Introduction to $\mathcal{H}^2$-matrices

Steffen Börm

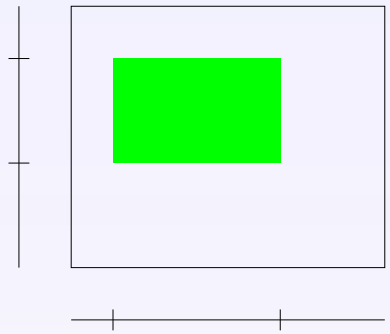
Christian-Albrechts-Universität, Kiel

Winterschool on Hierarchical Matrices

Uniform matrices

Interpolation in one coordinate yields

$$g(x,y) \approx \sum_{\nu} \mathcal{L}_\nu(x) g(\xi_\nu, y).$$

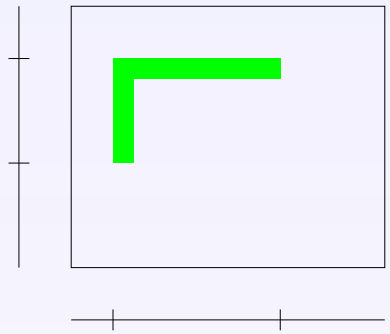


Linear algebra: $G|_{\hat{t}\times\hat{s}} \approx A_b B_b^*$ for $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$

Uniform matrices

Interpolation in one coordinate yields

$$g(x,y) \approx \sum_{\nu} \mathcal{L}_\nu(x) g(\xi_\nu, y).$$

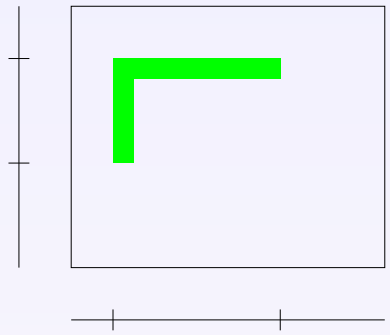


Linear algebra: $G|_{\hat{t}\times\hat{s}} \approx A_b B_b^*$ for $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$

Uniform matrices

Interpolation in one coordinate yields

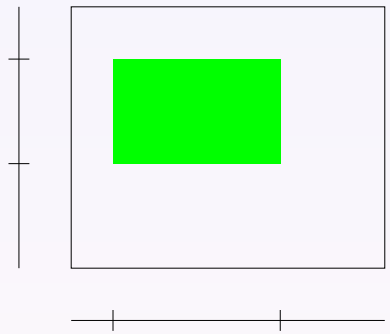
$$g(x,y) \approx \sum_{\nu} \mathcal{L}_\nu(x) g(\xi_\nu, y).$$



Linear algebra: $G|_{\hat{t}\times\hat{s}} \approx A_b B_b^*$ for $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$

Idea: Interpolation to both variables.

$$g(x,y) \approx \sum_{\nu}\sum_{\mu} \mathcal{L}_\nu(x) g(\xi_\nu, \xi_\mu) \mathcal{L}_\mu(y)$$



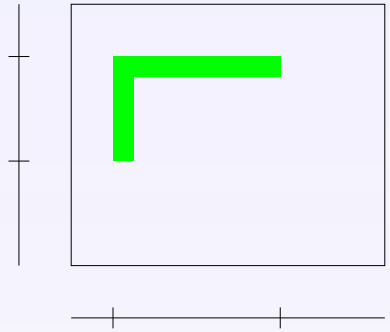
Linear algebra: $G|_{\hat{t}\times\hat{s}} \approx V_t S_b W_s^*$ for $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$

Important: V_t and W_s depend on clusters, but not on the block.
$\rightarrow$ Storage per block reduced to k^2.

Uniform matrices

Interpolation in one coordinate yields

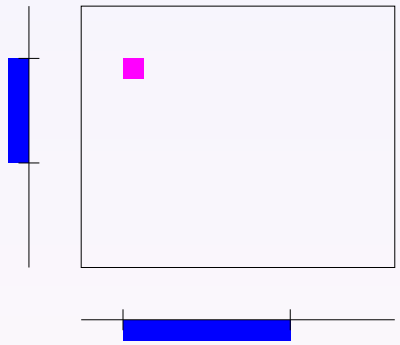
$$g(x,y) \approx \sum_{\nu} \mathcal{L}_\nu(x) g(\xi_\nu, y).$$



Linear algebra: $G|_{\hat{t}\times\hat{s}} \approx A_b B_b^*$ for $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$

Idea: Interpolation to both variables.

$$g(x,y) \approx \sum_{\nu}\sum_{\mu} \mathcal{L}_\nu(x) g(\xi_\nu, \xi_\mu) \mathcal{L}_\mu(y)$$



Linear algebra: $G|_{\hat{t}\times\hat{s}} \approx V_t S_b W_s^*$ for $b = (t,s) \in \mathcal{T}_{\mathcal{I}\times\mathcal{J}}$

Important: V_t and W_s depend on clusters, but not on the block.
$\rightarrow$ Storage per block reduced to k^2.

Cluster basis

Uniform matrix: Approximate admissible blocks $b = (t, s)$ by

$$G|_{\hat{t}\times\hat{s}} \approx V_t S_b W_s^*$$

Coupling matrices (S_b) require only $\mathcal{O}(nk)$ units of storage.

Cluster bases (V_t), (W_s) require $\mathcal{O}(nk \log n)$ units of storage.

Goal: More efficient representation of cluster bases.
$\rightarrow$ Linear complexity $\mathcal{O}(nk)$.

Nested representation

Observation: Interpolation is a projection.

$$\mathcal{L}_{t,\nu} = \sum_{\nu'} \mathcal{L}_{t,\nu}(\xi_{t',\nu'})\mathcal{L}_{t',\nu'}$$

holds for arbitrary clusters t, t'.

Transfer matrix for $t' \in \operatorname{sons}(t)$ represents change of basis

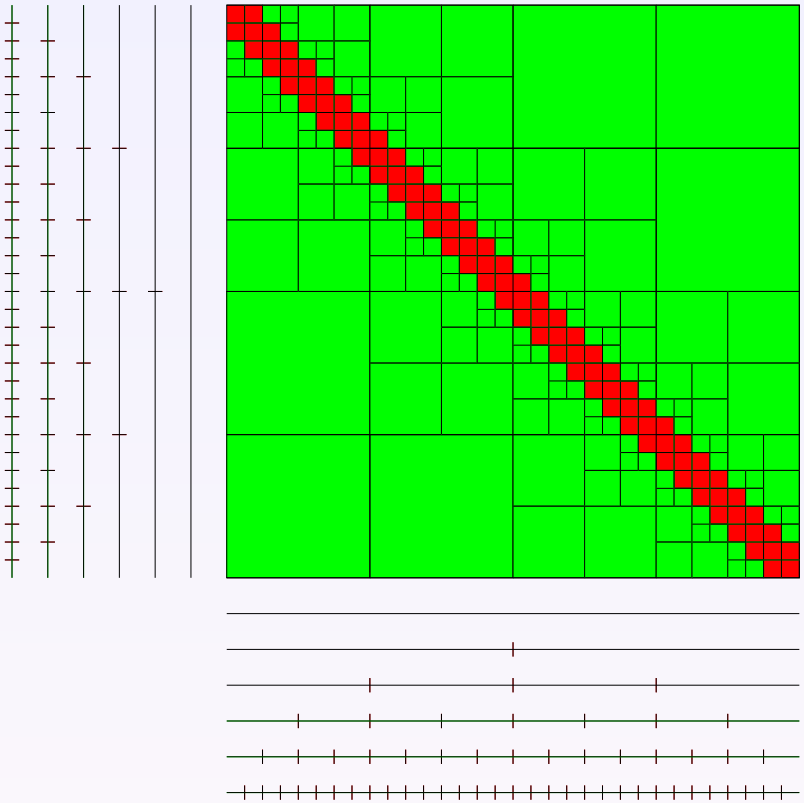
$$(E_{t'})_{\nu'\nu} = \mathcal{L}_{t,\nu}(\xi_{t',\nu'}).$$

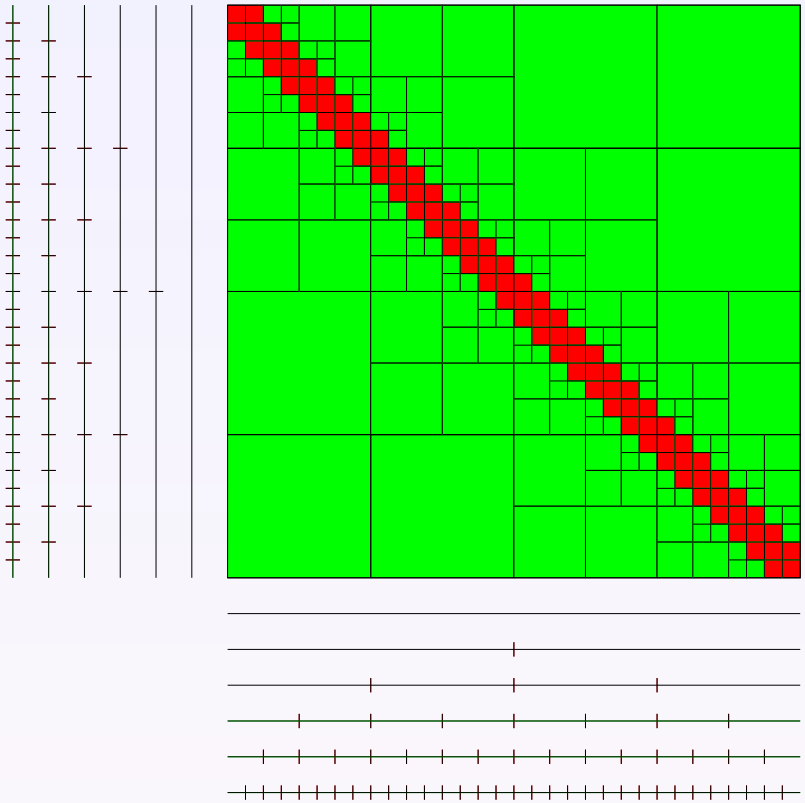
Nested basis defined by

$$(V_t)_{i\nu} = \sum_{\nu'} (V_{t'})_{i\nu'} (E_{t'})_{\nu'\nu}.$$

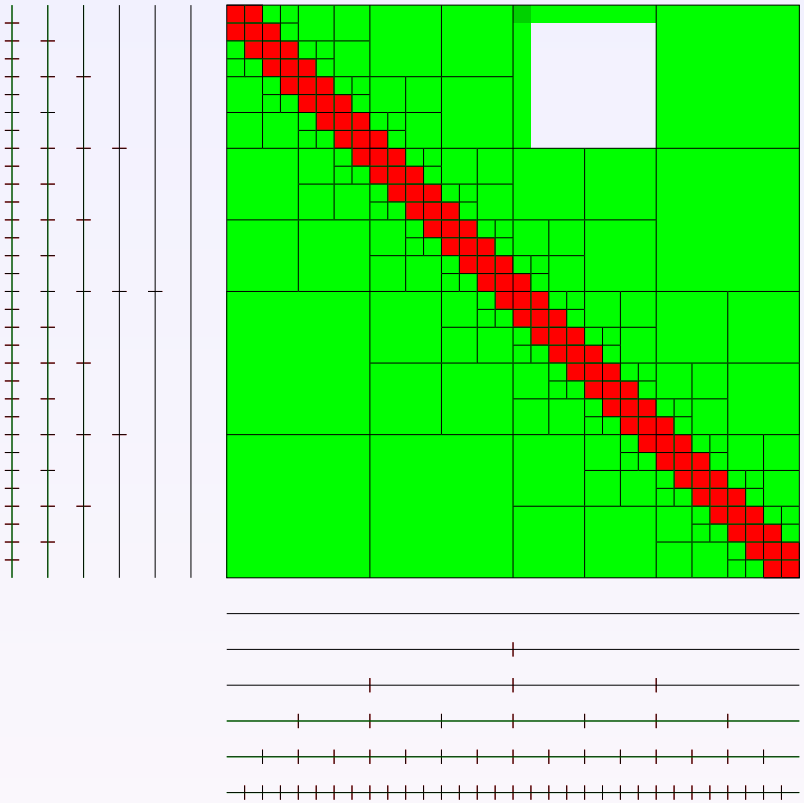
Idea: Do not store V_t, store transfer matrix $E_{t'}$ instead.
$\rightarrow k^2$ coefficients instead of $(\#\hat{t})k$.

$\mathcal{H}^2$-matrix representation

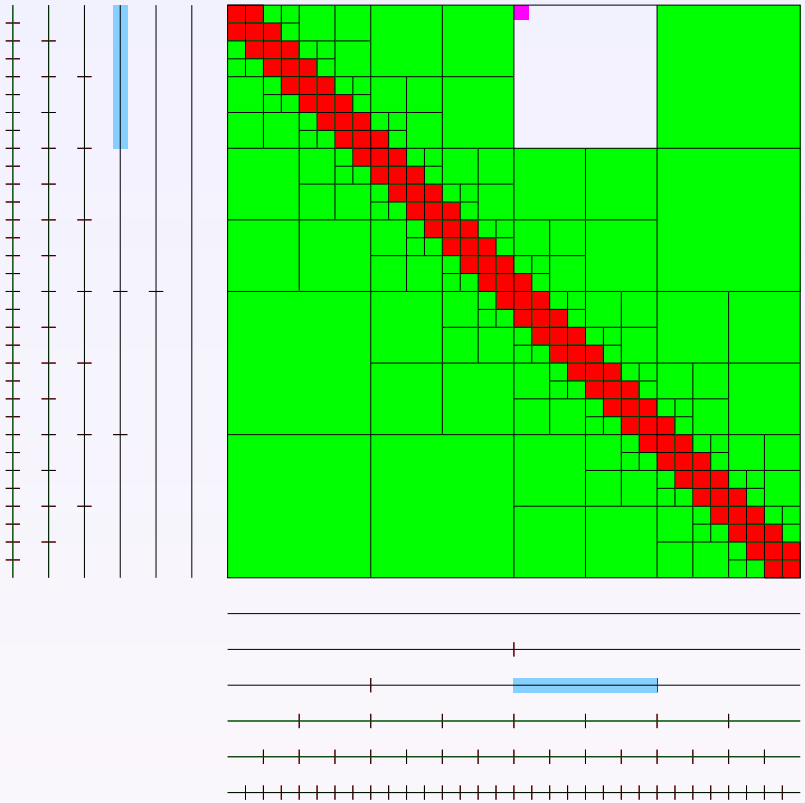




$\mathcal{H}^2$-matrix representation

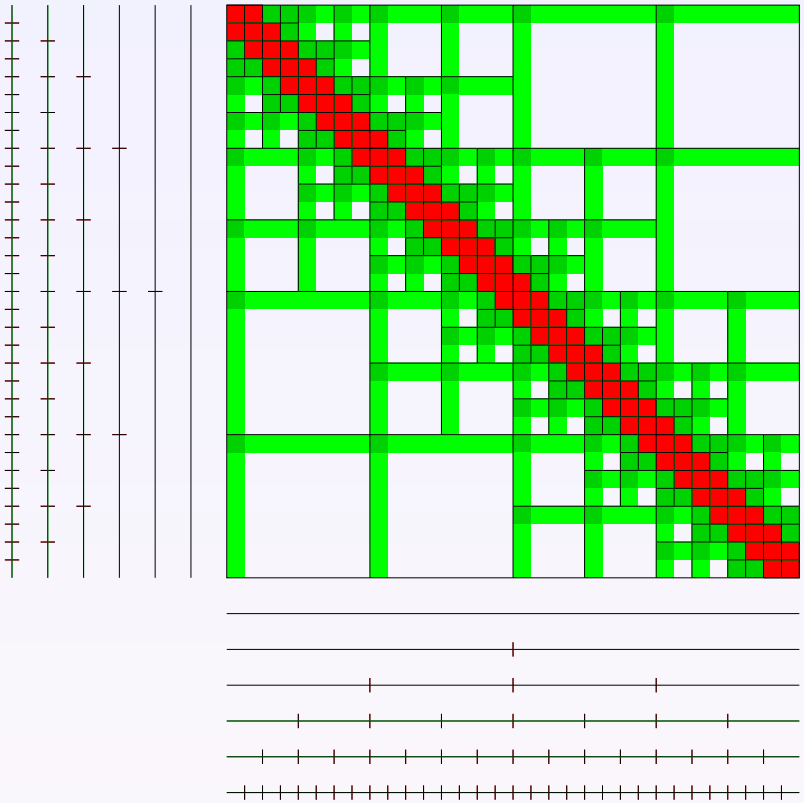


$$G|_{\hat{t}\times\hat{s}} \approx A_b B_b^*$$

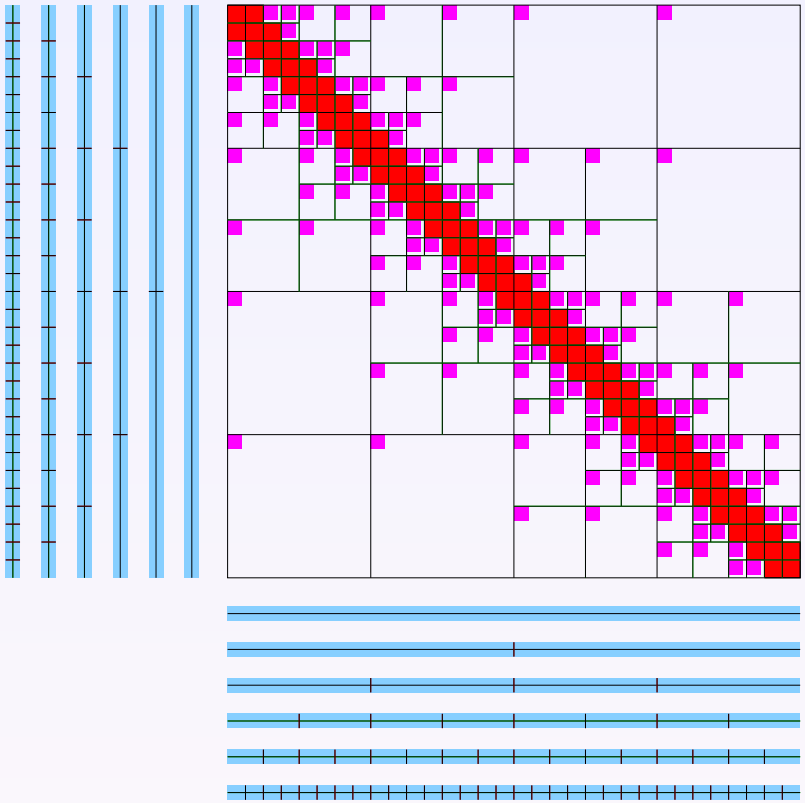


$$G|_{\hat{t}\times\hat{s}} \approx V_t S_b W_s^*$$

$\mathcal{H}^2$-matrix representation

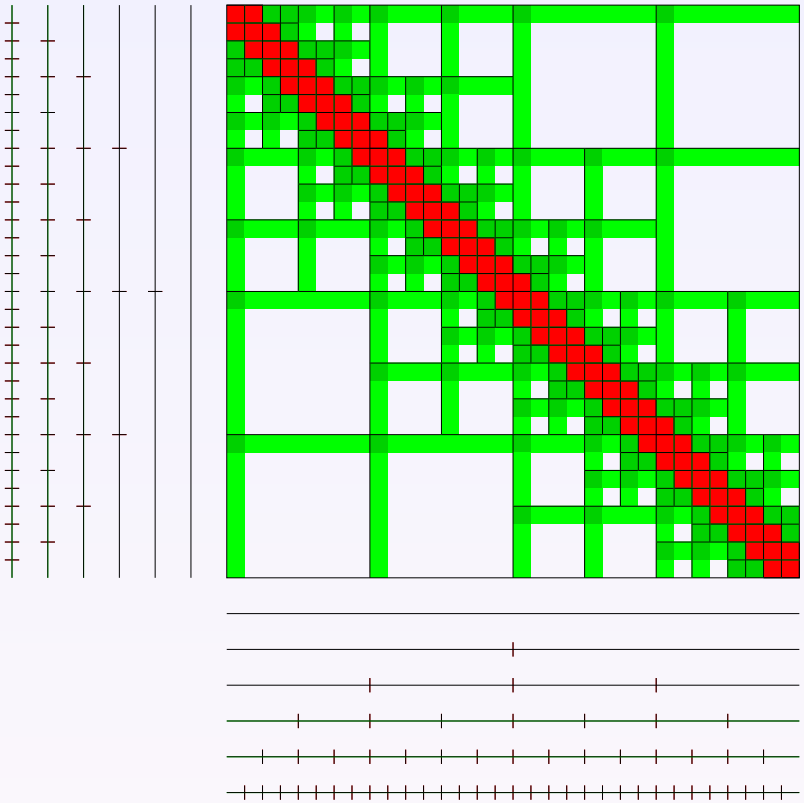


$$G|_{\hat{t}\times\hat{s}} \approx A_b B_b^*$$



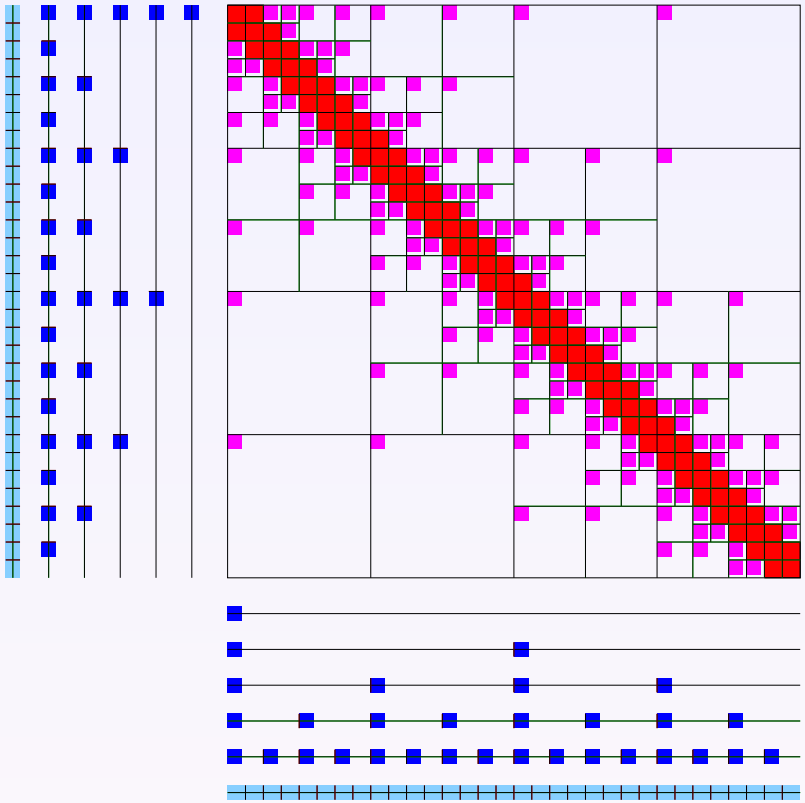
$$G|_{\hat{t}\times\hat{s}} \approx V_t S_b W_s^*$$

$\mathcal{H}^2$-matrix representation



$$G|_{\hat{t}\times\hat{s}} \approx A_b B_b^*$$

Complexity $\mathcal{O}(nk \log n)$



$$G|_{\hat{t}\times\hat{s}} \approx V_t S_b W_s^*$$
$$V_t|_{\hat{t}'\times k} = V_{t'} E_{t'}$$

Complexity $\mathcal{O}(nk)$

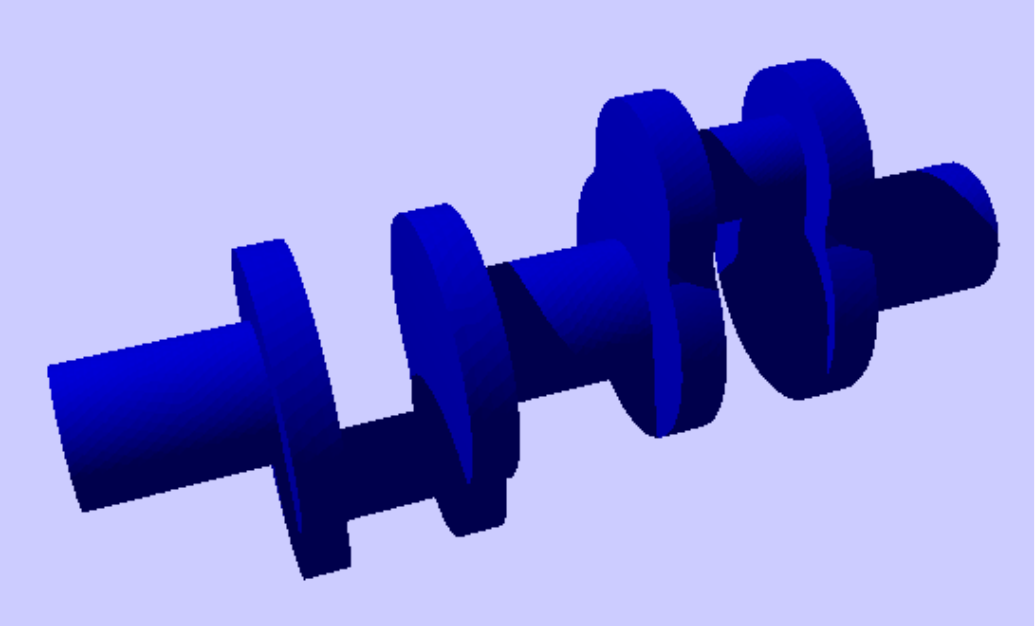
Experiment: How important is one logarithm?

Problem: One-dimensional model problem, approximated by $\mathcal{H}$- and $\mathcal{H}^2$-matrices using interpolation.

	$\mathcal{H}$-matrix			$\mathcal{H}^2$-matrix			$\mathcal{H}^2$-matrix (var.)		
n	Bld	M$/n$	Err	Bld	M$/n$	Err	Bld	M$/n$	Err
128	0.00	0.6	2.8_{-5}	0.00	0.5	3.0_{-5}	0.00	0.5	1.9_{-4}
256	0.01	0.8	2.7_{-5}	0.00	0.5	3.2_{-5}	0.00	0.5	1.1_{-4}
512	0.02	1.0	2.6_{-5}	0.01	0.5	3.2_{-5}	0.01	0.5	5.7_{-5}
1024	0.04	1.2	2.6_{-5}	0.01	0.5	3.2_{-5}	0.01	0.5	2.9_{-5}
2048	0.10	1.4	2.6_{-5}	0.03	0.5	3.3_{-5}	0.03	0.5	1.4_{-5}
4096	0.22	1.6	2.6_{-5}	0.05	0.5	3.3_{-5}	0.05	0.5	7.3_{-6}
8192	0.50	1.8	2.6_{-5}	0.10	0.5	3.3_{-5}	0.10	0.5	3.6_{-6}
⋮	⋮	⋮		⋮	⋮		⋮	⋮	
524288	52.97	2.9		6.62	0.5		6.62	0.5	

Experiment: How far can we go?

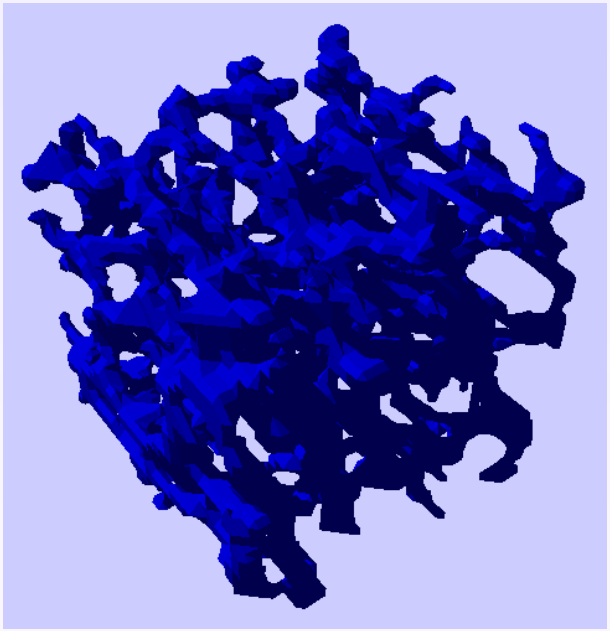
Idea: Cross approximation and $\mathcal{H}^2$-matrix recompression.



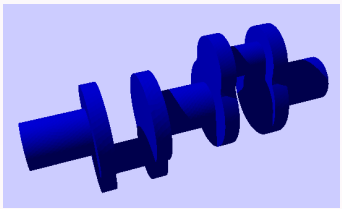
Crank shaft,
taken from
Joachim Schöberl's
NetGen package

Experiment: How far can we go?

Idea: Cross approximation and $\mathcal{H}^2$-matrix recompression.



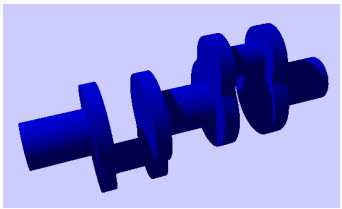
Nickle foam,
courtesy of
von Heiko Andrä
and Günther Of

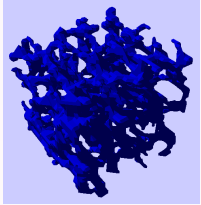


Experiment: How far can we go?

Idea: Cross approximation and $\mathcal{H}^2$-matrix recompression.

	$\mathcal{H}$-matrix			$\mathcal{H}^2$-matrix		
n	Build	Mem	Error	Build	Mem	Error
25744	67.0	181.0	1.9_{-8}	67.9	135.4	1.4_{-8}
102976	283.0	1135.7	1.2_{-9}	288.6	550.7	8.4_{-10}
411904	1416.2	6756.9	5.9_{-11}	1464.3	2224.6	5.1_{-11}
1647616	6247.5	37196.1	3.8_{-12}	6626.6	9795.4	3.0_{-12}
6590464	33981.7	198867.2	2.2_{-13}	36224.1	41610.5	1.8_{-13}
28952	177.6	294.3	8.8_{-8}	172.4	387.4	7.0_{-8}
115808	681.9	1840.8	5.8_{-9}	699.7	1228.1	3.7_{-9}
463232	3144.6	11364.9	2.8_{-10}	3175.6	4933.0	2.2_{-10}
1852928	17384.1	69570.8	1.2_{-11}	17628.9	15905.2	1.3_{-11}





Storage: Coupling matrices

Block cluster required to be sparse:

$$\#\{s \in \mathcal{T}_\mathcal{J} \ : \ (t,s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}}\} \leq C_{\mathrm{sp}} \qquad \text{for all } t \in \mathcal{T}_\mathcal{I}$$

Storage for coupling matrices:

$$\sum_{(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}} k^2 = \sum_{t\in\mathcal{T}_\mathcal{I}} \sum_{\substack{s\in\mathcal{T}_\mathcal{J}\\(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}}} k^2 \leq C_{\mathrm{sp}} \sum_{t\in\mathcal{T}_\mathcal{I}} k^2 = C_{\mathrm{sp}} k^2 \#\mathcal{T}_\mathcal{I}$$

Improvement: If cluster tree regular, i.e.,

$$k \lesssim \#\hat{t} \qquad \text{for all leaves } t \in \mathcal{L}_\mathcal{I},$$

we have $\#\mathcal{L}_\mathcal{I} \lesssim n_\mathcal{I}/k$, and therefore $\#\mathcal{T}_\mathcal{I} \lesssim n_\mathcal{I}/k$.
$\to \mathcal{O}(n_\mathcal{I} k)$ units of storage required.

Storage: Nearfield matrices

Resolution: Maximal size of leaves of the cluster trees.

$$r_{\mathcal{I}} := \max\{\#\hat{t} \ : \ t \in \mathcal{L}_{\mathcal{I}}\}, \qquad r_{\mathcal{J}} := \max\{\#\hat{s} \ : \ s \in \mathcal{L}_{\mathcal{J}}\}.$$

Usually controlled by the user when constructing the cluster tree.

Admissible block tree: For inadmissible leaves $b = (t, s) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}$, both t and s are leaves.

$$(\#\hat{t})(\#\hat{s}) \leq (\#\hat{t})r_{\mathcal{J}}, \qquad t \in \mathcal{L}_{\mathcal{I}}, \qquad \text{for all } b = (t, s) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}.$$

Storage for nearfield matrices:

$$\sum_{(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}} (\#\hat{t})(\#\hat{s}) \leq \sum_{t\in\mathcal{L}_{\mathcal{I}}} \sum_{\substack{s\in\mathcal{T}_{\mathcal{J}} \\ (t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}}} (\#\hat{t})r_{\mathcal{J}} \leq C_{\mathrm{sp}} r_{\mathcal{J}} \sum_{t\in\mathcal{L}_{\mathcal{I}}} \#\hat{t} = C_{\mathrm{sp}} r_{\mathcal{J}} \#\mathcal{I}.$$

$\to \mathcal{O}(n_{\mathcal{I}} r_{\mathcal{J}})$ units of storage required, for $r_{\mathcal{J}} \sim k$ this means $\mathcal{O}(n_{\mathcal{I}} k)$.

Storage: Cluster basis

Leaf matrices only stored for leaves.

$$\sum_{t\in\mathcal{L}_\mathcal{I}} (\#\hat{t})k = k\#\mathcal{I}, \qquad \sum_{s\in\mathcal{L}_\mathcal{J}} (\#\hat{s})k = k\#\mathcal{J}.$$

Transfer matrices stored for all clusters.

$$\sum_{t\in\mathcal{T}_\mathcal{I}} k^2 = k^2\#\mathcal{T}_\mathcal{I}, \qquad \sum_{s\in\mathcal{T}_\mathcal{J}} k^2 = k^2\#\mathcal{T}_\mathcal{J}.$$

Improvement: If cluster tree regular, we have

$$\sum_{t\in\mathcal{T}_\mathcal{I}} k^2 = k^2\#\mathcal{T}_\mathcal{I} \lesssim n_\mathcal{I} k, \qquad \sum_{s\in\mathcal{T}_\mathcal{J}} k^2 = k^2\#\mathcal{T}_\mathcal{J} \lesssim n_\mathcal{J} k.$$

$\to \mathcal{O}(n_\mathcal{I} k + n_\mathcal{J} k)$ units of storage required.

Summary: Storage requirements

Notation:

- Number of rows and columns: $n_{\mathcal{I}} := \#\mathcal{I}$, $n_{\mathcal{J}} := \#\mathcal{J}$.
- Number of clusters: $c_{\mathcal{I}} := \#\mathcal{T}_{\mathcal{I}}$, $c_{\mathcal{J}} := \#\mathcal{T}_{\mathcal{J}}$.
- Regular trees: $c_{\mathcal{I}} \lesssim n_{\mathcal{I}}/k$, $c_{\mathcal{J}} \lesssim n_{\mathcal{J}}/k$.

Coupling matrices: $\mathcal{O}(c_{\mathcal{I}} k^2)$ or $\mathcal{O}(c_{\mathcal{J}} k^2)$ in general,
$\mathcal{O}(n_{\mathcal{I}} k)$ or $\mathcal{O}(n_{\mathcal{J}} k)$ for regular cluster trees.

Nearfield matrices: $\mathcal{O}(n_{\mathcal{I}} r_{\mathcal{J}})$ or $\mathcal{O}(n_{\mathcal{J}} r_{\mathcal{I}})$.

Cluster bases: $\mathcal{O}(c_{\mathcal{I}} k^2 + n_{\mathcal{I}} k)$ and $\mathcal{O}(c_{\mathcal{J}} k^2 + n_{\mathcal{J}} k)$ in general,
$\mathcal{O}(n_{\mathcal{I}} k)$ and $\mathcal{O}(n_{\mathcal{J}} k)$ for regular cluster trees.

Important: Differently from $\mathcal{H}$-matrix case, the depth of the trees is irrelevant for $\mathcal{H}^2$-matrices.

Matrix-vector multiplication

Goal: Compute product of $\mathcal{H}^2$-matrix and vector.

$$y \leftarrow y + Gx, \qquad y \leftarrow y + G^* x.$$

Naive algorithm: Treat all blocks individually.

$$y|_{\hat{t}} \leftarrow y|_{\hat{t}} + G|_{\hat{t}\times\hat{s}} x|_{\hat{s}} \qquad \text{for all } b = (t,s) \in \mathcal{L}_{\mathcal{I}\times\mathcal{J}},$$
$$y|_{\hat{t}} \leftarrow y|_{\hat{s}} + V_t S_b W_s^* x|_{\hat{s}} \qquad \text{for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}.$$

Problems: Complexity too high, V_t and W_s only available in leaves.

Idea: Perform multiplications by W_s^*, S_b and V_t in separate steps.

Forward transformation

Goal: Compute the auxiliary vectors

$$\hat{x}_s := W_s^* x|_{\hat{s}} \qquad \text{for all } s \in \mathcal{T}_{\mathcal{J}}.$$

Leaf clusters: W_s available, compute $\hat{x}_s$ directly.

Non-leaf clusters: Take advantage of nested structure. Simple case:

$$W_s = \begin{pmatrix} W_{s_1} F_{s_1} \\ W_{s_2} F_{s_2} \end{pmatrix}$$

Recursion: Compute $\hat{x}_{s_1}$ and $\hat{x}_{s_2}$ first, then use

$$\hat{x}_s = W_s^* x|_{\hat{s}} = \begin{pmatrix} F_{s_1}^* & F_{s_2}^* \end{pmatrix} \begin{pmatrix} W_{s_1}^* x|_{\hat{s}_1} \\ W_{s_2}^* x|_{\hat{s}_2} \end{pmatrix} = F_{s_1}^* \hat{x}_{s_1} + F_{s_2}^* \hat{x}_{s_2}.$$

Interaction step

Admissible blocks:

$$\hat{y}_t \leftarrow \hat{y}_t + S_b \hat{x}_s \qquad \text{for all } b = (t,s) \in \mathcal{L}^+_{\mathcal{I}\times\mathcal{J}}.$$

Inadmissible blocks:

$$y|_{\hat{t}} \leftarrow y|_{\hat{t}} + G|_{\hat{t}\times\hat{s}} x|_{\hat{s}} \qquad \text{for all } b = (t,s) \in \mathcal{L}^-_{\mathcal{I}\times\mathcal{J}}.$$

Backward transformation

Goal: Compute the final result

$$y|_{\hat{t}} \leftarrow y|_{\hat{t}} + V_t \hat{y}_t \qquad \text{for all } t \in \mathcal{T}_{\mathcal{I}}.$$

Non-leaf clusters: Add coefficients to sons.

$$\begin{pmatrix} V_{t_1} \hat{y}_{t_1} \\ V_{t_2} \hat{y}_{t_2} \end{pmatrix} + V_t \hat{y}_t = \begin{pmatrix} V_{t_1} \hat{y}_{t_1} + V_{t_1} E_{t_1} \hat{y}_t \\ V_{t_2} \hat{y}_{t_2} + V_{t_2} E_{t_2} \hat{y}_t \end{pmatrix} = \begin{pmatrix} V_{t_1} (\hat{y}_{t_1} + E_{t_1} \hat{y}_t) \\ V_{t_2} (\hat{y}_{t_2} + E_{t_2} \hat{y}_t) \end{pmatrix}$$

Recursion: Update $\hat{y}_{t_1}$ and $\hat{y}_{t_2}$ first by

$$\hat{y}_{t_1} \leftarrow \hat{y}_{t_1} + E_{t_1} \hat{y}_t, \qquad \hat{y}_{t_2} \leftarrow \hat{y}_{t_2} + E_{t_2} \hat{y}_t,$$

then proceed recursively to t_1 and t_2.

Summary: Matrix-vector multiplication

Forward transformation: Computes $\hat{x}_s = W_s^* x|_{\hat{s}}$ by recursion from the leaves towards the root.

Interaction step: Computes far- and nearfield interactions.

Backward transformation: Computes $y|_{\hat{t}} \leftarrow y|_{\hat{s}} + V_t \hat{y}_t$ by recursion from the root towards the leaves.

Complexity: Each coefficient in (S_b), (V_t), (W_s) and nearfield requires not more than two operations.
$\rightarrow \mathcal{O}(n_{\mathcal{I}}(k + r_{\mathcal{J}}))$ or $\mathcal{O}(n_{\mathcal{J}}(k + r_{\mathcal{I}}))$ operations.

supermatrix

Required: Add `clusterbasis` and `uniformmatrix`.

```
struct _supermatrix {
  int rows;
  int cols;
  int block_rows;
  int block_cols;
  pclusterbasis row;
  pclusterbasis col;
  puniformmatrix u;
  prkmatrix r;
  pfullmatrix f;
  psupermatrix* s;
};
```

uniformmatrix

Goal: Matrix representation by $G = V_t S_b W_s^*$.

```
struct _uniformmatrix {
  int rows;
  int cols;
  int kr;
  int kc;
  double *S;
  pclusterbasis row;
  pclusterbasis col;
};
```

clusterbasis

Goal: Store V_t for leaves and $E_{t'}$ for non-leaves.

```
struct _clusterbasis {
  pccluster t;
  double **T;
  double *V;
  double *xt;
  double *yt;
  int k;
  int n;
  int sons;
  pclusterbasis *son;
};
```