$\mathcal{H}^2$-Matrices for boundary integral equations

Steffen Börm
(with Sven Christophersen, Jessica Gördes and Knut Reimer)
funded in part by DFG grants BO 3289-2/1 and BO 3289-4/1

Christian-Albrechts-Universität zu Kiel

BEM++ Workshop, UC London, 19th of September, 2013

Overview

Overview

Model problem

Simple example: Given $f : [0,1] \to \mathbb{R}$, find $u : [0,1] \to \mathbb{R}$ such that

$$\int_0^1 g(x,y)u(y)\,dy = f(x), \qquad g(x,y) := -\log|x-y|.$$

Galerkin discretization using piecewise constant basis functions $\varphi_i = 1_{[(i-1)/n,i/n]}$ leads to matrix $G \in \mathbb{R}^{n\times n}$ with

$$g_{ij} = \int_{(i-1)/n}^{i/n} \int_{(j-1)/n}^{j/n} g(x,y)\,dy\,dx.$$

Challenges:

- G is dense, standard representation requires n^2 units of storage.
- G becomes ill-conditioned for large n, so Krylov methods converge only slowly.

Degenerate kernel functions

Idea: Approximate kernel function in $\tau \times \sigma$ by tensor products.

$$g(x,y) \approx \sum_{\nu=1}^{k} a_{\tau\sigma,\nu}(x) b_{\tau\sigma,\mu}(y) \qquad \text{for all } x \in \tau,\ y \in \sigma.$$

Discretization of degenerate kernel function yields

$$g_{ij} = \int_{(i-1)/n}^{i/n} \int_{(j-1)/n}^{j/n} g(x,y)\, dy\, dx$$

Degenerate kernel functions

Idea: Approximate kernel function in $\tau \times \sigma$ by tensor products.

$$g(x,y) \approx \sum_{\nu=1}^{k} a_{\tau\sigma,\nu}(x) b_{\tau\sigma,\mu}(y) \qquad \text{for all } x \in \tau,\ y \in \sigma.$$

Discretization of degenerate kernel function yields

$$g_{ij} \approx \int_{(i-1)/n}^{i/n} \int_{(j-1)/n}^{j/n} \sum_{\nu=1}^{k} a_{\tau\sigma,\nu}(x) b_{\tau\sigma,\mu}(y)\, dy\, dx$$

Degenerate kernel functions

Idea: Approximate kernel function in $\tau \times \sigma$ by tensor products.

$$g(x,y) \approx \sum_{\nu=1}^{k} a_{\tau\sigma,\nu}(x) b_{\tau\sigma,\mu}(y) \qquad \text{for all } x \in \tau,\ y \in \sigma.$$

Discretization of degenerate kernel function yields

$$g_{ij} \approx \sum_{\nu=1}^{k} \int_{(i-1)/n}^{i/n} a_{\tau\sigma,\nu}(x)\,dx \int_{(j-1)/n}^{j/n} b_{\tau\sigma,\nu}(y)\,dy$$

Degenerate kernel functions

Idea: Approximate kernel function in $\tau \times \sigma$ by tensor products.

$$g(x,y) \approx \sum_{\nu=1}^{k} a_{\tau\sigma,\nu}(x) b_{\tau\sigma,\mu}(y) \qquad \text{for all } x \in \tau,\ y \in \sigma.$$

Discretization of degenerate kernel function yields

$$g_{ij} \approx \sum_{\nu=1}^{k} \underbrace{\int_{(i-1)/n}^{i/n} a_{\tau\sigma,\nu}(x)\,dx}_{=:a_{\tau\sigma,i\nu}} \underbrace{\int_{(j-1)/n}^{j/n} b_{\tau\sigma,\nu}(y)\,dy}_{=:b_{\tau\sigma,j\nu}} = (A_{\tau\sigma} B_{\tau\sigma}^*)_{ij}$$

Result: $G|_{\hat{\tau}\times\hat{\sigma}} \approx A_{\tau\sigma} B_{\tau\sigma}^*$ with $\hat{\tau}, \hat{\sigma} \subseteq \{1,\ldots,n\}$ such that

$$i \in \hat{\tau} \Rightarrow [(i-1)/n, i/n] \subseteq \tau, \qquad j \in \hat{\sigma} \Rightarrow [(j-1)/n, j/n] \subseteq \sigma.$$

Advantage: $\mathcal{O}((\#\hat{\tau} + \#\hat{\sigma})k)$ storage and runtime.

Example: Interpolation

Idea: Lagrangian interpolation yields degenerate approximation.

$$g(x,y) \approx \sum_{\nu=1}^{k} \mathcal{L}_{\tau,\nu}(x) g(\xi_{\tau,\nu}, y) \qquad \text{for all } x \in \tau,\ y \in \sigma.$$

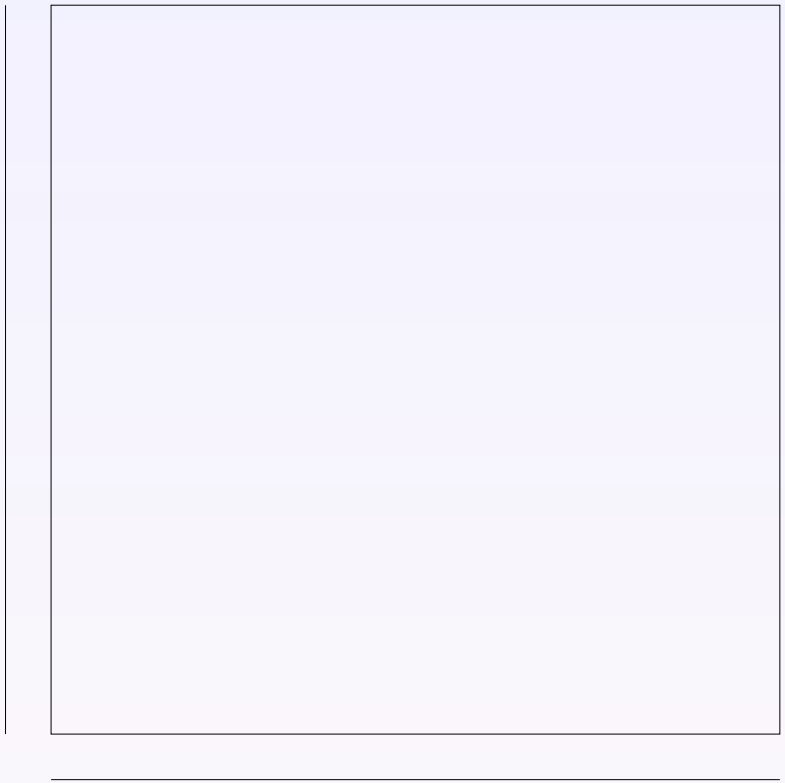
Advantages: Very robust, implementation straightforward.

Approximation error bounded by

$$\|g - \tilde{g}\|_{\infty,\tau\times\sigma} \lesssim \left(\frac{\eta}{\eta+1}\right)^k, \qquad \eta = \frac{\operatorname{diam}(\tau)}{\operatorname{dist}(\tau,\sigma)}.$$

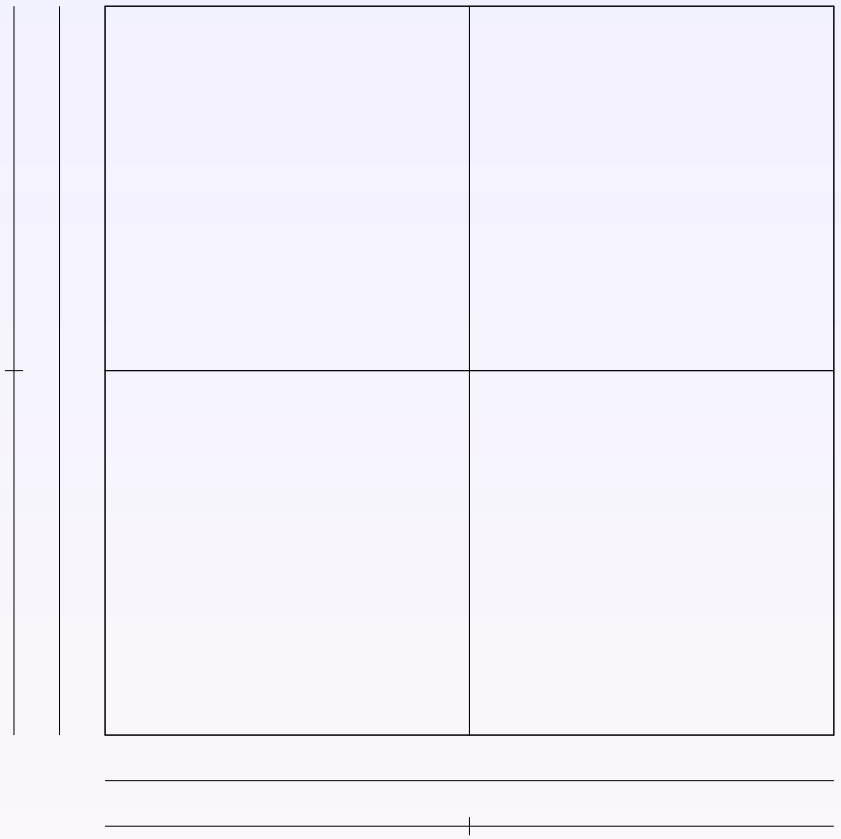
Admissibility: In order to guarantee a uniformly bounded rate of convergence, we have to ensure that η is uniformly bounded.

Hierarchical matrix



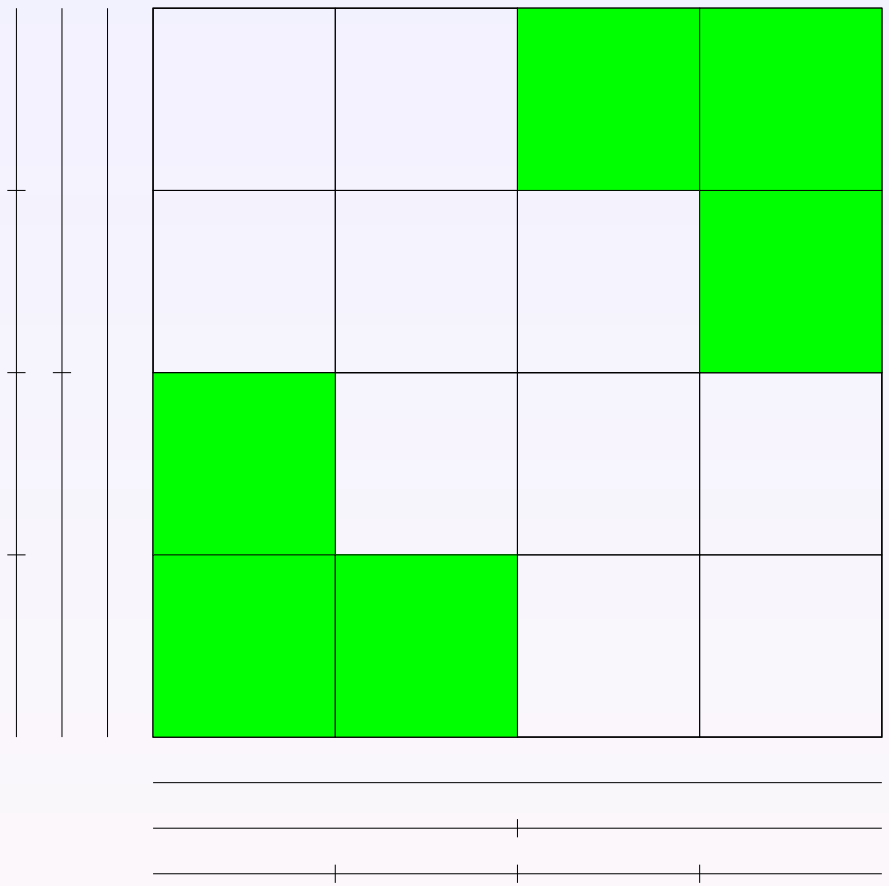
Matrix split into blocks $\hat{\tau} \times \hat{\sigma}$ consisting of clusters $\hat{\tau}$, $\hat{\sigma}$ related to domains τ, σ.

Hierarchical matrix



Matrix split into blocks $\hat{\tau} \times \hat{\sigma}$ consisting of clusters $\hat{\tau}$, $\hat{\sigma}$ related to domains τ, σ.

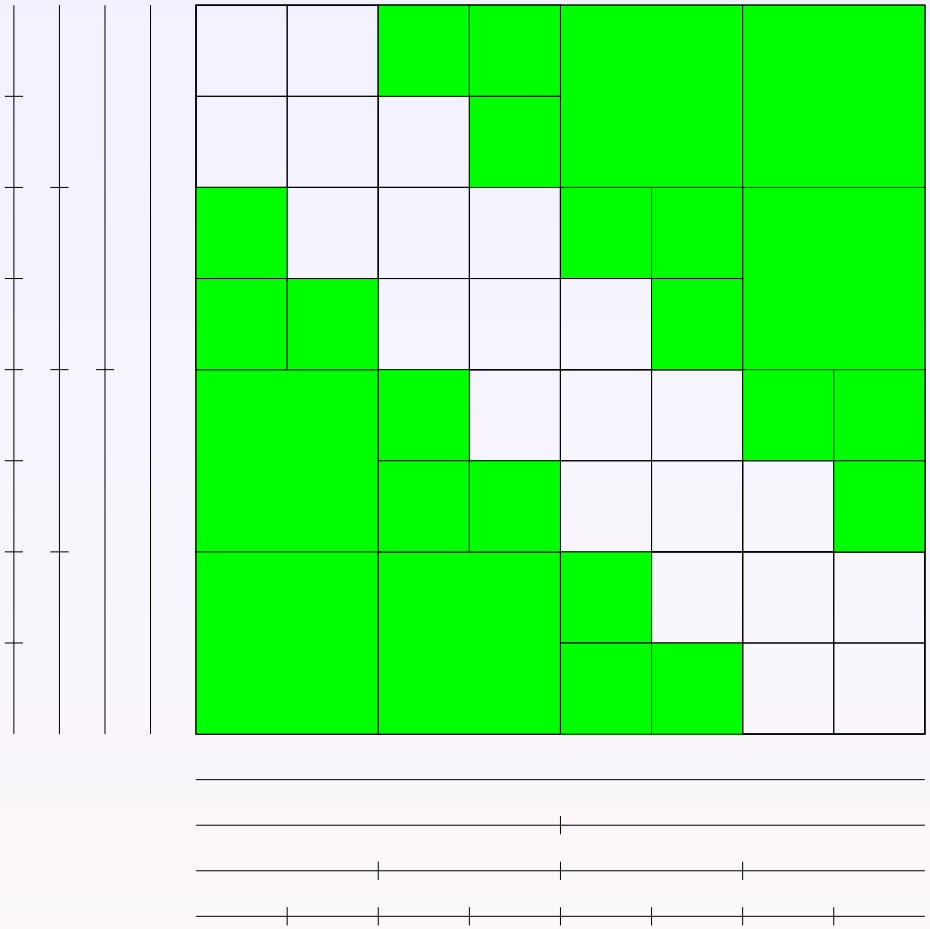
Hierarchical matrix



Matrix split into blocks $\hat{\tau} \times \hat{\sigma}$ consisting of clusters $\hat{\tau}$, $\hat{\sigma}$ related to domains τ, σ.

Clusters chosen from a hierarchical cluster tree.

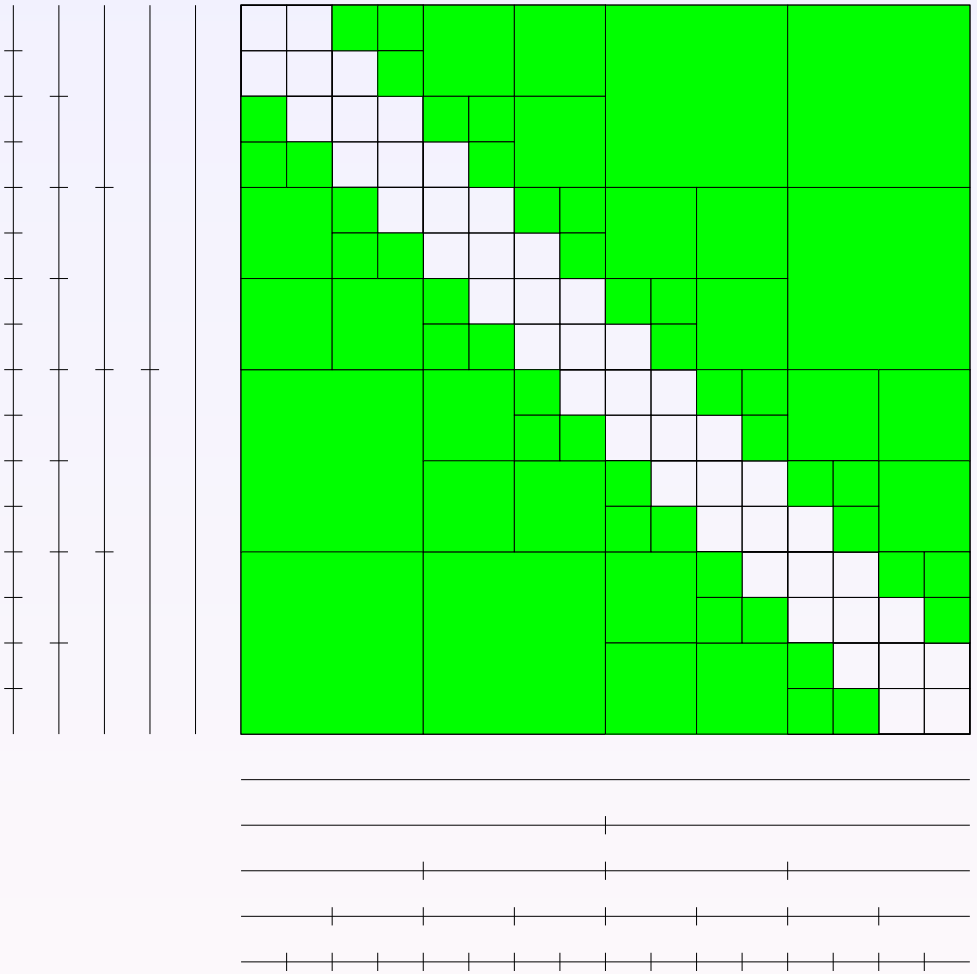
Hierarchical matrix



Matrix split into blocks $\hat{\tau} \times \hat{\sigma}$ consisting of clusters $\hat{\tau}$, $\hat{\sigma}$ related to domains τ, σ.

Clusters chosen from a hierarchical cluster tree.

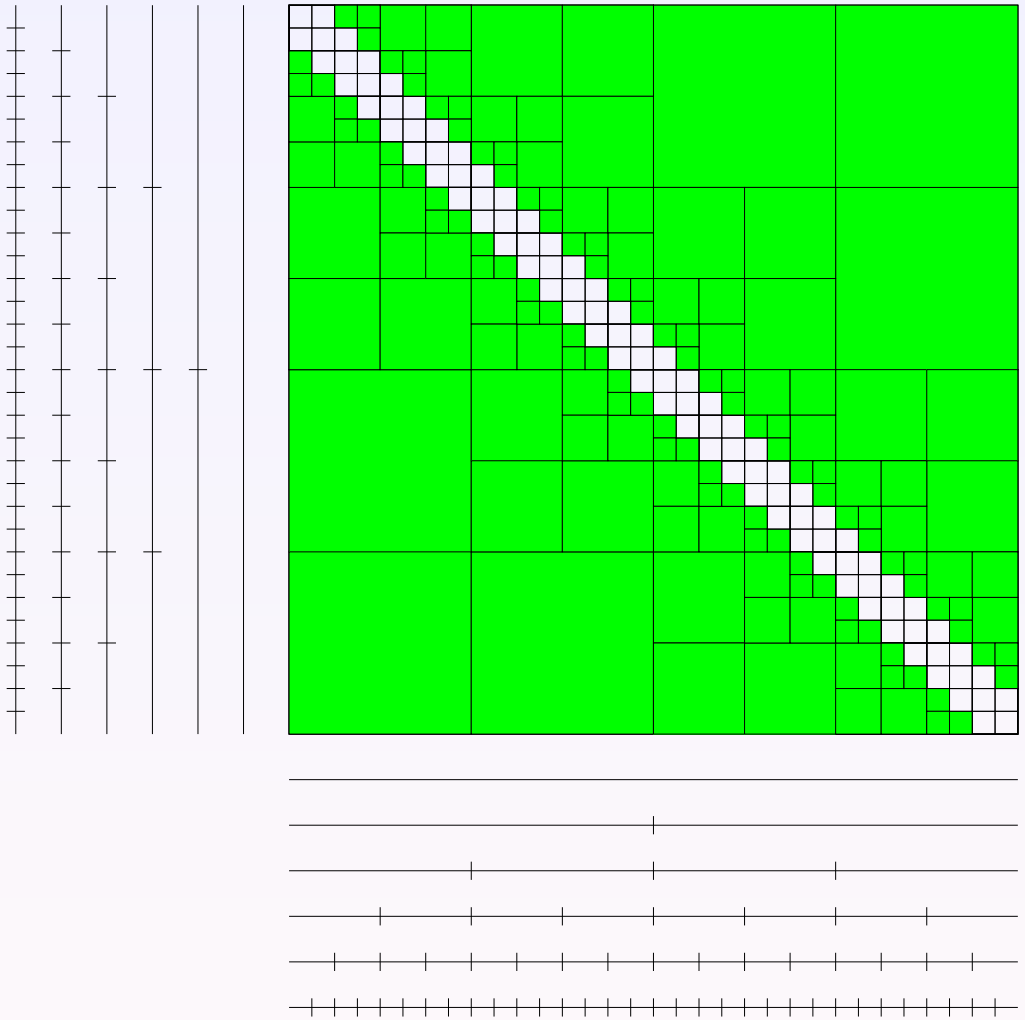
Hierarchical matrix



Matrix split into blocks $\hat{\tau} \times \hat{\sigma}$ consisting of clusters $\hat{\tau}$, $\hat{\sigma}$ related to domains τ, σ.

Clusters chosen from a hierarchical cluster tree.

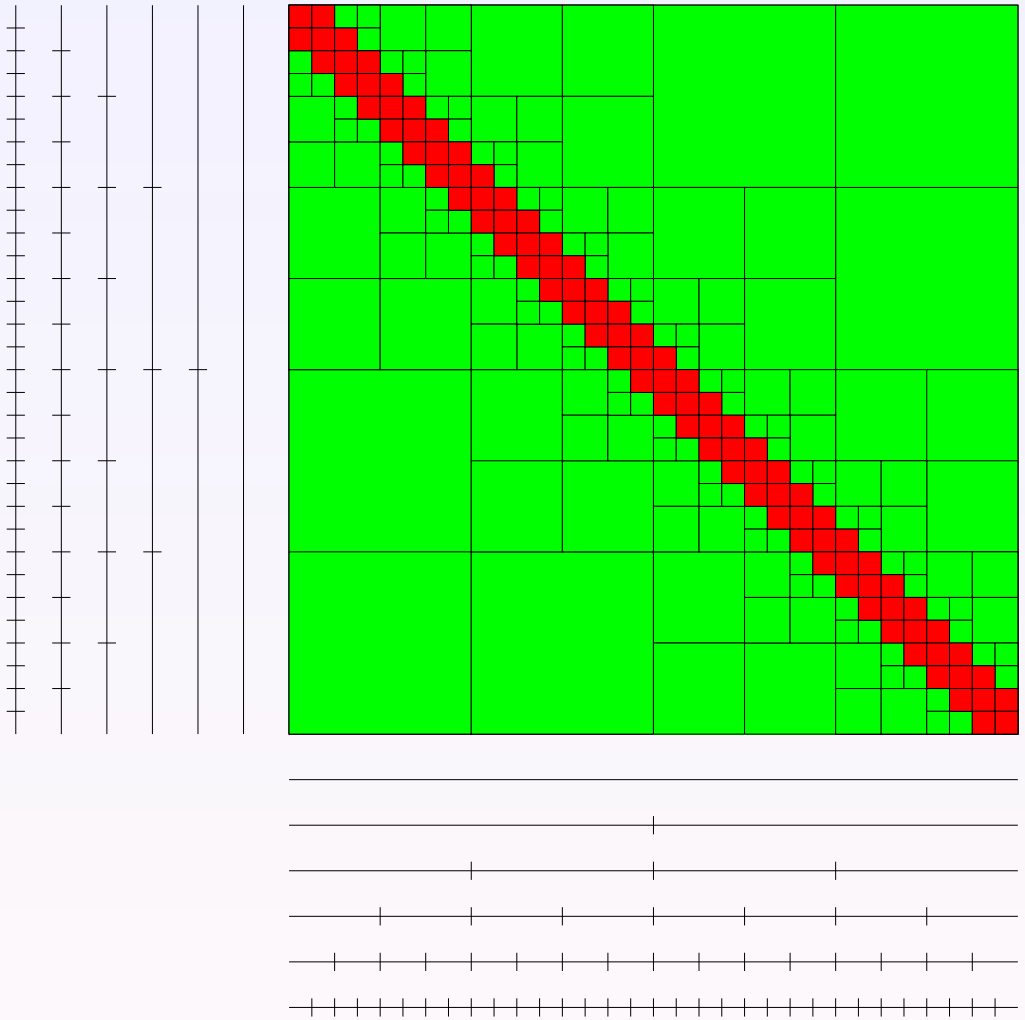
Hierarchical matrix



Matrix split into blocks $\hat{\tau} \times \hat{\sigma}$ consisting of clusters $\hat{\tau}$, $\hat{\sigma}$ related to domains τ, σ.

Clusters chosen from a hierarchical cluster tree.

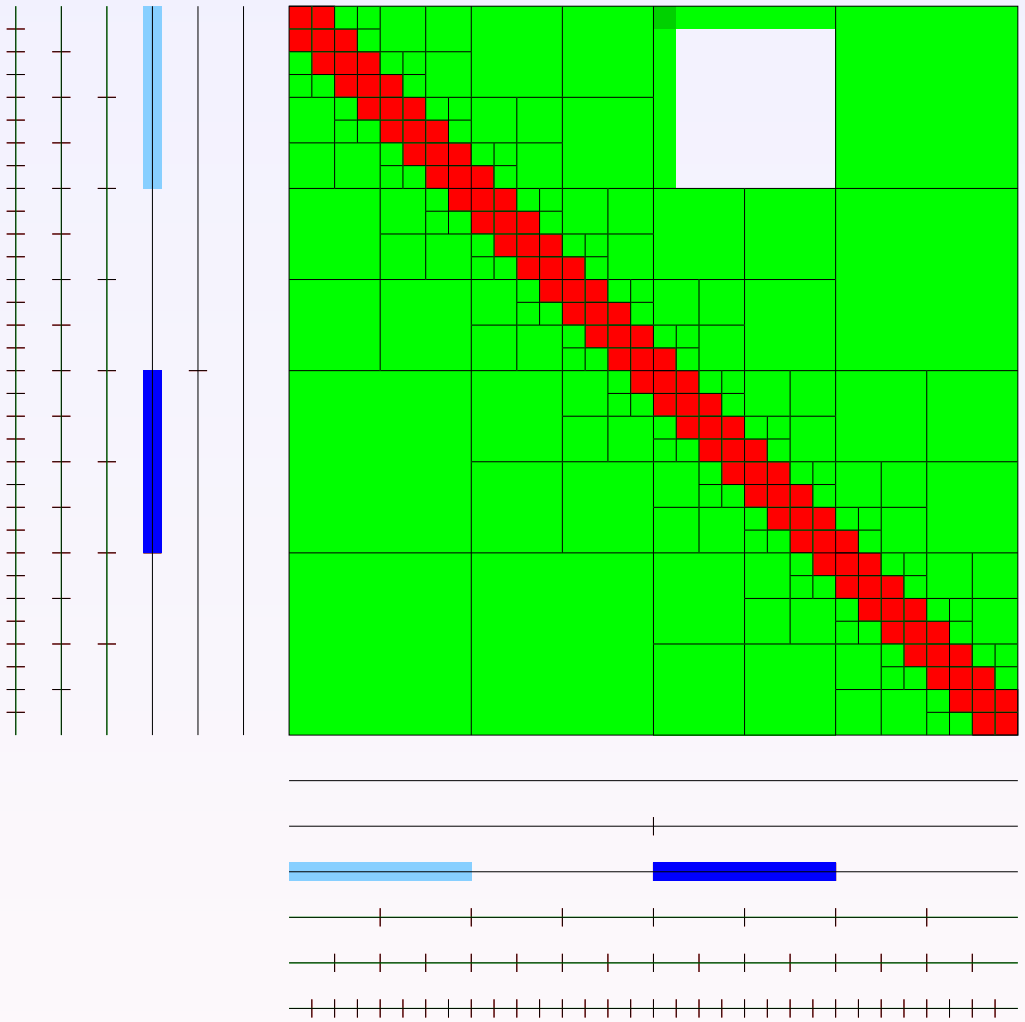
Hierarchical matrix



Matrix split into blocks $\hat{\tau} \times \hat{\sigma}$ consisting of clusters $\hat{\tau}$, $\hat{\sigma}$ related to domains τ, σ.

Clusters chosen from a hierarchical cluster tree.

Hierarchical matrix



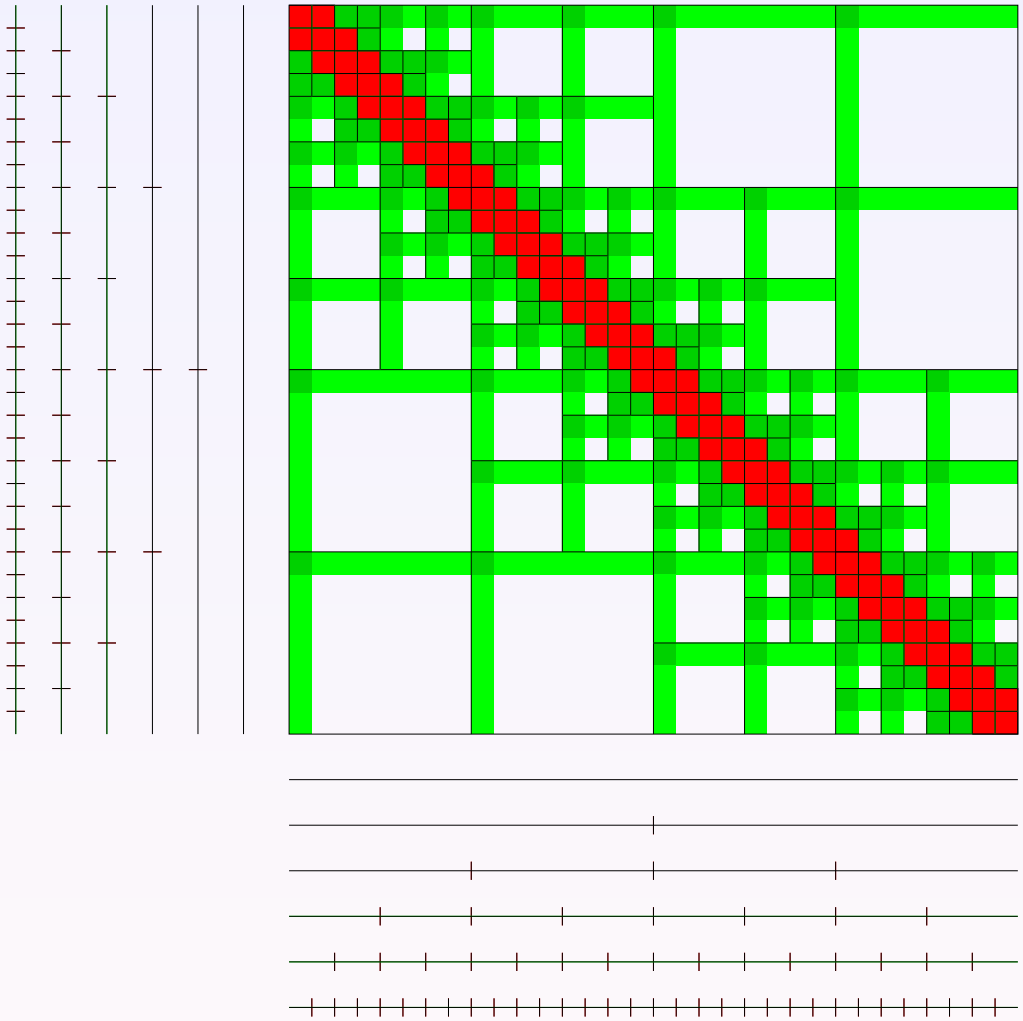
Matrix split into blocks $\hat{\tau} \times \hat{\sigma}$ consisting of clusters $\hat{\tau}$, $\hat{\sigma}$ related to domains τ, σ.

Clusters chosen from a hierarchical cluster tree.

Blocks represented by low-rank factorizations

$$G|_{\hat{\tau}\times\hat{\sigma}} \approx A_{\tau\sigma} B_{\tau\sigma}^{*}.$$

Hierarchical matrix



Matrix split into blocks $\hat{\tau} \times \hat{\sigma}$ consisting of clusters $\hat{\tau}$, $\hat{\sigma}$ related to domains τ, σ.

Clusters chosen from a hierarchical cluster tree.

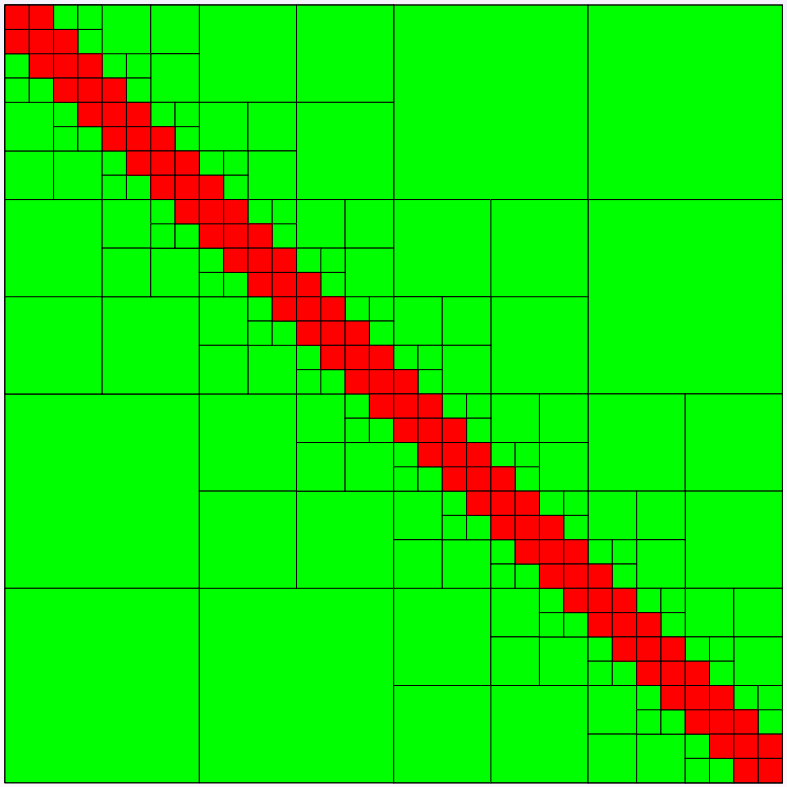
Blocks represented by low-rank factorizations

$$G|_{\hat{\tau}\times\hat{\sigma}} \approx A_{\tau\sigma} B_{\tau\sigma}^*.$$

Result: $\mathcal{O}(nk \log n)$ storage, $\mathcal{O}(nk^\alpha \log^\beta n)$ runtime.

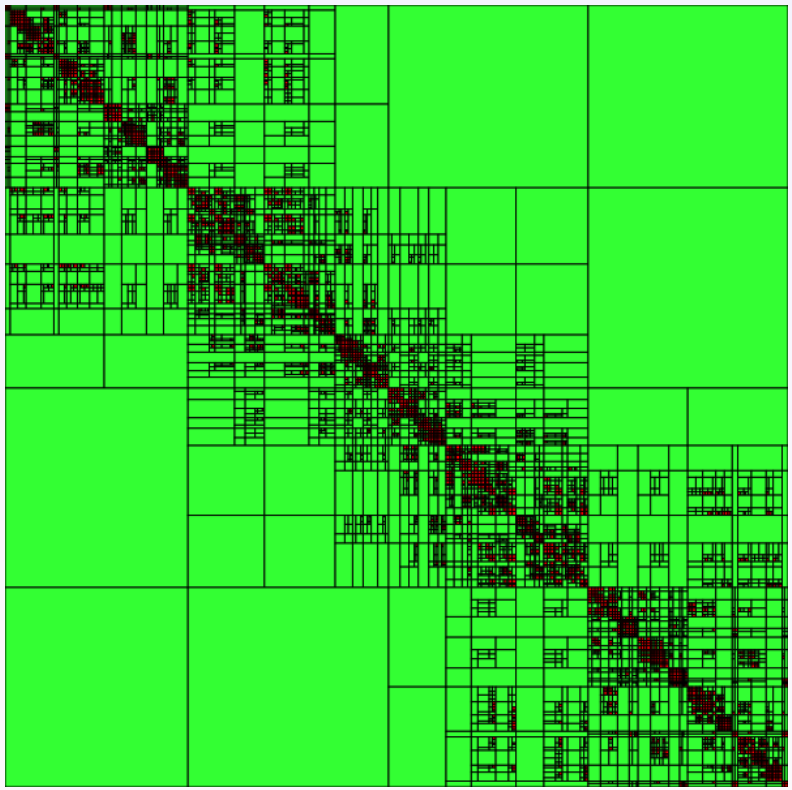
General block structure

Important: In real applications, the block structure depends on the domain and can be quite complicated.



General block structure

Important: In real applications, the block structure depends on the domain and can be quite complicated.



Optimal low-rank approximation

Singular value decomposition exists for any matrix X.

$$X = U \begin{pmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_k & & \\ & & & \sigma_{k+1} & \\ & & & & \ddots \end{pmatrix} V^*$$

U, V orthogonal, singular values $\sigma_1 \geq \sigma_2 \geq \ldots$ in descending order.

Optimal low-rank approximation

Singular value decomposition exists for any matrix X.

$$X = U \begin{pmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_k & & \\ & & & \sigma_{k+1} & \\ & & & & \ddots \end{pmatrix} V^* \approx U \begin{pmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_k & & \\ & & & 0 & \\ & & & & \ddots \end{pmatrix} V^*$$

U, V orthogonal, singular values $\sigma_1 \geq \sigma_2 \geq \ldots$ in descending order.

Dropping small singular values yields low-rank approximation.

Optimal low-rank approximation

Singular value decomposition exists for any matrix X.

$$X = U \begin{pmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_k & & \\ & & & \sigma_{k+1} & \\ & & & & \ddots \end{pmatrix} V^* \approx U \begin{pmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_k & & \\ & & & 0 & \\ & & & & \ddots \end{pmatrix} V^*$$

U, V orthogonal, singular values $\sigma_1 \geq \sigma_2 \geq \ldots$ in descending order.

Dropping small singular values yields low-rank approximation.

Optimality: For any rank-k matrix $\widetilde{X}$ we have

$$\|X - \widetilde{X}\|_2 \geq \sigma_{k+1},$$

equality holds for the SVD-based approximation.

Rank-revealing LR factorization

Idea: Compute partial LR factorization.

$$X = \begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

Rank-revealing LR factorization

Idea: Compute partial LR factorization.

$$X = \begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$
$$= \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & L_{22} R_{22} \end{pmatrix}$$

Rank-revealing LR factorization

Idea: Compute partial LR factorization.

$$X = \begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$
$$\approx \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \end{pmatrix}$$

If $\|L_{22}R_{22}\|$ is small enough, we can drop it and obtain a low-rank approximation.

Rank-revealing LR factorization

Idea: Compute partial LR factorization.

$$X = \begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$
$$\approx \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \end{pmatrix}$$

If $\|L_{22}R_{22}\|$ is small enough, we can drop it and obtain a low-rank approximation.

Advantage: Only the first rows and columns are required.

Rank-revealing LR factorization

Idea: Compute partial LR factorization.

$$X = \begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$
$$\approx \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \end{pmatrix}$$

If $\|L_{22}R_{22}\|$ is small enough, we can drop it and obtain a low-rank approximation.

Advantage: Only the first rows and columns are required.

Disadvantage: LR factorization potentially unstable, particularly without full pivoting.

Rank-revealing LR factorization

Idea: Compute partial LR factorization.

$$X = \begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$
$$\approx \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \end{pmatrix}$$

If $\|L_{22}R_{22}\|$ is small enough, we can drop it and obtain a low-rank approximation.

Advantage: Only the first rows and columns are required.

Disadvantage: LR factorization potentially unstable, particularly without full pivoting.
Full pivoting too time-consuming, so less reliable methods are used.
$\rightarrow$ Adaptive cross approximation (ACA).

References

Panel clustering: Nowak/Hackbusch 1988

Multipole expansion: Rokhlin 1985, Greengard/Rokhlin 1987

Interpolation: B./Grasedyck 2002

Cross approximation: Tyrtyshnikov 1996, Goreinov/Tyrtyshnikov/Zamarashkin 1997, Bebendorf 2000, Bebendorf/Rjasanow 2003, Bebendorf/Grzhibovskis 2006

Arithmetic operations: Hackbusch 1999, Hackbusch/Khoromskij 2000, Grasedyck/Hackbusch 2003

Parallelization: Kriemann 2005, Grasedyck/Kriemann/LeBorne 2008

Overview

Symmetric interpolation

Idea: Use interpolation in both variables.

$$g(x,y) \approx \sum_{\nu=1}^{k} \sum_{\mu=1}^{k} \mathcal{L}_{\tau,\nu}(x) g(\xi_{\tau,\nu}, \xi_{\sigma,\mu}) \mathcal{L}_{\sigma,\mu}(y) \qquad \text{for all } x \in \tau,\ y \in \sigma.$$

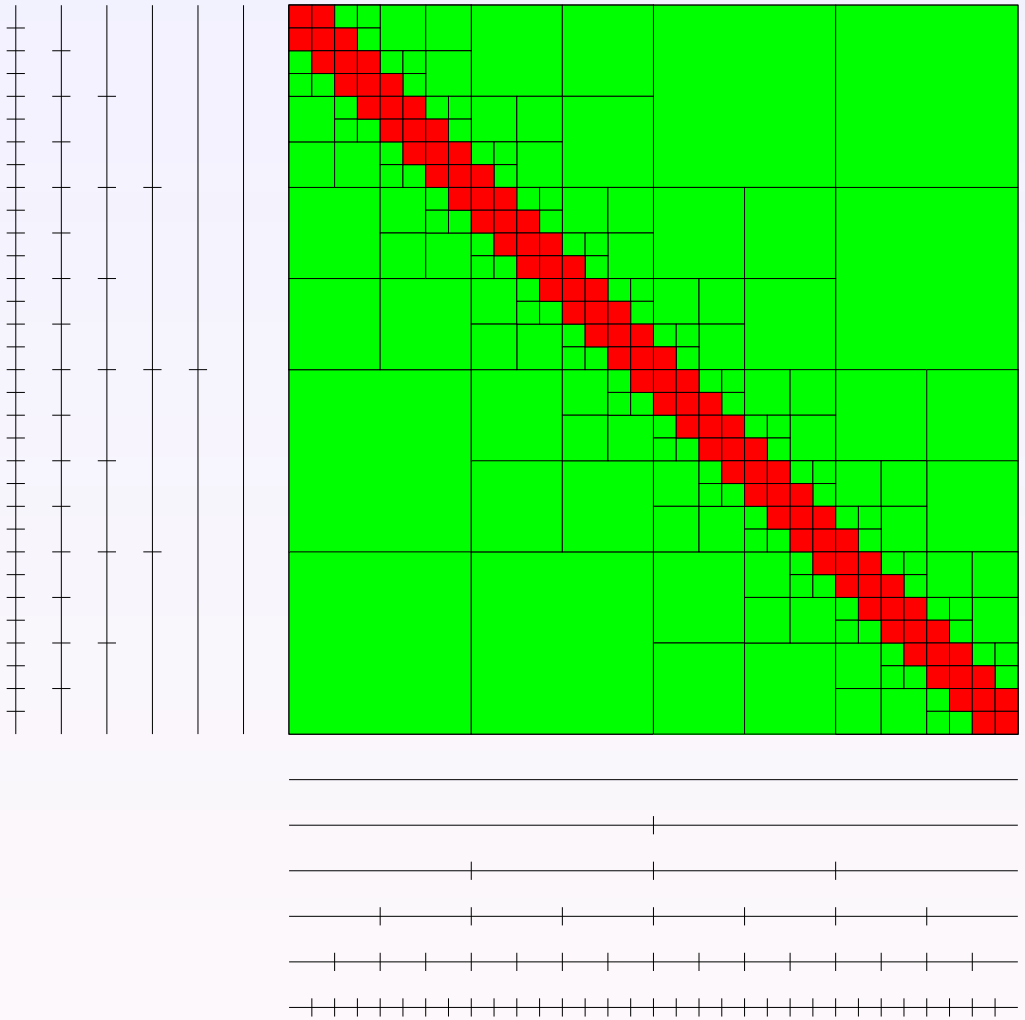
Result: Low-rank factorization in three terms.

$$G|_{\hat{\tau}\times\hat{\sigma}} \approx V_\tau S_{\tau\sigma} W_\sigma^*.$$

Admissibility: Requires $\max\{\operatorname{diam}(\tau), \operatorname{diam}(\sigma)\} \lesssim \operatorname{dist}(\tau,\sigma)$.

Important: V_τ depends only on τ, W_σ depends only on σ.

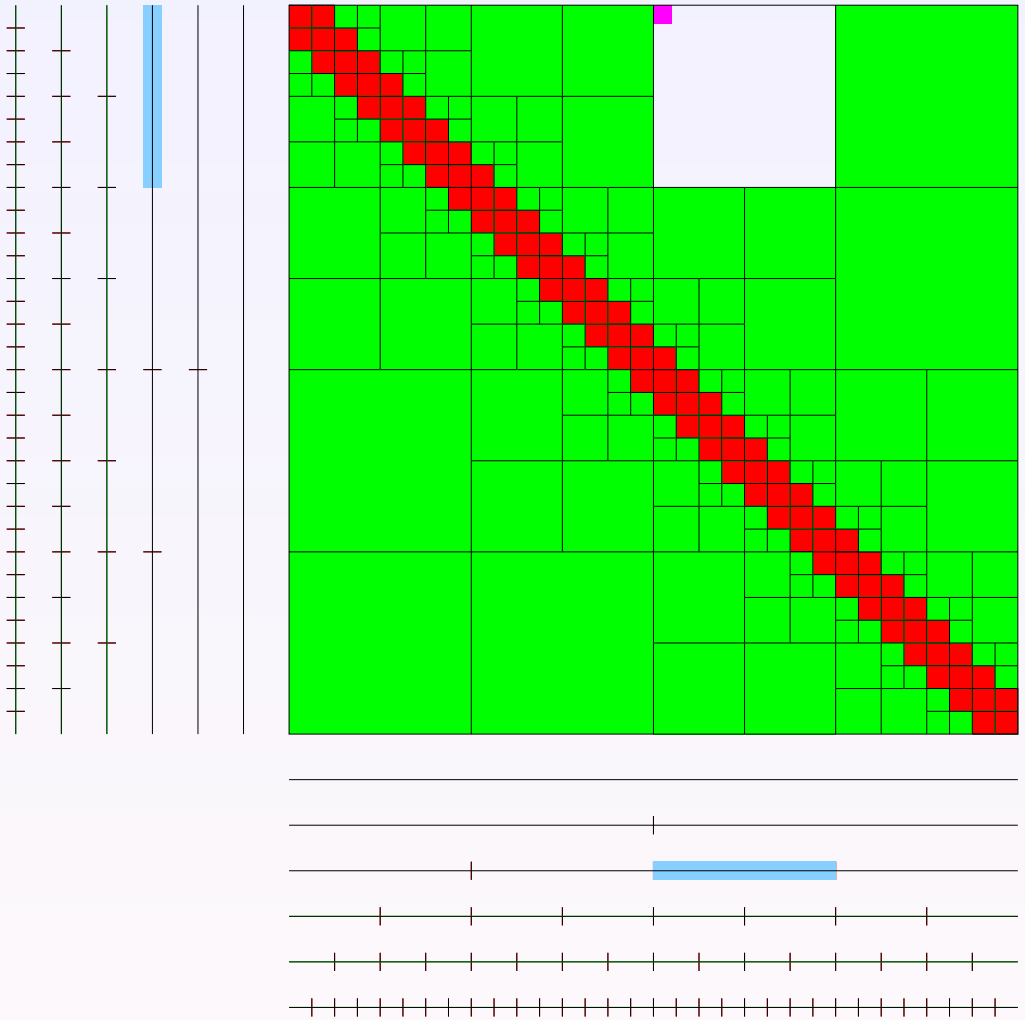
Uniform hierarchical matrix



Idea: Use three factors.

$$G|_{\hat{\tau}\times\hat{\sigma}} \approx V_\tau S_{\tau\sigma} W_\sigma^*.$$

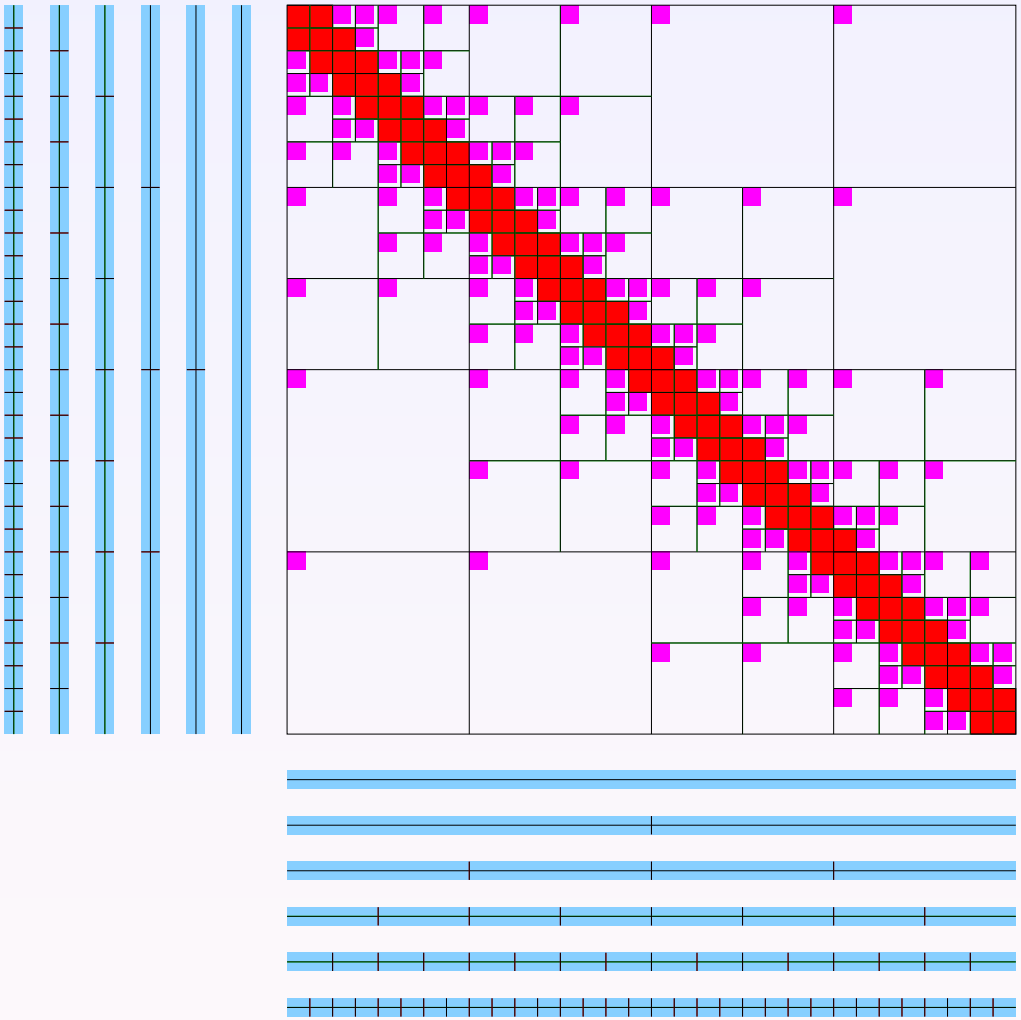
Uniform hierarchical matrix



Idea: Use three factors.

$$G|_{\hat{\tau}\times\hat{\sigma}} \approx V_\tau S_{\tau\sigma} W_\sigma^*.$$

Uniform hierarchical matrix



Idea: Use three factors.

$$G|_{\hat{\tau}\times\hat{\sigma}} \approx V_\tau S_{\tau\sigma} W_\sigma^*.$$

Result: Only $\mathcal{O}(nk)$ storage for coupling matrices $S_{\tau\sigma}$, but still $\mathcal{O}(nk \log n)$ for cluster bases V_τ and W_σ.

Nested cluster basis

Observation: Interpolation leaves polynomials unchanged.

$$\mathcal{L}_{\tau,\nu} = \sum_{\nu'=1}^{k} \mathcal{L}_{\tau,\nu}(\xi_{\tau',\nu'})\mathcal{L}_{\tau',\nu'}.$$

Result: Factorized representation of cluster basis.

$$v_{\tau,i\nu} = \int_{(i-1)/n}^{i/n} \mathcal{L}_{\tau,\nu}(x)\,dx$$

Nested cluster basis

Observation: Interpolation leaves polynomials unchanged.

$$\mathcal{L}_{\tau,\nu} = \sum_{\nu'=1}^{k} \mathcal{L}_{\tau,\nu}(\xi_{\tau',\nu'})\mathcal{L}_{\tau',\nu'}.$$

Result: Factorized representation of cluster basis.

$$v_{\tau,i\nu} = \int_{(i-1)/n}^{i/n} \sum_{\nu'=1}^{k} \mathcal{L}_{\tau,\nu}(\xi_{\tau',\nu'})\mathcal{L}_{\tau',\nu'}(x)\,dx$$

Nested cluster basis

Observation: Interpolation leaves polynomials unchanged.

$$\mathcal{L}_{\tau,\nu} = \sum_{\nu'=1}^{k} \mathcal{L}_{\tau,\nu}(\xi_{\tau',\nu'})\mathcal{L}_{\tau',\nu'}.$$

Result: Factorized representation of cluster basis.

$$v_{\tau,i\nu} = \sum_{\nu'=1}^{k} \mathcal{L}_{\tau,\nu}(\xi_{\tau',\nu'}) \int_{(i-1)/n}^{i/n} \mathcal{L}_{\tau',\nu'}(x)\,dx$$

Nested cluster basis

Observation: Interpolation leaves polynomials unchanged.

$$\mathcal{L}_{\tau,\nu} = \sum_{\nu'=1}^{k} \mathcal{L}_{\tau,\nu}(\xi_{\tau',\nu'}) \mathcal{L}_{\tau',\nu'}.$$

Result: Factorized representation of cluster basis.

$$v_{\tau,i\nu} = \sum_{\nu'=1}^{k} \underbrace{\mathcal{L}_{\tau,\nu}(\xi_{\tau',\nu'})}_{=:e_{\tau',\nu'\nu}} \underbrace{\int_{(i-1)/n}^{i/n} \mathcal{L}_{\tau',\nu'}(x)\,dx}_{=v_{\tau',i\nu'}}$$

Nested cluster basis

Observation: Interpolation leaves polynomials unchanged.

$$\mathcal{L}_{\tau,\nu} = \sum_{\nu'=1}^{k} \mathcal{L}_{\tau,\nu}(\xi_{\tau',\nu'})\mathcal{L}_{\tau',\nu'}.$$

Result: Factorized representation of cluster basis.

$$v_{\tau,i\nu} = \sum_{\nu'=1}^{k} \underbrace{\mathcal{L}_{\tau,\nu}(\xi_{\tau',\nu'})}_{=:e_{\tau',\nu'\nu}} \underbrace{\int_{(i-1)/n}^{i/n} \mathcal{L}_{\tau',\nu'}(x)\,dx}_{=v_{\tau',i\nu'}} = (V_{\tau'}E_{\tau'})_{i\nu}$$

Nested cluster basis

Observation: Interpolation leaves polynomials unchanged.

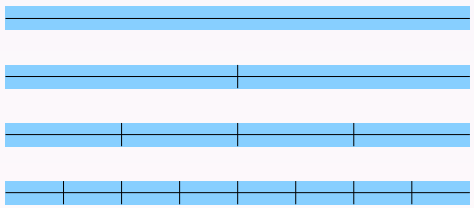
$$\mathcal{L}_{\tau,\nu} = \sum_{\nu'=1}^{k} \mathcal{L}_{\tau,\nu}(\xi_{\tau',\nu'})\mathcal{L}_{\tau',\nu'}.$$

Result: Factorized representation of cluster basis.

$$v_{\tau,i\nu} = \sum_{\nu'=1}^{k} \underbrace{\mathcal{L}_{\tau,\nu}(\xi_{\tau',\nu'})}_{=:e_{\tau',\nu'\nu}} \underbrace{\int_{(i-1)/n}^{i/n} \mathcal{L}_{\tau',\nu'}(x)\,dx}_{=v_{\tau',i\nu'}} = (V_{\tau'}E_{\tau'})_{i\nu}$$

Idea: Represent V_τ by son's matrices and small transfer matrices.

$$V_\tau = \begin{pmatrix} V_{\tau_1}E_{\tau_1} \\ V_{\tau_2}E_{\tau_2} \end{pmatrix}$$



Nested cluster basis

Observation: Interpolation leaves polynomials unchanged.

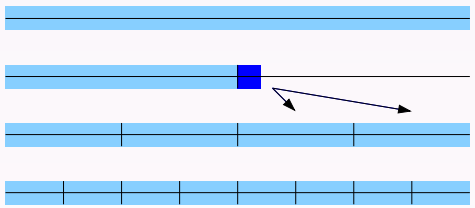
$$\mathcal{L}_{\tau,\nu} = \sum_{\nu'=1}^{k} \mathcal{L}_{\tau,\nu}(\xi_{\tau',\nu'})\mathcal{L}_{\tau',\nu'}.$$

Result: Factorized representation of cluster basis.

$$v_{\tau,i\nu} = \sum_{\nu'=1}^{k} \underbrace{\mathcal{L}_{\tau,\nu}(\xi_{\tau',\nu'})}_{=:e_{\tau',\nu'\nu}} \underbrace{\int_{(i-1)/n}^{i/n} \mathcal{L}_{\tau',\nu'}(x)\,dx}_{=v_{\tau',i\nu'}} = (V_{\tau'}E_{\tau'})_{i\nu}$$

Idea: Represent V_τ by son's matrices and small transfer matrices.

$$V_\tau = \begin{pmatrix} V_{\tau_1}E_{\tau_1} \\ V_{\tau_2}E_{\tau_2} \end{pmatrix}$$



Nested cluster basis

Observation: Interpolation leaves polynomials unchanged.

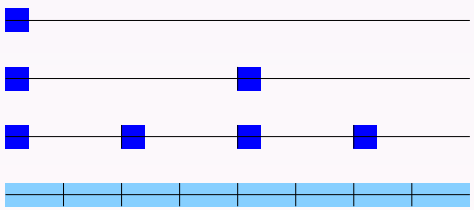
$$\mathcal{L}_{\tau,\nu} = \sum_{\nu'=1}^{k} \mathcal{L}_{\tau,\nu}(\xi_{\tau',\nu'})\mathcal{L}_{\tau',\nu'}.$$

Result: Factorized representation of cluster basis.

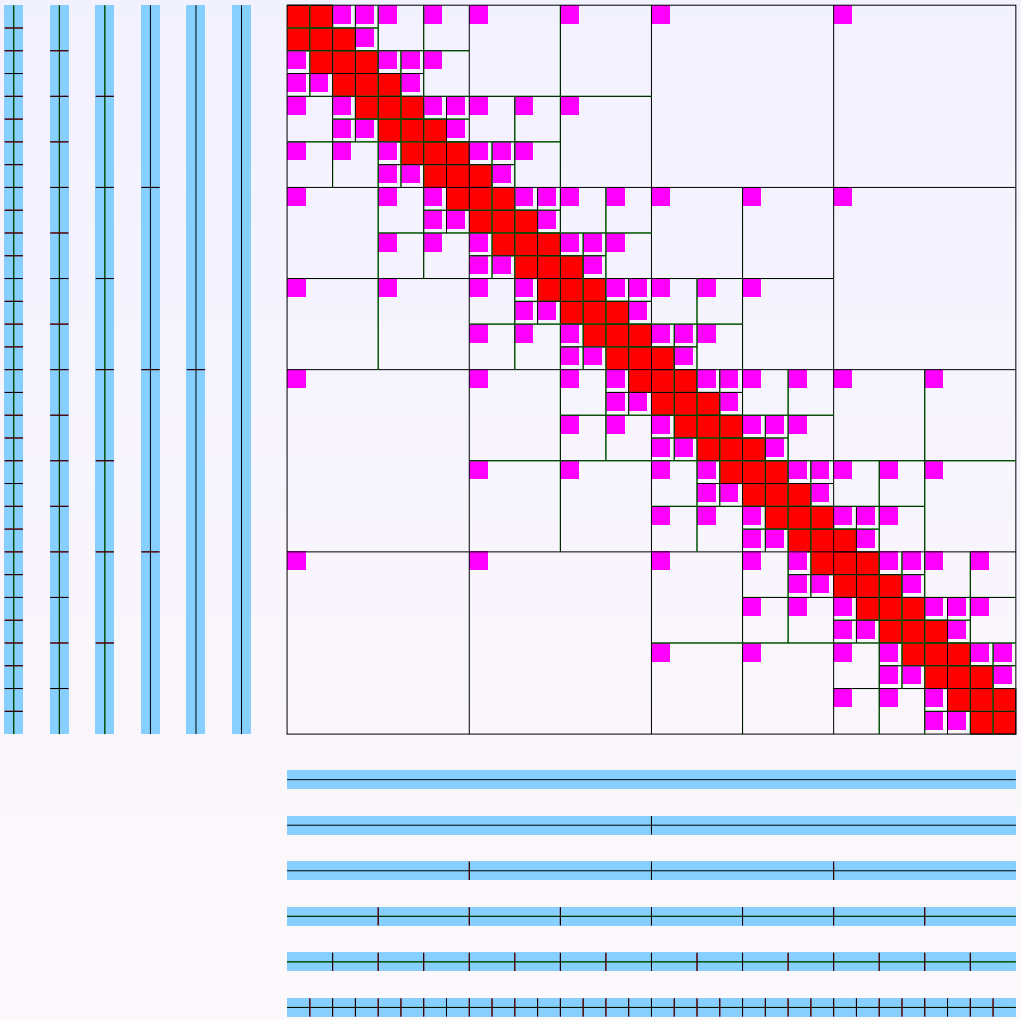
$$v_{\tau,i\nu} = \sum_{\nu'=1}^{k} \underbrace{\mathcal{L}_{\tau,\nu}(\xi_{\tau',\nu'})}_{=:e_{\tau',\nu'\nu}} \underbrace{\int_{(i-1)/n}^{i/n} \mathcal{L}_{\tau',\nu'}(x)\,dx}_{=v_{\tau',i\nu'}} = (V_{\tau'}E_{\tau'})_{i\nu}$$

Idea: Represent V_τ by son's matrices and small transfer matrices.

$$V_\tau = \begin{pmatrix} V_{\tau_1}E_{\tau_1} \\ V_{\tau_2}E_{\tau_2} \end{pmatrix}$$



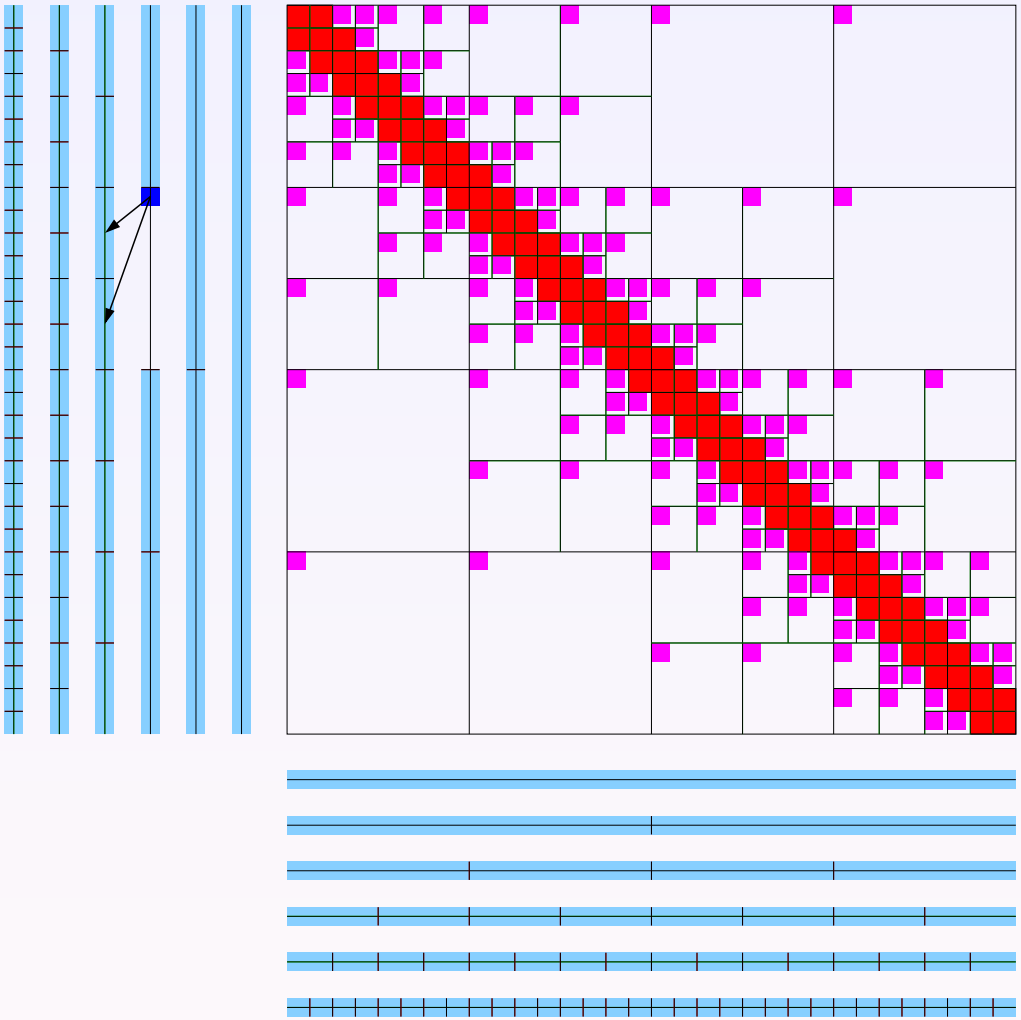
$\mathcal{H}^2$-matrix



Blocks approximated by

$$G|_{\hat{\tau}\times\hat{\sigma}} \approx V_\tau S_{\tau\sigma} W_\sigma^*$$

$\mathcal{H}^2$-matrix



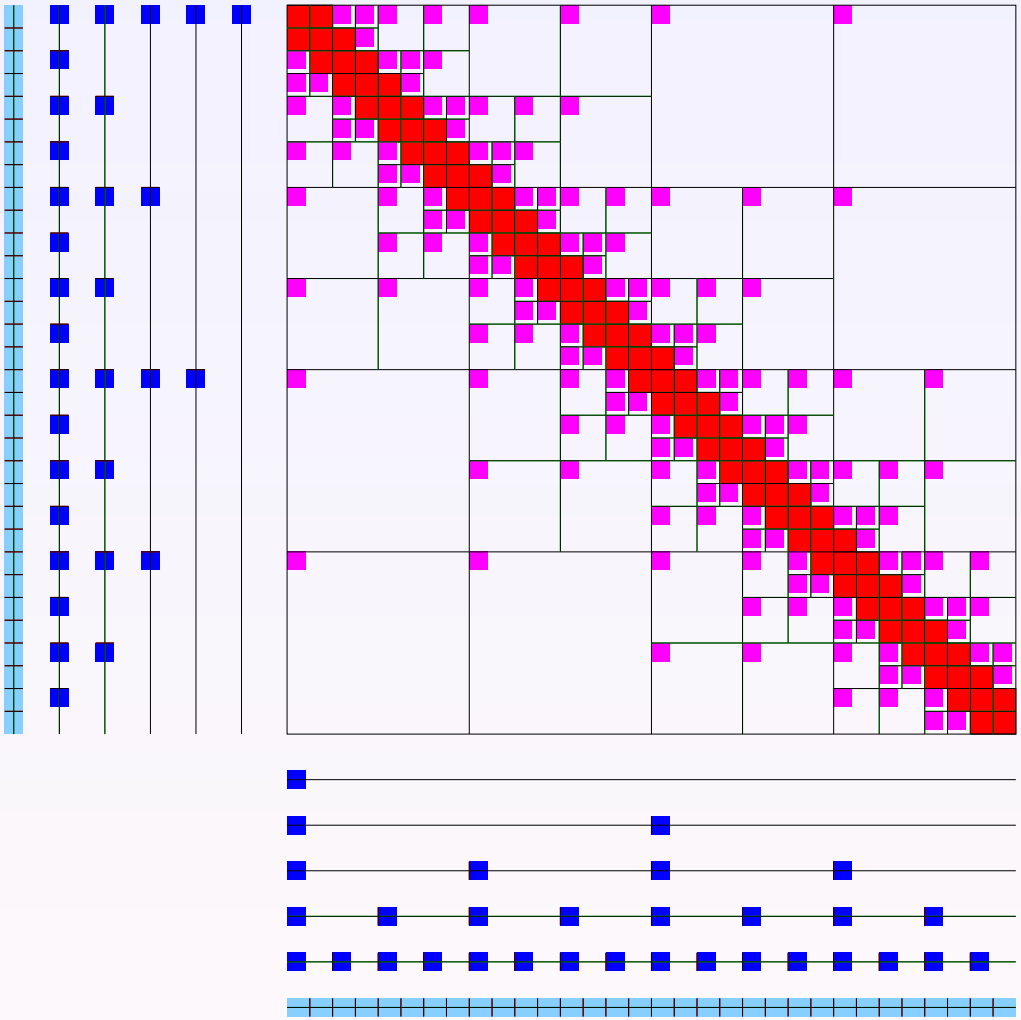
Blocks approximated by

$$G|_{\hat{\tau}\times\hat{\sigma}} \approx V_\tau S_{\tau\sigma} W_\sigma^*$$

Cluster basis nested:

$$V_\tau = \begin{pmatrix} V_{\tau_1} E_{\tau_1} \\ V_{\tau_2} E_{\tau_2} \end{pmatrix}$$

$\mathcal{H}^2$-matrix



Blocks approximated by

$$G|_{\hat{\tau}\times\hat{\sigma}} \approx V_\tau S_{\tau\sigma} W_\sigma^*$$

Cluster basis nested:

$$V_\tau = \begin{pmatrix} V_{\tau_1} E_{\tau_1} \\ V_{\tau_2} E_{\tau_2} \end{pmatrix}$$

Result: 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}
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$	$\vdots$	
524288	52.97	2.9		6.62	0.5		6.62	0.5	

References

Fast multipole: Rokhlin/Greengard 1987

Variable order: Sauter 2000, B./Löhndorf/Melenk 2005, B./Sauter 2005

Integral operators: Hackbusch/Khoromskij/Sauter 2000, B./Hackbusch 2002

Compression: B./Hackbusch 2002, B. 2005, B. 2007

Algebraic operations: B. 2006, B. 2010

Overview

Boundary integral method

Model problem: Given Dirichlet boundary values v, solve

$$\Delta u = 0, \qquad u|_{\partial\Omega} = v.$$

Representation formula: Given a fundamental solution γ, we have

$$u(x) = \int_{\partial\Omega} \gamma(x,z)\frac{\partial u}{\partial n}(z) - \frac{\partial \gamma}{\partial n_z}(x,z)u(z)\,dz \qquad \text{for all } x \in \Omega.$$

Dirichlet problem: If Dirichlet values $u|_{\partial\Omega}$ given, find Neumann values $\partial u/\partial n$ by solving

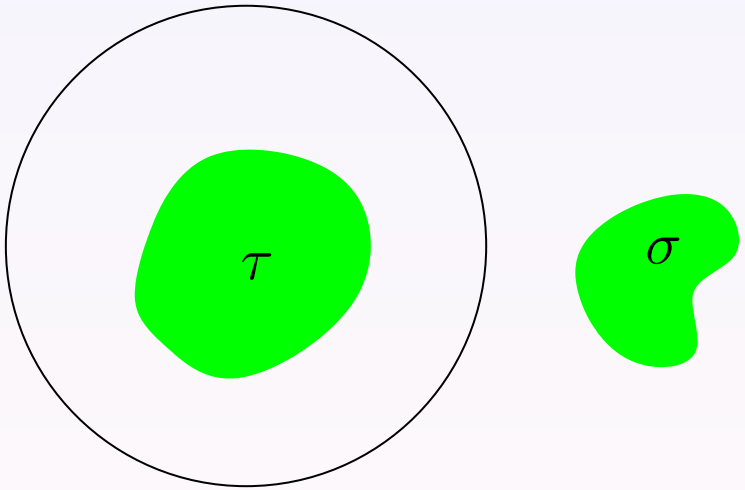
$$\frac{1}{2}u(x) + \int_{\partial\Omega} \frac{\partial \gamma}{\partial n_z}(x,z)u(z)\,dz = \int_{\partial\Omega} \gamma(x,z)\frac{\partial u}{\partial n}(z)\,dz \quad \text{for all } x \in \partial\Omega.$$

Goal: Find data-sparse approximation of these integral operators.

Quadrature approximation

Idea: Apply Green's representation formula to $u(x) = \gamma(x, y)$ and a suitable Lipschitz domain $\omega \supseteq \tau$.

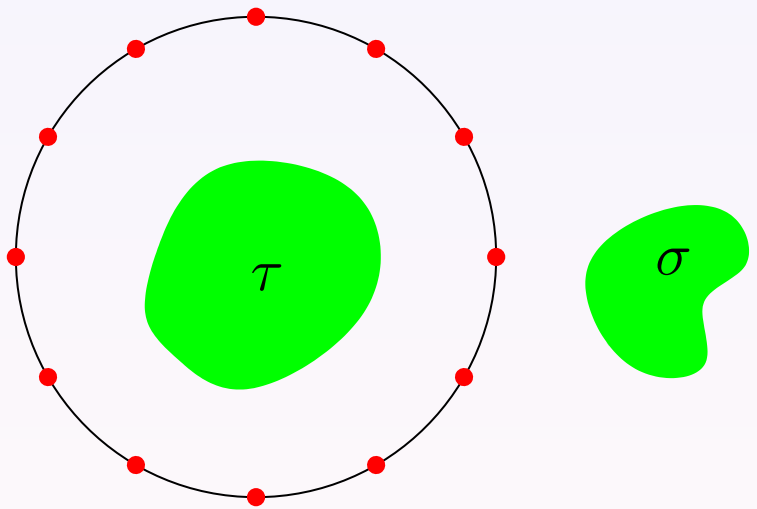
$$\gamma(x,y) = \int_{\partial\omega} \gamma(x,z)\frac{\partial\gamma}{\partial n_z}(z,y) - \frac{\partial\gamma}{\partial n_z}(x,z)\gamma(z,y)\,dz$$



Quadrature approximation

Idea: Apply Green's representation formula to $u(x) = \gamma(x,y)$ and a suitable Lipschitz domain $\omega \supseteq \tau$. Approximate integrals by quadrature to obtain

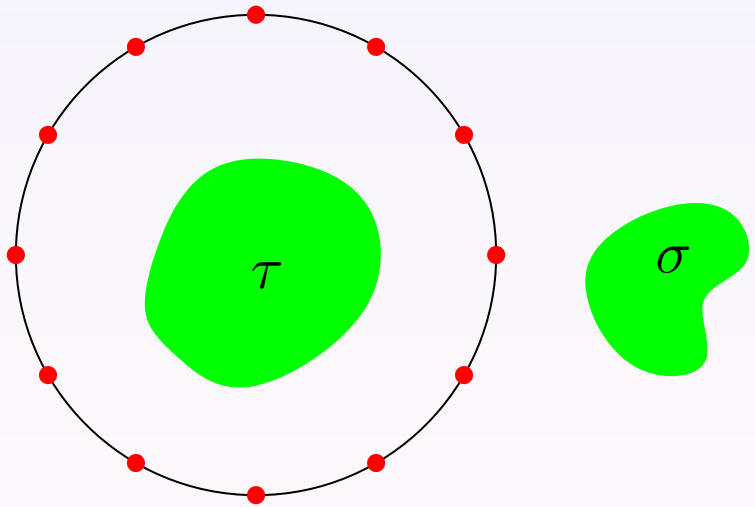
$$\gamma(x,y) \approx \sum_{\nu=1}^{k} w_\nu \gamma(x,z_\nu) \frac{\partial \gamma}{\partial n_z}(z_\nu,y) - w_\nu \frac{\partial \gamma}{\partial n_z}(x,z_\nu)\gamma(z_\nu,y)$$



Quadrature approximation

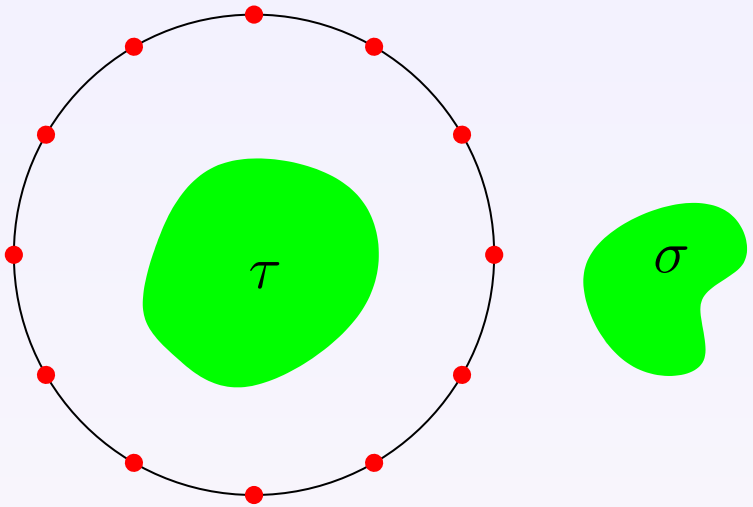
Idea: Apply Green's representation formula to $u(x) = \gamma(x, y)$ and a suitable Lipschitz domain $\omega \supseteq \tau$. Approximate integrals by quadrature to obtain

$$\gamma(x,y) \approx \sum_{\nu=1}^{k} w_\nu \gamma(x,z_\nu) \frac{\partial \gamma}{\partial n_z}(z_\nu, y) - w_\nu \frac{\partial \gamma}{\partial n_z}(x,z_\nu)\gamma(z_\nu,y)$$



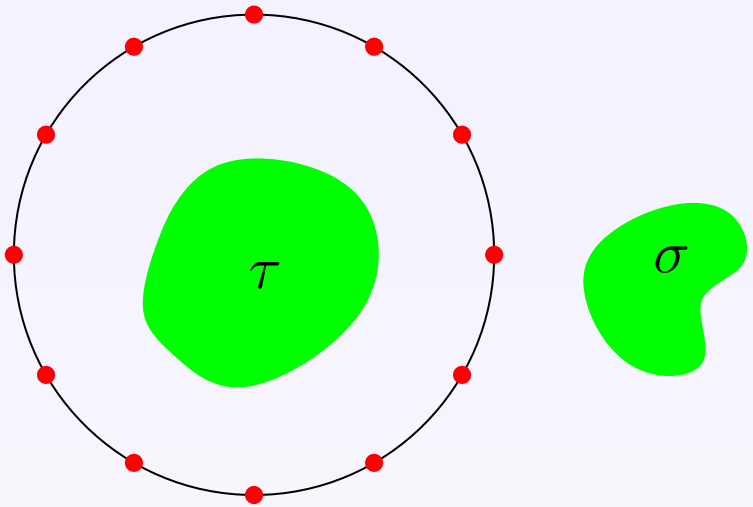
Result: Degenerate approximation with $2k$ terms.

Properties



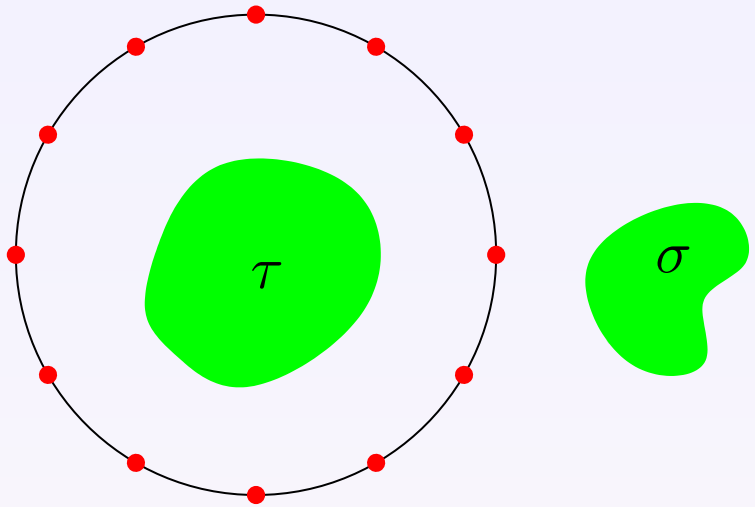
- Exponential convergence if $\partial\omega$ sufficiently far from τ and σ.

Properties



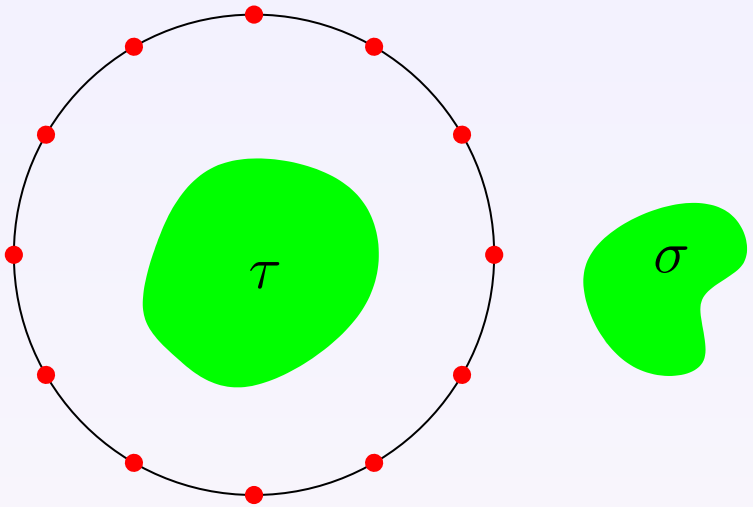
- Exponential convergence if $\partial\omega$ sufficiently far from τ and σ.
- $\mathcal{O}(m^{d-1})$ terms for m-th order approximation in d dimensions.

Properties



- Exponential convergence if $\partial\omega$ sufficiently far from τ and σ.
- $\mathcal{O}(m^{d-1})$ terms for m-th order approximation in d dimensions.
- Representation formula as in the boundary integral formulation.

Properties



- Exponential convergence if $\partial\omega$ sufficiently far from τ and σ.
- $\mathcal{O}(m^{d-1})$ terms for m-th order approximation in d dimensions.
- Representation formula as in the boundary integral formulation.
- Requires only evaluation of γ and its normal derivatives.

Hybrid method

Problem: Quadrature approximation leads to relatively high ranks, particularly for three-dimensional problems.

Idea: Combine with cross approximation algorithm.

- Quadrature method yields $G|_{\hat{\tau}\times\hat{\sigma}} \approx AB^*$.

Hybrid method

Problem: Quadrature approximation leads to relatively high ranks, particularly for three-dimensional problems.

Idea: Combine with cross approximation algorithm.

- Quadrature method yields $G|_{\hat{\tau}\times\hat{\sigma}} \approx AB^*$.
- Cross approximation yields $A \approx A|_{\hat{\tau}\times\pi_k} A|^{-1}_{\pi_\tau\times\pi_k} A|_{\pi_\tau\times k}$ with suitable index sets $\pi_\tau \subseteq \hat{\tau}$ and $\pi_k \subseteq \{1,\ldots,k\}$.

Hybrid method

Problem: Quadrature approximation leads to relatively high ranks, particularly for three-dimensional problems.

Idea: Combine with cross approximation algorithm.

- Quadrature method yields $G|_{\hat{\tau}\times\hat{\sigma}} \approx AB^*$.
- Cross approximation yields $A \approx A|_{\hat{\tau}\times\pi_k} A|^{-1}_{\pi_\tau\times\pi_k} A|_{\pi_\tau\times k}$ with suitable index sets $\pi_\tau \subseteq \hat{\tau}$ and $\pi_k \subseteq \{1,\ldots,k\}$.
- This leads to $G|_{\hat{\tau}\times\hat{\sigma}} \approx V_\tau A|_{\pi_\tau\times k} B^*$ with $V_\tau = A|_{\hat{\tau}\times\pi_k} A|^{-1}_{\pi_\tau\times\pi_k}$.

Hybrid method

Problem: Quadrature approximation leads to relatively high ranks, particularly for three-dimensional problems.

Idea: Combine with cross approximation algorithm.

- Quadrature method yields $G|_{\hat{\tau}\times\hat{\sigma}} \approx AB^*$.
- Cross approximation yields $A \approx A|_{\hat{\tau}\times\pi_k} A|^{-1}_{\pi_\tau\times\pi_k} A|_{\pi_\tau\times k}$ with suitable index sets $\pi_\tau \subseteq \hat{\tau}$ and $\pi_k \subseteq \{1,\ldots,k\}$.
- This leads to $G|_{\hat{\tau}\times\hat{\sigma}} \approx V_\tau A|_{\pi_\tau\times k} B^*$ with $V_\tau = A|_{\hat{\tau}\times\pi_k} A|^{-1}_{\pi_\tau\times\pi_k}$.
- "Reversing" the first step gives us $G|_{\hat{\tau}\times\hat{\sigma}} \approx V_\tau G|_{\pi_\tau\times\hat{\sigma}}$.

Hybrid method

Problem: Quadrature approximation leads to relatively high ranks, particularly for three-dimensional problems.

Idea: Combine with cross approximation algorithm.

- Quadrature method yields $G|_{\hat{\tau}\times\hat{\sigma}} \approx AB^*$.
- Cross approximation yields $A \approx A|_{\hat{\tau}\times\pi_k} A|^{-1}_{\pi_\tau\times\pi_k} A|_{\pi_\tau\times k}$ with suitable index sets $\pi_\tau \subseteq \hat{\tau}$ and $\pi_k \subseteq \{1,\ldots,k\}$.
- This leads to $G|_{\hat{\tau}\times\hat{\sigma}} \approx V_\tau A|_{\pi_\tau\times k} B^*$ with $V_\tau = A|_{\hat{\tau}\times\pi_k} A|^{-1}_{\pi_\tau\times\pi_k}$.
- "Reversing" the first step gives us $G|_{\hat{\tau}\times\hat{\sigma}} \approx V_\tau G|_{\pi_\tau\times\hat{\sigma}}$.

Result: "Algebraic interpolation" with $\#\pi_\tau \leq 2k$ interpolation points.

- V_τ depends only on τ, but not on σ. $\rightarrow$ Efficient setup.
- Only a few entries of G required for each block.
- Experiments indicate $\#\pi_\tau \ll k$.

Symmetric approximation

Row interpolation: We can find $\pi_\tau \subseteq \hat{\tau}$ and $V_\tau \in \mathbb{R}^{\hat{\tau} \times \pi_\tau}$ such that

$$G|_{\hat{\tau} \times \hat{\sigma}} \approx V_\tau G|_{\pi_\tau \times \hat{\sigma}}.$$

Symmetric approximation

Row interpolation: We can find $\pi_\tau \subseteq \hat{\tau}$ and $V_\tau \in \mathbb{R}^{\hat{\tau} \times \pi_\tau}$ such that

$$G|_{\hat{\tau} \times \hat{\sigma}} \approx V_\tau G|_{\pi_\tau \times \hat{\sigma}}.$$

Column interpolation: Applying the same arguments to columns instead of rows yields $\pi_\sigma \subseteq \hat{\sigma}$ and $W_\sigma \in \mathbb{R}^{\hat{\sigma} \times \pi_\sigma}$ such that

$$G|_{\hat{\tau} \times \hat{\sigma}} \approx G|_{\hat{\tau} \times \pi_\sigma} W_\sigma^*.$$

Symmetric approximation

Row interpolation: We can find $\pi_\tau \subseteq \hat{\tau}$ and $V_\tau \in \mathbb{R}^{\hat{\tau} \times \pi_\tau}$ such that

$$G|_{\hat{\tau} \times \hat{\sigma}} \approx V_\tau G|_{\pi_\tau \times \hat{\sigma}}.$$

Column interpolation: Applying the same arguments to columns instead of rows yields $\pi_\sigma \subseteq \hat{\sigma}$ and $W_\sigma \in \mathbb{R}^{\hat{\sigma} \times \pi_\sigma}$ such that

$$G|_{\hat{\tau} \times \hat{\sigma}} \approx G|_{\hat{\tau} \times \pi_\sigma} W_\sigma^*.$$

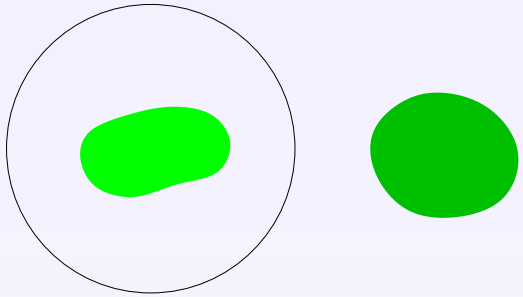
Symmetric interpolation combines both to obtain

$$G|_{\hat{\tau} \times \hat{\sigma}} \approx V_\tau G|_{\pi_\tau \times \hat{\sigma}} \approx V_\tau G|_{\pi_\tau \times \pi_\sigma} W_\sigma^*.$$

Advantage: Only very small number of coefficients required per block.

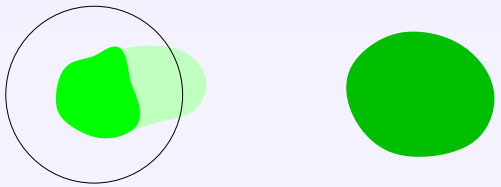
Nested cluster bases

Observation: Approximation scheme not only applicable to τ, but also to subdomains τ_1, τ_2.



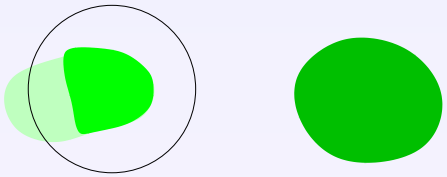
Nested cluster bases

Observation: Approximation scheme not only applicable to τ, but also to subdomains τ_1, τ_2.



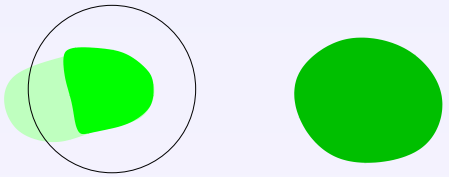
Nested cluster bases

Observation: Approximation scheme not only applicable to τ, but also to subdomains τ_1, τ_2.



Nested cluster bases

Observation: Approximation scheme not only applicable to τ, but also to subdomains τ_1, τ_2.



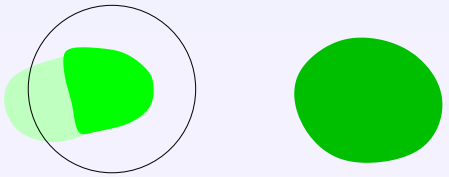
Idea: Construct matrices V_{τ_1}, V_{τ_2} and pivot indices π_{τ_1}, π_{τ_2} for subdomains τ_1, τ_2 first, then use these informations to speed up construction of V_τ and π_τ:

$$G|_{\hat{\tau}\times\hat{\sigma}} = \begin{pmatrix} G|_{\hat{\tau}_1\times\hat{\sigma}} \\ G|_{\hat{\tau}_2\times\hat{\sigma}} \end{pmatrix} \approx \begin{pmatrix} V_{\tau_1} G|_{\pi_{\tau_1}\times\hat{\sigma}} \\ V_{\tau_2} G|_{\pi_{\tau_2}\times\hat{\sigma}} \end{pmatrix} = \begin{pmatrix} V_{\tau_1} & \\ & V_{\tau_2} \end{pmatrix} \begin{pmatrix} G|_{\pi_{\tau_1}\times\hat{\sigma}} \\ G|_{\pi_{\tau_2}\times\hat{\sigma}} \end{pmatrix}$$

$$= \begin{pmatrix} V_{\tau_1} & \\ & V_{\tau_2} \end{pmatrix} G|_{(\pi_{\tau_1}\cup\pi_{\tau_2})\times\hat{\sigma}} \approx \underbrace{\begin{pmatrix} V_{\tau_1} & \\ & V_{\tau_2} \end{pmatrix} \widehat{V}_\tau}_{=:V_\tau} G|_{\pi_\tau\times\hat{\sigma}}.$$

Nested cluster bases

Observation: Approximation scheme not only applicable to τ, but also to subdomains τ_1, τ_2.



Idea: Construct matrices V_{τ_1}, V_{τ_2} and pivot indices π_{τ_1}, π_{τ_2} for subdomains τ_1, τ_2 first, then use these informations to speed up construction of V_τ and π_τ:

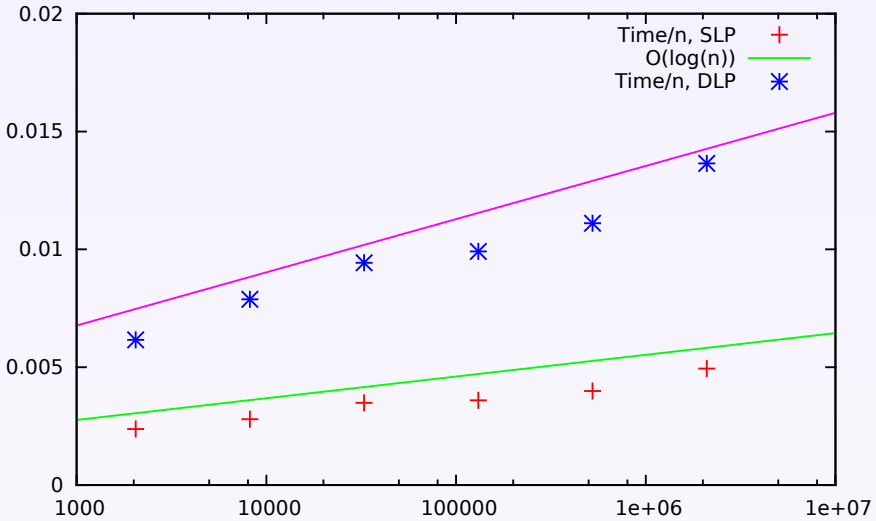
$$G|_{\hat{\tau}\times\hat{\sigma}} = \begin{pmatrix} G|_{\hat{\tau}_1\times\hat{\sigma}} \\ G|_{\hat{\tau}_2\times\hat{\sigma}} \end{pmatrix} \approx \begin{pmatrix} V_{\tau_1} G|_{\pi_{\tau_1}\times\hat{\sigma}} \\ V_{\tau_2} G|_{\pi_{\tau_2}\times\hat{\sigma}} \end{pmatrix} = \begin{pmatrix} V_{\tau_1} & \\ & V_{\tau_2} \end{pmatrix} \begin{pmatrix} G|_{\pi_{\tau_1}\times\hat{\sigma}} \\ G|_{\pi_{\tau_2}\times\hat{\sigma}} \end{pmatrix}$$

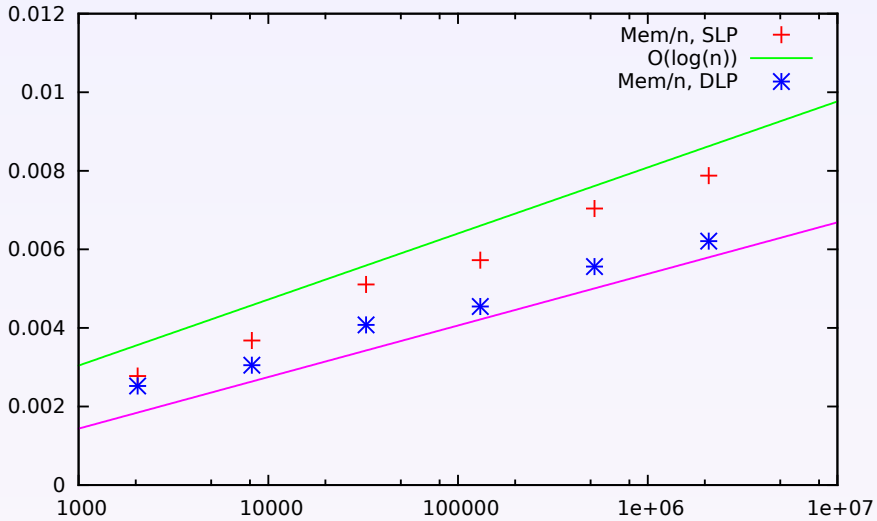
$$= \begin{pmatrix} V_{\tau_1} & \\ & V_{\tau_2} \end{pmatrix} G|_{(\pi_{\tau_1}\cup\pi_{\tau_2})\times\hat{\sigma}} \approx \underbrace{\begin{pmatrix} V_{\tau_1} & \\ & V_{\tau_2} \end{pmatrix} \widehat{V}_\tau}_{=:V_\tau} G|_{\pi_\tau\times\hat{\sigma}}.$$

Result: Nested cluster basis, constructed in $\sim nk^2$ operations.

Experiment: Hybrid Green approximation

Goal: Approximate single layer potential matrix, pw. constant basis.





Results:

- Tolerances chosen to ensure optimal $\mathcal{O}(h)$ convergence.
- Setup in $\sim n \log n$ operations.
- Storage requirements $\sim n \log n$.

Overview

LR factorization

Goal: Given an $\mathcal{H}^2$-matrix G, compute its LR factorization $G = LR$.

Approach: If G is inadmissible, compute $G = LR$ directly.
Otherwise, use submatrices

$$\begin{pmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ & R_{22} \end{pmatrix}$$

LR factorization

Goal: Given an $\mathcal{H}^2$-matrix G, compute its LR factorization $G = LR$.

Approach: If G is inadmissible, compute $G = LR$ directly.
Otherwise, use submatrices

$$\begin{pmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ & R_{22} \end{pmatrix}$$

This is equivalent to

$$G_{11} = L_{11} R_{11},$$
$$G_{12} = L_{11} R_{12}, \qquad G_{21} = L_{21} R_{11},$$
$$G_{22} - L_{21} R_{12} = L_{22} R_{22}.$$

If we can compute $Z \leftarrow Z + \alpha XY$ efficiently, we can use recursion to perform matrix forward substitution and find the LR factorization.

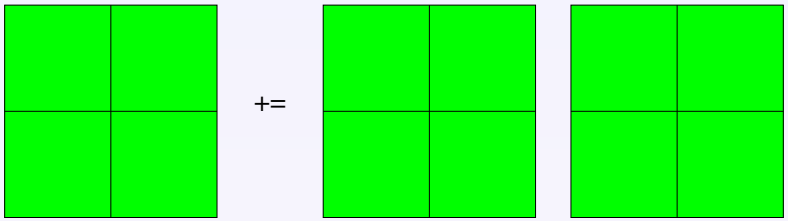
Multiplication

Goal: Perform update $Z|_{\hat{\tau}\times\hat{\varrho}} \leftarrow Z|_{\hat{\tau}\times\hat{\varrho}} + \alpha X|_{\hat{\tau}\times\hat{\sigma}} Y|_{\hat{\sigma}\times\hat{\varrho}}$ efficiently.

Idea: If $\tau \times \sigma$ and $\sigma \times \varrho$ are subdivided, treat them by recursion.

Multiplication

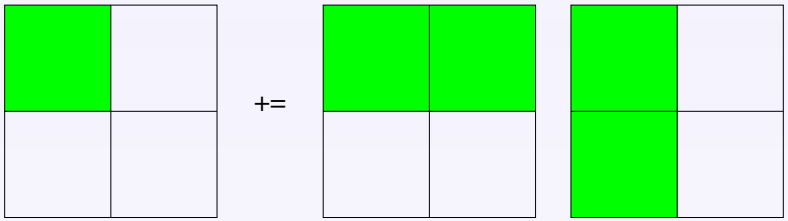
Goal: Perform update $Z|_{\hat{\tau}\times\hat{\varrho}} \leftarrow Z|_{\hat{\tau}\times\hat{\varrho}} + \alpha X|_{\hat{\tau}\times\hat{\sigma}} Y|_{\hat{\sigma}\times\hat{\varrho}}$ efficiently.



Idea: If $\tau \times \sigma$ and $\sigma \times \varrho$ are subdivided, treat them by recursion.

Multiplication

Goal: Perform update $Z|_{\hat{\tau}\times\hat{\varrho}} \leftarrow Z|_{\hat{\tau}\times\hat{\varrho}} + \alpha X|_{\hat{\tau}\times\hat{\sigma}} Y|_{\hat{\sigma}\times\hat{\varrho}}$ efficiently.

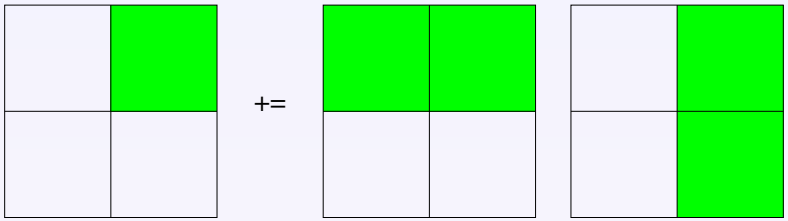


Idea: If $\tau \times \sigma$ and $\sigma \times \varrho$ are subdivided, treat them by recursion.

- $Z|_{\hat{\tau}_1\times\hat{\varrho}_1} \leftarrow Z|_{\hat{\tau}_1\times\hat{\varrho}_1} + \alpha X|_{\hat{\tau}_1\times\hat{\sigma}_1} Y|_{\hat{\sigma}_1\times\hat{\varrho}_1} + \alpha X|_{\hat{\tau}_1\times\hat{\sigma}_2} Y|_{\hat{\sigma}_2\times\hat{\varrho}_1}$

Multiplication

Goal: Perform update $Z|_{\hat{\tau}\times\hat{\varrho}} \leftarrow Z|_{\hat{\tau}\times\hat{\varrho}} + \alpha X|_{\hat{\tau}\times\hat{\sigma}} Y|_{\hat{\sigma}\times\hat{\varrho}}$ efficiently.

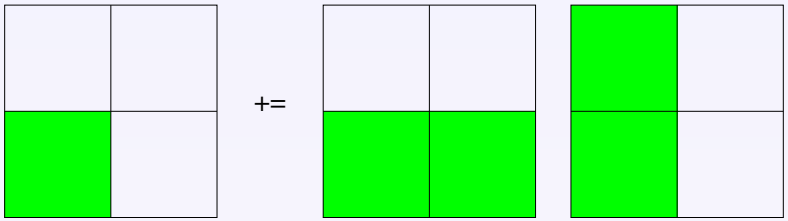


Idea: If $\tau \times \sigma$ and $\sigma \times \varrho$ are subdivided, treat them by recursion.

- $Z|_{\hat{\tau}_1\times\hat{\varrho}_1} \leftarrow Z|_{\hat{\tau}_1\times\hat{\varrho}_1} + \alpha X|_{\hat{\tau}_1\times\hat{\sigma}_1} Y|_{\hat{\sigma}_1\times\hat{\varrho}_1} + \alpha X|_{\hat{\tau}_1\times\hat{\sigma}_2} Y|_{\hat{\sigma}_2\times\hat{\varrho}_1}$
- $Z|_{\hat{\tau}_1\times\hat{\varrho}_2} \leftarrow Z|_{\hat{\tau}_1\times\hat{\varrho}_2} + \alpha X|_{\hat{\tau}_1\times\hat{\sigma}_1} Y|_{\hat{\sigma}_1\times\hat{\varrho}_2} + \alpha X|_{\hat{\tau}_1\times\hat{\sigma}_2} Y|_{\hat{\sigma}_2\times\hat{\varrho}_2}$

Multiplication

Goal: Perform update $Z|_{\hat{\tau}\times\hat{\varrho}} \leftarrow Z|_{\hat{\tau}\times\hat{\varrho}} + \alpha X|_{\hat{\tau}\times\hat{\sigma}} Y|_{\hat{\sigma}\times\hat{\varrho}}$ efficiently.

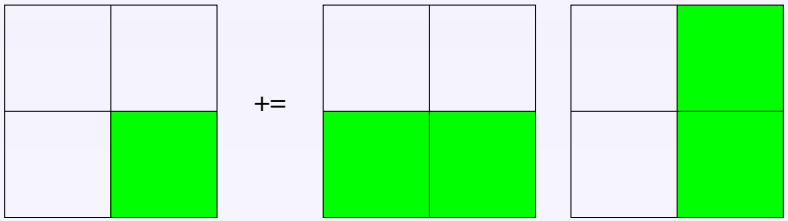


Idea: If $\tau \times \sigma$ and $\sigma \times \varrho$ are subdivided, treat them by recursion.

- $Z|_{\hat{\tau}_1\times\hat{\varrho}_1} \leftarrow Z|_{\hat{\tau}_1\times\hat{\varrho}_1} + \alpha X|_{\hat{\tau}_1\times\hat{\sigma}_1} Y|_{\hat{\sigma}_1\times\hat{\varrho}_1} + \alpha X|_{\hat{\tau}_1\times\hat{\sigma}_2} Y|_{\hat{\sigma}_2\times\hat{\varrho}_1}$
- $Z|_{\hat{\tau}_1\times\hat{\varrho}_2} \leftarrow Z|_{\hat{\tau}_1\times\hat{\varrho}_2} + \alpha X|_{\hat{\tau}_1\times\hat{\sigma}_1} Y|_{\hat{\sigma}_1\times\hat{\varrho}_2} + \alpha X|_{\hat{\tau}_1\times\hat{\sigma}_2} Y|_{\hat{\sigma}_2\times\hat{\varrho}_2}$
- $Z|_{\hat{\tau}_2\times\hat{\varrho}_1} \leftarrow Z|_{\hat{\tau}_2\times\hat{\varrho}_1} + \alpha X|_{\hat{\tau}_2\times\hat{\sigma}_1} Y|_{\hat{\sigma}_1\times\hat{\varrho}_1} + \alpha X|_{\hat{\tau}_2\times\hat{\sigma}_2} Y|_{\hat{\sigma}_2\times\hat{\varrho}_1}$

Multiplication

Goal: Perform update $Z|_{\hat{\tau}\times\hat{\varrho}} \leftarrow Z|_{\hat{\tau}\times\hat{\varrho}} + \alpha X|_{\hat{\tau}\times\hat{\sigma}} Y|_{\hat{\sigma}\times\hat{\varrho}}$ efficiently.

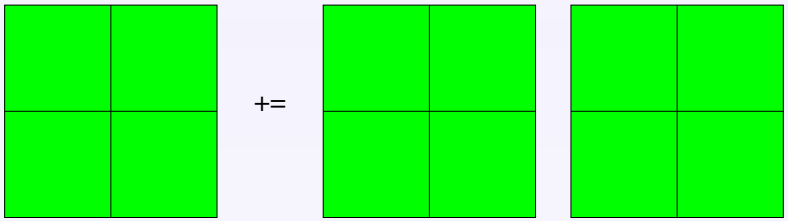


Idea: If $\tau \times \sigma$ and $\sigma \times \varrho$ are subdivided, treat them by recursion.

- $Z|_{\hat{\tau}_1\times\hat{\varrho}_1} \leftarrow Z|_{\hat{\tau}_1\times\hat{\varrho}_1} + \alpha X|_{\hat{\tau}_1\times\hat{\sigma}_1} Y|_{\hat{\sigma}_1\times\hat{\varrho}_1} + \alpha X|_{\hat{\tau}_1\times\hat{\sigma}_2} Y|_{\hat{\sigma}_2\times\hat{\varrho}_1}$
- $Z|_{\hat{\tau}_1\times\hat{\varrho}_2} \leftarrow Z|_{\hat{\tau}_1\times\hat{\varrho}_2} + \alpha X|_{\hat{\tau}_1\times\hat{\sigma}_1} Y|_{\hat{\sigma}_1\times\hat{\varrho}_2} + \alpha X|_{\hat{\tau}_1\times\hat{\sigma}_2} Y|_{\hat{\sigma}_2\times\hat{\varrho}_2}$
- $Z|_{\hat{\tau}_2\times\hat{\varrho}_1} \leftarrow Z|_{\hat{\tau}_2\times\hat{\varrho}_1} + \alpha X|_{\hat{\tau}_2\times\hat{\sigma}_1} Y|_{\hat{\sigma}_1\times\hat{\varrho}_1} + \alpha X|_{\hat{\tau}_2\times\hat{\sigma}_2} Y|_{\hat{\sigma}_2\times\hat{\varrho}_1}$
- $Z|_{\hat{\tau}_2\times\hat{\varrho}_2} \leftarrow Z|_{\hat{\tau}_2\times\hat{\varrho}_2} + \alpha X|_{\hat{\tau}_2\times\hat{\sigma}_1} Y|_{\hat{\sigma}_1\times\hat{\varrho}_2} + \alpha X|_{\hat{\tau}_2\times\hat{\sigma}_2} Y|_{\hat{\sigma}_2\times\hat{\varrho}_2}$

Multiplication

Goal: Perform update $Z|_{\hat{\tau}\times\hat{\varrho}} \leftarrow Z|_{\hat{\tau}\times\hat{\varrho}} + \alpha X|_{\hat{\tau}\times\hat{\sigma}} Y|_{\hat{\sigma}\times\hat{\varrho}}$ efficiently.



Idea: If $\tau \times \sigma$ and $\sigma \times \varrho$ are subdivided, treat them by recursion.

- $Z|_{\hat{\tau}_1\times\hat{\varrho}_1} \leftarrow Z|_{\hat{\tau}_1\times\hat{\varrho}_1} + \alpha X|_{\hat{\tau}_1\times\hat{\sigma}_1} Y|_{\hat{\sigma}_1\times\hat{\varrho}_1} + \alpha X|_{\hat{\tau}_1\times\hat{\sigma}_2} Y|_{\hat{\sigma}_2\times\hat{\varrho}_1}$
- $Z|_{\hat{\tau}_1\times\hat{\varrho}_2} \leftarrow Z|_{\hat{\tau}_1\times\hat{\varrho}_2} + \alpha X|_{\hat{\tau}_1\times\hat{\sigma}_1} Y|_{\hat{\sigma}_1\times\hat{\varrho}_2} + \alpha X|_{\hat{\tau}_1\times\hat{\sigma}_2} Y|_{\hat{\sigma}_2\times\hat{\varrho}_2}$
- $Z|_{\hat{\tau}_2\times\hat{\varrho}_1} \leftarrow Z|_{\hat{\tau}_2\times\hat{\varrho}_1} + \alpha X|_{\hat{\tau}_2\times\hat{\sigma}_1} Y|_{\hat{\sigma}_1\times\hat{\varrho}_1} + \alpha X|_{\hat{\tau}_2\times\hat{\sigma}_2} Y|_{\hat{\sigma}_2\times\hat{\varrho}_1}$
- $Z|_{\hat{\tau}_2\times\hat{\varrho}_2} \leftarrow Z|_{\hat{\tau}_2\times\hat{\varrho}_2} + \alpha X|_{\hat{\tau}_2\times\hat{\sigma}_1} Y|_{\hat{\sigma}_1\times\hat{\varrho}_2} + \alpha X|_{\hat{\tau}_2\times\hat{\sigma}_2} Y|_{\hat{\sigma}_2\times\hat{\varrho}_2}$

Multiplication with low-rank block

Important case: What happens if one of the factors is not subdivided?

Multiplication with low-rank block

Important case: What happens if one of the factors is not subdivided?

Example: Let $\sigma \times \varrho$ be an admissible block of Y.

$$X|_{\hat{\tau}\times\hat{\sigma}} Y|_{\hat{\sigma}\times\hat{\varrho}} = X|_{\hat{\tau}\times\hat{\sigma}} (V_\sigma S_{\sigma\varrho} W_\varrho^*)$$

Multiplication with low-rank block

Important case: What happens if one of the factors is not subdivided?

Example: Let $\sigma \times \varrho$ be an admissible block of Y.

$$X|_{\hat{\tau}\times\hat{\sigma}} Y|_{\hat{\sigma}\times\hat{\varrho}} = X|_{\hat{\tau}\times\hat{\sigma}}(V_\sigma S_{\sigma\varrho} W_\varrho^*) = (X|_{\hat{\tau}\times\hat{\sigma}} V_\sigma S_{\sigma\varrho}) W_\varrho^*$$

Multiplication with low-rank block

Important case: What happens if one of the factors is not subdivided?

Example: Let $\sigma \times \varrho$ be an admissible block of Y.

$$X|_{\hat{\tau}\times\hat{\sigma}} Y|_{\hat{\sigma}\times\hat{\varrho}} = X|_{\hat{\tau}\times\hat{\sigma}}(V_\sigma S_{\sigma\varrho} W_\varrho^*) = (X|_{\hat{\tau}\times\hat{\sigma}} V_\sigma S_{\sigma\varrho}) W_\varrho^* = A_{\tau\varrho} W_\varrho^*.$$

Multiplication with low-rank block

Important case: What happens if one of the factors is not subdivided?

Example: Let $\sigma \times \varrho$ be an admissible block of Y.

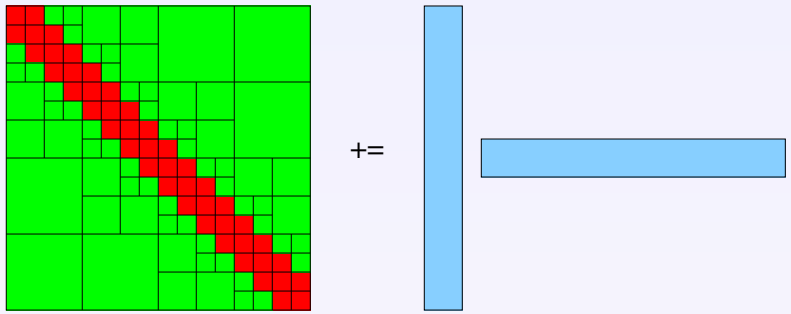
$$X|_{\hat{\tau}\times\hat{\sigma}} Y|_{\hat{\sigma}\times\hat{\varrho}} = X|_{\hat{\tau}\times\hat{\sigma}}(V_\sigma S_{\sigma\varrho} W_\varrho^*) = (X|_{\hat{\tau}\times\hat{\sigma}} V_\sigma S_{\sigma\varrho}) W_\varrho^* = A_{\tau\varrho} W_\varrho^*.$$

Idea: $A_{\tau\varrho}$ can be computed by fast matrix-vector multiplications.
We "only" need an efficient algorithm for low-rank updates

$$Z|_{\hat{\tau}\times\hat{\varrho}} \leftarrow Z|_{\hat{\tau}\times\hat{\varrho}} + \alpha A_{\tau\varrho} W_\varrho^*.$$

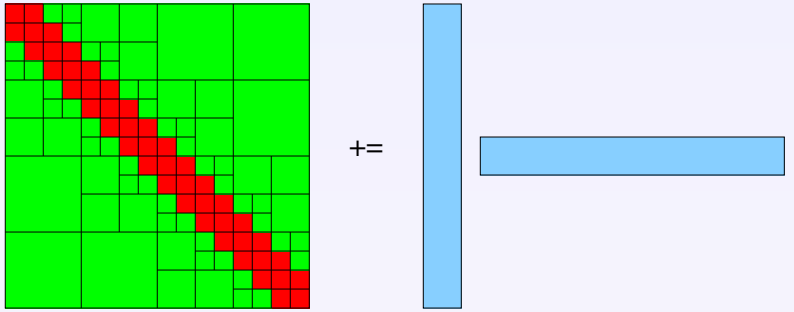
Low-rank update

Goal: Approximate $Z + AB^*$.



Low-rank update

Goal: Approximate $Z + AB^*$.

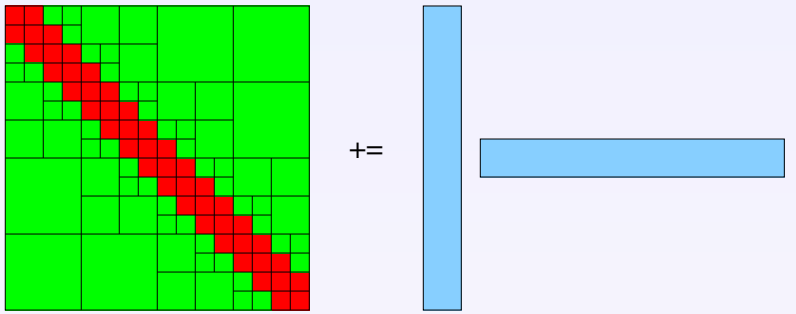


Idea: $Z + AB^*$ already is an $\mathcal{H}^2$-matrix, since for admissible $\tau \times \sigma$

$$
\begin{aligned}
(Z + AB^*)|_{\hat{\tau}\times\hat{\sigma}} &= V_\tau S_{\tau\sigma} W_\sigma^* + A|_{\hat{\tau}\times k} B|_{\hat{\sigma}\times k}^* \\
&= \underbrace{\begin{pmatrix} V_\tau & A|_{\hat{\tau}\times k} \end{pmatrix}}_{=:V_{\mathrm{new},\tau}} \underbrace{\begin{pmatrix} S_{\tau\sigma} & \\ & I \end{pmatrix}}_{=:S_{\mathrm{new},\tau\sigma}} \underbrace{\begin{pmatrix} W_\sigma & B|_{\hat{\sigma}\times k} \end{pmatrix}^*}_{=:W_{\mathrm{new},\sigma}^*}
\end{aligned}
$$

Low-rank update

Goal: Approximate $Z + AB^*$.



Idea: $Z + AB^*$ already is an $\mathcal{H}^2$-matrix, since for admissible $\tau \times \sigma$

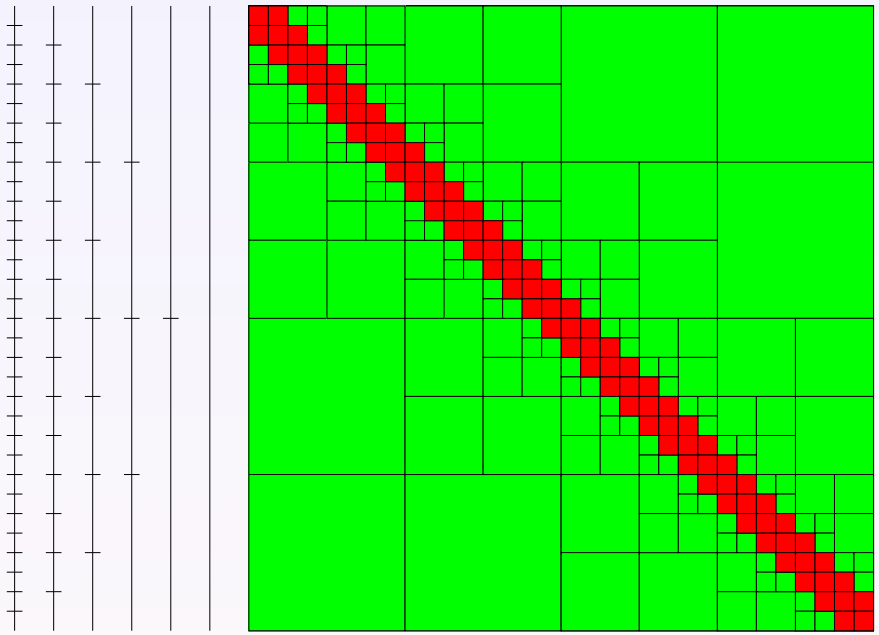
$$\begin{aligned}(Z + AB^*)|_{\hat{\tau}\times\hat{\sigma}} &= V_\tau S_{\tau\sigma} W_\sigma^* + A|_{\hat{\tau}\times k} B|_{\hat{\sigma}\times k}^* \\ &= \underbrace{\begin{pmatrix} V_\tau & A|_{\hat{\tau}\times k} \end{pmatrix}}_{=:V_{\mathrm{new},\tau}} \underbrace{\begin{pmatrix} S_{\tau\sigma} & \\ & I \end{pmatrix}}_{=:S_{\mathrm{new},\tau\sigma}} \underbrace{\begin{pmatrix} W_\sigma & B|_{\hat{\sigma}\times k} \end{pmatrix}^*}_{=:W_{\mathrm{new},\sigma}^*}\end{aligned}$$

Challenge: Each update increases storage requirements, although actual numerical rank may be low. $\rightarrow$ Recompression required.

Rank reduction for $\mathcal{H}^2$-matrices

Goal: Given an $\mathcal{H}^2$-matrix, reduce its rank.

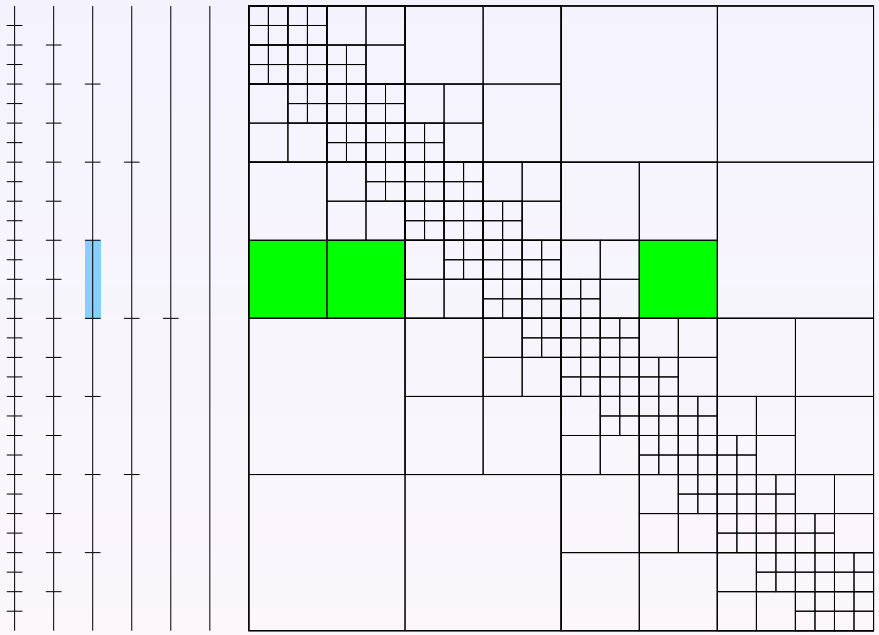
Hierarchical matrices: Compute SVD and truncate.



Rank reduction for $\mathcal{H}^2$-matrices

Goal: Given an $\mathcal{H}^2$-matrix, reduce its rank.

Hierarchical matrices: Compute SVD and truncate.

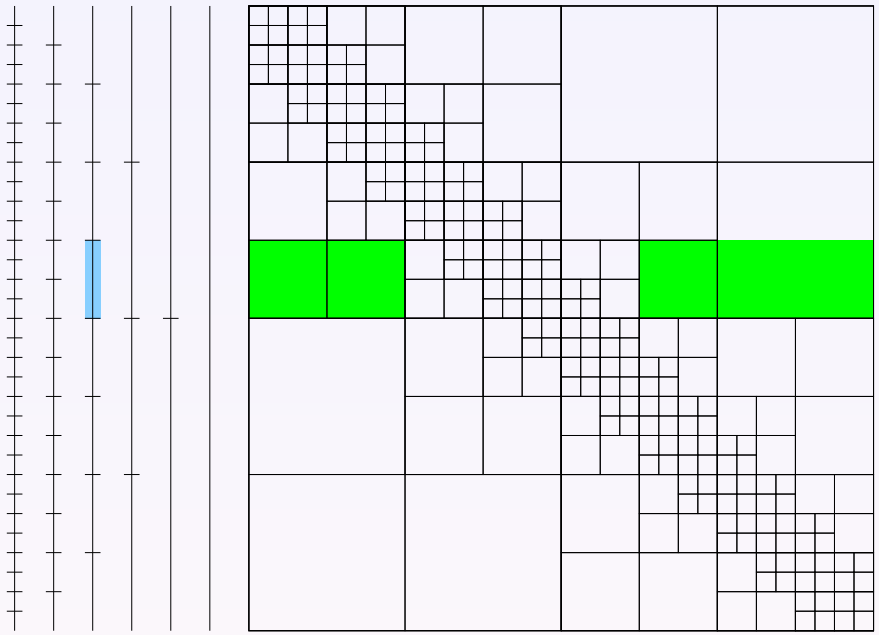


Clusterbasis V_τ should approximate all admissible blocks $\tau \times \sigma$.

Rank reduction for $\mathcal{H}^2$-matrices

Goal: Given an $\mathcal{H}^2$-matrix, reduce its rank.

Hierarchical matrices: Compute SVD and truncate.

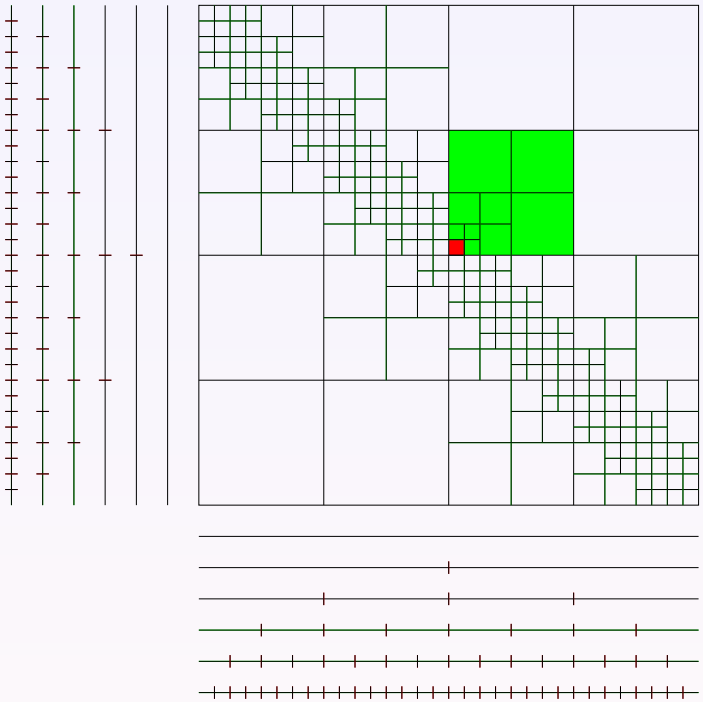


Clusterbasis V_τ should approximate all admissible blocks $\tau \times \sigma$.

Nested basis: Additionally admissible blocks $\tau^+ \times \sigma$ with $\tau \subseteq \tau^+$.

Local low-rank update

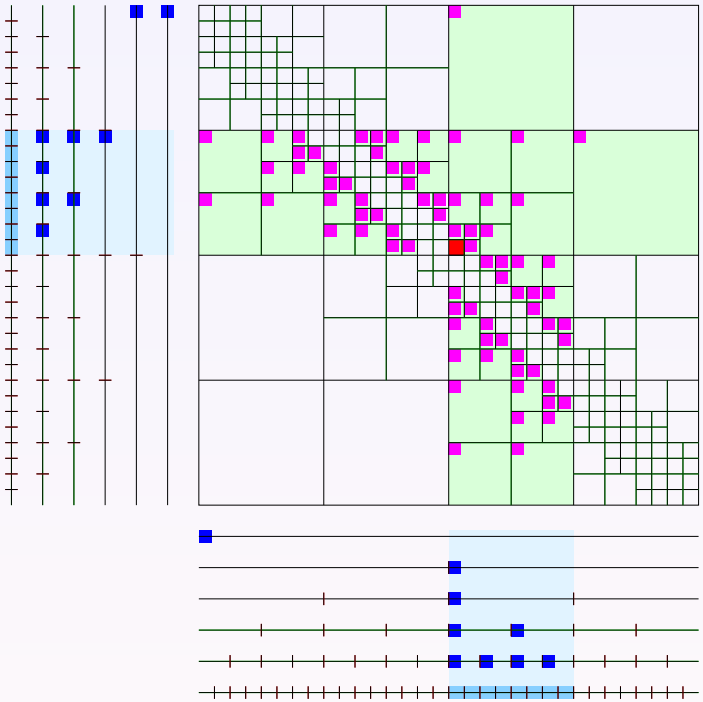
Goal: Local update $Z|_{\hat{\tau}\times\hat{\sigma}} \leftarrow Z|_{\hat{\tau}\times\hat{\sigma}} + AB^*$.



Local low-rank update

Goal: Local update $Z|_{\hat{\tau}\times\hat{\varrho}} \leftarrow Z|_{\hat{\tau}\times\hat{\varrho}} + AB^*$.

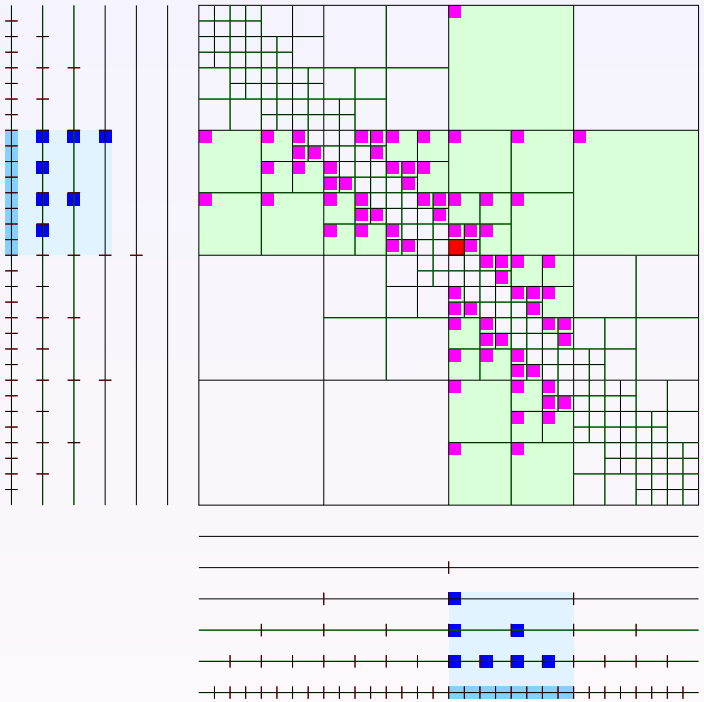
Problem: Changing the cluster bases influences all blocks intersecting rows in $\hat{\tau}$ or columns in $\hat{\sigma}$.



Local low-rank update

Goal: Local update $Z|_{\hat{\tau}\times\hat{\varrho}} \leftarrow Z|_{\hat{\tau}\times\hat{\varrho}} + AB^*$.

Problem: Changing the cluster bases influences all blocks intersecting rows in $\hat{\tau}$ or columns in $\hat{\sigma}$.

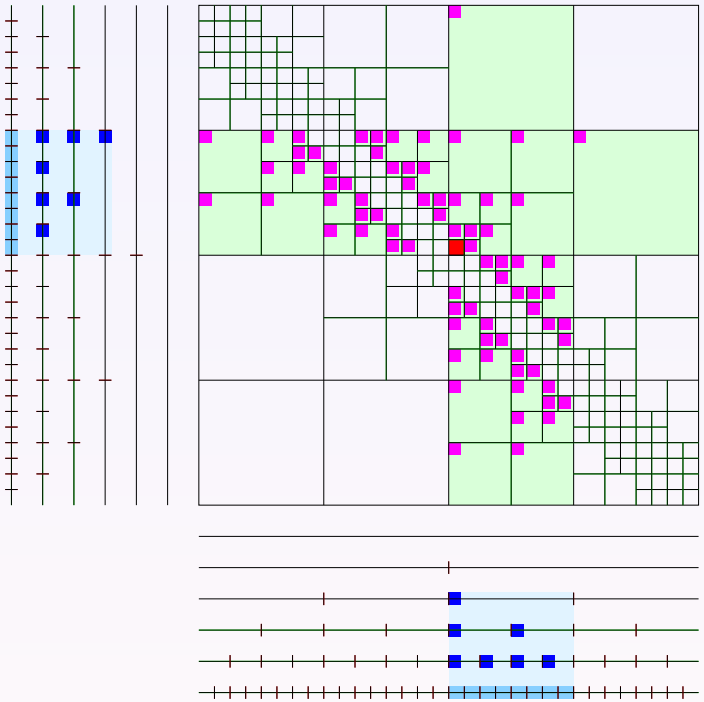


Observation: Nested cluster basis.
$\rightarrow$ Updating transfer matrix updates all ancestors of a cluster.

Local low-rank update

Goal: Local update $Z|_{\hat{\tau}\times\hat{\varrho}} \leftarrow Z|_{\hat{\tau}\times\hat{\varrho}} + AB^*$.

Problem: Changing the cluster bases influences all blocks intersecting rows in $\hat{\tau}$ or columns in $\hat{\sigma}$.



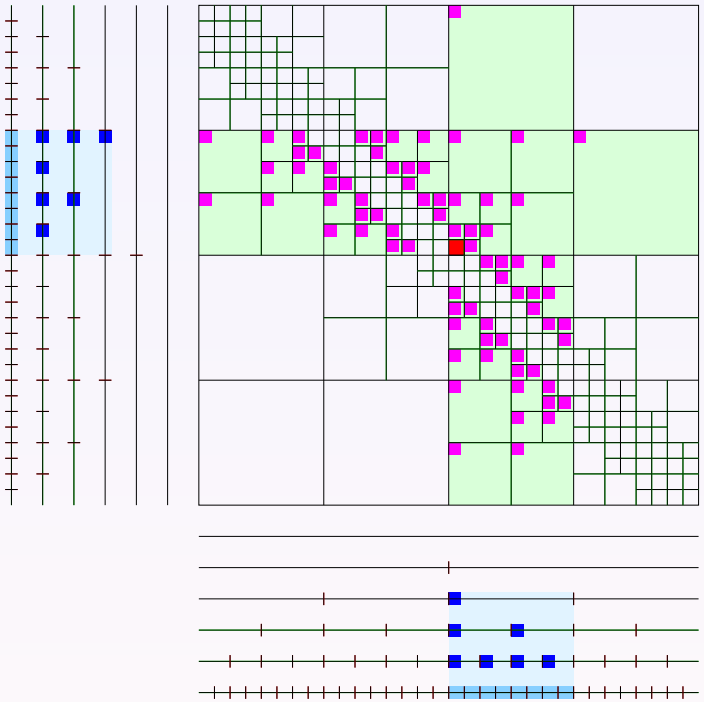
Observation: Nested cluster basis.
$\rightarrow$ Updating transfer matrix updates all ancestors of a cluster.

Important: Keep weights updated.
$\rightarrow$ No visits to ancestors required.

Local low-rank update

Goal: Local update $Z|_{\hat{\tau}\times\hat{\varrho}} \leftarrow Z|_{\hat{\tau}\times\hat{\varrho}} + AB^*$.

Problem: Changing the cluster bases influences all blocks intersecting rows in $\hat{\tau}$ or columns in $\hat{\sigma}$.



Observation: Nested cluster basis.
$\rightarrow$ Updating transfer matrix updates all ancestors of a cluster.

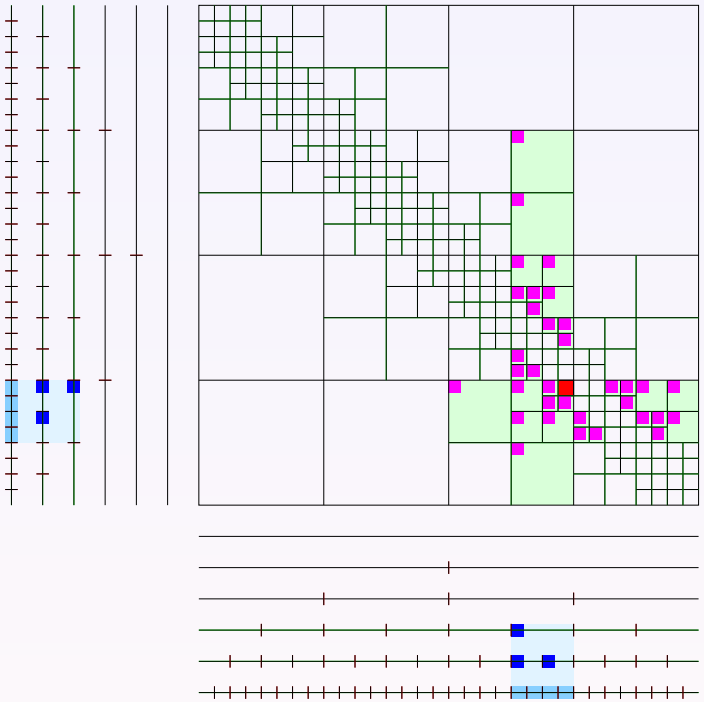
Important: Keep weights updated.
$\rightarrow$ No visits to ancestors required.

Result: Complexity $\mathcal{O}(k^2(\#\hat{\tau} + \#\hat{\sigma}))$

Local low-rank update

Goal: Local update $Z|_{\hat{\tau}\times\hat{\varrho}} \leftarrow Z|_{\hat{\tau}\times\hat{\varrho}} + AB^*$.

Problem: Changing the cluster bases influences all blocks intersecting rows in $\hat{\tau}$ or columns in $\hat{\sigma}$.



Observation: Nested cluster basis.
$\rightarrow$ Updating transfer matrix updates all ancestors of a cluster.

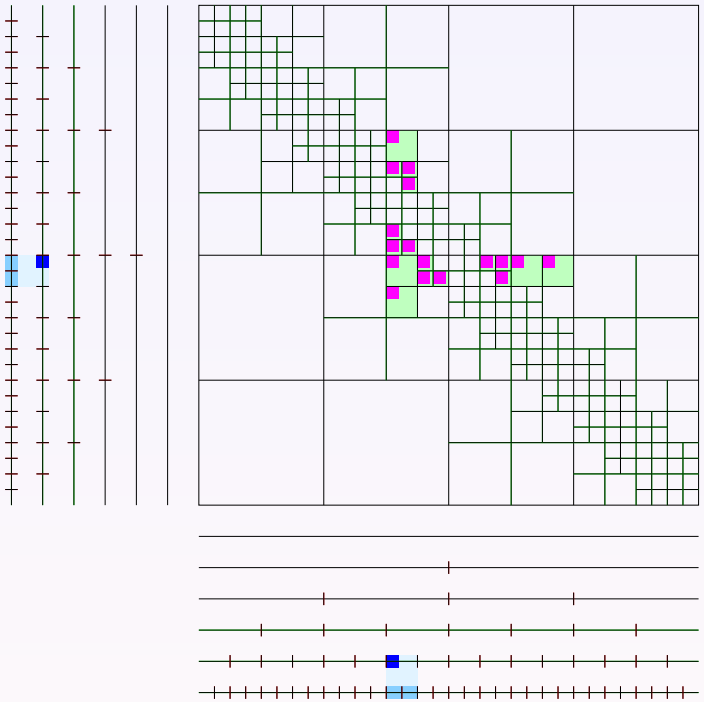
Important: Keep weights updated.
$\rightarrow$ No visits to ancestors required.

Result: Complexity $\mathcal{O}(k^2(\#\hat{\tau} + \#\hat{\sigma}))$

Local low-rank update

Goal: Local update $Z|_{\hat{\tau}\times\hat{\varrho}} \leftarrow Z|_{\hat{\tau}\times\hat{\varrho}} + AB^*$.

Problem: Changing the cluster bases influences all blocks intersecting rows in $\hat{\tau}$ or columns in $\hat{\sigma}$.



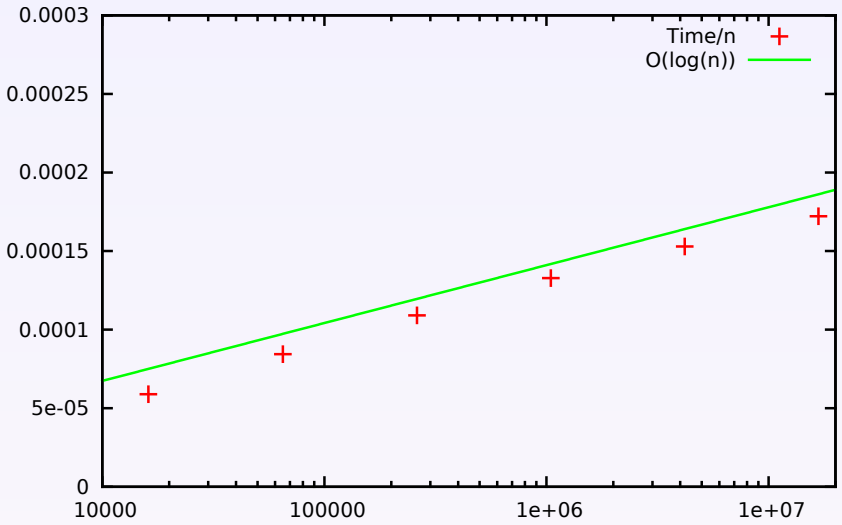
Observation: Nested cluster basis.
$\rightarrow$ Updating transfer matrix updates all ancestors of a cluster.

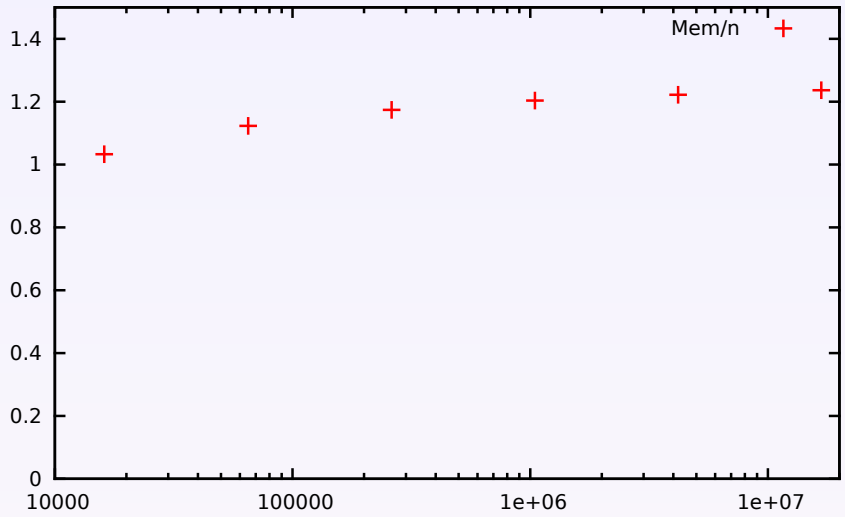
Important: Keep weights updated.
$\rightarrow$ No visits to ancestors required.

Result: Complexity $\mathcal{O}(k^2(\#\hat{\tau} + \#\hat{\sigma}))$

Experiment: FEM Cholesky decomposition

Goal: Approximate Cholesky decomposition of a FEM stiffness matrix.



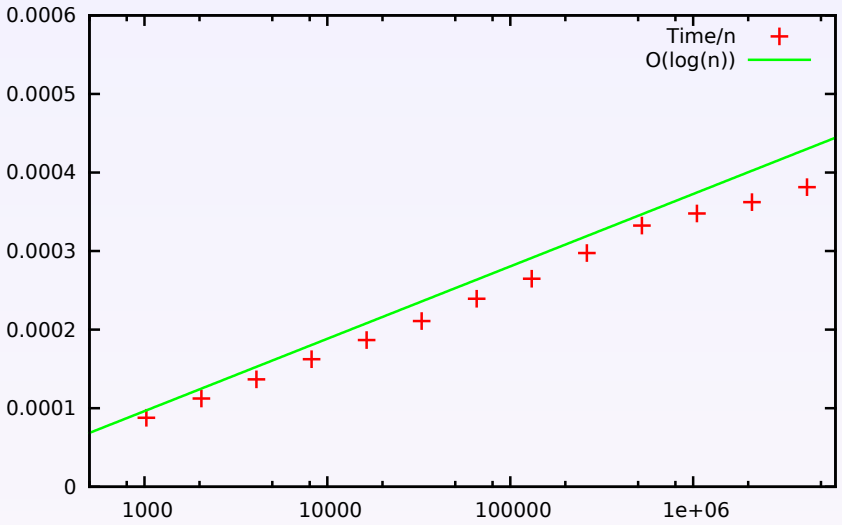


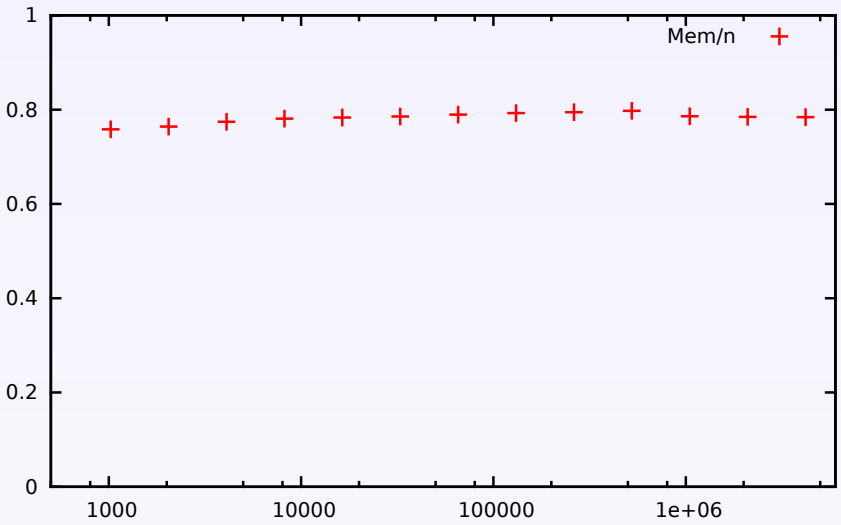
Results:

- Accuracy $\|I - \widetilde{L}^{-*}\widetilde{L}^{-1}A\|_2 \approx 0.1$.
- Factorization in $\sim n \log n$ operations.
- Storage requirements $\sim n$.

Experiment: BEM Cholesky decomposition

Goal: Approximate Cholesky decomposition of a BEM stiffness matrix.





Results:

- Accuracy $\|I - \widetilde{L}^{-*}\widetilde{L}^{-1}A\|_2 \approx 0.2$.
- Factorization in $\sim n \log n$ operations.
- Storage requirements $\sim n$.

Conclusion

$\mathcal{H}^2$-matrices can be far more efficient than hierarchical matrices.

Boundary integral matrices can be treated by a combination of Green's representation formula, quadrature and cross approximation.

Preconditioners for BE and FE problems can be constructed by local low-rank updates in $\mathcal{O}(n \log(n))$ operations with $\mathcal{O}(n)$ units of storage.

Conclusion

$\mathcal{H}^2$-matrices can be far more efficient than hierarchical matrices.

Boundary integral matrices can be treated by a combination of Green's representation formula, quadrature and cross approximation.

Preconditioners for BE and FE problems can be constructed by local low-rank updates in $\mathcal{O}(n \log(n))$ operations with $\mathcal{O}(n)$ units of storage.

Future work:

- First release of `H2Lib` package.
- Improve runtime for preconditioners.
- Parallelization, both OpenMP and MPI.