Massively Parallel Implementation of Divide-and-Conquer Jacobi Iterations Using Particle-Mesh Ewald for Force Field Polarization

Dominique Nocito and Gregory J. O. Beran*

Department of Chemistry, University of California, Riverside, California 92521, United States

E-mail: gregory.beran@ucr.edu

Phone: +1 951 827-7869

Abstract

To accelerate the evaluation of the self-consistent polarization in large condensed-phase systems with polarizable force fields, the new divide-and-conquer Jacobi iterations (DC-JI) solver is adapted for periodic boundary conditions with particle-mesh Ewald and implemented in a massively parallel fashion within the Tinker-HP software package. DC-JI captures the mutual polarization of close-range interactions within sub-clusters of atoms using Cholesky decomposition, and couples in the polarization effects between these clusters iteratively. Iterative convergence is accelerated with direct inversion of the iterative subspace (DIIS) extrapolation. Compared to widely used preconditioned conjugate gradient (PCG) or conventional Jacobi iterations (JI/DIIS) algorithms, DC-JI/DIIS solves the polarization equations ∼20–30% faster in protein systems ranging from ∼10,000–175,000 atoms run on hundreds of processor cores. This translates to ∼10–15% speed-ups in the number of nanoseconds of simulation time that can be achieved per day. Not only is DC-JI/DIIS faster than PCG, but it also gives more energetically robust solutions for a given convergence threshold. These improvements make numerically robust polarizable force field simulations more computationally tractable for chemical systems of interest.

1 Introduction

Quantum mechanical charge distributions respond to their environment, which significantly impacts system behaviors. The dipole moment of water can decrease $\sim$20% as it moves from bulk water to a non-polar protein pocket, for instance.[1] Capturing how the charge distribution responds to its environment is also critically important for maintaining the subtle balance of intra-protein and protein-environment interactions correctly[2] or for reproducing the proper dynamics in ionic liquids.[3] Reproducing these effects in classical force field simulations can be difficult—fixed-charge models are inherently incapable of describing dynamic changes in the charge distribution.

Polarizable force fields, on the other hand, can mimic these behaviors.[4–6] Unfortunately, the improved physical description provided by polarizable force fields comes at significantly increased computational cost. In the AMOEBA (atomic multipole optimized energetics for biomolecular applications) force field,[7,8] for example, where polarization is represented via an induced dipole model with Thole damping, obtaining the induced dipoles requires solving a set of linear equations of dimension three times the number of polarizable sites N. These equations can be solved "directly" using a finite number of operations, but such methods are only tractable for small systems due to their $O(N^3)$ complexity. Instead, iterative methods are traditionally used to solve for the induced dipoles. Two commonly used iterative solvers are Jacobi iterations (JI), accelerated with either over-relaxation (JOR)[8] or direct inversion of the iterative subspace (DIIS),[9,10] and the preconditioned conjugate gradient (PCG) algorithm.[11] While these iterative methods vary in their rate of convergence and overall computational cost, they can in principle solve the polarization equations to arbitrary precision. In practice, the solution of the polarization equations is converged to within a computationally tractable user-specified tolerance.

To minimize the computational cost, approximations can be made when solving the polarization equations. Truncated versions of Jacobi iterations[12,13] and preconditioned conjugate gradients.[14,15] have been introduced. These methods typically perform the algorithm for a fixed number of iterations, which leads to several benefits. First, the iterative methods can be "unrolled" to a form for which analytical gradients of the polarization with respect to atomic position can be derived. This potentially minimizes the energy drift associated with iteratively solving the systems of equations to a finite convergence tolerance. Second, the user has a "knob" to control the computational cost of the algorithm by choosing when to truncate. In this sense these methods might be characterized as "direct" in that they can be solved in a finite number of operations. However, the resulting dipole vector differs from the exact one, meaning that the potential energy surface on which the dynamics occur differs from the true one. Similarly, extended Lagrangian approaches eliminate the need to iterate the polarization equations at each time step by integrating the dynamics along an approximate shadow potential.[16–18] Other strategies express the polarization interactions via a truncated many-body expansion, such that mutual polarization is only considered up to 3-body interactions (3-AMOEBA),[19] or neglect mutual polarization completely and reparameterize the force field to compensate (iAMOEBA).[20]

In addition to algorithmic advances, improved hardware has opened the door to applying polarizable force fields for biological systems on chemically relevant timescales that would otherwise have been unattainable. The Tinker software package, which includes the polarizable AMOEBA force field, has expanded to a family of three codes that take advantage of hardware advances: canonical Tinker v8.1,[21] the Tinker-OpenMM[22] package which leverages graphical processing units for fast mixed-precision dynamics, and the Tinker-HP package[23] which enables massively parallel MPI applications on high performance computing systems.

Recently, we introduced a new, formally exact iterative polarization solver, the divide-and-conquer Jacobi iterations (DC-JI) algorithm.[24] DC-JI uses physically-motivated partitioning of the polarization problem to accelerate evaluation of the self-consistent induced dipoles. It partitions the set of polarizable sites into spatially local sub-clusters, which will generally include the strongest polarization interactions. The mutual polarization within each sub-cluster is solved directly within the field generated by more distant polarizable sites outside the sub-cluster. Mutual polarization between sub-clusters is captured iteratively. Convergence of the DC-JI solver can be accelerated further with DIIS and/or the use of fuzzy clustering.[24] Divide-and-conquer methods typically include a recursive component. Indeed, one could implement DC-JI in a recursive manner where the spatially proximal clusters are further sub-divided (potentially all the way down to the level of pairwise polarization), from which the many-body polarization would then be built-up. However, we have found that the single level of partitioning allows efficient solving of the polarization equations. The pseudo-direct nature of DC-JI/DIIS leads to a convergence of the polarization equations in a number of iterations comparable to PCG, but with a lower overall computational cost.

The initial implementation of DC-JI/DIIS in Tinker 7.1 performed ∼30–40% faster than PCG for non-periodic systems on a single processor.[24] In this paper, we adapt DC-JI/DIIS to periodic systems via the particle-mesh Ewald algorithm and introduce a massively parallel implementation within the Tinker-HP software package. Good parallel performance is demonstrated for systems containing hundreds of thousands of atoms and many hundreds of processor cores. Crucially, results presented here show that DC-JI/DIIS is not only faster than the conventional iterative polarization solvers like PCG, but that the solution it obtains at any given convergence threshold is simultaneously more energetically robust.

2 Theory

2.1 Background

The AMOEBA force field assigns permanent multipoles up to quadrupoles and an isotropic dipole polarizability to each atom in the system. The polarization equations are then solved to determine the induced dipoles on each site arising from the permanent multipole moments and the mutually induced dipoles on other sites. The polarization energy is computed by minimizing the functional,

$$E_{pol} = \frac{1}{2}\boldsymbol{\mu}^T\tilde{\mathbf{Z}}\boldsymbol{\mu} - \boldsymbol{\mu}^T\mathbf{V}_0 \tag{1}$$

where $\boldsymbol{\mu}$ is a vector of induced dipoles, $\tilde{\mathbf{Z}}$ is the response matrix defined below, and $\mathbf{V}_0$ is the electric field due to the permanent multipole moments. $\tilde{\mathbf{Z}}$ is a symmetric positive definite matrix with blocks for each atom A, B, C, etc:

$$\tilde{\mathbf{Z}} = \begin{bmatrix} (\boldsymbol{\alpha}_A)^{-1} & -\mathbf{T}^{AB} & -\mathbf{T}^{AC} & \cdots \\ -\mathbf{T}^{BA} & (\boldsymbol{\alpha}_B)^{-1} & -\mathbf{T}^{BC} & \cdots \\ -\mathbf{T}^{CA} & -\mathbf{T}^{CB} & (\boldsymbol{\alpha}_C)^{-1} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{2}$$

Here, $\boldsymbol{\alpha}_A$ is a 3×3 diagonal matrix with the inverse of the isotropic polarizability along the diagonal, and $\mathbf{T}^{AB}$ is the interaction matrix which captures the distance and orientational dependence of the dipole-dipole interaction between atomic sites A and B given by,

$$T^{AB}_{\alpha\beta} = \lambda_5 \frac{3R^{AB}_\alpha R^{AB}_\beta}{(R^{AB})^5} - \lambda_3 \frac{\delta_{\alpha\beta}}{(R^{AB})^3} \tag{3}$$

where R_α and R_β correspond to Cartesian components of the vector between the two sites, capturing the anisotropic contributions to the interactions, and $\delta_{\alpha\beta}$ is the Kronecker delta which switches on the isotropic term. The Thole damping functions λ_3 and λ_5 prevent polarization catastrophe and are defined elsewhere.[7] $\tilde{\mathbf{Z}}$ is symmetric and positive definite, but not diagonally dominant.[15]

The vector of induced dipoles that minimizes the polarization energy is obtained as the solution to the system of linear equations,

$$\tilde{\mathbf{Z}}\boldsymbol{\mu} = \mathbf{V}_0 \tag{4}$$

Upon substitution of Eq 4 into Eq 1, the minimized polarization energy is given by,

$$E_{pol} = \frac{1}{2}\boldsymbol{\mu}^T\mathbf{V}_0 - \boldsymbol{\mu}^T\mathbf{V}_0 = -\frac{1}{2}\boldsymbol{\mu}^T\mathbf{V}_0 \tag{5}$$

To compute forces for a molecular dynamics simulation, the derivative of the polarization energy with respect to atomic positions is given as

$$\frac{dE_{pol}}{dr_k^a} = \frac{\partial E_{pol}}{\partial r_k^a} + \frac{\partial E_{pol}}{\partial \boldsymbol{\mu}}\frac{\partial \boldsymbol{\mu}}{\partial r_k^a} \tag{6}$$

If the dipole vector minimizes the polarization energy, then $\frac{\partial E_{pol}}{\partial \boldsymbol{\mu}}$ is at a stationary point and the second term of Eq 6 vanishes. Iterative equation solvers converge towards the dipole vector that minimizes the energy, but they typically stop at a user-defined convergence threshold for computational expediency. Care must be taken to balance between using a looser-tolerance for computational efficiency and a tighter one for a more numerically exact solution. Loosely converged polarization equations will exhibit a non-zero second term in Eq 6 which can introduce energy drift during a molecular dynamics simulation.

2.2 Polarization Solvers

Jacobi iterations (JI) offers a simple iterative procedure for finding the dipole vector that minimizes the polarization energy functional. It corresponds to the intuitive picture in which site A polarizes site B, site B polarizes A, and the process iterates until self-consistency is achieved for the induced dipoles. Formally, the response matrix $\tilde{\mathbf{Z}}$ can be partitioned into its diagonal elements $\mathbf{D}$ and off-diagonal elements $\mathbf{Y}$, $\tilde{\mathbf{Z}} = \mathbf{D} + \mathbf{Y}$. Substitution of the partitioned $\tilde{\mathbf{Z}}$ into Eq 4 and rearrangement yields,

$$\mathbf{D}\boldsymbol{\mu} = \mathbf{V}_0 - \mathbf{Y}\boldsymbol{\mu} \tag{7}$$

The diagonal matrix $\mathbf{D}$ can be inverted trivially to solve for the induced dipoles $\boldsymbol{\mu}$,

$$\boldsymbol{\mu} = \mathbf{D}^{-1}\left(\mathbf{V}_0 - \mathbf{Y}\boldsymbol{\mu}\right) = \boldsymbol{\alpha}\left(\mathbf{V}_0 + \mathbf{T}\boldsymbol{\mu}\right) \tag{8}$$

However, since the right-hand side depends on $\boldsymbol{\mu}$, Eq 8 must be solved iteratively. JI converges poorly if the spectral radius $p(\mathbf{D}^{-1}(-\mathbf{Y}))$ is near 1 and diverges if $p(\mathbf{D}^{-1}(-\mathbf{Y}))$ is greater than 1. The convergence behavior can be improved by coupling JI with DIIS. That is, after establishing a short history of the induced dipoles during the first few iterations, the induced dipoles are extrapolated via DIIS after each Jacobi iteration. DIIS has a long history in computational chemistry.[25]

Alternatively, the conjugate gradient (CG) method can solve for the polarization by minimizing E_{pol} in Eq 1. Poor convergence behavior is observed when the condition number of $\tilde{\mathbf{Z}}$ is large, but this can be improved by applying a preconditioner (PCG) to solve a modified system of equations:

$$P^{-1}\tilde{\mathbf{Z}}\boldsymbol{\mu} = P^{-1}\mathbf{V}_0 \tag{9}$$

where P^{-1} is some easily computed matrix that approximates $\tilde{\mathbf{Z}}^{-1}$. The diagonal preconditioner used in Tinker-HP is easily parallelized and offers a satisfactory reduction in the condition number for high performance parallel implementations. The robustness of PCG depends on $\tilde{\mathbf{Z}}$ being symmetric and positive definite, and it can be sensitive to numeric precision.[23]

2.3 Divide-and-conquer Jacobi Iteration method

DC-JI amounts to a block JI method[26] with physically motivated blocking.[24] Because nearby polarizable sites are expected to polarize each other most strongly, DC-JI partitions the system into spatially localized sub-clusters of sites.[24] The polarization equations within each sub-cluster are solved directly via Cholesky decomposition. Polarization effects between the sub-clusters are captured iteratively through the contributions of the field in a JI-like fashion. Formally, DC-JI partitions $\tilde{\mathbf{Z}} = \mathbf{Z} + \mathbf{Y}$, but in this case $\mathbf{Z}$ is block diagonal matrix, rather than a diagonal one. Each block of $\mathbf{Z}$ corresponds to a sub-cluster of polarizable sites. These $\mathbf{Z}$ blocks include both the inverse of the polarizability tensors $\boldsymbol{\alpha}$ along the diagonal and dipole-dipole interaction tensors $\mathbf{T}$ between the atoms within the sub-cluster.

Exploiting the block structure of $\mathbf{Z}$, one can break Eq 7 into a separate matrix equation for each sub-cluster of polarizable sites,[24]

$$\mathbf{Z}_{II}\boldsymbol{\mu}_I = \mathbf{V}_{0,I} - \mathbf{Y}_{IJ}\boldsymbol{\mu}_J = \mathbf{V}_{0,I} + \sum_J \mathbf{T}_{IJ}\boldsymbol{\mu}_J \tag{10}$$

where I and J here refer to the different blocks of atoms. In other words, the induced dipoles on atoms in block I depend on the total field, which includes contributions from both the permanent field of the whole system $\mathbf{V}_{0,I}$ and the induced field from atoms in other blocks J, given by $\mathbf{T}_{IJ}\boldsymbol{\mu}_J$. With appropriately sized blocks, one can efficiently solve each block equation for $\boldsymbol{\mu}_I$ directly. Although solving the linear equations formally scales cubically with the block size, sufficiently small block sizes ensure this computational cost remains low, and the number of blocks grows linearly with system size.

2.3.1 Particle-Mesh Ewald

The original implementation of DC-JI was limited to non-periodic systems/direct space interactions.[24] For large systems, the evaluation of the electric field via multiplication of the matrix of the interaction tensors $\mathbf{T}$ and the vector of the multipole moments becomes cost prohibitive. Here, DC-JI is extended to large/periodic systems by combining it with the particle-mesh Ewald (PME) algorithm. PME partitions the total electric field $\mathbf{V}$ into short-range and long-range contributions,

$$\mathbf{V} = \mathbf{V}' + \tilde{\mathbf{V}} \tag{11}$$

The short-range contributions $\mathbf{V}'$ are treated in direct space, while the long-range contributions $\tilde{\mathbf{V}}$ are handled in reciprocal space using fast Fourier transforms. PME allows the long-range electric field contributions to be computed with only $O(N \log N)$ effort.

The direct space contribution is computed by direct particle-particle interactions, contracting the interaction tensors $\mathbf{T}$ with the multipole moments. The reciprocal space contribution is defined such that it excludes unphysical self-interaction terms and short-range interactions. A smoothing function is applied to avoid discontinuities in the forces. Further details on PME and computing $\tilde{\mathbf{V}}$ can be found elsewhere,[6,27,28] The specific details for how the reciprocal space contribution $\tilde{\mathbf{V}}$ is evaluated are not necessary for understanding the PME version of DC-JI here.

Within DC-JI, all interactions must either be captured directly on the left-hand side of Eq 10 or iteratively through the field on the right-hand side. Specific interaction tensor $\mathbf{T}$ elements in $\mathbf{Z}_{II}$ can be zeroed out on the left-hand side, and the corresponding contribution is then included in the field on the right-hand side. Optimal convergence with DC-JI will be achieved when the strongest interactions are captured directly within $\mathbf{Z}_{II}$ on the left-hand side. In the case of PME, the interactions come in two forms: the direct space interactions for which the interaction tensor is available, and reciprocal space ones for which the field is known but the interaction tensor is not readily available. Accordingly, the direct-space-only analog of the $\mathbf{Z}_{II}$ matrix, $\mathbf{Z}'_{II}$, is modified to include only the direct-space $\mathbf{T}'$ interaction tensor elements corresponding to the (typically strong) short-range interactions which are treated in direct space, while the longer-range direct-space interactions between blocks and

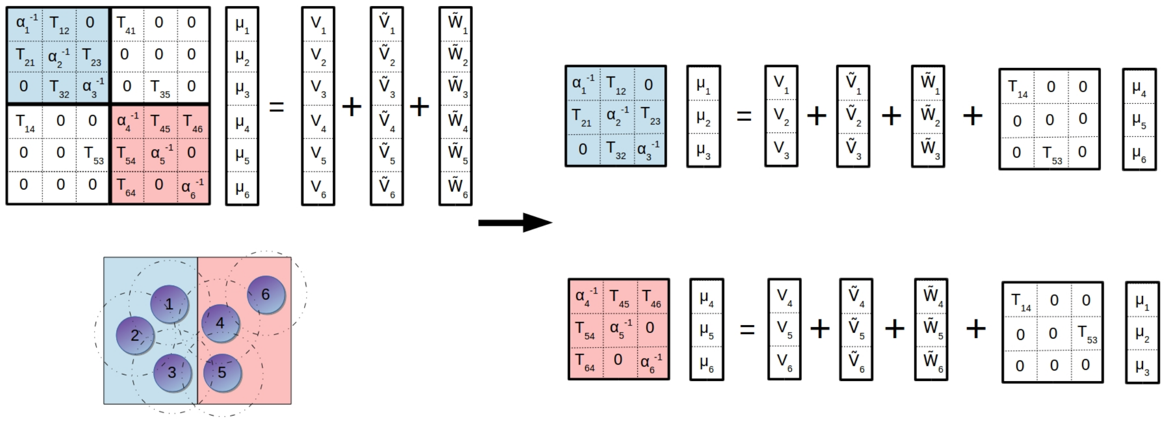


Figure 1: Representation of the polarization equations in DC-JI with PME for a system of six atoms. DC-JI reorganizes the matrix polarization equations on the left to obtain the set of coupled equations on the right that are solved iteratively for each block.

all reciprocal space interactions are treated through the field on the right-hand side. The resulting DC-JI equations with PME are given by,

$$\mathbf{Z}'_{II}\boldsymbol{\mu}_I = \mathbf{V}'_{0,I} + \tilde{\mathbf{V}}_{0,I} + \tilde{\mathbf{W}}_I + \sum_J \mathbf{T}'_{IJ}\boldsymbol{\mu}_J \tag{12}$$

where $\mathbf{V}'_{0,I}$ and $\tilde{\mathbf{V}}_{0,I}$ are the direct and reciprocal space contributions to the permanent field, $\tilde{\mathbf{W}}_I$ is the reciprocal space contribution to the induced field arising from long-range interactions, and $\mathbf{T}'_{IJ}\boldsymbol{\mu}_J$ gives the direct space contributions to the induced field from other blocks J.

Figure 1 depicts these matrix equations graphically for a toy system of six atoms clustered into blue and pink groups. The dotted circles illustrate the radius of the direct space interactions around each atom. For simplicity of illustration, the interactions in this toy system are handled entirely in either direct or reciprocal space, neglecting the smoothing used to transition between the two. The response matrix $\mathbf{Z}'$ is illustrated in the upper left of Figure 1, with diagonal blocks corresponding to the two sub-clusters highlighted in blue and pink. Interactions that lie outside the direct space radius are zeroed out in $\mathbf{Z}'$. On the right hand side of the equation, one finds the contributions from the permanent direct space field $\mathbf{V}'$, the permanent reciprocal space field $\tilde{\mathbf{V}}$, and the induced reciprocal space field $\tilde{\mathbf{W}}$ (whose contributions include the terms zeroed out in $\mathbf{Z}'$). DC-JI then rearranges this matrix equation for the entire system into the set of smaller coupled matrix equations shown on the right half of Figure 1. Those coupled equations corresponding to Eq 12 are solved iteratively.

3 Software Implementation

Tinker-HP offers a platform for massively parallel simulations using polarizable force fields.[23] Double precision operations are used throughout. Distributed memory setup also allows for modeling large systems, with the largest simulation to-date including over 23 million atoms.[23] This is accomplished by using a 3D spatial decomposition of the chemical system which can then be distributed across thousands of processors. Tinker-HP contains two parallelization strategies for the polarization solvers that differ in how/when the bottleneck evaluation of the electric field is handled. The "sequential" scheme evaluates the reciprocal and direct space contributions sequentially, distributing the work for each over all available processor cores. Because the reciprocal space contributions do not parallelize as efficiently as the direct space ones, the "load-balancing" scheme partitions the processor cores into two groups: a larger fraction of cores which focus on the direct space, and a smaller fraction that perform the reciprocal space evaluations. The load-balancing scheme is potentially faster if the processors are appropriately partitioned such that the two components finish at the same time. In practice, however, the sequential approach proved slightly faster in our testing with DC-JI/DIIS and PCG (at least for the numbers of processor cores used here), so the remainder of this work discusses that implementation.

Because the serial DC-JI/DIIS algorithm without PME has been described in detail,[24] this section focuses on features specific to the new parallel PME implementation. While that earlier work examined both overlapping ("fuzzy") and non-overlapping sub-clusters, for

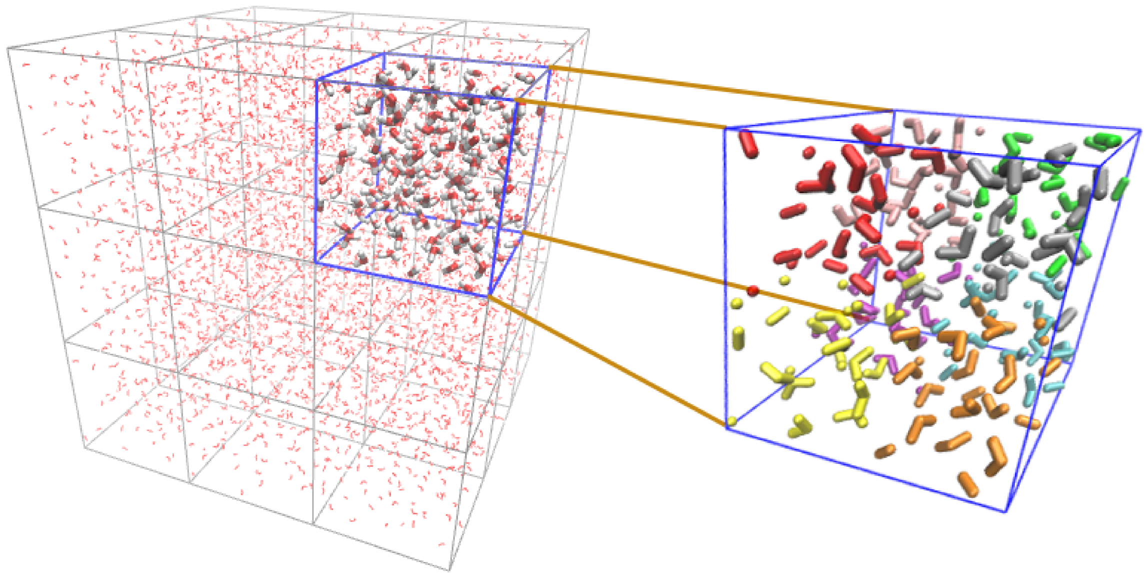


Figure 2: Cartoon of the partitioning scheme employed in the parallel DC-JI algorithm. First, Tinker-HP partitions the entire system into a set of domains, with one domain per processor (large boxes at left). For DC-JI, each domain is further partitioned into a sub-clusters according to the coloring in the enlarged domain at right. In practice, the sub-clusters average 60 atoms each.

simplicity of the parallel implementation the current work focuses on the non-overlapping case. Throughout the algorithm, high parallel performance is accomplished by starting communications as soon as possible and folding computational work into the period before a synchronization bottleneck. To do this non-blocking MPI routines are used with reception being done as early as possible and communication waits done as late as possible. The communications bottleneck occurs in the reciprocal space contributions. Communication for the direct space terms is comparatively minimal. PCG requires only sending the updated dipoles and the descent direction to all other processes at each iteration. Both DC-JI/DIIS and JI/DIIS communicate only the updated dipoles and the contributions to the DIIS extrapolation matrix.

The first step is to identify the sub-clusters of atoms that define the DC-JI blocking of $\mathbf{Z}'$. Previously, this was performed via K-means clustering of the entire system into a user-defined number of blocks.[24] For large systems, global K-means clustering could become expensive. Tinker-HP already implements a three-dimensional spatial domain decomposition,[23] dividing the system into one domain per processor (Figure 2), and each processor evaluates the forces and coordinate updates for the atoms in its domain. Calculations of interactions between two atoms in different domains are handled using the midpoint method, which assigns the interaction to the domain in which the midpoint of the two sites lies. These domains are defined before the software enters the polarization routines.

The parallel implementation of DC-JI exploits this existing domain decomposition as a starting point. Each processor further partitions its domain into a series of sub-clusters. Performing the DC-JI clustering within the existing Tinker-HP domains has two advantages: the resulting DC-JI algorithm implementation is modular and works regardless of the specific domain decomposition scheme, and it reduces communication by ensuring that each block is entirely contained within a single domain. This approach does mean that the solution to the polarization equations varies slightly with the number of processors—changing the number of processors alters the composition of the domains and sub-clusters, which in turn changes which portions of the problem will be solved iteratively and non-iteratively. However, the variations in the resulting polarization energies and induced dipoles are smaller than the user-chosen convergence criterion and do not present any practical issues.

With the system partitioned into smaller domains, K-means clustering could be performed efficiently within each domain. However, since the atoms have already been sorted into spatially local domains, we adopt a simpler, non-iterative approach for forming sub-clusters that divides the domain into the appropriate number of sub-systems, striving for similar volumes and roughly equal dimensions. The domain is partitioned by dividing it into evenly spaced portions along each coordinate axis such that the appropriate total number of blocks is obtained. Assuming uniform density of the atoms, this procedure will allocate similar numbers of atoms to each block. Dividing atoms from a single molecule over multiple sub-clusters does not cause any problems.[24] This scheme performs the clustering slightly faster than K-means (since it does not require the iterative atom-atom distance calculations needed in K-means), and the convergence of the DC-JI polarization equations is comparably good.

Empirically, the DC-JI algorithm performs fastest with block sizes of 60 ± 20 atoms. A given processor core will solve the DC-JI equations for all blocks lying within its domain. For context, a system of 175,000 atoms (e.g. roughly the size of the COX-2 system discussed in the Results section) run on 480 processor cores would be divided into 480 domains with $\sim$365 atoms each. The DC-JI partitioning further subdivides each domain into six blocks of $\sim$60 atoms each.

Once the DC-JI blocks are defined, remaining initialization steps involve constructing the permanent field (both direct and reciprocal space contributions), building the $\mathbf{Z}'_{II}$ blocks (and exploiting their symmetric nature), evaluating and storing the Cholesky decomposition of those blocks $\mathbf{Z}'_{II} = \mathbf{L}_I \mathbf{L}_I^T$ (for facile subsequent solution of the linear equations), and obtaining the initial guess induced dipole moments. The reciprocal space field is constructed in parallel as described elsewhere,[23] and the results are communicated between processors. In the first molecular dynamics time step, the guess dipoles are obtained as the polarizabilities contracted with the total permanent field, $\boldsymbol{\mu} = \boldsymbol{\alpha}\mathbf{V}_0$. In later time steps, guess dipoles are obtained from the previous steps (e.g. using the previous converged dipoles or Kolafa's always stable predictor-corrector algorithm[29]). Either way, the initial dipoles are communicated between processors.

Next, the iterative portion begins. The reciprocal contribution to the induced field $\tilde{\mathbf{W}}_I$ is constructed and communicated. The direct space contributions to the field for each block I from atoms in other blocks J are evaluated using a neighbors list. Once the total field (right-hand side of Eq 12) has been built, the induced dipoles in each block are solved via Cholesky decomposition (using the already factorized version of $\mathbf{Z}'_{II}$). In the first iteration, these dipoles are communicated and the algorithm proceeds to the next iteration. Starting at the end of the second iteration, DIIS extrapolation is performed before dipole communication. Each processor evaluates its contribution to the inner products of the residual vectors needed for DIIS[24] and communicates those. Then the processor extrapolates its induced dipoles and communicates them. The iterations continue until the root-mean-square change in the induced dipole moments is smaller than the user-selected convergence threshold. To summarize, each iteration requires communication of the reciprocal field contributions, the DIIS matrix updates, and the resulting induced dipole moments. All other work is done locally.

4 Computational Details

DC-JI/DIIS has been implemented in Tinker-HP v1.1, and will be publicly available in future releases. The parallel implementation was tested on Comet at the San Diego Supercomputer Center (SDSC). Each node includes two Intel Xeon E5-2680 v3 processors with 128 GB DDR4 DRAM (64 GB per socket). The network utilizes 56 Gbps fourteen data rate (FDR) InfiniBand with full bisection bandwidth on each rack and 4:1 oversubscription bandwidth between racks.

Intra-node communication quickly can become the bottleneck. This is primarily due to the communications necessary for the fast Fourier transforms (FFTs) used in computing the reciprocal space contribution to the electric field. For this reason we have performed test using varying amounts of cores on a node to test the optimal use of the network cards. For all three algorithms (JI/DIIS, PCG, and DC-JI/DIIS), optimal performance was seen when half the cores on a node were used. The relative performances of the three polarization solvers remained consistent with the varying fractions of cores used per node. Empirical testing found that for up to the 720 cores used here, the load balancing approach was slower than the sequential procedure. Of course, this might change if even more cores were used.

All molecular dynamics simulations were performed in the microcanonical (NVE) ensemble with the reversible reference system propagator algorithm (RESPA) multi-time step integrator[30] using a 1 fs time step for non-bonded forces and 0.5 fs time step for the bonded forces. A range of thresholds for converging the polarization solvers were examined, and the

10^{-5} threshold provides satisfactory stability. Kolafa's always stable predictor-corrector was used to obtain initial induced dipoles at the beginning of each time step.[29] All DC-JI/DIIS runs requested an average block size of 60 atoms. A 7 Å Ewald cutoff for PME and 9 Å cutoff for the van der Waals interactions were used.

Performance testing is carried out on three different protein systems in explicit aqueous solution: ubiquitin, dihydrofolate reductase (DHFR), and cyclooxygenase-2 (COX-2). These systems are representative of typical biological problems for which one might use polarizable force fields and span a wide range of system sizes. The smallest system, ubiquitin, has been extensively studied due to its prevalence in eukaryotic organisms. Here, it consists of the 1,227-atom protein surrounded by 2,835 waters (9,732 atoms total).[31] The ubiquitin system has unit cell dimensions of 54.99×41.91×41.91 Å and employed a 72×54×54 PME grid. The DHFR system was taken from the joint Amber/CHARM benchmark.[32] DHFR is necessary in the path to form purines and pyrimidines, which act as the building blocks for DNA and RNA. The system consists of 2,489 protein atoms in a box of 7,023 waters (23,558 atoms total). The DHFR system occupies a cubic box of dimension 62.23 Å, and a 64×64×64 PME grid was used. The COX-2 system was taken from the Tinker benchmark suite.[33] COX-2 is an enzyme that is a part of the rate limiting step in the formation of prostanoids. The system consist of the 17,742 protein atoms surrounded by 52,159 waters (174,219 atoms total). The COX-2 system has cubic box edge length of 120 Å, and a 128×128×128 PME grid was used. All three systems have been studied previously using Tinker-HP.[23]

5 Results And Discussion

5.1 Energy Convergence and Drift

Converging an iterative solver to a finite tolerance leaves residual error in the resulting induced dipoles. If the residual error is too large, it can cause energy drift as discussed in Section 2.1. Accordingly, we first investigate the energy conservation of DC-JI/DIIS and

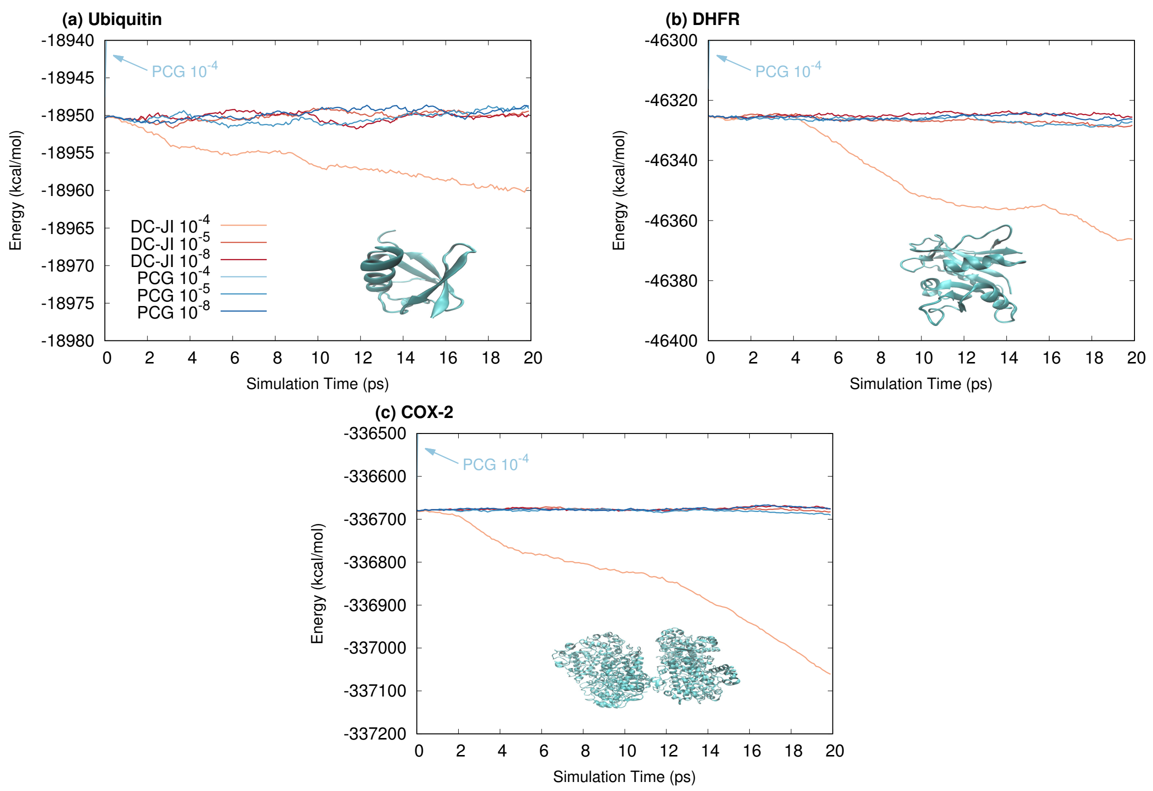


Figure 3: A 20 ps NVE simulation using a 1 fs time step and three different polarization convergence criteria for (a) Ubiquitin, (b) DHFR, and (c) COX-2. Note that PCG with a threshold of 10^{-4} D diverges out of frame within the first picosecond for all simulations.

PCG with various convergence thresholds. The comparison focuses on PCG (as implemented in Tinker-HP) because of its widespread use and good numerical and computational performance.[11,23,31] Figure 4 plots the energies for 20 ps NVE simulations for all three protein systems with a 1 fs time step and three different polarization convergence thresholds. The loose 10^{-4} Debye (D) convergence threshold leads to divergent behavior for both PCG and DC-JI/DIIS. However, whereas the PCG energy diverges out of the plot frame within a fraction of a picosecond, the DC-JI/DIIS energy drifts more slowly over the 20 ps simulation. This suggests that DC-JI/DIIS is effectively converging the equations more robustly. Tightening the convergence threshold by one order of magnitude to 10^{-5} D is sufficient to stabilize both the PCG and DC-JI/DIIS trajectories, with both algorithms exhibiting similar energy conservation. For both solvers, the energy conservation with a 10^{-5} D convergence criteria is similar to that from the much tighter 10^{-8} D criterion, with the the looser 10^{-5} D criterion

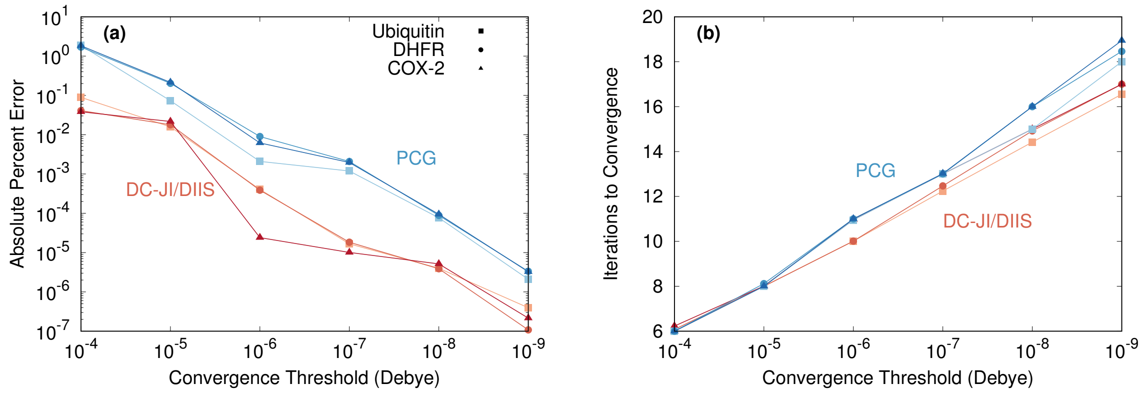


Figure 4: (a) Absolute error in the converged polarization energy relative to the DC-JI/DIIS results with a 10^{-10} Debye convergence threshold, and (b) the number of iterations necessary to achieve that convergence (without using dipoles from previous time steps). For a given convergence threshold, DC-JI/DIIS achieves more converged energy than PCG in the same number of iterations or fewer. Values were averaged over 100 configuration snapshots.

requiring fewer iterations (typically 4) and less overall computational effort to converge.

To obtain further insight into the differences between DC-JI/DIIS and PCG, 100 configuration snapshots were extracted from a 10 ps NVE simulation of each protein at 100 fs intervals. Single-point energy evaluations were performed on each snapshot using either PCG or DC-JI/DIIS and various convergence thresholds. Initial guess dipoles were obtained directly from the permanent field, without using any information from previous time steps. Figure 4a plots the absolute percent error in the polarization energy relative to the DC-JI/DIIS energy obtained with a 10^{-10} Debye convergence criterion, averaged over the 100 snapshots. As the convergence criterion is tightened, DC-JI/DIIS and PCG do converge to the same induced dipoles and polarization energy. The use of percent error normalizes across the different energy scales of the differently sized systems. For a given convergence threshold, DC-JI/DIIS consistently exhibits an energy error that is 1–2 orders of magnitude smaller than for PCG. Moreover, Figure 4b shows that DC-JI/DIIS achieves this better energy convergence in the same number of iterations as PCG or fewer.

The good numerical behavior of DIIS-JI/DIIS can be understood in terms of two factors. First, the direct solution of each block provides fully self-consistent polarization within the blocks at every iteration. For atoms near the center of each block, the iterative contributions

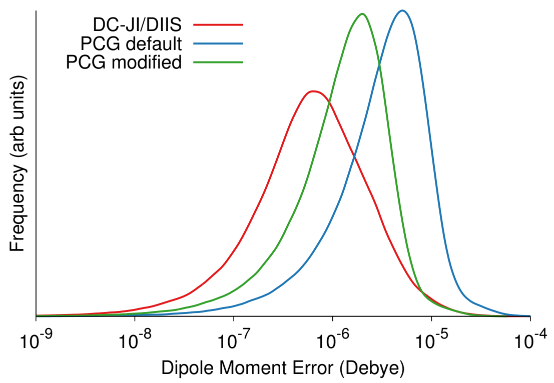


Figure 5: Distribution of the errors in the COX-2 induced dipole components for DC-JI/DIIS, PCG with the default convergence metric used in Tinker-HP, and PCG modified to use the same convergence metric as DC-JI/DIIS. For the same 10^{-5} Debye stopping criterion, the errors in the DC-JI/DIIS induced dipoles are considerably smaller. Comparable error distributions were observed in the other two protein systems (not shown).

from neighboring blocks will be comparatively weaker, leading to smaller errors in those induced dipoles. Second, the DC-JI/DIIS convergence criterion definition is based on the root-mean-square change in the dipole vector, while the PCG implementation in Tinker-HP weights those dipole residuals by the preconditioner. For a given convergence threshold, the PCG criterion leads to slightly less-converged dipoles in practice. Modifying the PCG implementation to use the same convergence criterion definition as DC-JI/DIIS causes PCG to require one additional iteration to converge. Doing so does bring the PCG energetics closer to those for DC-JI/DIIS, but the percent error in energy for modified PCG remains ∼3–10 times larger than for DC-JI/DIIS, depending on the convergence threshold.

Further insight is obtained by looking at the distribution of errors in the induced dipoles, as shown in Figure 5 for one COX-2 snapshot with a 10^{-5} convergence criterion. In this figure, the dipole moment errors were computed against those dipoles converged with a 10^{-16} Debye criterion, at which point the dipoles from PCG and DC-JI/DIIS are effectively identical within double precision. Compared to the default Tinker-HP implementation of PCG, DC-JI/DIIS obtains significantly smaller dipole errors at convergence, which translates to the improved energetics described above. The DC-JI/DIIS error distribution is broader, with many more dipoles exhibiting very small errors, which reflects how the direct solution effectively captures polarization within the blocks self-consistently at each iteration. If one modifies the PCG convergence metric to match the one used by DC-JI/DIIS, the extra iteration it requires to converge does improve the dipole error distribution. Still, DC-JI/DIIS still exhibits many more tightly converged induced dipoles for the same 10^{-5} Debye stopping criterion. It is possible that preconditioners more sophisticated than the diagonal one used in PCG here could alter this analysis.[11]

5.2 Computational Efficiency

Next, we examine the overall parallel performance of DC-JI/DIIS, JI/DIIS, and PCG. Figure 6 examines the total timings for one self-consistent polarization time step for each algorithm on 120 processor cores in the large COX-2 system. The results were averaged over 100 time steps taken sampled from a trajectory in which the initial dipoles were obtained via Kolafa's predictor-corrector guess. The vast majority of the computational effort is associated with construction of the permanent field (during initialization) and induced fields (at each iteration). The computational time required to evaluate the permanent electric field $\mathbf{V}$ and $\tilde{\mathbf{V}}$ is practically identical for PCG and JI/DIIS. Doing the same for DC-JI/DIIS takes slightly longer, since this step also includes building the $\mathbf{Z}'$ matrices for later use. All three algorithms use identical routines to build the induced reciprocal field $\tilde{\mathbf{W}}$ during the iterations, so the time required for that step varies only on the number of iterations required to converge the induced dipoles. Both DC-JI/DIIS and PCG required 4.0 iterations on average and therefore required virtually identical amounts of time to build $\tilde{\mathbf{W}}$, while JI/DIIS required more time due to its average of 5.3 iterations. Building the induced direct space field $\mathbf{W}$ is fastest for DC-JI/DIIS, since a portion of the interactions are already computed and stored during the initialization as part of $\mathbf{Z}'$, whereas PCG recalculates those at each iteration. JI/DIIS requires even longer, since it both recomputes all the interaction contributions at each iteration and requires additional iterations to converge.

Including all the other steps (factoring $\mathbf{Z}'$ during the initialization, solving for the dipoles, and other miscellaneous steps/communication), DC-JI/DIIS solves for the dipoles in 0.28 seconds. In about the same amount of time, PCG performs all steps except for the work associated with computing the initial descent direction. However, computing that initial descent direction costs about the same as one PCG iteration, making PCG more expensive overall at 0.33 seconds. In other words, at a 10^{-5} D convergence threshold, DC-JI/DIIS is faster than PCG by approximately the cost of a single iteration ($\sim$20%). At tighter thresholds of 10^{-6}–10^{-8} D, DC-JI/DIIS converges one iteration faster than PCG, so the net computational savings would correspond to approximately two iterations worth. Individual JI/DIIS iterations are less expensive than for the other two algorithms, but it requires more of them, and JI/DIIS requires a slightly longer 0.34 seconds overall to compute the induced dipoles.

Finally, Figure 7 examines parallel timings for the polarization solver alone and for the total simulation (i.e. including evaluation of all bonded and non-bonded contributions) for the three proteins with DC-JI/DIIS, PCG, and JI/DIIS. For the DHFR and ubiquitin systems, timings are reported up to 480 cores. On 480 cores, ubiquitin and DHFR have an average of 20 and 49 atoms per domain, respectively. This means that each domain consists of just one small DC-JI block. Increasing the number of cores further would decrease the DC-JI block size and lead to diminishing returns as the block size decreases. To utilize more cores, one would probably want to switch to the load-balancing parallelization strategy. Allocating some processors to work exclusively on the reciprocal space contributions translates to fewer, larger direct-space domains. For the much larger COX-2 system we test up to 720 cores, which corresponds to 242 atoms per domain, or about six DC-JI blocks per domain.

Figure 7 clearly demonstrates that solving the for the induced dipoles with DC-JI/DIIS is faster than using either PCG and JI/DIIS for all numbers of cores, and the computational savings increase with the number of cores employed. The analysis above indicates that DC-JI/DIIS is effectively one iteration faster than PCG in timings, or 20% faster at a 10^{-5} D convergence threshold. Indeed, for COX-2, DC/JI proves on average $\sim$20% faster than PCG, and ~27% faster than JI/DIIS. Similar average speed-ups of 17–18% over PCG and 29-31% over JI/DIIS are observed for the two smaller proteins. Larger 30–40% acceleration was seen in our earlier work on non-periodic systems.[24] The DC-JI algorithm accelerates only the treatment of direct space interactions. In the non-periodic case, all interactions are treated in direct space. For periodic systems modeled with PME, the DC-JI algorithm only impacts the non-reciprocal space portions of the calculation, thereby reducing the possible savings. Differences in the PCG preconditioner between Tinker-HP and Tinker 8.1 might also play a role.

Unsurprisingly, the best parallel performance is observed for the largest systems. However, the fractional speed-up provided by DC-JI/DIIS over the other two polarization solvers is fairly consistent across different numbers of processor cores, typically increasing by a few percentage points as the number of cores is increased. For sufficiently large numbers of cores, however, the parallel efficiency of all three algorithms decreases, due largely to the communication bottleneck associated with the reciprocal space terms. For COX-2, building the reciprocal space contributions to the permanent and induced fields in DC-JI/DIIS consumes around 30% of the computational time on 12 cores, but this increases to over 60% of the computational time on 720 cores. This behavior manifests in the plateauing of the parallel performance curves around 240 cores. Those computational costs are essentially the same across the three solvers, aside from any differences in the number of iterations required to reach convergence. Thanks to the minimal communication required, the direct-space portions of DC-JI/DIIS parallelize very efficiently. Perhaps the biggest limitation comes when the number of cores is sufficiently large that the number of atoms per domain falls below the optimal ~40–80 atom block size. For very small blocks (~10–20 atoms or fewer), fewer polarization interactions are solved for directly, and the number of iterations required to converge the dipoles begins to increase slowly as DC-JI/DIIS asymptotes toward the JI/DIIS model.[24]

Finally, switching focus from the polarization solvers to the total simulation time in

Figure 7, we observe similar overall parallel performance and relative efficiencies of the algorithms, except that the total computational savings from DC-JI/DIIS are smaller. The non-polarization portions account for about half the total simulation time, and therefore the ∼20–30% acceleration in the polarization solver translates to a ∼10–15% increase in the number of nanoseconds of simulation time per day.

6 Conclusions

In summary, the DC-JI/DIIS polarization solver has been implemented for periodic systems via particle-mesh Ewald in the massively parallel Tinker-HP software package. DC-JI/DIIS solves the AMOEBA polarization equations up to 20% faster than PCG and 30% faster than JI/DIIS. Factoring in all other steps in the force field evaluation, DC-JI/DIIS reduces the overall simulation time by ∼10–15%. Such performance is observed on chemical systems with tens or hundreds of thousands of atoms running on hundreds of processor cores. Furthermore, at a given convergence threshold, the polarization energies obtained from DC-JI/DIIS are closer to the exact solution than those from PCG. This translates to better numerical stability, as evidenced by the decreased energy drift in simulations with loosely-converged induced dipoles. Overall, given the combination of superior numerical behavior and decreased computational effort for DC-JI/DIIS over PCG and JI/DIIS, the new DC-JI solver can be recommended as an excellent alternative to traditional self-consistent polarization solvers.

The chief bottleneck in the parallel implementation is the treatment of reciprocal space contributions to the field. For the COX-2 system, the proportion of the calculation spent evaluating those terms more than doubled to ∼60%. Future work should consider strategies for further improving the parallel efficiency of the reciprocal space terms. To some extent this might be addressed by load-balancing schemes that partition the direct and reciprocal space portions across different numbers of processors. Alternatively, new algorithms for handling the long-range interactions efficiently on large numbers of processors would be very valuable.

Nevertheless, the current implementation of DC-JI/DIIS enables polarizable force fields with mutual polarization to be applied to systems with hundreds of thousands of atoms.

7 Acknowledgments

Funding for this work from the National Science Foundation (CHE-1665212) and supercomputer time from XSEDE (TG-CHE110064) are gratefully acknowledged. We thank Dr. Jean-Philip Piquemal and Dr. Louis Lagardere for providing guidance and helpful discussions.

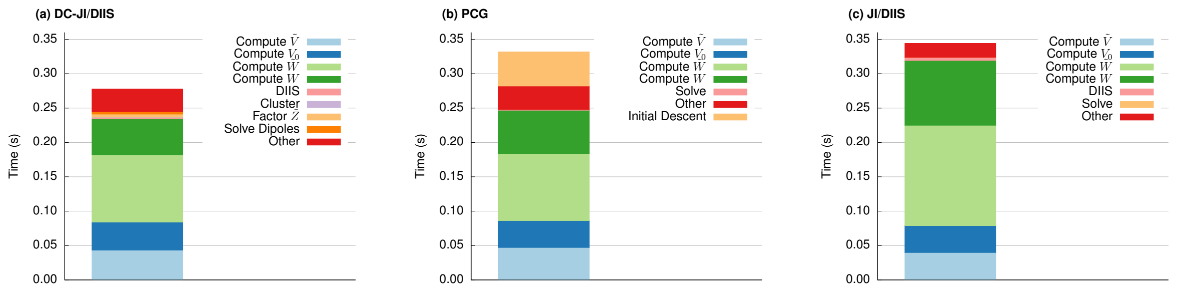


Figure 6: Timing breakdowns for the COX-2 system (174,219 atoms) averaged over 100 steps. A block size of ∼60 atoms was used for DC-JI/DIIS. Timings were performed using 120 cores on SDSC Comet. DC-JI/DIIS and PCG converged in 4.0 iterations on average, while JI/DIIS required an average 5.3 iterations. "Other" includes computing the guess dipoles, summing of the electric fields, evaluation of convergence criterion, and communications between processes, and any other minor steps not explicitly mentioned.

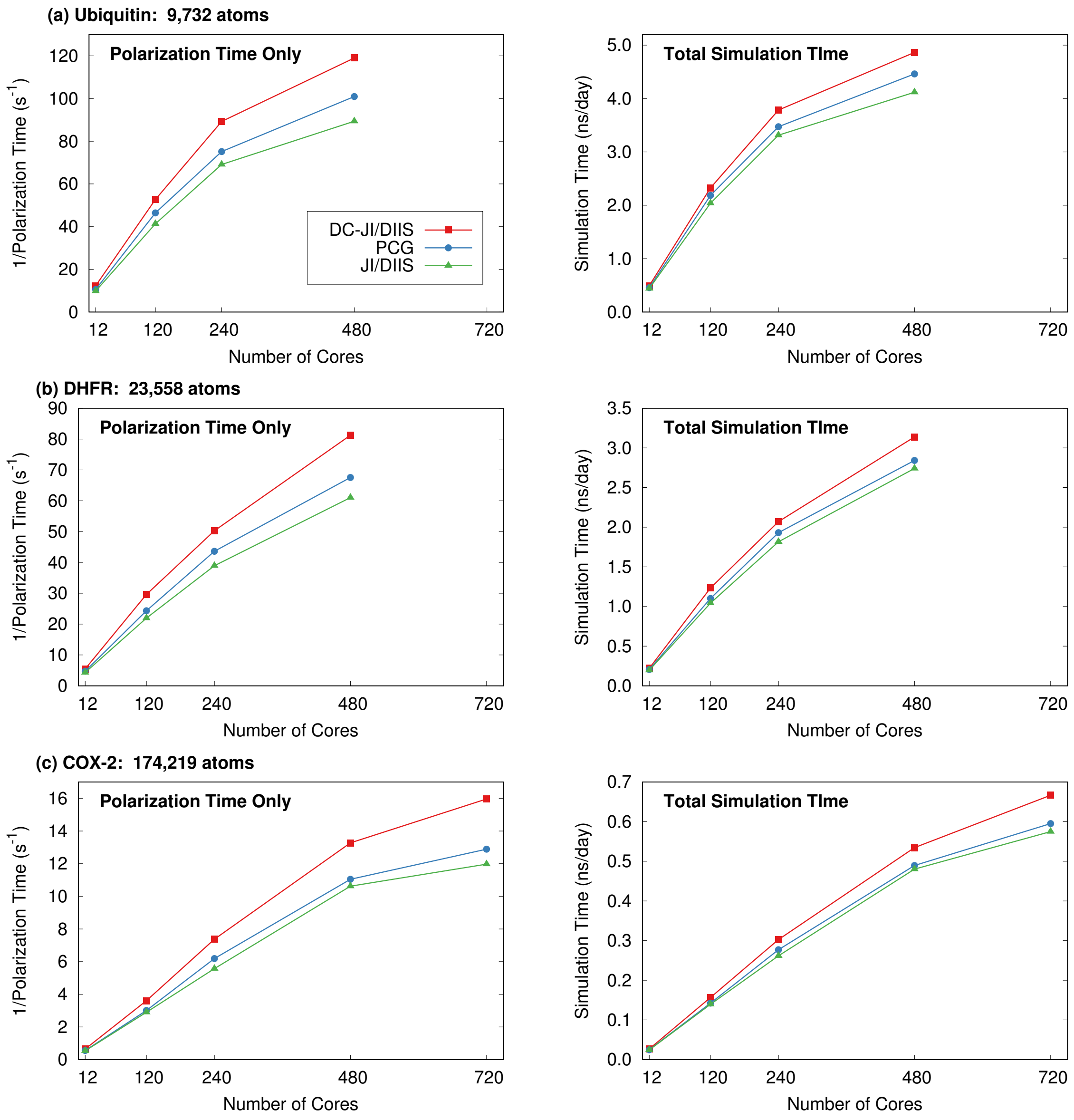


Figure 7: Performance of the DC-JI/DIIS, PCG, and JI/DIIS polarization solvers (left) and the corresponding total simulation time (right) for (a) Ubiquitin (b) DHFR, and (c) COX-2. The reciprocal of the polarization time is plotted for ease of comparison with the total simulation time.

References

(1) Morozenko, A.; Leontyev, I. V.; Stuchebrukhov, A. A. Dipole Moment and Binding Energy of Water in Proteins from Crystallographic Analysis. *Journal of Chemical Theory and Computation* **2014**, *10*, 4618–4623.

(2) Ji, C.; Mei, Y. Some Practical Approaches to Treating Electrostatic Polarization of Proteins. *Acc. Chem. Res.* **2014**, *47*, 2795–2803, PMID: 24883956.

(3) Yan, T.; Burnham, C. J.; Del Popolo, M. G.; Voth, G. A. Molecular Dynamics Simulation of Ionic Liquids: The Effect of Electronic Polarizability. *J. Phys. Chem. B* **2004**, *108*, 11877–11881.

(4) Warshel, A.; Kato, M.; Pisliakov, A. V. Polarizable force fields: History, test cases, and prospects. *Journal of Chemical Theory and Computation* **2007**, *3*, 2034–2045.

(5) Cieplak, P.; Dupradeau, F.; Duan, Y.; Wang, J. Polarization Effects in Molecular Mechanical Force Fields. *J. Phys. Condens. Mat.* **2009**, *21*, 333102.

(6) Cisneros, G. A.; Karttunen, M.; Ren, P.; Sagui, C. Classical Electrostatics for Biomolecular Simulations. *Chem. Rev.* **2014**, *114*, 779–814, PMID: 23981057.

(7) Ren, P.; Ponder, J. W. Polarizable Atomic Multipole Water Model for Molecular Mechanics Simulation. *J. Phys. Chem. B* **2003**, *107*, 5933–5947.

(8) Ren, P.; Ponder, J. W. Polarizable Atomic Multipole Water Model for Molecular Mechanics Simulation. *J. Phys. Chem. B* **2003**, *107*, 5933–5947.

(9) Pulay, P. Convergence acceleration of iterative sequences. the case of scf iteration. *Chem. Phys. Lett.* **1980**, *73*, 393 – 398.

(10) Pulay, P. Improved SCF convergence acceleration. *J. Comput. Chem.* **1982**, *3*, 556–560.

(11) Wang, W.; Skeel, R. D. Fast evaluation of polarizable forces. *J. Chem. Phys.* **2005**, *123*, 164107.

(12) Simmonett, A. C.; Pickard IV, F. C.; Shao, Y.; Cheatham III, T. E.; Brooks, B. R. Efficient treatment of induced dipoles. *J. Chem. Phys.* **2015**, *143*, 074115.

(13) Simmonett, A. C.; Pickard IV, F. C.; Shao, Y.; Cheatham III, T. E.; Brooks, B. R. Efficient treatment of induced dipoles. *J. Chem. Phys.* **2015**, *143*, 074115.

(14) Aviat, F.; Levitt, A.; Stamm, B.; Maday, Y.; Ren, P.; Ponder, J. W.; Lagardère, L.; Piquemal, J.-P. Truncated Conjugate Gradient: An Optimal Strategy for the Analytical Evaluation of the Many-Body Polarization Energy and Forces in Molecular Simulations. *J. Chem. Theory Comput.* **2017**, *13*, 180–190, PMID: 28068773.

(15) Aviat, F.; Lagardère, L.; Piquemal, J.-P. The truncated conjugate gradient (TCG), a non-iterative/fixed-cost strategy for computing polarization in molecular dynamics: Fast evaluation of analytical forces. *J. Chem. Phys.* **2017**, *147*, 161724.

(16) Albaugh, A.; Demerdash, O.; Head-Gordon, T. An efficient and stable hybrid extended Lagrangian/self-consistent field scheme for solving classical mutual induction. *J. Chem. Phys.* **2015**, *143*, 174104.

(17) Albaugh, A.; Niklasson, A. M. N.; Head-Gordon, T. Accurate Classical Polarization Solution with No Self-Consistent Field Iterations. *J. Phys. Chem. Lett.* **2017**, *8*, 1714–1723, PMID: 28350167.

(18) Albaugh, A.; Head-Gordon, T.; Niklasson, A. M. N. Higher-Order Extended Lagrangian Born-Oppenheimer Molecular Dynamics for Classical Polarizable Models. *J. Chem. Theory Comput.* **2018**, *14*, 499–511.

(19) Demerdash, O.; Head-Gordon, T. Convergence of the Many-Body Expansion for Energy

and Forces for Classical Polarizable Models in the Condensed Phase. *J. Chem. Theory Comput.* **2016**, *12*, 3884–3893, PMID: 27405002.

(20) Wang, L.-P.; Head-Gordon, T.; Ponder, J. W.; Ren, P.; Chodera, J. D.; Eastman, P. K.; Martinez, T. J.; Pande, V. S. Systematic Improvement of a Classical Molecular Model of Water. *J. Phys. Chem. B* **2013**, *117*, 9956–9972, PMID: 23750713.

(21) J. W. Ponder, TINKER v8.1, 2018, `http://dasher.wustl.edu/tinker/`; Accessed March 14, 2018.

(22) Harger, M.; Li, D.; Wang, Z.; Dalby, K.; Lagardère, L.; Piquemal, J.-P.; Ponder, J.; Ren, P. Tinker-OpenMM: Absolute and relative alchemical free energies using AMOEBA on GPUs. *J. Comput. Chem.* **2017**, *38*, 2047–2055.

(23) Lagardère, L.; Jolly, L.-H.; Lipparini, F.; Aviat, F.; Stamm, B.; Jing, Z. F.; Harger, M.; Torabifard, H.; Cisneros, G. A.; Schnieders, M. J.; Gresh, N.; Maday, Y.; Ren, P. Y.; Ponder, J. W.; Piquemal, J.-P. Tinker-HP: a massively parallel molecular dynamics package for multiscale simulations of large complex systems with advanced point dipole polarizable force fields. *Chem. Sci.* **2018**, *9*, 956–972.

(24) Nocito, D.; Beran, G. J. O. Fast divide-and-conquer algorithm for evaluating polarization in classical force fields. *J. Chem. Phys.* **2017**, *146*, 114103.

(25) Rohwedder, T.; Schneider, R. An analysis for the DIIS acceleration method used in quantum chemistry calculations. *J. Math. Chem.* **2011**, *49*, 1889.

(26) Demmel, J. W. *Applied Numerical Linear Algebra, 2nd ed*; SIAM, 1997.

(27) Wang, W.; Skeel, R. D. Fast evaluation of polarizable forces. *J. Chem. Phys.* **2005**, *123*, 164107.

(28) Lagardère, L.; Lipparini, F.; Polack, E.; Stamm, B.; Cancès, E.; Schnieders, M.; Ren, P.; Maday, Y.; Piquemal, J.-P. Scalable Evaluation of Polarization Energy and Associated

Forces in Polarizable Molecular Dynamics: II. Toward Massively Parallel Computations Using Smooth Particle Mesh Ewald. *J. Chem. Theory Comput.* **2015**, *11*, 2589–2599, PMID: 26575557.

(29) Kolafa, J. Time-reversible always stable predictor-corrector method for molecular dynamics of polarizable molecules. *J. Comp. Chem.* **2003**, *25*, 335–342.

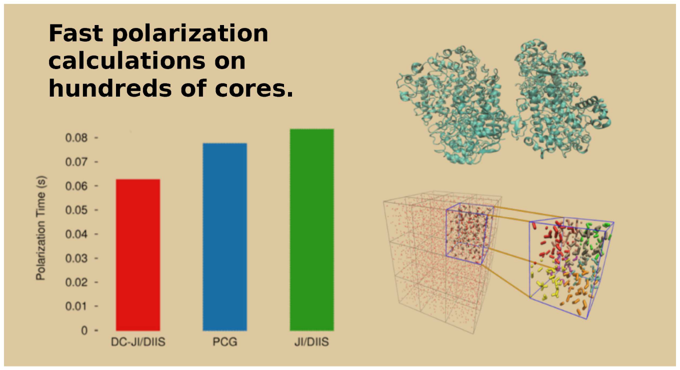
(30) Tuckerman, M.; Berne, B. J.; Martyna, G. J. Reversible multiple time scale molecular dynamics. *J. Chem. Phys.* **1992**, *97*, 1990–2001.

(31) Lipparini, F.; Lagardère, L.; Stamm, B.; Cancès, E.; Schnieders, M.; Ren, P.; Maday, Y.; Piquemal, J.-P. Scalable Evaluation of Polarization Energy and Associated Forces in Polarizable Molecular Dynamics: I. Toward Massively Parallel Direct Space Computations. *J. Chem. Theory Comput.* **2014**, *10*, 1638–1651.

(32) AMBER/CHARMM DHFR Benchmark. 2018, `http://ambermd.org/amber8.bench2.html`; Accessed April 2, 2018.

(33) Tinker Benchmark Suite. 2018, `https://dasher.wustl.edu/tinker/distribution/bench/`; Accessed April 2, 2018.

Graphical TOC Entry


Fast polarization calculations on hundreds of cores.
Polarization Time (s)
DC-JI/DIIS
PCG
JI/DIIS