

On the Implementation of a Fully Discrete Multiscale Galerkin BEM

Ch. Lage, Ch. Schwab

Seminar für Angewandte Mathematik, ETH-Zürich

CH-8092 Zürich, Switzerland

Abstract

We present a boundary element method on general polyhedral surfaces in space. The shape functions are piecewise constant multiwavelets. The $N \times N$ stiffness matrix is numerically sparse, i.e. only $O(N(\log N)^2)$ entries with a-priori known locations need to be computed. Numerical results for $> 10^5$ unknowns are presented.

1 Model Problem and BIE Formulation

Let $\Omega \subset \mathbb{R}^3$ be a bounded polyhedron with N_0 straight faces Γ_k and boundary $\Gamma = \partial\Omega$. In Ω , consider the problem

$$\Delta U = 0 \text{ in } \Omega, \quad U = f \text{ on } \Gamma. \quad (1)$$

We look for $U(x)$ in the form of a double layer potential, i.e.

$$U(x) = \int_{\Gamma} \frac{n(y) \cdot (x - y)}{4\pi|x - y|^3} u(y) dS(y) =: \int_{\Gamma} K(x, y) u(y) dS(y) \quad (2)$$

with $n(y)$ the exterior unit normal at $y \in \Gamma$, and obtain the BIE

$$-\frac{1}{2}u(x) + \int_{\Gamma} K(x, y) u(y) dS(y) = f(x) \quad (3)$$

for any point $x \in \Gamma$ that is not on an edge or a vertex.

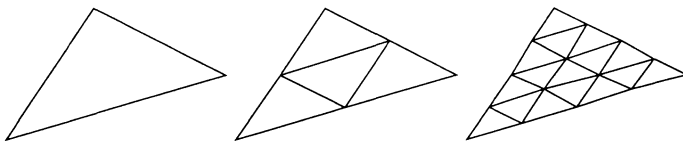


Figure 1: Sequence $\{\mathcal{M}_k^l\}_{l=0}^2$ of meshes on Γ_k .

2 Galerkin BEM

We assume each side Γ_k of Γ to be triangular (this can always be achieved). We generate a sequence of meshes $\{\mathcal{M}_k^l\}_{l \leq L}$ on Γ_k by successively halving the sides of each triangle $T \in \mathcal{M}_k^l$ (Figure 1). The corresponding mesh on Γ is $\mathcal{M}_l = \bigcup_{k=1}^{N_0} \mathcal{M}_k^l$ with $N_l = N_0 4^l$ triangles T_j^l , $j = 0, \dots, N_l - 1$, on the surface.

Similar to the classical panel method, we look for an approximate solution

$$u^L = \sum_{j=0}^{N_L-1} \{u_\varphi^L\}_j \varphi_j^L \quad (4)$$

where φ_j^L is one on triangle $T_j^l \in \mathcal{M}_l$, and zero elsewhere. Inserting (4) into (3) and collocating at the barycenters leads to the panel method. Here we use instead the Galerkin method where we multiply the BIE (3) by test functions φ_i^L and integrate over Γ :

$$\begin{aligned} \int_{\Gamma} \varphi_i^L(x) \left(-\frac{1}{2} u^L(x) + \int_{\Gamma} K(x, y) u^L(y) dS(y) \right) dS(x) \\ = \int_{\Gamma} \varphi_i^L(x) f(x) dS(x) \end{aligned} \quad (5)$$

or, in matrix formulation,

$$\left(-\frac{1}{2} [M_\varphi^L] + [K_\varphi^L] \right) \{u_\varphi^L\} = \{f_\varphi^L\} \quad (6)$$

where mass- and stiffness matrix are given by

$$[M_\varphi^L] = \text{diag} \left\{ \int_{\Gamma} (\varphi_i^L(x))^2 dS(x) \right\}, \quad (7)$$

$$[K_\varphi^L] = \left\{ \int_{\Gamma} \varphi_i^L(x) \int_{\Gamma} K(x, y) \varphi_j^L(y) dS(y) dS(x) \right\}_{0 \leq i, j < N_L} \quad (8)$$

and $\{u_\varphi^L\}$, $\{f_\varphi^L\}$ denote the solution- and load vector, respectively.

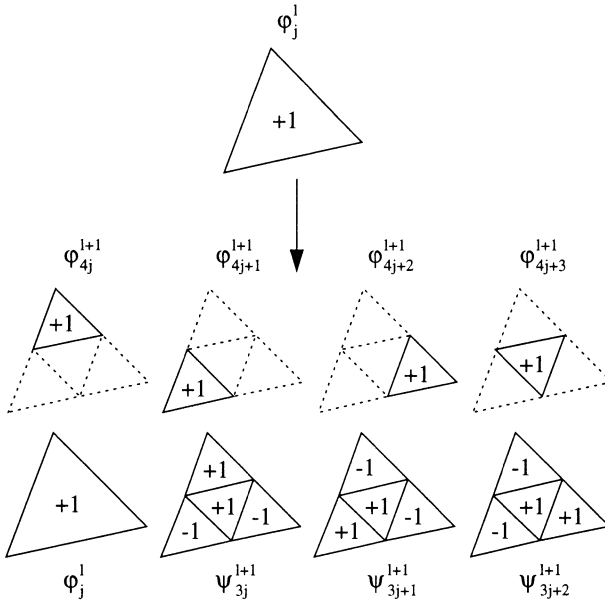


Figure 2: Old and new refinement.

3 Wavelet Basis

The idea is now to pick a set special of shape functions $\{\psi_j^l\}_{l \leq L}$ for which most entries in the corresponding stiffness matrix can be neglected. Rather than taking all degrees of freedom (DOF) on the finest mesh $\mathcal{M}_L$, we now keep some DOF on the coarser meshes and only add the DOF which give new information (Figure 2). We observe that the new scheme leads to three new shape functions which are discontinuous, piecewise constant and have *vanishing average*. Clearly the new functions are equivalent to the old ones, i.e. using the basis $\{\psi_j^l\}_{l \leq L}$ instead of $\{\varphi_j^L\}$ does not change the BEM solution:

$$u^L = \sum_{j=0}^{N_L-1} \{u_\varphi^L\}_j \varphi_j^L = \sum_{j=0}^{N_0-1} \{u_\psi^0\}_j \psi_j^0 + \sum_{l=0}^{L-1} \sum_{j=0}^{3N_l-1} \{u_\psi^{l+1}\}_j \psi_j^{l+1}. \quad (9)$$

We may easily pass from the coefficient vector $\{u_\psi^l\}_{l \leq L}$ to $\{u_\varphi^L\}$ by the pyramid scheme:

$$\begin{array}{ccccccc} \{u_\varphi^0\} & \xrightarrow{[H_0]} & \{u_\varphi^1\} & \xrightarrow{[H_1]} & \cdots & \longrightarrow & \{u_\varphi^{L-1}\} & \xrightarrow{[H_{L-1}]} & \{u_\varphi^L\} \\ \{u_\psi^1\} & \nearrow [G_0] & \{u_\psi^2\} & \nearrow [G_1] & & & \{u_\psi^L\} & \nearrow [G_{L-1}] & \end{array},$$

i.e. $\{u_\varphi^{l+1}\} = [H_l] \{u_\varphi^l\} + [G_l] \{u_\psi^{l+1}\}$ with block diagonal matrices of the form

$$\begin{aligned} [H_l] &= \text{blockdiag} \left\{ \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \right\} \in \mathbb{R}^{N_{l+1} \times N_l}, \\ [G_l] &= \text{blockdiag} \left\{ \begin{pmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \right\} \in \mathbb{R}^{N_{l+1} \times 3N_l}. \end{aligned}$$

The main point in using the wavelet basis is that the stiffness matrix $[K_\psi^L]$ can be “sparsified”: Let $S_j^l = \{x \in \Gamma : \psi_j^l(x) \neq 0\}$ be the support of wavelet ψ_j^l . Then define the sparse matrix $[\tilde{K}_\psi^L]$ by

$$[\tilde{K}_\psi^L]_{(l,j)(l',j')} = \begin{cases} [K_\psi^L]_{(l,j)(l',j')} & \text{if } \text{dist}(S_j^l, S_{j'}^{l'}) \leq a 2^{L-l-l'}, \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

and determine the solution $\{\tilde{u}_\psi^L\}$ of the system

$$\left(-\frac{1}{2} [M_\psi^L] + [\tilde{K}_\psi^L] \right) \{\tilde{u}_\psi^L\} = \{f_\psi^L\}. \quad (11)$$

Note that $[M_\psi^L]$ is diagonal due to the $L^2(\Gamma)$ orthogonality of the ψ_j^l . We scale the basis $\{\psi_j^l\}_{l \leq L}$ such that $[M_\psi^L]$ is the identity matrix. It is known [5][4][1] that the accuracy of the solution $\{\tilde{u}_\psi^L\}$ is comparable to that of $\{u_\varphi^L\}$ in (6), but that the *number of nonzeros* in the compressed matrix $[\tilde{K}_\psi^L]$ is $\leq CN_L(\log N_L)^2$.

4 Numerical Integration

In general, it is not possible to evaluate the entries of the stiffness matrix exactly such that approximations must be used. For example Gaussian quadrature formulas converge for analytic integrands, as it is well-known, exponentially with a rate depending on the size of the integrand’s domain of analyticity. The only difficulty that arises is that for given $\psi_j^l, \psi_{j'}^{l'}$ the domain of analyticity of $K(x, y)\psi_j^l(x)\psi_{j'}^{l'}(y)$ and consequently the rate of convergence of Gaussian quadrature applied to (8) may be very small if $\text{dist}(S_j^l, S_{j'}^{l'}) < \max \{\text{diam}(S_j^l), \text{diam}(S_{j'}^{l'})\}$ such that the quadrature cannot be performed in logarithmic complexity. This can be prevented by subdivision of the larger panel (Figure 3) using the given sequence of meshes as proposed in [5]. Now the integrals over each combination of the smaller panel with a panel of the subdivision can be handled by means of Gaussian quadrature of variable order depending on the level and the distance

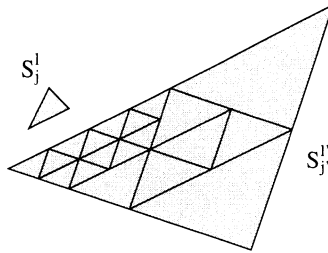


Figure 3: Subdivision of the larger panel.

of the panels. Moreover, due to the subdivision the singular cases, i.e. $\text{dist}(S_j^l, S_{j'}^{l'}) = 0$, are reduced to three basic cases: identical panels, panels sharing a common edge or panels sharing a vertex. For these cases efficient quadrature rules are available [6][3].

It turns out that the total cost of the assembly of the compressed stiffness matrix $[\tilde{K}_\psi^L]$ is not greater than $O(N_L(\log N_L)^3)$ kernel evaluations [3].

5 Implementation

For the implementation of our method we used the object oriented framework for boundary element methods described in [2]. This framework provides the mathematical concepts of boundary elements such as subspaces, dualforms, operators or functions, in terms of classes. Two major extensions of the framework are necessary to cover the demands of our wavelet discretization: a specialization of the base class `DualForm` endowed with the variable order, composite quadrature rule of the previous Section and a specialization of the base class `Operator` to assemble and store the compressed stiffness matrix.

For the sake of brevity, we only discuss the algorithmic aspects of the underlying classes and omit their precise definition.

5.1 Quadrature Algorithm

The assembly of the stiffness matrix imposes the evaluation of the element matrices $E_{TT'}^\psi$, $T, T' \in \bigcup_{l=0}^{L-1} \mathcal{M}_l$. We manage this task by calculating the corresponding 4×4 element matrices related to the standard basis functions $E_{TT'}^\varphi$ using the variable order, composite quadrature rule of the previous Section and applying a transformation $E_{TT'}^\varphi \mapsto E_{TT'}^\psi$ afterwards. Algorithm 1 illustrates the recursive implementation of the subdivision. The initial call $\text{integrate}(T, T')$ evaluates the matrix $E_{TT'}^\varphi$ as well as its symmetric counter-

640 Boundary Elements

```

integrate( $T, T'$ ) {
  let  $l, l' \in \mathbb{N}_0$  such that  $T \in \mathcal{M}_l$  and  $T' \in \mathcal{M}_{l'}$ 
  if  $\text{dist}(T, T') \geq 2^{-l'}$  or  $l = l'$ 
    evaluate the element matrices  $E_{TT'}^\varphi$  and  $E_{T'T}^\varphi$ 
    using Gaussian quadrature
  else
    for every  $\tilde{T} \subset T'$  with  $\tilde{T} \in \mathcal{M}_{l'+1}$ 
      update  $E_{TT'}^\varphi, E_{T'T}^\varphi$  with  $\text{integrate}(T, \tilde{T})$ 
    return  $E_{TT'}^\varphi$  and  $E_{T'T}^\varphi$ 
}

```

Algorithm 1: Composite quadrature rule, $l \geq l'$.

part $E_{T'T}^\varphi$ simultaneously. In this way the construction of the subdivision which is identical in both cases is fully exploited. In the interests of efficiency temporary matrices generated during the subdivision on higher levels are cached, such that subsequent calls can profit. It turns out that a cache size of $O(\log N_L)$ together with a special calling sequence is sufficient to guarantee an optimal usage of data.

5.2 Assembly and Compression

For a practical implementation it is essential, however, that the nonzero entries in the compressed stiffness matrix can be localized also in essentially $O(N_L)$ operations, i.e. without scanning the whole N_L^2 entries of the full matrix. This objective is easily achieved if one exploits the tree structure of the sequence of meshes as it is done in the function *assemble()* sketched in Algorithm 2. With this function the compressed stiffness matrix is generated by:

```

for  $T \in \mathcal{M}_0 \cup \dots \cup \mathcal{M}_{L-1}$ 
  for  $T' \in \mathcal{M}_0$  assemble( $T, T'$ )

```

with a complexity proportional to the number of nonzero entries, i.e. $O(N_L (\log N_L)^2)$.

For our compression technique we borrow from the well know encoding algorithm of Ziv, Lempel and Welch which encodes repeated substrings. In our case we only encode sequences of zero entries in the stiffness matrix by their length. Note, that the symmetry with respect to l, l' of the threshold value $a 2^{L-l-l'}$ of the compression implies a symmetric pattern of zero and nonzero entries of the stiffness matrix (Figure 4), such that the simultaneous evaluation of $E_{TT'}^\psi$ and $E_{T'T}^\psi$ in Algorithm 1 is justified. Moreover, the same

```

assemble( $T, T'$ ) {
  let  $l, l' \in \mathbb{N}_0, j, j' \in \mathbb{N}_0$  such that  $T = T_j^l$  and  $T' = T_{j'}^{l'}$ 
  if  $\text{dist}(T, T') \leq a 2^{L-l-l'}$  and  $(l > l' \text{ or } j \geq j')$  {
     $(E_{TT'}^\psi, E_{T'T}^\psi) \leftarrow (E_{TT'}^\varphi, E_{T'T}^\varphi) \leftarrow \text{integrate}(T, T')$ 
    store  $E_{TT'}^\psi, E_{T'T}^\psi$  in compressed form
    if  $l > l'$ 
      for every  $\tilde{T} \subset T'$  with  $\tilde{T} \in \mathcal{M}_{l'+1}$ 
        assemble( $T, \tilde{T}$ )
  }
  return
}

```

Algorithm 2: Assembly and Compression.

pattern of compression can be applied to rows and columns of the stiffness matrix. For a detailed discussion of this topic see [3].

6 Numerical Results

Here we present some preliminary numerical experiments obtained with the described implementation of the multiscale scheme on a SUN Ultra-Enterprise. We solved (1) on several polyhedral domains with quite similar performance. We report here only those from the prism $\Omega = T \times (0, 1)$ where T is an equilateral triangle with sides of length 1. The initial triangulation was the coarsest possible one to cover Γ and contained $N_0 = 8$ triangles. For the right hand side f of (1) we chose the harmonic function $\frac{1}{2} \sin(\frac{\pi}{2}x) \cos(\pi y) \sinh(\frac{\sqrt{5}}{2}\pi z)$.

In Figure 5, we show the CPU time used to generate the sparse stiffness matrix $[\tilde{K}_\psi^L]$; this includes the time to locate the nonzero entries as well as the set-up of the sparse data structure. We see here a $O(N(\log N)^2)$ behaviour showing that the contribution of $O(N(\log N)^3)$ kernel evaluations in the quadrature to the overall complexity is small. We also note that the total memory required to store the compressed matrix at $L = 7$ and $N_L = 131072$ unknowns amounts to 421MB which just fits in the core of our workstation. At this level, the compression rate is, inclusive the overhead for administering the sparse data structure, $3 \cdot 10^{-3}$.

The number of iterations for the iterative solution (GMRes)

level	1	2	3	4	5	6	7
iterations	15	19	20	21	22	22	22

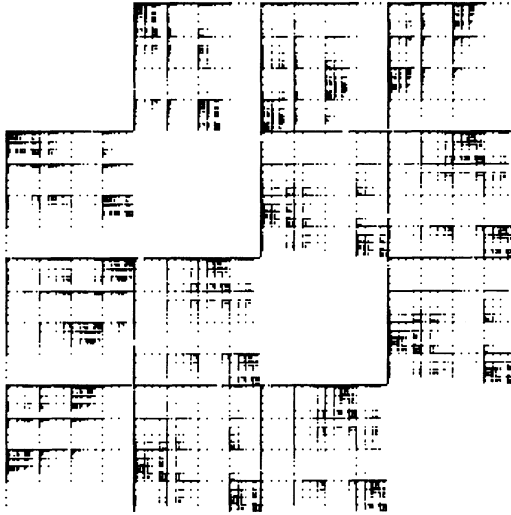


Figure 4: Pattern of zero and nonzero entries of the compressed stiffness matrix, $\Omega = \text{tetrahedron}$, $L = 4$. The wavelets ψ_j^l are ordered according to the preorder depth first traversal of the triangles in Algorithm 2.

is independent of the number of levels validating the bounded condition numbers of the compressed stiffness matrices as predicted in [5]. We point out that the CPU-time for the solution accounts only for about 20 per cent of the total CPU-time. Therefore, with the present method the BEM-paradigm that most of the work is spent for quadrature is still valid and a speed up similar to the one for dense matrices can be achieved with the parallelization of the matrix assembly.

In Figure 6 the error at interior points is depicted. The dashed line shown illustrates the bound $O(N^{-1})$. It can be observed that the expected convergence of essentially $O(h^2)$ is preserved by the compression. The same is true for the convergence of the L^2 -norm of the discrete density u^l (Figure 7) showing the behaviour $\|u^l\|_0 - \|u\|_0 \leq Ch^\kappa$ with $\kappa \sim 1.3$.

References

- [1] W. Dahmen, S. Pröbldorf, and R. Schneider. Wavelet approximation methods for pseudodifferential equations. II: Matrix compression and fast solution. *Advances in Computational Mathematics*, 1:259–335, 1993.
- [2] Ch. Lage. The application of object oriented methods to boundary elements. *Computer Methods in Applied Mechanics and Engineering*,

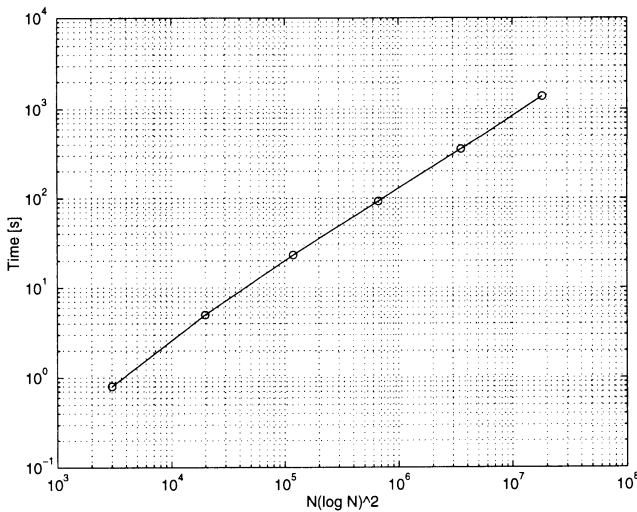


Figure 5: CPU-time for the assembly of the compressed stiffness matrix.

1997. To appear.

- [3] Ch. Lage and Ch. Schwab. A galerkin wavelet algorithm for boundary integral equations. Technical report, Seminar für Angewandte Mathematik, ETH Zürich, CH-8092 Zürich, 1997. To appear.
- [4] T. von Petersdorff, R. Schneider, and Ch. Schwab. Multiwavelets for second kind integral equations. Technical Report BN-1153, IPST, University Maryland College Park, 1994. In press in SINUM.
- [5] T. von Petersdorff and Ch. Schwab. Fully discrete multiscale Galerkin BEM. Technical Report 95-08, Seminar für Angewandte Mathematik, ETH Zürich, CH-8092 Zürich, 1995. To appear in the volume "Multiresolution Analysis and Partial Differential Equations", W. Dahmen, P. Kurdila and P. Oswald (Eds.), in the series "Wavelet Analysis and its Applications", Academic Press (1997).
- [6] S. Sauter. Cubature techniques for 3-D Galerkin BEM. In W. Hackbusch and G. Wittum, editors, *Boundary Elements: Implementation and Analysis of Advanced Algorithms*, volume 54 of *Notes on Numerical Fluid Mechanics*, Braunschweig, Wiesbaden, Germany, 1996. Vieweg Verlag.

644 Boundary Elements

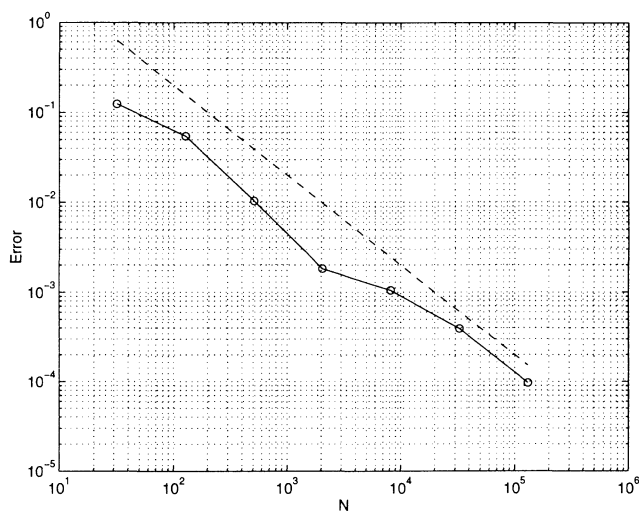


Figure 6: Error at interior points versus N with bound $O(N^{-1})$.

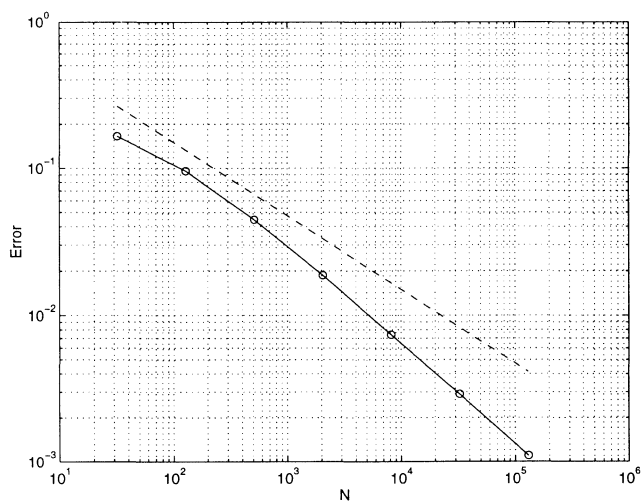


Figure 7: $\|u^l\|_0 - \|u\|_0$ versus N with bound $O(N^{-\frac{1}{2}})$.