Hierarchical Tensor Approximation

Steffen Börm, Dirk Boysen, and Isabelle Greff

Christian-Albrechts-Universität zu Kiel

Dagstuhl Seminar
"Uncertainty Quantification and High Performance Computing"
12th of September, 2016

Overview

Overview

Model problem

Consider the solutions of

$$-\Delta u(x\quad) = f(x\quad), \qquad \text{for all } x \in D,$$
$$u(x\quad) = 0 \qquad \text{for all } x \in \partial D$$

Model problem

Consider the solutions of

$$\begin{aligned} -\Delta u(x,\omega) &= f(x,\omega), && \text{for all } x \in D, \\ u(x,\omega) &= 0 && \text{for all } x \in \partial D \end{aligned}$$

with $\omega \in \Omega$ in a probability space $(\Omega, \mathcal{M}, P)$.

Model problem

Consider the solutions of

$$\begin{aligned} -\Delta u(x,\omega) &= f(x,\omega), && \text{for all } x \in D, \\ u(x,\omega) &= 0 && \text{for all } x \in \partial D \end{aligned}$$

with $\omega \in \Omega$ in a probability space $(\Omega, \mathcal{M}, P)$.

Goal: Compute expected values and correlations, i.e.,

$$E_u(x) = \int_\Omega u(x,\omega)\, dP(\omega), \qquad C_u(x,y) = \int_\Omega u(x,\omega) u(y,\omega)\, dP(\omega)$$

without solving the original equation for all $\omega \in \Omega$.

Equations for expected value and correlation

Schwab/Todor: Expected value and correlation are solutions of partial differential equations.

$$-\Delta E_u(x) = E_f(x), \qquad E_f(x) = \int_\Omega f(x,\omega)\,dP(\omega),$$

$$\Delta_x \Delta_y C_u(x,y) = C_f(x,y), \qquad C_f(x,y) = \int_\Omega f(x,\omega) f(y,\omega)\,dP(\omega).$$

Result: Expected values can be computed directly by solving a partial differential equation in the domain D.

Challenge: To compute the correlation, we have to solve a partial differential equation in $D \times D$.
$\rightarrow$ Significantly higher computational complexity.

Simple finite element approach

Variational formulation obtained by partial integration:

$$\int_{D\times D} \langle \nabla_x \nabla_y v(x,y), \nabla_x \nabla_y C_u(x,y) \rangle \, d(x,y) = \int_{D\times D} v(x,y) C_f(x,y) \, d(x,y)$$

for all $v \in V \otimes V$, $V = H_0^1(D)$.

Finite elements: Given FE space V_h with basis $(\varphi_i)_{i=1}^n$ for D, approximate $V \otimes V$ by tensor FE space

$$V_h \otimes V_h = \operatorname{span}\{\varphi_i \otimes \varphi_j \ : \ i,j \in [1:n]\}.$$

Result: Fairly standard FE discretization, but n^2 degrees of freedom.
$\rightarrow$ High computational complexity if $n \gg 1$.

Overview

Matrix equation

Idea: Collect tensor basis coefficients in a matrix $U \in \mathbb{R}^{n \times n}$,

$$C_u \approx \sum_{i,j=1}^{n} u_{ij}\, \varphi_i \otimes \varphi_j.$$

Result: FE discretization leads to

$$AUA = B,$$

with the system matrix A of the original PDE and the right-hand side

$$b_{ij} = \int_D \int_D \varphi_i(x) \varphi_j(y) C_f(x,y)\, dy\, dx.$$

Hierarchical matrices

Observation: Certain matrices, e.g., A^{-1} and B, can be split into submatrices that can be approximated by low rank.
$\rightarrow$ Hierarchical matrix representation.

- Storage requirement $\mathcal{O}(nk \log n)$ with local rank k,
- matrix-vector multiplication in $\mathcal{O}(nk \log n)$,
- approximated matrix-matrix multiplication in $\mathcal{O}(nk^2 \log^2 n)$,
- approximated inversion and factorization in $\mathcal{O}(nk^2 \log^2 n)$.

Pentenrieder/Schwab, Dölz/Harbrecht/Schwab: Solution C_u is smooth in certain subdomains $\tau \times \sigma$.
$\rightarrow$ Matrix U can be approximated by a hierarchical matrix.

Preconditioned Richardson iteration

Goal: Solve $AUA = B \iff U = U + A^{-1}(B - AUA)A^{-1}$.

Dölz/Harbrecht/Schwab: Approximate A^{-1} and B by hierarchical matrices, iterate

$$U^{(m+1)} := U^{(m)} + \widetilde{A^{-1}}(\widetilde{B} - AU^{(m)}A)\widetilde{A^{-1}}$$

using approximated matrix-matrix multiplications.

+ Fully adaptive method.
+ Straightforward implementation if $\mathcal{H}$-matrix arithmetic operations available (HLib, AHMED, HLib pro, H2Lib).
- Time-consuming.

Matrix Galerkin method

Variational formulation: $AUA = B$ is equivalent with

$$\langle V, AUA \rangle_F = \langle V, B \rangle_F \qquad \text{for all } V \in \mathbb{R}^{n \times n}.$$

B./Boysen: Use subspace $\mathcal{H} \subseteq \mathbb{R}^{n \times n}$ of $\mathcal{H}^2$-matrices, find $\widetilde{U} \in \mathcal{H}$ with

$$\langle \widetilde{V}, A\widetilde{U}A \rangle_F = \langle \widetilde{V}, B \rangle_F \qquad \text{for all } \widetilde{V} \in \mathcal{H}.$$

+ Best-approximation property.
+ Block basis leads to sparse linear system.
+ Parallelization fairly straightforward.
- Ill-conditioned system.
- Time-consuming.

Overview

Goal

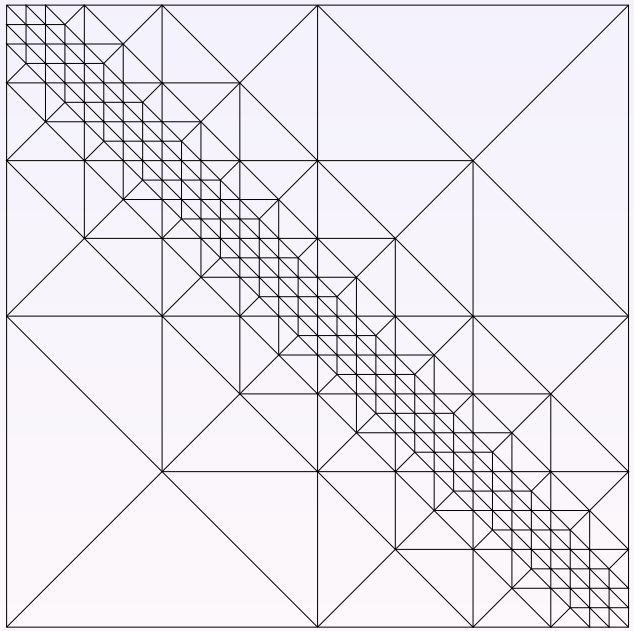
Treat variational problem

$$\int_D \int_D \langle \nabla_x \nabla_y v(x,y), \nabla_x \nabla_y u(x,y) \rangle \, dy \, dx = \int_D \int_D v(x,y) C_f(x,y) \, dy \, dx$$

directly by suitable Galerkin discretization.

Challenge: Local refinement essential for acceptable complexity.
Standard techniques quite complicated if $D \times D$ is four- or six-dimensional.

Desirable: Variable polynomial degrees to obtain optimal order of complexity.



Partition of unity

Babuška/Melenk: Construct trial space using a partition of unity.

$$\sum_{b\in\mathcal{G}} \chi_b = 1 \qquad \longrightarrow \qquad u = \sum_{b\in\mathcal{G}} \chi_b u \approx \sum_{b\in\mathcal{G}} \chi_b p_b$$

with $u|_{\operatorname{supp}\chi_b} \approx p_b$ with a polynomial p_b.

B./Sauter: Local “mesh refinement” via hierarchical partition of unity.

Idea: Construct hierarchical partition of unity for $D \times D$ using a standard finite element hierarchy for the domain D.

Partition of unity

Babuška/Melenk: Construct trial space using a partition of unity.

$$\sum_{b\in\mathcal{G}} \chi_b = 1 \qquad \longrightarrow \qquad u = \sum_{b\in\mathcal{G}} \chi_b u \approx \sum_{b\in\mathcal{G}} \chi_b p_b$$

with $u|_{\operatorname{supp}\chi_b} \approx p_b$ with a polynomial p_b.

B./Sauter: Local “mesh refinement” via hierarchical partition of unity.

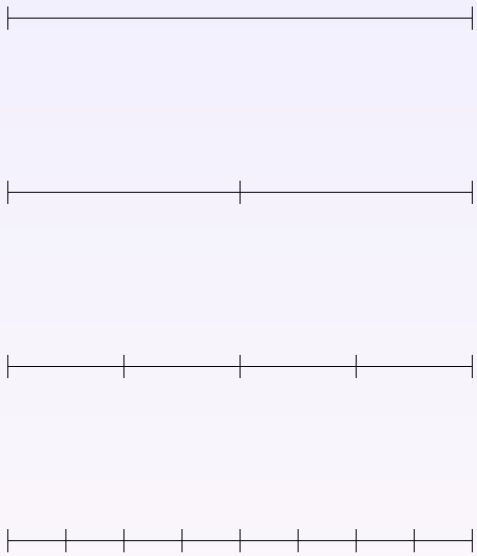
Idea: Construct hierarchical partition of unity for $D \times D$ using a standard finite element hierarchy for the domain D.
(Try to preserve tensor structure)

Finite element hierarchy

Approach: Use standard finite element hierarchy in D.

$$V_0 \subseteq V_1 \subseteq \ldots \subseteq V_L, \qquad V_\ell = \operatorname{span}\{\varphi_{\ell,i} \;:\; i \in \mathcal{I}_\ell\}.$$

Different resolutions on different levels.



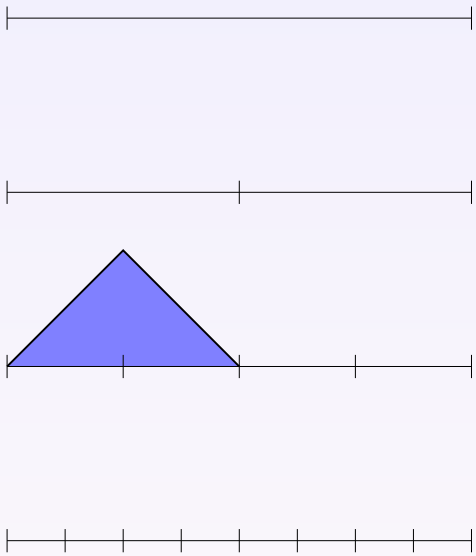
Finite element hierarchy

Approach: Use standard finite element hierarchy in D.

$$V_0 \subseteq V_1 \subseteq \ldots \subseteq V_L, \qquad V_\ell = \operatorname{span}\{\varphi_{\ell,i} \;:\; i \in \mathcal{I}_\ell\}.$$

Different resolutions on different levels.

H^1-conforming nodal basis with nodes $\xi_{\ell,i}$.



Finite element hierarchy

Approach: Use standard finite element hierarchy in D.

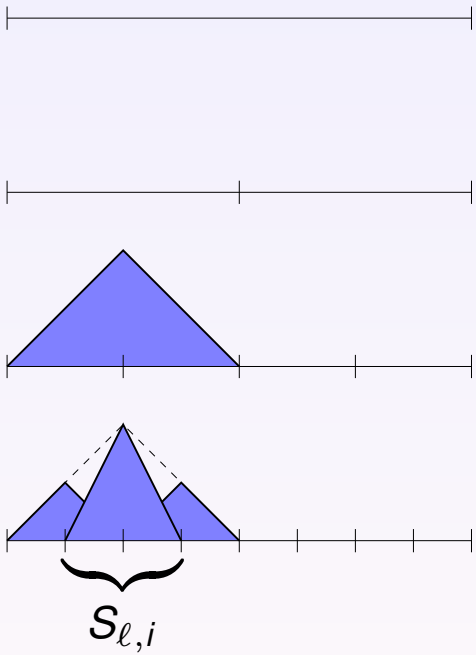
$$V_0 \subseteq V_1 \subseteq \ldots \subseteq V_L, \qquad V_\ell = \operatorname{span}\{\varphi_{\ell,i} \;:\; i \in \mathcal{I}_\ell\}.$$

Different resolutions on different levels.

H^1-conforming nodal basis with nodes $\xi_{\ell,i}$.

Grid transfer by sparse interpolation.

$$\varphi_{\ell,i} = \sum_{j \in S_{\ell,i}} \varphi_{\ell,i}(\xi_{\ell+1,j})\, \varphi_{\ell+1,j}$$



Finite element hierarchy

Approach: Use standard finite element hierarchy in D.

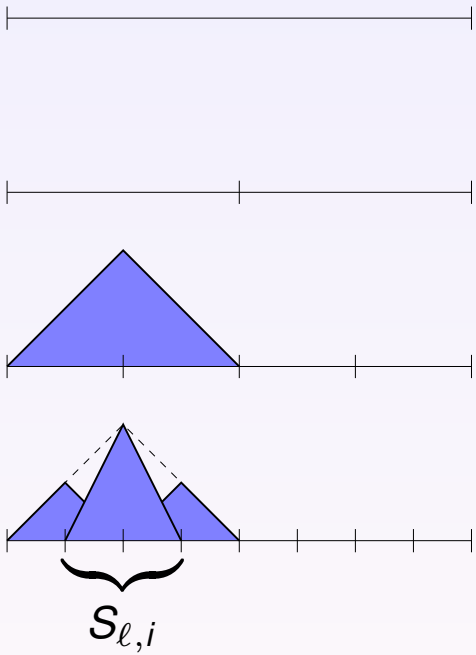
$$V_0 \subseteq V_1 \subseteq \ldots \subseteq V_L, \qquad V_\ell = \operatorname{span}\{\varphi_{\ell,i} \ : \ i \in \mathcal{I}_\ell\}.$$

Different resolutions on different levels.

H^1-conforming nodal basis with nodes $\xi_{\ell,i}$.

Grid transfer by sparse interpolation.

$$\varphi_{\ell,i} = \sum_{j \in S_{\ell,i}} \varphi_{\ell,i}(\xi_{\ell+1,j})\, \varphi_{\ell+1,j}$$



Notation: $\mathcal{T}_\ell = \{(\ell, i) \ : \ i \in \mathcal{I}_\ell\}$, $\mathcal{T} = \bigcup_{\ell=0}^{L} \mathcal{T}_\ell$,
$\operatorname{level}(t) = \ell$, $\tau = \operatorname{supp} \varphi_t$ for all $t = (\ell, i) \in \mathcal{T}$.

Finite element hierarchy

Approach: Use standard finite element hierarchy in D.

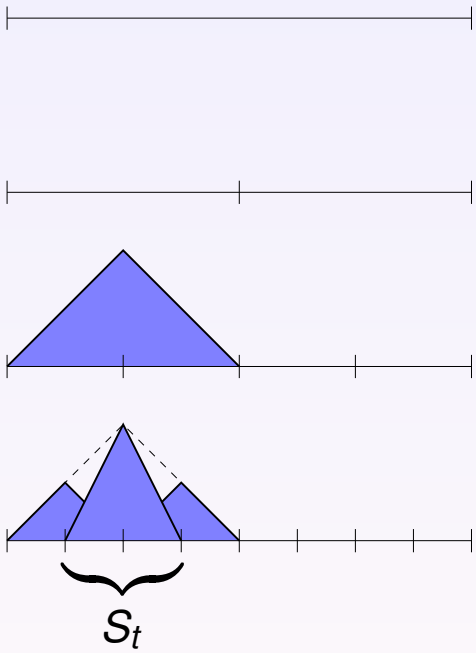
$$V_0 \subseteq V_1 \subseteq \ldots \subseteq V_L, \qquad V_\ell = \operatorname{span}\{\varphi_{\ell,i} \ : \ i \in \mathcal{I}_\ell\}.$$

Different resolutions on different levels.

H^1-conforming nodal basis with nodes ξ_t.

Grid transfer by sparse interpolation.

$$\varphi_t = \sum_{t' \in S_t} \varphi_t(\xi_{t'})\, \varphi_{t'}$$



Notation: $\mathcal{T}_\ell = \{(\ell, i) \ : \ i \in \mathcal{I}_\ell\}$, $\mathcal{T} = \bigcup_{\ell=0}^{L} \mathcal{T}_\ell$,
$\operatorname{level}(t) = \ell$, $\tau = \operatorname{supp} \varphi_t$ for all $t = (\ell, i) \in \mathcal{T}$.

Hierarchical partition of unity I

For a standard nodal basis, we have

$$\sum_{t,s\in\mathcal{T}_0} \varphi_t \otimes \varphi_s = 1,$$

i.e., a partition of unity. $\rightarrow$ Starting point for refinement.

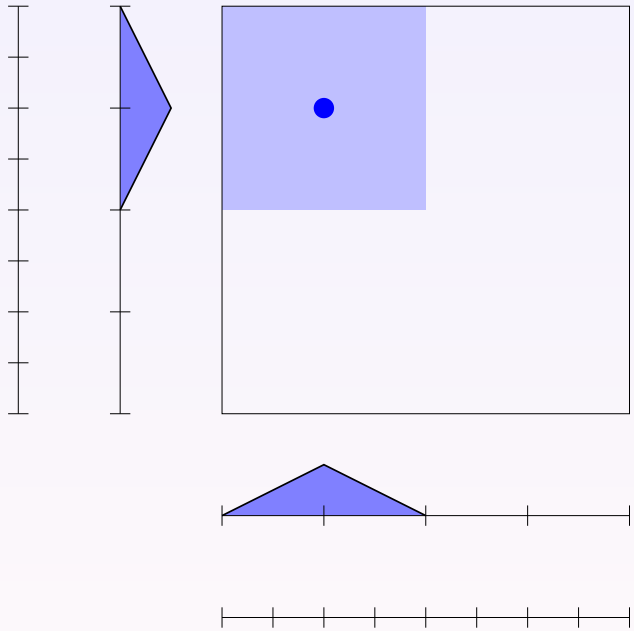
Hierarchical partition of unity I

For a standard nodal basis, we have

$$\sum_{t,s\in\mathcal{T}_0} \varphi_t \otimes \varphi_s = 1,$$

i.e., a partition of unity. $\rightarrow$ Starting point for refinement.

Idea: If the support of $\varphi_t \otimes \varphi_s$ is too large, replace by fine-level functions $\varphi_{t'} \otimes \varphi_{s'}$ for $t' \in S_t$ and $s' \in S_s$.



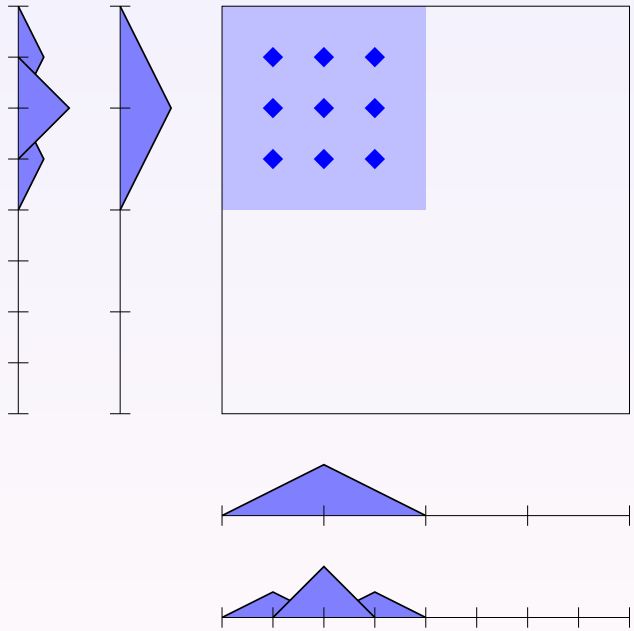
Hierarchical partition of unity I

For a standard nodal basis, we have

$$\sum_{t,s\in\mathcal{T}_0} \varphi_t \otimes \varphi_s = 1,$$

i.e., a partition of unity. $\rightarrow$ Starting point for refinement.

Idea: If the support of $\varphi_t \otimes \varphi_s$ is too large, replace by fine-level functions $\varphi_{t'} \otimes \varphi_{s'}$ for $t' \in S_t$ and $s' \in S_s$.



Hierarchical partition of unity I

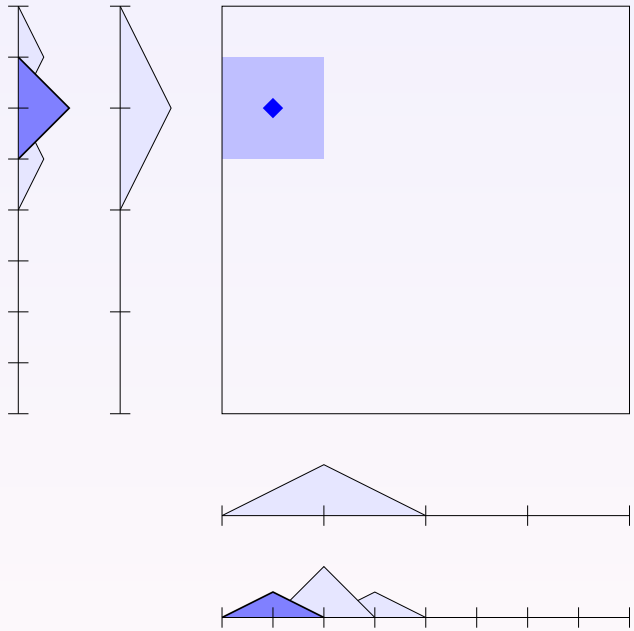
For a standard nodal basis, we have

$$\sum_{t,s\in\mathcal{T}_0} \varphi_t \otimes \varphi_s = 1,$$

i.e., a partition of unity. $\rightarrow$ Starting point for refinement.

Idea: If the support of $\varphi_t \otimes \varphi_s$ is too large, replace by fine-level functions $\varphi_{t'} \otimes \varphi_{s'}$ for $t' \in S_t$ and $s' \in S_s$.

Result: Supports halved in size.
$\rightarrow$ "Grid refinement"



Hierarchical partition of unity I

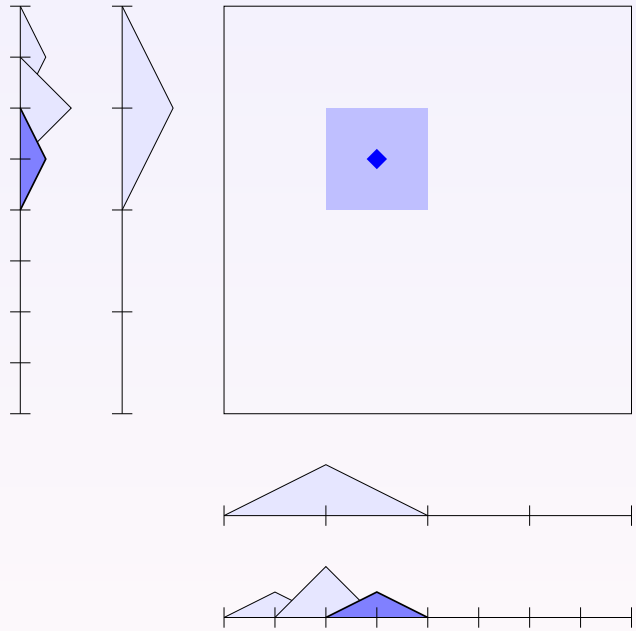
For a standard nodal basis, we have

$$\sum_{t,s\in\mathcal{T}_0} \varphi_t \otimes \varphi_s = 1,$$

i.e., a partition of unity. $\rightarrow$ Starting point for refinement.

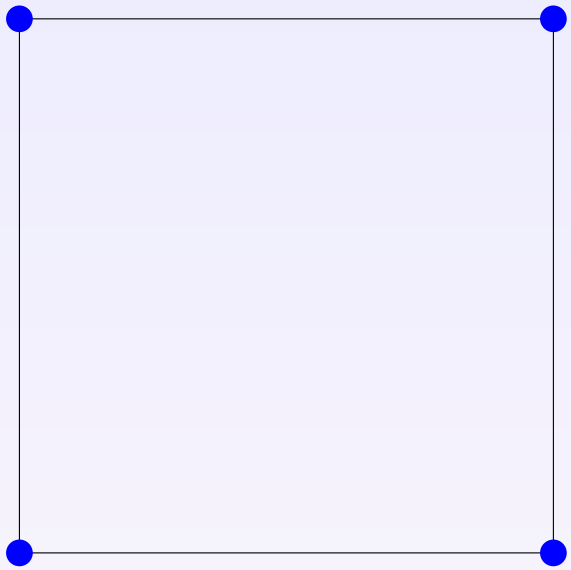
Idea: If the support of $\varphi_t \otimes \varphi_s$ is too large, replace by fine-level functions $\varphi_{t'} \otimes \varphi_{s'}$ for $t' \in S_t$ and $s' \in S_s$.

Result: Supports halved in size.
$\rightarrow$ "Grid refinement"



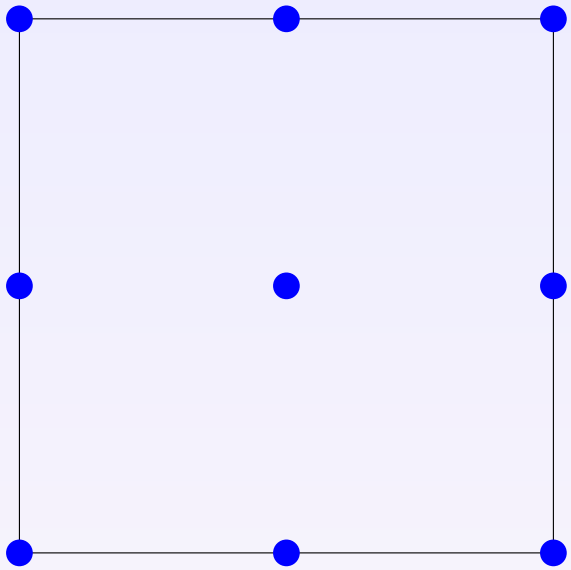
Hierarchical partition of unity II

Construction: If $\tau \times \sigma$ is too close to a singularity, replace the basis function $\varphi_t \otimes \varphi_s$ by $\varphi_{t'} \otimes \varphi_{s'}$ for $t' \in S_t$ and $s' \in S_s$.



Hierarchical partition of unity II

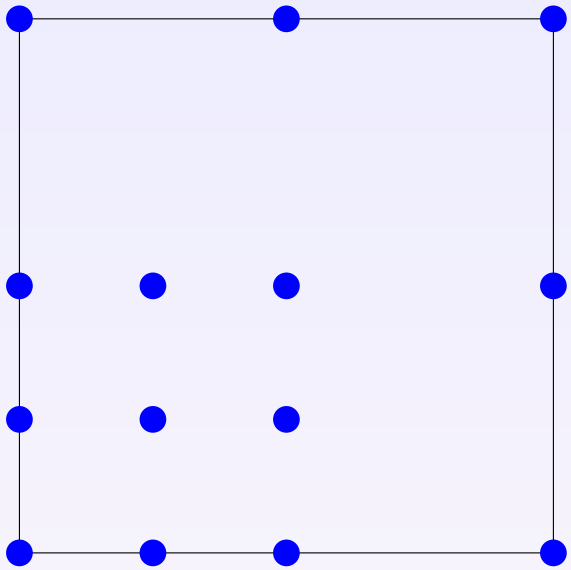
Construction: If $\tau \times \sigma$ is too close to a singularity, replace the basis function $\varphi_t \otimes \varphi_s$ by $\varphi_{t'} \otimes \varphi_{s'}$ for $t' \in S_t$ and $s' \in S_s$.



Hierarchical partition of unity II

Construction: If $\tau \times \sigma$ is too close to a singularity, replace the basis function $\varphi_t \otimes \varphi_s$ by $\varphi_{t'} \otimes \varphi_{s'}$ for $t' \in S_t$ and $s' \in S_s$.

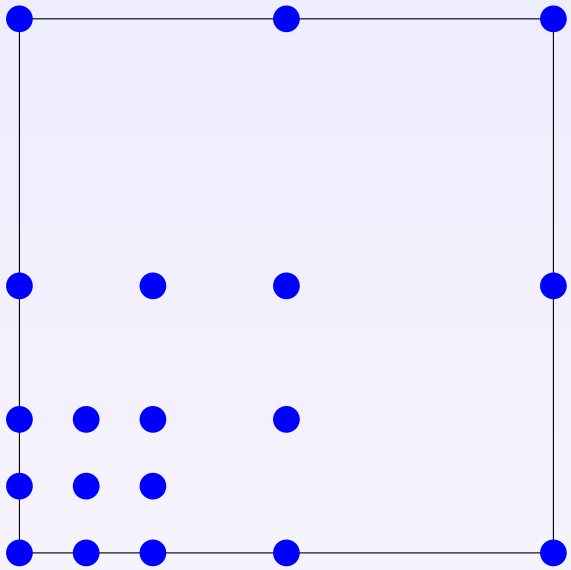
Repeat if necessary



Hierarchical partition of unity II

Construction: If $\tau \times \sigma$ is too close to a singularity, replace the basis function $\varphi_t \otimes \varphi_s$ by $\varphi_{t'} \otimes \varphi_{s'}$ for $t' \in S_t$ and $s' \in S_s$.

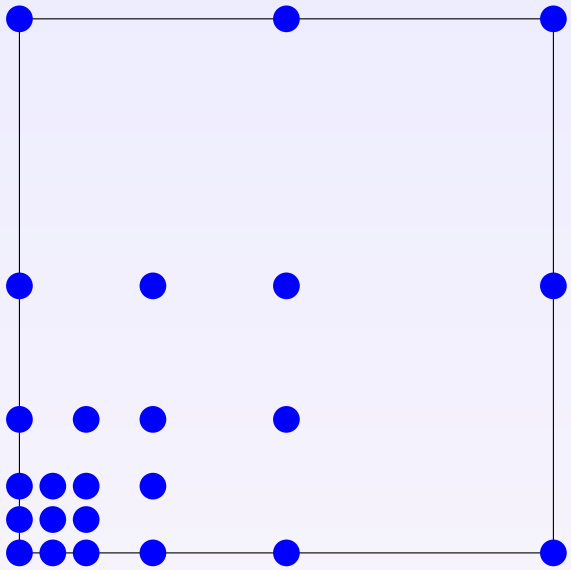
Repeat if necessary



Hierarchical partition of unity II

Construction: If $\tau \times \sigma$ is too close to a singularity, replace the basis function $\varphi_t \otimes \varphi_s$ by $\varphi_{t'} \otimes \varphi_{s'}$ for $t' \in S_t$ and $s' \in S_s$.

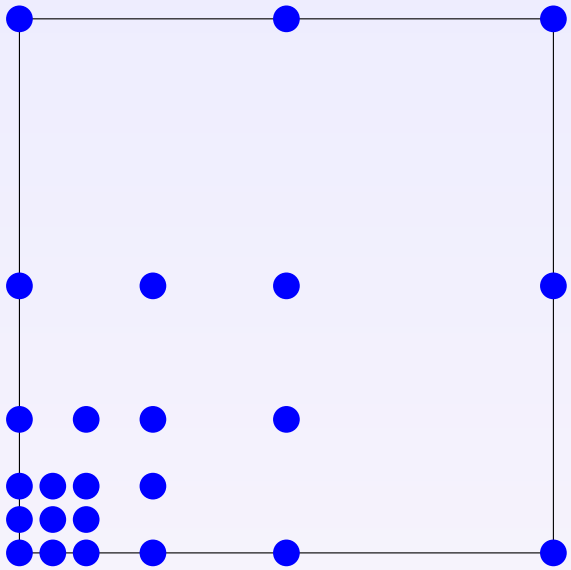
Repeat if necessary until maximal level L has been reached.



Hierarchical partition of unity II

Construction: If $\tau \times \sigma$ is too close to a singularity, replace the basis function $\varphi_t \otimes \varphi_s$ by $\varphi_{t'} \otimes \varphi_{s'}$ for $t' \in S_t$ and $s' \in S_s$.

Repeat if necessary until maximal level L has been reached.



Result: Hierarchical partition of unity.

$$\sum_{(t,s)\in\mathcal{G}} \alpha_{t,s}\, \varphi_t \otimes \varphi_s = 1.$$

Trial space

Approach: For each $(t, s) \in \mathcal{G}$, we are looking for an approximation

$$u|_{\tau\times\sigma} \approx \sum_{\nu,\mu=1}^{k} u_{ts,\nu\mu}\, p_{t,\nu} \otimes p_{s,\mu}$$

with polynomial bases $(p_{t,\nu})_{\nu=1}^{k}$, $(p_{s,\mu})_{\mu=1}^{k}$.

Partition of unity: Global approximation given by

$$u \approx \sum_{(t,s)\in\mathcal{G}} \sum_{\nu,\mu=1}^{k} u_{ts,\nu\mu}\, \varphi_t \otimes \varphi_s\, p_{t,\nu} \otimes p_{s,\mu}.$$

Trial space

Approach: For each $(t, s) \in \mathcal{G}$, we are looking for an approximation

$$u|_{\tau\times\sigma} \approx \sum_{\nu,\mu=1}^{k} u_{ts,\nu\mu}\, p_{t,\nu} \otimes p_{s,\mu}$$

with polynomial bases $(p_{t,\nu})_{\nu=1}^{k}$, $(p_{s,\mu})_{\mu=1}^{k}$.

Partition of unity: Global approximation given by

$$u \approx \sum_{(t,s)\in\mathcal{G}} \sum_{\nu,\mu=1}^{k} u_{ts,\nu\mu}\, (\varphi_t p_{t,\nu}) \otimes (\varphi_s p_{s,\mu}).$$

Trial space

Approach: For each $(t, s) \in \mathcal{G}$, we are looking for an approximation

$$u|_{\tau\times\sigma} \approx \sum_{\nu,\mu=1}^{k} u_{ts,\nu\mu}\, p_{t,\nu} \otimes p_{s,\mu}$$

with polynomial bases $(p_{t,\nu})_{\nu=1}^{k}$, $(p_{s,\mu})_{\mu=1}^{k}$.

Partition of unity: Global approximation given by

$$u \approx \sum_{(t,s)\in\mathcal{G}} \sum_{\nu,\mu=1}^{k} u_{ts,\nu\mu}\, (\varphi_t p_{t,\nu}) \otimes (\varphi_s p_{s,\mu}).$$

Trial space $V_{\mathcal{G}}$ spanned by tensor functions $(\varphi_t p_{t,\nu}) \otimes (\varphi_s p_{s,\mu})$.

Overview

Variational equation

Reminder: We are looking for $u \in V_{\mathcal{G}}$ with

$$a(v,u) = \lambda(v) \qquad \text{for all } v \in V_{\mathcal{G}}$$

with a functional λ and

$$a(v,u) = \int_D \int_D \langle \nabla_x \nabla_y v(x,y), \nabla_x \nabla_y u(x,y) \rangle \, dy \, dx.$$

Variational equation

Reminder: We are looking for $u \in V_{\mathcal{G}}$ with

$$a(v,u) = \lambda(v) \qquad \text{for all } v \in V_{\mathcal{G}}$$

with a functional λ and

$$a(v,u) = \int_D \int_D \langle \nabla_x \nabla_y v(x,y), \nabla_x \nabla_y u(x,y) \rangle \, dy \, dx.$$

Advantage: $V_{\mathcal{G}}$ spanned by tensor functions $v = v_t \otimes v_s$, $u = u_t \otimes u_s$.

$$a(v,u) = \int_D \int_D \langle \nabla_x \nabla_y v(x,y), \nabla_x \nabla_y u(x,y) \rangle \, dy \, dx$$

Variational equation

Reminder: We are looking for $u \in V_{\mathcal{G}}$ with

$$a(v,u) = \lambda(v) \qquad \text{for all } v \in V_{\mathcal{G}}$$

with a functional λ and

$$a(v,u) = \int_D \int_D \langle \nabla_x \nabla_y v(x,y), \nabla_x \nabla_y u(x,y) \rangle \, dy \, dx.$$

Advantage: $V_{\mathcal{G}}$ spanned by tensor functions $v = v_t \otimes v_s$, $u = u_t \otimes u_s$.

$$a(v,u) = \int_D \int_D \langle \nabla_x \nabla_y (v_t \otimes v_s)(x,y), \nabla_x \nabla_y (u_t \otimes u_s)(x,y) \rangle \, dy \, dx$$

Variational equation

Reminder: We are looking for $u \in V_{\mathcal{G}}$ with

$$a(v,u) = \lambda(v) \qquad \text{for all } v \in V_{\mathcal{G}}$$

with a functional λ and

$$a(v,u) = \int_D \int_D \langle \nabla_x \nabla_y v(x,y), \nabla_x \nabla_y u(x,y) \rangle \, dy \, dx.$$

Advantage: $V_{\mathcal{G}}$ spanned by tensor functions $v = v_t \otimes v_s$, $u = u_t \otimes u_s$.

$$a(v,u) = \int_D \int_D \langle (\nabla v_t) \otimes (\nabla v_s)(x,y), (\nabla u_t) \otimes (\nabla u_s)(x,y) \rangle \, dy \, dx$$

Variational equation

Reminder: We are looking for $u \in V_{\mathcal{G}}$ with

$$a(v,u) = \lambda(v) \qquad \text{for all } v \in V_{\mathcal{G}}$$

with a functional λ and

$$a(v,u) = \int_D \int_D \langle \nabla_x \nabla_y v(x,y), \nabla_x \nabla_y u(x,y) \rangle \, dy \, dx.$$

Advantage: $V_{\mathcal{G}}$ spanned by tensor functions $v = v_t \otimes v_s$, $u = u_t \otimes u_s$.

$$a(v,u) = \int_D \langle \nabla v_t(x), \nabla u_t(x) \rangle \, dx \int_D \langle \nabla v_s(y), \nabla u_s(y) \rangle \, dy.$$

Variational equation

Reminder: We are looking for $u \in V_{\mathcal{G}}$ with

$$a(v,u) = \lambda(v) \qquad \text{for all } v \in V_{\mathcal{G}}$$

with a functional λ and

$$a(v,u) = \int_D \int_D \langle \nabla_x \nabla_y v(x,y), \nabla_x \nabla_y u(x,y) \rangle \, dy \, dx.$$

Advantage: $V_{\mathcal{G}}$ spanned by tensor functions $v = v_t \otimes v_s$, $u = u_t \otimes u_s$.

$$a(v_t \otimes v_s, u_t \otimes u_s) = a_D(v_t, u_t)\, a_D(v_s, u_s),$$

$$a_D(w,z) = \int_D \langle \nabla w(x), \nabla z(x) \rangle \, dx.$$

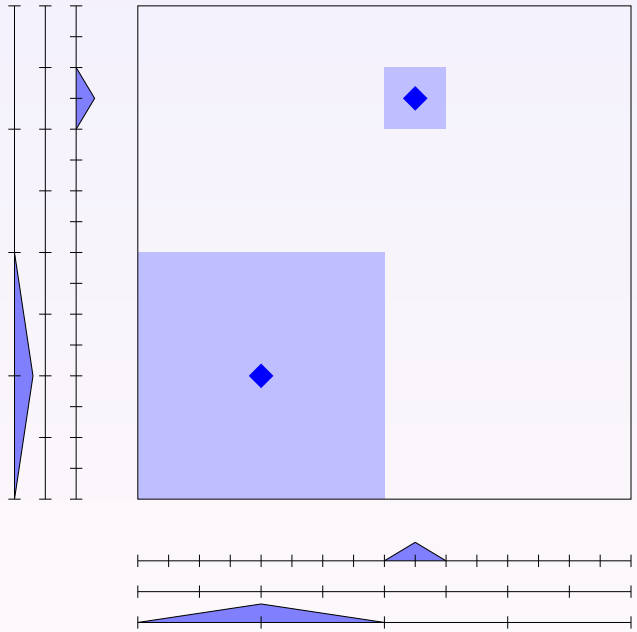
Quadrature algorithms only required for the original domain D.

Sparsity pattern

Goal: Find all pairs $(t, s), (r, p) \in \mathcal{G}$ with

$$(\tau \times \sigma) \cap (\varrho \times \pi) \neq \emptyset.$$

Challenge: (t, s) and (r, p) may correspond to different levels, but we only know the neighbours within the same level.

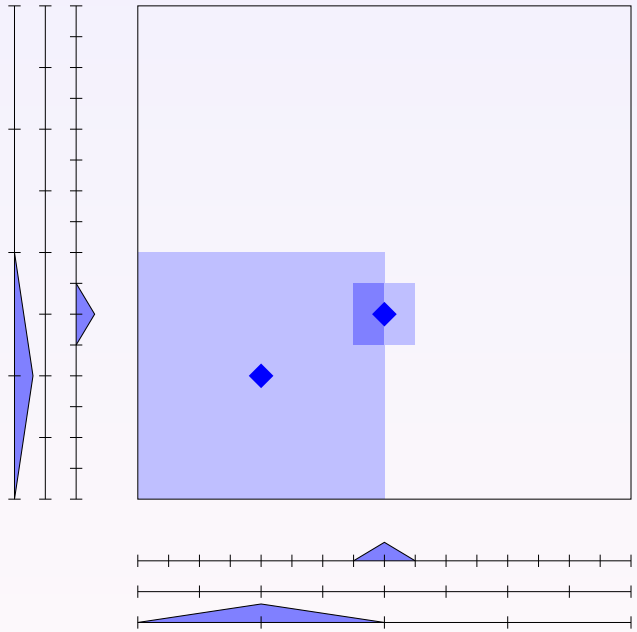


Sparsity pattern

Goal: Find all pairs $(t, s), (r, p) \in \mathcal{G}$ with

$$(\tau \times \sigma) \cap (\varrho \times \pi) \neq \emptyset.$$

Challenge: (t, s) and (r, p) may correspond to different levels, but we only know the neighbours within the same level.



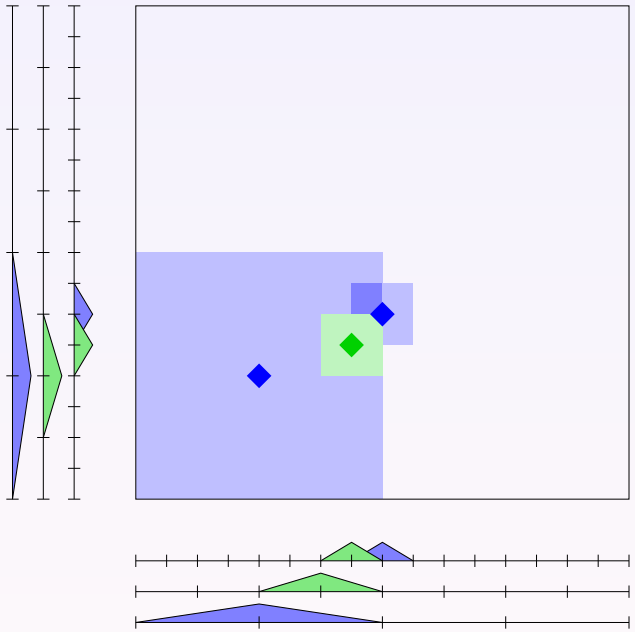
Sparsity pattern

Goal: Find all pairs $(t, s), (r, p) \in \mathcal{G}$ with

$$(\tau \times \sigma) \cap (\varrho \times \pi) \neq \emptyset.$$

Challenge: (t, s) and (r, p) may correspond to different levels, but we only know the neighbours within the same level.

Idea: If $\operatorname{level}(t, s) < \operatorname{level}(r, p)$, check descendants t^* and s^* on the level of (r, p).



Sparsity pattern

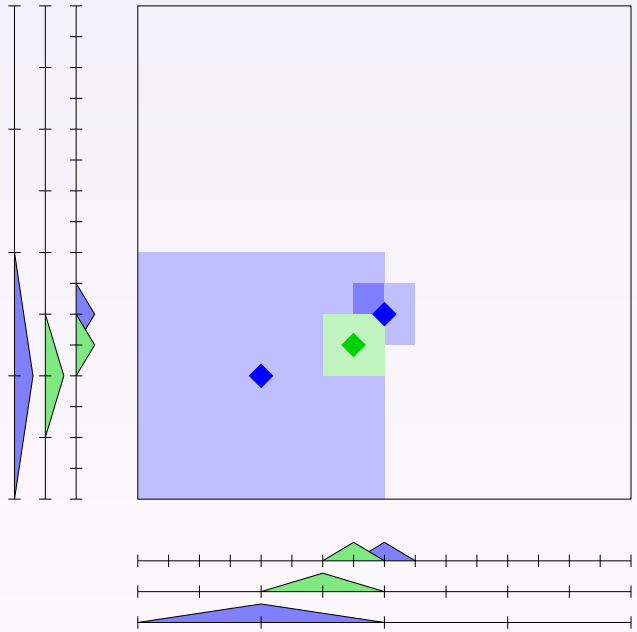
Goal: Find all pairs $(t, s), (r, p) \in \mathcal{G}$ with

$$(\tau \times \sigma) \cap (\varrho \times \pi) \neq \emptyset.$$

Challenge: (t, s) and (r, p) may correspond to different levels, but we only know the neighbours within the same level.

Idea: If $\operatorname{level}(t, s) < \operatorname{level}(r, p)$, check descendants t^* and s^* on the level of (r, p).

Important: Consider only descendants that are neighbours of the ancestors of r and p.



Matrix entries

Goal: Compute matrix coefficients

$$a_D(\varphi_t p_{t,\nu}, \varphi_r p_{r,\mu})$$

efficiently even if $\operatorname{level}(t) \neq \operatorname{level}(r)$.

Idea: If $\operatorname{level}(t) < \operatorname{level}(r)$, use nested structure

$$\varphi_t = \sum_{t' \in S_t} \varphi_t(\xi_{t'}) \varphi_{t'}, \qquad p_{t,\nu} = \sum_{\nu'} c_{t't,\nu'\nu} p_{t',\nu'}$$

to obtain

$$a_D(\varphi_t p_{t,\nu}, \varphi_r p_{r,\mu}) = \sum_{t' \in S_t} \sum_{\nu'} \varphi_t(\xi_{t'})\, c_{t't,\nu'\nu}\, a_D(\varphi_{t'} p_{t',\nu'}, \varphi_r p_{r,\mu}).$$

$\rightarrow$ Recursive reduction to the case $\operatorname{level}(t') = \operatorname{level}(r)$.

Improvements

Variable order: Use polynomials of order $m_\ell := c(L - \ell)$ on level ℓ, use nested interpolation higher-order polynomials.
$\rightarrow$ On the finest level, we only need entries of A.

Adaptivity: We can refine and unrefine the hierarchical partition of unity by adding and subtracting coefficients of nodal basis functions.

Parallelization: Should work essentially as in standard FE codes.

Conclusion

Goal: Take advantage of local regularity of C_u to reduce the complexity.

Approach: Hierarchical tensor finite element method (HITFEM).

- Use standard FE hierarchy in D to construct hierarchical partition of unity for $D \times D$,
- set up trial space by multiplying with polynomials,
- build system matrix using only standard grid-transfer and neighbourhood information.

Result: $\mathcal{O}(n)$ operations for setup, $\mathcal{O}(n)$ storage.

Challenge: Efficient solver for the resulting system.