Scalable Evaluation of Polarization Energy and Associated Forces in Polarizable Molecular Dynamics: II.Towards Massively Parallel Computations using Smooth Particle Mesh Ewald

Louis Lagardère,[†,‡] Filippo Lipparini,[¶,‡,†,§] Étienne Polack,[¶,‡] Benjamin Stamm,[¶,∥] Éric Cancès,[⊥] Michael Schnieders,[#] Pengyu Ren,[@] Yvon Maday,[¶,△,▽] and Jean-Philip Piquemal[*,‡]

UPMC Univ. Paris 06, Institut du Calcul et de la Simulation, F-75005, Paris, France, UPMC Univ. Paris 06, UMR 7617, Laboratoire de Chimie Théorique, F-75005, Paris, France, UPMC Univ. Paris 06, UMR 7598, Laboratoire Jacques-Louis Lions, F-75005, Paris, France, current address: Institut für Physikalische Chemie, Universität Mainz, Duesbergweg 10-14, D-55128 Mainz, Germany, CNRS, UMR 7598 and 7616, F-75005, Paris, France, Université Paris-Est, CERMICS, Project-team Micmac, INRIA-Ecole des Ponts, 6 & 8 avenue Blaise Pascal, 77455 Marne-la-Vallée Cedex 2, France, Departments of Biomedical Engineering and Biochemistry, The University of Iowa, Iowa City, Iowa 52358, United States, Department of Biomedical Engineering, The University of Texas at Austin, Austin, Texas 78712, United States, Institut Universitaire de France, and Brown Univ, Division of Applied Maths, Providence, RI, USA

E-mail: jpp@lct.jussieu.fr

Abstract

In this paper, we present a scalable and efficient implementation of dipole-based polarizable force fields for molecular dynamics (MD) simulations with periodic boundary conditions. The Smooth Particle-Mesh Ewald technique is combined with two optimal iterative strategies, namely, a preconditioned conjugate gradient solver and a Jacobi solver in conjunction with the Direct Inversion in the Iterative Subspace for convergence acceleration, to solve the polarization equations, as introduced in a previous paper. We show that both solvers exhibit very good parallel performances and overall very competitive timings in an energy-forces computation needed to perform a MD step. Various tests on large systems are provided in the context of the polarizable AMOEBA force field as implemented in the newly developed Tinker-HP package. We show that using a large number of cores offers a significant acceleration of the overall process involving the iterative methods within the context of SPME and a noticeable improvement of the memory management giving access to very large systems (hundreds of thousands of atoms) as the algorithm naturally distributes the data on the different cores. Coupled with advanced MD techniques, gains ranging from 2 to 3 orders of magnitude in time are now possible compared to non-optimized, scalar implementations giving new directions for polarizable molecular dynamics in periodic boundary conditions using massively parallel implementations.

[*]To whom correspondence should be addressed

[†]UPMC Univ. Paris 06, Institut du Calcul et de la Simulation, F-75005, Paris, France

[‡]UPMC Univ. Paris 06, UMR 7617, Laboratoire de Chimie Théorique, F-75005, Paris, France

[¶]UPMC Univ. Paris 06, UMR 7598, Laboratoire Jacques-Louis Lions, F-75005, Paris, France

[§]current address: Institut für Physikalische Chemie, Universität Mainz, Duesbergweg 10-14, D-55128 Mainz, Germany

[‖]CNRS, UMR 7598 and 7616, F-75005, Paris, France

[⊥]Université Paris-Est, CERMICS, Project-team Micmac, INRIA-Ecole des Ponts, 6 & 8 avenue Blaise Pascal, 77455 Marne-la-Vallée Cedex 2, France

[#]Departments of Biomedical Engineering and Biochemistry, The University of Iowa, Iowa City, Iowa 52358, United States

[@]Department of Biomedical Engineering, The University of Texas at Austin, Austin, Texas 78712, United States

[△]Institut Universitaire de France

[▽]Brown Univ, Division of Applied Maths, Providence, RI, USA

1 Introduction

Polarizable force fields have been, in the last decade, the subject of an intense development.[1–8] The ability of anisotropic polarizable molecular mechanics (APMM) to model complex systems, including charged or very polar ones, biological substrates containing metal ions, weakly interacting molecules or ionic liquids is of prime interest in the molecular dynamics community,[8–12] as it is their robustness and transferability.[8] Furthermore, such force fields have been recently successfully employed in conjunction with quantum mechanical models[13–23] in order to accurately reproduce environmental effects on structural and photophysical properties of chromophores in complex biological substrates, expanding their use to the QM/MM community. Several strategies can be used to introduce polarization in a classical force field, including fluctuating charges,[20,24–26] Drude's oscillators,[23,27] the Kriging method[28] and induced dipoles.[29–33] Independent of the model, all polarizable force fields share the need to determine the polarization degrees of freedom for a given geometry, which requires to solve a set of linear equations;[34,35] furthermore, a polarization term is added to the energy and thus it is required to compute the associated forces. With respect to standard, additive force fields, such a characteristic introduces a heavy computational overhead, as for each step of a molecular dynamics simulation (or for each value of the QM density in a QM/MM computation), one needs to solve a linear system whose size depends on the number of the polarizable sites. A standard, direct approach, such as the use of LU or Cholesky decomposition, becomes rapidly unfeasible for large systems, as it requires a computational effort that scales as the cube of the number of atoms, together with quadratic storage: the development of efficient iterative techniques is therefore a mandatory step in extending the range of applicability of polarizable force fields to large and very large molecular systems.

Iterative techniques require one to perform several matrix-vector multiplications, representing a computational effort quadratic in the size of the system: an efficient implementation has therefore three different points to adress: i) a fast convergent iterative procedure, in order to limit as much as possible the number of matrix-vector multiplications, ii) a fast matrix-vector multiplication technique, and in particular a linear scaling technique that allows one to overcome the quadratic bottleneck and iii) an efficient parallel setup, in order to distribute the computational work among as many processors as possible, exploiting thus modern, parallel computers.

In a recent paper,[34] we focused on the various iterative techniques suitable to solve the polarization equations for dipole-based polarizable force fields and we found that two algorithms, namely, the preconditioned conjugate gradient (PCG) method and Jacobi iterations with convergence acceleration based on Pulay's Direct Inversion in the Iterative Subspace (JI/DIIS), were particularly suitable for the purpose, both in terms of convergence properties and in terms of parallel implementation, addressing therefore the first and the third points of an optimal implementation. The aim of this paper is to extend such results to simulations performed by using periodic boundary conditions (PBC), for which we present a new, improved and tailored parallel implementation of both the PCG and JI/DIIS solvers to compute the polarization energy together with a specific procedure to compute the associated forces.

The use of periodic boundary conditions is a natural choice when simulating intrinsically periodic systems, such as crystals or pure liquids, and is a commonly used procedure to simulate solvated systems with explicit solvent molecules including proteins, ions, etc. Notice that an alternative approach, based on non-periodic boundary conditions and polarizable continuum solvation models, has also been recently proposed for APMM.[36,37]

When PBC are employed, the use of the particle-mesh Ewald method becomes mandatory because of both its computational efficiency and of the physical accuracy of the results it provides. The PME method is based on the Ewald summation[38] introduced to compute the electrostatic energy of a system in PBC and has been originally formulated by Darden et al.[39]. This first formulation, that involved Lagrange polynomias, introduced forces that were not analytical derivatives of the energy and another formulation using B-splines, the Smooth Particle Mesh Ewald (SPME), was given by Essmann et al.[40], allowing analytical differentiation to get the forces and so ensuring energy conservation. Such an approach, first developped for distributed point charges only, was extended to distributed multipoles by Sagui et al.[41] and to distributed hermite gaussian densities by Cisneros et al.[42].

A crucial feature of the SPME method is that it allows to compute the involved matrix-vector products required to determine the induced dipoles in iterative procedures with a $\mathcal{O}(N \log N)$ computational effort, addressing therefore the second aforementioned point in order to get an efficient and scalable implementation. In fact, the SPME method computes the electrostatic interactions as the sum of a direct space contribution, which is limited to close neighbors, plus a reciprocal (Fourier) space one, which can be efficiently computed thanks to the Fast Fourier transform (FFT). However, this subdivision and the FFT itself make achieving a scalable parallel implementation a more complex task than for direct space simulations, as various load-balancing and communication issues have to be addressed within the FFT. A detailed analysis of these aspects is presented in this paper and various technical solutions to overcome these difficulties are proposed. Our new implementation, in the context of the Tinker-HP software, exhibits not only a promising parallel scaling, but also very competitive overall computational times, allowing one to compute the polarization energy and forces of a system composed by almost three hundred thousands atoms in as little as 0.7 seconds.

The paper is organized as follows. In Section 2, we introduce the SPME formalism in the general context of APMM using distributed point multipoles. In Section 3, we explicitly formulate the polarization energy and introduce the different iterative methods to solve the polarization equations. In Section 4, we explore the parallel behavior of the different iterative algorithms as well as the parallel behavior of the computation of the forces. In Section 5, some numerical results are provided. We conclude the paper in Section 6 with some conclusions and perspectives.

2 PME with multipolar interactions

2.1 Notations

In this paper, we will use the same notation as in our previous paper.[43] Let $\mathbf{r}^{\{N\}}$ be the system composed of a neutral unit cell U containing N atoms at positions $\{\vec{r}_i\}_{i=1}^{N}$ replicated in all directions. Let $(\vec{a}_1, \vec{a}_2, \vec{a}_3)$ be the vectors defining the unit cell. We will indicate vector quantities by using an arrow if the vectors are in $\mathbb{R}^3$ and with the bold font if they represent a collection of three-vectors. For instance, $\boldsymbol{\mu}$ will be a $3N$-dimensional column vector $(\vec{\mu}_1, \ldots, \vec{\mu}_N)^T$, where each $\vec{\mu}_i = (\mu_i^x, \mu_i^y, \mu_i^z)$ is a three-dimensional column vector (in particular, a dipole). We will use Latin indexes as a subscript to refer to different atomic sites and Greek indexes as superscripts to indicate the Cartesian component of a vector, so that μ_i^α denotes the α-th Cartesian component of the vector $\vec{\mu}_i$.

For any integers n_1, n_2, n_3 and m_1, m_2, m_3 we will note

$$\vec{n} = n_1\vec{a}_1 + n_2\vec{a}_2 + n_3\vec{a}_3 \qquad \vec{m} = m_1\vec{a}_1^* + m_2\vec{a}_2^* + m_3\vec{a}_3^*,$$

where $(\vec{a}_1^*, \vec{a}_2^*, \vec{a}_3^*)$ is the dual basis of $(\vec{a}_1, \vec{a}_2, \vec{a}_3)$ (*i.e.* $\vec{a}_\alpha^* \cdot \vec{a}_\beta = \delta_{\alpha\beta}$).
Let us define the fractional coordinates of the atoms:

$$\vec{a}_\alpha^* \cdot \vec{r}_i, \quad \alpha = 1, \ldots, 3; i = 1, \ldots, N. \tag{1}$$

In the AMOEBA polarizable force field with which we have been working, the charge density of a system is approximated by a set of permanent atomic multipoles (up to quadrupoles) located at the atomic positions $\vec{r}_i$, which we will denote as follows: $q_i, \vec{\mu}_{si}, \Theta_i$ for $i = 1, \ldots, N$.

2.2 Ewald Summation and Smooth Particle Mesh Ewald

In this subsection, we will review how to compute electrostatic interactions between permanent multipoles in periodic boundary conditions by using the Ewald summation and the

Smooth Particle Mesh Ewald method. The idea is to split the electrostatic energy into three terms, two being sums with good convergence properties and the third not depending on the atomic positions.

The electrostatic potential created by the permanent multipoles at a point $\vec{x}$ is obtained by applying the associated multipole operator L_i to $|\vec{r}_i - \vec{x}|^{-1}$ where, for $i = 1, \ldots, N$:

$$L_i = q_i + \vec{\mu}_{si} \cdot \nabla_i + \Theta_i : \nabla_i \nabla_i, \tag{2}$$

where $\nabla_i \nabla_i$ is the 3×3 matrix defined by: $(\nabla_i \nabla_i)_{\alpha\beta} = \partial_{r_i^\alpha} \partial_{r_i^\beta}$ Let us define the multipole structure factor $S(\vec{m})$:

$$S(\vec{m}) = \sum_{j=1}^{N} \tilde{L}_j(\vec{m}) \exp(2i\pi \vec{m} \cdot \vec{r}_j) \tag{3}$$

with $\tilde{L}_j$ being the Fourier transform of the multipole operator associated to site j:

$$\tilde{L}_i(\vec{m}) = q_i + 2i\pi \vec{\mu}_{si} \cdot \vec{m} - (2\pi)^2 \Theta_i : M, \tag{4}$$

where $(M)_{\alpha\beta} = m_\alpha m_\beta$.

The total electrostatic energy E_{elec} is given by :

$$E_{\text{elec}}(\mathbf{r}^{\{N\}}) = \frac{1}{2} \sum_{\vec{n}} \sum_{1 \le i,j \le N}^{(n)} L_i L_j \left(\frac{1}{|\vec{r}_j - \vec{r}_i + \vec{n}|} \right), \tag{5}$$

where the exponent (n) means that the terms $i = j$ are not summed up when $\vec{n} = \vec{0}$. The Ewald summation technique[38] allows[44] to separate the total electrostatic energy of $\mathbf{r}^{\{N\}}$ (equation (5)) in three components: a contribution E_{dir} consisting of short range interactions computed in real space, a global contribution E_{rec} computed in the reciprocal and a self energy term E_{self}, which is a bias arising from the Ewald summation and has no physical meaning. These contributions are given by:

$$E_{\text{dir}} = \frac{1}{2}\sum_{\vec{n}}\sum_{1\le i,j\le N}^{(n)} L_i L_j \left(\frac{\text{erfc}\left(\beta|\vec{r}_j-\vec{r}_i+\vec{n}|\right)}{|\vec{r}_j-\vec{r}_i+\vec{n}|}\right), \tag{6}$$

$$E_{\text{rec}} = \frac{1}{2\pi V}\sum_{\vec{m}\neq 0}\frac{\exp(-\pi^2\vec{m}^2/\beta^2)}{\vec{m}^2}S(\vec{m})S(-\vec{m}) \tag{7}$$

$$E_{\text{self}} = -\sum_{1\le j\le N}\left(\frac{\beta}{\sqrt{\pi}}q_j^2 + \frac{2\beta^3}{3\sqrt{\pi}}\vec{\mu}_j^2 + \frac{2\beta^3}{9\sqrt{\pi}}q_j\,\text{Tr}(\Theta_j) + \frac{8\beta^5}{45\sqrt{\pi}}\Theta_j:\Theta_j\right) \tag{8}$$

$$. \tag{9}$$

where V is the volume of the unit cell: $V = \vec{a}_1\cdot(\vec{a}_2\wedge\vec{a}_3)$.
This formulation introduces a parameter β that controls the range of the direct term and conversely the importance of the reciprocal term. The direct term is computed using a cutoff (that depends on β) as the erfc function in this term decreases to zero on a characteristic distance that depends on this parameter. We therefore have:

$$E_{\text{elec}} = E_{\text{dir}} + E_{\text{rec}} + E_{\text{self}} \tag{10}$$

Note that, as done by Wang and Skeel[35], we neglect the surface term that arises during the derivation of the ewald summation. Note that we have chosen to use the self energy that is commonly used in the literature[31,45–47] and that this does not change the associated forces in molecular dynamics. However, as will be shown by Polack et al.[48], other forms may be derived for this term.

Scaling factors and damping functions[30,31,49] are often used in force fields to parametryze electrostatic interactions at short range, for instance in order to exclude close (1-2, 1-3, 1-4) neighbors; here, we assume that the only interactions concerned are between atoms within the unit cell. Because the Ewald summation corresponds to unscaled interactions, one has to take this into account by adding a correction to the energy computed with this method. Suppose that the interaction between multipoles of sites i and j is scaled and/or damped by a factor s_{ij}, then a correction

$$(s_{ij} - 1) L_i L_j \left(\frac{1}{|\vec{r}_j - \vec{r}_i|} \right) \tag{11}$$

has to be added to the total energy. Let us define E_{corr} the sum of all these terms:

$$E_{\text{corr}} = \frac{1}{2} \sum_{i=1}^{N} \sum_{j \in M(i)} (s_{ij} - 1) L_i L_j \left(\frac{1}{|\vec{r}_j - \vec{r}_i|} \right) \tag{12}$$

where $M(i)$ is the list of atom sites whose multipoles interaction with multipoles of site i is scaled. For the case of the computation of the polarization energy, where all dipole-dipole interactions are included, such scaling parameters originate from Thole's damping, which is introduced in order to avoid the so-called polarization catastrophe.[30,49]

As explained in ref. 46, the total electrostatic potential can be split in four contributions $\Phi_{\text{dir}}, \Phi_{\text{rec}}$, Φ_{self} and Φ_{corr}, with

$$\Phi_{\text{rec}}(\vec{r}_i) = \frac{1}{\pi V} \sum_{\vec{m} \neq 0} \frac{\exp(-\pi^2 \vec{m}^2 / \beta^2)}{\vec{m}^2} S(\vec{m}) \exp(-2i\pi \vec{m} \cdot \vec{r}_i) \tag{13}$$

In the context of the SPME method, the fractional coordinates of the atoms that appear in the reciprocal term are scaled by three factors (one for each coordinate). Indeed, let K_1, K_2, K_3 be such factors then the corresponding scaled fractional coordinates of atom i are given by u_{1i}, u_{2i}, u_{3i} .

$$u_{\alpha i} = K_\alpha \vec{a}^*_\alpha \cdot \vec{r}_i, \tag{14}$$

In consequence, the complex exponentials present in the reciprocal potential can be written:

$$\exp(2i\pi \vec{m}\vec{r}_j) = \exp(2i\pi m_1 \frac{u_{1j}}{K_1}) \exp(2i\pi m_2 \frac{u_{2j}}{K_2}) \exp(2i\pi m_3 \frac{u_{3j}}{K_3}) \tag{15}$$

The complex exponentials appearing in the structure factor are interpolated on a grid of size

$K_1 \times K_2 \times K_3$ using B-splines of a certain order p. Using B-splines makes the reciprocal potential continuously differentiable and allows to derive forces that are analytical derivatives of E_{rec}. Then the reciprocal potential can be approximated by:

$$\Phi_{\text{rec}}(\vec{r}_i) \approx \sum_{\vec{n}} \theta_p(u_{1i} - n_1)\theta_p(u_{2i} - n_2)\theta_p(u_{3i} - n_3)(G^R \star Q^R)(\vec{n}) \tag{16}$$

where θ_p is the usual cardinal B-spline, G^R is the influence function defined by its Fourier transform in Sagui et al[50] and Q^R is the real space multipolar array associated to the atomic multipoles of the system:

$$Q^R(k_1, k_2, k_3) = \sum_{j=1}^{N} \sum_{n_1,n_2,n_3} L_j(\theta_p(u_{1j} - k_1 - K_1 n_1)\theta_p(u_{2j} - k_2 - K_2 n_2)\theta_p(u_{3j} - k_3 - K_3 n_3)) \tag{17}$$

Thus, two sets of parameters controls the approximation level of the SPME algorithm: K_1, K_2, K_3 that defines the size of the interpolation grid along each axis, and p the order of the B-splines that these complex exponential are interpolated with.

From a computational point of view, the standard SPME algorithm follows these steps (the computation of the self term is trivial):

1. Compute the real space potential. The derivatives of the erfc function can be computed recursively as it is shown in ref. 51. One can also use a McMurchie Davidson recursion to compute these derivatives as shown in ref. 50. and 38

2. Compute the correction potential due to scaling factors of the electrostatic interactions (usually done at the same time than the computation of the real space potential)

3. Compute the reciprocal potential

   (a) Build the multipolar array Q^R associated with the atomic multipoles of the system

   (b) Compute the FFT of this array Q^F

(c) Multiply Q^F with G^F, i.e., the Fourier transform of G^R (convolution in Fourier space)

(d) Compute the backward FFT of the result to get the convolution of Q^R and G^R

(e) Multiply the result with the values of the B-splines at the position where the potential is to be evaluated.

3 Evaluation of the Polarization Energy and its Derivatives in Periodic Boundary Conditions

In this section we will see how the previously described methods, i.e., Ewald summation and Smooth Particle Mesh Ewald, can be used in order to compute the polarization energy and the associated forces in a dipole-based polarizable force-field.

In this kind of force field, each atom is endowed by a polarizability $\alpha_i \in \mathbb{R}^{3x3}$, which describes the linear response of such atom's contribution to the density of charge to an electric field. Let us call $\{\vec{\mu}_i\}_{i=1}^N$ the corresponding induced dipole moments. Let us define the vectors $\vec{E}_i$ for $i = 1, \ldots, N$ the electric fields created by the permanent multipoles of the system and their image at site i. In a similar fashion to what was described in the last section, one can apply the Ewald summation to compute the electric fields $\vec{E}_i$.

$$\vec{E}_i = \vec{E}_{i,\mathrm{dir}} + \vec{E}_{i,\mathrm{rec}} + \vec{E}_{i,\mathrm{corr}} + \vec{E}_{i,\mathrm{self}} \tag{18}$$

where

$$\vec{E}_{i,\mathrm{dir}} = \sum_{\vec{n}} \sum_{j=1}^{N}{}^{(n)} \frac{\partial}{\partial \vec{r}_i} L_j \left(\frac{\mathrm{erfc}(\beta|\vec{r}_j - \vec{r}_i + \vec{n}|)}{|\vec{r}_j - \vec{r}_i + \vec{n}|} \right) \tag{19}$$

$$\vec{E}_{i,\mathrm{rec}} = \frac{1}{\pi V} \sum_{\vec{m}\neq 0} \frac{\partial}{\partial \vec{r}_i} \frac{\exp(-\pi^2 \vec{m}^2/\beta^2)}{\vec{m}^2} S(\vec{m}) \exp(-2i\pi \vec{m}\cdot\vec{r}_i) \tag{20}$$

$$\vec{E}_{i,\mathrm{corr}} = \sum_{j\in M(i)} (s_{ij} - 1) \frac{\partial}{\partial \vec{r}_i} L_j \left(\frac{1}{|\vec{r}_j - \vec{r}_i + \vec{n}|} \right) \tag{21}$$

$$\vec{E}_{i,\mathrm{self}} = \frac{4\beta^3}{3\sqrt{\pi}} \vec{\mu}_{si} \tag{22}$$

Applying the SPME method, one can approximate $\vec{E}_{i,\mathrm{rec}}$ by

$$\vec{E}_{i,\mathrm{rec}} \approx \sum_{\vec{n}} \frac{\partial}{\partial \vec{r}_i} \theta_p(u_{1i} - n_1)\theta_p(u_{2i} - n_2)\theta_p(u_{3i} - n_3)(G^R \star Q^R)(\vec{n}) \tag{23}$$

as explained in section 2

3.1 The Polarization Matrix

By using the Ewald summation, four kind of electric fields are involved in interactions between the induced dipoles: direct, reciprocal, correction and self. These can be written, for $i \neq j$:

$$T_{ij,\mathrm{dir}}\vec{\mu}_j, \quad T_{ij,\mathrm{rec}}\vec{\mu}_j, \quad T_{ij,\mathrm{corr}}\vec{\mu}_j, \quad T_{i,self}\vec{\mu}_i \tag{24}$$

where

$$T_{ij,\mathrm{dir}}^{\alpha\beta} = \sum_{\vec{n}}^{(n)} \frac{\partial}{\partial r_i^\alpha} \frac{\partial}{\partial r_j^\beta} \frac{\mathrm{erfc}(\beta|\vec{r}_j - \vec{r}_i + \vec{n}|)}{|\vec{r}_j - \vec{r}_i + \vec{n}|} \tag{25}$$

$$T_{ij,\mathrm{rec}}^{\alpha\beta} = \frac{2i}{V} \sum_{\vec{m}\neq 0} \frac{2i\pi \exp(-\pi^2\vec{m}^2/\beta^2)}{\vec{m}^2} m^\alpha m^\beta \exp(2i\pi\vec{m}(\vec{r}_j - \vec{r}_i)) \tag{26}$$

$$T_{ij,\mathrm{corr}}^{\alpha\beta} = \sum_{j\in M(i)} (s_{ij} - 1) \frac{\partial}{\partial r_i^\alpha} \frac{\partial}{\partial r_j^\beta} \left(\frac{1}{|\vec{r}_j - \vec{r}_i + \vec{n}|} \right) \tag{27}$$

$$T_{i,self}^{\alpha\beta} = -\frac{4\beta^3}{3\sqrt{\pi}} \delta_{\alpha\beta} \tag{28}$$

Applying the SPME method we have then:

$$T_{ij,rec}^{\alpha\beta} \approx \sum_{\vec{n}} \frac{\partial\theta_p(u_{\alpha i} - n_\alpha)}{\partial r_i^\alpha} \theta_p(u_{\beta i} - n_\beta)\theta_p(u_{\gamma i} - n_\gamma) G^R \star Q_j^{R\beta}(\vec{n}) \tag{29}$$

where $Q_j^{R\beta}$ is the multipolar array associated with the β component of the j-th induced dipole:

$$Q_j^{R\beta}(k_1, k_2, k_3) = \sum_{n_1,n_2,n_3} \frac{\partial}{\partial r_j^\beta} (\theta_p(u_{1j} - k_1 - K_1 n_1)\theta_p(u_{2j} - k_2 - K_2 n_2)\theta_p(u_{3j} - k_3 - K_3 n_3)) \tag{30}$$

The expression of the polarization energy is then:

$$\mathcal{E} = -\sum_{i=1}^N E_i^\alpha \mu_i^\alpha + \frac{1}{2}\sum_{i=1}^N [\alpha_i^{-1}]^{\alpha\beta} \mu_i^\alpha \mu_i^\beta + \frac{1}{2}\sum_{i=1}^N \sum_{j\neq i} (T_{ij,\mathrm{dir}} + T_{ij,\mathrm{rec}} + T_{ij,\mathrm{corr}})^{\alpha\beta} \mu_i^\alpha \mu_j^\beta + \frac{1}{2}\sum_{i=1}^N T_{i,\mathrm{self}}^{\alpha\beta} \mu_i^\alpha \mu_i^\beta \tag{31}$$

Note that the first term represents the interactions between the inducing field and the dipoles, and the others the interactions between dipoles. In (25) we introduced 3×3 tensors in order to compute the electric fields created by one induced dipole and all his images on a site. Such tensors consists in three contributions, when $i \neq j$, and a self contribution has to be included adding a self interaction between dipoles to the polarization energy. In order to simplify the notation, let us introduce the $3N \times 3N$ polarization matrix $\mathbf{T}$, which is composed by the

3×3 T_{ij} tensors as off diagonal blocks, $T_{ij} = T_{ij,\mathrm{dir}} + T_{ij,\mathrm{rec}} + T_{ij,\mathrm{corr}}$, and by the inverse polarizability tensor plus the self contribution tensors as diagonal blocks.

$$\mathbf{T} = \begin{pmatrix} \alpha_1^{-1} + T_{1,\mathrm{self}} & T_{12} & T_{13} & \dots & T_{1N} \\ T_{21} & \alpha_2^{-1} + T_{2,\mathrm{self}} & T_{23} & \dots & T_{2N} \\ T_{31} & T_{32} & \ddots & & \\ \vdots & \vdots & & & \vdots \\ T_{N1} & T_{N2} & & \dots & \alpha_N^{-1} + T_{N,\mathrm{self}} \end{pmatrix}, \tag{32}$$

By introducing the vectors

$$\mathbf{E} = \begin{pmatrix} \vec{E}_1 \\ \vec{E}_2 \\ \vdots \\ \vec{E}_N \end{pmatrix}, \quad \boldsymbol{\mu} = \begin{pmatrix} \vec{\mu}_1 \\ \vec{\mu}_2 \\ \vdots \\ \vec{\mu}_N \end{pmatrix}, \tag{33}$$

we can rewrite the energy functional, when computed with SPME, in a more compact way:

$$\mathcal{E} = \frac{1}{2}\boldsymbol{\mu}^\dagger \mathbf{T} \boldsymbol{\mu} - \mathbf{E}^\dagger \boldsymbol{\mu}. \tag{34}$$

The minimizer $\boldsymbol{\mu}$ of (34) satisfies then the optimality condition:

$$\mathbf{T}\boldsymbol{\mu} = \mathbf{E}. \tag{35}$$

3.2 Forces

The forces associated with the polarization energy are obtained by differentiating it with respect to the position of the atoms.

$$-F_k^\alpha = \frac{d\mathcal{E}}{dr_k^\alpha} = \frac{\partial \mathcal{E}}{\partial r_k^\alpha}, \quad k = 1, \cdots, N.$$

where we have exploited the variational formulation of the polarization energy. More in detail:

$$\begin{aligned}
-F_k^\alpha &= -\boldsymbol{\mu}^\dagger \frac{\partial \mathbf{E}}{\partial r_k^\alpha} + \frac{1}{2}\boldsymbol{\mu}^\dagger \frac{\partial \mathbf{T}}{\partial r_k^\alpha}\boldsymbol{\mu} \\
&= -\sum_{i=1}^{N} \frac{\partial E_i^\beta}{\partial r_k^\alpha}\mu_i^\beta + \frac{1}{2}\sum_{i=1}^{N} \frac{\partial [\alpha_i^{-1}]^{\beta\gamma}}{\partial r_k^\alpha}\mu_i^\beta \mu_i^\gamma + \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i} \frac{\partial \mathbf{T}_{ij}^{\beta\gamma}}{\partial r_k^\alpha}\mu_i^\beta \mu_i^\gamma
\end{aligned} \tag{36}$$

The second term of (36) originates from the fact that the polarizability tensor needs to be defined in some molecular frame and the rotated in the lab frame; similar contributions are also implied in the first term due to the static dipoles and quadrupoles. Such contributions to the forces require to take into account the derivatives of the rotation matrices used to switch from the local frame where the polarizabilities and multipoles are expressed into the global frame. A complete derivation of these terms, which is straightforard, but very cumbersome can be found in ref. 34.

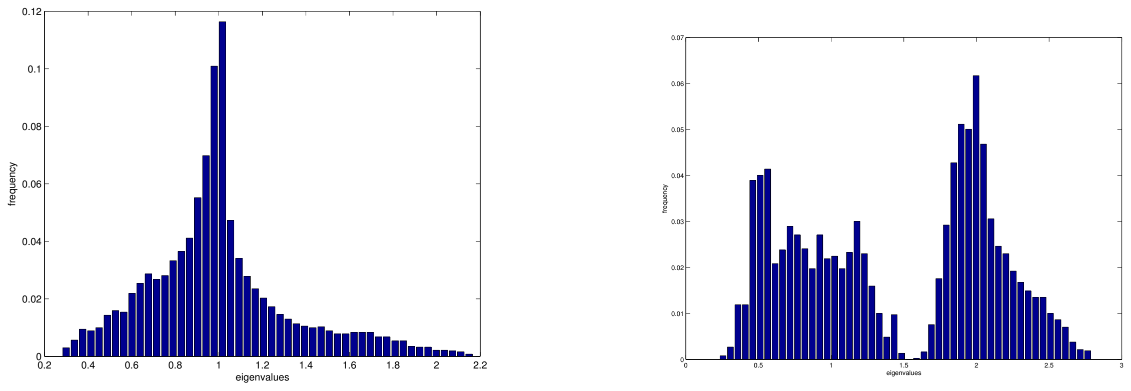
3.3 Iterative Schemes to solve the Polarization Equations

In section 3.1 we introduced the polarization equations (35) which require to invert the $3N \times 3N$ polarization matrix $\mathbf{T}$ to compute the induced dipoles. As N can be very large (systems of size up to several hundred thousand of atoms can be studied in molecular dynamics) it is usually not possible to use exact methods such as LU or Cholesky factorization. Indeed, the computational cost of such methods scales as N^3 (for a $N \times N$ matrix) and they require quadratic storage. A more convenient strategy is to use an iterative method.

In a recent paper[34] we explored various iterative strategies to solve the polarization equations. We suggested that two methods seem to be especially suitable for the computation of the induced dipoles within a parallel implementation: the Preconditioned Conjugate Gradient (PCG) and the Jacobi method coupled with Direct Inversions in the Iterative Subspace (DIIS).

As the spectrum of the Polarization Matrix in Periodic boundary Conditions computed with PME has in general almost the same structure as for the direct space case (see fig.1 for a schematic representation), both methods have the same rate of convergence for our particular problem as what was described in our previous paper. The convergence of such methods is shown in figure 2 in comparison with the self-overrelaxation method (SOR) which has traditionally been used in conjunction with the AMOEBA force field. Similarly to what was observed in direct space computation, both the JI/DIIS and PCG exhibit good convergence properties. Notice that the simple Jacobi iterations are not convergent.

Figure 1: Eigenvalues of the polarization matrix for Ubiquitin in periodic boundary conditions with PME, using the AMOEBABIO09 force field, with (left) and without (right) block diagonal preconditioning.



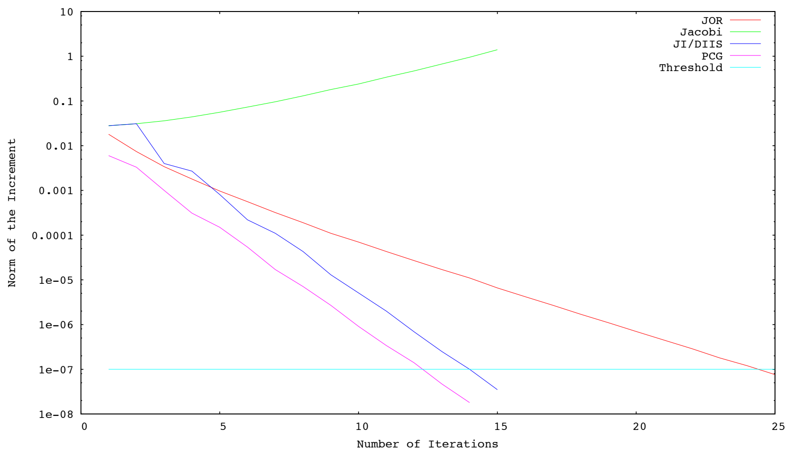
Although the rate of convergence is the same as for the previously studied direct space case, the division of the polarization matrix $\mathbf{T}$ in three matrices (four with the diagonal) with particular properties makes the parallel implementation of the solvers different.

4 Parallel Implementation

Let us recall the steps of both the PCG and JI/DIIS iterative methods in order to comment their parallel implementations.

Starting from an initial guess $\boldsymbol{\mu}_0$, they define a new set of induced dipoles at each iteration. In the case of the JI/DIIS solver, the new induced dipoles are obtained after multiplying the non-diagonal block part of the polarization matrix with their current values and by then

Figure 2: Norm of the increment as a function of the number of iterations for Ubiquitin and different iterative methods.



making a DIIS extrapolation. This extrapolation requires one to assemble the so called DIIS matrix which is done by making scalar products of the past increments with the new one.[52] Let ϵ be the convergence treshold and *maxit* the maximum number of iterations. Let also $\mathbf{O}$ be the off-diagonal part of the polarization matrix $\mathbf{T}$ and $\boldsymbol{\alpha}$ be the $(3N \times 3N)$ block-diagonal matrix with the polarizability tensors of the atoms of the system on its diagonal. Let also $\mathbf{T}_{self}$ be the $(3N \times 3N)$ block-diagonal matrix collecting the self terms, and $\mathbf{D}$ the $(3N \times 3N)$ block-diagonal matrix: $\mathbf{D} = \boldsymbol{\alpha}^{-1} + \mathbf{T}_{self}$ so that $\mathbf{T} = \mathbf{O} + \mathbf{D}$. Then the JI/DIIS can be summarized as follows:

```
**while** $it \leq maxit$ and $inc \geq \epsilon$ **do**
    $\tilde{\boldsymbol{\mu}}_{it+1} = \mathbf{D}^{-1}(\mathbf{E} - \mathbf{O}\boldsymbol{\mu}_{it})$: Jacobi step
    $inc = \|\tilde{\boldsymbol{\mu}}_{it+1} - \tilde{\boldsymbol{\mu}}_{it}\|_{l^2(\mathbb{R}^{3N})}$
    $\boldsymbol{\mu}_{it+1} \leftarrow \tilde{\boldsymbol{\mu}}_{it+1}$: DIIS extrapolation
**end while**
```

The procedure is slightly more complicated for the conjugate gradient method because the new set of induced dipoles are obtained by updating them using a descent direction.

Let $\mathbf{p}_{it}$ be such a descent direction at iteration it, $\mathbf{r}_{it}$ the residual at the same iteration and

$\mathbf{p}_0 = -\mathbf{r}_0$. Then the Conjugate Gradient is described by the following pseudo code (the preconditioned conjugate gradient only differs by the fact that $\mathbf{T}$ is replaced by $\mathbf{D}^{-1}\mathbf{T}$ and $\mathbf{E}$ by $\mathbf{D}^{-1}\mathbf{E}$):

```
while $it \leq maxit$ and $\|\mathbf{r}_{it}\|_{l^2(\mathbb{R}^{3N})} \geq \epsilon$ do
    $\gamma_{it} = \frac{\mathbf{r}_{it}^T \mathbf{r}_{it}}{\mathbf{p}_{it}^T \mathbf{T} \mathbf{p}_{it}}$
    $\boldsymbol{\mu}_{it+1} = \boldsymbol{\mu}_{it} + \gamma_{it}\mathbf{p}_{it}$: new dipoles
    $\mathbf{r}_{it+1} = \mathbf{r}_{it} + \gamma_{it}\mathbf{T}\mathbf{p}_{it}$: new residual
    $\beta_{it+1} = \frac{\mathbf{r}_{it+1}^T \mathbf{r}_{it+1}}{\mathbf{r}_{it}^T \mathbf{r}_{it}}$
    $\mathbf{p}_{it+1} = -\mathbf{r}_{it+1} + \beta_{it+1}\mathbf{p}_{it}$: new descent direction
end while
```

In both cases, matrix-vector products are required at each iteration, but these involve the current values of the dipoles for the JI/DIIS and the descent direction for the Conjugate Gradient.

In terms of parallelism, the parts that require communications between processes are these matrix-vector products (communications of the dipoles or of the descent direction) and the scalar products (reductions).

A comparison between the two solvers reveals that, because of the global reductions needed to compute the DIIS matrix at each iteration, the JI/DIIS might in principle be less suited for parallelization than the PCG. There are two intertwined considerations that need to be done. First, our implementation is based on non-blocking communication, both for the dipoles/descent directions and for the global reductions. This means that the MPI communication calls return immediately even if the processes involved are not complete: communication is performed while the computation is still going. Notice that a "MPI_Wait" or "MPI_Probe" has to be executed after the computation in order to make sure that all the processes received the needed information. The use of non-blocking communication allows us to efficiently cover the communication time, limiting thus the impact of assembling the DIIS matrix on the performances of the JI/DIIS. Second, as communication is done before

the convergence check, the converged dipoles/descent directions are always broadcast. As in MD simulation one needs to compute the forces, which require the converged dipoles to be assembled, one needs a further communication step in the PCG solver, i.e., the dipoles at convergence, whereas this is not necessary for the JI/DIIS, as the converged dipoles are already available. These two considerations combined justify the slightly superior performances of the JI/DIIS solver.
As explained in the introduction, we have improved our last parallel implementation of the solvers so that the only important difference between them is an additional round of communications (of the converged induced dipoles) in the case of the PCG. For this reason, we expect them to have a similar parallel behaviour.

4.1 Direct Part (real space)

In this section, we will recall the main steps of the previously studied parallel implementation of the polarization solvers without periodic boundary conditions and explain the differences for the case of the direct part of the interactions when SPME is used.

When no periodic boundary conditions are imposed, as studied in ref. 34, no cutoff is used so that global communications of the current dipoles (JI/DIIS) or descent direction (PCG) are mandatory and a particle decomposition, where the atoms are distributed among processes without taking into account their position, suitable. These global communications, whose number grows quadratically with the number of processes, are then the obvious bottleneck to the effectiveness of the parallel implementation.

Things are more complicated for SPME computations. First, the use of a cutoff for the short range real space part of the interactions makes the broadcast of all the dipoles at each iteration unnecessary: each process only has to receive at each iteration the value of the induced dipoles that are at a distance inferior to the real space cutoff.
To take advantage of this, we choose to use a spatial decomposition load balancing where each process is assigned to a region of the elementary cell. This means that each process

computes the induced dipoles arising on every atomic site lying in his part of the elementary cell. For now our algorithm only considers 1D spatial decomposition and an iterative procedure has been implemented in order to adapt the size of these domains for non homogeneous systems. More advanced load balancing procedures have been proposed in the literature (ref. 53). The tuning of these techniques for the polarization problem is under active investigation in our team and they will be included in future implementations.

To reduce the total number of communications, a good strategy is to send the induced dipoles by block: a process receives the whole part of the induced dipoles vector treated by another process as long as he needs to receive at least one of its values.

Also, we have observed that using Newton's third law, that is to compute the distance involving a pair of atom sites only once, is an important obstacle to a scalable implementation. Indeed, although it allows to reduce the computations of such distances by half, one has to make a global reduction on the electric fields over all the processes when using this technique and this has a great impact on the parallel efficiency of the algorithm for large systems and/or when a large number of cores is employed. Furthermore, we observed that the loss in efficiency for a few cores is quickly compensated by the gain in parallel efficiency when a larger number of cores is used.

4.2 Reciprocal Space Part

The computation of the reciprocal polarization matrix/induced dipoles matrix vector product can be divided in four steps as explained in section 2.

1. The first one is to fill the multipolar array Q^R: this can be easily distributed among the processes.

2. The second is to compute the FFT of this array. We use the parallel version of the FFTW library[54] to perform this task. Unfortunately, the number of All to All communications that it requires, whose number grows as the square of the number of

processes used, limits its parallel scaling to a relatively low number of cores for the 3D grids normally used for the PME method (up to 128x128x128 for example). The impact of this can be limited by increasing the real space cutoff while decreasing the size of the FFT grid in order to keep the same accuracy for the PME while limiting the proportional time spent doing FFTs.

Furthermore, as a 1D decomposition is used to split the work among the processors, no performance gain can be obtained by using more cores than the length of one of the grid axis.

Note also that the AMOEBA force field[55] requires one to compute two sets of induced dipoles, that correspond to two sets of electric fields created by the permanent multipoles on the atom sites with different scaling parameters.[36] However, the arrays involved in the FFT being all real, it is possible to get all the results by calling only one complex to complex FFT calculation.

As mentioned above, the grid points are distributed among the processes with a 1D decomposition when using the MPI version of the FFTW library. Suppose that K_1, K_2 and K_3 are the sizes of the grid axis on the three dimensions the FFT has to be done on. Then each MPI process treats slabs of data of size $K_1 x K_2 x K_{3loc}$, where the different value of K_{3loc} are as close as possible between processes in order for the computational load to be well distributed between them. We impose for each process to compute the contribution to the grid of the dipoles whose scaled fractional coordinate u_{3i} belongs to the portion of the grid treated by this process. Depending on how the spatial decomposition is made initially, this may require a few communications between neighboring processes.

As the B-splines are non zero only on a small portion (depending on their order) of the grid around the fractional coordinates of the corresponding atoms, only processes treating a neighboring portion of the grid may have contributions to the array representing the dipoles that overlap each other grids domain. In terms of communication, this means that before calling the FFT routine, each process needs to receive the contribution of its neighboring processes to its part of the grid, and send their contribution to the part of the grid of their neighboring processes. These local communications are not a limiting factor in terms of parallel scaling.

3. Then the convolution in Fourier space is easily distributed among the processes and the backward FFT can be directly computed in parallel as each process has already his whole portion of the grid.

4. Finally, for the same reasons as above, processes have to communicate their part of the grid to neighboring processes in order to extract the reciprocal potentials by multiplying the grid values with the appropriate B-spline values.

As explained above, reductions have to be done during the iterative procedure. This can be done without affecting the parallel scaling by using the non blocking collective operations from the MPI3 standard that allow to cover efficiently communications with computations.

4.3 Forces

Things are less complicated for the computation of the forces associated with the polarization energy: no iterative procedure is necessary and one can make use of previous computations in order to limit the number of calls to the FFT routines. Indeed, as the first step of the computation of the induced dipoles is to compute the electric fields due to the permanent multipoles of the system $\mathbf{E}$, the associated multipolar grid $G^R \star Q^R(\vec{n})$ can be stored and reused to get the derivatives of the reciprocal part of this electric fields by just multiplying this grid by the appropriate derivative of the B-splines at the appropriate grid points. This can not be done to get the electric fields created by the induced dipoles as the only associated grid that could be stored would be the one associated with the dipoles one iteration before convergence.

5 Numerical results

Computational Details The parallel implementation tested on the Stampede supercomputer of the TACC whose architecture consists, for the part we did our tests on, in 6400 compute nodes with two intel Xeon E5-2680 CPUs with eight cores at 2.7Ghz and 32 GB of DDR3 DIMM RAM. These nodes communicate within a 56 GB/s InfiniBand network. The following numerical results are all based on computations made on this Supercomputer. To benchmark our algorithms, we considered three water molecules clusters of different sizes: one of 20000 molecules (60000 atoms), one of 32000 molecules (96000 atoms) and one of 96000 molecules (288000 atoms). In all three cases, the convergence treshold was set to 10^{-5} for the dipoles increment; we used B-splines of order 5 and a real space cutoff parameter of 1. The AMOEBA09 force field was used for every computation. The characteristics of these test systems, together with the PME grid sizes used and the size of the elementary cubic cell, are given in table 1. Notice that the PME parameters are chosen in order to increase

Table 1: Caracteristics of the test sytems used to benchmark the parallel implementation and sizes of the FFT grids. Box size is the size of the edge of cubic box, in nanometers

System	Number of atoms	Box size	Grid Size
S1	60000	8.40	72x72x72
S2	96000	9.85	80x80x80
S3	288000	14.30	128x128x128

the computational cost of the (more parallel) direct part of the interactions and decreasing the importance of the reciprocal part, while keeping a good accuracy of the global results.

We tested our implementation of both solvers and of the forces on the Stampede cluster. A first SIMD (Single Instruction Multiple Data) implementation showed that the poor parallel scaling of the 3D FFT routines limited the global parallel scaling of the algorithm: no gain could be obtained by using more than 64-128 cores. To limit the impact of the poor parallel scaling of the FFT routines a popular strategy is to take advantage of the independence of the direct and reciprocal contribution by assigning their treatment to separate group of processes[53],[56] in a MIMD (Multiple Instructions Multiple Data) scheme. The reciprocal space part is faster than the direct space one (by a factor that depends on the parameter β used); however the latter has a better parallel scaling: it is therefore possible to use more processors for the better scaling part, balancing the distribution of cores in such a way that the reciprocal and direct space computations take the same time doing each their part of the matrix/vector products. The MIMD strategy allow us to increase the scalability of our code up to 512 cores.

The scaling of the PCG and JI/DIIS solvers is illustrated in figures 3–5.

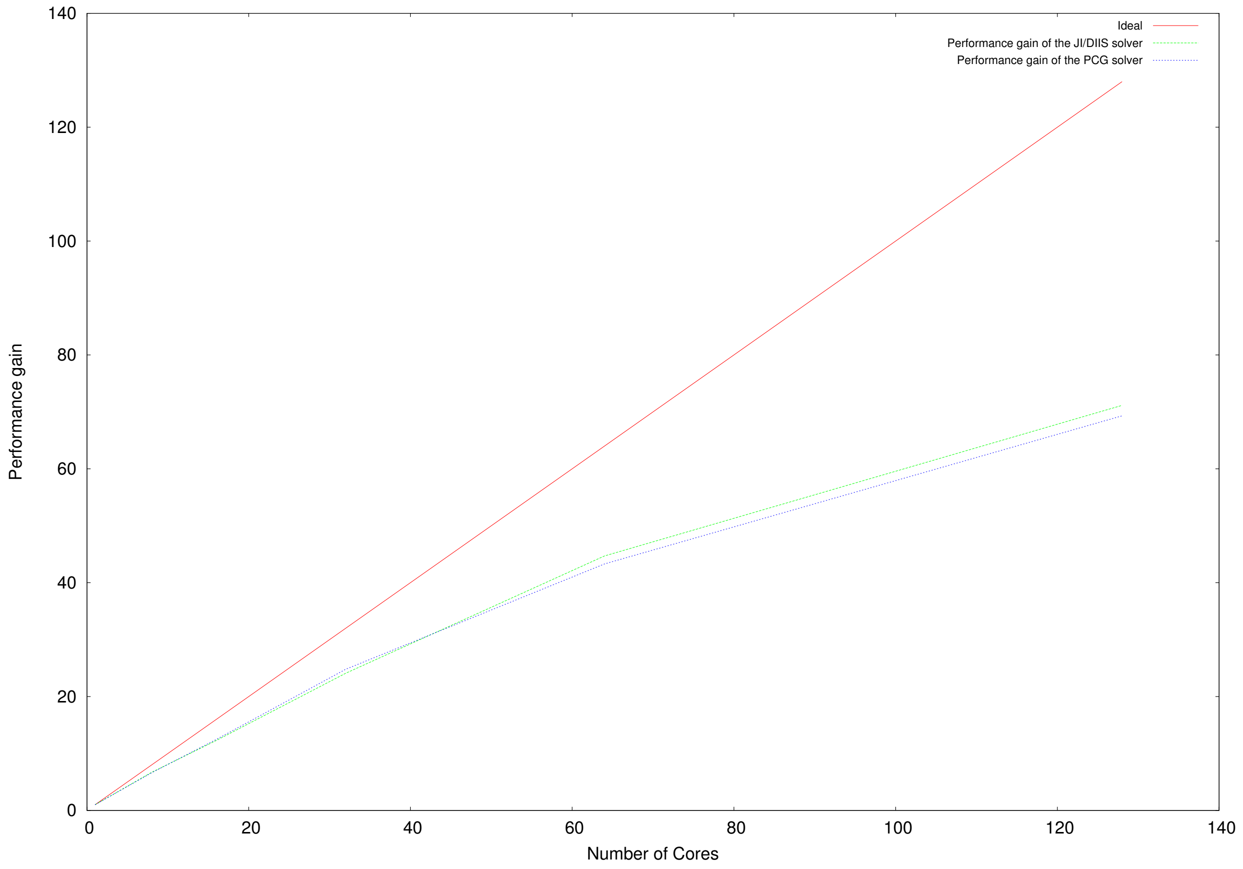


Figure 3: Parallel scaling of the induced dipoles calculation (PCG and JI/DIIS) for the S1 system, using a separate group of processes to compute the reciprocal space contribution

The absolute best timings for the solvers are reported in table2. The best timings obtained with the public version of TINKER are also reported for comparison. The public TINKER is OpenMP parallel and uses a different implementation of the PCG solver; notice

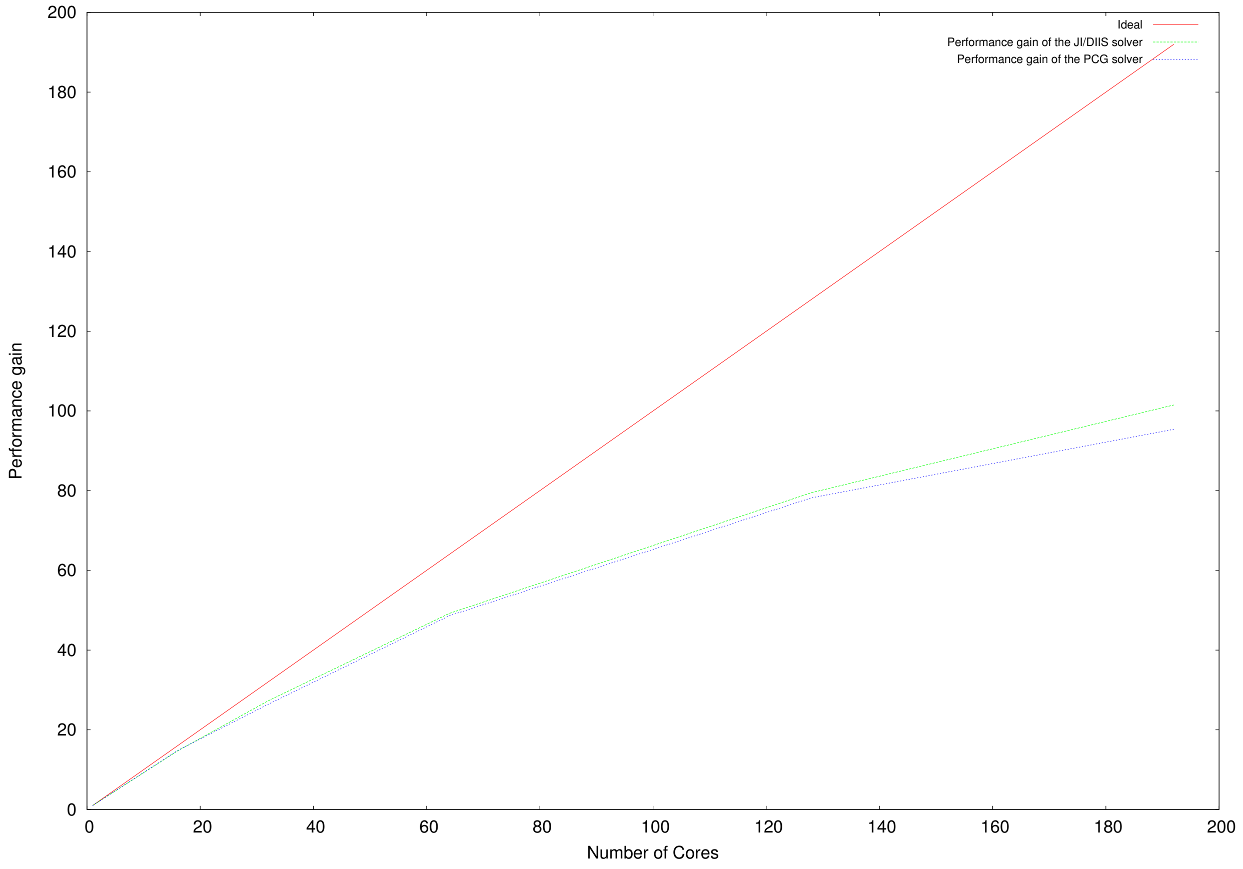


Figure 4: Parallel scaling of the induced dipoles calculation (PCG and JI/DIIS) for the S2 system using a separate group of processes to compute the reciprocal space contribution

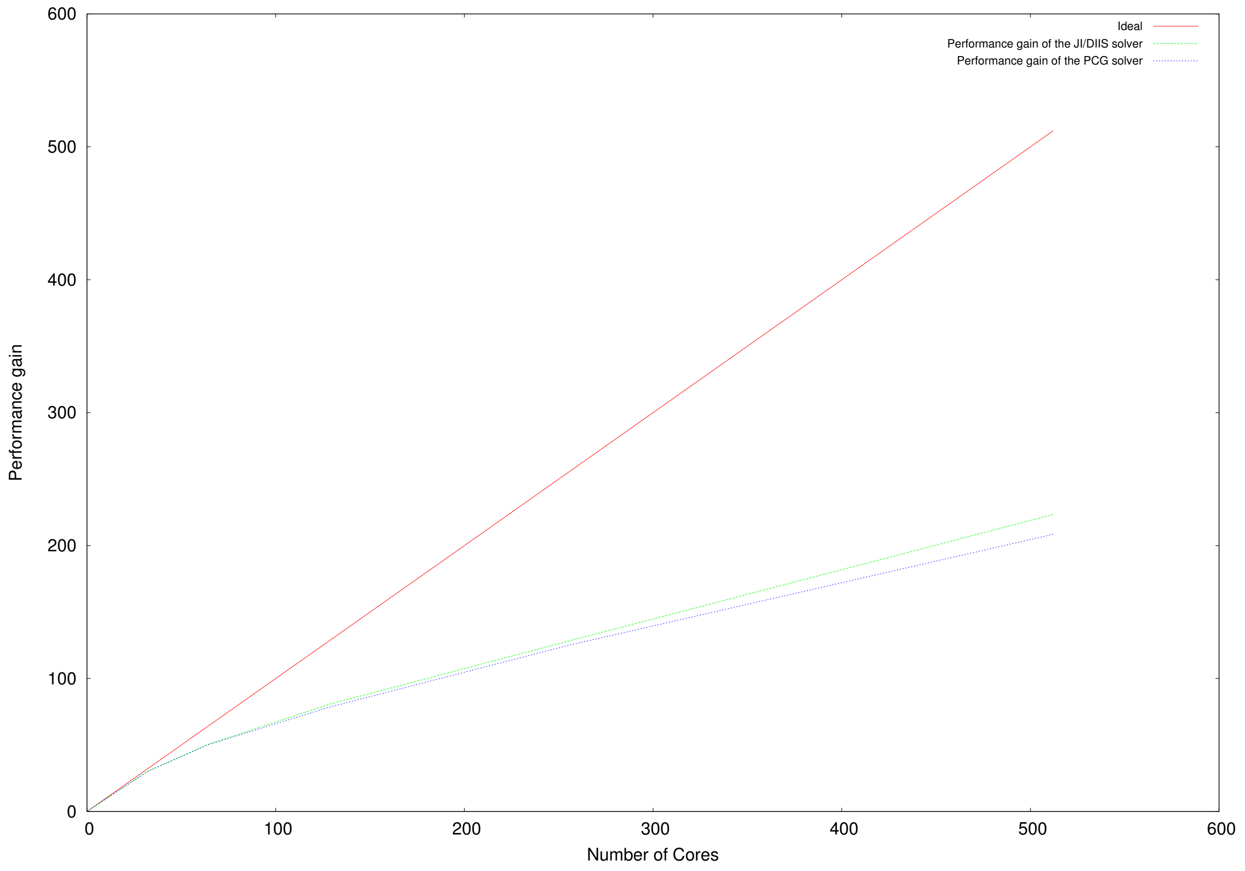


Figure 5: Parallel scaling of the induced dipoles calculation (PCG and JI/DIIS) for the S3 system using a separate group of processes to compute the reciprocal space contribution

that no speedup could be obtained by using more than 4 OpenMP threads with this implementation on the systems described above, running on the TACC supercomputer. Both the

Table 2: Absolute timings (in seconds) and number of cores used (in parentheses) for the PCG and JI/DIIS parallel solvers in our implementation (TINKER HP) and in the public Tinker release (TINKER 6.3).

System	TINKER HP		TINKER 6.3
	PCG	JI/DIIS	
S1	0.45 (128)	0.44 (128)	6.48 (4)
S2	0.50 (192)	0.47 (192)	11.50 (4)
S3	0.92 (512)	0.88 (512)	37.8 0(4)

JI/DIIS and the PCG solvers share a similar parallel behavior; the JI/DIIS solver exhibits a small edge in terms of parallel efficiency, this being due, as explained in section 4, to the additional communications happening after convergence in that case in order to compute the forces. Nevertheless, the absolute best timings of the two solvers are comparable.
The parallel computation of the forces associated with the polarization energy also naturally benefits from the separate calculation of the direct and reciprocal contribution. The scaling of our implementation is reported in figure 6. The absolute best timings are also reported in table 3, where we also report the best timings obtained with the public version of TINKER. The number of cores that could be used with the public, OpenMP implementation was limited by memory requirements to respectively 4-8 threads for the system studied here.

Table 3: Absolute best timings (in seconds) and number of cores used (in parentheses) for the computation of the forces associated to the AMOEBApolarization energy in our (TINKER HP) and in the public (TINKER 6.3) implementation

System	TINKER HP	TINKER 6.3
S1	0.07 (256)	1.90 (8)
S2	0.10 (256)	3.78 (6)
S3	0.20 (512)	19.0 (4)

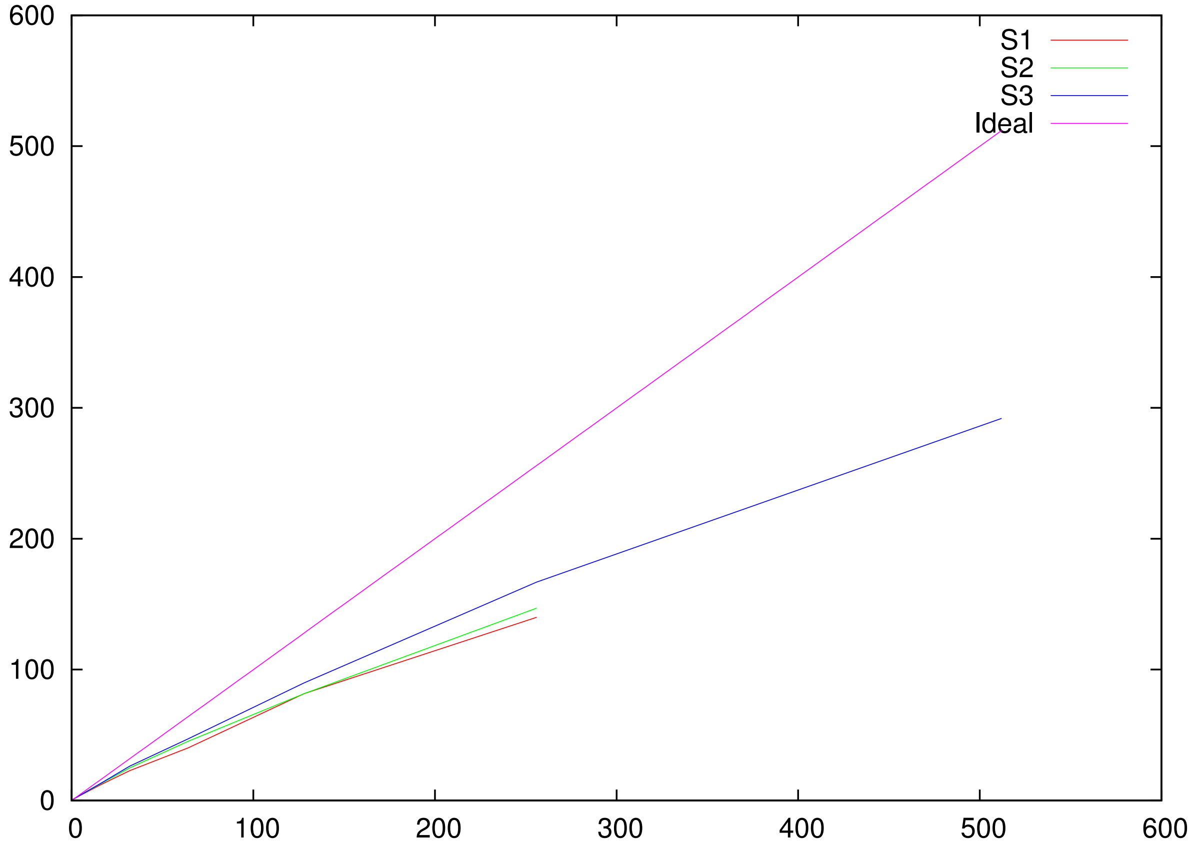


Figure 6: Parallel scaling of the computation of the forces associated to the polarization energy for the S1, S2 and S3 systems using a separate group of processes to compute the reciprocal space contribution. Notice that negligible performance gains were observed for S1 and S2 when increasing the number of threads beyond 256.

5.1 Further performance improvements

All the loops involved in the solvers and in the computation of the forces can be parallelized with OpenMP within each MPI process in order to limit the number of necessary communications for a given number of cores and to take advantage of the shared memory architecture within each node of the supercomputer. Furthermore, the FFTW library also enables to use such a computational paradigm. We found experimentally that the best results using this hybrid OpenMP/MPI paradigm were obtained by using 2 OpenMP threads per MPI process, allowing to improve the overall best timing of the solvers and of the computation of the forces. Notice that this hybrid paradigm can be used to extend the range of scalability of the MPI implementation; however, whenever the MPI implementation has not reached its scaling plateau, the pure MPI code is always more efficient than the hybrid one. Nevertheless, the Hybrid implementation allows one to take advantage of a larger number of processors, when available.

The best overall timings obtained with the hybrid implementation are given in table 4 for both solvers and for the forces. The total energy and forces best timings are also reported. The numbers in the last column of table 3 give a measure of the overall computational

Table 4: Best Hybrid MPI/OpenMP timings (in seconds) and maximum number of CPU cores for the solvers and for the polarization forces. The overall dipoles plus forces best timings is also reported.

System	Cores	Solver		Forces	Total
		PCG	JI/DIIS		
S1	256	0.32	0.30	0.04	0.34
S2	512	0.35	0.35	0.08	0.43
S3	1024	0.61	0.60	0.10	0.70

overhead introduced by the use of polarizable force field in a PBC MD simulation: even for a system as large as S3 (almost 300000 atoms) such an overhead can be reduced to less than a second of wall time by using our JI/DIIS solver together with our new Parallel implementation. However, such numbers were computed by using a guess for the dipoles that doesn't take advantage of previous information, for instance, the value of the dipoles at previous steps of the simulation. As discussed in a recent publication,[34] the results can be further improved by using as a guess the predictor step of Kolafa's always stable predictor corrector:[57] we report the results obtained by using such a guess, which represents our default choice for MD simulations, in table 5.

Table 5: Best Hybrid MPI/OpenMP timings (in seconds) and maximum number of CPU cores for the solvers and for the polarization forces by using the predictor step of Kolafa's always stable predictor corrector integrator as a guess. The overall dipoles plus forces best timings is also reported.

System	Cores	Solver		Forces	Total
		PCG	JI/DIIS		
S1	256	0.16	0.15	0.04	0.19
S2	414	0.18	0.18	0.08	0.26
S3	1024	0.32	0.30	0.10	0.40

In conclusion, our new implementation allows to treat systems as large as S3 introducing an overhead as little as 0.7s on 1024 cores, which is reduced during a simulation to 0.4s per time step, allowing polarizable MD simulations in PBC to be performed on large and very large systems.

6 Conclusion

In this paper, we presented a new, improved implementation of the polarization energy and forces for the AMOEBA force field which is suited to perform polarizable MD simulations using periodic boundary conditions on parallel computers. We stated in the introduction that three ingredients are needed in order to achieve an efficient implementation of a polarizable force field: a fast convergent iterative solver, a fast matrix-vector multiplication technique and an efficient parallel setup. The three points have been addressed in this paper and the resulting implementation tested on large to very large systems. Let us recapitulate here the main results of this work.

The first point, which had already been addressed by a dedicated publication,[34] has been here extended to the context of PBC simulations. The two solvers that we proposed for the polarization equations, i.e., the PCG and JI/DIIS solvers, have been reimplemented and adapted.

The second point is central to this paper and has been addressed, for PBC simulations, by using the smooth Particle-Mesh Ewald method, which is used to compute the far-field portion of the $\mathbf{T}\boldsymbol{\mu}$ matrix-vector product. In particular, such a matrix-vector product is rewritten as a convolution, which is performed with negligible cost in the reciprocal space: the use of the FFT to transform the involved quantities is what avoids the $\mathcal{O}(N^2)$ computational bottleneck. A new implementation of the SPME for distributed multipoles based, polarizable force fields is hence one of the main contents of this paper.

Finally, the paper addresses the third point with an extended discussion of the various technical issues that affect the specific polarization problems. The two iterative solvers have been analyzed in a real-life context, i.e., the one of MD simulations, when the forces have to be computed after the linear equations have been solved. Both solvers are well suited for a parallel implementation; however, the JI/DIIS one seems slightly superior to the PCG for very scalable implementations. Finally, the use of a multiple instructions, multiple data has been analyzed in order to compensate the non-optimal scaling of the FFT by using only a portion of the available CPUs to do the various reciprocal space computations, while employing the other processors for the more scalable direct space ones. The scaling of our implementation is overall quite good up to as much as 512-1024 cores: we would like to point out that a parallel implementation of polarizable force fields is much more challenging than for additive force fields, as solving the polarization linear systems implies a large number of synchronization barriers, one per iteration, in particular, the effect of which can be only partially mitigated by covering communication with computation.

Overall, we showed how our new implementation is well suited to treat large and very large systems, extending thus the range of applicability of dipole-based polarizable force fields such as AMOEBA to very large systems: we remark that, by using 1024 cores, as little as 0.4 seconds per time step are needed to handle the polarization energy and forces for a system composed of as much as 288000 atoms, when Kolafa's predictor step is used to provide a guess for the iterative solver.

There are still various open challenges and improvements that need to be addressed for polarizable MD simulations. The use of more advanced load-balancing techniques, in the spirit of what was done for example in GROMACS,[53,58] certainly deserves more future investigation. The extension of the presented machinery to other, advanced force fields, such as SIBFA[8] and GEM[42] is another interesting development which is being actively investigated in our lab. On a different side, we have recently proposed a complementary approach, that is, the use of polarizable continuum solvation methods instead than PBC to provide a boundary: a comparison between the two approaches, especially for difficult systems such as highly charged metal ions or biological molecules, as well as a comparison of the performances of the two approaches, is in progress. The presented methods will also be included in the FFX package.[59] We will present in an upcoming paper an improved global scaling of the AMOEBA force field within the massively parallel Tinker-HP pacage.

7 Acknowledgements

We wish to thanks Marie-France Couret for technical support. This work was supported in part by French state funds managed by CalSimLab and the ANR within the Investissements d'Avenir program under reference ANR-11-IDEX-0004-02. Pengyu Ren is grateful for support by the Robert A. Welch Foundation (F-1691) and the National Institutes of Health (GM106137), as well as the high performance computing resources provided by TACC and XSEDE (TG-MCB100057). Funding from French CNRS through a PICS grant between UPMC and UT Austin is acknowldeged. Filippo Lipparini gratefully acknowledges the Alexander von Humboldt foundation for funding.

References

(1) Jorgensen, W. L. *J. Chem. Theory Comput.* **2007**, *3*, 1877–1877, and references therein.

(2) Warshel, A.; Kato, M.; Pisliakov, A. V. *J. Chem. Theory Comput.* **2007**, *3*, 2034–2045.

(3) Cieplak, P.; Dupradeau, F.-Y.; Duan, Y.; Wang, J. *J. Phys. Condens. Mat.* **2009**, *21*, 333102.

(4) Lopes, P. E.; Roux, B.; MacKerell, J., Alexander D. *Theor. Chem. Acc.* **2009**, *124*, 11–28.

(5) Piquemal, J.-P.; Jordan, K. D. *Theor. Chem. Acc.* **2012**, *131*, 1–2.

(6) Ji, C.; Mei, Y. *Acc. Chem. Res.* **2014**, *47*, 2795–2803.

(7) Cisneros, G. A.; Karttunen, M.; Ren, P.; Sagui, C. *Chem. Rev.* **2014**, *114*, 779–814.

(8) Gresh, N.; Cisneros, G. A.; Darden, T. A.; Piquemal, J.-P. *J. Chem. Theory Comput.* **2007**, *3*, 1960–1986.

(9) Freddolino, P. L.; Harrison, C. B.; Liu, Y.; Schulten, K. *Nature Physics* **2010**, *6*, 751–758.

(10) Tong, Y.; Mei, Y.; Li, Y. L.; Ji, C. G.; Zhang, J. Z. H. *J. Am. Chem. Soc.* **2010**, *132*, 5137–5142.

(11) Luo, Y.; Jiang, W.; Yu, H.; MacKerell, A. D.; Roux, B. *Faraday Discuss.* **2013**, *160*, 135–149.

(12) Huang, J.; Lopes, P. E. M.; Roux, B.; MacKerell, A. D. *J. Phys. Chem. Lett.* **2014**, *5*, 3144–3150.

(13) Nielsen, C. B.; Christiansen, O.; Mikkelsen, K. V.; Kongsted, J. *J. Chem. Phys.* **2007**, *126*, 154112.

(14) Kongsted, J.; Osted, A.; Mikkelsen, K. V.; Christiansen, O. *J. Chem. Phys.* **2003**, *118*, 1620–1633.

(15) Steindal, A. H.; Olsen, J. M. H.; Ruud, K.; Frediani, L.; Kongsted, J. *Phys. Chem. Chem. Phys.* **2012**, *14*, 5440–5451.

(16) Marini, A.; Muñoz-Losa, A.; Biancardi, A.; Mennucci, B. *J. Phys. Chem. B* **2010**, *114*, 17128–17135.

(17) Jacobson, L. D.; Herbert, J. M. *J. Chem. Phys.* **2010**, *133*, 154506.

(18) Arora, P.; Slipchenko, L. V.; Webb, S. P.; DeFusco, A.; Gordon, M. S. *J. Phys. Chem. A* **2010**, *114*, 6742–6750.

(19) Caprasecca, S.; Curutchet, C.; Mennucci, B. *J. Chem. Theory Comput.* **2012**, *8*, 4462–4473.

(20) Lipparini, F.; Barone, V. *J. Chem. Theory Comput.* **2011**, *7*, 3711–3724.

(21) Lipparini, F.; Cappelli, C.; Barone, V. *J. Chem. Theory Comput.* **2012**, *8*, 4153–4165.

(22) Lipparini, F.; Cappelli, C.; Barone, V. *J. Chem. Phys.* **2013**, *138*, 234108.

(23) Boulanger, E.; Thiel, W. *J. Chem. Theory Comput.* **2012**, *8*, 4527–4538.

(24) Rick, S. W.; Stuart, S. J.; Berne, B. J. *J. Chem. Phys.* **1994**, *101*, 6141–6156.

(25) Verstraelen, T.; Speybroeck, V. V.; Waroquier, M. *J. Chem. Phys.* **2009**, *131*, 044127.

(26) Piquemal, J.-P.; Chelli, R.; Procacci, P.; Gresh, N. *J. Phys. Chem. A* **2007**, *111*, 8170–8176.

(27) Lamoureux, G.; Roux, B. *J. Chem. Phys.* **2003**, *119*, 3025–3039.

(28) Mills, M. J.; Popelier, P. L. *Comp. Theor. Chem.* **2011**, *975*, 42–51.

(29) Applequist, J.; Carl, J. R.; Fung, K.-K. *J. Am. Chem. Soc.* **1972**, *94*, 2952–2960.

(30) Thole, B. *Chem. Phys.* **1981**, *59*, 341–350.

(31) Wang, J.; Cieplak, P.; Li, J.; Wang, J.; Cai, Q.; Hsieh, M.; Lei, H.; Luo, R.; Duan, Y. *J. Phys. Chem. B* **2011**, *115*, 3100–3111.

(32) Ponder, J. W.; Wu, C.; Ren, P.; Pande, V. S.; Chodera, J. D.; Schnieders, M. J.; Haque, I.; Mobley, D. L.; Lambrecht, D. S.; DiStasio, R. A.; Head-Gordon, M.; Clark, G. N. I.; Johnson, M. E.; Head-Gordon, T. *J. Phys. Chem. B* **2010**, *114*, 2549–2564.

(33) Gordon, M. S.; Slipchenko, L.; Li, H.; Jensen, J. H. *Annu. Rep. Comput. Chem.* **2007**, *3*, 177–193.

(34) Lipparini, F.; Lagardère, L.; Stamm, B.; Cancès, E.; Schnieders, M.; Ren, P.; Maday, Y.; Piquemal, J.-P. *J. Chem. Theory Comput.* **2014**, *10*, 1638–1651.

(35) Wang, W.; Skeel, R. D. *J. Chem. Phys.* **2005**, *123*, 164107.

(36) Lipparini, F.; Lagardère, L.; Raynaud, C.; Stamm, B.; Cancès, E.; Mennucci, B.; Schnieders, M.; Ren, P.; ; Maday, Y.; Piquemal, J.-P. *J. Chem. Theory Comput.* **2015**, *11*, 623–634.

(37) Caprasecca, S.; Jurinovich, S.; Lagardère, L.; Stamm, B.; Lipparini, F. *J. Chem. Theory Comput.* **2015**, *11*, 694–704.

(38) Ewald, P. P. *Ann. Phys.* **1921**, *369*, 253.

(39) Darden, T.; York, D.; Pedersen, L. *J. Chem. Phys.* **1993**, *98*, 10089–10092.

(40) Essmann, U.; Perera, L.; Berkowitz, M. L.; Darden, T.; Lee, H.; Pedersen, L. G. *J. Chem. Phys* **1995**, *103*, 8577–8593.

(41) Sagui, C.; Pedersen, L. G.; Darden, T. A. *J. Chem. Phys.* **2004**, *120*, 73–87.

(42) Cisneros, G. A.; Piquemal, J.-P.; Darden, T. A. *J. Chem. Phys.* **2006**, *125*, 184101.

(43) Lipparini, F.; Lagardère, L.; Stamm, B.; Cancès, E.; Schnieders, M.; Ren, P.; Maday, Y.; Piquemal, J.-P. *J. Chem. Theory Comput.* **2014**, *10*, 1638–1651.

(44) Smith, E. R. *Proc. R. Soc. A* **1981**, *375*, 475–505.

(45) Aguado, A.; Madden, P. A. *J. Chem. Phys.* **2003**, *119*, 7471–7483.

(46) Toukmaji, A.; Sagui, C.; Board, J.; Darden, T. *J. Chem. Phys.* **2000**, *113*, 10913–10927.

(47) Nymand, T. M.; Linse, P. *J. Chem. Phys.* **2000**, *112*, 6152–6160.

(48) Polack, E.; Lagardère, L.; Stamm, B.; Darden, T. A.; Maday, Y.; Piquemal, J.-P. *To be published.*

(49) Sala, J.; Guàrdia, E.; Masia, M. *J. Chem. Phys.* **2010**, *133*, 234101.

(50) Sagui, C.; Pedersen, L. G.; Darden, T. A. *J. Chem. Phys.* **2004**, *120*, 73–87.

(51) Smith, W. *CCP5 Newsletter* **1998**, *46*, 18–30.

(52) Rohwedder, T.; Schneider, R. *J. Math. Chem.* **2011**, *49*, 1889–1914.

(53) Hess, B.; Kutzner, C.; Van Der Spoel, D.; Lindahl, E. *J. Chem. Theory Comput.* **2008**, *4*, 435–447.

(54) Frigo, M.; Johnson, S. G. FFTW: An adaptive software architecture for the FFT. Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on. 1998; pp 1381–1384.

(55) Ponder, J. W.; Wu, C.; Ren, P.; Pande, V. S.; Chodera, J. D.; Schnieders, M. J.; Haque, I.; Mobley, D. L.; Lambrecht, D. S.; DiStasio, R. A.; Head-Gordon, M.; Clark, G. N. I.; Johnson, M. E.; Head-Gordon, T. *J. Phys. Chem. B* **2010**, *114*, 2549–2564.

(56) Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kale, L.; Schulten, K. *J. Comput. Chem.* **2005**, *26*, 1781–1802.

(57) Kolafa, J. *J. Comput. Chem.* **2004**, *25*, 335–342.

(58) Pronk, S.; Páll, S.; Schulz, R.; Larsson, P.; Bjelkmar, P.; Apostolov, R.; Shirts, M. R.; Smith, J. C.; Kasson, P. M.; van der Spoel, D.; Hess, B.; Lindahl, E. *Bioinformatics* **2013**, *29*, 845–854.

(59) Schnieders, M. J.; Fenn, T. D.; Pande, V. S. *J. Chem. Theory Comput.* **2011**, *7*, 1141–1156.