BIT manuscript No.
(will be inserted by the editor)

A fast direct solver for elliptic problems on Cartesian meshes in 3D

Phillip G. Schmitz · Lexing Ying

Received: date / Accepted: date

Abstract We present a fast direct algorithm for solutions to linear systems arising from three dimensional elliptic equations. We follow the approach of Xia *et al.* (2009) on combining the multifrontal method with hierarchical matrices in two dimensions and extend it to three dimensional case. Linear time complexity is shown and a more practical variant with worse scaling is demonstrated.

Keywords elliptic equations · fast algorithms · multifrontal methods · hierarchical matrices · sparse matrix

Mathematics Subject Classification (2000) MSC 65F05 · MSC 65F30 · MSC 65F50

1 Introduction

In this paper we will consider the solution of an elliptic problem of form

$$-\mathrm{div}(a(\mathbf{x})\nabla u(\mathbf{x})) + V(\mathbf{x})u(\mathbf{x}) = f(\mathbf{x}) \quad \text{on } \Omega, \qquad u = 0 \quad \text{on } \partial\Omega \tag{1.1}$$

on a Cartesian domain Ω in $\mathbb{R}^3$.

There are two main classes of solvers for sparse linear systems: direct [4] and iterative [20] methods. We will only be concerned with direct methods in this paper. Clearly a sparse Cholesky decomposition should be used instead of the naïve inversion of the sparse matrix. In order to do so efficiently one has to choose a reordering that reduces fill-in of non-zeros in the factors. Finding the optimal ordering (especially for matrices arising from problems in three dimensions) in general is difficult although various graph-theoretic approaches such as the (approximate) minimum degree algorithm [1] or nested dissection [13] can be used to determine a reasonably good reordering.

In three dimensions direct methods such as the multifrontal method [4, 6, 7, 17] have a computational cost that scales like $\mathcal{O}(N^2)$ where N is the number of degrees of freedom, much slower than in two dimensions. Iterative methods are therefore more competitive but sometimes encounter difficulties for problems where $a(\mathbf{x})$ exhibits large contrast (large variations in magnitude for different areas of the domain). However, it has been observed in [18, 23] that fill-in blocks of the LDL^t factorization are highly compressible using the H-matrix [11] or hierarchical semiseparable matrix frameworks [22] and thus the calculations can be done much more efficiently. In two dimensions Xia *et al.* [23] showed how to combine the multifrontal method with hierarchical matrices and we extended this approach in [21] to unstructured and adaptive meshes. This paper will show how to extend this line of work to three dimensional problems while maintaining linear $\mathcal{O}(N)$ complexity.

Phillip G. Schmitz
Department of Mathematics
University of Texas at Austin E-mail: pschmitz@math.utexas.edu

Lexing Ying
Department of Mathematics and ICES
University of Texas at Austin E-mail: lexing@math.utexas.edu

Recently, direct solvers of this type have also been developed for linear systems for boundary integral equations. To name a few examples, in [19] an essentially linear complexity algorithm is presented for the 2D non-oscillatory integral equations, while an $\mathcal{O}(N^{1.5})$ algorithm has appeared recently in [10] for the 3D non-oscillatory case. Fast direct solvers for oscillatory kernels are still unavailable both in 2D and 3D.

The rest of the paper is organized as follows. Section 2 shows the geometric domain decomposition used in this paper. We describe in Section 3 the multifrontal method in three dimensions and in Section 4 the hierarchical matrix algebra that we adopt. The combination of hierarchical matrices and the multifrontal method leads to the algorithm detailed in Section 5. The complexity of our algorithm is discussed in Section 6 for two different approaches to determining the structure of the hierarchical matrices used.

2 Domain Decomposition

Consider the domain $\Omega = [0,1]^3$. We introduce a uniform $(P2^Q+1)\times(P2^Q+1)\times(P2^Q+1)$ Cartesian grid covering $[0,1]^3$, where P is a positive integer of $\mathcal{O}(1)$ and Q will turn out to be the depth of the hierarchical decomposition. Out of the $(P2^Q+1)^3$ nodes in total, some lie on the boundary of the domain and there are $N=(P2^Q-1)^3$ nodes in the interior.

The equation (1.1) is often discretized with a finite difference or a finite element scheme. In the finite element case, each cell of our Cartesian is subdivided into 6 tetrahedra. Based on this tetrahedron mesh, we can define piecewise linear basis functions $\{\phi_\kappa(\mathbf{x})\}$, one for each interior node κ. Each basis function takes value 1 at the node κ and zero at the rest. The discrete version of of (1.1) under this discretization is given by

$$Mu = f,$$

where u and f are vectors formed by the combination of all the u_κ and f_κ, and M is a sparse matrix with values given by

$$(M)_{\kappa\lambda} = \int_{[0,1]^3} \nabla\phi_\kappa(\mathbf{x})\cdot a(\mathbf{x})\nabla\phi_\lambda(\mathbf{x}) + V(\mathbf{x})\phi_\kappa(\mathbf{x})\phi_\lambda(\mathbf{x})\mathrm{d}\mathbf{x}.$$

Due to the locality of the piecewise linear finite element functions, $(M)_{\kappa\lambda}$ is non-zero only if the nodes κ and λ are adjacent to each other.

Now divide the domain at level Q into 2^{3Q} subcubes, labelled by

$$\mathcal{D}_{Q;i,j,k} = \left[\frac{i}{2^Q},\frac{i+1}{2^Q}\right]\times\left[\frac{j}{2^Q},\frac{j+1}{2^Q}\right]\times\left[\frac{k}{2^Q},\frac{k+1}{2^Q}\right].$$

for $0\le i,j,k<2^Q$ using 2^Q+1 equally spaced planes at $\frac{0}{2^Q},\frac{1}{2^Q},\frac{2}{2^Q},\ldots,\frac{2^Q}{2^Q}$ in each dimension. Some nodes will lie on the boundary of the subcubes and will be shared with their neighbors. The overlapping set of $(P+1)\times(P+1)\times(P+1)$ nodes (at the domain boundary cubes may however contain fewer nodes) in each subcube will be called

$$\mathcal{V}_{Q;i,j,k} \qquad \text{where} \qquad 0\le i,j,k<2^Q.$$

Some nodes will be contained solely within one subcube and will be in the subcube's *interior* while those shared with other cubes (or on the boundary of $[0,1]^3$) will be on the subcube's *boundary*. If a node's position is divisible by P in some direction it will be on a subcube boundary otherwise it will be in a subcube interior.

The nodes in the interior of one subcube do not contribute via M to the values in the interior of another subcube. Since the entry $M_{\kappa\lambda}$ is non-zero only when κ and λ are adjacent to each other, the boundary layer between cubes is sufficient to decouple the nodes in the interior of each subcube at level Q from those in another subcube at the same level.

Another way of looking at the set of nodes that make up each subcube can be obtained if we divide the $P2^Q+1$ nodes $0,1,2,\ldots,P2^Q$ in the x-direction into $2^{Q+1}+1$ groups of alternating sizes 1 and $P-1$, that is node 0 in group 0, nodes $1,\ldots,P-1$ in group 1, node P in group 2, nodes $P+1,\ldots,2P-1$ in group 3, until eventually node 2^QP is in group 2^{Q+1}. That is, all the nodes whose position X is a multiple of P are in the even groups and the other nodes in between are in the odd groups. Now perform the same division into these alternating groups in the y- and z-directions.

A node is in even or odd groups for a particular direction depending on the divisibility of that node's position by P. Thus every node will be in a group (i,j,k) for $0\le i,j,k\le 2^{Q+1}$ where i is the group from the x-direction grouping, j the group

from the y-direction and k the group from the z-direction. These different sets of nodes on the last level Q are called *leaf elements* and labelled by

$$\mathcal{E}_{Q;i,j,k} \qquad \text{where} \qquad 0 \le i,j,k \le 2^{Q+1}.$$

Now, depending on the number of even or odd i,j,k in the element's label, we call it a *corner, segment, facet* or *volume* element. Thus a element can be

- *Corner element*, which consists of a single node and is of size 1 by 1 by 1.
- *Segment element*, which extends in 1 basis direction and is of size 1 by 1 by $P-1$ (or permutation).
- *Facet element*, which extends in 2 basis directions and is of size 1 by $P-1$ by $P-1$ (or permutation).
- *Volume element*, which extends in all 3 basis directions, and is of size $P-1$ by $P-1$ by $P-1$.

These names have geometric significance because one can also think of the various ways the planes parallel to the axes can intersect.

If we consider the parity of each of the group numbers in turn there are $2^3 = 8$ different types of nodes.

Element	$i \pmod 2$	$j \pmod 2$	$k \pmod 2$
Corner	0	0	0
x-direction segment	1	0	0
y-direction segment	0	1	0
z-direction segment	0	0	1
x-parallel facet	0	1	1
y-parallel facet	1	0	1
z-parallel facet	1	1	0
Volume	1	1	1

Each subcube at level Q is made up of a $3 \times 3 \times 3 = 27$ elements

$$\mathcal{V}_{Q;i,j,k} = \biguplus_{0 \le i',j',k' \le 2} \mathcal{E}_{Q;2i+i',2j+j',2k+k'},$$

where we have used the symbol $\uplus$ to indicate a *disjoint union* so as to distinguish it from the more general union. These elements can be further classified as interior and boundary ones as follows:

- $\mathcal{I}_{Q;i,j,k}$: the interior that contains 1 volume element.
- $\mathcal{B}_{Q;i,j,k}$: the boundary that contains 6 facet elements (2 parallel to each axis), 12 segment elements (4 parallel to each axis), and 8 corner elements.

More precisely, we have $\mathcal{V}_{Q;i,j,k} = \mathcal{I}_{Q;i,j,k} \uplus \mathcal{B}_{Q;i,j,k}$ with

$$\mathcal{I}_{Q;i,j,k} := \mathcal{E}_{Q;2i+1,2j+1,2k+1}$$

for the interior and

$$\mathcal{B}_{Q;i,j,k} := \biguplus_{\substack{0 \le i',j',k' \le 2 \\ (i',j',k') \ne (1,1,1)}} \mathcal{E}_{Q;2i+i',2j+j',2k+k'}$$

for the boundary of the subcube. In Figure 2.1 we illustrate an example of 8 overlapping leaf cubes distinguishing between the different kinds of elements and showing how the cubes meet. Notice that each node (which is indeed a point) is plotted as a small cube in Figure 2.1 in order to be able to show corner and segment elements more clearly. It shall not be confused with the actual subcube that contains these nodes.

Based on what we introduced so far, we can define the vertex sets and elements for all other levels from bottom up. Suppose that $\mathcal{V}_{q+1;i,j,k}$, $\mathcal{I}_{q+1;i,j,k}$, $\mathcal{B}_{q+1;i,j,k}$, and $\mathcal{E}_{q+1;i,j,k}$ are already defined for subcubes $\mathcal{D}_{q+1;i,j,k}$ on level $q+1$. For a subcube $\mathcal{D}_{q;i,j,k}$ on level q, the vertex set is

$$\mathcal{V}_{q;i,j,k} := \bigcup_{0 \le i',\, j',\, k' \le 1} \mathcal{B}_{q+1;2i+i',2j+j',2k+k'}.$$

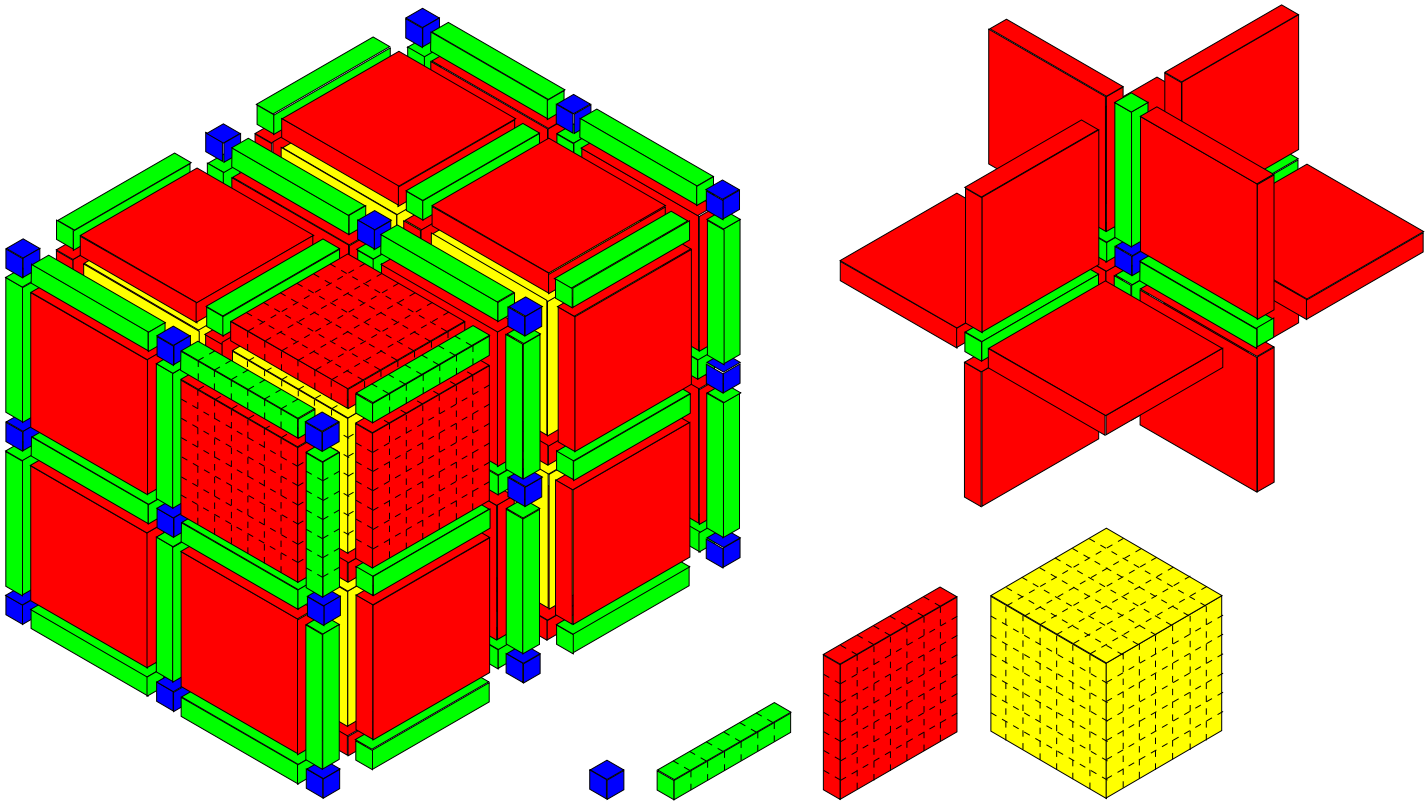


Fig. 2.1: 8 overlapping leaf subcubes of size 9^3. The four types of node sets are shown as well as the interior planes.

This set is further partitioned into two parts: $\mathcal{I}_{q;i,j,k}$, the interior and $\mathcal{B}_{q;i,j,k}$, the boundary. In terms of the elements defined on level $q+1$, the sets $\mathcal{I}_{q;i,j,k}$ and $\mathcal{B}_{q;i,j,k}$ can be written as

$$\mathcal{I}_{q;i,j,k} := \biguplus_{\substack{1 \le i',j',k' \le 3 \\ \text{at least one } 2}} \mathcal{E}_{q+1;4i+i',4j+j',4k+k'}$$

and

$$\mathcal{B}_{q;i,j,k} := \biguplus_{\substack{0 \le i',j',k' \le 4 \\ \text{at least one } 0 \text{ or } 4}} \mathcal{E}_{q+1;4i+i',4j+j',4k+k'},$$

respectively. A simple counting shows that $\mathcal{V}_{q;i,j,k}$ consists of 117 elements from level $q+1$: 36 facets, 54 segments and 27 corners. They are allocated into $\mathcal{I}_{q;i,j,k}$ and $\mathcal{B}_{q;i,j,k}$ as follows:

- Interior $\mathcal{I}_{q;i,j,k}$ consists of 19 elements: 12 facets, 6 segments, and 1 corner unique to the 8 siblings.
- Boundary $\mathcal{B}_{q;i,j,k}$ consists of 98 elements: 24 facets, 48 segments, and 26 corners which are shared with other subcubes.

To give an example, the elements of $\mathcal{B}_{q;i,j,k}$ on a slice parallel to the z-axis are displayed in the following table.

$\mathcal{E}_{q+1;4i,4j+4,4k}$	$\mathcal{E}_{q+1;4i+1,4j+4,4k}$	$\mathcal{E}_{q+1;4i+2,4j+4,4k}$	$\mathcal{E}_{q+1;4i+3,4j+4,4k}$	$\mathcal{E}_{q+1;4i+4,4j+4,4k}$
$\mathcal{E}_{q+1;4i,4j+3,4k}$	$\mathcal{E}_{q+1;4i+1,4j+3,4k}$	$\mathcal{E}_{q+1;4i+2,4j+3,4k}$	$\mathcal{E}_{q+1;4i+3,4j+3,4k}$	$\mathcal{E}_{q+1;4i+4,4j+3,4k}$
$\mathcal{E}_{q+1;4i,4j+2,4k}$	$\mathcal{E}_{q+1;4i+1,4j+2,4k}$	$\mathcal{E}_{q+1;4i+2,4j+2,4k}$	$\mathcal{E}_{q+1;4i+3,4j+2,4k}$	$\mathcal{E}_{q+1;4i+4,4j+2,4k}$
$\mathcal{E}_{q+1;4i,4j+1,4k}$	$\mathcal{E}_{q+1;4i+1,4j+1,4k}$	$\mathcal{E}_{q+1;4i+2,4j+1,4k}$	$\mathcal{E}_{q+1;4i+3,4j+1,4k}$	$\mathcal{E}_{q+1;4i+4,4j+1,4k}$
$\mathcal{E}_{q+1;4i,4j,4k}$	$\mathcal{E}_{q+1;4i+1,4j,4k}$	$\mathcal{E}_{q+1;4i+2,4j,4k}$	$\mathcal{E}_{q+1;4i+3,4j,4k}$	$\mathcal{E}_{q+1;4i+4,4j,4k}$

In order to support the algorithms to be described below, one needs to introduce a decomposition of $\mathcal{B}_{q;i,j,k}$ in terms of elements on its own level (level q), instead of the child level (level $q+1$). We define *parent elements* $\mathcal{E}_{q;i,j,k}$ on level q as follows. Each parent corner consists of a single corner from level $q+1$, for example

$$\mathcal{E}_{q;2i,2j,2k} := \mathcal{E}_{q+1;4i,4j,4k}.$$

The child segment, corner and segment combine into a parent segment

$$\mathcal{E}_{q;2i+1,2j,2k} := \mathcal{E}_{q+1;4i+1,4j,4k} \uplus \mathcal{E}_{q+1;4i+2,4j,4k} \uplus \mathcal{E}_{q+1;4i+3,4j,4k}$$

and similarly for segments in the other directions, while the parent facet consists of 4 facets, 4 segments, and 1 corner:

$$\mathcal{E}_{q;2i,2j+1,2k+1} := \begin{array}{c} \mathcal{E}_{q+1;4i,4j+1,4k+1} \uplus \mathcal{E}_{q+1;4i,4j+2,4k+1} \uplus \mathcal{E}_{q+1;4i,4j+3,4k+1} \uplus \\ \mathcal{E}_{q+1;4i,4j+1,4k+2} \uplus \mathcal{E}_{q+1;4i,4j+2,4k+2} \uplus \mathcal{E}_{q+1;4i,4j+3,4k+2} \uplus \\ \mathcal{E}_{q+1;4i,4j+1,4k+3} \uplus \mathcal{E}_{q+1;4i,4j+2,4k+3} \uplus \mathcal{E}_{q+1;4i,4j+3,4k+3} \end{array}.$$

The parent facets parallel to the other axes are defined similarly. This combination is trickier than in two dimensions (see [21]), because there many more pieces to consider and there are more ways to order the combination of child elements in a parent facet. In Figure 2.2 we show exactly how the boundary elements of one level (level $q+1$) are combined to form larger elements one level higher (level q).

In terms of the new elements introduced on level q, the slice parallel to the z-axis that we showed earlier can be now represented as follows.

$\mathcal{E}_{q;2i,2j+2,2k}$	$\mathcal{E}_{q;2i+1,2j+2,2k}$	$\mathcal{E}_{q;2i+2,2j+2,2k}$
$\mathcal{E}_{q;2i,2j+1,2k}$	$\mathcal{E}_{q;2i+1,2j+1,2k}$	$\mathcal{E}_{q;2i+2,2j+1,2k}$
$\mathcal{E}_{q;2i,2j,2k}$	$\mathcal{E}_{q;2i+1,2j,2k}$	$\mathcal{E}_{q;2i+2,2j,2k}$

Thus we can rewrite

$$\mathcal{B}_{q;i,j,k} := \biguplus_{\substack{0\le i',j',k'\le 2 \\ \text{at least one } 0 \text{ or } 2}} \mathcal{E}_{q;2i+i',2j+j',2k+k'}.$$

In this way, the 98 elements of $\mathcal{B}_{q;i,j,k}$ on level $q+1$ are combined into 26 combined elements on level q as follows.

	98 Divided					
	26 Corners	48 Segments	24 Facets	$\longrightarrow$	26 Combined	
$8\times$	1				8	Corners
$12\times$	1	2			12	Segments
$6\times$	1	4	4		6	Facets

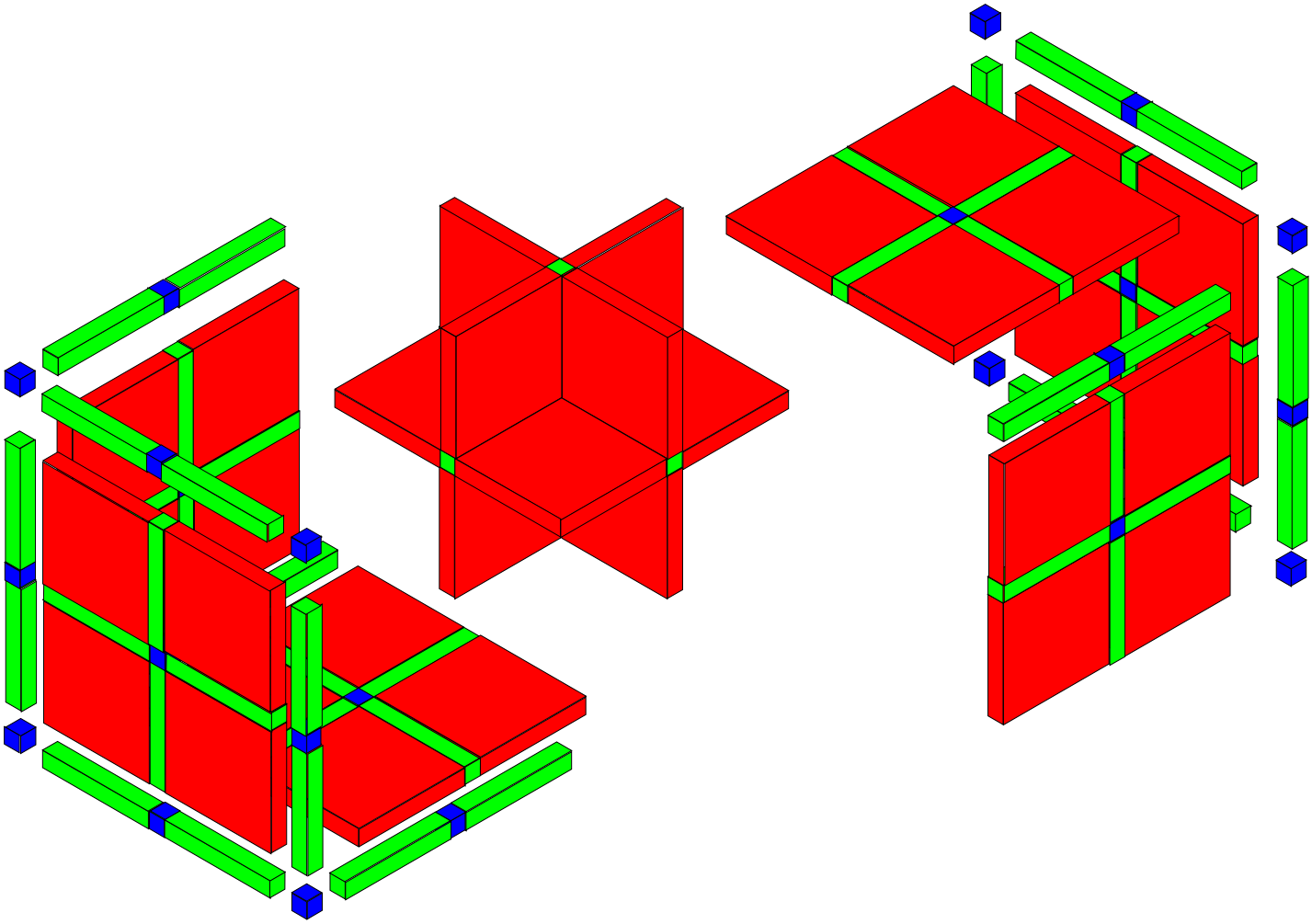


Fig. 2.2: Hierarchical division of the cube: 6 combined facets, 8 combined segments and 8 corners.

3 Multifrontal Method

The multifrontal method essentially constructs and stores a block LDL^t transform of M in an efficient way. The basic tool of this process is Schur complement. For a matrix

$$\begin{pmatrix} A & B^t \\ B & C \end{pmatrix},$$

this would be

$$\begin{pmatrix} A & B^t \\ B & C \end{pmatrix} = \begin{pmatrix} I & \\ BA^{-1} & I \end{pmatrix} \begin{pmatrix} A & \\ & S \end{pmatrix} \begin{pmatrix} I & A^{-1}B^t \\ & I \end{pmatrix},$$

where $S = C - BA^{-1}B^t$. For each cube $\mathcal{D}_{Q;i,j,k}$, we can then consider the small matrix $M_{Q;i,j,k}$, which is the restriction of M to the cube $\mathcal{D}_{Q;i,j,k}$) formed via

$$(M_{Q;i,j,k})_{\kappa\lambda} = \int_{\mathcal{D}_{Q;i,j,k}} \nabla\phi_\kappa(\mathbf{x}) \cdot a(\mathbf{x})\nabla\phi_\lambda(\mathbf{x}) + V(\mathbf{x})\phi_\kappa(\mathbf{x})\phi_\lambda(\mathbf{x})\mathrm{d}\mathbf{x}, \tag{3.1}$$

where κ and λ are restricted to the vertices in $\mathcal{D}_{Q;i,j,k}$ since all basis functions centered on vertices outside $\mathcal{D}_{Q;i,j,k}$ are zero inside $\mathcal{D}_{Q;i,j,k}$. These matrices $M_{Q;i,j,k}$ sum (after suitable injection) to the full matrix M.

Start from level Q and fix a leaf cube $\mathcal{D}_{Q;i,j,k}$ and its matrix $M_{Q;i,j,k}$. The decomposition of the nodes $\mathcal{V}_{Q;i,j,k} = \mathcal{I}_{Q;i,j,k} \uplus \mathcal{B}_{Q;i,j,k}$ leads to a 2×2 block matrix decomposition of $M_{Q;i,j,k}$:

$$M_{Q;i,j,k} = \begin{pmatrix} A_{Q;i,j,k} & B^t_{Q;i,j,k} \\ B_{Q;i,j,k} & C_{Q;i,j,k} \end{pmatrix} = L_{Q;i,j,k} \begin{pmatrix} A_{Q;i,j,k} & \\ & S_{Q;i,j,k} \end{pmatrix} L^t_{Q;i,j,k}, \tag{3.2}$$

where $A_{Q;i,j,k} \colon \mathcal{I}_{Q;i,j,k} \to \mathcal{I}_{Q;i,j,k}$, $B_{Q;i,j,k} \colon \mathcal{I}_{Q;i,j,k} \to \mathcal{B}_{Q;i,j,k}$, $C_{Q;i,j,k} \colon \mathcal{B}_{Q;i,j,k} \to \mathcal{B}_{Q;i,j,k}$, and

$$L_{Q;i,j,k} = \begin{pmatrix} I_{\mathcal{I}_{Q;i,j,k}} & \\ B_{Q;i,j,k}A^{-1}_{Q;i,j,k} & I_{\mathcal{B}_{Q;i,j,k}} \end{pmatrix}$$

We can extend $L_{Q;i,j,k}$ to the whole vertex set by setting it to be identity on the complement of $\mathcal{V}_{Q;i,j,k}$. Since the interior vertex sets $\mathcal{I}_{Q;i,j,k}$ are disjoint for different blocks $\mathcal{D}_{Q;i,j,k}$, each one of $L_{Q;i,j,k}$ commutes with another distinct $L_{Q;i',j',k'}$ and as a result $L_Q := \prod_{i,j,k} L_{Q;i,j,k}$ is well defined.

We will develop a suitable ordering for the rows and columns of M as we proceed. Let us define

$$\mathcal{I}_Q := \biguplus_{i,j,k} \mathcal{I}_{Q;i,j,k} \quad \text{and} \quad \mathcal{B}_Q := \bigcup_{i,j,k} \mathcal{B}_{Q;i,j,k}.$$

Since the union of the two is the entire set of nodes for which we constructed M, we have the following 2×2 block form for M

$$M = \begin{pmatrix} A_Q & B^t_Q \\ B_Q & C_Q \end{pmatrix} = L_Q \begin{pmatrix} A_Q & \\ & S_Q \end{pmatrix} L^t_Q,$$

where $A_Q \colon \mathcal{I}_Q \to \mathcal{I}_Q$, $B_Q \colon \mathcal{I}_Q \to \mathcal{B}_Q$, $C_Q \colon \mathcal{B}_Q \to \mathcal{B}_Q$, and $S_Q = C_Q - B_Q A_Q^{-1} B_Q^t$. For each subcube $\mathcal{D}_{Q-1;i,j,k}$ at level $Q-1$, let us define $M_{Q-1;i,j,k}$ to be the sum of the Schur complement matrices $S_{Q;i',j',k'}$ over $\mathcal{D}_{Q-1;i,j,k}$'s 8 children. Then it is not difficult to see that S_Q is in fact equal to the sum of all $M_{Q-1;i,j,k}$ (if we extend $M_{Q-1;i,j,k}$ by setting it to be zero outside $\mathcal{V}_{Q-1;i,j,k}$).

Now recall that each $\mathcal{V}_{Q-1;i,j,k}$ decomposes into $\mathcal{I}_{Q-1;i,j,k}$ and $\mathcal{B}_{Q-1;i,j,k}$. It leads to a 2×2 block form for $M_{Q-1;i,j}$

$$M_{Q-1;i,j,k} := \begin{pmatrix} A_{Q-1;i,j,k} & B^t_{Q-1;i,j,k} \\ B_{Q-1;i,j,k} & C_{Q-1;i,j,k} \end{pmatrix}$$

where $A_{Q-1;i,j} \colon \mathcal{I}_{Q-1;i,j,k} \to \mathcal{I}_{Q-1;i,j,k}$, $B_{Q-1;i,j} \colon \mathcal{I}_{Q-1;i,j,k} \to \mathcal{B}_{Q-1;i,j,k}$, and $C_{Q-1;i,j} \colon \mathcal{B}_{Q-1;i,j,k} \to \mathcal{B}_{Q-1;i,j,k}$. We can perform a Schur complement on this 2×2 block matrix at this level to obtain

$$L_{Q-1;i,j,k} \begin{pmatrix} A_{Q-1;i,j,k} & \\ & S_{Q-1;i,j,k} \end{pmatrix} L^t_{Q-1;i,j,k}.$$

By introducing the decomposition of $\mathcal{B}_Q$ into the union of

$$\mathcal{I}_{Q-1} := \biguplus_{i,j} \mathcal{I}_{Q-1;i,j,k} \quad \text{and} \quad \mathcal{B}_{Q-1} := \bigcup_{i,j} \mathcal{B}_{Q-1;i,j,k}.$$

and defining

$$L_{Q-1} := \prod_{i,j,k} L_{Q-1;i,j,k},$$

we can rewrite the above computation as factoring S_Q into

$$L_{Q-1}\begin{pmatrix} A_{Q-1} & \\ & S_{Q-1}\end{pmatrix}L_{Q-1}^t.$$

As a result we get

$$M = L_Q\begin{pmatrix} A_Q & \\ & S_Q\end{pmatrix}L_Q^t = L_QL_{Q-1}\begin{pmatrix} A_Q & & \\ & A_{Q-1} & \\ & & S_{Q-1}\end{pmatrix}L_{Q-1}^tL_Q^t.$$

Continuing in this fashion over all levels from bottom up, we eventually reach level 0 with $\mathcal{B}_1 = \mathcal{I}_0$, $\mathcal{B}_0 = \emptyset$ (due to the zero Dirichlet boundary condition), and

$$M = L_QL_{Q-1}\cdots L_1\begin{pmatrix} A_Q & & & & \\ & A_{Q-1} & & & \\ & & \ddots & & \\ & & & A_1 & \\ & & & & A_0\end{pmatrix}L_1^t\cdots L_{Q-1}^tL_Q^t,$$

where $A_q\colon \mathcal{I}_q \to \mathcal{I}_q$. Each of the $A_q,\ q = 0,\ldots,Q$ will, in fact, be block diagonal if we treat

$$\mathcal{I}_q := \biguplus_{0\le i<2^q}\ \biguplus_{0\le j<2^q}\ \biguplus_{0\le k<2^q} \mathcal{I}_{q;i,j,k}$$

taking each of the sets $\mathcal{I}_{q;i,j,k}$ in turn for our ordering.

The solution to (1.1) can then be found by applying

$$M^{-1} = L_Q^{-t}L_{Q-1}^{-t}\cdots L_1^{-t}\begin{pmatrix} A_Q^{-1} & & & & \\ & A_{Q-1}^{-1} & & & \\ & & \ddots & & \\ & & & A_1^{-1} & \\ & & & & A_0^{-1}\end{pmatrix}L_1^{-1}\cdots L_{Q-1}^{-1}L_Q^{-1}. \tag{3.3}$$

In terms of computational complexity, the multifrontal factorization of M can be constructed in $\mathcal{O}(N^2)$ steps and solving $Mu = f$ by applying (3.3) to f takes $\mathcal{O}(N^{4/3})$ steps. To see this, first notice that the number of levels is $Q+1$ and the total number of unknowns is $N \simeq (P2^Q)^3 = P^32^{3Q}$. For each level q, let us use $s(q) \simeq P2^{Q-q}$ to denote the sidelength of the subcube at this level. Since each facet has size $s(q)^2$, the largest matrices appearing at level q in the multifrontal algorithm have size $O(s(q)^2)$. Thus the cost of multiplying and inverting matrices for each subcube will be $\mathcal{O}((s(q)^2)^3)$, while the cost of a matrix-vector multiply will be $\mathcal{O}((s(q)^2)^2)$. Thus the total cost, suppressing constants, for setting up the factorization for M and M^{-1} is

$$\sum_{q=0}^{Q} s(q)^6 2^{3q} = \sum_{q=0}^{Q}(P2^{Q-q})^6 2^{3q} = \mathcal{O}(N^2)$$

and that for applying M^{-1} to a vector is

$$\sum_{q=0}^{Q} s(q)^4 2^{3q} = \sum_{q=0}^{Q}(P2^{Q-q})^4 2^{3q} = \mathcal{O}(N^{4/3})$$

as claimed.

As mentioned in [21] in two dimensions Xia *et al.* [23] suggested using hierarchical semiseparable matrices [22] or hierarchical matrices [11] to represent the matrices that appeared in their decomposition. This allows the Schur complements at all levels to be performed in almost linear time. Extending the approach of [23] to the three dimensional case is not straightforward.

Our recapitulation of their ideas in [21] can however be more easily extended to three dimensional problems. Once we have established the hierarchical structure of elements and the relationships between boundary and interior elements of cubes on various levels the general approach is similar to the one elucidated before. In three dimensions the structure of the hierarchical matrices is harder to determine as the ways different interacting pieces in the domain can relate to one another are much more complex than in two dimensions.

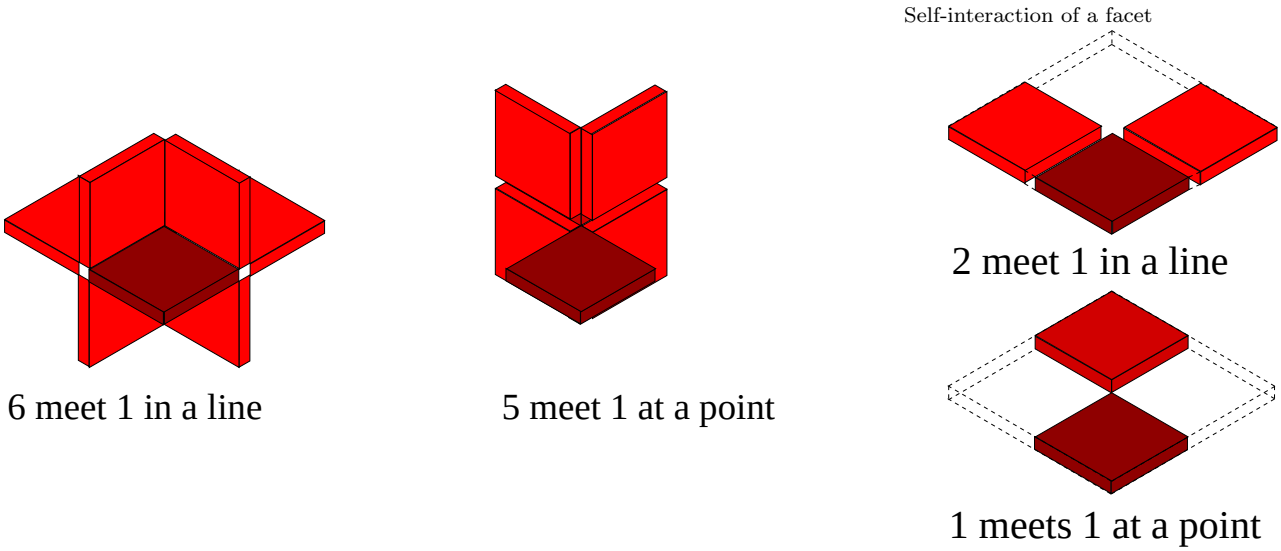


Fig. 4.1: Left: Examples of facets meeting in a line or at a point for the facets in $\mathcal{I}_{q;i,j,k}$. Right: self-interaction of a facet

4 Hierarchical Matrix Decomposition

For each subcube, the decomposition of $\mathcal{V}_{q;i,j,k}$ in terms of elements at level $q+1$ provides a block structure for the matrices in the Schur complement computation. For example, $A_{q;,i,j,k}$ is of size 19×19, $B_{q;,i,j,k}$ of size 98×19, and $C_{q;i,j,k}$ and $S_{q;i,j,k}$ of size 98×98. When written using the elements on level q, $S_{q;i,j,k}$ is of size 26×26. The hierarchical matrix decomposition for these matrices depends on the choices made as to which submatrices are represented by hierarchical matrices and which are represented by factorized low rank matrices. The choice of which submatrices can be represented by low rank matrices depends on the *admissibility* criteria [11, 12] used. In our case this reveals itself in the choice to represent the interaction between different facets larger than a chosen size as one factorized matrix instead of the hierarchical matrix required by the lack of separation between the facets.

Well separated facets, such as those on the top and bottom of a cube, will always have their interaction represented in factorized form. For nearby facets, such as the 12 facet elements in the interior $\mathcal{I}_{q;i,j,k}$ of $\mathcal{D}_{q;i,j,k}$, we distinguish between two different settings (as illustrated in Figure 4.1), depending on whether the minimum distance between two facets is achieved

- At a point, or
- On a line.

Those facets achieving minimum distance at a point will always have a low rank interaction and be represented in factorized form. On the other hand, those facets achieving minimum distance in a line will be represented in two different ways

- *Type I*: by a hierarchical matrix,
- *Type II*: in a factorized form.

The advantage of Type I is that the growth of the maximum rank is better controlled, but at the cost of a deeper and more complex hierarchical decomposition.

In Figure 4.2, we illustrate a possible Type I hierarchical division of two adjacent facets of a subcube. This situation appears between two level $q+1$ facets in $A_{q;i,j,k}$ or two level q facets in $S_{q;i,j,k}$. The main pieces of the subdivision of a facet are the 4 sub-facets labelled $1, 3, 7, 9$ for one facet and a, c, g, i for the other. The interactions of the adjacent sub-facets (for example between 1 and a and between 7 and c) are represented hierarchically while the rest are low-rank. Considering the further subdivisions of 1 and a the sub-sub-facets show the same pattern of interaction with $\tilde{1}$ & $\tilde{a}$ and $\tilde{7}$ & $\tilde{c}$ represented densely while the others are low-rank. Depending on the sizes of the remaining sub-sub-facets one might continue further subdivision and hierarchical representation over the dense one. For Type II we would have one big factorized matrix as the two large adjacent (undivided) facets meet in a line.

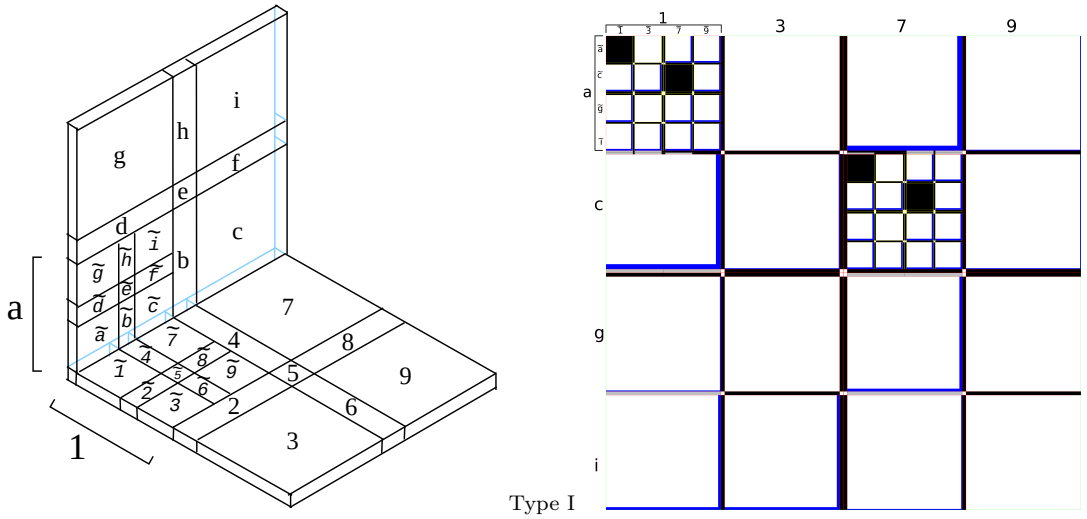
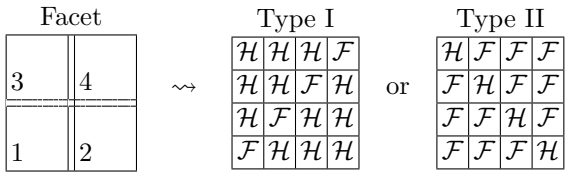


Fig. 4.2: Type I hierarchical division of adjacent Facets of the cube and associated interaction matrix with factorized matrices represented by the symbol ⌜ or ⌟

For the case of facet self-interaction let us consider a facet that has been divided into the 9 child elements as shown in Figure 4.3 (dominated by the 4 sub-facets labeled 1,3,7 and 9) with subfacet 1 further divided into $\tilde{1}$ to $\tilde{9}$. If we only look at the interaction of 4 subfacets of a facet we would see the following patterns.



In Type I only the interaction between facets diagonally opposite each other is represented in factorized form, while in Type II only the self-interaction is represented in hierarchical form.

5 Algorithm

By incorporating the hierarchical matrix algebra into the framework of the multifrontal method, we see that creating the factorized form of M happens in two stages

1. At the leaf level we calculate $M_{Q;i,j,k}$ which is the restriction M to $\mathcal{D}_{Q;i,j,k}$,
2. Move up level by level combining the 8 child $S_{q+1;i',j',k'}$ matrices into the parent $M_{q;i,j,k}$.

This is detailed in Algorithm 1. In the description, we use the MATLAB notation for referring to submatrices. For example, if $G \in \mathbb{R}^{|\mathcal{J}|\times|\mathcal{J}|}$ is a matrix whose rows and columns are labelled by the index set $\mathcal{J}$ then for $\mathcal{X} \subset \mathcal{J}$ we write $G(\mathcal{X},\mathcal{X}) \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}$ for the submatrix of G consisting of the rows and columns in $\mathcal{X}$. Manipulating this matrix affects the underlying values in G.

Algorithm 1 (Setup the factorization of M)

1: **for** $i = 0$ to $2^Q - 1$ **do**
2: **for** $j = 0$ to $2^Q - 1$ **do**
3: **for** $k = 0$ to $2^Q - 1$ **do**
4: Calculate the matrix $M_{Q;i,j,k}$.
5: Invert $A_{Q;i,j,k}$ using dense matrix methods.

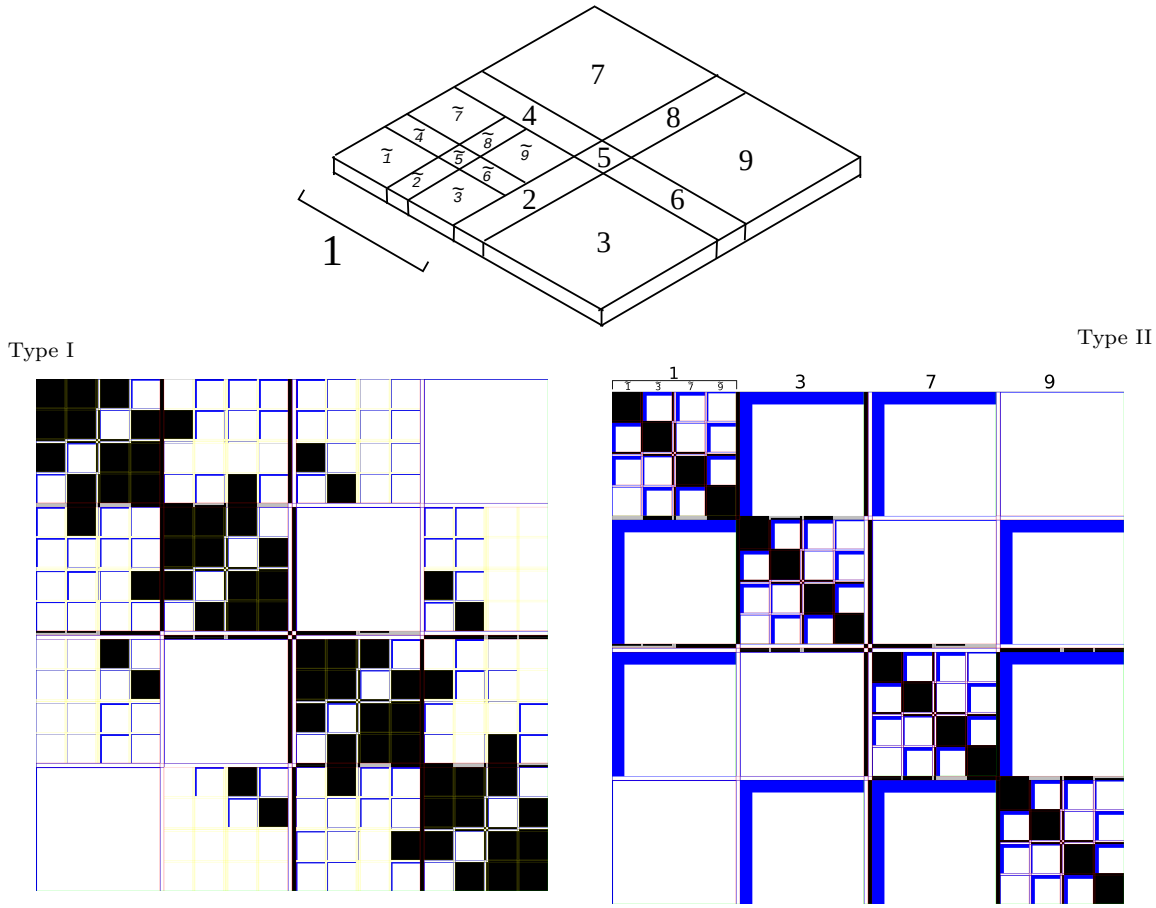


Fig. 4.3: Hierarchical division of self-interaction of a facet of the cube in Type I and Type II matrix decompositions. This would correspond to one part on the block diagonal of the full interaction matrix for the 6 facets (and the rest of the cube) shown in Figure 6.1 and Figure 6.2

.

$S_{Q;i,j,k} \leftarrow C_{Q;i,j,k} - B_{Q;i,j,k} A^{-1}_{Q;i,j,k} B^t_{Q;i,j,k}$
Store $A^{-1}_{Q;i,j,k}$, $B_{Q;i,j,k}$ and $S_{Q;i,j,k}$
end for
end for
end for
for $q = Q - 1$ to 1 **do**
for $i = 0$ to $2^q - 1$ **do**
for $j = 0$ to $2^q - 1$ **do**
for $k = 0$ to $2^q - 1$ **do**
Start with zero $M_{q;i,j,k}$
for $i' = 0, 1$ **do**
for $j' = 0, 1$ **do**
for $k' = 0, 1$ **do**
Add $S_{q+1,2i+i',2j+j',2k+k'}$ to $M_{q;i,j,k}(\mathcal{B}_{q+1;2i+i',2j+j',2k+k'}, \mathcal{B}_{q+1;2i+i',2j+j',2k+k'})$
end for
end for
end for
end for
Decompose $M_{q;i,j,k}$ into 2×2 block structure $M_{q;i,j} = \begin{pmatrix} A_{q;i,j} & B^t_{q;i,j} \\ B_{q;i,j} & C_{q;i,j} \end{pmatrix}$.

Invert $A_{q;i,j,k}$.
$S_{q;i,j,k} \leftarrow C_{q;i,j,k} - B_{q;i,j,k} A^{-1}_{q;i,j,k} B^t_{q;i,j,k}$
Store $A^{-1}_{q;i,j,k}$ and $B_{q;i,j,k}$
Merge and Store $S_{q;i,j,k}$
end for
end for
end for
end for
Start with zero $M_{0;0,0,0}$
for $i' = 0, 1$ **do**
for $j' = 0, 1$ **do**
for $k' = 0, 1$ **do**
Add $S_{1,i',j',k'}$ to $M_{0;0,0,0}(\mathcal{B}_{1;i',j',k'}, \mathcal{B}_{1;i',j',k'})$
end for
end for
end for
Invert $A_{0;0,0,0}$.
Store $A_{0;0,0,0}$.

The step *Merge* $S_{q;i,j,k}$ in Algorithm 1 is required because we need to reinterpret the 98×98 submatrices corresponding to the 98 boundary elements at level $q+1$ as 26×26 submatrices corresponding to the 26 merged boundary elements at level q. 8 corner nodes are unaffected. The segment-corner-segment merging into a parent segment is reflected in combining $3 \times 3 = 9$ submatrices into a new submatrix. Merging of 9 child elements in a plane into a parent facet corresponds to $9 \times 9 = 81$ submatrices being joined together. The vertex ordering we built up from the leaf level ensures that, in fact, these 9 submatrices form a contiguous 3×3 group and the 81 submatrices form a contiguous 9×9 group. Thus no rearrangement of the rows and columns of S is required.

In terms of the hierarchical matrix representation, if the new submatrix should have a hierarchical representation this is achieved by simply reinterpreting the 3×3 (in the case of a parent segment) or 9×9 (in the case of a parent facet) submatrices as part of a new hierarchical matrix decomposition. On the other hand if the new submatrix should be represented in factorized form, this conversion can be performed efficiently using standard QR and SVD [11].

This correspondence between the hierarchical decomposition of the geometric elements, segments and corners, on different levels and the hierarchical matrix decomposition of the matrices for those sets of nodes means that the choice of hierarchical structure for the matrices emerges naturally from our domain decomposition.

To solve the original $Mu = f$, we compute $u = M^{-1} f$ using the following formula

$$M^{-1} f = L_Q^{-t} L_{Q-1}^{-t} \cdots L_1^{-t} \begin{pmatrix} A_Q^{-1} & & & & \\ & A_{Q-1}^{-1} & & & \\ & & \ddots & & \\ & & & A_1^{-1} & \\ & & & & A_0^{-1} \end{pmatrix} L_1^{-1} \cdots L_{Q-1}^{-1} L_Q^{-1} f,$$

where $L_q^{-1} = \prod_{i,j,k} L^{-1}_{q;i,j,k}$. First we apply each $L^{-1}_{Q;i,j,k}$ in L_Q^{-1}, then those from L_{Q-1}^{-1} and so on. Once we have completed all the $L^{-1}_{q;i,j,k}$, we apply the block diagonal $A^{-1}_{q;i,j,k}$, and then all the $L^{-t}_{q;i,j,k}$. If we write $u_{\mathcal{I}_{q;i,j,k}}$ for the (consecutive) group of components of u corresponding to the set of nodes $\mathcal{I}_{q;i,j,k}$, and similarly $u_{\mathcal{B}_{q;i,j,k}}$, then the solution can be calculated as in Algorithm 2 where we combine the action of $A^{-1}_{q;i,j,k}$ and $L^{-1}_{q;i,j,k}$ since they are the only ones which affect $u_{\mathcal{I}_{q;i,j,k}}$ on the first pass from the leaves to the root of the tree.

Algorithm 2 (Solving $Mu = f$)

$u \leftarrow f$
for $q = Q$ to 1 **do**
for $i = 0$ to $2^q - 1$ **do**
for $j = 0$ to $2^q - 1$ **do**

for $k = 0$ to $2^q - 1$ **do**
$u_{\mathcal{I}_{q;i,j,k}} \leftarrow A^{-1}_{q;i,j,k} u_{\mathcal{I}_{q;i,j,k}}$
$u_{\mathcal{B}_{q;i,j,k}} \leftarrow u_{\mathcal{B}_{q;i,j,k}} - B_{q;i,j,k} A^{-1}_{q;i,j,k} u_{\mathcal{I}_{q;i,j,k}}$
end for
end for
end for
end for
$u_{\mathcal{I}_{0;0,0,0}} \leftarrow A^{-1}_{0;0,0,0} u_{\mathcal{I}_{0;0,0,0}}$
for $q = 1$ to Q **do**
for $i = 0$ to $2^q - 1$ **do**
for $j = 0$ to $2^q - 1$ **do**
for $k = 0$ to $2^q - 1$ **do**
$u_{\mathcal{I}_{q;i,j,k}} \leftarrow u_{\mathcal{I}_{q;i,j,k}} - A^{-1}_{q;i,j,k} B^t_{q;i,j,k} u_{\mathcal{B}_{q;i,j,k}}$
end for
end for
end for
end for

6 Complexity

Let us investigate the complexity of the algorithms presented in Section 5. We recall that a leaf node at level Q contains $(P+1) \times (P+1) \times (P+1)$ nodes and $N \simeq (P2^Q)^3 = P^3 2^{3Q} = O(2^{3Q})$ since $P = \mathcal{O}(1)$. Here all logarithms are taken with base 2.

The cost of the hierarchical levels depends on the depth of the decomposition tree and the ranks of the factorized approximations. Now the cost of these operations is given in [8] as $\mathcal{O}(r^2(\log n)^2 n)$ where r is the maximum rank of the factorized parts, $n \times n$ is the full size of the matrix, and $\log n$ comes from the number of subdivision levels (depth of the decomposition tree) in the hierarchical matrix representation.

The size of the matrices involved for a given subcube is dominated by the number of nodes in the covered facets. Fix a cube at $\mathcal{D}_{q;i,j,k}$ at level q. For $A_{q;i,j,k}$, there are 12 facets and we have $\mathcal{O}(s(q)^2)$ size. Similarly the corresponding $B_{q;i,j,k}$ and $S_{q;i,j,k}$ will also have size $n = \mathcal{O}(s(q)^2)$. The formation of the Schur complement for each subcube involves an inversion of a hierarchical matrix, two multiplications of hierarchical matrices, and an addition of hierarchical matrices.

The difference in complexity for the Type I and Type II hierarchical matrix decompositions comes from the differing rank r of the factorized parts. We have observed in numerical experiments that the rank of the factorized parts for the Type I decomposition grows like $\mathcal{O}(\log s(q))$ while that for Type II grows like $\mathcal{O}(s(q))$ as shown in Table 6.1.

6.1 Type I Hierarchical Matrices

In this subsection we consider the complexity when using matrices of Type I such as that illustrated in Figure 6.1. We have seen numerically that the rank r of the factorized parts will be approximately $\mathcal{O}(\log s(q))$ at level q. Allowing slightly stronger growth let us consider poly-logarithmic growth $\mathcal{O}(\log^\rho s(q))$ for some integer $\rho \geq 1$. Consider first the complexity of Algorithm 1. The cost for each subcube is, suppressing constants,

$$\mathcal{O}(r^2(\log n)^2 n) = \mathcal{O}((\log s(q))^{2\rho} \cdot (\log s(q))^2 \cdot s(q)^2) = \mathcal{O}((\log s(q))^{2\rho+2} \cdot s(q)^2).$$

Summing over 2^{3q} Schur complements at each level and Q levels in total, we obtain

$$\sum_{q=0}^{Q} (Q-q)^{2\rho+2} \cdot 2^{2(Q-q)} \cdot 2^{3q} = \mathcal{O}(2^{3Q}) = O(N).$$

In Algorithm 2, the dominant cost is the matrix vector multiplication in the hierarchical matrix form and for each cube on level q, it takes

$$\mathcal{O}(r^2(\log n) n) = \mathcal{O}((\log s(q))^{2\rho} \cdot (\log s(q)) \cdot s(q)^2) = \mathcal{O}((\log s(q))^{2\rho+1} \cdot s(q)^2)$$

	Type I		
$s(q)$	225	961	3969
A^{-1}	20	27	29
B	25	33	26
S	25	33	41
s	15	31	63

	Type II		
$s(q)$	225	961	3969
A^{-1}	63	122	234
B	75	161	-
S	75	161	331
s	15	31	63

Table 6.1: The maximum ranks observed for factorized submatrices using the Type I and Type II hierarchical matrix decompositions for $-\Delta u = f$ with $\epsilon_r = 10^{-6}$ and $\epsilon_a = 10^{-12}$. For Type I these grow like $\mathcal{O}(\log s(q))$ while for type II they grow like $\mathcal{O}(s(q))$.

steps. Therefore, the total cost over all subcubes and levels is equal to

$$\sum_{q=0}^{Q}(Q-q)^{2\rho+1}\cdot 2^{2(Q-q)}\cdot 2^{3q} = \mathcal{O}(2^{3Q}) = O(N)$$

again. In summary, both algorithms are of linear complexity.

6.2 Type II Hierarchical Matrices

In this subsection we consider the complexity when using matrices of Type II such as that illustrated in Figure 6.2. Consider first the complexity of Algorithm 1. Since the observed ranks are proportional to the segment lengths ($r \simeq s(q)$), the cost for each subcube will be

$$\mathcal{O}(r^2(\log n)^2 n) = \mathcal{O}(s(q)^2\cdot(\log s(q))^2\cdot s(q)^2) = \mathcal{O}((\log s(q))^2\cdot s(q)^4).$$

Summing over 2^{3q} Schur complements at each level and Q levels in total gives us

$$\sum_{q=0}^{Q}(Q-q)^2\cdot 2^{4(Q-q)}\cdot 2^{3q} = \mathcal{O}(Q^2 2^{4Q}) = O(N^{4/3}(\log N)^2).$$

Next let us consider the complexity of Algorithm 2. The matrix vector multiplication in the hierarchical matrix form for each cube takes

$$\mathcal{O}(r^2(\log n)n) = \mathcal{O}(s(q)^2\cdot(\log s(q))\cdot s(q)^2) = \mathcal{O}(\log s(q)\cdot s(q)^4)$$

steps. Therefore, the number of steps of Algorithm 2 in Type II case is equal to

$$\sum_{q=0}^{Q}(Q-q)\cdot 2^{4(Q-q)}\cdot 2^{3q} = \mathcal{O}(Q 2^{4Q}) = O(N^{4/3}\log N).$$

As we shall see in the numerical results displayed in the next section, although the first approach is asymptotically linear and the second is worse by a factor $\mathcal{O}(N^{1/3}\log^2 N)$, the second approach turns out to have better performance in the regime we could test. Eventually the algorithm with better scaling will provide better results but that may be for very large N (ignoring the $\log^2 N$ factor the growth rate of $\mathcal{O}(N^{4/3})$ means that if N increases by 8 then $N^{4/3}$ increases by about 16. Thus if the algorithm with worse scaling is C times faster for a particular N, it loses that advantage by the time N has increased by a factor of about C^3).

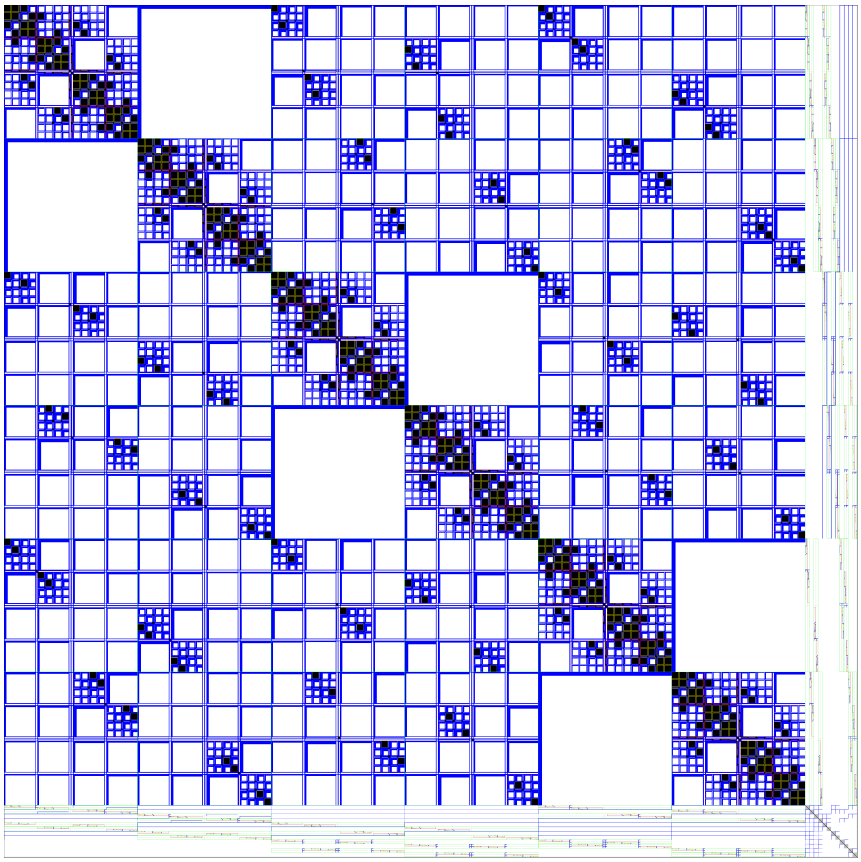


Fig. 6.1: Type I hierarchical division of the interaction matrix of a cube boundary (merged $S_{2;1,1,1}$) with 31×31 facets switching from hierarchical to dense matrices for size 7×7 sub-facets.

7 Numerical Results

All numerical tests are performed on a 2.13GHz processor. Execution times were measured in seconds for the *setup* phase (Algorithm 1) and the *solve* phase (Algorithm 2).

To test our algorithm, we setup the factorized form of M and use it to solve 100 random problems generated as follows: Select $x^\star \in \mathbb{R}^N$ as a random unit vector, and calculate $f = Mx^\star$ using the sparse original M. Then solve $Mx = f$ and determine the worst relative L_2 error

$$\frac{||\, x - x^\star \,||_2}{||\, x^\star \,||_2}$$

over the 100 samples.

Following [11] we construct the low rank approximations at the hierarchical levels using common matrix manipulations such as QR and SVD. During these procedures we keep only (the part of the decomposition corresponding to) those singular values

1. larger than the *absolute cutoff* ϵ_a and
2. within the *relative cutoff* ϵ_r of the largest singular value.

Addition and multiplication of hierarchical matrices also involves these kinds of *truncated SVD*. These two parameters, ϵ_a and ϵ_r, can be varied depending on the specific problem and the desired accuracy of the output.

In the first test, we solve $-\Delta u = f$ with zero Dirichlet boundary condition. In Table 7.1 we compare the runtimes obtained using the Type I and Type II hierarchical matrix decompositions with $\epsilon_a = 10^{-12}$ and $\epsilon_r = 10^{-6}$ and leaf level $P = 4$.

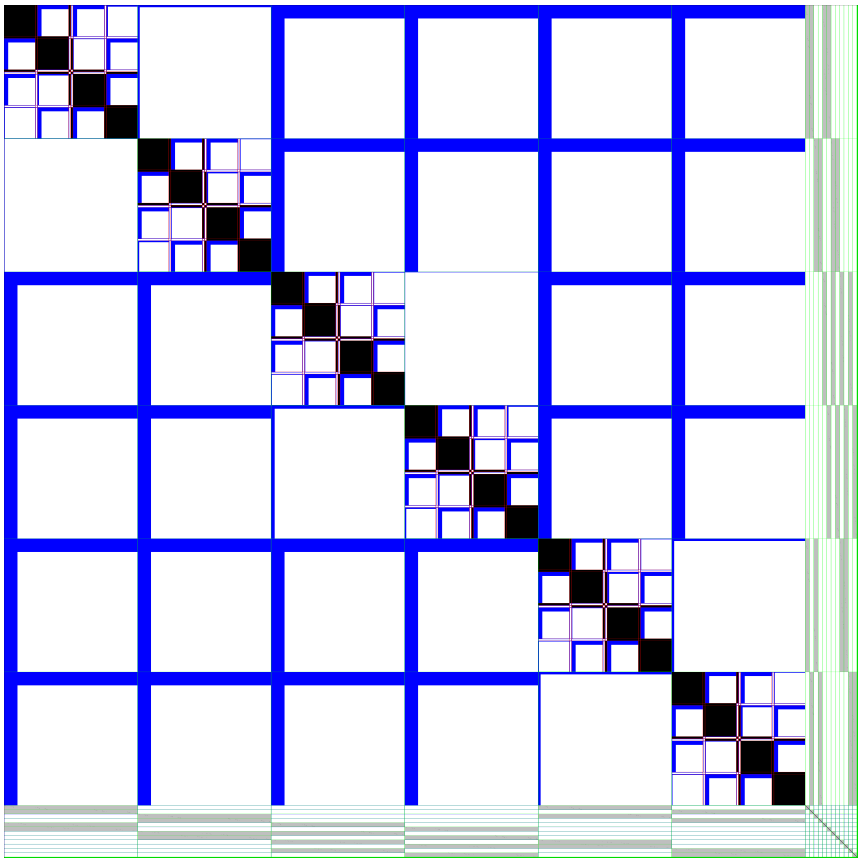


Fig. 6.2: Type II hierarchical division of the interaction matrix of a cube boundary (merged $S_{2;1,1,1}$) with 31×31 facets where the factorized form is used for all facet-facet interactions except self-interaction, switching from hierarchical to dense for 15×15 sub-facets.

In the actual implementation, dense matrix computations are used for the leaf level and the one level above that since at these two levels the matrix sizes are rather small so that dense linear algebra is more efficient than the hierarchical matrix algebra.

			Type I			Type II		
Q	N		Setup	Solve	Error	Setup	Solve	Error
3	31^3	29791	50.13	0.138	2.35e-07	32.25	0.132	8.94e-07
4	63^3	250047	1132.54	1.736	1.35e-06	551.52	1.405	2.13e-06
5	127^3	2048383	15777.77	17.471	4.03e-07	8315.93	13.719	9.75e-07

Table 7.1: A comparison of the Type I and Type II hierarchical matrix decomposition approach to solving $-\Delta u = f$ using $\epsilon_a = 10^{-12}$ and $\epsilon_r = 10^{-6}$ and $P = 4$ with dense matrices used for the lowest 2 levels.

Now, the scaling observed in our experiments is close to the expected theoretical value for Type II but not for type I. This is probably due to inefficiencies in our implementation but one clear reason is that the zero boundary conditions tend to reduce the amount of work required and disrupt the expected scaling. If we consider the top level calculation with $s = 63$ when $N = 63^3$ we see that no calculation of S is actually required while in the corresponding level of $N = 127^3$ there are

8 calculations of S required. Even away from the top level we should not expect the runtime for each level to scale exactly by 8 because the proportion of cubes touching the domain boundary (which induce fewer calculations because of the zero boundary conditions) will not be the same when we double the number of cubes along each axis. The scaling observed for an analogous calculation without the zero boundary condition is much closer to the expected value. In Table 7.2 we analyze the scaling level by level for the Type I results of Table 7.1.

	Setup			Ratios	
				$31^3 \to 63^3$	$63^3 \to 127^3$
N	31^3	63^3	127^3	8.4	8.2
Hierarchical	-	-	**3767.21**	-	-
Hierarchical	-	**408.79**	4752.30	-	**11.63**
Hierarchical	**34.45**	528.86	6071.92	**16.92**	10.42
Hierarchical	6.22	59.28	514.03	9.53	8.67
Dense	7.35	63.64	522.50	8.7	8.2
Dense	2.11	17.96	149.73	8.5	8.3

Table 7.2: A level by level breakdown of the runtime scaling for setup phase using the Type I hierarchical matrix decomposition approach to solving $-\Delta u = f$ using $\epsilon_a = 10^{-12}$ and $\epsilon_r = 10^{-6}$ and $P = 4$ with dense matrices used for the lowest 2 levels.

We see that the ratio for the dense levels is close to that for N. The ratio for the first hierarchical level, which is not the top level for any of our sizes, is also reasonable. Once we move to the next hierarchical level which is the top level for $N = 31^3$ (indicated in bold) but not for $N = 63^3$ we see the jump in scaling. The ratio for the same level for $63^3 \to 127^3$ is much closer to the theoretical value since then we are below the top level. Note that this top level anomaly was almost 17 for the $31^3 \to 63^3$ size change but decreases to less than 12 for $63^3 \to 127^3$. Also, the new top level for each size increase contributes to the observed runtime being more than 8 times the previous size – at size $N = 63^3$ it is 36% of the runtime and at size $N = 127^3$ it is 24% – as the top level contributes a smaller and smaller fraction of the runtime for larger and larger N both these effects should diminish.

We can also use the approach of [15] to calculate the diagonal of the inverse of M, which we call *diagonal extraction*. In Table 7.3 the error is given as the maximum difference between $M_{\lambda\lambda}$ as obtained via the diagonal extraction algorithm and via the solution of $Mu = e_\lambda$ for the 100 choices of e_λ made above.

			Type I		Type II	
Q		N	Extract	Error	Extract	Error
3	31^3	29791	35.20	1.43e-11	30.37	7.07e-11
4	63^3	250047	665.22	1.75e-09	680.85	5.08e-09
5	127^3	2048383	10231.10	7.33e-10	11957.85	1.16e-08

Table 7.3: A comparison of diagonal extraction using the Type I and Type II hierarchical matrix decomposition approach.

The additional error is almost 2 orders of magnitude smaller than that due to the usual setup and solve algorithms so that the error of diagonal extraction is essentially the same as that for Algorithm 2. The scaling for Type II is as expected but that for Type I is not. This is due to the same sorts of factors we discussed for the results of Algorithm 1 with the "top level anomaly" playing an even larger role here. Because of the zero boundary conditions there is almost no work ($\leq 1.2\%$ runtime for $N = 63^3$, $\leq 0.3\%$ for $N = 127^3$) done for the top level, while in the next larger calculation the corresponding level is one below the top and is one of the two largest contributers to the runtime ($\geq 30\%$). Again these effects should be mitigated as problem size increases.

In the final test, we solve $-\mathrm{div}(a(\mathbf{x})\nabla u) = f$ with a random $a(\mathbf{x})$, which is independently set at each node from a uniform distribution on $[10^{-3}, 10^3]$. In Table 7.4 we describe the results with again $\epsilon_a = 10^{-12}$ and $\epsilon_r = 10^{-6}$.

			Type I			Type II			Type I RanMult		
Q	N		Setup	Solve	Error	Setup	Solve	Error	Setup	Solve	Error
3	31^3	29791	27.24	0.111	5.97e-07	33.89	0.107	1.73e-06	27.23	0.109	5.97e-07
4	63^3	250047	691.41	1.412	4.52e-06	576.96	1.140	2.93e-06	629.19	1.308	4.94e-06
5	127^3	2048383	13193.96	15.917	2.69e-05	9840.03	11.483	6.30e-06	10268.84	13.761	2.69e-05

Table 7.4: To illustrate how we can handle non-identity $a(\mathbf{x})$ we use an $a(\mathbf{x})$ which is independently set at each node from a uniform distribution on $[10^{-3}, 10^3]$ with $\epsilon_a = 10^{-12}$ and $\epsilon_r = 10^{-6}$.

The block multiplications involved in the Schur complement computation $S = C - BA^{-1}B^t$ have many parts to the inner sum (recall the boundary has 98 components, and the majority of entries in the full S matrix correspond to the 24 child facets) for each entry in the final product. So for the entries in the hierarchical decomposition of S that are known to have factorized form we can circumvent the usual matrix multiplication procedure and use the idea of a probabilistic low rank approximation [14] to derive an approximation of certain blocks in the product without deriving any intermediate approximations. This would require a good estimate of the expected rank but that can be derived from preliminary calculations using the existing approach.

In the last set of results in Table 7.4 we use the Type I decomposition but this modified multiplication algorithm where block multiplication that results in a factorized result for the sum of products is computed using a randomized approach. Notice that this makes the Type I approach more competitive with the Type II approach for our problem sizes.

8 Conclusion and Future Work

This approach can be extended to unstructured meshes of tetrahedra and more general discretization schemes as we detailed for the two dimensional case in [21]. The correct generalizations of corners and segments are required because the subdomains may not meet in precisely the same regular manner as in the Cartesian case. Not only will the lengths of segments vary on the same level but some corners (analogous to the *generalized corner* of [21]) will need to be composed of multiple nodes to recover the relationship between the leaf elements of the decomposition. Also, a *generalized segment* will be needed because the pattern of 4 subdomains of tetrahedra meeting precisely along a segment will not hold for unstructured meshes – they may meet at a vertex which is only in tetrahedra from 3 different subdomains.

The algorithm can be adjusted to improve performance in the situation where $a(\mathbf{x})$ and/or $V(\mathbf{x})$ is perturbed locally and repeated calculations are required – a calculated factorization could be largely reused as only the parents of the cubes containing the vertices with changed values would have to be recalculated. Similarly one could reduce the amount of recalculation required in an h-adaptive mesh refinement system [5] as in [9] where new degrees of freedom could be incorporated incrementally and only the parts of the mesh which have been refined (and their parents in the decomposition tree) would require new calculations. The adaptive decomposition algorithm would have to be run alongside the h-refinement to distribute the new degrees of freedom and form new leaf cubes as required. Extending this idea to the degrees of freedom added and removed via a p-adaptive system one could eventually integrate with hp-adaptive systems.

We have not discussed parallelization of the calculations [16] but the hierarchical matrix algebra could be parallelized for the individual multiplications and additions. Also since all of the calculations on the same level are independent they could be performed in parallel. As long as there are more $\mathcal{D}_{q;i,j,k}$ per level than processors extensive inter-processor communication would be avoided. Only the uppermost levels would not lead to close to maximum speedup from parallelization. The possible gains of parallelization have been demonstrated by large scale parallel multifrontal solvers such as MUMPS [2].

We have only demonstrated the approach for piecewise linear elements but one could generalize the approach to work with different discretizations such as spectral elements or different methods such as the discontinuous Galerkin [3] method. The support of the bases could also be increased at the cost of thicker border elements between subcubes.

References

1. Amestoy, P.R., Davis, T.A., Duff, I.S.: An approximate minimum degree ordering algorithm. SIAM Journal on Matrix Analysis and Applications **17**(4), 886–905 (1996). DOI 10.1137/S0895479894278952
2. Amestoy, P.R., Duff, I.S., Vömel, C.: Task scheduling in an asynchronous distributed memory multifrontal solver. SIAM J. Matrix Anal. Appl. **26**(2), 544–565 (2004/05). DOI 10.1137/S0895479802419877
3. Arnold, D.N., Brezzi, F., Cockburn, B., Marini, L.D.: Unified analysis of discontinuous Galerkin methods for elliptic problems. SIAM Journal on Numerical Analysis **39**(5), 1749–1779 (2001/02). DOI 10.1137/S0036142901384162
4. Davis, T.A.: Direct methods for sparse linear systems. No. 2 in Fundamentals of algorithms. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2006)
5. Demkowicz, L., Kurtz, J., Pardo, D., Paszyński, M., Rachowicz, W., Zdunek, A.: Computing with *hp*-adaptive finite elements. Vol. 2. Chapman & Hall/CRC Applied Mathematics and Nonlinear Science Series. Chapman & Hall/CRC, Boca Raton, FL (2008). Frontiers: three dimensional elliptic and Maxwell problems with applications
6. Duff, I.S., Reid, J.K.: The multifrontal solution of indefinite sparse symmetric linear equations. ACM Transactions on Mathematical Software **9**(3), 302–325 (1983)
7. George, J.A.: Nested dissection of a regular finite element mesh. SIAM Journal on Numerical Analysis **10**(2), 345–363 (1973). DOI 10.1137/0710032
8. Grasedyck, L., Hackbusch, W.: Construction and Arithmetics of $\mathcal{H}$-matrices. Computing **70**(4), 295–334 (2003). DOI 10.1007/s00607-003-0019-1
9. Grasedyck, L., Hackbusch, W., Le Borne, S.: Adaptive geometrically balanced clustering of $\mathcal{H}$-matrices. Computing **73**(1), 1–23 (2004). DOI 10.1007/s00607-003-0042-2
10. Greengard, L., Gueyffier, D., Martinsson, P.G., Rokhlin, V.: Fast direct solvers for integral equations in complex three-dimensional domains. Acta Numerica **18**, 243–275 (2009). DOI 10.1017/S0962492906410011
11. Hackbusch, W., Grasedyck, L., Börm, S.: An introduction to hierarchical matrices. Tech. Rep. 21, Max-Plank-Instituit für Mathematik in den Naturwissenschaften, Leipzig (2001)
12. Hackbusch, W., Khoromskij, B.N., Kriemann, R.: Hierarchical matrices based on a weak admissibility criterion. Computing **73**(3), 207–243 (2004). DOI 10.1007/s00607-004-0080-4
13. Hendrickson, B., Rothberg, E.: Improving the run time and quality of nested dissection ordering. SIAM Journal on Scientific Computing **20**(2), 468–489 (1998). DOI 10.1137/S1064827596300656
14. Liberty, E., Woolfe, F., Martinsson, P.G., Rokhlin, V., Tygert, M.: Randomized algorithms for the low-rank approximation of matrices. Proceedings of the National Academy of Sciences of the USA **104**(51), 20,167–20,172 (2007). DOI 10.1073/pnas.07096-40104
15. Lin, L., Lu, J., Ying, L., Car, R., E, W.: Fast algorithm for extracting the diagonal of the inverse matrix with application to the electronic structure analysis of metallic systems. Communications in Mathematical Sciences **7**(3), 755–777 (2009)
16. Lin, L., Yang, C., Lu, J., Ying, L., E, W.: A fast parallel algorithm for selected inversion of structured sparse matrices with applications to 2D electronic structure calculations. Tech. Rep. LBNL-2677E, Lawrence Berkeley National Lab (2009)
17. Liu, J.W.H.: The multifrontal method for sparse matrix solution: Theory and practice. SIAM Review **34**(1), 82–109 (1992)
18. Martinsson, P.G.: A fast direct solver for a class of elliptic partial differential equations. Journal of Scientific Computing **38**(3), 316–330 (2009). DOI 10.1007/s10915-008-9240-6
19. Martinsson, P.G., Rokhlin, V.: A fast direct solver for boundary integral equations in two dimensions. Journal of Computational Physics **205**(1), 1–23 (2005). DOI 10.1016/j.jcp.2004.10.033
20. Saad, Y.: Iterative Methods for Sparse Linear Systems, 2 edn. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2003)
21. Schmitz, P.G., Ying, L.: A fast direct solver for elliptic problems on general meshes in 2D. Preprint
22. Xia, J., Chandrasekaran, S., Gu, M., Li, X.S.: Fast algorithms for hierarchically semiseparable matrices. Numerical Linear Algebra with Applications (2009). DOI 10.1002/nla.691
23. Xia, J., Chandrasekaran, S., Gu, M., Li, X.S.: Superfast multifrontal method for large structured linear systems of equations. SIAM Journal on Matrix Analysis and Applications **31**(3), 1382–1411 (2009). DOI 10.1137/09074543X