COMMUN. MATH. SCI.
Vol. 7, No. 3, pp. 755–777

FAST ALGORITHM FOR EXTRACTING THE DIAGONAL OF THE INVERSE MATRIX WITH APPLICATION TO THE ELECTRONIC STRUCTURE ANALYSIS OF METALLIC SYSTEMS *

LIN LIN†, JIANFENG LU†, LEXING YING‡, ROBERTO CAR§, AND WEINAN E†¶

Abstract. We propose an algorithm for extracting the diagonal of the inverse matrices arising from electronic structure calculation. The proposed algorithm uses a hierarchical decomposition of the computational domain. It first constructs hierarchical Schur complements of the interior points for the blocks of the domain in a bottom-up pass and then extracts the diagonal entries efficiently in a top-down pass by exploiting the hierarchical local dependence of the inverse matrices. The overall cost of our algorithm is $O(N^{3/2})$ for a two dimensional problem with N degrees of freedom. Numerical results in electronic structure calculation illustrate the efficiency and accuracy of the proposed algorithm.

Key words. Diagonal extraction, hierarchical Schur complement, electronic structure calculation.

AMS subject classifications. 65F30, 65Z05.

1. Introduction

1.1. Motivation The focus of this paper is fast algorithms for extracting the diagonal of the inverse of a given matrix. The particular application in our mind is electronic structure calculation, especially for models based on effective one-electron Hamiltonians such as the tight-binding models or density functional theory models. Given a matrix H, coming from the discretization of a Hamiltonian operator, one wants to evaluate

$$\rho = 2\,\mathrm{diag}\,\phi_{\mathrm{FD}}(H-\mu) = \mathrm{diag}\,\frac{2}{1+e^{\beta(H-\mu)}}. \tag{1.1}$$

Here H is the Hamiltonian given by $H = -\frac{1}{2}\Delta + V(x)$ with a potential $V(x)$, ϕ_{FD} is the Fermi-Dirac function: $\phi(z) = 1/(1+e^{\beta z})$, β is the inverse temperature, and μ is the chemical potential but can also be understood as a Lagrange multiplier for the constraint that the total density equals to the total number of electrons in the system: $\int \rho dx = K$ where K is the number of electrons in the system.

Direct evaluation of (1.1) requires the diagonalization of the matrix H, and hence results in a $\mathcal{O}(N^3)$ algorithm, where N is the dimension of the matrix H. Developing algorithms with less cost has attracted a lot of attention in the past two decades. Much progress has been made for developing more efficient algorithms for insulating systems. But developing better algorithms for metallic systems has remained to be a big challenge [12].

Among the various ideas proposed, methods based on polynomial and rational expansion of the Fermi-Dirac function [1, 7, 12, 16] are particularly promising, since

*Received: May 21, 2009; accepted: July 29, 2009. Communicated by Shi Jin.
†Program in Applied and Computational Mathematics, Princeton University, Princeton, NJ 08544.
‡Department of Mathematics and ICES, University of Texas at Austin, 1 University Station/C1200, Austin, TX 78712.
§Department of Chemistry and Princeton Center for Theoretical Science, Princeton University, Princeton, NJ 08544.
¶Department of Mathematics, Princeton University, Princeton, NJ 08544.

they provide a unified framework for both insulating and metallic systems. The Fermi operator can be represented using the pole expansion (Matsubara expansion in physical terms) [18]:

$$\rho = 1 - \frac{4}{\beta}\,\mathrm{diag}\,\mathrm{Re}\,\sum_{l=1}^{\infty}\frac{1}{(H-\mu)-(2l-1)\pi i/\beta}. \tag{1.2}$$

The details and features of this representation will be summarized in section 3 below. Using this, the problem is reduced to evaluating the diagonal elements of a series of inverse matrices with poles on the imaginary axis. We also remark here that a more efficient rational approximation for the Fermi-Dirac function was recently constructed in [19].

1.2. Related work

For a typical quantum chemistry problem, the domain is taken to be a periodic box $[0,n]^d$ (d is the dimension) after normalization, $V(x)$ is a potential that oscillates on the $O(1)$ scale, and μ is often on the order of $O(n^d)$. As a result, the operator $(H-\mu)$ is far from being positive definite. In many computations, the Hamiltonian is sampled with a constant number of points per unit length. Therefore, the discretization of $(H-\mu)-(2l-1)i\pi/\beta$, denoted by M, is a matrix of dimension $N=\mathcal{O}(n^d)$. In view of (1.2), we are interested in extracting the diagonal of its inverse matrix $G=M^{-1}$.

The most obvious algorithm is to first calculate the whole inverse matrix G, and then extract the diagonal trivially. Of course, naive inversion leads to an algorithm that scales as $\mathcal{O}(N^3)$, the same as diagonalization. A better strategy would be to explore the structure of the inverse matrix, and use an iterative method such as Newton-Schulz iteration. For an insulating system, for which it is known that the matrix elements of G decay exponentially fast away from the diagonal and hence can be truncated, Newton-Schulz iteration gives rise to an $\mathcal{O}(N)$ algorithm, as was already discussed in [18]. However, such algorithms cannot be used for metallic systems since the elements of G decay slowly away from the diagonal. Consequently, such a truncation will introduce large error. Therefore, one needs to explore other representation of the inverse matrix G.

For the case when M is a positive definite matrix, several ingenious approaches have been developed to represent and manipulate G efficiently. One strategy is to represent M and G using multi-resolution basis like wavelets [3, 4]. It is well known that for positive definite M, the wavelet basis offers an optimally sparse representation for both M and $G=M^{-1}$. Together with the Newton-Schulz iteration, it gives rise to a linear scaling algorithm for calculating the inverse matrix G from M. In one dimension, assuming that we use L levels of wavelet coefficients and if we truncate the matrix elements that correspond to the wavelets which are centered at locations with distance larger than R, then the the cost of matrix-matrix multiplication is roughly $\mathcal{O}(R^2L^3N)$. In $2D$, a naive extension using the tensor product structure will lead to a complexity of $\mathcal{O}(R^4L^6N)$. This has linear scaling, but the prefactor is rather large: Consider a moderate situation with $R=10$ and $L=8$, R^4L^6 is on the order of 10^9. This argument is rather crude, but it does reveal a paradoxical situation with the wavelet representation: although in principle linear scaling algorithms can be derived using wavelets, they are not practical in $2D$ and $3D$, unless much more sophisticated techniques are developed to reduce the prefactor.

Another candidate for positive definite M is to use the hierarchical matrices [5, 14]. The main observation is that the off-diagonal blocks of the matrix G are

numerically low-rank and thus can be approximated hierarchically using low rank factorizations. The cost of multiplying and inverting hierarchical matrices scales as $\mathcal{O}(N)$. Therefore, by either combining with the Newton-Schulz iteration, or directly inverting the hierarchical matrices with block LU decomposition, one obtains a linear scaling algorithm.

Both of these two approaches are quite successful for M being positive definite. Unfortunately as we pointed out earlier, for the application to electronic structure analysis, our matrix M is far from being positive definite. In fact, the matrix elements of G are highly oscillatory due to the shift of chemical potential in the Hamiltonian. Consequently, the inverse matrix G does not have an efficient representation in either the wavelet basis or the hierarchical matrix framework.

So far, we have focused our discussion on constructing the inverse matrix G. Yet another line of research is the Monte Carlo approach for extracting the diagonal using only the matrix vector product $Gv = M^{-1}v$. One algorithm of this type was introduced in [2]. This type of algorithm can be quite efficient if fast algorithms for calculating Gv are available (without the explicit knowledge of G). Examples of such fast algorithms include multigrid methods [6], the fast multipole method [13] (combined with iterative solvers), and the discrete symbol calculus [9], to name a few. One shortcoming of this approach is that the accuracy is relatively low for general G due to its Monte Carlo nature. The accuracy can be greatly improved if G is banded or its elements decay exponentially fast away from the diagonal. However, neither of these conditions are satisfied for the kind of systems we are interested in.

1.3. Main observation and contribution of this work

Our algorithm is based on the following observation for extracting a particular part of the inverse matrix of a symmetric matrix M. For clarity, let us assume for the moment that M comes from the discretization of a differential operator in a two-dimensional domain, on a grid shown in figure 1.1. Assume that the points in the domain are indexed by $\mathcal{I} = \{1, \cdots, N\}$, and $\mathcal{I}$ is partitioned into three disjoint sets $\mathcal{I} = \mathcal{I}_1\mathcal{I}_2\mathcal{I}_3$,[1] as shown in figure 1.1. The matrix elements of M between $\mathcal{I}_1$ and $\mathcal{I}_3$ are zero: $M(\mathcal{I}_1, \mathcal{I}_3) = M(\mathcal{I}_3, \mathcal{I}_1) = 0$, i.e., M takes the form

$$M = \begin{pmatrix} A & B & 0 \\ B^T & C & D \\ 0 & D^T & E \end{pmatrix}. \tag{1.3}$$

For instance, if M is obtained from a finite difference discretization of the Hamiltonian operator using a five-point stencil or a tight-binding model with nearest-neighbor interaction, then M is of the form (1.3) if the domain is partitioned as in figure 1.1.

Let $G = M^{-1}$. Assume that we are only interested in $G(\mathcal{I}_1, \mathcal{I}_1)$. For instance, $G(\mathcal{I}_1, \mathcal{I}_1)$ contains the diagonal elements of the inverse matrix for the vertices in $\mathcal{I}_1$. We can calculate $G(\mathcal{I}_1, \mathcal{I}_1)$ using block Gaussian elimination:

$$G = \begin{pmatrix} I & -A^{-1}B & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} A^{-1} & 0 & 0 \\ 0 & & \\ 0 & & G_1 \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ -B^T A^{-1} & I & 0 \\ 0 & 0 & I \end{pmatrix}, \tag{1.4}$$

where

$$G_1 = M_1^{-1}, M_1 = \begin{pmatrix} C & D \\ D^T & E \end{pmatrix} - \begin{pmatrix} B^T A^{-1} B & 0 \\ 0 & 0 \end{pmatrix}. \tag{1.5}$$

[1] Here and in the following, we will use the notation $\mathcal{I}\mathcal{J}$ to denote the union of the two index sets

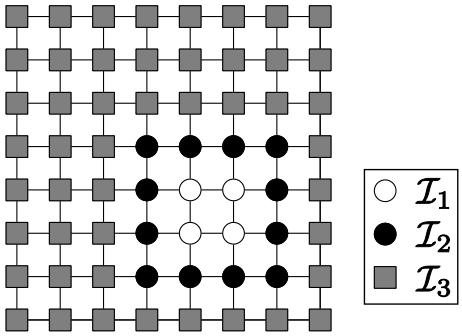

FIG. 1.1. *Partition of the domain. The corresponding matrix M has the structure of* (1.3).

Here M_1 is the Schur complement of A, and G_1 is the inverse of M_1. Note that M_1 differs from $M(\mathcal{I}_2\mathcal{I}_3,\mathcal{I}_2\mathcal{I}_3)$ only for matrix elements that represent interactions between grid points in $\mathcal{I}_2$. The interaction between points in $\mathcal{I}_2$ and $\mathcal{I}_3$ and the self-interaction inside $\mathcal{I}_3$ are unchanged.

From (1.4) one can compute the matrix G explicitly as

$$G = \begin{pmatrix} A^{-1} + \left(-A^{-1}B\ 0\right) G_1 \left(-A^{-1}B\ 0\right)^T & \left(-A^{-1}B\ 0\right) G_1 \\ G_1 \left(-A^{-1}B\ 0\right)^T & G_1 \end{pmatrix}. \tag{1.6}$$

Therefore

$$\begin{aligned} G(\mathcal{I}_1,\mathcal{I}_1) &= A^{-1} + \left(-A^{-1}B\ 0\right) G_1 \left(-A^{-1}B\ 0\right)^T \\ &= A^{-1} + \left(A^{-1}B\right) \cdot G_1(\mathcal{I}_2,\mathcal{I}_2) \cdot \left(B^T A^{-1}\right). \end{aligned} \tag{1.7}$$

In order to compute $G(\mathcal{I}_1,\mathcal{I}_1)$, one does not need the whole inverse G_1 of the Schur complement. It is only necessary to know the value of G_1 on the domain that has interaction in the direct matrix M, i.e., $G_1(\mathcal{I}_2,\mathcal{I}_2)$. The observation can be summarized as

$$G(\mathcal{I}_1,\mathcal{I}_1) \textit{ is determined by } G_1(\mathcal{I}_2,\mathcal{I}_2).$$

Note that, in terms of $G_1(\mathcal{I}_2,\mathcal{I}_2)$, our problem is again to seek a particular part of an inverse matrix, but with the direct matrix changed from M to M_1. Thus the same strategy can be used recursively. For example, if $\mathcal{I}_3$ can be further decomposed into two regions $\mathcal{I}_4$ and $\mathcal{I}_5$, with $\mathcal{I}_5$ not interacting directly with $\mathcal{I}_2$: $M_1(\mathcal{I}_2,\mathcal{I}_5) = 0$. Then similarly, $G_1(\mathcal{I}_2,\mathcal{I}_2)$ can be computed if we have the inverse matrix of the Schur complement restricted to the region $\mathcal{I}_4$. This suggests a hierarchical Schur complements method based on a hierarchical domain decomposition strategy [11, 10, 20] in order to compute all diagonal elements of the inverse matrix of M.

We would like to mention that Li et al[17] have recently proposed an algorithm that has a similar spirit to our approach. However, our algorithm is derived from a different viewpoint, and the numerical results suggest that our approach is more efficient for the problems considered here.

The rest of the paper is organized as follows. In section 2, the details of the algorithm for extracting the diagonal of the inverse matrix are presented. We consider application to electronic structure calculation in section 3. Section 4 contains the numerical results.

2. The algorithm

The algorithm for extracting the diagonal of an inverse matrix has two steps. The first step is bottom-up. Starting with a hierarchical decomposition of the physical domain, we construct the hierarchy of Schur complements by block Gaussian elimination. The idea of constructing a hierarchy of Schur complements is not new. It has been used in the context of numerical linear algebra for quite some time. They date back to George [11] and were extended by various algorithms, e.g. multifrontal method [10, 20]. Our presentation illustrates a more geometric picture for the construction. The next step, which is new and the main contribution of this paper, is a top-down pass in which we extract the diagonal of the inverse matrix using the hierarchy of Schur complements constructed. Here, we do not need the whole inverse matrix for the Schur complements, but only the diagonal blocks.

We will focus on $2D$ systems and postpone the discussion of $3D$ systems to the end of this subsection.

2.1. Construction of the hierarchy of Schur complements

To explain the algorithm, we take our computational domain to be a 16×16 lattice, indexed by the index set $\mathcal{J}_0$ (e.g., row major ordering). For the sake of clarity, we assume that the domain is partitioned into disjoint blocks hierarchically: on the top level (Level 3), the domain is partitioned into 4 blocks, and each block is further decomposed into 4 sub-blocks at the lower level. We stop at the bottom level (Level 1), where the domain is partitioned into $4\times 4=16$ blocks, as shown in figure 2.1. A more efficient version of the algorithm actually uses blocks with shared edges and is implemented for real calculation. We will comment on this improvement later in this section.

2.1.1. Level 1

The domain is partitioned into $4\times 4=16$ blocks. In each block, we distinguish "interior points" that do not interact directly with points in the other blocks, from "boundary points" that may interact with points in the other blocks. We denote by $\mathcal{I}_{1;ij}$ the index set of the interior points for each block (white points in figure 2.1), and $\mathcal{J}_{1;ij}$ the index set of the boundary points of each block (black points in figure 2.1). It follows that $M(\mathcal{I}_{1;ij},\mathcal{I}_{1;i'j'}\mathcal{J}_{1;i'j'})=0$ if $(i,j)\neq(i',j')$.

We will use block Gaussian elimination to eliminate the interior points and reduce our problem to the boundary points only. To do so, let us permute the rows and columns of the matrix M such that all the interior points are ordered before the boundary points, i.e. the order of indices is changed from $\mathcal{J}_0$ to

$$\mathcal{J}_0 \xrightarrow{P_1} (\mathcal{I}_{1;11}\mathcal{I}_{1;12}\cdots\mathcal{I}_{1;44}|\mathcal{J}_{1;11}\mathcal{J}_{1;12}\cdots\mathcal{J}_{1;44}). \tag{2.1}$$

The notation of $|$ is used to separate interior points and boundary points. This step is done by introducing a permutation matrix P_1 such that for the matrix M defined according to the index set $\mathcal{J}_0$, and

$$M_1 = P_1^{-1} M P_1 \tag{2.2}$$

is indexed according to $(\mathcal{I}_1|\mathcal{J}_1)$. Here

$$\mathcal{I}_1 = \mathcal{I}_{1;11}\mathcal{I}_{1;12}\cdots\mathcal{I}_{1;44}$$

is the union of all the interior points, and similarly for $\mathcal{J}_1$.

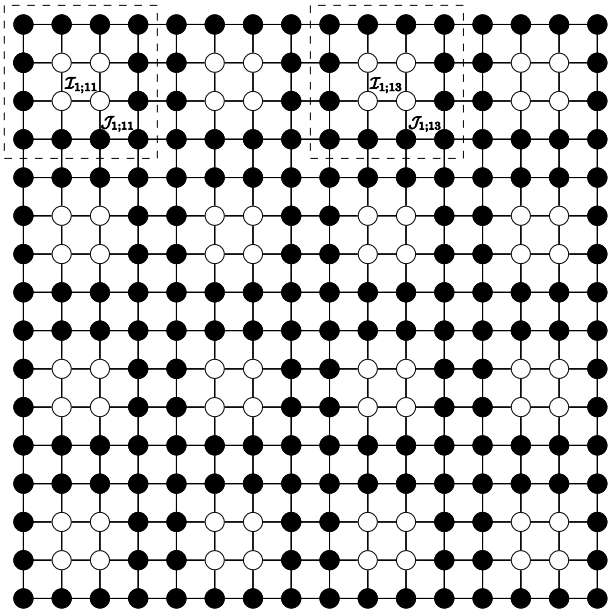

FIG. 2.1. *The first level in the hierarchical domain decomposition. The interior points are denoted as white points and boundary points are denoted as black points. The dash lines denote blocks in the decomposition (only two are shown in the figure). In this and following figures, the domain is partitioned into disjoint blocks, however, we emphasize again that in the actual implementation, we use blocks with shared edges.*

Let us write M_1 as

$$M_1 = P_1^{-1} M P_1 = \begin{pmatrix} A_1 & B_1 \\ B_1^T & C_1 \end{pmatrix}. \tag{2.3}$$

The interior points that belong to different blocks do not interact with each other, and thus A_1 is a block diagonal matrix.

$$A_1 = M_1(\mathcal{I}_1, \mathcal{I}_1) = \begin{pmatrix} A_{1;11} & & & \\ & A_{1;12} & & \\ & & \ddots & \\ & & & A_{1;44} \end{pmatrix} \tag{2.4}$$

with

$$A_{1;ij} = M_1(\mathcal{I}_{1;ij}, \mathcal{I}_{1;ij}). \tag{2.5}$$

Interior points inside each block only interact with the boundary points in the same block, hence B_1 is also block diagonal:

$$B_1 = M_1(\mathcal{I}_1, \mathcal{J}_1) = \begin{pmatrix} B_{1;11} & & & \\ & B_{1;12} & & \\ & & \ddots & \\ & & & B_{1;44} \end{pmatrix} \tag{2.6}$$

with

$$B_{1;ij} = M_1(\mathcal{I}_{1;ij}, \mathcal{J}_{1;ij}). \tag{2.7}$$

We do not assume any structure for C_1, and just write

$$C_1 = M_1(\mathcal{J}_1, \mathcal{J}_1). \tag{2.8}$$

Since A_1 is block diagonal, its inverse is given by

$$A_1^{-1} = \begin{pmatrix} A_{1;11}^{-1} & & & \\ & A_{1;12}^{-1} & & \\ & & \ddots & \\ & & & A_{1;44}^{-1} \end{pmatrix}. \tag{2.9}$$

Therefore, by Gaussian elimination, the inverse of M_1^{-1} is given by

$$M_1^{-1} = \begin{pmatrix} A_1 & B_1 \\ B_1^T & C_1 \end{pmatrix}^{-1} = \begin{pmatrix} I & -A_1^{-1}B_1 \\ 0 & I \end{pmatrix} \begin{pmatrix} A_1^{-1} & 0 \\ 0 & (C_1 - B_1^T A_1^{-1} B_1)^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -B_1^T A_1^{-1} & I \end{pmatrix}. \tag{2.10}$$

To simplify notation, let us denote

$$L_1 = \begin{pmatrix} I & 0 \\ B_1^T A_1^{-1} & I \end{pmatrix}. \tag{2.11}$$

Since B_1^T and A_1^{-1} are both block diagonal matrices, L_1 can also be computed independently within each block, as

$$B_1^T A_1^{-1} = \begin{pmatrix} B_{1;11}^T A_{1;11}^{-1} & & & \\ & B_{1;12}^T A_{1;12}^{-1} & & \\ & & \ddots & \\ & & & B_{1;44}^T A_{1;44}^{-1} \end{pmatrix}. \tag{2.12}$$

Moreover, the matrix $B_1^T A_1^{-1} B_1$ is also block diagonal

$$B_1^T A_1^{-1} B_1 = \begin{pmatrix} B_{1;11}^T A_{1;11}^{-1} B_{1;11} & & & \\ & B_{1;12}^T A_{1;12}^{-1} B_{1;12} & & \\ & & \ddots & \\ & & & B_{1;44}^T A_{1;44}^{-1} B_{1;44} \end{pmatrix}. \tag{2.13}$$

Using equation (2.10), we have

$$G = P_1 M_1^{-1} P_1^{-1} = P_1 L_1^T \begin{pmatrix} A_1^{-1} & 0 \\ 0 & G_1^{-1} \end{pmatrix} L_1 P_1^{-1}, \tag{2.14}$$

where G_1 is the Schur complement of A_1, given by

$$G_1 = (C_1 - B_1^T A_1^{-1} B_1)^{-1}. \tag{2.15}$$

We have now eliminated the interior points, and the problem reduces to a smaller matrix $C_1 - B_1^T A_1^{-1} B_1$.

2.1.2. Level 2

At the second level, the problem is to find G_1, defined on the index set $\mathcal{J}_1$ (boundary points or black points in the first level). At this level, the domain is decomposed into 4 blocks, we again distinguish interior points with boundary points in each block so that interior points only interact with points in the same block. As in the first level, we reindex $\mathcal{J}_1$ by a permutation matrix P_2 as

$$\mathcal{J}_1 \xrightarrow{P_2} (\mathcal{I}_{2;11}\mathcal{I}_{2;12}\cdots\mathcal{I}_{2;22}|\mathcal{J}_{2;11}\mathcal{J}_{2;12}\cdots\mathcal{J}_{2;22}) = (\mathcal{I}_2|\mathcal{J}_2). \tag{2.16}$$

Here $\mathcal{I}_{2;ij}$ are interior points (white points) and $\mathcal{J}_{2;ij}$ are boundary points (black points) inside the block (i,j) (see figure 2.2).

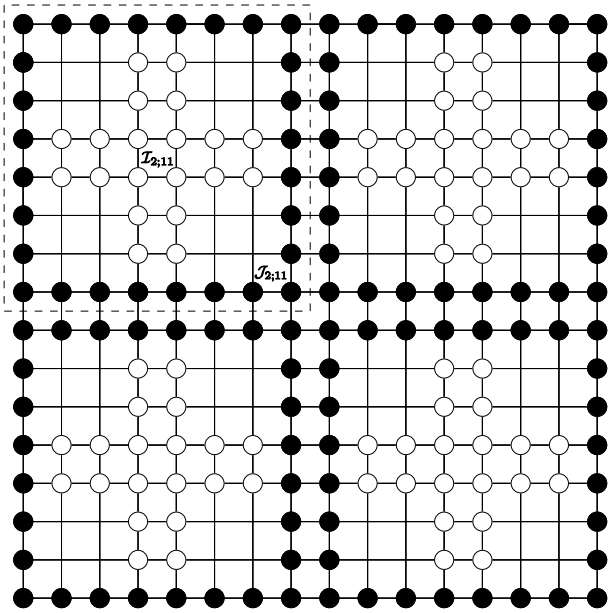

FIG. 2.2. *The second level in the hierarchical domain decomposition. The interior points are denoted as white points and boundary points are denoted as black points. The dash lines denote blocks in the decomposition (only one is shown in the figure).*

Following the same strategy as in the first level, denote

$$M_2 = P_2^{-1}\left(C_1 - B_1^T A_1^{-1} B_1\right) P_2 = \begin{pmatrix} A_2 & B_2 \\ B_2^T & C_2 \end{pmatrix} \tag{2.17}$$

with

$$A_2 = M_2(\mathcal{I}_2,\mathcal{I}_2), \qquad B_2 = M_2(\mathcal{I}_2,\mathcal{J}_2), \qquad C_2 = M_2(\mathcal{J}_2,\mathcal{J}_2). \tag{2.18}$$

A_2, B_2 are block diagonal matrices (since in the first level, the update for C_1 is block diagonal). Then

$$M_2^{-1} = \begin{pmatrix} A_2 & B_2 \\ B_2^T & C_2 \end{pmatrix}^{-1} = L_2^T \begin{pmatrix} A_2^{-1} & 0 \\ 0 & G_2 \end{pmatrix} L_2 \tag{2.19}$$

with

$$L_2 = \begin{pmatrix} I & 0 \\ B_2^T A_2^{-1} & I \end{pmatrix}, \qquad G_2 = (C_2 - B_2^T A_2^{-1} B_2)^{-1}. \tag{2.20}$$

Combining the decomposition of G_1 in the second level, we obtain

$$G = P_1 L_1^T \begin{pmatrix} A_1^{-1} & 0 \\ 0 & P_2 L_2^T \begin{pmatrix} A_2^{-1} & 0 \\ 0 & G_2 \end{pmatrix} L_2 P_2^{-1} \end{pmatrix} L_1 P_1^{-1}. \tag{2.21}$$

2.1.3. Level 3

At the top level, we set the interior and boundary points as shown in figure 2.3. Again, we reindex the points in $\mathcal{J}_2$ into $\mathcal{I}_3$ and $\mathcal{J}_3$, by a permutation matrix P_3:

$$\mathcal{J}_2 \xrightarrow{P_3} (\mathcal{I}_{3;11} | \mathcal{J}_{3;11}) = (\mathcal{I}_3 | \mathcal{J}_3). \tag{2.22}$$

The final formula for G is

$$G = P_1 L_1^T \begin{pmatrix} A_1^{-1} & 0 \\ 0 & P_2 L_2^T \begin{pmatrix} A_2^{-1} & 0 \\ 0 & P_3 L_3^T \begin{pmatrix} A_3^{-1} & 0 \\ 0 & G_3 \end{pmatrix} L_3 P_3^{-1} \end{pmatrix} L_2 P_2^{-1} \end{pmatrix} L_1 P_1^{-1} \tag{2.23}$$

with

$$G_3 = (C_3 - B_3^T A_3^{-1} B_3)^{-1}. \tag{2.24}$$

At the top level, G_3 is inverted directly.

2.1.4. Summary of construction

In order to construct a hierarchy of Schur complements for a matrix M on a domain of size $N \times N$, we start with a hierarchical domain decomposition scheme. At each level, the points inside each block are distinguished into interior and boundary points, so that interior points only interact with points inside the block. We reindex the points accordingly. At each level, we eliminate the interior points. This reindexing procedure is described as in figure 2.4.

We define for each level

$$G_i = \begin{cases} G = M^{-1}, & i = 0, \\ (C_i - B_i^T A_i^{-1} B_i)^{-1}, & i > 0. \end{cases} \tag{2.25}$$

The inverse of the Schur complements have the following recursive relation G_{l-1} and G_l as

$$G_{l-1} = P_l L_l^T \begin{pmatrix} A_l^{-1} & 0 \\ 0 & G_l \end{pmatrix} L_l P_l^{-1}. \tag{2.26}$$

Using this procedure, starting from the bottom level, we can construct the hierarchy of Schur complements.

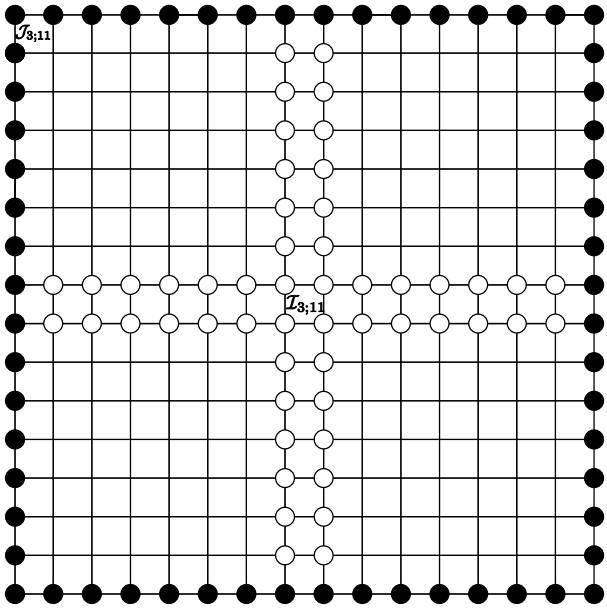

FIG. 2.3. *The top level in the hierarchical domain decomposition. The interior points are denoted as white points and boundary points are denoted as black points.*

$$
\begin{array}{l}
\qquad\quad \mathcal{J}_0 \\
P_1^{-1}\Big\| \Big\backslash P_1 \\
(\mathcal{I}_1|\mathcal{J}_1) = (\mathcal{I}_{1;11}\mathcal{I}_{1;12}\cdots\mathcal{I}_{1;44}|\mathcal{J}_{1;11}\mathcal{J}_{1;12}\cdots\mathcal{J}_{1;44}) \\
P_2^{-1}\Big\| \Big\backslash P_2 \\
(\mathcal{I}_2|\mathcal{J}_2) = (\mathcal{I}_{2;11}\mathcal{I}_{2;12}\cdots\mathcal{I}_{2;22}|\mathcal{J}_{2;11}\mathcal{J}_{2;12}\cdots\mathcal{J}_{2;22}) \\
P_3^{-1}\Big\| \Big\backslash P_3 \\
(\mathcal{I}_3|\mathcal{J}_3) = (\mathcal{I}_{3;11}|\mathcal{J}_{3;11})
\end{array}
$$

FIG. 2.4. *The index structure for the hierarchical domain decomposition*

2.2. Extracting the diagonal

Having obtained the hierarchy of Schur complements, we now apply the observation made in the introduction to extract the diagonal elements of the inverse matrix G. The key is that we do not need to calculate the whole inverse of the Schur complements G_l at each level.

Our observation, in more precise terms, states that

$$G_{l-1}(\mathcal{I}_{l;ij}\mathcal{J}_{l;ij},\mathcal{I}_{l;ij}\mathcal{J}_{l;ij}) \text{ is determined by } G_l(\mathcal{J}_{l;ij},\mathcal{J}_{l;ij}).$$

Therefore, at each level l, we only need to compute $G_l(\mathcal{J}_{l;ij},\mathcal{J}_{l;ij})$, instead of the whole matrix G_l. This enables us to develop an algorithm for extracting the diagonal elements of G with a cost that scales as $\mathcal{O}(N^{3/2})$ (instead of N^3).

2.2.1. Level 3 The extraction process starts from the top level (Level 3 in the toy example). At the third level, given that G_3 is computed directly, G_2 is given by the formula

$$G_2 = P_3 L_3^T \begin{pmatrix} A_3^{-1} & 0 \\ 0 & G_3 \end{pmatrix} L_3 P_3^{-1} = P_3 \begin{pmatrix} A_3^{-1} + A_3^{-1} B_3 G_3 B_3^T A_3^{-1} & -A_3^{-1} B_3 G_3 \\ -G_3 B_3^T A_3^{-1} & G_3 \end{pmatrix} P_3^{-1}. \tag{2.27}$$

The matrix in the middle is indexed by $(\mathcal{I}_{3;11}|\mathcal{J}_{3;11})$. Due to the permutation matrix P_3, G_2 is indexed according to

$$\mathcal{J}_2 = \mathcal{J}_{2;11}\mathcal{J}_{2;12}\mathcal{J}_{2;21}\mathcal{J}_{2;22}. \tag{2.28}$$

We are climbing up the ladder in figure 2.4.

Since only $G_2(\mathcal{J}_{2;ij}, \mathcal{J}_{2;ij})$ is necessary for extracting the diagonal and we do not need the off-diagonal blocks, we will write G_2 as

$$G_2 = \begin{pmatrix} G_{2;11} & * & * & * \\ * & G_{2;12} & * & * \\ * & * & G_{2;21} & * \\ * & * & * & G_{2;22} \end{pmatrix} \tag{2.29}$$

with $G_{2;ij} = G_2(\mathcal{J}_{2;ij}, \mathcal{J}_{2;ij})$.

2.2.2. Level 2

Proceeding to the second level, we now have

$$G_1 = P_2 L_2^T \begin{pmatrix} A_2^{-1} & 0 \\ 0 & G_2 \end{pmatrix} L_2 P_2^{-1} = P_2 \begin{pmatrix} A_2^{-1} + A_2^{-1} B_2 G_2 B_2^T A_2^{-1} & -A_2^{-1} B_2 G_2 \\ -G_2 B_2^T A_2^{-1} & G_2 \end{pmatrix} P_2^{-1}. \tag{2.30}$$

Note that A_2^{-1} and B_2 are all block diagonal matrices, we have

$$A_2^{-1} B_2 G_2 B_2^T A_2^{-1} = \begin{pmatrix} A_{2;11}^{-1} B_{2;11} G_{2;11} B_{2;11}^T A_{2;11}^{-1} & * & * & * \\ * & A_{2;12}^{-1} B_{2;12} G_{2;12} B_{2;12}^T A_{2;12}^{-1} & * & * \\ * & * & A_{2;21}^{-1} B_{2;21} G_{2;21} B_{2;21}^T A_{2;21}^{-1} & * \\ * & * & * & A_{2;22}^{-1} B_{2;22} G_{2;22} B_{2;22}^T A_{2;22}^{-1} \end{pmatrix}, \tag{2.31}$$

$$A_2^{-1} B_2 G_2 = \begin{pmatrix} A_{2;11}^{-1} B_{2;11} G_{2;11} & * & * & * \\ * & A_{2;12}^{-1} B_{2;12} G_{2;12} & * & * \\ * & * & A_{2;21}^{-1} B_{2;21} G_{2;21} & * \\ * & * & * & A_{2;22}^{-1} B_{2;22} G_{2;22} \end{pmatrix}. \tag{2.32}$$

The matrix in the middle of the right hand side of (2.30) is indexed by $(\mathcal{I}_2|\mathcal{J}_2)$. By applying the permutation matrix P_2, we climbed up another step in the ladder of figure 2.4, now G_1 is indexed by

$$\mathcal{J}_1 = \mathcal{J}_{1;11}\mathcal{J}_{1;12}\cdots\mathcal{J}_{1;44}, \tag{2.33}$$

and we only need to keep the diagonal blocks $G_1(\mathcal{J}_{1;ij}, \mathcal{J}_{1;ij})$.

It is helpful to see again why $G_1(\mathcal{J}_{1;i'j'},\mathcal{J}_{1;i'j'})$ is determined by the part of G_2 that we have kept. Assume that the block (i',j') at level 1 is contained in block (i,j) at level 2. By (2.30), (2.31) and (2.32), we have

$$G_1(\mathcal{I}_{2;ij}\mathcal{J}_{2;ij},\mathcal{I}_{2;ij}\mathcal{J}_{2;ij}) = \begin{pmatrix} A_{2;ij}^{-1} + A_{2;ij}^{-1}B_{2;ij}G_{2;ij}B_{2;ij}^T A_{2;ij}^{-1} & -A_{2;ij}^{-1}B_{2;ij}G_{2;ij} \\ -G_{2;ij}B_{2;ij}^T A_{2;ij}^{-1} & G_{2;ij} \end{pmatrix}. \tag{2.34}$$

Therefore, to calculate $G_1(\mathcal{I}_{2;ij}\mathcal{J}_{2;ij},\mathcal{I}_{2;ij}\mathcal{J}_{2;ij})$, only $G_{2;ij} = G_2(\mathcal{J}_{2;ij},\mathcal{J}_{2;ij})$ is needed. Of course, since the index set $\mathcal{J}_{1;i'j'}$ is a subset of $\mathcal{I}_{2;ij}\mathcal{J}_{2;ij}$, $G_1(\mathcal{J}_{1;i'j'},\mathcal{J}_{1;i'j'})$ is also computed. Note that, again, we only need to compute and keep the diagonal blocks of G_1, indexed by $\mathcal{J}_1$ since this is the part we will need at the next level.

2.2.3. Level 1

Following the same procedure as in Level 2, one can compute $G(\mathcal{J}_{0;ij},\mathcal{J}_{0;ij})$ just by knowing $G_1(\mathcal{J}_{1;ij},\mathcal{J}_{1;ij})$ obtained from Level 2. Note that $\mathcal{J}_0$ is indeed the index set for all the points in the domain, the diagonal elements of G is simply obtained by combining all the diagonal elements of $G(\mathcal{J}_{0;ij},\mathcal{J}_{0;ij})$ together.

2.3. Summary

We start with a hierarchical decomposition of the domain. At each level from the top, the points inside each block are divided into two groups: interior points and boundary points. Interior points only interact with points in the same block, and are eliminated using block Gaussian elimination. Necessary permutation of the indices is done by a permutation matrix P_l as shown in figure 2.4. At each level, the computational procedure for each block is independent, and hence can be trivially parallelized. The recursive relation between the inverse of the Schur complements $\{G_l\}$ for different levels is given by (2.26).

For the extraction step, one goes backwards starting from the top level. We do not compute the whole inverse matrix of the Schur complement G_l, but only its diagonal blocks $G_l(\mathcal{J}_{l;ij},\mathcal{J}_{l;ij})$. We emphasize that these diagonal blocks are sufficient for the computation of the diagonal blocks on the next level. Since the ordering of G_{l-1} is different from that of G_l, we also need to apply the permutation matrix in order to get back to the ordering in level $l-1$, as shown in figure 2.4.

Finally, at the lowest level, we obtain $G(\mathcal{J}_{0;ij},\mathcal{J}_{0;ij})$, and hence the diagonal elements of G.

We can organize the whole algorithm in Algorithm 1. Note that the reindexing is implicitly included in the algorithm, when we use the index sets $\mathcal{J}_{l;ij}$ for G_l and use the index sets $\mathcal{I}_{l;ij}$ and $\mathcal{J}_{l;ij}$ for M_l.

2.4. Complexity

To analyze the complexity of the proposed algorithm, assume that the domain contains $N = \sqrt{N}\times\sqrt{N}$ points. Set $\sqrt{N} = 2^L$ and the number of levels in the hierarchical domain decomposition is given by $l_{\mathrm{MAX}} < L$.

First consider the construction step. Denoting the number of blocks at level l by $n_B(l)$, and we have

$$n_B(l) = 2^{2(l_{\mathrm{MAX}}-l)}. \tag{2.35}$$

Also denote the number of points that each block contains as $n_P(l)$ (note that the interior points of the previous levels have been already eliminated), we have

$$n_P(l) = \begin{cases} 2^{2(L+1-l_{\mathrm{MAX}})}, & l=1; \\ 2^{L+l-l_{\mathrm{MAX}}+3}-16, & l>1. \end{cases} \tag{2.36}$$

Algorithm 1 Extracting the diagonal of M^{-1}

1: Determine l_{MAX} and decompose domain hierarchically.
2: Generate index sets $\mathcal{I}_{l;ij}$ and $\mathcal{J}_{l;ij}$.
3: $M_1 \leftarrow M$.
4: **for** $l = 1$ to l_{MAX} **do**
5: $M_{l+1} \leftarrow M_l(\mathcal{J}_l, \mathcal{J}_l)$.
6: **for** $(i,j) \in \{$block index at level $l\}$ **do**
7: $A_{l;ij} \leftarrow M_l(\mathcal{I}_{l;ij}, \mathcal{I}_{l;ij})$.
8: $B_{l;ij} \leftarrow M_l(\mathcal{I}_{l;ij}, \mathcal{J}_{l;ij})$.
9: Calculate $A_{l;ij}^{-1}$.
10: Calculate $K_{l;ij} \leftarrow -B_{l;ij}^T A_{l;ij}^{-1}$.
11: Calculate $M_{l+1}(\mathcal{J}_{l;ij}, \mathcal{J}_{l;ij}) \leftarrow M_{l+1}(\mathcal{J}_{l;ij}, \mathcal{J}_{l;ij}) + K_{l;ij} B_{l;ij}$.
12: **end for**
13: **end for**
14: Calculate $G_{l_{\mathrm{MAX}}} \leftarrow M_{l_{\mathrm{MAX}}+1}^{-1}$.
15: **for** $l = l_{\mathrm{MAX}}$ to 1 **do**
16: **for** $(i,j) \in \{$block index at level $l\}$ **do**
17: Calculate $G_{l-1}(\mathcal{I}_{l;ij}, \mathcal{I}_{l;ij}) \leftarrow A_{l;ij}^{-1} + K_{l;ij}^T G_l(\mathcal{J}_{l;ij}, \mathcal{J}_{l;ij}) K_{l;ij}$.
18: Calculate $G_{l-1}(\mathcal{I}_{l;ij}, \mathcal{J}_{l;ij}) \leftarrow K_{l;ij}^T G_l(\mathcal{J}_{l;ij}, \mathcal{J}_{l;ij})$.
19: $G_{l-1}(\mathcal{J}_{l;ij}, \mathcal{I}_{l;ij}) \leftarrow G_{l-1}(\mathcal{I}_{l;ij}, \mathcal{J}_{l;ij})^T$.
20: $G_{l-1}(\mathcal{J}_{l;ij}, \mathcal{J}_{l;ij}) \leftarrow G_l(\mathcal{J}_{l;ij}, \mathcal{J}_{l;ij})$.
21: **end for**
22: **end for**

At level l, for each block, we need to invert $A_{l;ij}$ (step 9 in Algorithm 1), multiply the inverse with $B_{l;ij}^T$ to get $K_{l;ij}$ (step 10), and then update $M_{l+1}(\mathcal{J}_{l;ij}, \mathcal{J}_{l;ij})$ (step 11). The computational cost for these steps is $\mathcal{O}(n_P(l)^3)$. Since there are $n_B(l)$ blocks, the total cost for level l is $\mathcal{O}(n_B(l) n_P(l)^3)$.

Since

$$\begin{aligned}\sum_{l=1}^{l_{\mathrm{MAX}}} n_B(l) n_P(l)^3 &\le 2^{2(l_{\mathrm{MAX}}-1)} 2^{6(L+1-l_{\mathrm{MAX}})} + \sum_{l=2}^{l_{\mathrm{MAX}}} 2^{2(l_{\mathrm{MAX}}-l)} 2^{3(L+l-l_{\mathrm{MAX}}+3)} \\ &\le C\left(2^{6L-4l_{\mathrm{MAX}}} + \sum_{l=2}^{l_{\mathrm{MAX}}} 2^{3L+l-l_{\mathrm{MAX}}}\right) \\ &\le C\big(2^{6L-4l_{\mathrm{MAX}}} + 2^{3L}\big),\end{aligned}$$

where C is a constant independent of L and l_{MAX}. Let $l_{\mathrm{MAX}} = \mathcal{O}(L)$ (e.g., $l_{\mathrm{MAX}} = L - 1$, so that at the first level, each block has 16 points inside), the total computational cost for the construction step is $\mathcal{O}(N^{3/2})$.

The dimension of the matrix $M_{l_{\mathrm{MAX}}+1}$ is $\mathcal{O}(N^{1/2})$ (since there are $\mathcal{O}(N^{1/2})$ points on the boundary for the whole domain), hence the cost of the inversion step 14 in Algorithm 1 is also $\mathcal{O}(N^{3/2})$.

Now, let us look at the extraction step. At level l, for each block, we need to calculate $G_{l-1}(\mathcal{I}_{l;ij}, \mathcal{I}_{l;ij})$ (step 17) and $G_{l-1}(\mathcal{I}_{l;ij}, \mathcal{J}_{l;ij})$ (step 18), the computational cost for these two steps is clearly $\mathcal{O}(n_P(l)^3)$. Hence, the total computational cost for level l is $\mathcal{O}(n_B(l) n_P(l)^3)$. A similar calculation as in the construction step shows that the complexity for the extraction step is also $\mathcal{O}(N^{3/2})$.

Therefore, the total complexity for Algorithm 1 is $\mathcal{O}(N^{3/2})$. We would like to emphasize that no approximation is taken in our algorithm and all calculations are exact. Like the multifrontal algorithms, our approach relies only on the sparsity structure of M.

So far the presentation assumes that all the blocks on the same level are disjoint from each other. As we pointed out earlier, a more efficient version uses a decomposition in which two adjacent blocks share an edge. The modifications are fairly straightforward, the algorithm remains almost the same, and the computational complexity is still $\mathcal{O}(N^{3/2})$. However, the prefactor is significantly reduced. For example, the structure of the domain decomposition on the second level now takes the form shown in figure 2.5 as opposed to figure 2.2 in the disjoint case (periodic boundary condition is used). From figure 2.5, it is clear that the number of interior nodes to be eliminated is roughly halved, and as a result we gain a speedup factor of 5–8 in practice.

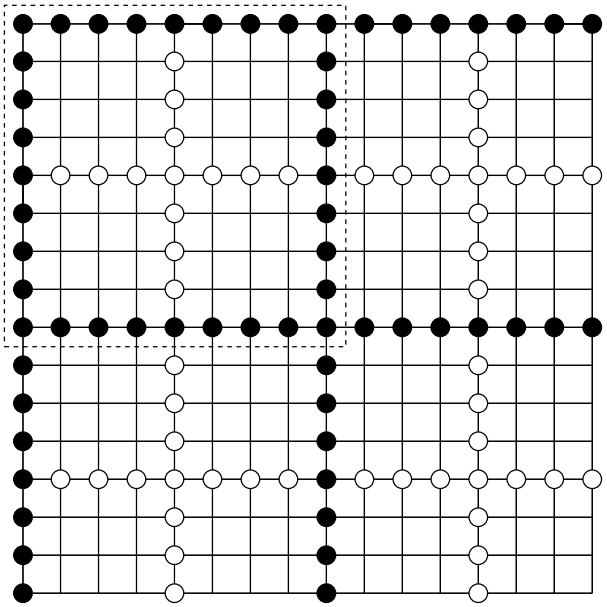

FIG. 2.5. *The second level in the hierarchical domain decomposition using blocks with shared edges. The interior points are denoted as white points and boundary points are denoted as black points. The dash lines denote blocks in the decomposition (only one is shown in the figure). A periodic boundary condition is assumed.*

2.5. Extensions

2.5.1. Generalization to higher order finite difference stencil

The above discussion assumes that in the matrix M, a point only interacts with its nearest neighbors. In other words, the matrix M comes from a discretization of five-point stencil. It is straightforward to extend the algorithm for higher order stencils: at each level, for each block, there will be fewer interior points, since the interaction range becomes longer. The computational cost will still be $\mathcal{O}(N^{3/2})$, however, the prefactor will be increased, since there will be more boundary points, and hence matrices of larger dimension.

For this reason, it is of interest to use compact stencils instead of non-compact ones. For example, if a compact nine-point stencil is used, the structure of the interior points will remain the same as for the five-point stencil. However, for compact stencils, the above algorithm needs some slight modifications as we discuss now.

Consider

$$(-\Delta+V)u=f. \tag{2.37}$$

Denote by D_5 the discretization of the Laplacian using the five-point stencil and D_9 the discretization using a compact nine-point stencil (both understood as a matrix). Using the five-point stencil discretization, we have the discretized equation

$$D_5u_h+V_hu_h=f_h, \tag{2.38}$$

where V_h is a diagonal matrix with diagonal entries given by V evaluated at the grid. The Green's function is given by $(D_5+V_h)^{-1}$, therefore, we may extract the diagonal of the discrete Green's function using Algorithm 1. For a compact nine-point stencil, however, the discretized problem becomes

$$D_9u_h=(I+\tfrac{1}{12}D_5)(-V_hu_h+f_h), \tag{2.39}$$

or, moving u_h terms to the left,

$$(D_9+(I+\tfrac{1}{12}D_5)V_h)u_h=(I+\tfrac{1}{12}D_5)f_h. \tag{2.40}$$

The matrix $M=D_9+(I+\frac{1}{12}D_5)V_h$ has the same structure as a nine-point stencil: a point only interacts with its surrounding eight points. However, the discrete Green's function is given by $M^{-1}(I+\frac{1}{12}D_5)$, therefore, the diagonal of the Green's function, which is what we want, is not any more the diagonal elements of M^{-1}, but the diagonal elements of the product matrix.

Fortunately, since the matrix $I+\frac{1}{12}D_5$ is a matrix for which points only interacting with their nearest-neighbors, to evaluate the diagonal of $M^{-1}(I+\frac{1}{12}D_5)$, we only need the matrix elements of M^{-1} that represent interaction between the grid points with itself (the diagonal) and with its nearest-neighbors.

Let us go back to (2.34) in the step of extracting the diagonal, note that when going from an upper level to a lower level, the matrix elements between the boundary points at the upper level will not be changed (the bottom-right block on the right hand side is still $G_{2;ij}$). Therefore, the matrix elements between the boundary points at level l is determined by G_l, and will not be changed when we go to lower levels.

Therefore, to extract the matrix elements of G between a point with its neighbors, we still proceed in the same way as we did for extracting the diagonal. However, now at level l, if the point (say x) and one of its neighbors (say y) are both in the set of boundary points $\mathcal{J}_l$, we keep the matrix elements of G_l between these two points. By the observation above, we have $G(x,y)=G_l(x,y)$. At the bottom level, for each block, we have actually calculated $G(\mathcal{I}_{1;ij}\mathcal{J}_{1;ij},\mathcal{I}_{1;ij}\mathcal{J}_{1;ij})$ for each block (i,j). We have obtained the elements of the inverse matrix between the points in each block. Therefore, for every point x, the matrix elements between that represent self-interaction and interaction with its neighbors have been calculated from the G_l we obtained at different levels, and hence, the diagonal of the matrix $M^{-1}(I+\frac{1}{12}D_5)$ can be calculated.

The extension to higher order compact stencil (e.g., 25-point compact stencil) is also straightforward. The algorithm can be obtained by slight modifications of the algorithm for higher order non-compact stencils as discussed above.

2.5.2. Linear scaling algorithm for elliptic problem

We can exploit additional structure of the matrices involved in the algorithm, and further reduce the computational cost. For instance, in the case that the matrix M comes from the discretization of an elliptic operator, it is known that for the intermediate matrices, the off-diagonal blocks can be approximated by a low-rank submatrix, known as hierarchical matrices [14] or hierarchical semi-separable matrices [8, 22]. For such matrices, the complexity of inversion and multiplication operations is $\mathcal{O}(n)$, where n is the dimension of the matrix. If these approximations have been used, the computational cost for the algorithm is reduced to $\mathcal{O}(N)$. Hence we obtain a linear scaling algorithm for extracting the diagonal of the inverse matrix.

2.5.3. Generalization to three dimension

For a quasi-$2D$ system, namely, a $3D$ system with size $\sqrt{N}\times\sqrt{N}\times m$, where $m=\mathcal{O}(1)$, it is clear that the algorithm described above can be easily generalized. We decompose the domain according to the two significant dimensions, and proceed with hierarchical Schur complements such that the interior points are eliminated at each level. It is not hard to see that the complexity of such an extension will still be $\mathcal{O}(N^{3/2})$ with prefactor depending on m. This would be the cost, for example, if the algorithm is used on a Hamiltonian describing graphene sheets.

For real $3D$ systems with $\sqrt[3]{N}\times\sqrt[3]{N}\times\sqrt[3]{N}$ points (bulk system), the extension of the algorithm starts with a hierarchical domain decomposition in three dimension: at the top level, the domain is partitioned into 8 blocks with equal volume, and then each block is partitioned into 8 blocks in the lower level. The algorithm is similar to the $2D$ algorithm, at each level, the interior points in each block are eliminated. Going from the bottom level up to the top level, the problem is reduced to a problem defined on the surface of the system. Now, on the top level, the matrix M is of dimension $\mathcal{O}(N^{2/3})$, since we have $\mathcal{O}(N^{2/3})$ points on the surface. The computational cost of the algorithm in $3D$ will be $\mathcal{O}(N^2)$. While this naive extension to $3D$ still has a better scaling then the direct inversion, which costs $\mathcal{O}(N^3)$, the gain in performance is not as substantial as in $2D$.

3. Application to electronic structure

In density functional theory, at finite temperature, the system is completely determined by the two-body density matrix P, which is represented by the Fermi operator

$$P=\frac{2}{1+e^{\beta(H-\mu)}}. \tag{3.1}$$

Here $\beta=1/k_BT$ is the inverse temperature and μ is the chemical potential of the system. The factor 2 comes from spin degeneracy. The diagonal elements of P in real space, denoted by n, is called electronic density profile, and quantifies important concepts such as chemical bonds *etc.* Another quantity of interest in electronic structure analysis is the energy defined by

$$E=\mathrm{Tr}[PH]. \tag{3.2}$$

Electronic energy directly gives rise to potential energy surface for nuclei and plays an central role in *ab initio* quantum chemistry.

Recently, a pole expansion algorithm was proposed in [18] (also called Matsubara expansion in theoretical many body physics [21]) for electronic structure analysis. The Fermi operator (3.1) is represented as a sum of infinite number of poles located

on the imaginary axis

$$P = 1 - \frac{4}{\beta}\mathrm{Re}\sum_{l=1}^{\infty}\frac{1}{(H-\mu)-(2l-1)\pi i/\beta}. \tag{3.3}$$

$1/[H-\mu-(2l-1)\pi i/\beta]$ is called the l-th pole of the Fermi operator, and the imaginary number $\mu_l = (2l-1)\pi i/\beta$ is called the Matsubara frequency. We remark that a similar expansion is obtained in [16] based on grand canonical potential formalism. Other rational expansions based on contour integrals, for example, those proposed in [23] and [15] can also be used in this context.

Pole expansion cannot be calculated directly since (3.3) contains an infinite number of terms. In practice, (3.3) is split into a pole part and a tail part

$$P = -\frac{4}{\beta}\mathrm{Re}\sum_{l=1}^{N_{\mathrm{Pole}}}\frac{1}{H-\mu-\mu_l} + P_{\mathrm{tail}}(H;N_{\mathrm{Pole}}). \tag{3.4}$$

The tail part is therefore defined as

$$P_{\mathrm{tail}}(H;N_{\mathrm{Pole}}) = 1 - \frac{4}{\beta}\mathrm{Re}\sum_{l=N_{\mathrm{Pole}}+1}^{\infty}\frac{1}{H-\mu-\mu_l}. \tag{3.5}$$

It should be emphasized that the tail part also has an explicit formula [18]

$$P_{\mathrm{tail}}(H;N_{\mathrm{Pole}}) = 1 - \frac{2}{\pi}\mathrm{Im}\ \psi\left(N_{\mathrm{Pole}} - \frac{1}{2} + \frac{i}{2\pi}\beta(H-\mu)\right). \tag{3.6}$$

Here, ψ is the digamma function $\psi(x) = \Gamma'(x)/\Gamma(x)$.

Using (3.4), we have the formula for electronic density

$$\rho = -\frac{4}{\beta}\mathrm{Re}\sum_{l=1}^{N_{\mathrm{Pole}}}\mathrm{diag}\left[\frac{1}{H-\mu-\mu_l}\right] + \mathrm{diag}\left[P_{\mathrm{tail}}(H;N_{\mathrm{Pole}})\right]. \tag{3.7}$$

For each pole, the diagonal of the matrix involved can be readily calculated using Algorithm 1. The tail part can be calculated efficiently using standard Fermi operator expansion method, e.g., Chebyshev expansion with truncated matrix-matrix multiplication [12]. In each matrix-matrix multiplication, we truncate the resulting matrix A (on a two dimensional domain) with cutoff radius R_{Cut}, i.e.,

$$A[(i,j),(i',j')] = 0 \qquad \text{if } |i-i'| > R_{\mathrm{Cut}} \text{ or } |j-j'| > R_{\mathrm{Cut}}. \tag{3.8}$$

The electronic energy for the system is the sum of the diagonal elements of PH, and we have

$$\begin{aligned} E = \mathrm{Tr}[PH] &= \mathrm{Tr}\left[-\frac{4}{\beta}\mathrm{Re}\sum_{l=1}^{N_{\mathrm{Pole}}}\left(\frac{1}{H-\mu-\mu_l}H\right) + P_{\mathrm{tail}}(H;N_{\mathrm{Pole}})H\right] \\ &= \mathrm{Tr}\left[-\frac{4}{\beta}\mathrm{Re}\sum_{l=1}^{N_{\mathrm{Pole}}}\left(1+\frac{\mu+\mu_l}{H-\mu-\mu_l}\right) + P_{\mathrm{tail}}(H;N_{\mathrm{Pole}})H\right] \\ &= -\frac{4N_{\mathrm{Pole}}}{\beta} - \frac{4}{\beta}\mathrm{Re}\sum_{l=1}^{N_{\mathrm{Pole}}}(\mu+\mu_l)\,\mathrm{Tr}\left[\frac{1}{H-\mu-\mu_l}\right] + \mathrm{Tr}[P_{\mathrm{tail};N_{\mathrm{Pole}}}(H)H]. \end{aligned} \tag{3.9}$$

The first term in (3.9) is a scalar. The second term has already been computed in the calculation of ρ. For the third term, since $P_{\text{tail}}(H;N_{\text{Pole}})$ is expressed as polynomials of H, another factor of H can be easily incorporated. Once the density function ρ is calculated, the energy can also be calculated very efficiently.

We summarize the whole algorithm to calculate the electronic density and energy in Algorithm 2.

Algorithm 2 Pole expansion based electronic structure algorithm

1: Determine N_{Pole}. Set electronic density profile ρ to be a zero vector, and electronic energy $E=0$.
2: **for** $l=1$ to N_{Pole} **do**
3: Calculate ρ_p (the diagonal of the l-th pole) using Algorithm 1.
4: Update density profile $\rho \leftarrow \rho - \frac{4}{\beta}\text{Re } \rho_p$.
5: Update electronic energy $E \leftarrow \frac{4}{\beta}\left\{1-\int dx \text{Re } [(\mu+\mu_l)\rho_p]\right\}$.
6: **end for**
7: Determine the target accuracy for Chebyshev expansion and R_{Cut}.
8: Calculate N_{Cheb} and corresponding Chebyshev expansion coefficients $C_k, k=1,\ldots,N_{\text{Cheb}}$.
9: Set $T_1=I, T_2=H$.
10: Set $P_{\text{tail}}=C_1T_1+C_2T_2$.
11: **for** $k=3$ to N_{Cheb} **do**
12: Calculate $T_3 \leftarrow 2HT_2-T_1$ using matrix matrix multiplication with truncation at a cutoff radius R_{Cut}.
13: Update $P_{\text{tail}} \leftarrow P_{\text{tail}}+C_kT_3$.
14: $T_1 \leftarrow T_2; T_2 \leftarrow T_3$.
15: **end for**
16: Update density profile $\rho \leftarrow \rho + \text{diag}[P_{\text{tail}}]$.
17: Update electronic energy with another truncated matrix matrix multiplication $E \leftarrow E + \text{Tr}[P_{\text{tail}}H]$.

We have analyzed the complexity for extracting the diagonal in section 2. The computational cost for the tail part is determined by two quantities: one is the order of Chebyshev polynomials needed to represent the scalar valued function $P_{\text{tail}}(x;N_{\text{Pole}})$, denoted by N_{Cheb}. The other is the cutoff radius during matrix-matrix multiplication R_{Cut}. It is easy to see that the complexity is $\mathcal{O}(N_{\text{Cheb}}R_{\text{Cut}}^4N)$. It has been shown in [18] that N_{Cheb} is given by $\beta\Delta E/N_{\text{Pole}}$, where β is the inverse temperature of the system and ΔE is the spectrum width of the discretized Hamiltonian. This can be understood as by taking out N_{Pole} poles, the temperature for the tail part is effectively increased by a factor of N_{Pole}.

The truncation radius R_{Cut} depends on the decay of the off diagonal elements of P_{tail}. By the same observation of effective temperature increase, the decay is exponential with rate proportional to N_{Pole}/β, since this is the decay behavior for the density matrix for metal at finite temperature (see for example [12]). Hence, for a given accuracy requirement, R_{Cut} decays exponentially with increasing N_{Pole}. Note that the decay rate does not depend on system size, and therefore the cutoff radius required does not depend on system size either.

Hence, the complexity of Algorithm 2 is

$$\mathcal{O}\Big(N_{\text{Pole}}N^{3/2}+\frac{\beta\Delta E}{N_{\text{Pole}}}N\exp(-CN_{\text{Pole}}/\beta)\Big). \tag{3.10}$$

We remark that although in principle one can directly use a Chebyshev polynomial approximation for the original Fermi operator (set $N_{\mathrm{Pole}}=0$) and achieve a linear scaling algorithm, the prefactor will be quite large for metallic system. We need a higher order Chebyshev polynomial and a larger R_{Cut} since the density matrix decays much slowly (recall that R_{Cut} decays exponentially with increasing N_{Pole}). It is much preferable in practice to use a larger N_{Pole}. This will be further illustrated in the next section.

4. Numerical results

We test our algorithm on the Anderson model. The computational domain is a $\sqrt{N}\times\sqrt{N}$ lattice with periodic boundary condition. Under the nearest neighbor tight binding approximation, the matrix components of the Hamiltonian of the system can be written as following. All parameters are reported using atomic units.

$$H_{i'j';ij}=\begin{cases}2+V_{ij}, & i'=i,j'=j,\\ -1/2+V_{ij}, & i'=i\pm1,j'=j \text{ or } i'=i,j'=j\pm1.\end{cases} \tag{4.1}$$

The on-site potential energy V_{ij} is chosen to be a uniform random number between 0 and 10^{-3}. The temperature is 300K. The chemical potential μ is set to satisfy the condition that

$$\mathrm{Tr}\,P=N_{\mathrm{Electron}}. \tag{4.2}$$

N_{Electron} is proportional to system size. For example, for a 32×32 system, we choose $N_{\mathrm{Electron}}=32$; therefore for a 64×64 system, $N_{\mathrm{Electron}}=128$ *etc.* The spectrum width $\Delta E\approx4$, the inverse temperature $\beta=1/k_BT\approx1000$, and hence $\beta\Delta E\approx4000$. The random potential gives rise to an energy gap of the order 5×10^{-4}, which is comparable to the thermal energy $k_BT\approx10^{-3}$. With this tiny energy gap and low temperature, the system is metallic in nature and the density matrix is a full matrix. Figure 4.1 visualizes the typical behavior of density matrix for this system. The domain size is 128×128. The plotted part is $P(r_0,r)$, with $r_0=(1,1)$, and $r=(1,j)$, $j=1,\ldots,128$. It can be seen that the density matrix decays slowly in the off-diagonal direction.

Table 4.1 shows the computational time and accuracy of Algorithm 1, compared to the $O(N^3)$ scaling dense matrix inversion algorithm, which is also referred to as *direct inversion*. The computations are carried out with a MATLAB code on a machine with Intel Xeon 3GHz CPU with 64GB memory. The accuracy is measured by relative error of the diagonal elements in L^1 norm. Direct inversion clearly scales as $\mathcal{O}(N^3)$. Algorithm 1 scales slightly better than $\mathcal{O}(N^{3/2})$. This is due to the fact that in the bottom level our algorithm is indeed $\mathcal{O}(N)$, and the $\mathcal{O}(N^{3/2})$ part only dominates at large N. Moreover, Algorithm 1 is already faster than direct inversion starting from $\sqrt{N}=32$, in which case one can gain a speedup of a factor around 35. Note that it takes only about 3 minutes for Algorithm 1 in the case of $\sqrt{N}=1024$, while direct inversion becomes impractical starting from $\sqrt{N}=256$.

As for accuracy, Table 4.1 shows that the error introduced by the pole part is at the level of machine accuracy. Therefore the error for the entire algorithm comes from the tail part. In order to show that the tail part can also be computed effectively, we first test our algorithm for $\sqrt{N}=32$.

We measure the accuracy by two quantities. One is the error of the energy per electron defined by

$$\Delta\epsilon_{\mathrm{rel}}=\frac{|\widehat{E}-E|}{N_{\mathrm{Electron}}}. \tag{4.3}$$

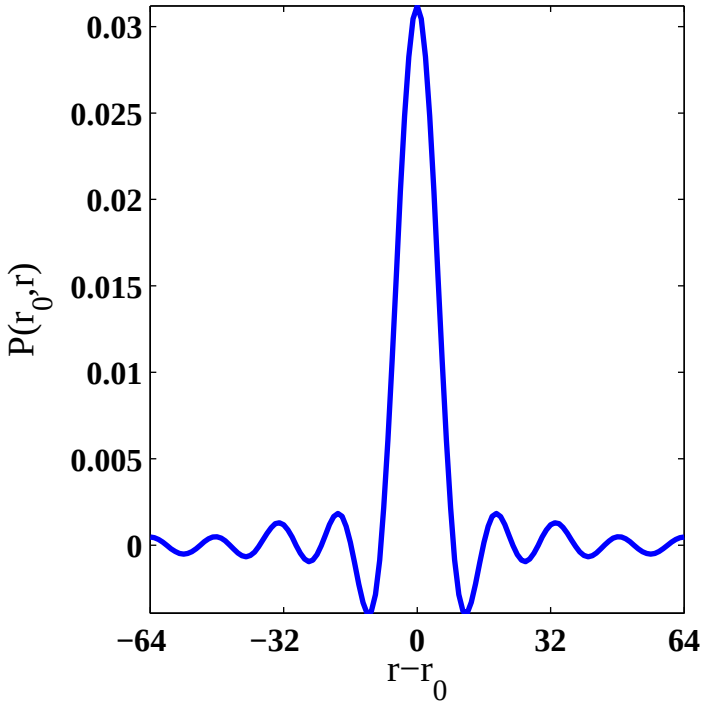

FIG. 4.1. *Elements of a typical density matrix* $P(r_0,r)$. $r_0=(1,64)$, $r=(1,j), j=1,\ldots,128$. $r-r_0$ *is used as x-axis for better visualization. The density matrix decays slowly in the off-diagonal direction and this is also the typical behavior for metallic system.*

$\sqrt{N}$	Algorithm 1 time (sec)	Direct inversion time	L^1 relative accuracy
32	0.03	1.07 secs	4.87×10^{-14}
64	0.16	60.01 secs	1.18×10^{-14}
128	0.86	3672.53 secs	5.16×10^{-14}
256	4.68	2.72 days (est)	
512	28.61	174.11 days (est)	
1024	190.44	30.53 years (est)	

TABLE 4.1. *Comparison of the computational time between Algorithm 1 and the direct inversion algorithm.*

On the right hand side, E is the exact electronic energy and $\widehat{E}$ is the energy computed using Algorithm 2. The other is the L^1 error of the electronic density profile per electron

$$\Delta\rho_{\mathrm{rel}}=\frac{\mathrm{Tr}|\widehat{P}-P|}{N_{\mathrm{Electron}}}. \tag{4.4}$$

Table 4.2 compares the performance using different number of poles and at different cutoff radius R_{Cut}. The target accuracy for the Chebyshev expansion is 10^{-5}. If standard Fermi operator expansion is used for this system (i.e., set $N_{\mathrm{Pole}}=0$), about

N_{Pole}	N_{Cheb}	$R_{\mathrm{Cut}}=4$		$R_{\mathrm{Cut}}=8$	
		$\Delta\rho_{\mathrm{rel}}$	$\Delta\epsilon_{\mathrm{rel}}$	$\Delta\rho_{\mathrm{rel}}$	$\Delta\epsilon_{\mathrm{rel}}$
20	354	7.47×10^{-2}	2.79×10^{-2}	5.74×10^{-3}	8.62×10^{-4}
40	178	3.96×10^{-2}	6.05×10^{-3}	2.71×10^{-4}	1.12×10^{-4}
80	90	5.30×10^{-3}	2.86×10^{-3}	2.35×10^{-5}	5.29×10^{-6}

TABLE 4.2. *Comparison of accuracy for Algorithm 2 with different sets of parameters for a* 32×32 *system. Pole expansion allows the tail part to be represented by a small number of Chebyshev polynomials. The resulting matrix for the tail part is also well localized, even when the whole density matrix is a dense matrix.*

N_{pole}	N_{Cheb}	$R_{\mathrm{Cut}}=4$		$R_{\mathrm{Cut}}=8$	
		$\Delta\rho_{\mathrm{rel}}^{(64)}-\Delta\rho_{\mathrm{rel}}^{(32)}$	$\Delta\epsilon_{\mathrm{rel}}^{(64)}-\Delta\epsilon_{\mathrm{rel}}^{(32)}$	$\Delta\rho_{\mathrm{rel}}^{(64)}-\Delta\rho_{\mathrm{rel}}^{(32)}$	$\Delta\epsilon_{\mathrm{rel}}^{(64)}-\Delta\epsilon_{\mathrm{rel}}^{(32)}$
20	354	-1.42×10^{-5}	5.08×10^{-6}	1.47×10^{-5}	-5.20×10^{-6}
40	178	1.94×10^{-7}	-1.31×10^{-7}	-1.17×10^{-7}	-4.29×10^{-8}
80	90	-2.22×10^{-9}	1.62×10^{-8}	2.43×10^{-10}	-1.28×10^{-10}

TABLE 4.3. *Comparison of accuracy of electron density profile and energy for* 32×32 *and* 64×64 *system. Numerical results show that the accuracy of Algorithm 2 does not deteriorate with increasing system sizes.*

19000 order polynomials are needed for the same accuracy requirement. For the Chebyshev expansion expansion of the tail part $P_{\mathrm{tail}}(H;20)$, the order of polynomial terms drops quickly from 19000 to 354. Starting from 20, the linear relation between N_{Cheb} and N_{Pole} is clear.

Table 4.2 also confirms the trade-off between R_{Cut} and N_{Pole}. In order to improve the accuracy of the entire algorithm, one can either increase R_{Cut} or N_{Pole}. When N_{Pole} is small, a large R_{Cut} is needed for the tail part and therefore leads to larger computational cost (recall the R_{Cut}^4 dependence). We remark also that if $N_{\mathrm{Pole}}=0$, we need R_{Cut} to be so large that effectively the matrices involved are dense matrices, and hence end up with computational cost $\mathcal{O}(N^3)$. The tail part becomes more localized with larger N_{Pole}. As R_{Cut} decays exponentially with respect to N_{Pole}, in practice, N_{Pole} can be taken as a moderate number.

It should also be emphasized that the errors listed in Table 4.2 depend at most weakly on the system size. Since the relative error measured scales the same way as the average error per unit size. Thus the errors do not change with system size either. This is confirmed by the results for $\sqrt{N}=64$. In Table 4.3, we list the difference of $\Delta\epsilon_{\mathrm{rel}}^{(64)}-\Delta\epsilon_{\mathrm{rel}}^{(32)}$ and $\Delta\rho_{\mathrm{rel}}^{(64)}-\Delta\rho_{\mathrm{rel}}^{(32)}$. It can be easily observed that the error depends weakly on the system size. Also note that the dependence becomes even weaker when we involve more poles, since the tail part are more localized.

The result of Table 4.3 confirms the uniform accuracy of Algorithm 2 for different system sizes. Therefore for $\sqrt{N}=1024$, if we choose $N_{\mathrm{pole}}=40$, $N_{\mathrm{Cheb}}=178$, and use a cutoff radius $R_{\mathrm{Cut}}=8$, the relative error of energy and relative L^1 error should be less than 0.05%. For the computational time, the average cost of each pole is 190.44 secs, so 40 poles cost 7618 secs. The tail part costs 4162 secs. The total wall clock time is 11780 secs $\sim$3.27 hours .

5. Conclusion and future work

We have proposed an algorithm for extracting the diagonal of matrices arising from electronic structure calculation. Our algorithm is numerically exact and the overall cost is $O(N^{3/2})$ for a two dimensional problem with N degrees of freedom. We have applied our algorithm to problems in electronic structure calculations, and the numerical results clearly illustrate the efficiency and accuracy of our approach.

We have successfully addressed problems up to one million degrees of freedom. However, many challenging problems arising from real applications involve systems with even larger number of degrees of freedom. Therefore, one natural direction for future work is to develop a parallel version of our algorithm.

We have mentioned a naive extension of our algorithm to three dimensional problems. However, the complexity becomes $\mathcal{O}(N^2)$, which is not yet satisfactory. Therefore, a more challenging task is to see whether the ideas presented here can be used to develop more efficient three dimensional algorithms.

Acknowledgement. This work was partially supported by DOE under Contract No. DE-FG02-03ER25587 and by ONR under Contract No. N00014-01-1-0674 (L. L., J. L. and W. E), by DOE under Contract No. DE-FG02-05ER46201 and NSF-MRSEC Grant DMR-02B706 (L. L. and R. C.), and by an Alfred P. Sloan fellowship and a startup grant from the University of Texas at Austin (L. Y.). We thank Laurent Demanet from Stanford University for providing computing facility and Ming Gu and Gunnar Martinsson for helpful discussions.

REFERENCES

[1] S. Baroni and P. Giannozzi, *Towards very large-scale electronic-structure calculations*, Europhys. Lett., 17, 547–552, 1992.
[2] C. Bekas, E. Kokiopoulou, and Y. Saad, *An estimator for the diagonal of a matrix*, Applied Numerical Mathematics, 57, 1214–1229, 2007.
[3] G. Beylkin, R. Coifman, and V. Rokhlin, *Fast wavelet transforms and numerical algorithms I*, Comm. Pure Appl. Math, 44, 141–183, 1991.
[4] G. Beylkin, N. Coult, and M.J. Mohlenkamp, *Fast spectral projection algorithms for density-matrix computations*, J. Comput. Phys., 152, 32 – 54, 1999.
[5] S. Börm, L. Grasedyck, and W. Hackbusch, *Hierarchical Matrices*, Max-Planck-Institute Lecture Notes, 2006.
[6] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31, 333–390, 1977.
[7] M. Ceriotti, T.D. Kühne, and M. Parrinello, *An efficient and accurate decomposition of the Fermi operator*, J. Chem. Phys, 129, 024707, 2008.
[8] S. Chandrasekaran, M. Gu, X. S. Li, and J. Xia, *Superfast multifrontal method for structured linear systems of equations*, preprint, 2007.
[9] L. Demanet and L. Ying, *Discrete symbol calculus*, sumbitted, 2008.
[10] J.S. Duff and J.K. Reid, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Trans. Math. Software, 9, 302–325, 1983.
[11] J. A. George, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 10, 345–363, 1973.
[12] S. Goedecker, *Linear scaling electronic structure methods*, Rev. Mod. Phys., 71, 1085–1123, 1999.
[13] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73, 325–348, 1987.
[14] W. Hackbusch, *A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices.*, Computing, 62, 89–108, 1999.
[15] N. Hale, N. J. Higham, and L. N. Trefethen, *Computing A^α, $\log(A)$, and related matrix functions by contour integrals*, SIAM J. Numer. Anal., 46, 2505–2523, 2008.
[16] F.R. Krajewski and M. Parrinello, *Stochastic linear scaling for metals and nonmetals*, Phys. Rev. B, 71, 233105, 2005.

[17] S. Li, S. Ahmed, G. Klimeck, and E. Darve, *Computing entries of the inverse of a sparse matrix using the FIND algorithm*, J. Comput. Phys., 227, 9408–9427, 2008.
[18] L. Lin, J. Lu, R. Car, and W. E, *Multipole representation of the Fermi operator with application to the electronic structure analysis of metallic systems*, Phys. Rev. B, 115133, 2009.
[19] L. Lin, J. Lu, L. Ying, and W. E, *Pole-based approximation of the Fermi-Dirac function*, Chinese Ann. Math Ser. B (in press).
[20] J.W.H. Liu, *The multifrontal method for sparse matrix solution: Theory and practice*, SIAM Rev., 34, 82–109, 1992.
[21] G.D. Mahan, *Many-particle Physics*, Plenum Pub Corp, 2000.
[22] P. G. Martinsson, *A fast direct solver for a class of elliptic partial differential equations*, J. Sci. Comp., 38, 316–330, 2009.
[23] T. Ozaki, *Continued fraction representation of the Fermi-Dirac function for large-scale electronic structure calculations*, Phys. Rev. B, 75, 035123, 2007.