# Wavelet Galerkin BEM on unstructured meshes by aggregation*

G. Schmidlin and C. Schwab

Research Report No. 2000-15
December 2000

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

*This work was supported in part under the TMR network "Multiscale Methods in Numerical Analysis" of the EC by the Swiss government under grant number BBW 97.0404

# Wavelet Galerkin BEM on unstructured meshes by aggregation[*]

G. Schmidlin and C. Schwab

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

Research Report No. 2000-15 December 2000

**Abstract**

We investigate the numerical solution of strongly elliptic boundary integral equations on unstructured surface meshes $\Gamma$ in $\mathbb{R}^3$ by Wavelet-Galerkin boundary element methods (BEM). They allow complexity-reduction for matrix setup and solution from quadratic to polylogarithmic (i.e. from $O(N^2)$ to $O(N(\log N)^a)$ for some small $a \geq 0$, see, e.g. [2,3,9,10] and the references there). We introduce an agglomeration algorithm to coarsen arbitrary surface triangulations on boundaries $\Gamma$ with possibly complicated topology and to construct stable wavelet bases on the coarsened triangulations in linear complexity. We describe an algorithm to generate the BEM stiffness matrix in standard form in polylogarithmic complexity. The compression achieved by the agglomerated wavelet basis appears robust with respect to the complexity of $\Gamma$. We present here only the main results and ideas - full details will be reported elsewhere.

[*]This work was supported in part under the TMR network "Multiscale Methods in Numerical Analysis" of the EC by the Swiss government under grant number BBW 97.0404

## 1 Introduction

### 1.1 Problem formulation

We consider the numerical solution of the boundary integral equation

$$\mathcal{A}u = f \quad \text{on } \Gamma. \tag{1.1}$$

Here $\Gamma$ is the boundary of a bounded Lipschitz domain $\Omega \in \mathbb{R}^d$ $(d = 2, 3)$, $f \in L^2(\Gamma)$ and $\mathcal{A}$ a boundary integral operator of order 0. We assume that $\mathcal{A}$ is injective and strongly elliptic in $L^2(\Gamma)$ in the sense that it satisfies a Gårding inequality. Then, under these conditions, Galerkin discretizations of (1.1) based on a dense sequence of subspaces of $L^2(\Gamma)$ converge quasioptimally.

Galerkin discretizations of (1.1) lead to fully populated stiffness matrices due to the nonlocal operator $\mathcal{A}$ entailing $O(N^2)$ complexity. It was shown in [9,10] and the references there that this complexity can be reduced to polylogarithmic order by the use of suitable wavelet bases on $\Gamma$ and by compression of the stiffness matrix, i.e. by dropping most of the $O(N^2)$ entries in the wavelet stiffness matrix. In [9,10], certain compression strategies were shown to preserve the optimal asymptotic rate of convergence. The analysis of [9,10] indicated, however, a strong dependence of the constants in the compression estimates on the geometry of the surface and required a nested sequence of meshes on $\Gamma$ in an essential way. Numerical experiments confirmed that this was not an artifact of the analysis, but that indeed the performance of these wavelet algorithms deteriorates on complex surfaces. In practice, the assumption of a nested sequence of triangulations on $\Gamma$ is unrealistic, since for complex surfaces the coarsest possible mesh already contains a large number of degrees of freedom. The present paper presents algorithms which coarsen arbitrary triangulations on surfaces $\Gamma$ with possibly complicated topology and then construct stable wavelet bases on the coarsened triangulations in linear complexity. We sketch an algorithm which directly generates the compressed stiffness matrix in standard form in polylogarithmic complexity.

### 1.2 Boundary Element Method

We illustrate our approach for the simplest boundary elements: we triangulate $\Gamma$ into a quasiuniform mesh $\mathcal{T}_N(\Gamma)$ of $N$ panels $\{\pi_i\}_{i=1}^N$ with

$$\Gamma = \overline{\bigcup_{1 \le i \le N} \pi_i} \quad \text{and} \quad \pi_i \cap \pi_j = \emptyset, \quad i \neq j. \tag{1.2}$$

To each panel $\pi$ we assign the radius $r_\pi$ and the center $c_\pi$

$$(r_\pi, c_\pi) = \inf\{(r, c) : \pi \subset B_r(c), \quad r, c \in \mathbb{R}^d\} \tag{1.3}$$

where $B_r(c)$ is the ball with radius $r$ and center $c$. With

$$h := \max\{r_\pi : \pi \in \mathcal{T}_N(\Gamma)\} \tag{1.4}$$

the uniformity condition of the panels reads

$$r_\pi > \frac{h}{C_u} \quad \forall \pi \in \mathcal{T}_N \tag{1.5}$$

with a constant $C_u$. We further assume that there is a constant $C_\mu$ such that the part of $\mathcal{T}_N$ in $B_r(c)$ satisfies

$$|B_r(c) \cap \mathcal{T}_N| \leq C_\mu r^{d-1} \quad \forall r \geq 0, c \in \mathbb{R}^d. \tag{1.6}$$

Throughout, $|S|$ denotes the surface measure of the panels in the subset $S \subset \mathcal{T}_N$.

We consider the Galerkin Boundary Elements based on piecewise constant boundary elements with basis functions given by

$$\phi_i(\underline{x}) = \begin{cases} 1, \, \underline{x} \in \pi_i \\ 0, \text{ else} \end{cases} \qquad i = 1, \ldots, N \tag{1.7}$$

to discretize the integral equation (1.1). This leads to the linear system

$$\underline{\underline{A}}^N \, \underline{u}^N = \underline{b}^N \tag{1.8}$$

for the coefficients of the approximate solution $u^N(\underline{x}) = \sum_{i=1}^N u_i^N \phi_i(\underline{x}) \in V_N$ where $V_N = \operatorname{span}\{\phi_i, i = 1, \ldots, N\} \subset L^2(\mathcal{T}_N)$. Here

$$\underline{\underline{A}}_{ij}^N = \int_\Gamma \phi_i(\underline{x}) c(x) \phi_j(\underline{x}) ds_{\underline{x}} + \int_\Gamma \phi_i(\underline{x}) \int_\Gamma k(\underline{x}, \underline{y}) \phi_j(\underline{y}) ds_{\underline{y}} ds_{\underline{x}} = (\phi_i, \mathcal{A}\phi_j)$$

$$\underline{u}^N = (u_1^N, u_2^N, \ldots, u_N^N)^\top, \quad \underline{b}_i^N = \int_{\pi_i} \phi_i(\underline{x}) f(\underline{x}) ds_{\underline{x}}$$

and the integrals in the definition of $(\phi_i, \mathcal{A}\phi_j)$ must be understood, in general, as Cauchy principal values. We assume that the kernel $k(\underline{x}, \underline{y})$ satisfies the estimates

$$\left| D_{\underline{x}}^\alpha D_{\underline{y}}^\beta k(\underline{x}, \underline{y}) \right| \leq \frac{C}{\left|\underline{x} - \underline{y}\right|^{2+|\alpha|+|\beta|}} \qquad |\alpha|, |\beta| \leq 1 \tag{1.9}$$

where $D_{\underline{x}}, D_{\underline{y}}$ denote partial derivatives with respect to the Cartesian coordinates $x, y \in \mathbb{R}^3$ in a fixed open neighborhood of the boundary $\Gamma$. We generalize the wavelet algorithm and the compression analysis of [7,9,10] to general, nonnested triangulations on $\Gamma$.

## 2 Construction of the Wavelet basis

On the possibly unstructured mesh $\mathcal{T}_N$, we construct an agglomerated wavelet basis $\{\psi_{\tau,j}\}$ with vanishing moments of order 1, i.e. the mean value of the wavelets with respect to the surface measure vanishes. It is well known that this implies the smallness of certain entries of the stiffness matrix [7,9,10].

We emphasize, however, that in the compression estimates for the wavelets constructed here the parametric boundary representation and its derivatives do not enter. Only the regularity of the kernel function $k(x, y)$ in the ambient space matters.

To construct the agglomerated wavelet basis, we group the panels $\{\pi\}$ to clusters thereby introducing a hierarchical structure among them. As in [6] we define

**Definition 2.1.** *1. A cluster $\tau$ is defined as a non empty set of panels*

$$\tau = \bigcup_k \pi_{i_k}, \qquad \pi_{i_k} \in \mathcal{T}_N.$$

*Each cluster is assigned its radius $r_\tau$ and its center $c_\tau$*

$$(r_\tau, c_\tau) = \inf\{(r, c) : \tau \subset B_r(c), \;\; r, c \in \mathbb{R}^d\}. \tag{2.1}$$

2. *A cluster pool $P(\mathcal{T}_N) = P$ defines an arbitrary but fixed selected subset of all clusters of $\mathcal{T}_N$ with $\mathcal{T}_N \subset P$ and $\pi_i \subset P$, $\forall \pi_i \in \mathcal{T}_N$.*
3. *A cluster tree $\mathcal{P}(\mathcal{T}_N) = \mathcal{P}$ is a cluster pool $P$ with a hierarchical structure*

$$\forall \tau, \tilde{\tau} \in \mathcal{P} : \quad \tau \cap \tilde{\tau} \in \{\emptyset, \tau, \tilde{\tau}\}.$$

4. *For $\tau \in \mathcal{P}$, with a cluster tree $\mathcal{P}$, we define* $\operatorname{child}(\tau)$ *by the set of all children of the cluster $\tau \in \mathcal{P}$*

$$\operatorname{child}(\tau) := \{\tau' \in \mathcal{P} : \tau' \subset \tau \wedge (\tau' \subset \tau'' \Rightarrow \tau \subset \tau'', \forall \tau'' \in \mathcal{P})\}.$$

5. *The level $L_\mathcal{P} \leq 0$ of a cluster tree $\mathcal{P}$ is defined by*

$$L_\mathcal{P} := - \max_{\tau \in \mathcal{P}} \{l \in \mathbb{N}_0 : \tau \subsetneq \tau_{i_1} \subsetneq \ldots \subsetneq \tau_{i_{l-1}} \subsetneq \tau_{i_l} = \mathcal{T}_N, \tau_{i_1}, \ldots, \tau_{i_l} \in \mathcal{P}\}$$

6. *The level $l_\tau \leq 0$ of a cluster $\tau$ in the cluster tree $\mathcal{P}$ is defined by*

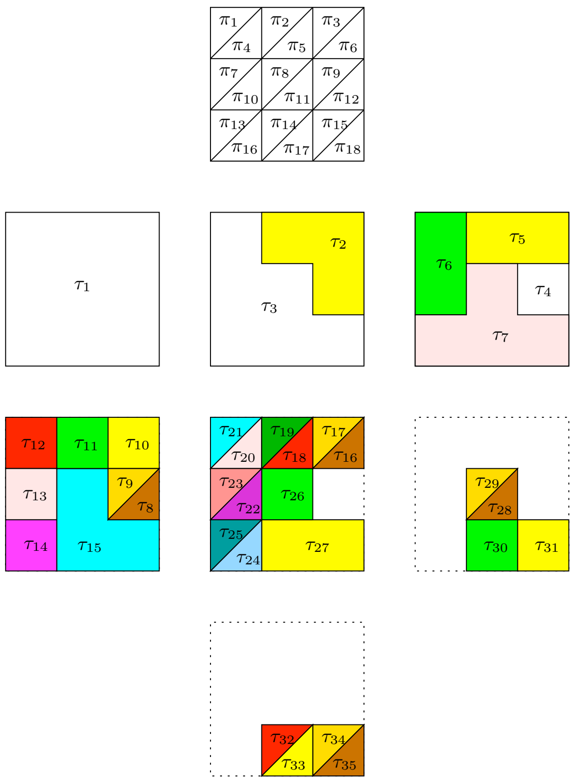
$$l_\tau := L_\mathcal{P} + \max\{l \in \mathbb{N}_0 : \tau \subsetneq \tau_{i_1} \subsetneq \ldots \subsetneq \tau_{i_{l-1}} \subsetneq \tau_{i_l} = \mathcal{T}_N, \tau_{i_1}, \ldots, \tau_{i_l} \in \mathcal{P}\}.$$

7. *The root of the cluster tree $\mathcal{P}$ is the cluster* $\operatorname{root}(\mathcal{P})$ *which fulfills*

$$l_{\operatorname{root}(\mathcal{P})} = L_\mathcal{P}.$$

The example in Fig. 2.1 and 2.2 illustrates these notions. In these figures, $\mathcal{T}_{18} = \{\pi\}_{i=1}^{18}$. The cluster tree $\mathcal{P}$ of Fig. 2.2 consists of the clusters $\{\tau\}_{i=1}^{35}$ shown in the lower part of Fig. 2.1. In this example we have

$$\begin{aligned} L_\mathcal{P} = l_{\tau_1} &= -6 \quad (\tau_1 \text{ is the root of } \mathcal{P}) \\ l_{\tau_2} = l_{\tau_3} &= -5 \\ l_{\tau_4} = l_{\tau_5} = l_{\tau_6} = l_{\tau_7} &= -4 \\ l_{\tau_{32}} = l_{\tau_{33}} = l_{\tau_{34}} = l_{\tau_{35}} &= 0 \\ \operatorname{child}(\tau_3) &= \{\tau_6, \tau_7\}. \end{aligned}$$



$\pi_1$ $\pi_2$ $\pi_3$ $\pi_4$ $\pi_5$ $\pi_6$
$\pi_7$ $\pi_8$ $\pi_9$ $\pi_{10}$ $\pi_{11}$ $\pi_{12}$
$\pi_{13}$ $\pi_{14}$ $\pi_{15}$ $\pi_{16}$ $\pi_{17}$ $\pi_{18}$
$\tau_1$
$\tau_2$ $\tau_3$
$\tau_5$ $\tau_6$ $\tau_4$ $\tau_7$
$\tau_{12}$ $\tau_{11}$ $\tau_{10}$ $\tau_{13}$ $\tau_9$ $\tau_8$ $\tau_{14}$ $\tau_{15}$
$\tau_{21}$ $\tau_{19}$ $\tau_{17}$ $\tau_{20}$ $\tau_{18}$ $\tau_{16}$ $\tau_{23}$ $\tau_{26}$ $\tau_{22}$ $\tau_{25}$ $\tau_{24}$ $\tau_{27}$
$\tau_{29}$ $\tau_{28}$ $\tau_{30}$ $\tau_{31}$
$\tau_{32}$ $\tau_{34}$ $\tau_{33}$ $\tau_{35}$

**Fig. 2.1.** Partition $\mathcal{T}_N = \{\pi_i\}_{i=1}^{18}$ with a cluster tree $\mathcal{P} = \{\tau_i\}_{i=1}^{35}$.

With the hierarchical structure of the mesh given by the cluster tree $\mathcal{P}$ we can construct a wavelet basis of $V_N$. For this construction we need the following conditions. The function $\hat{\mu}$ on $\mathcal{P}$ is defined recursively by

$$\hat{\mu}(\mathcal{T}_N) := 1, \quad \hat{\mu}(\tau') := \frac{\hat{\mu}(\tau)}{|\text{child}(\tau)|} \qquad \text{for } \tau' \in \text{child}(\tau).$$

This means that the division of $\tau \in \mathcal{P}$ into $|\text{child}(\tau)|$ sons should correspond to a division of the area into $|\tau'| \approx |\tau|/|\text{child}(\tau)|$. The precise condition that there are constants $0 < k_l$, $k$ and $C_{\mathcal{P}}$ such that

$$|\text{child}(\tau)| = \begin{cases} 0, & \text{or} \\ k_{l_\tau} \geq 2 & \end{cases} \tag{2.2}$$

$$k_{l_\tau} \leq k \tag{2.3}$$

$$|\mathcal{T}_N| \hat{\mu}(\tau) \leq C_{\mathcal{P}} \tag{2.4}$$

$$r_\tau \leq C_{\mathcal{P}} r_{\text{root}(\mathcal{P})} (\hat{\mu}(\tau))^{1/(d-1)} \tag{2.5}$$

for all $\tau \in \mathcal{P}$.

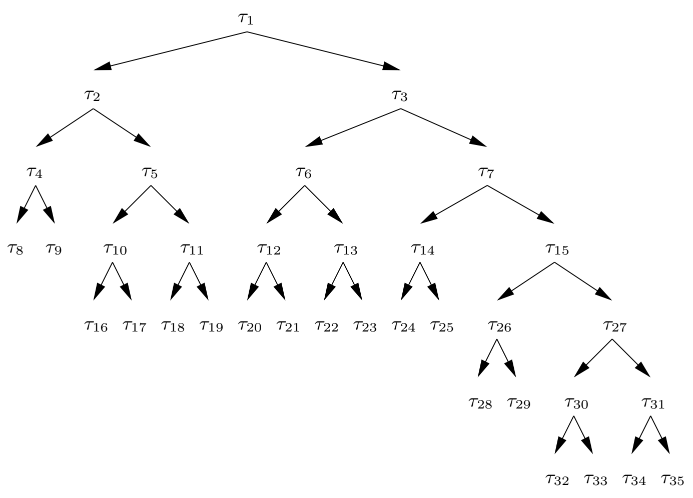


**Fig. 2.2.** Cluster tree $\mathcal{P}$ corresponding to Fig. 2.1.

**Remark 2.2.** Note that the conditions (1.2), (1.6) and (2.3)-(2.5) are analogous to those in [5] for the multipole method. They are not really a restriction for reasonable surfaces $\Gamma$. Condition (2.2) is not a restriction on the surface but depends on the type of tree used in the implementation.

The wavelet basis $\{\psi_{\tau,j}\}$ is constructed from the scaling functions $\{\phi_i\}_{i=1}^N$ in (1.7) by the following agglomeration procedure. In a first step, we compute for each $\tau \in \mathcal{P}$ with $\mathrm{child}(\tau) = \emptyset$ the piecewise constant orthonormal basis functions $\phi_\tau$ $((\phi_\tau, \phi_{\tau'}) = \delta_{\tau\tau'})$

$$\phi_\tau(\underline{x}) = \begin{cases} 1/\sqrt{|\tau|}, & \underline{x} \in \tau \\ 0, & \text{else} \end{cases} \tag{2.6}$$

with $|\tau| = \int_\tau ds_{\underline{x}}$. Afterwards for each cluster $\tau$ with $\mathrm{child}(\tau) \neq \emptyset$ the wavelets $\psi_\tau = \{\psi_{\tau,j}\}_{j=1}^{|\mathrm{child}(\tau)|-1}$ and the coarsened scaling function $\phi_\tau$ are computed by the local wavelet transformation $\underline{\underline{M}}_\tau$. The constant normed function $\phi_\tau$ with $\mathrm{supp}(\phi_\tau) = \tau$ is needed to get the wavelets on the next lower level. At the end of this recursive bottom up, the wavelet basis consists of the functions

$$\phi_{\mathrm{root}(\mathcal{P})} \text{ and } \{\psi_{\tau,j}\}_{(\tau,j)\in\mathcal{J}} \tag{2.7}$$

where $\mathcal{J} = \{(\tau, j) : \tau \in \mathcal{P} \wedge \mathrm{child}(\tau) \neq \emptyset,\ 1 \le j < |\mathrm{child}(\tau)|\}$. The construction of the agglomerated wavelet basis is realized by Algorithm 2.3.

**Algorithm 2.3.**

```
  for $l = 0$ down to $L_{\mathcal{P}}$ {
      for all $\tau \in \mathcal{P}$ with $l_\tau = l$ {
        if child($\tau$) = $\emptyset$ {
          compute $\phi_\tau$;
        } else {
          compute $\begin{pmatrix}\phi_\tau\\ \psi_\tau\end{pmatrix} = \underline{\underline{M}}_\tau^\top \begin{pmatrix}\phi_{\tau_{i_1}}\\ \vdots\\ \phi_{\tau_{i_{|\mathrm{child}(\tau)|}}}\end{pmatrix}$, $\tau_{i_\lambda} \in \mathrm{child}(\tau)$
        }
      }
    }
```

## 3 Local transformation $\underline{\underline{M}}_\tau$

For each tree $\mathcal{P}(\mathcal{T})$ which satisfies the conditions 2.2 and 2.3 we can construct the agglomerated wavelet basis of the space $V_N$ of piecewise constants by applying Algorithm 2.3 and the local transformations $\underline{\underline{M}}_\tau$.

The local transformation $\underline{\underline{M}}_\tau$ converts the scaling functions $\phi_{\tau_{i_\lambda}}$ with $\tau_{i_\lambda} \in \mathrm{child}(\tau)$ into the local wavelet basis $\psi_\tau = \{\psi_{\tau,j}\}_{j=1}^{|\mathrm{child}(\tau)|-1}$ and the coarsened scaling function $\phi_\tau$:

$$\begin{pmatrix}\phi_\tau\\ \psi_\tau\end{pmatrix} = \underline{\underline{M}}_\tau^\top \begin{pmatrix}\phi_{\tau_{i_1}}\\ \vdots\\ \phi_{\tau_{i_{|\mathrm{child}(\tau)|}}}\end{pmatrix}. \tag{3.1}$$

To get an orthonormal wavelet basis of $V_N$ the local transformation matrix $\underline{\underline{M}}_\tau = (\underline{m}_1^\tau, \ldots, \underline{m}_{\mathrm{child}(\tau)}^\tau)$ has to satisfy

$$\underline{\underline{M}}_\tau^\top \underline{\underline{M}}_\tau = I\,. \tag{3.2}$$

For each cluster $\tau$ in the tree $\mathcal{P}$ we compute the local transformation matrix $\underline{\underline{M}}_\tau$ by defining $\underline{\underline{\tilde{M}}}_\tau = (\underline{m}_1^\tau,\ \underline{\tilde{m}}_2^\tau, \ldots, \underline{\tilde{m}}_{|\mathrm{child}(\tau)|}^\tau)$ with

$$(\underline{m}_1^\tau)^\top = \frac{1}{\sqrt{|\tau|}}(\sqrt{|\tau_{i_1}|}, \ldots, \sqrt{|\tau_{i_{|\mathrm{child}(\tau)|}}|}), \quad \tau_{i_\lambda} \in \mathrm{child}(\tau)$$

and $\underline{\tilde{m}}_2^\tau, \ldots, \underline{\tilde{m}}_{|\mathrm{child}(\tau)|}^\tau$ such that

$$\det(\underline{\underline{\tilde{M}}}_\tau) \neq 0, \quad (\underline{m}_1^\tau)^\top \underline{\tilde{m}}_i^\tau = 0, \quad 2 \leq i \leq |\mathrm{child}(\tau)|.$$

Applying the singular value decomposition (SVD) to the vectors $\underline{\tilde{m}}_2^\tau, \ldots, \underline{\tilde{m}}_{|\mathrm{child}(\tau)|}^\tau$, we get the matrix $\underline{\underline{M}}_\tau$

$$(\underline{\underline{U}}_\tau, \underline{\underline{S}}_\tau, \underline{\underline{V}}_\tau) = \mathrm{SVD}((\underline{\tilde{m}}_2^\tau, \ldots, \underline{\tilde{m}}_{|\mathrm{child}(\tau)|}^\tau)) \tag{3.3}$$

$$\underline{\underline{M}}_\tau = (\underline{m}_1^\tau,\ \underline{\underline{U}}_\tau) \tag{3.4}$$

which satisfies (3.2).

**Proposition 3.1.** *With the local transformation matrix $\underline{\underline{M}}_\tau$ defined above, we get a local basis $\phi_\tau$, $\psi_\tau$ which is orthonormal with respect to the surface measure on $\Gamma$, i.e.*

$$\begin{aligned} \|\phi_\tau\|_{L^2(\Gamma)} &= 1 && && (3.5)\\ (\phi_\tau, \psi_{\tau,j}) &= 0 && 1 \le j < |\text{child}(\tau)| && (3.6)\\ (\psi_{\tau,j}, \psi_{\tau,j'}) &= \delta_{jj'} && 1 \le j, j' < |\text{child}(\tau)|. && (3.7) \end{aligned}$$

In our implementation, we use the particular choice of $\underline{\underline{\tilde{M}}}_\tau$ given in Assertion 3.2.

**Assertion 3.2.** *The matrix*

$$\underline{\underline{\tilde{M}}}_\tau = \begin{bmatrix} \tilde{m}_1^\tau & -\tilde{m}_2^\tau & 0 & 0 & \cdots & 0 \\ \tilde{m}_2^\tau & \tilde{m}_1^\tau & -\tilde{m}_3^\tau & 0 & \dots & 0 \\ \tilde{m}_3^\tau & 0 & \tilde{m}_2^\tau & -\tilde{m}_4^\tau & & \vdots \\ \vdots & \vdots & & \ddots & \ddots & 0 \\ \tilde{m}_{n-1}^\tau & 0 & & & \tilde{m}_{n-2}^\tau & -\tilde{m}_n^\tau \\ \tilde{m}_n^\tau & 0 & & \cdots & 0 & \tilde{m}_{n-1}^\tau \end{bmatrix} \qquad (3.8)$$

*with $\tau \in \mathcal{P}$, $\tau_{i_\lambda} \in \text{child}(\tau)$, $\tilde{m}_\lambda^\tau = \sqrt{|\tau_{i_\lambda}|/|\tau|}$ and $n = |\text{child}(\tau)|$ has full rank.*

**Remark 3.3.** The local transformation can be seen as an application of the lifting scheme presented in [11,12] or the stable completion in [1,2]. We start with the scaling function $\phi_\tau$ and the wavelets

$$\tilde{\psi}_{\tau,j} = (\underline{\tilde{m}}_{j+1}^\tau)^\top \begin{pmatrix} \phi_{\tau_{i_1}} \\ \vdots \\ \phi_{\tau_{i_{|\text{child}(\tau)|}}} \end{pmatrix} \qquad 1 \le j < |\text{child}(\tau)|$$

defined by the matrix $\underline{\underline{\tilde{M}}}_\tau$ and $\{\phi_{\tau_\lambda}\}$. These wavelets are lifted to an orthonormal basis on $\Gamma$ with vanishing mean value with respect to the surface measure $ds$ on $\Gamma$. This is done using a singular value decomposition.

**Remark 3.4.** In case of a binary tree, the local transformation matrix $\underline{\underline{M}}_\tau$ is

$$\underline{\underline{M}}_\tau = \begin{bmatrix} \sqrt{|\tau_1|/|\tau|} & -\sqrt{|\tau_2|/|\tau|} \\ \sqrt{|\tau_2|/|\tau|} & \sqrt{|\tau_1|/|\tau|} \end{bmatrix}$$

for $\tau_1, \tau_2 \in \text{child}(\tau)$. Then the singular value decomposition can be omitted.

**Remark 3.5.** Note that the support of the wavelets is not necessarily connected. Therefore, we can collect the surface patches in a certain region of $\mathbb{R}^d$ in a cluster independent of the boundary $\Gamma$. This leads to a compression rate of the stiffness matrix which is not spoiled by the geometry.

## 4 Properties of the agglomerated wavelet basis

For any triangulation $\mathcal{T}_N$ on $\Gamma$, Algorithm 2.3 generates an agglomerated wavelet basis. The forward and backward transformation of the pyramid scheme for vectors $\underline{f}^{\phi}$ and $\underline{f}^{\psi}$ are standard and omitted, since they are almost identical to Algorithm 2.3. The agglomerated wavelet basis has the desirable properties:

**Proposition 4.1.** *The wavelet basis obtained from Algorithm 2.3 is an $L^2(\Gamma, ds)$ orthonormal basis of $V_N$.*

*a) The wavelets fulfill the vanishing moment property*

$$\int_{\Gamma} \psi_{\tau,j}(\underline{x}) ds_{\underline{x}} = 0 \qquad \text{for } (\tau, j) \in \mathcal{J} \tag{4.1}$$

*with $\mathcal{J}$ defined as in (2.7).*

*b) The agglomerated wavelet basis satisfies Paseval's equation in $L^2(\Gamma, ds)$.*

$$\|u\|^2_{L^2(\Gamma)} = (u, \phi_{\mathrm{root}(\mathcal{P})})^2 + \sum_{(\tau,j)\in\mathcal{J}} (u, \psi_{\tau,j})^2 \tag{4.2}$$

*for all $u \in V_N$.*

*c) With $W_l = \mathrm{span}\{\psi_{\tau,j} : l_\tau = l-1, 1 \leq j < |\mathrm{child}(\tau)|\}$, $l > L_{\mathcal{P}}$, it holds*

$$V_N = W_{L_{\mathcal{P}}} \oplus W_{L_{\mathcal{P}}+1} \oplus \ldots \oplus W_0 \tag{4.3}$$

*where we have set $W_{L_{\mathcal{P}}} = \mathrm{span}\{\phi_{\mathrm{root}(\mathcal{P})}\}$*

Equation (4.2) is a consequence of the clusters on the same level being disjoint and of the orthogonality of the wavelets between different levels. The orthogonality between different levels follows from (3.6).

**Remark 4.2.** If we refine $\mathcal{T}_N$ beyond level $l = 0$ by regular subdivision of each $\pi_i \in \mathcal{T}_N$, the corresponding spaces of piecewise constants are dense in $L^2(\Gamma, ds)$ and the finite sum in Parseval's equality (4.2) can be extended to an infinite one as for example in [7,10].

## 5 Examples

We give examples of the agglomerated wavelets and exhibit some special properties of them.

In the first example, we computed some wavelet coefficients of the function $f(\underline{x}) = (1 - x_1)(1 - x_2)$ on the unit square. As shown in Fig. 5.1, we chose a uniform triangulation with $N = 200$ triangles. We effect the agglomeration of the panels by a binary tree (other types of trees are also possible). We construct the tree recursively by computing the Čebyšev center and radius of

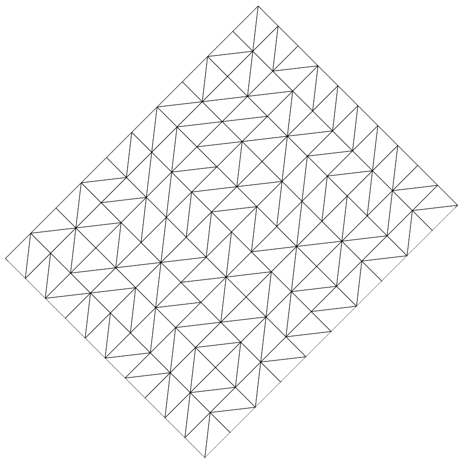


**Fig. 5.1.** Unit square with uniform, nonhierarchical mesh – 200 triangles.

a set of the panels and a dyadic subdivision of this panel set. An algorithm to compute the Čebyšev center and radius efficiently can be found in [6]. The recursion starts with the whole geometry and stops when only two panels belong to the panel set of the node. To each node in the binary tree we can assign a node of the cluster tree $\mathcal{P}$. In Fig. 5.2 the approximation $f_N$ of $f$ is plotted. Figure 5.3 shows the different levels of the wavelets. In the left picture the approximation of $f$ is restricted to the wavelets on level $L_{\mathcal{P}} + 4$ on the right hand side the level $L_{\mathcal{P}} + 5$ is plotted.

The second example is the surface shown in Fig. 5.4 left. The agglomeration is done as in the previous example. The clusters are assigned to Čebyšev balls of the binary tree instead of Čebyšev circles of the 2d example. A single wavelet is shown in Fig. 5.5 – one clearly sees that the support consists of two connected parts, since with the binary tree the panels are aggregated in space and not on the surface anymore. The agglomerated wavelets are not restricted to hierarchical meshes anymore. In addition, our scheme generates wavelets with supports of "small" diameter. This pays in the matrix compression.

## 6 Complexity of the transformation

In this section the complexity of Algorithm 2.3 is estimated. In a first step of the algorithm we generate the cluster tree $\mathcal{P}$. With the assumptions in section 1 and 2 on the boundary $\Gamma$, we see that we can compute a cluster tree $\mathcal{P}$ with $L_{\mathcal{P}} = O(\log N)$ (see [5]). To insert the leafs $\pi_i$ into the cluster tree we need for each leaf at most $L_{\mathcal{P}} = O(\log N)$ operations. Therefore, we can compute the cluster tree $\mathcal{P}$ for $N$ leafs in $O(N \log N)$ operations. In [6] a binary tree is presented which fulfills the conditions (2.2) and (2.3). With this cluster tree we apply Algorithm 2.3.

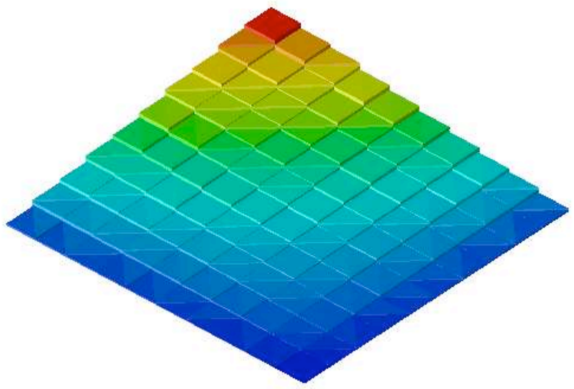


**Fig. 5.2.** Function $f_N(\underline{x}) = \sum_{i=1}^{N} f_i \phi_i(\underline{x})$ with $N = 200$ on the unit square

**Proposition 6.1.** *Algorithm 2.3 takes $W_{tot} = O(N)$ operations.*

***Proof*** Each leaf in the cluster tree $\mathcal{P}$ corresponds to a degree of freedom in the one scale basis $\{\phi\}_{i=1}^{N}$. Therefore we have

$$|\{\tau \in \mathcal{P} : \text{child}(\tau) = \emptyset\}| = N. \tag{6.1}$$

Using (2.2), we find that each cluster $\tau$ with child$(\tau) \neq \emptyset$ corresponds to at least one degree of freedom in the wavelet basis. This leads to

$$|\{\tau \in \mathcal{P} : \text{child}(\tau) \neq \emptyset\}| \leq N. \tag{6.2}$$

With

$$|\{\tau \in \mathcal{P}\}| = |\{\tau \in \mathcal{P} : \text{child}(\tau) = \emptyset\}| + |\{\tau \in \mathcal{P} : \text{child}(\tau) \neq \emptyset\}|$$

and the equations (6.1), (6.2) we find

$$|\{\tau \in \mathcal{P}\}| \leq 2N. \tag{6.3}$$

The work for the local transformations in each cluster is bounded by $Ck^3$ where $C$ is independent of $k$ and $N$. The term $k^3$ stems from the singular value decomposition of the $|\text{child}(\tau)| \times |\text{child}(\tau)|$ local transformation matrix $\underline{\underline{\tilde{M}}}_{\tau}$ (see [4]) where $|\text{child}(\tau)|$ is estimated by $k$ in (2.3). With equation (6.3) we get

$$W_{\text{loc}} \leq Ck^3 \Rightarrow W_{\text{tot}} \leq Ck^3 \cdot |\{\tau \in \mathcal{P}\}| \;\leq\; Ck^3 2N.$$

$\square$

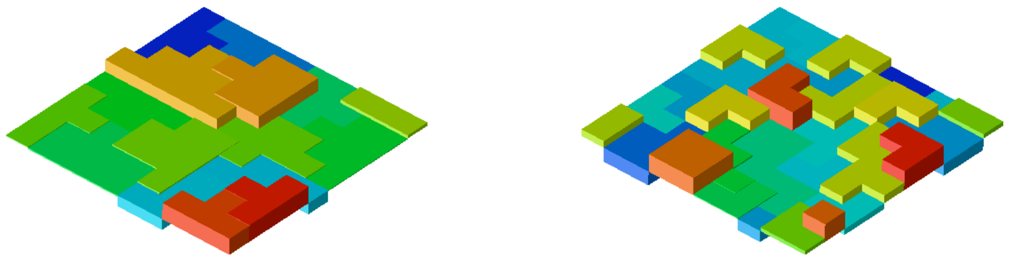


**Fig. 5.3.** Contribution to the wavelet representation of the function $f$ by the wavelets on level $l$. Left: $l = L_{\mathcal{P}} + 4$ and Right: $l = L_{\mathcal{P}} + 5$.

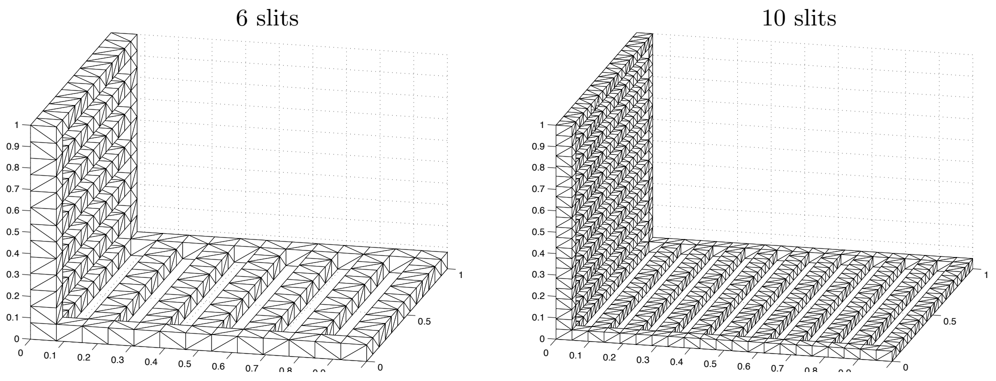


**Fig. 5.4.** L-shaped domains with slits.

We illustrate in Fig. 6.1 the result of Proposition 6.1. The forward and backward transformation is applied to a vector in the one scale basis. The diagram shows the cpu-time for the pyramid scheme for the unit-square problem and for the L-shaped domain with slits. The thickness of the L-shaped domain and the slits is always one triangle. Therefore an increase of the degrees of freedom in the L-shaped domain problem induces also more slits and changes consequently the geometry. This leads to a series of domains where in Fig. 5.4 the domains with 6 ($N = 1500$) and 10 slits ($N = 3772$) are plotted. The increase in the number of degrees of freedom in the square is due to the subdivision of the domain into smaller triangles. We see that the cpu-time of the algorithm for the unit-square problem and for the L-shaped domain with slits is almost equal and behaves like $O(N)$ as stated in Proposition 6.1.

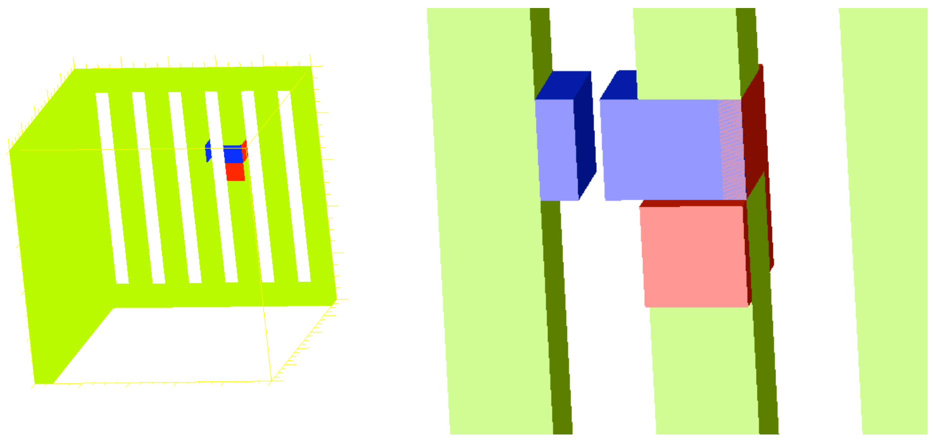


**Fig. 5.5.** Wavelet with non connected support.

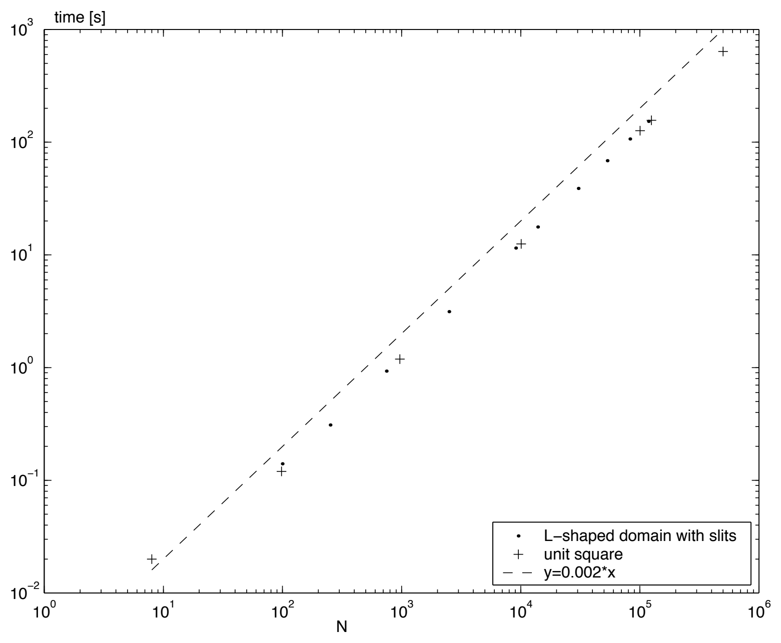


**Fig. 6.1.** cpu-time to compute the cluster tree and to apply the pyramid scheme to a vector versus the number of degrees of freedom. In the diagram we compare the L-shaped domains with slits (Fig. 5.4) and the unit square problem (Fig. 5.1).

## 7 Evaluation of the compressed stiffness matrix

In the preceding sections we discussed an algorithm to construct the agglomerated wavelet basis. In this section an algorithm is derived to compress the stiffness matrix using the new wavelets. The algorithm to locate the nonzero entries is essentially the same as presented in [7] but the computation of the entries is different. Contrary to [7], we do not require a hierarchical mesh and the computation of entries in the compressed stiffness matrix is more difficult, since through the aggregation process the shape of the clusters is a priori unknown and no standard integration formula can be applied. A naive splitting of the cluster into panels leads to a $O(N^2)$ algorithm. We overcome this difficulty by approximating the compressed stiffness matrix entries by a multipole-type algorithm. We directly obtain the standard form of the compressed stiffness matrix in contrast to [13] where the non-standard form is computed.

### 7.1 Localization of entries in the compressed stiffness matrix

To describe our localization algorithm for the compressed stiffness matrix, we begin by presenting some needed definitions. Analogously to (1.8) and (1.9), we write the linear equation system in the wavelet basis as

$$\underline{\underline{A}}^L \underline{u}^L = \underline{b}^L \tag{7.1}$$

with

$$\begin{aligned} \underline{\underline{A}}^L_{(\tau,j)(\tau',j')} &= \int_\tau \psi_{\tau,j}(\underline{x})(\mathcal{A}\psi_{\tau',j'})(\underline{x})ds_{\underline{x}} \\ \underline{u}^L &= (u_{\mathrm{root}(\mathcal{P})}, u^L_{\tau_1,1}, \dots, u^L_{\tau_t,|\mathrm{child}(\tau_t)-1|}) \\ \underline{b}^N_{\tau,j} &= \int_\tau \psi_{\tau,j}(\underline{x})f(\underline{x})ds_{\underline{x}} \end{aligned} \tag{7.2}$$

where $t = |\{\tau \in \mathcal{P} : \mathrm{child}(\tau) \neq \emptyset\}|$. The matrix $\underline{\underline{A}}^L$ can be subdivided into the subblocks $\underline{\underline{A}}^L_{ll'}$ which represents all entries $\underline{\underline{A}}^L_{(\tau,j)(\tau',j')}$, $1 \le j < |\mathrm{child}(\tau)|$, $1 \le j' < |\mathrm{child}(\tau')|$ with $l_\tau = l$ and $l_{\tau'} = l'$. For each blockmatix $\underline{\underline{A}}^L_{ll'}$ there exists a truncation parameter $\delta_{ll'}$ at our disposal. Setting

$$\delta_{ll'} \ge \max\{a \prod_{\mu=L_\mathcal{P}}^{0} k_\mu^{-\frac{1}{2}} \prod_{\mu=l}^{0} k_\mu^{\frac{\alpha}{2}} \prod_{\mu=l'}^{0} k_\mu^{\frac{\tilde{\alpha}}{2}}, \prod_{\mu=L_\mathcal{P}}^{l-1} k_\mu^{-\frac{1}{2}}, \prod_{\mu=L_\mathcal{P}}^{l'-1} k_\mu^{-\frac{1}{2}}\} \tag{7.3}$$

for $a > 1$, $s, \tilde{s} \in [0,1]$ and $\alpha \ge \frac{s+1}{2}$, $\tilde{\alpha} \ge \frac{\tilde{s}+1}{2}$ we approximate the matrix $\underline{\underline{A}}^L$ by the sparse one

$$\tilde{\underline{\underline{A}}}^L_{ll',(\tau,j)(\tau',j')} := \begin{cases} \underline{\underline{A}}^L_{ll',(\tau,j)(\tau',j')} & \text{if } \mathrm{dist}(\tau,\tau') \le \delta_{ll'} \\ 0 & \text{otherwise} \end{cases}. \tag{7.4}$$

Note that $\delta_{ll'}$ corresponds to the truncation parameter in [10].

**Proposition 7.1.** *The compressed wavelet Galerkin matrix $\underline{\underline{\tilde{A}}}^L$ has*

$$\mathcal{N} = \begin{cases} O(N\log^2(N)) & \text{if } \alpha = \tilde{\alpha} = 1 \\ O(N\log(N)) & \text{otherwise} \end{cases}$$

*nonzero entries (see also [10]).*

To compute the entries efficiently, we need an algorithm which locates these entries in $O(\mathcal{N})$ operations.

**Remark 7.2.** Later on we will show that once we have located the entries that we will be able to compute them in $O(\log^3(N))$ operations.

The observation

$$\delta_{ll'} \geq \delta_{\tilde{l}\tilde{l}'}, \quad \text{for } l \leq \tilde{l}, l' \leq \tilde{l}'.$$

leads to a fast algorithm because with this we can skip the whole subtree $\tilde{\tau}' \subset \tau'$ if $\operatorname{dist}(\tau, \tau') > \delta_{l_\tau l_{\tau'}}$. In Algorithm 7.4 we also use the symmetry of the matrix $(\delta_{ll'})$ which leads to a symmetric sparsity pattern of $\underline{\underline{\tilde{A}}}^L$.

To define the location algorithm we need a lexicographical order of the clusters. Because of the hierarchical structure of the cluster tree $\mathcal{P}$ we can introduce a breadth-first ordering.

**Definition 7.3.** *We denote by $(\tau, \tau')$ the cluster in $\mathcal{P}$ on the highest possible level such that it contains $\tau$ and $\tau'$.*

$$\tau, \tau' \subset (\tau, \tau') \wedge (\tau, \tau' \subset \hat{\tau} \in \mathcal{P} \Rightarrow (\tau, \tau') \subset \hat{\tau}). \tag{7.5}$$

*A lexicographical order of the clusters $\tau, \tau' \in \mathcal{P}$ can now be defined by*

$$\tau < \tau' \Leftrightarrow l_\tau < l_{\tau'} \text{ or } (l_\tau = l_{\tau'} \text{ and } \lambda < \lambda') \tag{7.6}$$

*for $\tau \subset \hat{\tau}_{i_\lambda}$, $\tau' \subset \hat{\tau}_{i_{\lambda'}}$ and $\hat{\tau}_{i_\lambda}, \hat{\tau}_{i_{\lambda'}} \in \operatorname{child}((\tau, \tau'))$, $\lambda, \lambda' \in \{1, \ldots, |\operatorname{child}((\tau, \tau'))|\}$.*

With this definition we get for example for the cluster tree in Fig. 2.2 the ordering

$$\begin{gathered}\tau_1 < \tau_2 < \tau_3 < \tau_4 < \tau_5 < \tau_6 < \tau_7 < \tau_8 < \tau_9 < \tau_{10} < \tau_{11} < \tau_{12} < \tau_{13} \\ \tau_{13} < \tau_{14} < \tau_{15} < \tau_{16} < \tau_{17} < \tau_{18} < \tau_{19} < \tau_{20} < \tau_{21} < \tau_{22} < \tau_{23} < \tau_{24} \\ \tau_{24} < \tau_{25} < \tau_{26} < \tau_{27} < \tau_{28} < \tau_{29} < \tau_{30} < \tau_{31} < \tau_{32} < \tau_{33} < \tau_{34} < \tau_{35}\end{gathered}$$

With the lexicographical order of the clusters the algorithm to locate the nonzero entries is [7]:

**Algorithm 7.4.**

```
assemble($\tau, \tau'$) {
  if ($\operatorname{dist}(\tau, \tau') \leq \delta_{l_\tau l_{\tau'}}$) and ($\tau' < \tau$ or $\tau = \tau'$) {
    evaluate $E^\psi_{\tau\tau'}$, $E^\psi_{\tau'\tau}$ and update $\underline{\underline{\tilde{A}}}^L$

    for all $\tilde{\tau}' \in \operatorname{child}(\tau')$ with $\operatorname{child}(\tilde{\tau}') \neq \emptyset$
      assemble($\tau, \tilde{\tau}'$)
  }
}
```

with the calling sequence

```
for all $\tau \in \{\tau' \in \mathcal{P} : \text{child}(\tau') \neq \emptyset\}$
    assemble$(\tau, \text{root}(\mathcal{P}))$
```

The matrices $E^{\psi}_{\tau\tau'}$ and $E^{\psi}_{\tau'\tau}$ denote the element stiffness matrices in the wavelet basis.

Because of the symmetric sparsity pattern of $\underline{\underline{\tilde{A}}}^L$ we have to call *assemble* at most $\mathcal{N} + N < 2\mathcal{N}$ times (at most $\mathcal{N}$ recursive calls and $N$ calls from the calling sequence). Without computing the matrices $E^{\psi}_{\tau\tau'}$ and $E^{\psi}_{\tau'\tau}$ each call of *assemble* takes at most $O(k)$ operations (condition (2.3)). This leads to

**Assertion 7.5.** *Algorithm 7.4 locates the nonzero entries of* $\underline{\underline{\tilde{A}}}^L$ *in* $O(\mathcal{N})$ *operations.*

## 7.2 Computation of the entries in the compressed stiffness matrix

To get polylogarithmic complexity $O(N \log^5(N)) = O(\mathcal{N})O(\log^3(N))$ for the matrix setup, we have to compute any entry in the stiffness matrix $\underline{\underline{\tilde{A}}}^L$ in $O(\log^3(N))$ work. This is done with the panel clustering method with $\eta \in (0,1)$. The nearfield and farfield of a cluster $\tau \in \mathcal{P}$ are defined by

$$\mathcal{N}_\eta(\tau) = \{\tau' \in \mathcal{P} : r_\tau + r_{\tau'} \geq \eta \|c_\tau - c_{\tau'}\|\} \tag{7.7}$$

$$\mathcal{F}_\eta(\tau) = \{\tau' \in \mathcal{P} : r_\tau + r_{\tau'} < \eta \|c_\tau - c_{\tau'}\|\}. \tag{7.8}$$

We assume that all integrations can be executed exactly. That means that there is no error in the evaluation of the multipole coefficients and in the interaction between two panels (this assumption is generally not valid in practice but with an appropriate numerical integration scheme as presented in [10] the error becomes sufficiently small while preserving the polylogarithmic work-estimates).

The algorithm to compute the compressed stiffness matrix can be subdivided into four steps:

**Step 1** The multipole coefficients $\Phi^\tau_p$, $0 \leq p \leq m$, of order $m$ in $c_\tau$ are computed for each leaf of the cluster tree $\mathcal{P}$ introduced in section 2.

**Algorithm 7.6.**

```
for all $\{\tau \in \mathcal{P} : \text{child}(\tau) = \emptyset\}$
    compute $\Phi^\tau_p$, $0 \leq p \leq m$
```

**Step 2** For each cluster $\tau$ which is not a leaf compute the multipole coefficients $\Phi^\tau_p$, $0 \leq p \leq m$, of order $m$ in $c_\tau$ by shifting the coefficients of its children to the cluster center and multiplication with the local transformation matrix.

**Algorithm 7.7.**

```
for $l = 0$ down to $L_{\mathcal{P}}$
  for all $\{\tau \in \mathcal{P} : \mathrm{child}(\tau) \neq \emptyset$ and $l_\tau = l\}$
    for all $\tau_{\lambda_j} \in \mathrm{child}(\tau)$, $j = 1, \dots, |\mathrm{child}(\tau)|$
      shift the multipole coefficients
      from $c_{\tau_j}$ to $c_\tau$ ($\Phi_p^{\tau_j} \to (\hat{\underline{\Phi}}_p^\tau)_j$)
    $\Phi_p^\tau = (\underline{m}_1^\tau)^\top \hat{\underline{\Phi}}_p^\tau$
```

**Step 3** Compute the nearfield entries of the one scale matrix on each level bottom up. To simplify notation we use for $\tau, \tau' \in \mathcal{P}$, $\tau_{i_\lambda} \in \mathrm{child}(\tau)$, $1 \leq \lambda \leq |\mathrm{child}(\tau)|$ and $\tau'_{j_{\lambda'}} \in \mathrm{child}(\tau')$, $1 \leq \lambda' \leq |\mathrm{child}(\tau')|$ the abbreviations

$$\begin{aligned}
\underline{\underline{a}}_{\phi_\tau,\phi_{\tau'}} &= \left( \int_{\tau_{i_\lambda}} \phi_{\tau_{i_\lambda}} (\mathcal{A}\phi_{\tau'_{j_{\lambda'}}}) ds \right)_{\tau_{i_\lambda} \tau'_{j_{\lambda'}}} \\
\underline{\underline{a}}^{\mathcal{N}}_{\phi_\tau,\phi_{\tau'}} &= \left( \int_{\tau_{i_\lambda}} \phi_{\tau_{i_\lambda}} (\mathcal{A}\phi_{\tau'_{j_{\lambda'}}}) ds \right)_{\tau_{i_\lambda} \tau'_{j_{\lambda'}}} && \tau'_{j_{\lambda'}} \in \mathcal{N}_\eta(\tau_{i_\lambda}) \\
\underline{\underline{a}}^{\mathcal{F}}_{\phi_\tau,\phi_{\tau'}} &= \left( \int_{\tau_{i_\lambda}} \phi_{\tau_{i_\lambda}} (\mathcal{A}\phi_{\tau'_{j_{\lambda'}}}) ds \right)_{\tau_{i_\lambda} \tau'_{j_{\lambda'}}} && \tau'_{j_{\lambda'}} \in \mathcal{F}_\eta(\tau_{i_\lambda}) \\
a_{\phi_\tau,\phi_{\tau'}} &= \int_\tau \phi_\tau (\mathcal{A}\phi_{\tau'}) ds \\
\underline{\underline{MPE}}^{\mathcal{F}}_{\tau,\tau'} &= \left( \sum_{p,p'=0}^{m} \Phi_p^{\tau_{i_\lambda}} F_{pp'}^{\tau_{i_\lambda} \tau'_{j_{\lambda'}}} \Phi_{p'}^{\tau'_{j_{\lambda'}}} ) \right)_{\tau_{i_\lambda} \tau'_{j_{\lambda'}}} && \tau'_{j_{\lambda'}} \in \mathcal{F}_\eta(\tau_{i_\lambda})
\end{aligned}$$

where the matrices are expanded with zeros up to a dimension of $|\mathrm{child}(\tau)| \times |\mathrm{child}(\tau')|$. $F_{pp'}^{\tau_{i_\lambda} \tau'_{j_{\lambda'}}}$ denotes the $(p, p')$ derivative of the kernel function evaluated at the centers $c_{\tau_{i_\lambda}}$ and $c_{\tau'_{j_{\lambda'}}}$ and $\Phi_p^{\tau_{i_\lambda}}$ is the $p$-th multipole moment with respect to the cluster center $c_{\tau_{i_\lambda}}$.
With $l_\tau = l_{\tau'}$ we can write the entries of the one scale matrix on each level by the following formula

$$\begin{aligned}
a_{\phi_\tau,\phi_{\tau'}} &= (\underline{m}_1^\tau)^\top \underline{\underline{a}}_{\phi_\tau,\phi_{\tau'}} \underline{m}_1^{\tau'} \\
&= (\underline{m}_1^\tau)^\top \underline{\underline{a}}^{\mathcal{N}}_{\phi_\tau,\phi_{\tau'}} \underline{m}_1^{\tau'} + (\underline{m}_1^\tau)^\top \underline{\underline{a}}^{\mathcal{F}}_{\phi_\tau,\phi_{\tau'}} \underline{m}_1^{\tau'} \\
&\approx (\underline{m}_1^\tau)^\top \underline{\underline{a}}^{\mathcal{N}}_{\phi_\tau,\phi_{\tau'}} \underline{m}_1^{\tau'} + (\underline{m}_1^\tau)^\top \underline{\underline{MPE}}^{\mathcal{F}}_{\tau,\tau'} \underline{m}_1^{\tau'}
\end{aligned}$$

The algorithm for step three looks like

**Algorithm 7.8.**

```
for $l = 0$ down to $L_{\mathcal{P}}$
  for all $\{\tau \in \mathcal{P} : l_\tau = l\}$
    for all $\tau' \in \mathcal{N}_\eta(\tau)$ and $l_{\tau'} = l_\tau$ {
      $a_{\phi_\tau,\phi_{\tau'}} = (\underline{m}_1^\tau)^\top \underline{\underline{a}}^{\mathcal{N}}_{\phi_\tau,\phi_{\tau'}} \underline{m}_1^{\tau'} + (\underline{m}_1^\tau)^\top \underline{\underline{MPE}}^{\mathcal{F}}_{\tau,\tau'} \underline{m}_1^{\tau'}$
```

**Step 4** For $\tau, \tau' \in \mathcal{P}$ and $l_\tau \geq l_{\tau'}$ compute the approximate element matrix $\underline{\underline{E}}^{\psi}_{\tau\tau'}$. To describe the algorithm we need in addition to the abbreviations of step 3 the notation

$$\underline{\underline{a}}_{\psi_\tau,\psi_{\tau'}} = \left(\int_{\tau_i} \psi_{\tau,i}(\mathcal{A}\psi_{\tau',j})ds\right)_{i,j}$$
$$\underline{\underline{\hat{M}}}_\tau = (\underline{m}_2^\tau, \ldots, \underline{m}^\tau_{|\mathrm{child}(\tau)|})$$

with $1 \leq i < |\mathrm{child}(\tau)|$ and $1 \leq j < |\mathrm{child}(\tau')|$. The element matrices can be approximated by the formula

$$\begin{aligned}
\underline{\underline{E}}^{\psi}_{\tau\tau'} = \underline{\underline{a}}_{\psi_\tau,\psi_{\tau'}} &= \underline{\underline{\hat{M}}}_\tau^\top \underline{\underline{a}}_{\phi_\tau,\phi_{\tau'}} \underline{\underline{\hat{M}}}_{\tau'} \\
&= \underline{\underline{\hat{M}}}_\tau^\top \underline{\underline{a}}^{\mathcal{N}}_{\phi_\tau,\phi_{\tau'}} \underline{\underline{\hat{M}}}_{\tau'} + \underline{\underline{\hat{M}}}_\tau^\top \underline{\underline{a}}^{\mathcal{F}}_{\phi_\tau,\phi_{\tau'}} \underline{\underline{\hat{M}}}_{\tau'} \\
&\approx \underline{\underline{\hat{M}}}_\tau^\top \underline{\underline{a}}^{\mathcal{N}}_{\phi_\tau,\phi_{\tau'}} \underline{\underline{\hat{M}}}_{\tau'} + \underline{\underline{\hat{M}}}_\tau^\top \underline{\underline{MPE}}^{\mathcal{F}}_{\tau,\tau'} \underline{\underline{\hat{M}}}_{\tau'}
\end{aligned} \tag{7.9}$$

The term $\underline{\underline{\hat{M}}}_\tau^\top \underline{\underline{a}}^{\mathcal{N}}_{\phi_\tau,\phi_{\tau'}} \underline{\underline{\hat{M}}}_{\tau'}$ is split recursively in a sum over the nearfield and the farfield of the children cluster of $\tau'$. This splitting is done up to level $l_\tau$. On level $l_\tau$ the nearfield entries were computed in step three and the recursion ends.
The algorithm of step four consists of two parts. The first part computes the element matrices $\underline{\underline{E}}^{\psi}_{\tau\tau'}$ and $\underline{\underline{E}}^{\psi}_{\tau'\tau}$ needed by Algorithm 7.4. The second part is the recursive function *elementMatrix* which performs equation (7.9).

**Algorithm 7.9.**

```
$\underline{\underline{E}}^{\psi}_{\tau\tau'} = \underline{\underline{\hat{M}}}_\tau^\top$ elementMatrix($\tau$, $\tau'$)$\underline{\underline{\hat{M}}}_{\tau'}$
$\underline{\underline{E}}^{\psi}_{\tau'\tau} = \underline{\underline{\hat{M}}}_{\tau'}^\top$ elementMatrix($\tau'$, $\tau$)$\underline{\underline{\hat{M}}}_{\tau}$
```

**Algorithm 7.10.**

```
$\underline{\underline{E}} \leftarrow$ elementMatrix($\tau$, $\tau'$) {
  $\underline{\underline{E}} = \underline{\underline{MPE}}^{\mathcal{F}}_{\tau,\tau'}$
  if $l_\tau > l_{\tau'}$
    for all $\{\tau'_{i_\lambda} \in \mathrm{child}(\tau') : \tau'_{i_\lambda} \in \mathcal{N}_\eta(\hat{\tau}), \hat{\tau} \in \mathrm{child}(\tau),$
            $1 \leq \lambda \leq |\mathrm{child}(\tau')|\}$
      $\underline{\underline{E}} = \underline{\underline{E}} +$ elementMatrix($\tau$, $\tau'_{i_\lambda}$)$\underline{m}_1^{\tau'_{i_\lambda}} \underline{e}_\lambda^\top$
  else if $l_\tau < l_{\tau'}$
    for all $\{\tau_{i_\lambda} \in \mathrm{child}(\tau) : \tau_{i_\lambda} \in \mathcal{N}_\eta(\hat{\tau}'), \hat{\tau}' \in \mathrm{child}(\tau'),$
            $1 \leq \lambda \leq |\mathrm{child}(\tau)|\}$
      $\underline{\underline{E}} = \underline{\underline{E}} + \underline{e}_\lambda(\underline{m}_1^{\tau_{i_\lambda}})^\top$ elementMatrix($\tau_{i_\lambda}$, $\tau'$)
  else
    $\underline{\underline{E}} = \underline{\underline{E}} + \underline{\underline{a}}^{\mathcal{N}}_{\phi_\tau,\phi_{\tau'}}$
}
```

$\underline{e}_\lambda$ denotes the $\lambda$'th unit vector $(0,\dots,0,1,0,\dots,0)^\top$.

## 8 Numerical Experiments

In this section we present a numerical experiment obtained with the described implementation of the multiscale scheme (see Section 7). In a series of L-shaped domains with slits (see Section 6 and Fig. 5.4) we consider the interior Dirichlet Problem

$$\begin{aligned} \Delta U &= 0 \qquad \text{on } \Omega \\ U &= f \qquad \text{on } \Gamma \end{aligned} \tag{8.1}$$

where $f$ is the function

$$f(\underline{x}) = \frac{1}{\sqrt{(x_1-2)^2+(x_2-2)^2+(x_3-2)^2}} \ .$$

This problem was solved with the double layer ansatz $U(\underline{x}) = \int_\Gamma k(\underline{x},\underline{y})u(\underline{y})ds_{\underline{y}}$ with

$$k(\underline{x},\underline{y}) = \frac{1}{4\pi}\frac{\underline{n}(\underline{y})\cdot(\underline{y}-\underline{x})}{|\underline{y}-\underline{x}|^3} \ . \tag{8.2}$$

where $k$ satisfies equation (1.9). We measured the cpu-time to compute the compressed stiffness matrix. The result was obtained on a single processor (*SUN UltraSPARC-II*, 336 MHz) with 1 GB RAM using MATLAB and the *g++ 2.95* Compiler for some C routines. In Fig. 8.1 we plotted the cpu-time versus the number of unknowns for the parameters $\eta = 0.5$, $m = 6$, $a = 1$ and $\alpha = \tilde{\alpha} = 0.5$ described in (7.3), (7.7), (7.8) and on page 16. This time includes the construction of the cluster tree, the computation of the multipole expansion and of the matrix entries. The cpu-time grows with $O(N\log(N))$ (lower dashed line) which confirms Proposition 7.1. The upper dashed line corresponds to $O(N\log^2(N))$. This leads to the supposition that our estimations in Section 7.2 were too pessimistic and that the cpu-time increases linearly with the number of non-zero entries.

## 9 Conclusions

In this report we presented an agglomerated Haar wavelet basis based on piecewise constant functions. These wavelets have one vanishing moment and satisfy Parseval's equation on the boundary. With these ingredients we are able to compress the wavelet stiffness matrix. The compression rate with the new wavelets is geometry insensitive because the wavelets are orthogonal to constant functions in $\mathbb{R}^3$ with respect to the surface measure.

In the second part we introduced an algorithm to compute directly the compressed wavelet stiffness matrix in the standard form for complex geometries. The algorithm is a combination of the panel clustering and the standard wavelet algorithm.

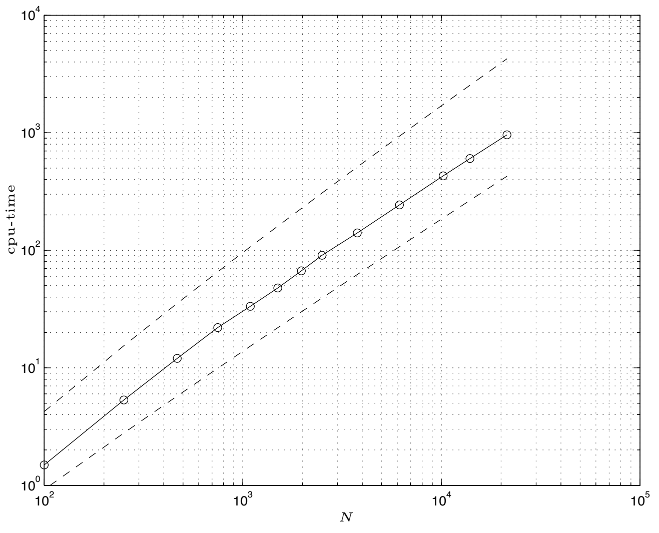


**Fig. 8.1.** Generating the compressed stiffness matrix in standard form for L-shaped domains with slits: cpu-time versus number of unknowns.

## References

1. J.M. Carnicer, W. Dahmen, J.M. Peña. *Local decomposition of refinable spaces and wavelets*, Applied and Computational Harmonic Analysis, 3, pp. 127-153, 1996.
2. W. Dahmen. *Wavelet and Multiscale Methods for Operator Equations* Acta Numerica, pp. 55-228, 1997
3. W. Dahmen, S. Prössdorf, R. Scheider. *Wavelet approximation methods for pseudodifferential equations II: Matrix compression and fast solution* Adv. Comput. Math. 1, pp. 259-335, 1993
4. G.H. Golub, C.F. van Loan. *Matrix Computations*, The Johns Hopkins University Press, 1989
5. W. Hackbusch, Z. P. Nowak. *On the Fast Matrix Multiplication in the Boundary Element Method by Panel Clustering*, Numer. Math. 54, pp. 463-491, 1989.
6. Ch. Lage. *Softwareentwicklung zur Randelementmethode: Analyse und Entwurf effizienter Techniken*, Dissertation at Christian-Albrechts-Universität Kiel, 1995.
7. Ch. Lage, Ch. Schwab. *Wavelet Galerkin Algorithms for Boundary Integral Equations*, SIAM J. Sci. Comput., Vol 20, No. 6, pp. 2195-2222, 1999.
8. P. Oswald. *Multilevel Finite Element Approximation: Theory and Applications* Teubner Skripte zur Numerik, B. G. Teubner Stuttgart, 1994.
9. R. Schneider. *Multiskalen- und Wavelet-Matrixkompression* Advances in Numerical Mathematics, B.G. Teubner Stuttgart, 1998.
10. T. von Petersdorff, Ch. Schwab. *Fully Discrete Multiscale Galerkin BEM*, Wavelet Analysis and its Application, Volume 6 (287-346), Academic Press 1997.

11. W. Sweldens. *The Lifting Scheme: A Custom-Design Construction of Biorthogonal Wavelets*, Appl. Comput. Harmon. Anal., Vol. 3, 2, pp. 186-200, 1996.
12. W. Sweldens. *The Lifting Scheme: A Construction of Second Generation Wavelets*, SIAM J. Math. Anal., Vol. 29, 2, pp. 511-546, 1998.
13. J. Tausch, J White *Multiscale bases for the sparse representation of boundary integral operators on complex geometry*, SMU Math Report 2000-01