathcal{H}$-matrices are based on the given block cluster tree $T_{\mathcal{I}\times\mathcal{I}}$.
- The block cluster tree $T_{\mathcal{I}\times\mathcal{I}}$ is constructed using the cluster tree $T_{\mathcal{I}}$ (and an admissibility condition).
- The cluster tree $T_{\mathcal{I}}$ is determined and based on a partitioned grid τ and an index set $\mathcal{I}$.

In general: $\#$Basis functions $= |\mathcal{I}|$.
Example: Piece-wise constant ansatz leads to the $|\tau| = |\mathcal{I}|$.

Geometrically regular clustering

Compute a box, that contains the whole domain to whom grid τ belongs.

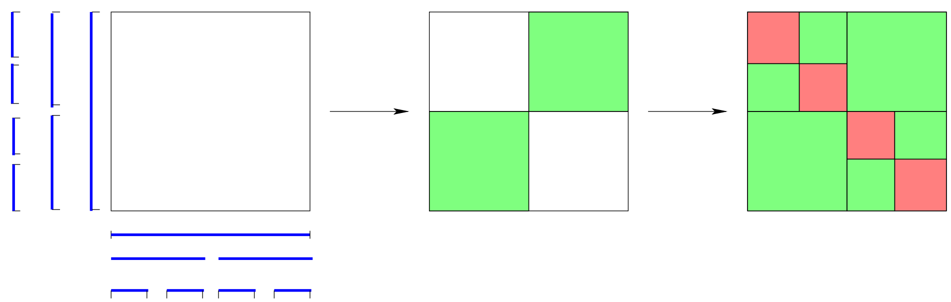
1. Determine the maximal extent.
2. Split box in that direction.
3. Repeat the process as long as it is necessary.



Remark: This clustering routine is independent of the grid.

Given: cluster tree $T_{\mathcal{I}}$ with root $\mathcal{I} = \{1, \ldots, n\}$

Seeking: block cluster tree $T_{\mathcal{I}\times\mathcal{I}}$

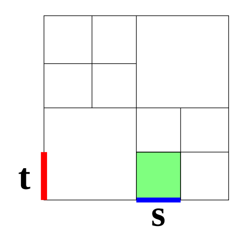


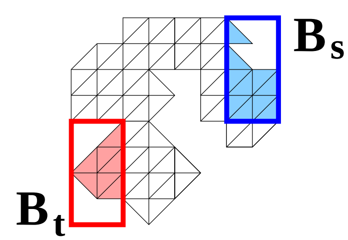
Start: $\mathcal{I} \times \mathcal{I}$. Iterate: subdivide inadmissible blocks:

$$\mathrm{sons}(t \times s) := \mathrm{sons}(t) \times \mathrm{sons}(s).$$

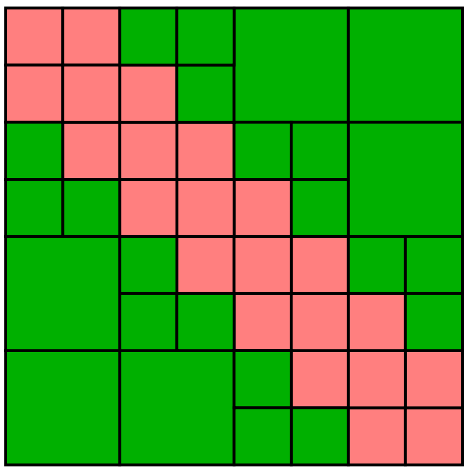
Admissibility condition:

$\min(\mathrm{diam}(B_t), \mathrm{diam}(B_s)) \leq \eta \, \mathrm{dist}(B_t, B_s)$

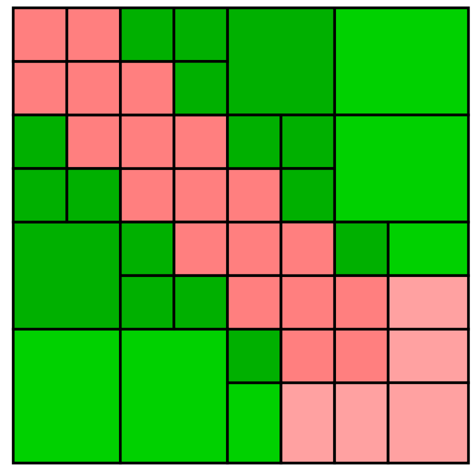




The grid τ **locally refined**
Cluster tree $T_{\mathcal{I}}$, based on $\mathcal{I}$.
Matrix $G \in \mathcal{H}(T_{\mathcal{I}\times\mathcal{I}}, k)$.



Grid τ' is obtained.
Cluster tree $T_{\mathcal{I}'}$, based on $\mathcal{I}'$.
Matrix $G' \in \mathcal{H}(T_{\mathcal{I}'\times\mathcal{I}'}, k)$.



Question: Can G be **used** in the construction of the G', an $\mathcal{H}$-matrix that corresponds to the refined grid τ'?

Idea: **Recycle** the $\mathcal{H}$-matrix instead of constructing new one = **Update** of $\mathcal{H}$-matrix.

$\mathcal{H}$-Matrix Update Algorithm

Let $G \in \mathcal{H}(T_{\mathcal{I}\times\mathcal{I}}, k)$ be an $\mathcal{H}$-matrix. Update of G can be done in three steps:

- Update of cluster tree $T_{\mathcal{I}}$ (removing old and adding new indices in tree).
- Update of block cluster tree $T_{\mathcal{I}\times\mathcal{I}}$ using already changed cluster tree $T_{\mathcal{I}}$.
- Update of `Rk` and `full` matrix blocks from G.

- τ' is the grid obtained after local refinement of the grid τ.
- $\mathcal{I}'$ is an index set that corresponds to the grid τ'.
- For $\mathcal{I}'$ there holds:

$$\mathcal{I}' = (\mathcal{I} \setminus \mathcal{I}_{out}) \quad \dot{\cup} \quad \mathcal{I}_{in}$$

- $\mathcal{I}_{out} \subset \mathcal{I}$ is the set of indices that have to be removed from $\mathcal{I}$.
- $(\mathcal{I} \setminus \mathcal{I}_{out})$ is the set of indices that correspond to the unchanged basis functions.
- $\mathcal{I}_{in}$ is the set of indices, that correspond to the new basis functions.

Update of the cluster tree

The cluster tree $T_{\mathcal{I}'}$ that corresponds to the refined grid τ' can be represented as

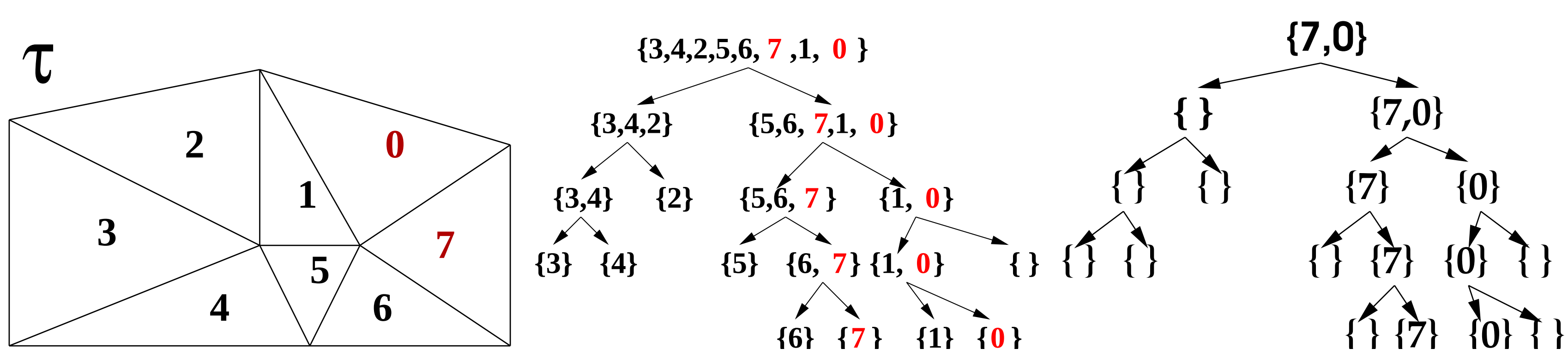
$$T_{\mathcal{I}'} = (T_{\mathcal{I}} \setminus T_{\mathcal{I}_{out}}) \quad \cup \quad T_{\mathcal{I}_{in}}$$

where $T_{\mathcal{I}_{out}}, T_{\mathcal{I}_{in}}$ are the cluster trees that correspond to the index sets $\mathcal{I}_{out}$ and $\mathcal{I}_{in}$.

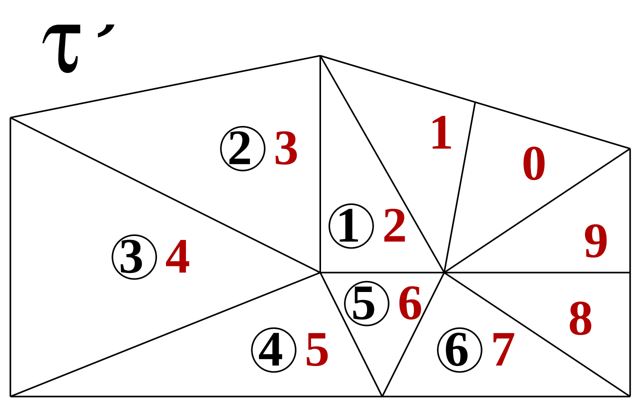
Problem: How to construct those cluster trees?

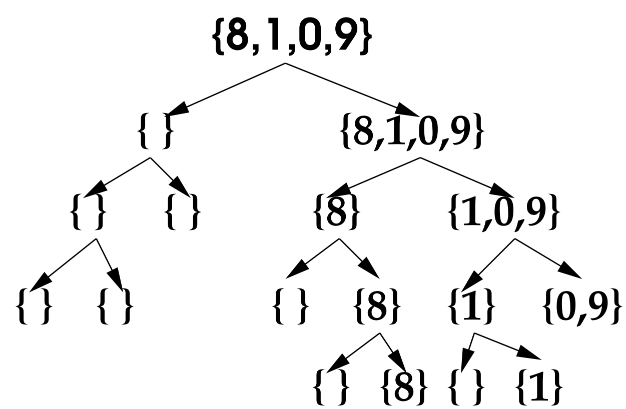
The algorithm that describes the construction (not clustering) of the tree $T_{\mathcal{I}'}$ has the following steps:

1a.**Construct** the cluster tree $T_{\mathcal{I}_{out}}$ as **index reproduction** of the given cluster tree $T_{\mathcal{I}}$.

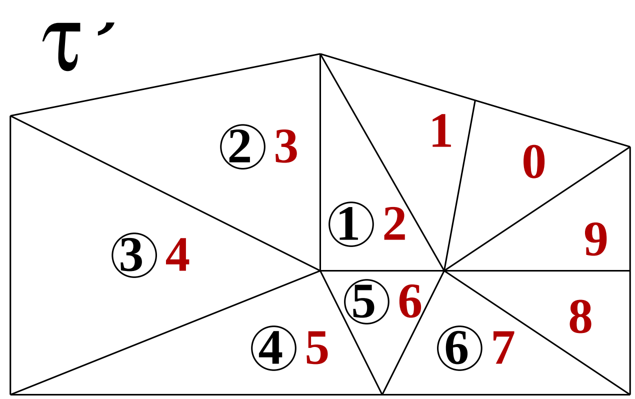


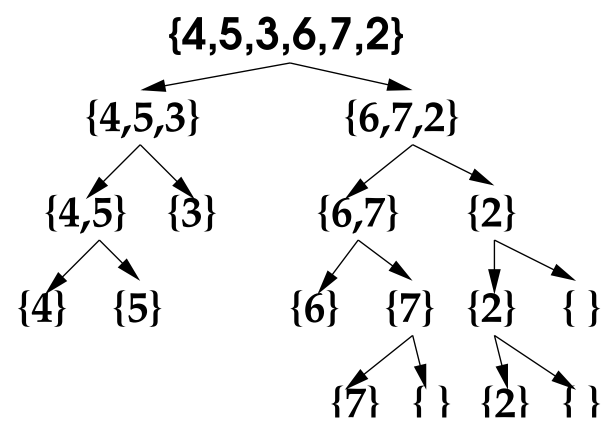
1b.**Construct** the cluster tree $T_{\mathcal{I}_{in}}$ as **bounding box reproduction** of the given cluster tree $T_{\mathcal{I}}$.





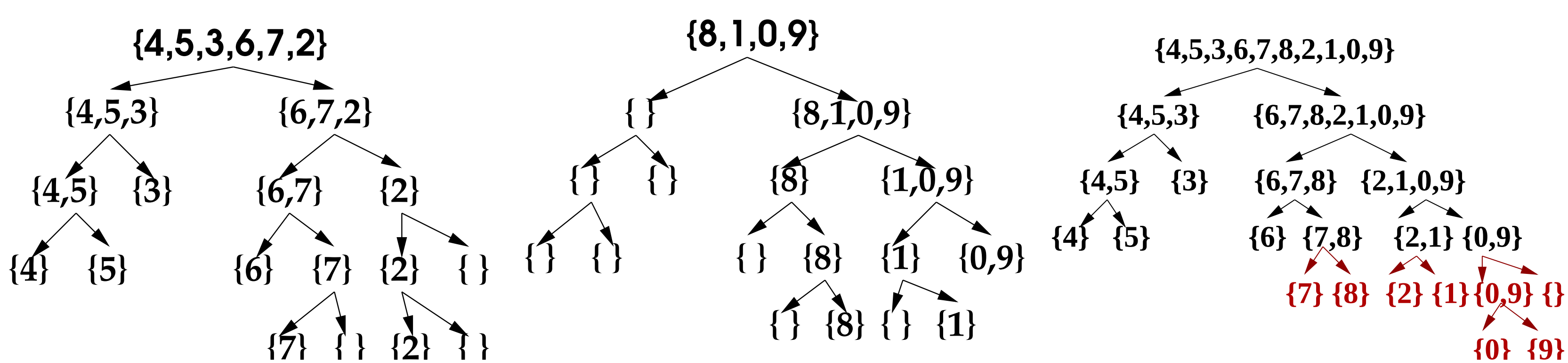
2. **Restrict** the cluster tree $T_{\mathcal{I}}$. The result is the cluster tree $(T_{\mathcal{I}} \setminus T_{\mathcal{I}_{out}})$.





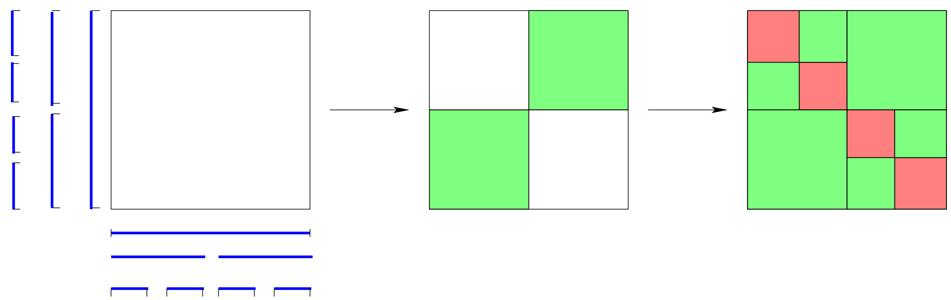
3. Do a **fusion** (union) of the trees $T_{\mathcal{I}} \setminus T_{\mathcal{I}_{out}}$ and $T_{\mathcal{I}_{in}}$. We obtain the preliminary tree $T'_{\mathcal{I}'}$, that might have some leaves whose size is greater than given n_{min}.

3a. **Subdivide** only those leaves whose size are larger than n_{min}. The final result is the tree $T_{\mathcal{I}'}$.



Given: cluster tree $T_{\mathcal{I}}$ with root $\mathcal{I} = \{1, \ldots, n\}$

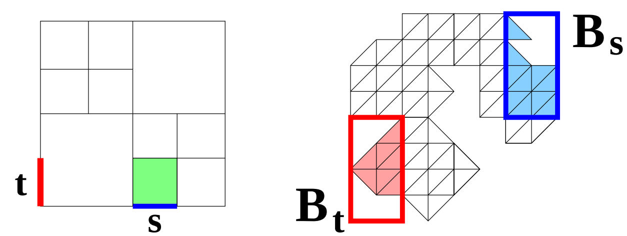
Seeking: block cluster tree $T_{\mathcal{I}\times\mathcal{I}}$



Start: $\mathcal{I} \times \mathcal{I}$. Iterate: subdivide inadmissible blocks:

$$\mathrm{sons}(t \times s) := \mathrm{sons}(t) \times \mathrm{sons}(s).$$

Admissibility condition:
$\min(\mathrm{diam}(B_t), \mathrm{diam}(B_s)) \le \eta\, \mathrm{dist}(B_t, B_s)$

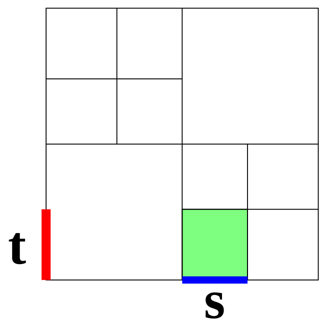


Update: Since the cluster trees $T_{\mathcal{I}}$ and $T_{\mathcal{I}'}$ have the same bounding boxes, there is one-one correspondence between block cluster trees $T_{\mathcal{I}\times\mathcal{I}}$ and $T_{\mathcal{I}'\times\mathcal{I}'}$.

Update of the $\mathcal{H}$-matrix $G \in \mathcal{H}(T_{\mathcal{I}\times\mathcal{I}}, k)$,$G' \in \mathcal{H}(T_{\mathcal{I}'\times\mathcal{I}'}, k)$

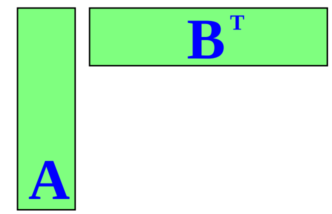
The algorithm for constructing the matrix G' (based on the block cluster tree $T_{\mathcal{I}'\times\mathcal{I}'}$) using the $\mathcal{H}$-matrix G is:

- if the leaf of the block cluster tree remained unchanged then **copy** the corresponding block matrix (`Rk` or `full` block).
- if the leaf of the block cluster tree contains all new indices than construct **new** matrix block.
- if the leaf of the block cluster tree contains some new indices **update** the corresponding block.



$$G_{ij} = \int_\Gamma \int_\Gamma \phi_i(x) g(x,y) \phi_j(y) \, \mathrm{d}\Gamma_x \mathrm{d}\Gamma_y$$

$$G|_{t\times s} \approx AB^T, A \in \mathbb{R}^{\#t\times k}, B \in \mathbb{R}^{\#s\times k}.$$



Interpolation:

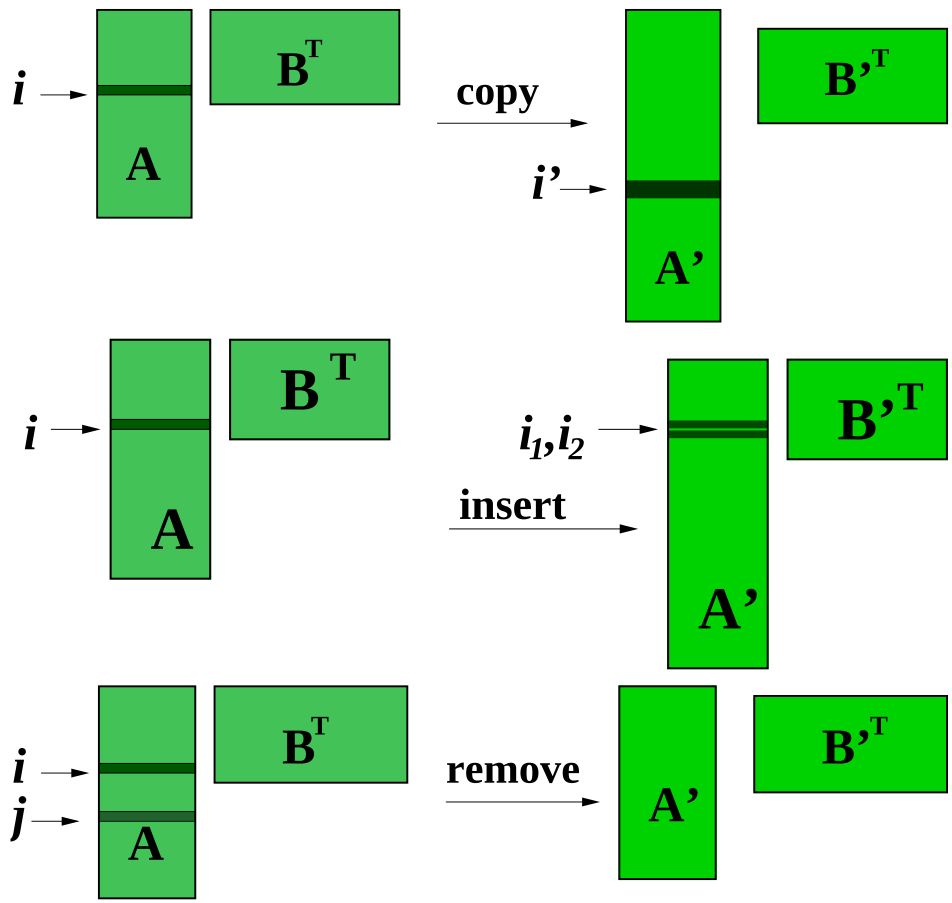
$$g(x,y) \approx \sum_{\nu=1}^{m^3} L_\nu(x) g(x_\nu, y)$$

$$A_{i\nu} = \int_\Gamma \phi_i(x) L_\nu(x) \, \mathrm{d}\Gamma_x, \quad B_{j\nu} = \int_\Gamma \phi_j(y) g(x_\nu, y) \, \mathrm{d}\Gamma_y$$

$$G'|_{t'\times s'} = A'B'^T \in \mathbb{R}^{\#t'\times\#s'}$$

$$A'_{i\nu} = \begin{cases} A_{i\nu} & i \in t \\ \int_\Gamma \phi_i(x) L_\nu(x) \, \mathrm{d}\Gamma_x & i \in t' \setminus t \end{cases}$$

$$B'_{j\nu} = \begin{cases} B_{j\nu} & j \in s \\ \int_\Gamma \phi_j(y) g(x_\nu, y) \, \mathrm{d}\Gamma_y & j \in s' \setminus s \end{cases}$$

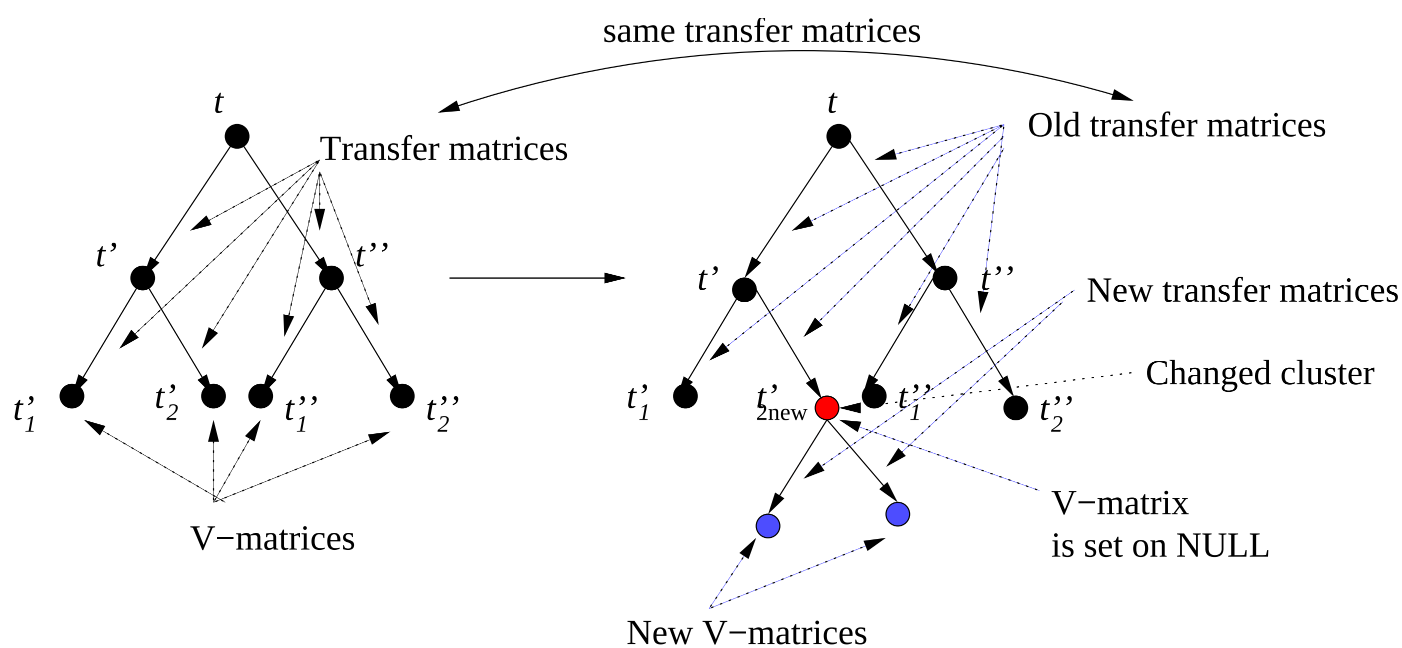

i
A
B'T
copy
i'
A'
B''T
i
A
BT
i1,i2
insert
A'
B'T
i
j
A
BT
remove
A'
B'T

$$
\begin{aligned}
\tilde{g}^{t,s}(x,y) &:= (\mathcal{J}_m^t \otimes \mathcal{J}_m^s)[g](x,y) = \sum_{\nu\in K}\sum_{\mu\in K} g(x_\nu^t, x_\mu^s)\mathcal{L}_\nu^t(x)\mathcal{L}_\mu^s(y). \\
\tilde{G}_{ij} &:= \int_\Omega \varphi_i(x) \int_\Omega \tilde{g}^{t,s}(x,y)\varphi_j(y)\,\mathrm{d}y\,\mathrm{d}x \\
&= \sum_{\nu\in K}\sum_{\mu\in K} g(x_\nu^t, x_\mu^s) \underbrace{\left(\int_\Omega \varphi_i(x)\mathcal{L}_\nu^t(x)\,\mathrm{d}x\right)}_{=V_{i\nu}^t} \underbrace{\left(\int_\Omega \varphi_j(y)\mathcal{L}_\mu^s(y)\,\mathrm{d}y\right)}_{=V_{j\mu}^s} \\
&= V^t S^{t,s} V^{s\top} \\
S_{\nu\mu}^{t,s} &:= g(x_\nu^t, x_\mu^s). \\
T_{\nu',\nu}^{t',t} &:= \mathcal{L}_\nu^t(x_{\nu'}^{t'}), \quad V^t = \begin{pmatrix} V^{t_1}\cdot T^{t_1,t} \\ V^{t_2}\cdot T^{t_2,t} \end{pmatrix}.
\end{aligned}
$$

- Matrix S depends **only** on bounding box.
- Matrices V^t depend only on the cluster but they will be computed only for the leaves.
- Transfer matrices $T^{t,t'}$ depend also only on bounding boxes.

Update of $\mathcal{H}^2$-Matrices

- $T^{t,t'}$ and S matrices need not to be updated, since bounding boxes do not change.
- V matrices are updated in the same way as Rk matrices are updated.



$$\mathcal{G}[u](x) = f(x), \quad f := \mathcal{V}\partial_n u, \quad x \in \Gamma := \partial\Omega, \Omega := [-1,1]^3$$

- $\mathcal{G}$ is the double layer potential operator
  $\mathcal{G}[u](x) = \frac{1}{2}u(x) + \frac{1}{4\pi}\int_\Gamma \frac{\langle n(y), x-y\rangle u(y)}{\|x-y\|^3} d\Gamma_y$
- $\mathcal{V}$ is the single layer potential operator
  $\mathcal{V}[u](x) := \frac{1}{4\pi}\int_\Gamma \frac{\partial_n u(y)}{\|x-y\|} d\Gamma_y$.
- $u(x) = \frac{1}{\|x-y_0\|}$, $y_0 = (1.0, 1.0, 1.001)$.
- Machine: `SUN ULTRASPARC III` with 900 MHz CPU clock rate and 150 MHz memory clock rate.

Time (in seconds) for the update of the (double-layer potential operator) $\mathcal{H}$-matrix compared to reassembly starting with n_1 degrees of freedom.

n_1 =12288	n_2 =12422	n_2 =13002	n_2 =15806
new	2.1%	10.9%	44.5%
adaptive (G')	1.02	5.76	23.16
reassembly (G'')	29.5	31.67	40.82
savings(costs)	97%(3%)	82%(18%)	44%(56%)
$\frac{\|G''-G'\|_2}{\|G''\|_2}$	6.04×10^{-16}	6×10^{-16}	5.29×10^{-16}

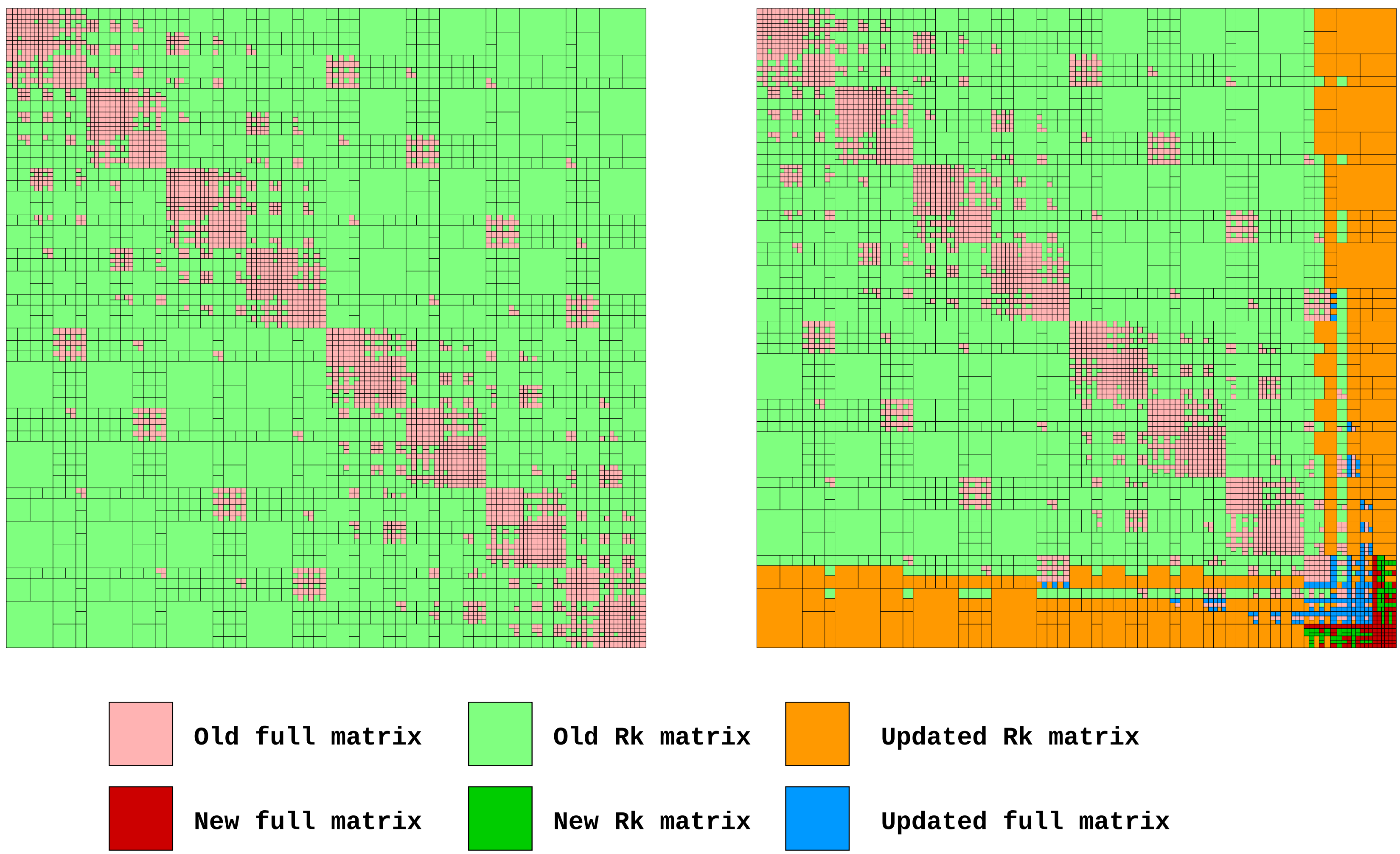
$n_1 = 49152$	n_2 =49682	n_2 =51880	n_2 =62544
new	2.1%	10.5%	42.8%
adaptive (G')	6.05	31.8	121.78
reassembly (G'')	169	209.6	252.7
savings(costs)	97%(3%)	85%(15%)	52%(48%)
$\frac{\|G''-G'\|_2}{\|G''\|_2}$	7.05×10^{-16}	7.05×10^{-16}	6.44×10^{-16}

$n_1 = 196608$	n_2 =198686	n_2 =207190	n_2 = 248290
new	2.1%	10.2%	42%
adaptive (G')	42.91	147.65	543.55
reassembly (G'')	791	875.8	1068.48
savings(costs)	95%(5%)	83%(17%)	51%(49%)
$\frac{\|G''-G'\|_2}{\|G''\|_2}$	7.90×10^{-16}	7.80×10^{-16}	4.3×10^{-16}

Time (in seconds) for the update of the $\mathcal{H}^2$-matrix compared to reassembly starting with $n_1 = 49152(196608)$ degrees of freedom.

n_1 =49152	n_2 =49676	n_2 =51734	n_2 =62462
new	2.1%	10%	42.6%
adaptive	2.3	9.5	50.1
reassembly	67.6	71	86.8
savings(costs)	97%(3%)	87%(13%)	42%(58%)

n_1 = 196608	n_2 =198672	n_2 =206754	n_2 =247984
new	2.1%	9.8 %	41.4%
adaptive	14.1	55.3	291.4
reassembly	455.6	471.2	547.2
savings (costs)	97%(3%)	88%(12%)	47%(53%)


Old full matrix
Old Rk matrix
Updated Rk matrix
New full matrix
New Rk matrix
Updated full matrix

Outlook

- $\mathcal{H}$- and $\mathcal{H}^2$-matrices can be efficiently updated.
- If `Rk` matrices are computed using ACA (Adaptive Cross Approximation), updated of $\mathcal{H}$-matrices can be as well efficiently performed.

Current work

- Implementation of local error estimators.
- Update of $\mathcal{H}$-matrices if `Rk` matrices are computed using HCA (Hybrid Cross Approximation).

`www.hmatrix.org`

Adaptive Cross Approximation (ACA)

Aim Construct an approximation of the form $\sum_{\nu=1}^{k} a_\nu b_\nu^T$ to a matrix $M \in \mathbb{R}^{n\times m}$ up to a relative error $\|M - \sum_{\nu=1}^{k} a_\nu b_\nu^T\|_2 \approx \epsilon \|M\|_2$. **Algorithm**

Input: A function that returns the matrix entry M_{ij} for an index pair (i, j).

Step $\nu = 1 \ldots k$:

1. Determine (and **save**) a pivot index (i^*, j^*).
2. Compute the entries of the two vectors $a_\nu \in \mathbb{R}^n, b_\nu \in \mathbb{R}^m$ by

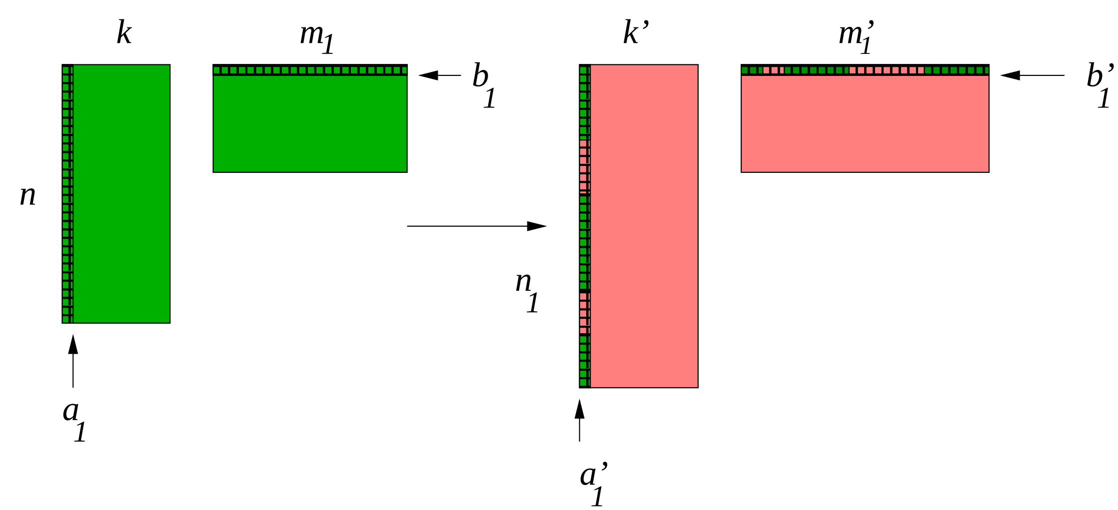
$$(a_\nu)_i := M_{ij^*} - \sum_{\nu=1}^{\mu-1} (a_\nu)_i (b_\nu)_{j^*}, (b_\nu)_j := \frac{1}{\delta}\Big(M_{i^*j} - \sum_{\nu=1}^{\mu-1} (a_\nu)_{i^*} (b_\nu)_j\Big).$$

Stop if $\|a_\nu\|_2 \|b_\nu\|_2 \le \epsilon \|a_1\|_2 \|b_1\|_2$.

Output: The factorisation $AB^T \approx M$.

Since the pivot elements are saved in the update of Rk matrix (computed by ACA) we distinguish three cases:

- All pivot pairs can be reused.
- First $t, t < k$ pivot pairs can be used, rest has to be computed as in the original algorithm.
- There is no pivot pair that can be used again, there is no update possible, i.e. Rk matrix is completely new.



Time (in seconds) for the update of the SLP $\mathcal{H}$-matrix compared to reassembly starting with $n_1 = 12288$ ($n_1 = 49152$) degrees of freedom

n_1 =12288 new	n_2 =12422 2.2%	n_2 =13014 11.2%	n_2 = 15790 44.4%
adaptive G_{ad}	1.5	8.6	32.4
reassembly G_{or}	33.7	35.5	44.1
savings (costs)	94%(6%)	76%(24%)	26%(74%)
$\|G_{ad} - G_{exact}\|_2$	9.5×10^{-7}	8.33×10^{-7}	1.54×10^{-6}
$\|G_{or} - G_{exact}\|_2$	9.22×10^{-7}	8.26×10^{-5}	6.70×10^{-7}

Numerical test for ACA

n_1 =49152	n_2 = 49682	n_2 =51870	n_2 =62494
new	2.1%	10.5%	42.7%
adaptive G_{ad}	10	43.2	146
reassembly G_{or}	167.2	177.1	213.7
savings(costs)	94%(6%)	76%(24%)	32%(68%)
$\|G_{ad} - G_{exact}\|_2$	3.37×10^{-7}	4.85×10^{-7}	4.21×10^{-7}
$\|G_{or} - G_{exact}\|_2$	3.36×10^{-7}	1.75×10^{-7}	1.93×10^{-7}