



LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

A scalable $O(N)$ algorithm for large-scale parallel First-Principles Molecular Dynamics simulations

D. Osei-Kuffuor, J. L. Fattebert

February 6, 2014

SIAM Journal on Scientific Computing

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

A scalable $O(N)$ algorithm for large-scale parallel First-Principles Molecular Dynamics simulations *

Daniel Osei-Kuffuor[†]

Jean-Luc Fattebert[†]

May 28, 2014

Abstract

Traditional algorithms for First-Principles Molecular Dynamics (FPMD) simulations only gain a modest capability increase from current petascale computers, due to their $O(N^3)$ complexity and their heavy use of global communications. To address this issue, we are developing a truly scalable $O(N)$ complexity FPMD algorithm, based on density functional theory (DFT), which avoids global communications. The computational model uses a general non-orthogonal orbital formulation for the DFT energy functional, which requires knowledge of selected elements of the inverse of the associated overlap matrix. We present a scalable algorithm for approximately computing selected entries of the inverse of the overlap matrix, based on an approximate inverse technique, by inverting local blocks corresponding to principal submatrices of the global overlap matrix. The new FPMD algorithm exploits sparsity and uses nearest neighbor communication to provide a computational scheme capable of extreme scalability. Accuracy is controlled by the mesh spacing of the finite difference discretization, the size of the localization regions in which the electronic orbitals are confined, and a cutoff beyond which the entries of the overlap matrix can be omitted when computing selected entries of its inverse. We demonstrate the algorithm's excellent parallel scaling for up to $O(100K)$ atoms on $O(100K)$ processors, with a wall-clock time of $O(1)$ minute per molecular dynamics time step.

1 Introduction

Classical Molecular Dynamics (MD) simulations rely on parameterized potentials that directly describe interactions between atoms, modeled as classical particles, without calculating the electronic structure. While classical MD can be used to model billions of atoms, it is not applicable in many physical situations where classical potentials fail or are not even available; for instance, in simulations involving various conditions of pressure and temperature, or in situations where the breaking (or making) of molecular bonds occur. Thus, simulating matter at the atomistic level often requires the much more computationally demanding calculation of the electronic structure (*i.e.* quantum electrons) to build realistic models.

First-Principles Molecular Dynamics (FPMD) is a very general and fundamental predictive tool to study matter at the atomistic scale. It includes calculating the electronic structure of the atoms, as well as the actual potential, which describes the “glue” that binds the atoms together. FPMD typically uses the Born-Oppenheimer approximation, which is characterized by classical ions surrounded by quantum electrons. The numerical solution requires solving an eigenvalue problem, the discretized form of the Kohn-Sham (KS) equations of Density Functional Theory (DFT), to model the quantum electronic wave functions (see, for instance, [9]).

Unlike classical problems in physics, such as fluid dynamics or elasticity, which can be modeled by partial differential equations with a number of variables (such as temperature, pressure, etc.) which is fixed and does not grow with the system size, the number of fields in quantum mechanics models (*i.e.* the electronic wave

*This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Work at LLNL was funded by the Laboratory Directed Research and Development Program under project tracking code 12-ERD-048.

[†]Center for Applied Scientific Computing, L-561, Lawrence Livermore National Laboratory, Livermore, CA 94551. email: {oseikuffuor1, fattebert1}@llnl.gov

functions) is proportional to the system size. This leads to $O(N^2)$ degrees of freedom to represent $O(N)$ electronic wave functions for a problem composed of N atoms, and to $O(N^3)$ operations for the numerical solution by standard eigensolvers.

The present practical FPMD computational limit is around 500 atoms, that is a 3D cell of about $8 \times 8 \times 8$ atoms. This is not enough to go much beyond simple atomic geometries involving single atomic species or small molecules. Many problems need to be simplified to be tractable, and the side effects of such simplifications can be quite damaging. With a thousand-fold increase in computer power and the same wall clock time requirements, an $O(N^3)$ algorithm would enable calculations with no more than about 5000 atoms, that is a 3D cell of about $17 \times 17 \times 17$ atoms only. Furthermore, many $O(N^3)$ algorithms are poorly scalable and communication hungry, requiring a high data throughput on parallel computers. Advanced $O(N^3)$ algorithms have been developed to distribute computational work efficiently on large parallel computers using hybrid distributions, where each processor is responsible for a fraction of the coefficients describing a fraction of the electronic wave functions [25]. Pushing such a strategy on today largest computers enable very large calculations (100,000 atoms in Ref. [26]), but with a time to solution far too long to be useful for any real application of interest to domain scientists.

To make an efficient use of current and emerging computing architecture with millions of cores, algorithms with $O(N)$ complexity and short-range communications are needed. This can enable large-scale simulations, where the number of atoms is directly proportional to the number of processors available. A lot of research has been carried out in the last 20 years in the physics and chemistry communities, in an effort to develop $O(N)$ algorithms for electronic structure calculations (see [7] for a recent review). Most $O(N)$ algorithms introduce some approximations or truncations of matrix coefficients to reduce computational complexity. It therefore becomes important to evaluate and control the accuracy of the resulting algorithms [18]. Sufficient accuracy often means that these $O(N)$ algorithms become competitive only at large scale (more than 500 atoms).

An $O(N)$ complexity is, however, not enough if one hopes to make an efficient use of petascale or future exascale computers. Optimal algorithms also need to avoid global communications. One category of algorithms with no major global communications is the so-called “Divide and Conquer” [40, 43]. In these types of algorithm, the global problem is first divided into sub-problems made up of a local subset of atoms and electrons. Then each sub-problem is solved independently, with a buffer region around the sub-domain associated with the sub-problem. The total energy and the total electronic density are obtained as the sums of sub-system contributions. Based on this idea, a few variants have been proposed [33, 40, 44]. However, dividing a problem into sub-problems can be quite tricky, and can lead to hard-to-quantify errors [33], and hard-to-solve (unphysical) sub-problems.

In this paper, we propose a new $O(N)$ algorithm with excellent parallel scaling properties. It relies on two major steps: The first step is the construction of a sparse representation of the invariant subspace corresponding to the N lowest eigenvalues of the Hamiltonian operator. To do that, we rely on techniques developed by the second author, and his coauthors, to find a solution of the Kohn-Sham equations given by a set of strictly localized non-orthogonal functions. We use finite differences to discretize the KS equations and represent these functions on a uniform real-space mesh. In the second step, selected elements of the inverse of the Gram matrix resulting from the dot products of all the pairs of these localized functions need to be computed in $O(N)$ operations. We propose a new technique, which is inherently parallel and highly scalable, to address this second step. The technique is justified by decay properties of the off-diagonal elements of the inverse of the Gram matrix. A brief description of our approach is published in [34], and we present here the details of the algorithm.

The paper is organized as follows: In Section 2 we present the formulation of the electronic structure problem and give a high level description of the numerical solution strategy. We also describe the parallel layout of the problem data. In Section 3, we discuss some algebraic properties of the Gram matrix and its inverse, and present mathematical and algorithmic details for computing selected elements of the inverse of the Gram matrix. We also present an efficient data communication algorithm for gathering and distributing local matrices between neighboring processors. Numerical results are presented in Section 4 that show the scaling, accuracy, and performance of the FPMD algorithm; and we conclude in Section 5 with a summary of our results and a discussion of future work.

2 Electronic structure problem Formulation

2.1 Kohn-Sham energy functional and potentials

Solving the electronic structure problem in Density Functional Theory requires the solution of the Kohn-Sham (KS) equations (see, *e.g.* [9, 41]). From a mathematical point of view, this boils down to determining the invariant subspace corresponding to the N lowest eigenvalues of a non-linear operator. Traditionally, this subspace is represented by a set of N orthogonal eigenfunctions, which correspond to the occupied states. The numerical solution is then typically obtained by solving a KS eigenvalue problem with a non-linear Hamiltonian operator. Here we assume a finite band gap between the eigenvalues associated with this invariant subspace and the rest of the spectrum, thus excluding applications to metallic systems. This is a necessary assumption to achieve $O(N)$ complexity using the algorithm described in this paper.

Following earlier work in $O(N)$ complexity algorithms [21], we adopt an alternative approach in which the problem is formulated as a minimization problem for a set of non-orthogonal electronic orbitals. For a system of N electrons and N_a atoms, the DFT energy functional to be minimized takes the form (in atomic units)

$$\begin{aligned}
 E_{KS}[\{\phi_i\}_{i=1}^N] &= \sum_{i,j=1}^N (S^{-1})_{ij} \int_{\Omega} \phi_i(\mathbf{r}) (-\nabla^2) \phi_j(\mathbf{r}) d\mathbf{r} \\
 &+ \frac{1}{2} \int_{\Omega} \int_{\Omega} \frac{\rho_e(\mathbf{r}_1)\rho_e(\mathbf{r}_2)}{|\mathbf{r}_1 - \mathbf{r}_2|} d\mathbf{r}_1 d\mathbf{r}_2 + E_{XC}[\rho_e] \\
 &+ \sum_{i,j=1}^N (S^{-1})_{ij} \int_{\Omega} \phi_i(\mathbf{r}) \left[\left(\sum_{a=1}^{N_a} V_a \right) \phi_j \right] (\mathbf{r}) d\mathbf{r}.
 \end{aligned} \tag{1}$$

where the ϕ_i s represent the electronic wavefunctions, and ρ_e represents the electronic charge density defined as:

$$\rho_e(\mathbf{r}) = \sum_{i,j=1}^N (S^{-1})_{ij} \phi_i(\mathbf{r}) \phi_j(\mathbf{r}). \tag{2}$$

In the remainder of this paper, the terms *electronic wavefunctions* or *electronic orbitals* or simply *functions* may be used interchangeably to refer to the functions $\{\phi_i\}_{i=1}^N$. In order to simplify the presentation of (1) and (2), we have neglected the spin and assumed that all functions are occupied by two electrons. We also assume here, and in the remainder of this paper, that the functions are linearly independent.

The $N \times N$ matrix S is the Gram or overlap matrix, defined by

$$S_{ij} = \int_{\Omega} \phi_i(\mathbf{r}) \phi_j(\mathbf{r}). \tag{3}$$

As is often the case for these types of problems, we assume periodic boundary conditions. On the right-hand side of (1), the first term represents the kinetic energy of the electrons; the second term is the classical electrostatic interactions between electrons; the third term is the exchange and correlation energy, a purely quantum interaction between electrons; and the last term represents the energy of the electrons in the potential field generated by the atomic cores. Note that by convention we use a positive function to describe the charge density of the electrons, that is $\rho(\mathbf{r}) \geq 0$, and a negative function for the atomic potentials, $V_a(\mathbf{r}) \leq 0$ (potential *well*).

Each atomic core is modeled by a non-local separable pseudopotential in its Kleinman-Bylander form [27]. With this approximation, the potential generated by each atomic core is given by the sum of a diagonal long-range operator (a multiplicative function with slow decay away from the atom center) and a low-rank short-range operator (usually a sum of up to nine projectors onto local functions centered on the atom).

The exchange and correlation term $E_{XC}[\rho_e]$ has no known exact form, and hence is typically approximated. Here, we assume a standard approximation as an integral of a local term

$$E_{XC}[\rho_e] = \int_{\Omega} \epsilon_{xc}(\rho_e(\mathbf{r}), \nabla \rho_e(\mathbf{r})) d\mathbf{r}, \tag{4}$$

where ϵ_{xc} is the exchange-correlation density. This includes the Localized Density Approximation (where ϵ_{xc} function of $\rho(\mathbf{r})$ only) and the Generalized Gradient Approximation, two of the most commonly used functionals in the field [41].

Remark 1. In the most common approach, the DFT energy functional minimization problem is formulated in terms of orthonormal eigenfunctions. That is, the minimization of (1) is subject to the constraint

$$\int_{\Omega} \phi_i(\mathbf{r})\phi_j(\mathbf{r}) = \delta_{ij}, \quad (5)$$

and (S^{-1}) is simply replaced by the identity matrix. The mathematically equivalent alternative chosen here alleviates the need for the orthonormality constraints and will allow for localization constraints to be imposed on each function (see Section 2.3). We will only assume that the functions $\{\phi_i\}_{i=1}^N$ are linearly independent.

2.2 Discretization and numerical formulation

We discretize the energy functional in (1) by finite differences on a uniform mesh. This is possible given the shape of the pseudopotentials used to describe the atomic cores. We use a fourth order finite difference scheme for the Laplacian and represent all the wave functions and potentials by their discrete values at the nodes of the mesh.

The electrostatic term in (1) is computed by solving a Poisson problem. Because of the periodic boundary conditions, it is necessary that the atomistic system be charge neutral. That is, we assume the charge of the electrons exactly neutralize the charge of the atomic cores. To have a well posed Poisson problem with a right hand-side corresponding to a neutral charge, we follow a standard procedure [17, 22]. We first associate to each atomic core, a spherically symmetric Gaussian charge distribution

$$\rho_a(\mathbf{r}) = -\frac{Z_a}{(\sqrt{\pi}r_c^a)^3} \exp\left(-\frac{|\mathbf{r} - \mathbf{R}_a|^2}{(r_c^a)^2}\right), a = 1, \dots, N_a \quad (6)$$

where Z_a is the charge of the atomic core, \mathbf{R}_a its position, and r_c^a is some parameter of order 1.

Let us define

$$\rho_s(\mathbf{r}) = \sum_{a=1}^{N_a} \rho_a(\mathbf{r}).$$

To compute the potential, v_C , due to the neutral charge distribution $\rho_e + \rho_s$, we solve the Poisson problem

$$-\nabla^2 v_C(\mathbf{r}) = 4\pi(\rho_e + \rho_s)(\mathbf{r}). \quad (7)$$

This Poisson problem is efficiently solved on the discretization grid by a multigrid preconditioned conjugate gradient (CG) algorithm [42].

The potential v_C , in addition to the Coulomb potential due to the electronic charge density ρ_e , also includes the potential, v_a , due to each charge distribution ρ_a . Each potential v_a is given by the analytical form

$$v_a(\mathbf{r}) = -\frac{Z_a}{|\mathbf{r} - \mathbf{R}_a|} \operatorname{erf}\left(\frac{|\mathbf{r} - \mathbf{R}_a|}{r_c^a}\right). \quad (8)$$

To compensate for that extra contribution in v_C , we subtract the potentials v_a from the pseudopotentials V_a . As a result, the energy functional (1) can be rewritten in the totally equivalent form (see [17] for example)

$$\begin{aligned} E_{KS}[\{\phi_i\}_{i=1}^N] &= \sum_{i,j=1}^N (S^{-1})_{ij} \int_{\Omega} \phi_i(\mathbf{r}) (-\nabla^2) \phi_j(\mathbf{r}) d\mathbf{r} \\ &+ \frac{1}{2} \int_{\Omega} v_C(\mathbf{r})(\rho_e(\mathbf{r}) + \rho_s(\mathbf{r})) d\mathbf{r} + E_{XC}[\rho_e] \\ &+ \sum_{i,j=1}^N (S^{-1})_{ij} \int_{\Omega} \phi_i(\mathbf{r}) \left(\sum_{a=1}^{N_a} (V_a - v_a) \phi_j \right)(\mathbf{r}) d\mathbf{r} + E_{\text{diff}} + C. \end{aligned} \quad (9)$$

Here

$$E_{\text{diff}} = \sum_{a,b=1,a<b}^{N_a} \frac{Z_a Z_b}{|\mathbf{R}_a - \mathbf{R}_b|} \text{erfc} \left(\frac{|\mathbf{R}_a - \mathbf{R}_b|}{\sqrt{(r_c^a)^2 + (r_c^b)^2}} \right),$$

for each pair of atoms, a and b , at positions R_a and R_b with core charges Z_a and Z_b respectively; and C is a constant dependent only on the number of atoms and their species.

There are a few important consequences of this formulation. First of all, E_{diff} can be computed as the sum of short-range terms in $O(N)$ operations, by dropping terms corresponding to pairs of atoms that are far apart. Secondly, for each atom, $V_a - v_a$ is a short-range potential. As a result, the term involving the atomic potential in (9) can be computed by restricting the sum over all the atoms, to just a sum over a few local atoms. Thus the only remaining global coupling is through the electronic structure represented by the set of functions $\{\phi_i\}_{i=1}^N$. The remainder of this paper is focused on showing how the calculation of these $\{\phi_i\}_{i=1}^N$ can be done in $O(N)$ operations with short-range (or nearest neighbor) communications only.

2.3 Electronic structure solver

The solution to the minimization problem (1) may be expressed as a matrix Φ , whose columns consist of the electronic wavefunctions $\phi_i, i = 1, \dots, N$. In principle, each of these ϕ_i s is dense, with non-zero entries at each mesh point in the global domain. In this form, the solution requires $O(N^2)$ storage and the numerical solvers require $O(N^3)$ operations.

However, since (1) does not assume any particular representation of the solution to the invariant subspace, one can possibly consider a sparse representation. For instance, the so-called Maximally Localized Wannier functions [32], which minimize simultaneously Eq.(1) and a functional representing the sum of the spreads of each function, are very localized for systems with a finite band gap. However, computing these functions requires $O(N^3)$ operations if no strict localization is assumed a priori.

An alternative, which we follow in this paper, is to search for an approximation to Φ by assuming its sparsity pattern is known a priori. One major advantage of formulating the DFT energy functional in the general form (1) without orthogonality constraints is the possibility of adding constraints on the functions $\{\phi_i\}_{i=1}^N$. It has been previously shown [16,18,19] that constraining each function to be non-zero only inside a finite spherical domain of radius R_c leads to $O(N)$ degrees of freedom, and $O(N)$ operations for the most expensive parts of the computation. This restriction on the solution of the minimization problem introduces a truncation error for physical quantities of interest, which decays exponentially with the size of the confinement regions, R_c . We note that R_c is fixed and does not grow with the problem size, and we shall refer to it as the localization radius.

The minimization procedure is based on the gradient of the energy functional, given by the residual of the KS equations. The details of that procedure and the minimization algorithms are discussed in recent papers [16,18,19] and only the general idea is described here. In our formulation, the gradient is given by

$$G^{(\Phi)} = H\Phi - \Phi S^{-1}(\Phi^T H\Phi). \quad (10)$$

We use (10) in a preconditioned steepest descent minimization algorithm, utilizing a multigrid preconditioner for fast convergence. The procedure is accelerated using an Anderson extrapolation scheme [1,15]. In order to maintain sparsity in Φ , we truncate the correction applied to Φ at each step of the iterative algorithm.

Using this approach, each wavefunction has a constant number of nonzero entries, which does not grow with the size of the problem. As a result, the total number of nonzeros in Φ is $O(N)$. Furthermore, all operations to evaluate the energy functional (1) and its gradient (10) become $O(N)$, except the computation of S^{-1} , which is traditionally an $O(N^3)$ operation. Later in this paper, we shall address ways to efficiently compute selected entries of S^{-1} that are needed to construct and solve the minimization problem.

2.4 Parallel data representation

The parallel algorithm utilizes a domain decomposition framework, with each processing unit owning a subdomain of the global finite difference mesh. As a result, each localized orbital is distributed across a finite number of neighboring processors. We assume that these localized orbitals are fairly evenly distributed over the 3D domain. Figure 1 shows an illustration of the data layout. Each processing unit owns the part

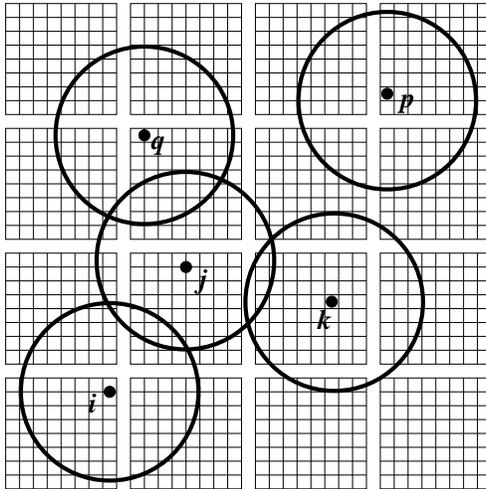


Figure 1: Parallel data representation, showing functions ϕ_i, ϕ_j, \dots , overlapping in different subdomains, as well as their centers.

of the electronic orbital that overlaps with its subdomain, and handles any local computations involving this part of the function. To compute the global Gram matrix, for instance, each processor computes a partial contribution to the global matrix corresponding to functions that overlap within its local subdomain. Thus, the entries in some row k of the Gram matrix are obtained by accumulating data from neighboring subdomains that compute partial contributions involving function k .

This data representation leads to a very convenient and efficient way of handling computations involving the electronic wavefunctions, such as computing entries of the Gram matrix. However, computing entries of the inverse of the Gram matrix in such an efficient and straightforward way, is far more challenging. While calculating S^{-1} typically requires $O(N^3)$ operations, it takes little time compared to other operations in a standard $O(N^3)$ FPMD method, and its cubic scaling can even be ignored in $O(N)$ FPMD methods for values of N up to a few thousands [16, 19]. Nonetheless, for large-scale parallel simulations, the exact calculation of S^{-1} can become a computational bottleneck. This is not only because of its growing cost, but mostly because of the large number of communications necessary to build the elements of S , distribute them according to a data layout appropriate for the linear solver, and then distribute the elements of S^{-1} to the processors which need them. Later on in this paper, we shall describe a scalable strategy for computing the entries of S^{-1} , in order to make the overall $O(N)$ FPMD algorithm truly scalable.

3 Properties of the Gram matrix and its inverse

From Equations (1) and (2), it is clear that we need not compute all the entries of the inverse of S . In particular, one only needs to compute $(S^{-1})_{ij}$ for i and j corresponding to the nonzero dot products between ϕ_i and ϕ_j , ϕ_i and $-\nabla^2\phi_j$, or the term involving the atomic potential in (9). In recent years, several fast algorithms have been proposed for computing exact selected entries of the inverse of sparse symmetric matrices [8, 30, 31]. These algorithms rely on efficient Cholesky factorizations and divide-and-conquer strategies, and their theoretical analysis have shown them to be more efficient than the traditional $O(N^3)$ direct inversion of the full matrix. The numerical results from the literature also show that they exhibit superior performance over the $O(N^3)$ method, on applications from electronic structure calculations. However, for large-scale computations, the nonlinear complexity of these exact inversion techniques will affect the overall complexity.

In [14], Demko *et al.* present some theoretical results on the exponential decay rates for the off diagonal elements of the inverses of banded matrices and general sparse matrices. They show that for an $N \times N$ positive definite matrix A that is bounded and bounded invertible, the size of the off-diagonal entries of A^{-1} outside some nonzero pattern defined by the pattern of A^α , $\alpha = 1, 2, \dots, N$, is bounded by an

exponentially decaying term with an exponent of the order $\alpha + 1$, and a decay factor that depends on the bounds of the eigenvalue spectrum of A . The authors further show this result to be true for a 1D periodic system. Later on, Benzi and Razouk [4] generalized this for $N \times N$ sparse matrices by using principles from graph theory, and by considering the distances between the nodes of the corresponding adjacency graph. Furthermore, they show that an $O(N)$ approximation for the inverse is feasible if, for a sequence $\{A_N\}$ of $N \times N$ matrices of increasing dimension N , the following are satisfied:

1. for all N , there exists a compact set bounded away from zero, that contains all the eigenvalue spectra of the matrices in $\{A_N\}$, and
2. the condition numbers of the corresponding eigenvector matrices are bounded independent of N .

Notice that if each matrix of the sequence is symmetric, then the second condition is trivially satisfied.

In order to make use of these results for an $O(N)$ approximation of S^{-1} , we show that the overlap matrix S defined in (3) satisfies the above conditions. To see this, we note that S , by construction, is symmetric and positive definite. It is however, not diagonally dominant. In what follows, let \mathcal{P} denote a problem of interest, that is a set of atoms defined by their pseudo-potentials and a number of electronic wave functions defined on the domain Ω . Consider a decomposition of \mathcal{P} into several subproblems associated with non-overlapping subdomains similar to the domain decomposition framework described in Section 2.4. We shall refer to each of these subproblems as a base problem. Now, suppose that each base problem has at least one electronic orbital overlapping with it, and let \mathcal{P}^e denote the e -th base problem. To each base problem \mathcal{P}^e one can associate a local overlap matrix, denoted by S^e , made up of the local contributions to the global Gram matrix S . That is, the parts of the dot product computed by integrating over the subdomain associated with \mathcal{P}^e . Let us denote by p_{max} the maximum number of subdomains that share (or overlap with) any given function. Using an analogy between a base problem and a finite element, we now state a proposition that is just a corollary of a result of Fried [20], but is of fundamental importance to the theory presented in this paper.

Proposition 1. *Let S be the Gram matrix corresponding to a problem \mathcal{P} , and let S^e be the Gram matrix for the e -th base problem obtained by a decomposition of \mathcal{P} . Define λ_{min} and λ_{max} to be the minimum and maximum eigenvalues of S , and define also λ_{min}^e and λ_{max}^e as the minimum and maximum eigenvalues of S^e . Then*

$$\lambda_{max} \leq p_{max} \max_e (\lambda_{max}^e) \quad (11)$$

and

$$\min_e (\lambda_{min}^e) \leq \lambda_{min} \quad (12)$$

From (11) and (12), we have the following bounds for the condition number $\kappa(S)$ of S :

$$1 \leq \kappa(S) \leq p_{max} \frac{\max_e (\lambda_{max}^e)}{\min_e (\lambda_{min}^e)}. \quad (13)$$

Notice that the value of p_{max} depends on the size of the base problem domain, relative to the size of the localization radius R_c . It is easy to see that defining each subdomain to have a side of length of at least $2R_c$ yields $p_{max} = 8$ (in 3D). From a practical point of view, (13) suggests that one can impose conditions on the wavefunctions so that S is well conditioned. Thus, we can assume there exist two positive constants γ and β , which bound the spectra of the base problems, such that

$$\gamma \leq \min_e (\lambda_{min}^e) \quad (14)$$

and

$$\max_e (\lambda_{max}^e) \leq \beta. \quad (15)$$

In practice, condition (14) is checked and satisfied at the beginning of each MD step to ensure that the localized wavefunctions are linearly independent.

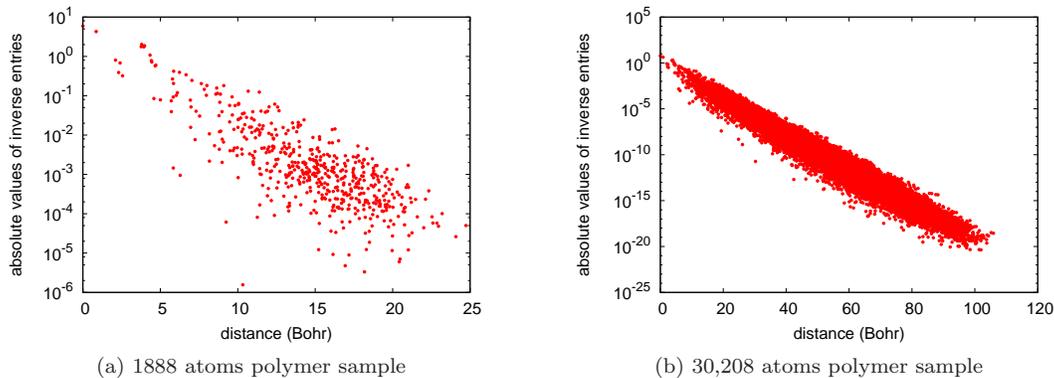


Figure 2: Inverse entries in one column of S^{-1} as a function of the distance between this column and the orbitals associated with its nonzero entries. Left: 1888 atoms polymer sample; and Right: 30,208 atoms polymer sample.

Let $\{S_N\}$ denote a sequence of overlap matrices. From Proposition 1 it follows that the eigenvalue spectrum of each matrix $S \in \{S_N\}$ is closed and bounded. Let $\mathcal{C} \subset \mathbb{R}$ denote the union of these eigenvalue spectra. Clearly \mathcal{C} is also closed and bounded. Furthermore, Proposition 1 and condition (14) imply

$$\gamma \leq \lambda_{min}$$

for all the matrices in $\{S_N\}$. Thus, the eigenvalue spectra of $\{S_N\}$ is bounded, and bounded away from zero by some positive distance.

These properties of S imply that S^{-1} has off-diagonal coefficients that decay exponentially, and thus, an $O(N)$ approximation to S^{-1} is feasible [4, 14]. Consider the adjacency graph of S , where there is an edge between nodes i and j if the wavefunctions i and j overlap. Then the decay property of S^{-1} implies that the longer the path from one orbital function to another, in the adjacency graph of S , the smaller the coefficient in S^{-1} . Figure 2 gives an illustration of the decay of the off-diagonal elements of S^{-1} obtained from a real application. The figure shows the absolute value of the entries in one column of S^{-1} , as a function of the geometric distance between the center of the corresponding localized wavefunction and the centers of all the other wavefunctions in the system. Here, S is obtained from a periodic system of polymers initially containing 472 atoms and 536 electronic orbital functions. Thus, S has size 536, with about $O(300)$ nonzero entries per row. The Figure 2a shows the decay for the initial Gram matrix and Figure 2b illustrates the same problem replicated by a factor of 4 in the X , Y , and Z - directions.

For sufficiently large problems, this decay property suggests that one can obtain a sparse representation of the inverse, by dropping small terms. Furthermore, as we will show later, an entry $(S^{-1})_{ij}$ need not be computed exactly, and may be appropriately approximated without any significant effect on the accuracy of the numerical solution. As shown in Figure 2a, the decay of the off-diagonal entries of S^{-1} leads to very little sparsity for small problems, primarily due to the shorter paths or distances between the nodes in the adjacency graph of S . Thus, approximating S^{-1} , or exploiting sparsity on S^{-1} by dropping small terms, is possible only for fairly large problems ($N \geq 1000$).

In [7], Bowler *et al.* review a number of methods for computing an approximation to the inverse of the Gram matrix for $O(N)$ electronic structure calculations. For these strategies to be effective, it is necessary that the error incurred in approximating the inverse be smaller than that of the overall algorithm. Many of these methods impose sparsity constraints on the matrix in order to achieve linear complexity. While these methods may be favorable for $O(N)$ algorithms, their parallel implementations generally require some global communication, which can affect parallel efficiency. Next, we present an efficient parallel strategy for approximately computing the entries of S^{-1} that are needed to evaluate (1) and (2).

3.1 Computing selected entries of S^{-1}

To compute entries in S^{-1} , we follow the approximate inverse strategy, commonly used to construct a preconditioner for the iterative solution of sparse linear systems [5, 6, 10, 28]. The basic idea is to find some $N \times N$ matrix M , such that $M \approx S^{-1}$. One way of achieving this is to decompose S into its Cholesky factors, $S = LL^T$. A matrix G , is then constructed such that $G \approx L^{-1}$, and thus $M = G^T G$. The approximation, G , may be computed in many different ways, and while these so-called factorized approximate inverse techniques are well-known in the literature (see for instance [5, 6, 28]), they are difficult to parallelize [7]. As a result, they are not very feasible for $O(N)$ algorithms at large-scale.

Here we adopt another alternative first introduced in [2, 3]. We consider an approximate inverse M that satisfies

$$\arg \min_{M \in \mathbb{R}^{N \times N}} \|SM - I\|_F^2 \quad (16)$$

in the Frobenius norm, where I is the identity matrix, and subject to the constraint that M is sparse. By definition of the Frobenius norm, this minimization problem naturally decouples into N smaller sub-problems

$$\arg \min_{m_j \in \mathbb{R}^N} \|Sm_j - e_j\|_2^2, \quad (17)$$

which can be handled efficiently in parallel. Here, m_j and e_j are the respective columns of M and I . Let S_j be a sparse $N \times k$ matrix, $k < N$, and whose columns correspond to k distinct columns of S , associated with some prescribed non-zero pattern for m_j . Define this set of k distinct columns as \mathcal{J} , and let \mathcal{I} represent the set of non-zero rows of S_j . Then the solution to (17) is typically obtained by constructing from S_j , an $r \times k$ matrix \bar{S}_j , where $k < r$ and $r < N$, and solving the least-squares problem

$$\bar{S}_j \hat{m}_j = \bar{e}_j. \quad (18)$$

Here, $\bar{S}_j = S(\mathcal{I}, \mathcal{J})$ is a restriction onto rows \mathcal{I} and columns \mathcal{J} of S , $\hat{m}_j = m_j(\mathcal{J})$ represents a restriction onto rows \mathcal{J} of m_j , and $\bar{e}_j = e_j(\mathcal{I})$ represents a restriction onto rows \mathcal{I} of e_j .

The solution to (18) may be obtained by using the QR decomposition of \bar{S}_j . Alternatively, the normal equations approach may be used [37, 38]. Notice that here, we have assumed that the non-zero pattern of M is known a priori. Variants of the non-factorized approximate inverse technique that adaptively determine the non-zero pattern of M , to satisfy a prescribed error tolerance, have also been proposed in the literature [12, 13, 23, 24]. More details on the approximate inverse technique (both factorized and non-factorized) may be found in [5–7, 10–13, 23, 24, 28] and references therein. Depending on the the accuracy required for the approximate inverse and the corresponding sizes of r and k , the QR decomposition of \bar{S}_j or the normal equation solution to (18) can be expensive to compute. A more efficient alternative can be obtained by minimizing the norm of the residual in (17) only for the components in the non-zero pattern of m_j . In other words, instead of (17), we find M whose columns satisfy:

$$\arg \min_{m_j \in \mathbb{R}^N} \|P^T(Sm_j - e_j)\|_2^2, \quad (19)$$

where $P \in \mathbb{R}^{N \times k}$ has orthonormal columns that are simply obtained from columns of the identity matrix corresponding to the non-zero pattern of m_j . That is, $P = I(:, \mathcal{J})$. Defining P in this way, we have that $S_j = SP$ and $\hat{m}_j = P^T m_j$. Defining also $\hat{S}_j = P^T S P$ and $\hat{e}_j = P^T e_j$, (19) may be rewritten as minimizing

$$\begin{aligned} \|P^T(SPP^T m_j - e_j)\|_2^2 &= \|P^T(S_j \hat{m}_j - e_j)\|_2^2 \\ &= \|\hat{S}_j \hat{m}_j - \hat{e}_j\|_2^2. \end{aligned} \quad (20)$$

The solution to (20) is obtained by solving the linear system

$$\hat{S}_j \hat{m}_j = \hat{e}_j, \quad (21)$$

The matrix \hat{S}_j is a square matrix that represents a principal submatrix of S , whose columns are prescribed by the non-zero pattern of m_j . As a result, the linear system can be solved using an iterative solver or a Cholesky decomposition of \hat{S}_j , and thus, avoids the need to compute a QR decomposition of \hat{S}_j or a normal

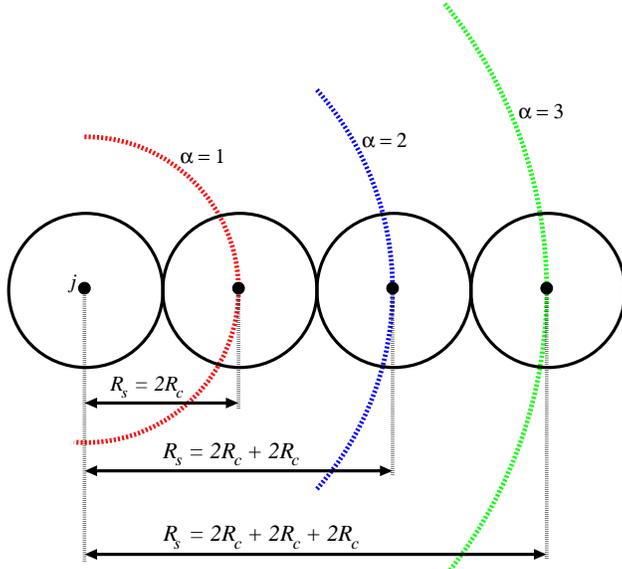


Figure 3: Plot of region covered by $R_s = 2\alpha R_c$ from orbital j . Red boundary: $\alpha = 1$; Blue boundary: $\alpha = 2$; Green boundary: $\alpha = 3$.

equation solution. This approach is equivalent to re-weighting the equations (or rows of S) in (17) in favor of those that are closest to the coefficients of m_j that are of interest, and leads to more accurate results compared to solving the original least-squares problem.

Defining \hat{S}_j as a principal submatrix of S leads to some desirable spectral properties for \hat{S}_j . First, \hat{S}_j is symmetric and positive definite by definition, since S is symmetric and positive definite. Furthermore, from the interlacing property of the eigenvalues of a symmetric matrix [36], we have that the eigenvalues of \hat{S}_j are contained within the spectrum of S . As a result, the following relation holds for the condition numbers of \hat{S}_j and S :

$$\kappa(\hat{S}_j) \leq \kappa(S).$$

3.2 Imposing a sparsity pattern on M

Prescribing an appropriate sparsity pattern that gives a good approximation to the inverse can be challenging, and adaptive methods have been shown to be more effective for general sparse matrices. Nonetheless, for matrices whose inverse exhibit strong off-diagonal decay, a suitable sparsity pattern can be prescribed a priori, which is generally more efficient than an adaptive approach. A common strategy is to use the sparsity pattern of powers of S , S^α , where $\alpha > 0$ is an integer [10, 11]. From the adjacency graph of S , this approach is equivalent to constructing \hat{S}_j from columns of S associated with nodes that are up to paths of length α from node j . A variant of this approach considers only the paths associated with neighboring nodes that are *strongly connected* to node j . This is algebraically achieved by dropping small entries in S before extracting the pattern of S^α [11, 29, 35]. In general, the bigger α is, the denser the j -th column of M . This results in a larger size for the set of columns \mathcal{J} , which typically yields a more accurate approximate inverse.

In this work, we exploit geometric information and use the geometric distances between the centers of the electronic orbitals to determine the sparsity pattern of M . For each column j of M , the non-zero pattern is obtained by considering only entries associated with localized orbitals centered within some distance R_s from the localized orbital j . In practice, R_s is typically chosen to be a multiple of the localization radius, R_c , used to define the confinement regions for the localized orbitals. Let c_i and c_j be the centers of the functions i and j respectively. Then two orbitals, i and j , will overlap and share an edge if $|c_i - c_j| \leq 2R_c$. In other words, S_{ij} will be a non-zero entry if the nodes i and j are separated by a path of length 1 in the adjacency graph of S . Figure 3 gives an illustration of a region prescribed by $R_s = 2\alpha R_c$ around a function j , for different values of α . From this definition of R_s , it follows that if some function k lies on the boundary

of the region defined by R_s , then the shortest path between nodes j and k in the adjacency graph of S has length

$$d(j, k) = \frac{R_s}{2R_c}.$$

Note that the distance $d(j, k)$ is based on graph theory, and does not translate to the geometric (or Euclidean) distance between the functions j and k . In general, it is possible to have another function i in the region, which may be closer to function j in terms of the Euclidean distance, but have $d(j, i) > d(j, k)$. Since function k is centered on the boundary of R_s , clearly the region prescribed by R_s also contains all functions l , such that $d(j, l) \leq d(j, k)$. Following this result, Proposition 2 establishes a relationship between the sparsity pattern prescribed by R_s , and that of S^α .

Proposition 2. *Let $\alpha > 0$ be some integer, and let the set \mathcal{F}_j contain all the orbitals corresponding to non-zero positions in column j of M , prescribed by $R_s = 2\alpha R_c$. Let the set \mathcal{Q}_j contain all the orbitals corresponding to non-zero positions in column j of M , prescribed by the non-zero pattern of S^α . Then $\mathcal{Q}_j \subseteq \mathcal{F}_j$.*

Proof. Clearly, for any integral multiple, α , the region prescribed by $R_s = 2\alpha R_c$ from some function j , contains all nodes k such that $d(j, k) \leq \alpha$. The result follows from the fact that for any integer power p of some matrix A , the non-zero pattern of some column j of A^p corresponds to nodes that are separated from node j by paths of length p or less, in the adjacency graph of A . \square

3.3 Parallel communication algorithm

In this section, we describe the algorithm we use to assemble matrix elements from the partial computations performed on each subdomain. This algorithm is used to build the matrix \hat{S} in Section 3.1 and to scatter the results to neighboring processors that need the computed selected elements of S^{-1} . It is also used in other parts of the FPMD algorithm to assemble other matrix elements needed to evaluate the energy functional (1) and its gradient (10).

Parallel scaling can benefit from a data communication strategy that takes advantage of the physical data representation of the underlying problem. Since the interactions between a pair of electronic orbitals cover only a small subset of subdomains, it makes sense to exploit a nearest-neighbor communication pattern to gather and distribute data between neighboring subdomains. In what follows, we define R_s as the spread radius used to prescribe the extents from which to gather and scatter data within a subset of subdomains. In other words, each subdomain or processor communicates within a 3D cube of neighboring processors, determined by R_s . We note that the parallel constructs used in this work are based on the Message Passing Interface (MPI) programming model.

One straightforward approach for communicating data among neighboring processors is to post multiple send and receive operations between each processor and its cube of neighbors. This approach takes advantage of the multiple communication links in the hardware of the parallel system, and can be ideal for architectures with smaller bandwidths but large concurrency (or link network), such as the IBM Blue Gene systems. However, depending on the size of the cube of neighbors and the size of the data to be communicated, the cost of communication by this approach can be expensive. In this section, we propose an alternative algorithm that processes data one direction at a time and allows for overlap between communication and data assembly. Algorithm 1 formally describes this approach. For each processor, let n_{steps} be the number of neighboring processors located within the distance R_s . Furthermore, notice that in each direction, each processor has two adjacent neighbors - one on the left, and one on the right. Note that a processor at the boundary will have a periodic neighbor as an adjacent neighbor. Let $-nb$ and nb denote the left and right neighbors, respectively.

Algorithm 1. *Parallel communication algorithm*

1. For each direction X, Y , and Z , do:
2. Copy local data to work buffer
3. For $dir = \{-nb, nb\}$, do:
4. Copy data from work buffer to send buffer
5. For $l = 1$ to n_{steps} , do:

6. *Send data to the dir neighbor, and receive data from the -dir neighbor*
7. *Merge received data with local data*
8. *if $l < n_{steps}$ Copy received buffer to send buffer*
9. *EndDo*
10. *EndDo*
11. *EndDo*

Algorithm 1 begins by looping over the X, Y and Z directions, respectively. For each direction, the algorithm sends data n_{steps} times to one adjacent neighbor, and receives data n_{steps} times from the other adjacent neighbor. Note that here, we have assumed that n_{steps} is the same in each direction, however in practice, depending on the processor grid topology, n_{steps} may be different for each direction. In order to avoid sending duplicate data to the neighboring processors, the initial local data for each subdomain is first copied into a work buffer, prior to the beginning of the data transfer, for a particular direction. At each step of the innermost loop of the algorithm, data is first sent to the left adjacent neighbor, and data is received from the right adjacent neighbor. The received data is then merged with the current local data, and the receive buffer is copied into the send buffer. These steps are repeated n_{steps} times, corresponding to the number of processor steps prescribed by the radius R_s . Note that since communication is only between the two adjacent neighbors, after the first n_{steps} iterations, the initial local data on each subdomain would be propagated up to the neighboring subdomain n_{steps} to the left. Likewise, each subdomain would have received the local data of its n_{steps} neighbors to the right. This loop is then repeated, this time sending data to the right, and receiving data from the left. In practice, step 8 for the current iteration and step 6 for the next iteration can be performed while still working on step 7 for the current iteration in the inner loop of the algorithm, thus overlapping communication with computation.

By the end of the pass in the X direction, each processor has accumulated data from its nearest n_{steps} neighbors to the left and right, respectively. The accumulated local data is then used to initialize the work buffer for the subsequent pass in the Y direction, and similarly in the Z direction.

At the end of the algorithm, each processor or subdomain, k , has accumulated data from the neighboring processors associated with subdomains within a cube of $(2 \times n_{steps} + 1) \times (2 \times n_{steps} + 1) \times (2 \times n_{steps} + 1)$ subdomains, where the subdomain k lies in the center of this cube. The total number of sends and receives for the algorithm is $3 \times (2 \times n_{steps}) = 6 \times n_{steps}$, which is significantly fewer than the $(2 \times n_{steps} + 1)^3 - 1$ communication operations that would be required by a more straightforward approach.

3.4 Parallel algorithm for computing S^{-1}

In what follows, let \mathcal{F}_j be the set of all electronic orbitals that are centered within some distance R_s from orbital j . We recall from the parallel data representation model that each processor holds some partial contribution to the global Gram matrix, S . Furthermore, in computing the inverse, each processor is responsible for solving (20) for the columns of M , corresponding to the orbitals centered in its local subdomain. In practice, the localized orbitals centered on each local subdomain are close in proximity to each other. Thus, it is practical to gather enough data to assemble a single matrix to be used in (21), for all local columns. That is, let \mathcal{C} denote the indices of the localized orbitals that are centered in the local subdomain, and define

$$\mathcal{F} = \bigcup_{j \in \mathcal{C}} \mathcal{F}_j.$$

Then one can construct a square matrix \hat{S} , by considering the interactions between electronic orbitals in \mathcal{F} . As a result, \hat{S} represents a principal submatrix of S , and may be used in place of \hat{S}_j . The solution to (21) then becomes simply solving a linear system with multiple right hand sides. Algorithm 2 formally describes the parallel computation of the entries in S^{-1} .

Algorithm 2. *Parallel algorithm for computing selected elements of S^{-1}*

1. *Define R_s*
2. *Gather data from neighboring processors according to R_s , and assemble \hat{S}*
3. *for $j \in \mathcal{C}$ do:*

4. solve $\hat{S}\hat{m}_j = \hat{e}_j$
5. EndDo
6. Distribute inverse data m_j to neighboring processors within some predefined radius

Algorithm 2 begins by first defining the radius R_s , which is used to determine the extents from which to gather data to construct \hat{S} in Line 2 of the algorithm. Since each processor owns some partial contributions to the columns of the overlap matrix, gathering data from neighboring processors and assembling \hat{S} can occur simultaneously. This is handled efficiently by the data communication algorithm (Algorithm 1) in Section 3.3. Once \hat{S} is constructed, (21) is solved for each column of M , corresponding to orbitals that are centered on the local subdomain (Lines 3 – 5). In order to locally compute the electronic density in (2) and evaluate (1), each processor requires certain entries of S^{-1} that may have been computed on a neighboring processor. Thus, in the last line of the algorithm, the computed columns of $M \approx S^{-1}$ are distributed to neighboring processors that need them for their local computations.

3.5 Some practical considerations

While in theory the principal submatrix \hat{S} possess desirable spectral properties with respect to the overlap matrix S , in practice, obtaining a well-conditioned submatrix by accumulating data from neighboring processors requires some care.

Recall that in assembling \hat{S} on each subdomain, the accumulated data consists of the sum of partial contributions of entries of the Gram matrix computed over the closest neighboring subdomains. As a result, \hat{S} may include columns or rows with incomplete entries and small diagonals. These incomplete columns or rows come from subdomains that lie on the boundary of the region prescribed by R_s ; since they may contain partial contributions from functions that overlap predominantly outside this region of interest. The result is that \hat{S} may be poorly conditioned, even though the global Gram matrix may be well conditioned.

One way to improve the conditioning of \hat{S} is by rescaling the small diagonals. An alternative approach, which we follow, is that during the accumulation of data from the farthest subdomains that lie on the boundary of the region prescribed by R_s (*i.e.* the last iteration of the inner loop of Algorithm 1), the data from a new column j (non-existent in closer subdomains) is merged with the current local data *only* if $|S_{jj}| \geq \tau$, for some threshold τ . This strategy essentially imposes some form of boundary condition on the data communication algorithm to yield a better conditioned matrix \hat{S} .

A typical MD code structure consists of an outer loop over the MD iterations during which atoms may advance from one position to another. Each MD iteration consists of a series of inner iterations to solve the electronic structure of the molecular system, for the current atomic positions. Within each of these inner iterations, the current overlap matrix is assembled, and its inverse is computed by solving (21), in order to compute the electronic density and update the solution to the energy functional. We solve (21) using a preconditioned Krylov solver consisting of the GMRES accelerator [39], coupled with an incomplete LU factorization (ILU) preconditioner. In practice, the preconditioner is only computed once at the beginning of each MD iteration, and is reused for all the inner iterations within that MD iteration. In this work, the number of inner iterations to update the electronic structure per MD iteration is $O(10)$.

The preconditioner and linear system matrix in (21) are stored in compressed sparse row (CSR) format. This allows for an easy and efficient assembly of the block matrix \hat{S} in the approximate inverse algorithm. For our numerical examples in Section 4, the ILU preconditioner used is the level-based ILU(0) preconditioner. For each linear system solve, we assume convergence whenever the initial residual is reduced by a factor of 10^{-12} . In practice, this relative residual tolerance is reached after about 5 iterations.

4 Numerical Results

We evaluate the performance of our algorithm on two condensed matter applications. The first problem is a molecular system of polymers, with periodic boundary conditions. The initial problem is a cubic unit cell of size 30.568 Bohr with 472 atoms (carbon and hydrogen), which results in 536 electronic orbital functions. This problem is replicated by a factor of 2, 3, 4, 5, and 6 in each 3D direction, to generate larger test problems with up to $O(100K)$ atoms. The second example is a simulation of liquid water within a cubic unit cell of size 23.46 Bohr. The initial problem consists of 192 atoms (*i.e.* 64 H_2O molecules), which corresponds

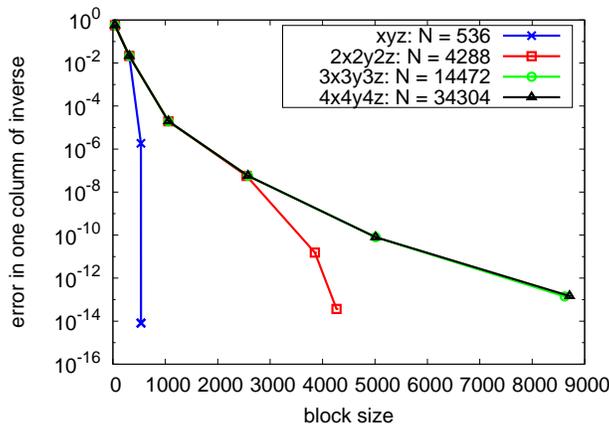


Figure 4: Plot of a single column error vs. block size for computing approximate inverse

to 256 electronic orbitals. For this example, the initial problem is replicated up to a factor of 8 in each 3D direction, to generate larger test problems.

All runs for the results that follow were conducted on an IBM BGQ machine at the Livermore Computing Center of the Lawrence Livermore National Laboratory.

4.1 Truncation error

Recall that Algorithm 1 is used to gather the data needed to build the block matrices used in computing the approximate inverse. In general, the larger R_s is, the larger the resulting block matrix, which typically leads to a more accurate approximation to the inverse. Figure 4 shows a plot of the error in one column of the inverse corresponding to the entries of interest for that column, on the polymer problem. The initial system has size of 472 atoms and 536 electronic orbital functions is replicated up to 4 \times in the X, Y, and Z directions so that the resulting system consists of up to 30,208 atoms and uses 34,304 electronic orbital functions. The figure shows the 2-norm of the error between the approximate inverse computed by using different block sizes, and the exact inverse. The different block sizes are obtained by choosing R_s as different integer multiples of R_c . The figure highlights two main observations: First, as expected, we see that the larger block sizes yield more accurate entries for the approximate inverse. A second and less obvious, yet important observation from these results is that the block size, needed to construct the inverse to a prescribed degree of accuracy, remains the same irrespective of the global problem size. These results show that we can obtain an approximation to the inverse for problems of arbitrarily large sizes, by inverting a small block matrix of dimension independent of the global problem size. This is a direct consequence of the decay property of S^{-1} , and highlights the inversion algorithm’s potential to scale. For completeness, we note that once the size of the region prescribed by R_s gets close to or larger than the size of the global domain, the submatrix \hat{S} gets close or equal to the global matrix S , and the error quickly converges to zero.

The above results highlight the effect of R_s on the error in the computed entries of the approximate inverse. However, in a practical MD simulation, what is important is the effect of this error on the physical quantities of interest, that is, on the atomic forces. In the following example, we evaluate the error on the atomic forces introduced by approximating the elements of S^{-1} . We do this by increasing the radius R_s , using multiples of R_c , to assemble the block matrix for computing the approximate inverse. The error on the atomic forces is then computed by comparing the results of each simulation against that of a simulation that uses exact entries of S^{-1} . In the results that follow, the cutoff radius, R_c , used to confine the electronic orbitals is set to 9.0 Bohr, which has been previously shown to be large enough to achieve the accuracy needed for these types of problems (see [18, 19]). Figure 5 shows an exponential decay of the error on the forces for increasing R_s , for both the water and the polymer problems. In practice, an error in the forces of about 4×10^{-4} or less is considered accurate enough. Thus, in the scaling tests that follow, we choose $R_s = 2R_c = 18$ Bohr.

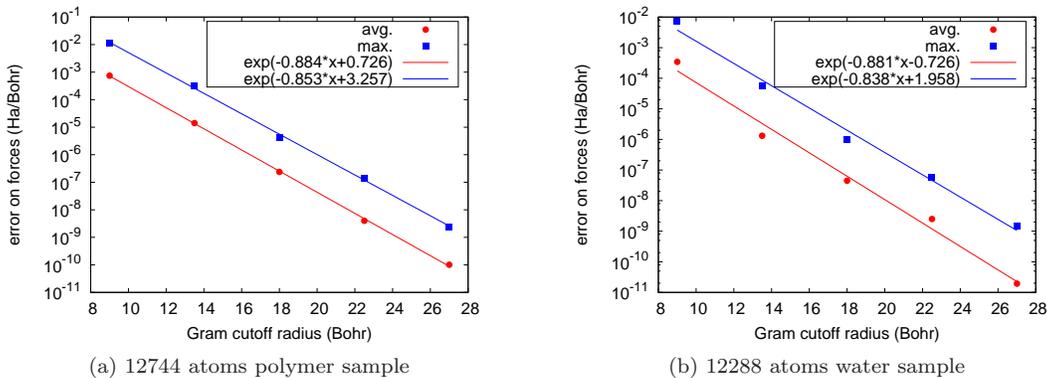


Figure 5: Error on forces as a function of radius R_s , used to form each block in the approximate inverse computation (average and maximum values over 12744 atoms polymer sample (left), and 12288 atoms water sample (right)).

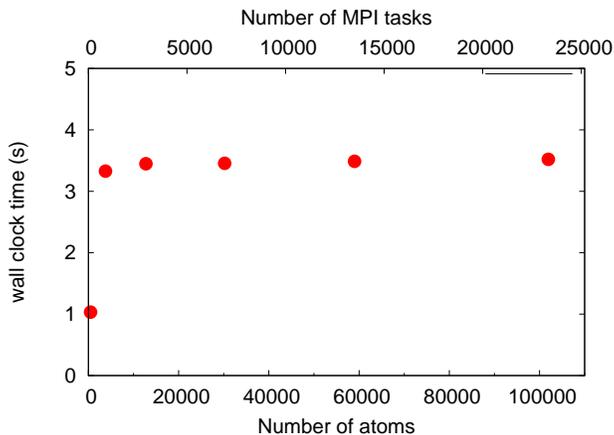


Figure 6: Parallel weak scaling: wall clock time for computing entries of S^{-1} on IBM BGQ architecture as a function of the number of atoms and number of MPI tasks

4.2 Parallel scaling

Next, we assess the weak scaling performance of the inversion algorithm on the polymer problem. Here, the initial problem is decomposed into a $3 \times 6 \times 6$ processor grid. We scale the number of processors with the number of atoms so that the number of MPI tasks per atom remains constant. Figure 6 shows the wall clock times for computing the inverse for each problem size. For this particular case, our choice of R_s prescribes that each processor communicates within a neighborhood of $5 \times 9 \times 9$ processors (subdomains are not cubic). Thus, as we scale up to sufficiently large problem sizes, each processor will perform a total of 20 ($4 + 8 + 8$) sends and receives to gather data to construct \hat{S} . This results in a block matrix of size about 2400 for each MPI task. The figure shows very good weak scaling up to 101,925 atoms using up to 23,328 MPI tasks. In this weak scaling study, the asymptotic regime is reached at the second data point, where the problem is large enough for the block principal submatrix \hat{S} to be smaller than the global S .

We demonstrate the scalability of our algorithm on the problem of liquid water with up to 131,072 electronic functions (98,304 atoms), using 100,352 processors. This choice of the ratio of the number of processors to electronic functions is chosen to be close to the strong scaling limit for this problem using the current implementation of our algorithm, and to obtain a balance between parallel efficiency and a fast time to solution. Figure 7 shows the strong scaling performance of our algorithm on the IBM BGQ machine. The processor grid topology used in the following weak scaling study corresponds to an MPI task per electron

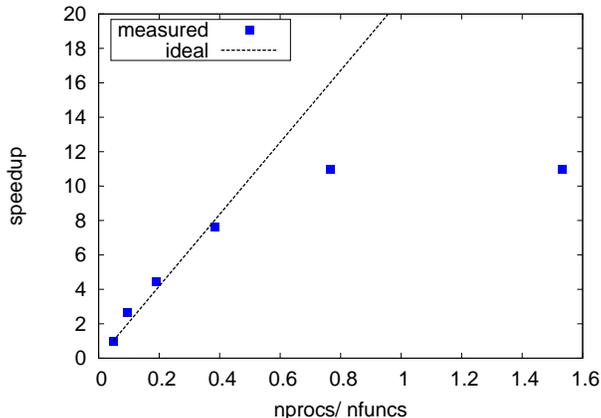


Figure 7: Parallel strong scaling: speedup on IBM BGQ architecture as a function of the number of MPI tasks per electronic wavefunction

function ratio of ≈ 0.8 . Figure 8 shows the wall clock times for one MD step of the weak scaling study, as well as the corresponding wall clock times for the inner iterations to compute the electronic structure within each MD step. Here, each MD iteration requires approximately 9 inner iterations to compute the electronic structure. Once the electronic structure has been computed, the forces on the atoms are computed and used to update the atomic positions and positions. These additional operations, in addition to some setup cost incurred at each MD step due in particular to the displacement of the localization regions, account for the difference between the time for each MD step and the time for computing the electronic structure.

At the beginning of this study, the problem sizes are small, and we observe $O(N^3)$ scaling between the first and second data points. However, as the problem sizes get larger, the profile begins to flatten out as we enter the asymptotic regime of weak scaling. The results indicate that the electronic structure part of the simulation exhibits excellent weak scaling. A careful observation of the scaling profile for the electronic structure part of the simulation reveals that the time to solution is a little faster for large problem sizes. This improvement in performance is primarily due to a reduction in the communication time for these problems. We believe this behavior is because the larger problems are better mapped to the BGQ network architecture, and therefore take advantage of more torus links for communication.

The overall MD simulation also shows good weak scaling, although some slow growth is observed. This observed growth is because at present, unlike the electronic orbitals, some atomic information is replicated on all the processors, merely out of convenience. For small problem sizes, this cost is not discernible. However, as we scale to larger problem sizes, it becomes a bottleneck. We are currently working on improving our implementation to alleviate this bottleneck. This, coupled with additional code optimization, in particular in our implementation of sparse data operations, can lead to a reduction in the time to solution on the BGQ system. We note that running the same example on a Linux cluster with a 2.6GHz Intel Xeon processor gave a time to solution of less than one minute per MD step, up to the fourth data point (in Figure 8). We did not include these results here since we did not have enough resources on the cluster to get data points for larger runs. Threading will be necessary on BGQ to reach such a fast time to solution.

5 Concluding Remarks

We have presented in this paper a scalable algorithm for First-Principles molecular dynamics (FPMD) based on the Kohn-Sham model of density functional theory (DFT). The computational model uses a general non-orthogonal orbital formulation for the energy functional in the DFT equations. We impose some local approximations to the electronic orbitals to obtain an $O(N)$ approximation to the numerical solution of the electronic structure. The energy functional formulation for general non-orthogonal orbitals in the DFT equations requires the knowledge of selected elements of the inverse of the associated Gram matrix S . We have presented a scalable algorithm for computing selected entries of the inverse of the overlap matrix using

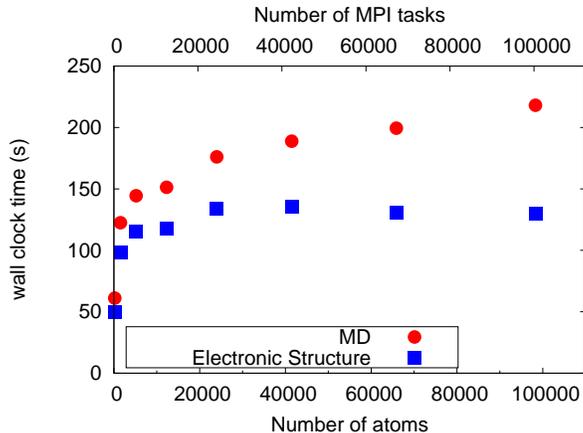


Figure 8: Parallel weak scaling for liquid water: wall clock time for computing 1 MD step, and the corresponding inner iterations to compute the electronic structure for each MD step, on IBM BGQ architecture as a function of the number of atoms and number of MPI tasks

the approximate inverse strategy. We have also presented theoretical and numerical results that support the accuracy of such an approximation.

The FPMD algorithm makes use of a domain decomposition framework and has a modest memory footprint, which is beneficial for large-scale parallel computations on current and future computers with millions of cores. An important ingredient for the efficiency of the parallel algorithm, and its potential to scale, is the use of an efficient data communication strategy. In computing the numerical solution, all (non-local) operations involve communication between only a subset of neighboring processors. A communication algorithm that takes advantage of this nearest neighbor communication pattern is developed and used in the computation of selected entries of S^{-1} . As a result, there is in principle no limit to scaling for the FPMD algorithm, and the numerical results presented demonstrate excellent weak scaling and good time to solution for realistic problems up to $O(100K)$ atoms using $O(100K)$ MPI tasks.

Nonetheless, there remain a few computational challenges that require some attention. First, several real-life applications require long-time MD simulations. As a result, the time to solution is important for these applications. Although our numerical results show good time to solution for each MD iteration, the current algorithm uses a purely MPI implementation. A faster time to solution may be achieved by using threading within each MPI task, in particular on the IBM BGQ architecture. Second, an a priori error estimator is needed to better model the error in the $O(N)$ approximation. This is of particular importance in MD applications since the electronic properties of a system may change over time and lead to a regime where initial truncation parameters are no longer appropriate.

Clearly, computing unoccupied states in general is not compatible with this algorithm, as it would require using a single particle Density Matrix instead of S^{-1} alone [16]. But if we want to determine the lowest unoccupied state only and compute the band gap, this can be done in $O(N)$ operations by computing the lowest eigenvalue of the Hamiltonian restricted to the subspace orthogonal to the occupied states $\{\phi_i\}_{i=1}^N$.

Finally, we note once again that the algorithm presented in this paper is designed for systems with finite band-gaps, that is insulators and semi-conductors. The water system used for the weak scaling study has a large enough band gap so that the inner iterations to compute the electronic structure for each MD step converges in $O(10)$ iterations. Systems with smaller band gaps may require a few extra inner iterations per MD step. The electronic structure calculation for metallic systems is also an important area of interest. However, efficient $O(N)$ algorithms for metallic systems are still a challenge, and are the subject of current research.

Acknowledgements

We wish to thank all the anonymous referees for reviewing this work and for their comments on improving the quality of this paper. We also wish to thank Sebastien Hamel of LLNL for providing us with the example on polymers, and for his time in several discussions on this work.

References

- [1] D. G. ANDERSON, *Iterative procedures for nonlinear integral equations*, J. Ass. Comput. Mach., 12 (1965), pp. 547–560.
- [2] M. W. BENSON, *Iterative solution of large scale linear systems*, Master’s Thesis, Lakehead University, Thunder Bay, Canada, 1973.
- [3] M. W. BENSON AND P. O. FREDERICKSON, *Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems*, Utilitas Math., 22 (1982), pp. 127–140.
- [4] M. BENZI AND N. RAZOUK, *Decay bounds and $O(n)$ algorithms for approximating functions of sparse matrices*, Elec. Trans. Num. Anal., 28 (2007), pp. 16–39.
- [5] M. BENZI AND M. TUMA, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 968–994.
- [6] ———, *A comparative study of sparse approximate inverse preconditioners*, Appl. Numer. Math., 30 (1999), pp. 305–340.
- [7] D. R. BOWLER AND T. MIYAZAKI, *$O(N)$ methods in electronic structure calculations*, Rep. Prog. Phys., 75 (2012), p. 036503.
- [8] K. BRANDHORST AND M. HEAD-GORDON, *Fast sparse Cholesky decomposition and inversion using nested dissection matrix reordering*, J. Chem. Theory and Comput., 7 (2011), pp. 351–368.
- [9] E. CANCES, M. DEFRANCESCHI, W. KUTZELNIGG, C. LEBRIS, AND Y. MADAY, *Computational quantum chemistry: A primer*, in Handbook of Numerical Analysis, P. Ciarlet and C. Le Bris, eds., vol. X, Elsevier Science, 2003, pp. 3–270.
- [10] K. CHEN, *Matrix Preconditioning Techniques and Applications*, Cambridge University Press, Cambridge, UK,, 2005.
- [11] E. CHOW, *A priori sparsity patterns for parallel sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (1999), pp. 1804–1822.
- [12] E. CHOW AND Y. SAAD, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., 19 (1998), pp. 995–1023.
- [13] J. D. F. COSGROVE, J. C. DIAZ, AND A. GRIEWANK, *Approximate inverse preconditioning for sparse linear systems*, Internat. J. Comput. Math., 44 (1992), pp. 91–110.
- [14] S. DEMKO, W. F. MOSS, AND P. W. SMITH, *Decay rates for inverses of band matrices*, Math. Comp, 43 (1984), p. 491.
- [15] J.-L. FATTEBERT, *Accelerated block preconditioned gradient method for large scale wave functions calculations in density functional theory*, J. Comput. Phys., 229 (2010), pp. 441–452.
- [16] J.-L. FATTEBERT AND J. BERNHOLC, *Towards grid-based $O(N)$ density-functional theory methods: Optimized non-orthogonal orbitals and multigrid acceleration*, Phys. Rev. B, 62 (2000), pp. 1713–1722.
- [17] J.-L. FATTEBERT AND M. BUONGIORNO NARDELLI, *Finite difference methods for ab initio electronic structure and quantum transport calculations of nanostructures*, in Handbook of Numerical Analysis, P. Ciarlet and C. Le Bris, eds., vol. X, Elsevier Science, 2003, p. 571.

- [18] J.-L. FATTEBERT AND F. GYGI, *Linear scaling first-principles molecular dynamics with controlled accuracy*, Comput. Phys. Commun., 162 (2004), pp. 24–36.
- [19] ———, *Linear-scaling first-principles molecular dynamics with plane-waves accuracy*, Phys. Rev. B, 73 (2006), p. 115124.
- [20] I. FRIED, *Bounds on the extremal eigenvalues of the finite element stiffness and mass matrices and their spectral condition number*, Journal of Sound and Vibration, 22 (1972), pp. 407–418.
- [21] G. GALLI AND M. PARRINELLO, *Large scale electronic structure calculations*, Phys. Rev. Lett., 69 (1992), pp. 3547–3550.
- [22] G. GALLI AND A. PASQUARELLO, *First-principles molecular dynamics*, in Computer Simulation in Chemical Physics, M. Allen and D. Tildesley, eds., vol. 397 of NATO ASI Series, Springer, Netherlands, 1993, pp. 261–313.
- [23] N. I. M. GOULD AND J. A. SCOTT, *Sparse approximate-inverse preconditioners using norm-minimization techniques*, SIAM J. Sci. Comput., 19 (1998), pp. 605–625.
- [24] M. GROTE AND T. HUCKLE, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.
- [25] F. GYGI, E. W. DRAEGER, M. SCHULZ, B. R. DE SUPINSKI, J. A. GUNNELS, V. AUSTEL, J. C. SEXTON, F. FRANCHETTI, S. KRAL, C. W. UEBERHUBER, AND J. LORENZ, *Large-scale electronic structure calculations of high-Z metals on the BlueGene/L platform*, in SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing, New York, NY, USA, 2006, ACM, p. 45.
- [26] Y. HASEGAWA, J.-I. IWATA, M. TSUJI, D. TAKAHASHI, A. OSHIYAMA, K. MINAMI, T. BOKU, F. SHOJI, A. UNO, M. KUROKAWA, H. INOUE, I. MIYOSHI, AND M. YOKOKAWA, *First-principles calculations of electron states of a silicon nanowire with 100,000 atoms on the k computer*, in Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, SC '11, New York, NY, USA, 2011, ACM, pp. 1–11.
- [27] L. KLEINMAN AND D. M. BYLANDER, *Efficacious form for model pseudopotentials*, Phys. Rev. Lett., 48 (1982), pp. 1425–1428.
- [28] L. KOLOTILINA AND A. YEREMIN, *Factorized Sparse Approximate Inverse Preconditionings I. Theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–48.
- [29] L. Y. KOLOTILINA, *Explicit preconditioning of systems of linear algebraic equations with dense matrices*, Journal of Soviet Mathematics, 43 (1988), pp. 2566–2573.
- [30] S. LI, S. AHMED, AND E. DARVE, *Fast inverse using nested dissection for NEGF*, J. Comput. Electronics, 6 (2007), pp. 187–190.
- [31] L. LIN, C. YANG, J. LU, L. YING, AND E. WEINAN, *A fast parallel algorithm for selected inversion of structured sparse matrices with application to 2d electronic structure calculations*, SIAM J. Sci. Comput., 33 (2011), pp. 1329 – 1351.
- [32] N. MARZARI AND D. VANDERBILT, *Maximally localized generalized Wannier functions for composite energy bands*, Phys. Rev. B, 56 (1997), p. 12847.
- [33] N. OHBA, S. OGATA, T. KOUNO, T. TAMURA, AND R. KOBAYASHI, *Linear scaling algorithm of real-space density functional theory of electrons with correlated overlapping domains*, Comput. Phys. Commun., 183 (2012), pp. 1664–1673.
- [34] D. OSEI-KUFFUOR AND J.-L. FATTEBERT, *Accurate and scalable $O(N)$ algorithm for First-Principles molecular-dynamics computations on large parallel computers*, Phys. Rev. Lett., 112 (2014), p. 046401.

- [35] W. PAI TANG, *Toward an effective sparse approximate inverse preconditioner*, SIAM J. Matrix Anal. Appl, 20 (1999), pp. 970–986.
- [36] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, U.S.A., 1980.
- [37] X. W. PING AND T. J. CUI, *The factorized sparse approximate inverse preconditioned conjugate gradient algorithm for finite element analysis of scattering problems*, Prog. Electromagnetics Res., 98 (2009), pp. 15–31.
- [38] Y. SAAD, *Iterative Methods for Sparse Linear Systems, 2nd edition*, SIAM, Philadelphia, PA, 2003.
- [39] Y. SAAD AND M. SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.
- [40] F. SHIMOJO, R. K. KALIA, A. NAKANO, AND P. VASHISHTA, *Divide-and-conquer density functional theory on hierarchical real-space grids: Parallel implementation and applications*, Phys. Rev. B, 77 (2008), p. 085103.
- [41] M. SPRINGBORG, *Methods of electronic-structure calculations*, Wiley, West Sussex, U.K.r, 2000.
- [42] O. TATEBE AND Y. OYANAGI, *Efficient implementation of the multigrid preconditioned conjugate gradient method on distributed memory machines*, in Supercomputing '94, Proceedings, 1994, pp. 194–203.
- [43] W. YANG, *Direct calculation of electron-density in density-functional theory*, Phys. Rev. Lett., 66 (1991), pp. 1438–1441.
- [44] Z. ZHAO, J. MEZA, AND L.-W. WANG, *A divide-and-conquer linear scaling three-dimensional fragment method for large scale electronic structure calculations*, J. Phys.: Condens. Matter, 20 (2008), p. 294203.