Development of High Performance Molecular Dynamics : Fast Evaluation of Polarization Forces

Félix Aviat

Sorbonne Université

École Doctorale de Chimie Physique et de Chimie Analytique de Paris Centre

Laboratoire de Chimie Théorique - UMR 7616

Development of High Performance Molecular Dynamics: Fast Evaluation of Polarization Forces

Thèse de doctorat de Chimie

par Félix AVIAT

sous la direction de Jean-Philip PIQUEMAL & Alessandra CARBONE

Composition du jury

Rapporteur :	Elise Dumont (Pr.)
Rapporteur :	Michel Caffarel (DR CNRS)
Directrice de thèse :	Alessandra Carbone (Pr.)
Directeur de thèse :	Jean-Philip Piquemal (Pr.)
Examinateur :	Matthieu Montes (Pr.)
Examinateur :	Rodolphe Vuilleumier (Pr.)
Examinateur invité :	Louis Lagardère (Dr.)

Introduction

Understanding the microscopic world, whether it's at the atomic or the protein scale, is a challenge for it is, in most cases, simply impossible to observe. Even if we had a magical microscope, allowing us to magnify at will an image, very small particles would still remain puzzling. They do not follow the uses of our macroscopic experience, ruled by our habitual Newtonian dynamics, and are governed by quantum mechanics, which can be very counter-intuitive.

All hope is however not lost, thanks the computer simulation tool. *In silico* experiments provide us with this extraordinary lense, allowing one to observe as closely as wanted the wobbling of atoms, the dance of molecules, the complex breathing of proteins. But there is a much greater power yet for the user to experiment: to play God within the tiny box that is being simulated. One can, as a demiurge, decide to turn off specific forces, to see exactly what their influence on the system is. This absolute freedom extends widely in this *in silico* world: parameters, such as temperature, pressure, intensity of interactions can be changed at will; entities, such as atoms, molecules, proteins can be modified, added, extracted; the properties of every single atom, from position and velocity to charges or polarizability can be controlled; all that happens can be fully decided by the maker of the model.

The remaining difficulty is to make sure that what is being simulated corresponds to reality, and this imperative drives the development, refinement, study of all models used for these numerical experiments.

In silico simulations of atomic behaviour split in two main families, the quantum and the classical models. They share a porous frontier, as tuning the classical models is usually done using results from *ab initio* quantum computations; QM/MM calculation, distinguishing a quantumly-treated system and its classical environment, show their possible collaboration.

Quantum models are more precise, since quantum physics properly describe the behaviour of mi-

croscopic particles. They explicitly takes into account electrons, using wavefunctions methods (HF, post-HF...), or electronic density (DFT), but come with a computational cost that can be prohibitive (the full CI method scales as $O(N_{e^-}!)$, with N_{e^-} the number of electrons). Typical system sizes don't exceed one or two hundred of atoms in DFT, even less when using wavefunction methods.

The second framework does not explicitly represents electrons. Instead, atoms are represented as a punctual mass, possibly carrying a charge, and all interactions between them are modeled through classical energy terms. Less universal, this approach requires the fitting of parameters (bond strengths, angular and torsion constants, *etc.*) to reproduce at best quantum calculation and experimental data. The use of classical (Newtonian) mechanics nonetheless greatly simplifies the simulations, allowing for much bigger systems that can count millions of atoms.

Looking for a compromise between computation complexity and accuracy of the model, polarizable force field were introduced, adding a term taking into account the polarizability of the electronic density around the atoms. While more expensive to use than a so-called "classical" force field, polarizable force field represent an important step in the modelization for they allow a more accurate description of various systems, from the biological domain to ionic liquids.

Progressing side to side with the physical models and parametrizations, the algorithms used for the simulations are also in constant evolution. They play a key role in theoretical chemistry, allowing the more and more involved physical models to be tested, on systems of increasing size. From the boundary conditions treatment to the management of parallel architectures, every aspect of the numerical simulation has to be well built to keep pushing the limits of *in silico* experiment.

Although considerable progresses have been made since the very first, two-dimensional simulations, some frontiers are still to be broken, both time-scale and size-wise. For example, simulating a whole cell using explicit atoms is out of reach. Computation of binding free-energies, especially relevant in the pharmaceutical domain, that would be both fast and reliable, is still one of the biggest challenges in computational biochemistry. All various models developed along the years, even when it comes to describing water molecules, suffer from their respective limitations. Besides, the study of problems of increasing complexity (drug testing, protein interactions) requires bigger and bigger supercomputers, working always longer, and effectively consuming substantial quantities of electrical power.

We thus need to do better: faster simulations, able to go on for longer periods, designed for large systems. This objective can only be achieved relying on the two pillars presented above: intelligent physical models and well designed algorithms, working in synergy.

The aim for this thesis was therefore to develop strategies to improve the performances of polarizable molecular dynamics. In the first chapter, the reader will find a brief introduction of the framework of this thesis, starting from the general scope of molecular dynamics, and presenting the most standard integrators. Classical force fields follow, then particular attention is given to polarizable force fields, as they are heart of our work. This chapter closes on a more practical aspect, the Tinker-HP code, in which all testing and implementations were carried out.

The second chapter focuses on the polarization issue, and begins with an overview of the current polarization solvers, from their mathematical causes to the algorithms they use. We then present the Truncated Conjugate Gradient, a new algorithm designed to improve polarizable solvers, both in terms of stability and speed. Implementation strategies and numerical results are then given.

In the third chapter, we discuss free energies, through a small presentation of the thermodynamic quantity itself, and of the methods used to compute it. We then discuss the performance of the Truncated Conjugate Gradient applied on free energies calculations.

Finally, in the last chapter, we look for another method to accelerate simulations by focusing on molecular dynamics integrators. After establishing a brief framework, we proceed to search for an optimal integration strategy, aiming at using very large time-steps having a minimal cost on accuracy.

Contents

Chapter 1

Molecular Dynamics – An overview

Contents

Historically, the very first numerical experiments were carried out by Fermi, Pasta, Ulam and Tsingou in 1955[1] on a purely theoretical system to study energy repartition in a chain of oscillators. The first condensed-phase molecular dynamics calculation was undertaken by Adler and Wainwright in 1957.[2] The authors studied equilibrium properties for a system consisting in hard spheres, computing its equation of state. The force field here was only square well potentials of attraction between particles.

Thankfully, Molecular Dynamics have considerably developed since then, as this first chapter will illustrate. Here, the reader will find a – succinct – presentation of the conceptual and material framework

on which the improvements described in next chapters are based. Should they be needed, references to more specialized work are given to allow for a deeper study.

The Molecular Dynamics technique will be explained, supplemented with a presentation of its most usual integrators. Force fields will then be introduced, both the classical and polarizable case. Finally, Tinker-HP, the code in which all developments are implemented, is described.

1.1 Introducing molecular dynamics

Let us start by defining the system we want to study and simulate, $\mathcal{S}$, contained in a box of arbitrary size and shape. We will suppose that $\mathcal{S}$ contains N atoms. Given an arbitrary index i between 1 and N, the three-dimensional vector representing the position of atoms i in Cartesian coordinates will be noted $\vec{r}_i$. Equivalent notations will be adopted for the velocity $\vec{v}_i$, the momentum $\vec{p}_i$, and the acceleration $\vec{a}_i$. By writing m_i the mass of atom i, we have $\vec{p}_i = m_i \vec{v}_i$. For the sake of clarity and simplicity, we will also write

$$\begin{pmatrix} \vec{r}_1 \\ \vec{r}_2 \\ \vdots \\ \vec{r}_N \end{pmatrix} = \mathbf{r}\,, \quad \begin{pmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vdots \\ \vec{v}_N \end{pmatrix} = \mathbf{v}\,, \quad \begin{pmatrix} \vec{p}_1 \\ \vec{p}_2 \\ \vdots \\ \vec{p}_N \end{pmatrix} = \mathbf{p}\,, \quad \begin{pmatrix} \vec{a}_1 \\ \vec{a}_2 \\ \vdots \\ \vec{a}_N \end{pmatrix} = \mathbf{a} \tag{1.1}$$

The masses m_i may also be conveniently gathered as a diagonal matrix $\mathbf{M}$ such that

$$\mathbf{M} = \begin{pmatrix} m_1 & & (0) \\ & m_2 & \\ & & \ddots \\ (0) & & m_N \end{pmatrix} \quad \text{and} \quad \mathbf{p} = \mathbf{M}\mathbf{v} \tag{1.2}$$

There are many objectives behind molecular simulations in theoretical chemistry. It can be simply aimed at observing the movement of atoms or molecules, in order to understand chemical or even biological phenomena (molecular arrangement, protein folding...). Questions more relevant to the statistical mechanics field, like computing properties such as free energies, can also be addressed using molecular dynamics. A wide variety of systems can be studied, both spanning a wide range of sizes (from tens to hundreds of thousands of atoms) and types (monoatomic gases, water phases, solvated proteins, ionic liquids...).

Statistical mechanics provide a secure framework, with strong mathematical and physical foundations, that will ensure our future numerical experiments and analysis are meaningful.

The focus of the study will then define the *statistical ensemble* in which simulations should be done. Each statistical ensemble is defined by a set of properties of the system that should remain fixed throughout the simulation.

For example, let us imagine an isolated system, with no environment. Its number of particles then remains fixed, and no energy can be be exchanged either. If we then suppose that the simulation box remains constant, the corresponding ensemble is the *microcanonical* ensemble, where N, V (the volume of the simulation box) and E (the system's total energy) are fixed.

If, on the other hand, we want to look at a system in contact with a thermostat (modelling, for example, the reaction medium, the surrounding cell...), where the temperature remains constant, but the energy can fluctuate, the proper statistical ensemble is the *canonical* one, where N, V and T are fixed.

By allowing the simulation box to evolve with time, it is also possible to work within the *isobaric* ensemble, with constant number of particles, temperature and pressure (N, P, T).

In a given ensemble, the average value of a quantity b, noted $\langle b \rangle$, is defined as follows:

$$\langle b \rangle = \frac{\int \mathrm{d}\mathbf{r}\mathrm{d}\mathbf{p} b(\mathbf{p}, \mathbf{r}) \rho(\mathbf{r}, \mathbf{p})}{\int \mathrm{d}\mathbf{r}\mathrm{d}\mathbf{p} \rho(\mathbf{r}, \mathbf{p})} \tag{1.3}$$

where ρ is the density of states in phase space. The simple integral symbol here is used to avoid too heavy notations, and designates an integration over each component of position and momentum for

each atom:

$$\int \mathrm{d}\mathbf{r}\,\mathrm{d}\mathbf{p} \equiv \int \mathrm{d}^3\vec{r}_1 \ldots \int \mathrm{d}^3\vec{r}_N \int \mathrm{d}^3\vec{v}_1 \;\ldots \int \mathrm{d}^3\vec{v}_N \tag{1.4}$$

One usually defines the widely used partition function $\mathcal{Z} = \int d\mathbf{r}d\mathbf{p}\rho(\mathbf{r},\mathbf{p})$ to rewrite ensemble averages as

$$\langle b \rangle = \frac{1}{\mathcal{Z}} \int d\mathbf{r}d\mathbf{p}b(\mathbf{p},\mathbf{r})\rho(\mathbf{r},\mathbf{p}) \tag{1.5}$$

Any thermodynamical property that one would like to extract from a system can be computed using these integrals.

So, if one were to study toy systems, such as a one-dimensional harmonic oscillator, analytic solutions to describe thermodynamic properties could simply be derived from these integrals. But for our real systems, much more complex, that option disappears: when studying a solvated protein, an integral over the phase space of thousands of atoms seems quite out of range if we wanted to compute it analytically (or would cost dramatic approximations, abandoning a lot of information on the system). A different way to evaluate this kind of integral is thus necessary.

If we suppose that for an infinitely long simulation, all the accessible phase space would be explored following the right probabilities (with high energy conformations being less visited than the low energy ones), then the system is considered to be *ergodic*, ensuring that

$$\langle b \rangle = \frac{\int \mathrm{d}\mathbf{r}\mathrm{d}\mathbf{p}b(\mathbf{p},\mathbf{r})\rho(\mathbf{r},\mathbf{p})}{\int \mathrm{d}\mathbf{r}\mathrm{d}\mathbf{p}\rho(\mathbf{r},\mathbf{p})} = \lim_{T\to\infty} \frac{1}{T} \int_0^T \mathrm{d}t\, b(\mathbf{r}_t,\mathbf{p}_t) \tag{1.6}$$

This opens the door to numerical experiments: simulating *in silico* the evolution of our system over a given period of time, a portion of the phase space will be explored, and we can average the properties we're looking for. Such a simulation can be seen as a (complicated) *thought experiment*[3] approximating real physical systems.

Having understood the relevance – and the need – for simulations, we have to define *how* to carry it out. And thus we enter the realm of Molecular Dynamics.

The main idea behind Molecular Dynamics (MD) is to model the interactions occurring at the microscopic level using classical models, and simulate the time evolution of the system through Newtonian dynamics. No explicit electrons are described (which, considering the classical and Newtonian nature

of the approach, is rather reassuring). In this work, we will focus on all-atoms simulations, where all atoms are explicitly represented. Coarse grain approximations,[4] where groups of atoms are modelled using single pseudo-atoms, or implicit solvents,[5] describing the surrounding solvation molecules as a continuous medium, have not been studied here.

So far, we have defined a system $\mathcal{S}$, a simulation box, a statistical ensemble to work in. To carry out simulations, we need two extra tools: one for computing forces acting on the atoms, namely a *force-field*, and one for updating **r** and **p** after a *time-step* δt has elapsed, namely an *integrator*.

In the following sections, we will present the most typical and straightforward numerical integrators, then an overview of force fields.

1.2 Integrating dynamics using finite differences

Let us assume that at a given time t, we know the positions $\mathbf{r}_t$ and velocities $\mathbf{v}_t$, and that we also have computed the forces acting on each atom $\vec{F}_i$ for each i (this will be the subject of sections 1.3 and 1.4). Let us also define δt, a small length of time, as our *time-step*. It designates the time elapsed between two successive computation of the systems position and velocities. Starting from these data, we want to compute the positions and velocities at time $t + \delta t$ ($\mathbf{r}_{t+\delta t}$ and $\mathbf{v}_{t+\delta t}$).

Newton's third law links the dynamical variables and the computed forces using

$$\vec{F}_i = m\vec{a}_i \tag{1.7}$$

A simple Euler approximation gives us the following expression for the accelerations and velocities.

$$\mathbf{a}(t+\delta t) = \frac{\mathbf{v}(t+\delta t) - \mathbf{v}(t)}{\delta t}, \quad \mathbf{v}(t+\delta t) = \frac{\mathbf{r}(t+\delta t) - \mathbf{r}(t)}{\delta t} \tag{1.8}$$

Combining these two together yields

$$\mathbf{a}(t+\delta t) = \frac{\mathbf{r}(t+\delta t) - \mathbf{r}(t) - (\mathbf{r}(t) - \mathbf{r}(t-\delta t))}{\delta t^2} \tag{1.9}$$

Inverting this relation to extract the positions at $t + \delta t$ gives the basic Strömer-Verlet scheme:[6]

$$\mathbf{r}(t + \delta t) = 2\mathbf{r}(t) - \mathbf{r}(t - \delta t) + \mathbf{a}(t)\delta t^2 + O(\delta t^2) \tag{1.10}$$

Velocities can here be computed *a posteriori* using the mean value theorem:

$$\mathbf{v}(t) = \frac{\mathbf{r}(t + \delta t) - \mathbf{r}(t - \delta t)}{2\delta t} + O(\delta t^2) \tag{1.11}$$

Using this integration scheme requires knowledge of the positions of the last two time-steps, $\mathbf{r}(t)$ and $\mathbf{r}(t - \delta t)$, which is in fact problematic when starting the dynamics.

To avoid this, one can use the Velocity-Verlet algorithm.[7] This second algorithm also explicitly computes the velocities at each time-step, which are useful when computing physical quantities such as the temperature. It is divided in three steps.

1. Unlike the Strömer-Verlet, we now start from a second order Taylor expansion of the positions:

$$\mathbf{r}(t + \delta t) = \mathbf{r}(t) + \mathbf{v}(t)\delta t + \frac{1}{2}\mathbf{a}(t)\delta t^2 + O(\delta t^2) \tag{1.12}$$

2. From the forces $\vec{F}_i$, one computes the accelerations $\mathbf{a}(t + \delta t)$ using Newton's second law (1.7) .

3. One finally has access to the velocities through

$$\mathbf{v}(t + \delta t) = \mathbf{v}(t) + \frac{1}{2}\left[\mathbf{a}(t + \delta t) - \mathbf{a}(t)\right]\delta t^2 + O(\delta t^2) \tag{1.13}$$

A more widely used implementation of the Velocity-Verlet uses half-step velocities, and expands to four steps (the computational cost will be sensibly equal to the previous implementation, as no extra expensive steps are taken).

1. Compute the "half-step" velocities (*i.e.* the velocities at time $t + \delta t/2$)

$$\mathbf{v}(t + \frac{\delta t}{2}) = \mathbf{v}(t) + \frac{\delta t}{2}\mathbf{a}(t) \tag{1.14}$$

2. Compute the positions at next time-step using the half-step velocities:

$$\mathbf{r}(t + \delta t) = \mathbf{r}(t) + \delta t \mathbf{v}(t + \frac{\delta t}{2}) \tag{1.15}$$

3. Update the forces and thus the accelerations $\mathbf{a}(t + \delta t)$ using the newly computed positions

4. Compute the "full-step" velocities:

$$\mathbf{v}(t + \delta t) = \mathbf{v}(t + \frac{\delta t}{2}) + \frac{\delta t}{2}\mathbf{a}(t + \delta t) \tag{1.16}$$

These methods ensure time-reversibility of the dynamics, which means that, starting from the positions and velocities at time $t+\delta t$, and using the Verlet or Velocity-Verlet algorithm to compute positions and velocity at time t (*i.e.* with a negative time-step $-\delta t$), we would recover the original phase-space point $(\mathbf{r}(t), \mathbf{v}(t))$.

Using such integrators also conserves *symplecticity*, which can be seen as conservation of the phase-space volume throughout dynamics. We won't go in further details here, as the reader can find more extensive explanations in [3].

Another family of integrators was proposed by Schofield and Beeman[8, 9] providing improved accuracy. These methods are based on the combined use of predictors and correctors, allowing for time savings. Unfortunately, for most methods, such speedups come at the cost of time-reversibility.

Now that the framework for our simulations is well defined, and we have a proper integration method, the only missing part is the force fields.

1.3 Force fields

A "force field" designates a set of mathematical models which associates an energy term to any atomic interactions the model takes into account. An important point that should be clarified before going any further is that we won't consider any chemical reaction in this work[i].

[i] For taking reactivity into account, force fields such as ReaxFF[10] (among others) have been developed for years now.

The forces that will drive the molecular dynamics are computed as gradients of the force field's energy terms. More explicitly, the force acting on particle i and arising from particle j is defined as

$$\vec{f_{j/i}} = -\overrightarrow{\text{grad}}\left(E_{ij}\right) = \begin{pmatrix} -\frac{dE_{ij}}{dx_{ij}} \\ -\frac{dE_{ij}}{dy_{ij}} \\ -\frac{dE_{ij}}{dz_{ij}} \end{pmatrix} \tag{1.17}$$

where β_{ij} denotes the β ($\beta = x, y, z$) component of the vector $\vec{r_j} - \vec{r_j}$, and E_{ij} is the interaction energy between particles i and j.

Pioneering work was done by Westheimer,[11] originally trying to link an energy measure with a molecules geometry; they were later implemented and improved by Hendrickson.[12] This was focused on organic chemistry compounds (the ballet of cyclohexane conformations is a wonderful problem for geometry optimization). The first popular force field (MM1) was then proposed by NL Allinger,[13] closely followed by MM2 in 1977.[14]

1.3.1 Classical force fields

Bonded terms

The very first force fields only aimed at understanding molecular conformation, and thus only encompassed intramolecular terms. Here, we can distinguish three types of energy terms: bond stretching, angle bending, and torsions.

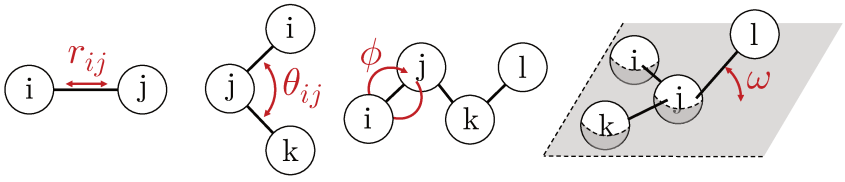


Figure 1.1: **Intramolecular energy terms.** From left to right : E_{bond}, E_{angle}, E_{torsion} and E_{improper} as presented in the text.

- The bond stretching are modelled using a simple Taylor expansion around an equilibrium posi-

tion r_0:

$$E_{\text{bond}} = k_{\text{bond}}(r_{ij} - r_0)^2 \tag{1.18}$$

(k_{bond} is a stiffness constant fitted beforehand, r_{ij} is the distance between atoms i and j). Of course, this expansion can be extended to higher orders, to include anharmonic corrections.

- Considering three atoms i, j and k, with i and k chemically bonded to j, and denoting θ_{ijk} the angle formed between ij and jk bonds, then the same harmonic approach can be adopted to account for the angle variation:

$$E_{\text{angle}} = k_{\text{angle}}(\theta_{ijk} - \theta_0)^2 \tag{1.19}$$

Again, higher order of this expansion can be taken into account for more refined description of the atomic behaviour.

- A *torsion* term is also to be added here to account for the conformation barrier when rotating an atom around an axe as shown in 1.1. Considered to be less stiff,[15] this torsion term is modelled using cosine functions rather than a harmonic approximation.

$$E_{\text{torsion}} = V_{\text{torsion}} \cos(n\phi - \phi_0) \tag{1.20}$$

Here, V_{torsion} is the energy of the barrier height, ϕ the angle as shown in figure 1.1, n the rotation periodicity, ϕ_0 the equilibrium angle (where the energy is minimal).

- One last energy term allows to reproduce the behaviour of planar (or locally planar) molecules, as for example around sp^2 carbon atoms. This is usually called *improper* torsion terms.

$$E_{\text{improper}} = k_{\text{imp}}(\omega - \omega_0)^2 \tag{1.21}$$

ω and ω_0 are the angle measuring deviation to the plane and the equilibrium one, respectively.

Intermolecular terms

Intermolecular terms represent all interactions between molecules at any distance. In the classical force field domain, there are two main terms: the electrostatic interactions, and the van der Waals ones.

- Electrostatic interactions, in the simplest form, are modelled through Coulomb's law, and account for the force arising between pairs of point-charges (the atoms). Considering atoms i and j and their respective point-charges q_i and q_j, the electric potential created at position $\vec{r_i}$ by atom j is

$$V_j(i) = \frac{q_j}{r_{ij}} \tag{1.22}$$

  Then, the energy arising from interaction of charge q_i with this potential reads

$$E_{\text{Coulomb}} = q_i V_j(i) = \frac{q_i q_j}{r_{ij}} \tag{1.23}$$

  While being a satisfying first order when describing long range interactions, atomic orbitals have an anisotropic effect (for the p and d ones), and are thus not well described. The point-charge model is in fact the first order of a point-multipole expansion, that can be expanded to dipoles, quadrupoles, or even octupoles.

  We will write $\vec{\mu_{P,i}}$ and $\Theta_{P,i}$ the *permanent* dipole and quadrupole located on atom i (not to be confused with the *induced* dipoles that will be studied later); and $\vec{E_j}(i)$ the electric field experienced at atom i's position, generated by another atom j. Then, the electrostatic energy term arising from the permanent multipoles of atom i under the influence of the field created by atom j is

$$E_{\text{electrostat}} = q_i V_j(i) + \vec{\mu}_{P,i}.\vec{E}_j(i) + \Theta_{P,i} : \nabla(\vec{E}_j(i)) \tag{1.24}$$

  ($\nabla(\vec{E_j}(i)$ is the 3×3 tensor containing the gradients of every component of the electric field $\vec{E_j}(i)$, and the ":" operator designates a simple contraction).

- The van der Waals interactions are usually represented using a Lennard-Jones potential, which shows a quite simple form:

$$E_{\text{LJ}} = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right] \tag{1.25}$$

ϵ characterizes the depth of the potential well, or more physically, the highest attraction energy between atoms i and j; σ is the distance for which the interaction is zero. Two terms are easily extractible from this formula. $\left(\frac{\sigma}{r_{ij}}\right)^{12}$ accounts for the repulsive part when two atoms are really close (it grows to very high energy if $r_{ij} \leq \sigma$). $-\left(\frac{\sigma}{r_{ij}}\right)^{6}$, on the other hand, accounts for the attractive interaction that occurs at long distance.

These terms depend on atomic positions, but also on parameters fixed beforehand, such as the harmonic constant k_{bond}, k_{angle}, or the ϵ and σ involved in Lennard-Jones potential. For example, let us imagine a diatomic molecule, whose chemical bond would be modelled as presented in eq. 1.18. Here, defining k and r_0 are very opened choices.

One may want to use the results of quantum computations, basing the model on *ab initio* calculations. One could also imagine using results from experiments (such as spectroscopy) for these two parameters. By a (slower) fitting process, another possibility would be to fit this length such that macroscopic properties computed through simulations would reproduce experimental ones.

Given the dependence of the bonding on the atom types involved, every possible pair of bonded atoms A, B should have its specific constant $k_{\text{bond}}(A, B)$: this fitting should thus be repeated for every atom pair that one wants to see bonded in a simulation.

This can even be pushed further, for the sake of accuracy of the model. The oxygen-oxygen bond in the O2 gas is completely different from the bond linking the same atoms in a carboxylic acid, just like the carbon-hydrogen bonds in benzene and in methane are expected to have different behaviours. There can thus be, for a parameter as simple as this spring constant k, many different values, possibly fitted for a specific molecule or type of molecule.

Fitting these parameters appears as a very system-dependent process, and over the years, force fields with very various domains of applications have been developed. A wide zoology of classical force fields thus exists, with differences in their parametrization and models. UFF[16] (Universal Force Field) is a force field with parametrization for all elements in the periodic table, including the actinids. CFF[17] (Consistent Force Field) targets more organic compounds, but also polymers and metals. CHARMM[18] early versions (Chemistry at HARvard Macromolecular Mechanics) and AMBER[19] (Assisted Model Building with Energy Refinement) are also classical force fields, both aiming at simulating biochemical systems

such as nucleic acids and proteins. OPLS[20] (Optimized Potentials for Liquid Simulations) was specifically fitted to reproduce liquid phase properties. This list is of course not extensive, and the interested reader could refer to [21], section II.D, for a wider zoology.

The search for more accurate and realistic simulations is of course limited by the deafening silence of electrons in classical force fields (apart maybe from the anisotropy of point-multipole expansions in the electrostatic terms). We will see in next section we can get closer to this conceptual border.

1.4 Polarizable force fields

While explicit electrons are out of the picture, given our classical mechanics framework, one could argue that all the energy terms presented so far actually implicitly take electrons into account, *i.e.* in covalent terms and electrostatics. The chemical bond for example, which we model as a Taylor expansion around an equilibrium position, is directly the consequence of electrons and nuclei interacting. The same can be said about all intramolecular terms in 1.3.1.

Non-bonded terms, and more specifically the electrostatics using multipolar expansion, represent the electronic density and its anisotropic nature. Yet the electronic density around an atom is not only directional: it changes with time, under the influence of the external electric field. Amongst others, this is represented by the *polarizability* (eq. 1.27). This electronic mobility will be taken into account in our model as a supplementary term that will follow us for quite some time in this work: the *induced polarization*. Rigorously speaking, denoting $\rho(\vec{r})$ as the electronic density at position $\vec{r}$, the dipolar moment can be expressed as:

$$\vec{\mu} = \int \rho(\vec{r})\vec{r}\mathrm{d}\vec{r} \tag{1.26}$$

and the polarizability is then simply the partial derivative of this vector with respect to the electric field $\vec{E}$

$$\alpha = \frac{\partial\vec{\mu}}{\partial\vec{E}} \tag{1.27}$$

As one could expect, this effect plays an important role when looking at systems including highly polarizable molecules, such as ionic liquids or electrolytes.[22, 23]

Working within the Born-Oppenheimer approximation, we consider that the movement of the nuclei can be decoupled from the one of electrons, the latter being much faster than the former. Induced polarization will thus be computed at each time-step of the simulation. For this reason, we need a *model* for this electronic effect, and its computational cost must stay acceptable.

1.4.1 Polarization models

Mean-field approximations

In early developments, polarization was not taken explicitly into account. Indeed, it was implicitly included as part of the van der Waals through parametrization. In practice, the induced dipole term simply modified the parameters in the Lennard-Jones (or equivalent) potential. Such strategy can be called a mean-field approximation. Of course, it comes with the cost of a lost anisotropy (directionality of the induced polarization). All the so-called "classical force fields" use this approximation, and provided over the years very extensive results on many fields of application ranging from biology (CHARMM,[24] AMBER[25]) to ionic liquids.[26]

Fluctuating charges

Considering that atomic nuclei charges are partially screened by their surrounding electrons, a first measure of the electronic density distribution could be done by looking at partial charges (within a molecule, if the partial charge on an atom i is high, then it could describe the polarization of the electrons away from it, subsequently diminishing the screening effect).

The fluctuating charges model (also called "electronegativity equalization") is based on such a redistribution of atomic partial charges within a molecule to recreate the fluctuation of the electronic density. Different implementations were proposed, relying on different invariants to perform the time integration. The most straightforward supposes conservation of the total molecular charge.[27] It is used in force fields such as CHARMM-FQ.[28, 29]

Because of the punctual nature of the partial charge distribution, however, the subsequent polarization effects are completely dependent on the geometry of the molecule. Indeed, when considering a planar one (such as benzene), redistributing charges allows for polarization parallel to the plane, but none perpendicular to it. Anisotropy of the polarization effects is thus not fully reproduced using this

model. Furthermore, charge conservation of the system is usually implemented by a global Lagrange multiplier leading to non-physical coupling between atoms regardless of their distance, which induces long-range charge-transfer that should not be observed.[30]

Drude oscillators

Another possible point of view to describe the electronic cloud, seemingly oversimplified, would be to represent its center (the mean position of the electrons) as a single fictitious point, bearing a (negative) partial charge. The Drude oscillator[ii] model uses this approach, and links this point-charge to its nuclei through a harmonic spring[iii]. Fluctuations of the electrostatic environment will then have direct repercussions on the point charge's dynamic, as a charge moving in an external electric field.

In the model, three energy terms are necessary. If we note N_D the total number of atoms with a Drude fictitious particle, $r_D(i)$ the position of the Drude particle attached to atom i, q_D its partial charge, r_i the position of atom i, k_D the stiffness constant of the spring, the total energy when using Drude oscillators is

$$E_{\text{Drude}} = \sum_{i=1}^{N} \frac{1}{2} k_D (r_D(i) - r_i)^2 + \sum_{\substack{i=1,N \\ j=1,N_D}}^{i \neq j} \frac{q_D(j) q_i}{|r_D(j) - r_i|} + \sum_{\substack{i,j=1 \\ i<j}}^{N_D} \frac{q_D(j) - q_D(i)}{|r_D(j) - r_D(i)|} \tag{1.28}$$

By order of appearance in eq. 1.28, it consists in the harmonic spring's energy, and two additional electrostatic energies: the interaction between atomic charge on atoms and Drude particles, and the interaction of Drude particles pairs.

This total energy term can be seen as an energy functional $E_{\text{Drude}}[\mathbf{r}_D]$, which has to be minimized for each time-step to find the correct position of Drude's fictitious particles. This minimization can be rewritten as a linear system to solve, or equivalently, as a matrix that one needs to invert.

To avoid the high cost of this method, Drude particles are traditionally used as supplementary degrees of freedom, in an extended Lagrangian scheme. The mass of each concerned atom is partitioned between the fictitious particle and the parent atom to which it is attached. If the mass attached to the fictitious particle is too small, it will result in very high-frequency motions, which will require very small

[ii] Also known as "core-shell model" in the solid state community.
[iii] Only non-hydrogen atoms carry this fictitious "Drude particle"

time-steps to be correctly simulated. On the other hand, if it is too heavy, the response of the Drude particles will not be fast compared to the evolution of the nuclei, which is contradictory with the Born-Oppenheimer approximation.

The motion of all particles is then computed with usual integration methods. This avoids the need for expensive matrix iterations, yet is not so computationally cheap. Indeed, the number of electrostatic interactions is larger (it now contains the three extra terms presented in equation 1.28). Furthermore, Drude oscillators treated within an extended Lagrangian scheme do not allow for large time-step integration methods (as presented in chap. 4), so they can not benefit from this important acceleration.

This model is implemented in various packages, such as CHARMM-Drude,[31] GROMACS,[32] OpenMM,[33] NAMD[34] (amongst others).

Induced dipoles

A simple mathematical object to represent polarization could be a vector, whose direction would signify the polarization spatial distribution, and whose norm would measure the intensity of the effect. In the limit of an infinitely small vector, on obtains a point dipole (similar to the multipolar expansion used in the electrostatic treatment in 1.3.1)[iv].

Since the electric field perceived at an atom's position drives the polarization of its electronic density, we will adopt the simplest possible relation between these two quantities, and assume that the induced dipole $\vec{\mu}_i$ is proportional to $\vec{E}_i$:

$$\vec{\mu}_i = \alpha_i \vec{E}_i \tag{1.29}$$

with α being the *polarizability*. Since we are talking about vectors here, α should be a 3×3 tensor rather than a simple scalar. Indeed, using a simple scalar would mean that every component of the induced dipoles would be proportional only to the same component of the electric field (*e.g.* $\mu_{i,x} = \alpha E_{i,x}$). This does not allow for anisotropy in the origin of the polarization (the x component of the field has no influence on the y component of the dipoles).

[iv] The Drude oscillator model, by using a couple of opposite charges each on a different point in space, actually defines such a dipole. Starting from this model, and looking at the limit where the distance between the atom and the Drude particle tends to zero, one recovers the induced dipole model.

Using a tensor, however, one can rewrite eq. 1.29 as

$$\vec{\mu}_i = \begin{pmatrix} \alpha_{xx} & \alpha_{xy} & \alpha_{xz} \\ \alpha_{yx} & \alpha_{yy} & \alpha_{yz} \\ \alpha_{zx} & \alpha_{zy} & \alpha_{zz} \end{pmatrix} \begin{pmatrix} E_{i,x} \\ E_{i,y} \\ E_{i,z} \end{pmatrix} \tag{1.30}$$

which allows for cross terms, as for example $\mu_{i,x} = \alpha_{xx}E_{i,x} + \alpha_{x,y}E_{i,y} + \alpha_{x,z}E_{i,z}$.

That being said, models usually use null cross-terms ($\alpha_{xy} = \alpha_{xz} = 0$), which essentially means that a simple proportionality holds between same components of both vector ($\mu_{i,\beta} = \alpha_{\beta\beta}E_{i,\beta}$).

For future reference, we will write $\boldsymbol{\alpha}$ the $3N \times 3N$ matrix containing all polarizability tensors:

$$\boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 & & 0 \\ & \ddots & \\ 0 & & \alpha_N \end{pmatrix} \quad \text{with } \alpha_i = \begin{pmatrix} \alpha_{i,xx} & 0 & 0 \\ 0 & \alpha_{i,yy} & 0 \\ 0 & 0 & \alpha_{i,zz} \end{pmatrix} \tag{1.31}$$

Induced dipoles have proven to be best suited for highly polarizable systems, such as ionic liquids, yielding slightly better accuracy[23, 35, 36] than Drude oscillator. Their real advantage is their flexibility in terms of time integration. Contrary to Drude oscillators simulations, where the movement of the fictitious particles is a fast motion limiting the possibilities to use large time-step integration, induced dipoles allow for higher time-steps. This designates this model as a better candidate for experiments on the integration methods, as chapter 4 will illustrate.

Another advantage over the Drude oscillator is in a simplified parametrization. The explicit presence of polarizabilities (α_i) allows direct use of experimental or *ab initio* results,[37] whereas Drude requires a non-trivial balancing between k_D and q_D to reproduce correct polarizabilities.

It should also be pointed out that, contrary to the Fluctuating Charges model, there is no risk of non-physical charge transfer here, as atomic charges are purely fixed parameters.

The induced point-dipoles model is the choice that we will work with in the remainder of this work. They are implemented in AMBER,[38] in the AMOEBA force field[39] within Tinker packages,[40, 41, 42] to only

cite a few. As such, we should also remind the approximations it supposes before further investigations.

- Firstly, the assumption that electronic density can simply be represented using point dipoles is quite strong, as it is known that its spatial extension is much more complex.

- Secondly, choosing a simplified polarizability tensor makes sense from a computational point of view, as a more involved choice would imply severe complications in the implementation. However, it supposes that the polarizability, eventhough it results from polarization of highly anisotropic entities (the atomic orbitals), will yield non-directional results.

- Thirdly, the electrostatic interactions are assumed to be stopped after the dipole-dipole term. Yet one could suppose that more complex shapes, in order to better represent the polarization, should be taken into account here (induced quadrupoles, for example). Here again, the effort needed to reach this higher order description would be tremendous.

The polarization catastrophe and Thole damping

One can define *molecular polarizability* as the inverse of the polarization matrix[a] restricted to the atoms within a single molecule. It relates to the molecular induced dipole moment, since for a diatomic molecule AB, $\vec{\mu_{\text{mol}}} = \vec{\mu_A} + \vec{\mu_B}$.
Applequist *et al.* showed that the molecular polarizabilities could diverge (reach infinite values) when atoms are close,[43] deriving a simple example on a diatomic molecule. This effect is known as the "polarization catastrophe". Applequist's first answer was to choose lower polarizability parameters, to reduce the molecular polarizabilities amplitudes. Thole *et al.* proposed a more general solution in the shape of a damping function compensating the divergence at short distances, effectively avoiding the catastrophe in simulation.[44]

[a] see chapter 2.

The zoology of force fields is of course wide, and this section is by no means an exhaustive study. Polarizable force fields have also been treated using topological atoms and machine learning.[45, 46] *Ab initio* force fields, encompassing more short-range quantum effects, were also developed: the SIBFA

model (Sum of Interactions Between Fragments *Ab initio* computed)[47] is a force field taking exchange-repulsion and charge transfer into account, thanks to a careful fitting based on *ab initio* calculations. Cisneros *et al.* also proposed a force field based on electronic density called GEM (Gaussian Electrostatic Model).[48, 49, 50]

Having chosen a physical model to carry out our simulations, we now finally need an infrastructure where we can develop and explore polarizable molecular dynamics: this is where our Tinker-HP code comes into play.

1.5 A massively parallel framework for Molecular Dynamics: Tinker-HP

All the developments and computations that will be presented in the next chapters were carried out using Tinker-HP (see [42], reproduced in Appendix). Tinker-HP is a high-performance version of the Tinker package,[40] initially developed by Jay Ponder at Washington University. It inherited from its simplicity in terms of implementation, and user-friendliness. Indeed, Tinker was primarily designed as a sandpit for experimenting, testing, creating force fields, algorithms and models.

Nevertheless, the initial Tinker implementations were really slow, and not competitive with the state-of-the-art simulation programs such as NAMD,[51] GROMACS,[52] etc. Tinker-HP, while maintaining the most useful features of the Tinker original package, is designed for high performance computations. Its MPI parallel structure can make efficient use of thousands of cores, and efforts were put in proper vectorization of the code, yielding substantial gains in computation speed[53] to take advantage of present petascale high-performance supercomputers but also to simply offer acceleration on everyday laboratory computer clusters.

The following sections briefly present the algorithms and methods on which these improvements were based. An introduction to the Particle Mesh Ewald treatment of boundary conditions is then given.

1.5.1 Parallel implementation

Spatial decomposition

When working with large computational resources, the goal is to efficiently divide the workload between the cores. In Tinker-HP, this is based on a *three dimensional spatial decomposition*. The simulation box

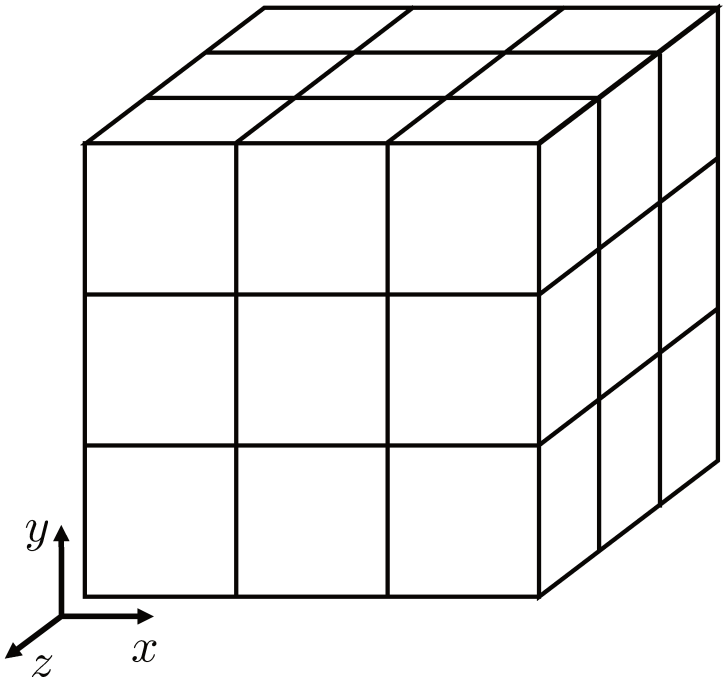


Figure 1.2: Division of the simulation box in 27 subdomains

is divided in d sub-domains of space, as schematized on figure 1.2. If we count a total of N_p processors, all computations concerning one sub-domain is realized by a subset of N_p/d processors. Each of these processor subsets is thus responsible for a specific region of the simulation box.

This division is based on assumption that interactions that are to be taken into account are short range enough, such that it makes sense to group near atoms, as the forces they experience will mostly be the consequence of their nearest neighbours. Usually, the use of cutoffs justifies this assumption (this is the case *e.g.* for PME's direct part).

Two main jobs thus fall to the processors:

1. the *computation*, which encompasses the calculation of forces given a subset of the atomic positions (see the *Midpoint technique*), and the update of positions and velocities using the forces;

2. the *communication*, which supposes to send (and receive) the quantities needed for the calculation of the forces, then to send (and receive) the result of the computations to the correct processors. It also contains the reassigning process: when an atom crosses its subdomain boundary to reach another one, it will be treated by a new processor. The list of atoms that are under the "responsibility" of one processor is thus updated at each time-step.

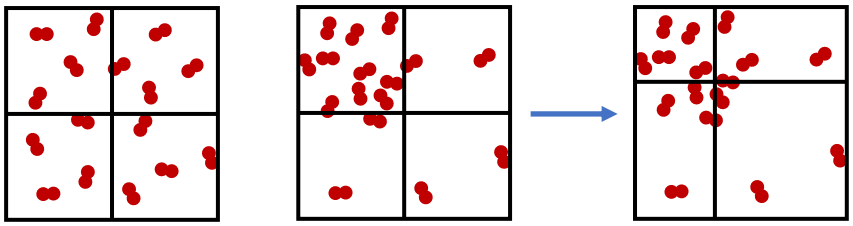


Figure 1.3: On the left, a homogeneous system divided in four subdomains. On the center and right, an inhomogeneous system, with an adaptated division in subdomains on the right panel.

To achieve a correct work balance between cores, a reasonably homogeneous system is the simplest and best candidate (it should be the case for liquid phase simulation). Indeed, discrepancies in the spatial distribution of the particles would mean that some processors would have more to do than others, effectively slowing down the whole computation. Resizing domains to have them contain approximately the same number of atoms can also be undertaken to avoid this problem (subdomains should be bigger in low-density regions and smaller in high-density ones). See fig. 1.3.

Midpoint technique

Communications can become a computational bottleneck in the parallel calculations, effectively slowing down the simulationn. This is especially the case when considering high numbers of CPU (or very large systems), where a lot of information has to be exchanged between processors. To minimize this loss of time, one can consider the pairwise nature of the elementary components of the forces driving our simulations: noting $\vec{f}_{j/i}$ a force applied on atom i from atom j, Newton's third law insures that $\vec{f}_{j/i} = -\vec{f}_{i/j}$.

Consequently, considering a pair of atoms a and b in two different subdomains, hence under the responsibility of two different processors, one has to choose one of the two processors to perform the computation and then communicate the result to the other. Very simple (almost naive) geometrical arguments could be invoked here (choosing the domain with the biggest x, y or z component for example), but following Shaw *et al.*,[54] Tinker-HP uses the midpoint technique, a choice adapted for many-body interactions minimizing the amount of information required for each processor.

1.5.2 Boundary conditions and the Particle Mesh Ewald

When trying to model condensed phase, solvation effects are particularly important. To deal with the electrostatic and polarization interactions composing these effects, one can use continuum solvation models, representing the solvent molecules implicitly as a continuous medium.[55] In our case, however, we work with fully explicit models, describing the solvent at the molecular level. A very simple, but fundamental problem then arises: how does one mimic an infinite medium using our finite computations ? The answer comes from the boundary treatment, where we can choose to use *periodic boundary conditions* (PBC). Periodic boundary conditions suppose that the simulation box (we will assume it is shaped as a parallelepiped) is repeated in every direction of space, as illustrated in fig. 1.4.

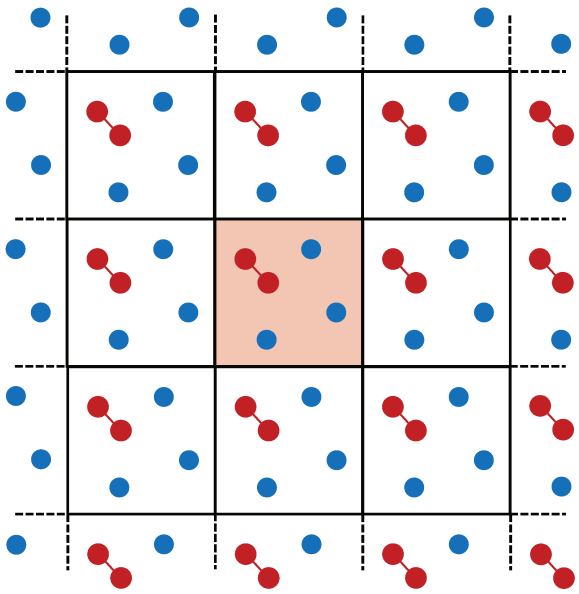


Figure 1.4: **Illustration of the Periodic Boundary Conditions** in two dimensions. The central square, colored, is the initial system's simulation box. All around are the replica.

Let us look at one of the simplest non-bonded energy terms here, namely the electrostatic interaction between charges. It reads

$$E_{qq} = \sum_{\vec{n}}^{(\vec{n})} \sum_{\substack{i=1,N \\ i<j}} \frac{q_i q_j}{||\vec{r}_{ij} + \vec{n}||} \tag{1.32}$$

where the sum over all $\vec{n}$ is the infinite sum over all the PBC replica, with each $\vec{n}$ vector representing a translation towards a specific replica[v]. $\sum^{(\vec{n})}$ means that for $\vec{n} = \vec{0}$, the terms with $i = j$ should not be

summed. We will illustrate the method presented in this section through this charge-charge term only, although permanent dipoles and quadrupoles are also used in our simulations.

A difficulty arises here, as each atom of our system now has an infinity of neighbors (represented in 1.32 by the infinite sum over $\vec{n}$) – which would mean an infinity of non-bonded interactions to compute.

Fortunately, the terms in this sum are decreasing as $\frac{1}{r}$. This means that, past a certain distance r_C, we can neglect the value of the interaction for being small enough. This distance r_C is called a *cutoff*, and acts as a limit for the interaction ranges. Practically, this means that given an atom i, its interaction with atom j (whether in the original simulation box or in one of the replicas) is only computed if the distance r_{ij} is smaller than the cutoff ($r_{ij} < r_C$). The typical values that are used in this cutoff have the same order of magnitude as the simulation box size. For each atom, the number of interactions to be computed is thus of the order N; as a consequence, the complexity of computing such a pairwise interaction, when using periodic boundary conditions and a cutoff, scales as $O(N^2)$.

This cost can however be improved. Ewald summation[56] splits this sum into two absolutely converging sums, a *direct sum* and a *reciprocal sum*, supplemented by a small correction term:

$$E_{\text{elec}} = E_{\text{direct}} + E_{\text{recip}} + E_{\text{self}} \tag{1.33}$$

This splitting is controlled by a a real, positive number β, whose value defines a distance separating the direct and reciprocal terms of the total sum. If β is such that only the simulation box, without any periodic image, is taken into account in the E_{direct} interactions terms, we can detail this sum as hereafter. Again, we are only looking at the simplest electrostatic term here, *i.e.* the sum of all interactions between

[v] If we note $\vec{a}$, $\vec{b}$ and $\vec{c}$ the vectors defining the simulation box (the *lattice* vectors), then each $\vec{n}$ are defined as $\vec{n} = i_a\vec{a} + i_b\vec{b} + i_c\vec{c}$, where i_a, i_b, i_c are three integers.

point-charges.

$$E_{\text{direct}} = \sum_{i=1}^{N} \sum_{j=i+1}^{N} q_i q_j \frac{\text{erfc}\left(\beta r_{ij}\right)}{r_{ij}} \tag{1.34}$$

$$E_{\text{recip}} = \sum_{i=1}^{N} \sum_{j=i+1}^{N} q_i q_j \Phi_{\text{rec}}\left(\vec{r}_{ij}; \beta\right) \tag{1.35}$$

$$E_{\text{self}} = \frac{\beta}{\sqrt{\pi}} \sum_{i=1}^{N} q_i^2 \tag{1.36}$$

erfc is the complementary error function, which allows for a smooth switching off of the direct contribution as a function of the distance r_{ij}[vi]. The $\Phi_{\text{rec}}(\vec{r}, \beta)$ potential is

$$\Phi_{\text{rec}}(\vec{r}, \beta) = \frac{1}{\pi V} \sum_{\vec{m} \neq \vec{0}} \frac{\exp\left(\frac{-\pi^2 \vec{m}^2}{\beta^2}\right)}{\vec{m}^2} \mathrm{e}^{2\pi \mathrm{i} \vec{m}.\vec{r}} \tag{1.38}$$

Here, the vectors $\vec{m}$ are defined as linear combinations of the reciprocal lattice vectors $\vec{a}^*$, $\vec{b}^*$, $\vec{c}^*$, such that $\vec{m} = i_a \vec{a}^* + i_b \vec{b}^* + i_c \vec{c}^*$ with i_a, i_b and i_c are integers. The parameter β can be chosen such that the total complexity of the computation drops to $O(N^{3/2})$ if it is well chosen (see [57]).

Darden proposed a method to improve the computation of the reciprocal sum (eq. 1.35) called Particle Mesh Ewald (PME).[57] The complex exponential terms of equation 1.38 are interpolated on a grid. This allows one to rewrite Φ_{rec} as a convolution product. Thankfully, when switching to the Fourier space, a convolution becomes a simple product. The procedure followed to compute this sum is thus: firstly, putting the charges on the grid; secondly, using a Fourier transform to compute the charge distribution in Fourier space; thirdly, compute the convolution product in Fourier space; finally, use a backwards Fourier transform to extract the reciprocal sum's value.

The use of Fourier transform effectively accelerates the computation of this expensive term, and using fast Fourier transforms (FFTs), the complexity becomes of order $O(N \log(N))$, which is a significative improvement. As a consequence, if one chooses β such that the direct-space sum scales with

[vi] The complementary error function is defined as

$$\text{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} \mathrm{e}^{-t^2} \mathrm{d}t \tag{1.37}$$

$O(N)$, the total computational complexity becomes $O(N \log(N))$.

Analytical derivatives were introduced thanks to the use of B-spline functions for interpolation.[58] Extension to multipoles was later derived by Sagui *et al.*,[59] and induced dipoles by Toukmaji *et al.*[60] Finally, the consistent formulation and derivation of the multipole expansion and induced dipoles Ewald self terms were given by Stamm *et al.*[61]

Bibliography

[1] E. Fermi, J. Pasta, S. Ulam, and M. Tsingou, “Studies in Nonlinear Problems I,” *Los Alamos Report*, vol. LA 1940, 1955.

[2] B. J. Alder and T. E. Wainwright, “Phase Transition for a Hard Sphere System,” *The Journal of Chemical Physics*, vol. 27, pp. 1208–1209, Nov. 1957.

[3] M. Tuckerman, *Statistical Mechanics: Theory and Molecular Simulation*. OUP Oxford, Feb. 2010. Google-Books-ID: Lo3JqcopgrcC.

[4] S. Kmiecik, D. Gront, M. Kolinski, L. Wieteska, A. E. Dawid, and A. Kolinski, “Coarse-Grained Protein Models and Their Applications,” *Chemical Reviews*, vol. 116, pp. 7898–7936, July 2016.

[5] C. J. Cramer and D. G. Truhlar, “Implicit Solvation Models: Equilibria, Structure, Spectra, and Dynamics,” *Chemical Reviews*, vol. 99, pp. 2161–2200, Aug. 1999.

[6] L. Verlet, “Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules,” *Physical Review*, vol. 159, pp. 98–103, July 1967.

[7] H. Grubmüller, H. Heller, A. Windemuth, and K. Schulten, “Generalized Verlet Algorithm for Efficient Molecular Dynamics Simulations with Long-range Interactions,” *Molecular Simulation*, vol. 6, pp. 121–142, Mar. 1991.

[8] P. Schofield, “Computer simulation studies of the liquid state,” *Computer Physics Communications*, vol. 5, pp. 17–23, Jan. 1973.

[9] D. Beeman, “Some multistep methods for use in molecular dynamics calculations,” *Journal of Computational Physics*, vol. 20, pp. 130–139, Feb. 1976.

[10] A. C. T. van Duin, S. Dasgupta, F. Lorant, and W. A. Goddard, "ReaxFF: A Reactive Force Field for Hydrocarbons," *The Journal of Physical Chemistry A*, vol. 105, pp. 9396–9409, Oct. 2001.

[11] F. H. Westheimer, "Chap. 12," in *Steric Effect in Organic Chemistry*, Wiley, m.s. newman, editor ed., 1956.

[12] J. B. Hendrickson, "Molecular Geometry. I. Machine Computation of the Common Rings," *Journal of the American Chemical Society*, vol. 83, pp. 4537–4547, Nov. 1961.

[13] N. L. Allinger, "Calculation of Molecular Structure and Energy by Force-Field Methods," in *Advances in Physical Organic Chemistry* (V. Gold and D. Bethell, eds.), vol. 13, pp. 1–82, Academic Press, Jan. 1976.

[14] N. L. Allinger, "Conformational analysis. 130. MM2. A hydrocarbon force field utilizing V1 and V2 torsional terms," *Journal of the American Chemical Society*, vol. 99, pp. 8127–8134, Dec. 1977.

[15] M. González, "Force fields and molecular dynamics simulations," *École thématique de la Société Française de la Neutronique*, vol. 12, pp. 169–200, 2011.

[16] A. K. Rappe, C. J. Casewit, K. S. Colwell, W. A. Goddard, and W. M. Skiff, "UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations," *Journal of the American Chemical Society*, vol. 114, pp. 10024–10035, Dec. 1992.

[17] S. Lifson and A. Warshel, "Consistent Force Field for Calculations of Conformations, Vibrational Spectra, and Enthalpies of Cycloalkane and n-Alkane Molecules," *The Journal of Chemical Physics*, vol. 49, pp. 5116–5129, Dec. 1968.

[18] J. Huang, S. Rauscher, G. Nawrocki, T. Ran, M. Feig, B. L. de Groot, H. Grubmüller, and A. D. MacKerell Jr, "Charmm36m: an improved force field for folded and intrinsically disordered proteins," *Nature Methods*, vol. 14, pp. 71–73, Jan. 2017.

[19] J. A. Maier, C. Martinez, K. Kasavajhala, L. Wickstrom, K. E. Hauser, and C. Simmerling, "ff14sb: Improving the accuracy of protein side chain and backbone parameters from ff99sb," *Journal of Chemical Theory and Computation*, vol. 11, pp. 3696–3713, Aug. 2015.

[20] M. J. Robertson, J. Tirado-Rives, and W. L. Jorgensen, "Improved Peptide and Protein Torsional Energetics with the OPLS-AA Force Field," *Journal of Chemical Theory and Computation*, vol. 11, pp. 3499–3509, July 2015.

[21] J. W. Ponder and D. A. Case, "Force fields for protein simulations," in *Advances in Protein Chemistry*, vol. 66, pp. 27–85, Elsevier, 2003.

[22] D. Bedrov, O. Borodin, Z. Li, and G. D. Smith, "Influence of polarization on structural, thermodynamic, and dynamic properties of ionic liquids obtained from molecular dynamics simulations," *The Journal of Physical Chemistry B*, vol. 114, no. 15, pp. 4984–4997, 2010. PMID: 20337454.

[23] D. Bedrov, J.-P. Piquemal, O. Borodin, A. D. MacKerell, B. Roux, and C. Schröder, "Molecular dynamics simulations of ionic liquids and electrolytes using polarizable force fields," *Chemical Reviews*, vol. 0, no. 0, p. null, 0.

[24] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus, "Charmm: A program for macromolecular energy, minimization, and dynamics calculations," *Journal of Computational Chemistry*, vol. 4, pp. 187–217, 1983.

[25] P. K. Weiner and P. A. Kollman, "AMBER: Assisted model building with energy refinement. A general program for modeling molecules and their interactions," *Journal of Computational Chemistry*, vol. 2, no. 3, pp. 287–303, 1981.

[26] T. Köddermann, D. Paschek, and R. Ludwig, "Molecular dynamic simulations of ionic liquids: A reliable description of structure, thermodynamics and dynamics," *ChemPhysChem*, vol. 8, no. 17, pp. 2464–2470, 2007.

[27] S. Patel and C. L. B. III, "Fluctuating charge force fields: recent developments and applications from small molecules to macromolecular biological systems," *Molecular Simulation*, vol. 32, pp. 231–249, Mar. 2006.

[28] S. Patel and C. L. Brooks, "CHARMM fluctuating charge force field for proteins: I parameterization and application to bulk organic liquid simulations," *Journal of Computational Chemistry*, vol. 25, no. 1, pp. 1–16, 2004.

[29] S. Patel, A. D. Mackerell, and C. L. Brooks, "CHARMM fluctuating charge force field for proteins: II Protein/solvent properties from molecular dynamics simulations using a nonadditive electrostatic model," *Journal of Computational Chemistry*, vol. 25, no. 12, pp. 1504–1514, 2004.

[30] P. P. Poier, L. Lagardere, J.-P. Piquemal, and F. Jensen, "Molecular Dynamics Using Non-Variational Polarizable Force Fields: Theory, Periodic Boundary Conditions Implementation and Application to the Bond Capacity Model," July 2019.

[31] G. Lamoureux and B. Roux, "Modeling induced polarization with classical Drude oscillators: Theory and molecular dynamics simulation algorithm," *The Journal of Chemical Physics*, vol. 119, pp. 3025–3039, July 2003.

[32] J. A. Lemkul, B. Roux, D. van der Spoel, and A. D. MacKerell, "Implementation of extended Lagrangian dynamics in GROMACS for polarizable simulations using the classical Drude oscillator model," *Journal of Computational Chemistry*, vol. 36, pp. 1473–1479, July 2015.

[33] J. Huang, J. A. Lemkul, P. K. Eastman, and A. D. MacKerell, "Molecular dynamics simulations using the drude polarizable force field on GPUs with OpenMM: Implementation, validation, and benchmarks," *Journal of Computational Chemistry*, vol. 39, no. 21, pp. 1682–1689, 2018.

[34] W. Jiang, D. J. Hardy, J. C. Phillips, A. D. Mackerell, K. Schulten, and B. Roux, "High-performance scalable molecular dynamics simulations of a polarizable force field based on classical Drude oscillators in NAMD," *The Journal of Physical Chemistry Letters*, vol. 2, no. 2, pp. 87–92, 2011.

[35] P. E. M. Lopes, B. Roux, and A. D. MacKerell, "Molecular modeling and dynamics studies with explicit inclusion of electronic polarizability: theory and applications," *Theoretical Chemistry Accounts*, vol. 124, pp. 11–28, Sept. 2009.

[36] M. Schmollngruber, V. Lesch, C. Schröder, A. Heuer, and O. Steinhauser, "Comparing induced point-dipoles and Drude oscillators," *Physical Chemistry Chemical Physics*, vol. 17, no. 22, pp. 14297–14306, 2015.

[37] S. S. J. Rick S. W., "Potentials and algorithms for incorporating polarizability in computer simulations," in *Reviews in Computational Chemistry Vol. 18*, pp. 89–146, Wiley-VCH, 2002.

[38] R. Salomon-Ferrer, D. A. Case, and R. C. Walker, "An overview of the Amber biomolecular simulation package," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 3, no. 2, pp. 198–210, 2013.

[39] J. W. Ponder, C. Wu, P. Ren, V. S. Pande, J. D. Chodera, M. J. Schnieders, I. Haque, D. L. Mobley, D. S. Lambrecht, R. A. DiStasio, M. Head-Gordon, G. N. I. Clark, M. E. Johnson, and T. Head-Gordon, "Current Status of the AMOEBA Polarizable Force Field," *The Journal of Physical Chemistry B*, vol. 114, pp. 2549–2564, Mar. 2010.

[40] J. A. Rackers, Z. Wang, C. Lu, M. L. Laury, L. Lagardère, M. J. Schnieders, J.-P. Piquemal, P. Ren, and J. W. Ponder, "Tinker 8: Software Tools for Molecular Design," *Journal of Chemical Theory and Computation*, vol. 14, pp. 5273–5289, Oct. 2018.

[41] M. Harger, D. Li, Z. Wang, K. Dalby, L. Lagardère, J.-P. Piquemal, J. Ponder, and P. Ren, "Tinker-OpenMM: Absolute and relative alchemical free energies using AMOEBA on GPUs," *Journal of Computational Chemistry*, vol. 38, no. 23, pp. 2047–2055, 2017.

[42] L. Lagardère, L.-H. Jolly, F. Lipparini, F. Aviat, B. Stamm, Z. F. Jing, M. Harger, H. Torabifard, G. A. Cisneros, M. J. Schnieders, N. Gresh, Y. Maday, P. Y. Ren, J. W. Ponder, and J.-P. Piquemal, "Tinker-hp: a massively parallel molecular dynamics package for multiscale simulations of large complex systems with advanced point dipole polarizable force fields," *Chemical Science*, vol. 9, pp. 956–972, Jan. 2018.

[43] J. Applequist, J. R. Carl, and K.-K. Fung, "Atom dipole interaction model for molecular polarizability. Application to polyatomic molecules and determination of atom polarizabilities," *Journal of the American Chemical Society*, vol. 94, pp. 2952–2960, May 1972.

[44] B. T. Thole, "Molecular polarizabilities calculated with a modified dipole interaction," *Chemical Physics*, vol. 59, pp. 341–350, Aug. 1981.

[45] M. J. L. Mills and P. L. A. Popelier, "Polarisable multipolar electrostatics from the machine learning method Kriging: an application to alanine," *Theoretical Chemistry Accounts*, vol. 131, p. 1137, Feb. 2012.

[46] T. L. Fletcher, S. J. Davie, and P. L. A. Popelier, "Prediction of Intramolecular Polarization of Aromatic Amino Acids Using Kriging Machine Learning," *Journal of Chemical Theory and Computation*, vol. 10, pp. 3708–3719, Sept. 2014.

[47] N. Gresh, G. A. Cisneros, T. A. Darden, and J.-P. Piquemal, "Anisotropic, polarizable molecular mechanics studies of inter- and intramolecular interactions and ligand–macromolecule complexes. a bottom-up strategy," *Journal of Chemical Theory and Computation*, vol. 3, pp. 1960–1986, Nov. 2007.

[48] G. A. Cisneros, J.-P. Piquemal, and T. A. Darden, "Generalization of the Gaussian electrostatic model: Extension to arbitrary angular momentum, distributed multipoles, and speedup with reciprocal space methods," *The Journal of Chemical Physics*, vol. 125, p. 184101, Nov. 2006.

[49] J.-P. Piquemal, G. A. Cisneros, P. Reinhardt, N. Gresh, and T. A. Darden, "Towards a force field based on density fitting," *The Journal of Chemical Physics*, vol. 124, p. 104101, Mar. 2006.

[50] R. E. Duke, O. N. Starovoytov, J.-P. Piquemal, and G. A. Cisneros, "GEM*: A Molecular Electronic Density-Based Force Field for Molecular Dynamics Simulations," *Journal of Chemical Theory and Computation*, vol. 10, pp. 1361–1365, Apr. 2014.

[51] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kalé, and K. Schulten, "Scalable molecular dynamics with NAMD," *Journal of Computational Chemistry*, vol. 26, no. 16, pp. 1781–1802, 2005.

[52] B. Hess, C. Kutzner, D. van der Spoel, and E. Lindahl, "GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation," *Journal of Chemical Theory and Computation*, vol. 4, pp. 435–447, Mar. 2008.

[53] L.-H. Jolly, A. Duran, L. Lagardère, J. W. Ponder, P. Ren, and J.-P. Piquemal, "Raising the performance of the tinker-hp molecular modeling package on intel's hpc architectures: a living review [article v1.0]," June 2019.

[54] K. J. Bowers, R. O. Dror, and D. E. Shaw, "The midpoint method for parallelization of particle simulations," *The Journal of Chemical Physics*, vol. 124, p. 184109, May 2006.

[55] F. Lipparini, L. Lagardère, C. Raynaud, B. Stamm, E. Cancès, B. Mennucci, M. Schnieders, P. Ren, Y. Maday, and J.-P. Piquemal, "Polarizable Molecular Dynamics in a Polarizable Continuum Solvent," *Journal of Chemical Theory and Computation*, vol. 11, pp. 623–634, Feb. 2015.

[56] P. P. Ewald, "Die Berechnung optischer und elektrostatischer Gitterpotentiale," *Annalen der Physik*, vol. 369, no. 3, pp. 253–287, 1921.

[57] T. Darden, D. York, and L. Pedersen, "Particle mesh Ewald: An Nlog(N) method for Ewald sums in large systems," *The Journal of Chemical Physics*, vol. 98, pp. 10089–10092, June 1993.

[58] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. G. Pedersen, "A smooth particle mesh Ewald method," *The Journal of Chemical Physics*, vol. 103, pp. 8577–8593, Nov. 1995.

[59] C. Sagui, L. G. Pedersen, and T. A. Darden, "Towards an accurate representation of electrostatics in classical force fields: Efficient implementation of multipolar interactions in biomolecular simulations," *The Journal of Chemical Physics*, vol. 120, pp. 73–87, Jan. 2004.

[60] A. Toukmaji, C. Sagui, J. Board, and T. Darden, "Efficient particle-mesh Ewald based approach to fixed and induced dipolar interactions," *The Journal of Chemical Physics*, vol. 113, pp. 10913–10927, Dec. 2000.

[61] B. Stamm, L. Lagardère, E. Polack, Y. Maday, and J.-P. Piquemal, "A coherent derivation of the Ewald summation for arbitrary orders of multipoles: The self-terms," *The Journal of Chemical Physics*, vol. 149, p. 124103, Sept. 2018.

Chapter 2

Accelerating the polarization

Contents

After setting up the Molecular Dynamics framework, and choosing the induced dipoles model for polarization, we will focus in this chapter on the self-consistent field problem. We first present an overview of the state-of-the-art solvers, and we then propose a new algorithm to improve the polarization treatment, both stability and velocity-wise: the Truncated Conjugate Gradient (TCG).

2.1 Polarization solvers – state of the art

A quick look at the very nature of the induced polarization model demonstrates how non-trivial researching new solvers will be. Indeed, let us consider two atoms A and B, each having an associated induced dipole vector ($\vec{\mu_A}$ and $\vec{\mu_B}$, respectively). Under the influence of the external electric field **E**, $\vec{\mu_A}$ will be modified (in order to minimize the dipole/field interaction, the dipole will have a tendency to align itself with the surrounding field).

Since the electric field arises from both the permanent multipoles distribution and the induced dipoles contribution, a change in $\vec{\mu_A}$ results in a modification of the electric field experienced in B ($\vec{E_B}$). As it was the case for A, the induced dipole in B will thus align itself with the field, effectively changing... And as a consequence, have an influence of the electric field experienced in A ($\vec{E_A}$).

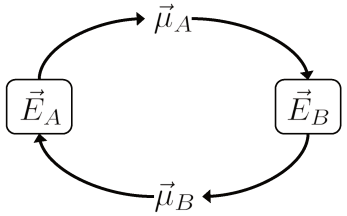


Figure 2.1: The induced dipoles: a self-consistent problem

This problem appears as quite involved, and will iterate until self-consistency between dipoles and field is reached. The following section first expresses the problem in mathematical terms, and then presents the strategies currently adopted to solve it.

2.1.1 The linear problem

The polarization matrix

Let us first define the energy functional describing the induced polarization. Since we assume that at any time during our simulation, the electrons will be at equilibrium, the correct induced dipoles themselves will be obtained by minimizing this functional.

Three terms should be taken into account here:[1]

- the interaction between the induced dipole and the electric field generated at its position by the permanent distribution of charges,
- the self-energy of the polarization (one could describe it as the energy of interaction between one dipole and the electric field that it generates),
- and the interaction between two distinct induced dipoles

The total then reads

$$E_{\text{pol}}[\boldsymbol{\mu}] = -\sum_{\substack{i=1,N \\ \beta=x,y,z}} E_i^{\beta}\mu_i^{\beta} + \frac{1}{2}\sum_{\substack{i=1,N \\ \beta,\gamma=x,y,z}} [\alpha_i^{-1}]^{\beta\gamma}\mu_i^{\beta}\mu_i^{\gamma} + \frac{1}{2}\sum_{\substack{i=1,N \\ j\neq i}}\sum_{\beta,\gamma=x,y,z} T_{ij}^{\beta\gamma}\mu_i^{\beta}\mu_j^{\gamma} \tag{2.1}$$

Here, E_i^{β} stands for the β component of the electric field experienced at atom i, and T_{ij} is the tensor accounting for the interaction between dipoles $\vec{\mu}_i$ and $\vec{\mu}_j$, defined as follows:

$$T_{ij}^{\beta\gamma} = -\frac{\delta_{\beta\gamma}}{r_{ij}^3} + 3\frac{r_{ij}^{\beta}r_{ij}^{\gamma}}{r_{ij}^5} \tag{2.2}$$

To illustrate it, let us say that $T_{ij}\vec{\mu}_j$ is the electric field created by dipole j on site i, which leads to ${\vec{\mu}_i}^T T_{ij}\vec{\mu}_j$ representing the interaction energy between these two induced dipoles.

Given the polarizability tensor presented in 1.4.1 as well as the interaction ones introduced above, we can now write the full *polarization matrix* **T**, using 3×3 blocks, as:

$$\mathbf{T} = \begin{pmatrix} \alpha_1^{-1} & -T_{12} & -T_{13} & \dots & -T_{1N} \\ -T_{21} & \alpha_2^{-1} & -T_{23} & \dots & -T_{2N} \\ -T_{31} & -T_{32} & \ddots & & \\ \vdots & \vdots & & & \vdots \\ -T_{N1} & -T_{N2} & & \dots & \alpha_N^{-1} \end{pmatrix} \tag{2.3}$$

Here, we voluntarily omitted the Thole damping factors, as they would only weigh down the notations[i].

T is symmetric, positive and definite. While the symmetry is obvious (the interaction U_{ij} between dipoles i and j is equivalent to U_{ji}). Positive definite means that all its eigenvalues should be strictly positive: this is ensured by the Thole damping[1] (see 1.4.1).

For future reference, we can also write the polarization matrix as $\mathbf{T} = \boldsymbol{\alpha}^{-1} - \mathcal{T}$, with $\mathcal{T}$ the matrix containing all the off-diagonal blocks T_{ij}.

Energy functional

Keeping notations presented in 1.1, we will write $\vec{E}_i$ (respectively $\vec{\mu}_i$) the (three-dimensional) electric field experienced (respectively the induced dipole) on one atom, and **E** (respectively $\boldsymbol{\mu}$) the $3N$ vector containing all $\vec{E}_i$ (respectively all $\vec{\mu}_i$).

Using the polarization matrix as defined above, we can finally simplify equation 2.1 as

$$E_{\text{pol}}[\boldsymbol{\mu}] = \frac{1}{2}\langle \boldsymbol{\mu}, \mathbf{T}\boldsymbol{\mu} \rangle - \langle \boldsymbol{\mu}, \mathbf{E} \rangle \tag{2.4}$$

Minimizing this energy functional, by equalizing its derivative along $\boldsymbol{\mu}$ to zero, can easily be done using the more explicit eq. 2.1:

$$\frac{\partial E_{\text{pol}}[\boldsymbol{\mu}]}{\partial \mu_i^\beta} = -E_i^\beta + [\alpha_i^{-1}]^{\beta\gamma}\mu_i^\beta + \sum_{i \neq j} T_{ij}^{\beta\gamma}\mu_j^\gamma = 0 \tag{2.5}$$

[i] Their explicit expressions can be found in [1].

Using our previously defined notations, this finally gives the following linear system:

$$\mathbf{T}\boldsymbol{\mu} = \mathbf{E} \tag{2.6}$$

and the polarization energy then reads:

$$E_{\text{pol}} = -\frac{1}{2}\langle \boldsymbol{\mu}, \mathbf{E} \rangle \tag{2.7}$$

The whole polarization question, when choosing the induced dipoles model, is essentially contained in this linear problem in equation 2.6: for each time-step, we will have to find the vector $\boldsymbol{\mu}$ solution, or equivalently invert the $\mathbf{T}$ matrix. Of course, computation of energies and forces will be necessary, but their computational cost do not exceed the price paid for solving this linear problem, as we will see in the next sections.

2.1.2 Solving the linear problem

Having identified the linear problem governing our model, we now need to propose a way to solve it. In the following, we will thus present the various *polarization solvers* that one can use.

Direct methods

When trying to invert a matrix, the first family of methods coming to mind are the *direct* ones, as they are *exact*. One can cite the LU decomposition,[2] which decomposes the matrix to be inverted in a product of a lower triangular one (L) with an upper triangular one (U). This allows one to change the non-trivial linear problem ($Ax = b$) in two successive straightforward operations (solving $Ly = b$ for y, then solving $Ux = y$).

Another candidate is the Cholesky decomposition,[2] only applicable to positive definite matrices. Much as the LU-decomposition, it expresses the matrix to be inverted as a matrix product, this time of a triangular matrix L and its conjugate transpose L^*.

Matrix inversion being a very common problem, the list of these methods is long and diverse, with refinements having been developed to exploit the various matrix shapes and properties.[3]

Their computational cost, however, is usually scaling as the cube of the matrix size ($O(N^3)$), which

becomes problematic as we want to to simulate systems containing tens to hundreds of thousands of atoms (when considering solvated proteins). Eventhough the computing resources are in constant progress, we still can not afford to perform such operations – even more so if it's required at each time-step of a simulation !

Predictor methods

A predictor is an approximation using the previous induced dipoles ($\boldsymbol{\mu}_{\text{corr}}(t-\Delta t)$, $\boldsymbol{\mu}_{\text{corr}}(t-2\Delta t)$...) to evaluate the induced dipoles at time t, namely $\boldsymbol{\mu}_{\text{pred}}(t)$. The electric field arising from this set of induced dipoles ($\mathbf{E}_{\text{pred}}$) is then used to propose a corrected version of the dipoles $\boldsymbol{\mu}_{\text{corr}}(t)$, more precise, that is then to be used in the following iterations as a basis for the predictors.

The simplest predictor version would be to use... the previous induced dipoles as the current ones:

$$\boldsymbol{\mu}_{\text{pred}}(t) = \boldsymbol{\mu}_{\text{corr}}(t-\Delta t) \tag{2.8}$$

The first order predictor reads:

$$\boldsymbol{\mu}_{\text{pred}}(t) = 2\boldsymbol{\mu}_{\text{corr}}(t-\Delta t) - \boldsymbol{\mu}_{\text{corr}}(t-2\Delta t) \tag{2.9}$$

Ahlström *et al.*[4] proposed an algorithm which only used an iterative solver every n time-step ($n \simeq 5$). It was deemed to keep satisfyingly conservative results by the authors, and used a *predictor* for every other time-step. The authors propose a range of predictors up to third order, but their result show a much better behaviour of the first-order one (eq. 2.9). The method, although it accelerates simulations, does not allow for stable simulations over long times.

Kolafa designed an "Always Stable Predictor-Corrector" (ASPC), to be used at each time-step at a cost of one self-consistent iteration, exhibiting better stability.[5] Although efforts were made to improve time-reversibility,[6] using data from previous time-steps (in this case the induced dipoles values) prevents proper conservation of this property. Time-reversibility is important as it guarantees a proper conservation of energy over the course of the simulation[ii].

[ii] Let us imagine our system at initial positions and velocities $\mathbf{q}_0$ and $\mathbf{v}_0$. After simulating a time t, suppose that an error ϵ was accumulated. One can then carry out a backwards integration of the system, which would bring the system back to the original $\mathbf{q}_0$ and $\mathbf{v}_0$. Yet it would also have accumulated an error ϵ, totaling into a 2ϵ error: for the system to come back to its exact initial position and velocity – that is, if the dynamics is truly time-reversible – this error must necessarily be $\epsilon = 0$.

Section 2.1.3 proposes iterative methods as an answer to the self-consistency problem, whose cost depends on the number of iterations needed to reach an acceptable solution. Kolafa's predictors presented above compute dipoles close to the converged solution that can be used as initial guess for the iterative solvers, such that the number of iterations required afterwards is minimized, at the cost of the time-reversibility of the dynamics.

Extended Lagrangians

Another approach to propose good approximate values for the induced dipoles before any self-consistent iteration is the extended Lagrangian scheme.[7] Induced dipoles are treated as independent degrees of freedom, affected with fictitious masses (as would be done in Car-Parrinello Molecular Dynamics[8] for molecular orbitals). We can write their associated kinetic energy as

$$T = \sum_{i=1,N} \frac{1}{2} m_p \dot{\mu_i}^2 \tag{2.10}$$

(where $\dot{\mu_i}$ denotes the time-derivative of μ_i, and m_p is the fictitious mass evoked above), while the energy functional defined in 2.1 can be used as the potential energy.

This allows one to derive equations of motion that will govern the dynamics of the induced dipoles, based on the definition of a so-called *extended Lagrangian* (description of the Lagrangian mechanics are beyond the scope of this work, and the reader can find more details in specialized work such as ref. [9], with a clarity and conciseness that yours truly could not hope to have). Essentially, Euler-Lagrange equation allows one to derive motion equations from the Lagrangian, guided by the minimization of the system's action. This yields a cheap method to compute induced dipoles, although they are not always exactly at the minimum of the polarization energy energy proposed in eq. 2.4.

Another use of the extended Lagrangian was later introduced by Niklasson *et al.*[10] in the Born-Oppenheimer Molecular Dynamics framework, where they proposed to use *auxiliary* degrees of freedom within the Car-Parrinello/extended Lagrangian framework. The role of these auxiliary degrees of freedom was in this case to serve as an initial guess of a Self-Consistent Field problem, in order to reach convergence faster (it is exactly the same purpose as presented in section 2.1.4).

Following this same idea, Head-Gordon *et al.* proposed to use auxiliary degrees of freedom to

describe induced dipoles[11] within polarizatble MD (a set $\boldsymbol{\mu}^{(N)}$ of auxiliary induced dipoles is thus used, and is constrained to stay close to the accurate value of the "real" induced dipoles). Although it means more equations to be integrated for each time-step, the gain obtained by minimizing the number of self-consistent iterations still makes it a valuable strategy for computing dipoles.

Another advantages here is the improved energy conservation allowed by the simple expression of the equations of motion of the supplementary degrees of freedom. However, the need for an additional thermostat for the auxiliary degrees of freedom is potentially problematic, as it introduces additional parameters. In practice, this extended Lagrangian strategy forbids the use of usual large time-steps integration such as RESPA, as the movement of the auxiliary degrees of freedom has to be finely resolved so as not to blow up.

2.1.3 Iterative solvers

Iterative solvers constitute an important family of solvers that has not been cited yet, although they are arguably the most used ones. They avoid the overwhelming cost of direct methods, while ensuring a better control over the precision on the computed dipoles.

The general idea behind iterative solvers is a trade-off between computation cost and precision: instead of fully inverting the matrix, which would yield the perfect solution of the linear problem (in our case, the exact induced dipoles), one decides to refine the solution until a certain convergence criterion is reached. This criterion can be chosen in different ways: it could be a threshold value on the residue norm, or on the norm of the difference between dipoles from one iteration to the next, amongst other choices.

Of course, this type of solver is only a valid solution if two requirements are met:

- the computational cost for one iteration must stay reasonable, so that there is indeed a gain in computation time; in practice, for the methods presented in the following, it scales with the cost of the operation which corresponds to the computational bottleneck in the solver. In our case, this operation is the *matrix-vector product*, such that their number are often used as the measuring standard to evaluate algorithms costs.

- for the same reasons, the total number of iterations should be as small as possible, as this will determine the overall cost of the solver.

In the following, we will present two distinct families of such methods, namely the fixed-point and Krylov ones.

Fixed point methods: the Jacobi example

Fixed point methods (or stationary methods) are built by splitting the matrix to invert: in the case of the polarization, we can write

$$\mathbf{T} = \boldsymbol{\alpha}^{-1} - \mathcal{T} \tag{2.11}$$

changing our linear problem (2.6) into

$$\boldsymbol{\alpha}^{-1}\boldsymbol{\mu} = \mathcal{T}\mu + \mathbf{E} \tag{2.12}$$

the solution then writes

$$\boldsymbol{\mu} = \boldsymbol{\alpha}(\mathbf{E} + \mathcal{T}\mu) \tag{2.13}$$

Here, $\boldsymbol{\mu}$ appears as the fixed point of a mapping. Starting from an initial guess $\boldsymbol{\mu}_0$, and computing the sequence of $\boldsymbol{\mu}_n$ defined as

$$\boldsymbol{\mu}_{n+1} = \boldsymbol{\alpha}(\mathbf{E} + \mathcal{T}\boldsymbol{\mu}_n) = \boldsymbol{\mu}_n + \boldsymbol{\alpha}\mathbf{r}_n \tag{2.14}$$

Picard's fixed point theorem[12] shows us that this sequence converges towards the solution of our linear problem. This is however only insured if the spectral radius of matrix $\boldsymbol{\alpha}\mathcal{T}$, noted $\rho(\boldsymbol{\alpha}\mathcal{T})$ is smaller than 1, which means that some cases may *not converge*.[13, 14]

This method thus illustrates itself through its simplicity, both in derivation and final algorithmic shape. The non-convergence risk, however, advocates for a more careful choice of the inversion algorithm. The properties of the polarization matrix also encourage us in looking for a method that would be more specific, and hopefully more efficient, to solve our linear problem.

Jacobi over Relaxation

An extra step can be performed at the end of an iterative solver's algorithm: by choosing a relaxation parameter ω, one can refine the Jacobi method as follows

$$\boldsymbol{\mu}_{n+1} = (1-\omega)\boldsymbol{\mu}_n + \omega(\boldsymbol{\mu}_n + \boldsymbol{\alpha}\mathbf{r}_n) = \boldsymbol{\mu}_n + \omega\boldsymbol{\alpha}\mathbf{r}_n \tag{2.15}$$

It supposes no extra cost if both $\boldsymbol{\mu}_n$ and $\mathbf{r}_n$ have already been computed, and yields an improved set of induced dipoles. It is convergent if $\rho(I_d - \omega\boldsymbol{\alpha}\mathbf{T}) < 1$. This is called the Jacobi over relaxation scheme, usually abbreviated as JOR. It may be observed that setting $\omega = 1$, the JOR method comes back to a simple Jacobi step. Asymptotically (*i.e.* for a large number of iterations), the optimal value for ω is the value that minimizes the spectral radius $\rho(I_d - \omega\boldsymbol{\alpha}\mathbf{T})$. In our symmetric positive definite case,it corresponds to

$$\omega_{\text{opt}} = \frac{2}{\lambda_{\text{min}} + \lambda_{\text{max}}} \tag{2.16}$$

where λ_{min} and λ_{max} are the smallest and biggest eigenvalue of matrix $\boldsymbol{\alpha}\mathbf{T}$. This value can be computed efficiently using Lanczos algorithm.[15]

ExPT and OPT*n* methods: a refitted Jacobi procedure

Recently, Simmonett *et al.*[16, 17] developed a new class of methods called ExPT and then OPTn, based on a perturbative approach of the induced dipole derivation. We can show that this is in fact a reexpression of the Jacobi iterations presented above.

Indeed, let us express the induced dipoles obtained by Jacobi iterations as a sum of dipoles of increasing order:

$$\boldsymbol{\mu}_n = \sum_{i=0}^{n} \boldsymbol{\mu}_{(i)} \tag{2.17}$$

Taking the direct field $\boldsymbol{\alpha}\mathbf{E}$ as the zero-th order $\boldsymbol{\mu}_{(0)} = \boldsymbol{\mu}_0$, applying 2.14 that the first order gives

$$\boldsymbol{\mu}_1 = \boldsymbol{\alpha}\mathbf{E} + \boldsymbol{\alpha}\mathcal{T}\boldsymbol{\alpha}\mathbf{E} \tag{2.18}$$

The second order term $\boldsymbol{\mu}_{(1)}$ is thus defined as $\boldsymbol{\alpha}\mathcal{T}\boldsymbol{\alpha}\mathbf{E}$, and pushing this derivation further shows the following expression for the i-th order term

$$\boldsymbol{\mu}_{(i)} = \boldsymbol{\alpha}\left(\boldsymbol{\alpha}\mathcal{T}\right)^{i}\mathbf{E} \tag{2.19}$$

By truncating this method after two iterations ($n = 3$), and using a linear combination of the odd dipoles, one defines the ExPT method:[16]

$$\boldsymbol{\mu}_{\text{ExPT}} = c_1\boldsymbol{\mu}_1 + c_3\boldsymbol{\mu}_3 \tag{2.20}$$

Interestingly, this expression is not iterative, and thus provides the user with an analytical expression of the dipoles. It means that differentiating the dipoles with respect to space would give simple, but analytical derivatives, that can then be used to improve the precision over the simulations, as will be shown in 2.2. It also exhibits a quite small computational cost (as it boils down to three matrix-vector products).

Nevertheless, the coefficients c_1 and c_3 are determined through a fitting procedure, and thus should be subject to cautious use, as they may yield wrong results for untested systems, since (as explained previously) Jacobi iterations are not necessary convergent.

This approach was then broadened to further orders in the follow-up OPTn methods,[17] encompassing $\boldsymbol{\mu}_{(i)}$ up to the fourth order. Improved results were obtained, particularly on heterogeneous and biological systems, but a fitting procedure remains nevertheless necessary here.

As for the extended Lagrangian methods evoked before, OPTn methods can count on analytical derivatives, which is a very good asset regarding the energy conservation along the simulation. However, they also share the same drawback: they both rely on parametric formulations (for the fictitious masses in the extended Lagrangian, and for the c_i coefficients here), which can sometimes lead to inconsistent energy values. Further efforts are thus needed to keep the good features (analyticity) while keeping a non-empirical concept.

Direct Inversion in the Iterative Subspace

The Direct Inversion in Iterative Subspace (DIIS) procedure was introduced by Pulay[18, 19] to improve the convergence rate of self-consistent calculations used in quantum mechanics. The method assumes that a good estimation of the final solution $\boldsymbol{\mu}_f$ can be expressed as a linear combination of m previous iterative solutions. This can be written

$$\boldsymbol{\mu}_{\text{DIIS}} = \sum_{i=1}^{m} \boldsymbol{\mu}_i = \sum_{i=1}^{m} c_i \left(\boldsymbol{\mu}_f + \mathbf{e}_i\right) = \boldsymbol{\mu}_f \sum_{i=1}^{m} c_i + \sum_{i=1}^{m} c_i \mathbf{e}_i \tag{2.21}$$

using $\mathbf{e}_i$ to denote the difference $\boldsymbol{\mu}_f - \boldsymbol{\mu}_i$, which is unknown, with $\sum_{i=1}^{m} c_i = 1$. This last condition can be used as a constraint within a Lagrange multiplier method in order to minimize the norm of the error $\sum_i c_i \mathbf{e}_i$. This allows one to extract a good set of coefficients c_i that can be used two generate the estimate $\boldsymbol{\mu}_{\text{DIIS}}$.

This is a very general method, and it can be used on any iterative self-consistent solver. Practically, one would simply compute an iteration of the chosen solver, then carry out the DIIS procedure (whose cost is negligible compared to a solver iteration). If the obtained $\boldsymbol{\mu}_{\text{DIIS}}$ reaches the chosen convergence criterion, self-consistent iterations can be stopped.

When used in a linear case, DIIS corresponds to the GMRES subfamily of the Krylov methods.[20]

Krylov subspace methods and the Conjugate Gradient

Krylov methods follow a different idea: with each successive iteration, a subset of space (called Krylov subspace) grows, and a functional is minimized over that subspace.

Given an initial guess for the dipoles $\boldsymbol{\mu}_0$ (or more generally of the vector solution of 2.6), let us define its associated residual $\mathbf{r}_0$ as

$$\mathbf{r}_0 = \mathbf{E} - \mathbf{T}\boldsymbol{\mu}_0 \tag{2.22}$$

The Krylov subspace of order p is defined as $\text{span}\{\mathbf{r}_0, \mathbf{T}\mathbf{r}_0, \mathbf{T}^2\mathbf{r}_0, \ldots, \mathbf{T}^{p-1}\mathbf{r}_0\}$.

Since the dimension of the Krylov susbpace grows with each iteration, minimization is performed over

an always increasing domain, which ensures that the method will converge to the exact solution $\boldsymbol{\mu}$, in a number of iterations $n_i \leq 3N$.

The functional that is chosen to be minimized determines which Krylov method is used: if one were to minimize the l^2-norm of the residue ($||\mathbf{r}_n||_{l^2}$), the method used would be GMRES.[15] If E_{pol} is minimized (provided that $\mathbf{T}$ is symmetric positive and definite), it's the *Conjugate Gradient* (CG). The Conjugate Gradient is optimal, in the sense that the dipoles values at iteration i are fully minimized over the Krylov subspace on which they were built. Otherwise put, no other vector built on the Krylov subspace that is used at iteration i would give a lower value of the polarization energy (or would be closer to the exactly minimized dipoles $\boldsymbol{\mu}_f$). For matrices veryfing less strong conditions than our $\mathbf{T}$ matrix, several other methods have been proposed, such as the BiCG or Minres algorithms (see [15]).

The Conjugate Gradient works with two distinct vectors, both belonging to the iteratively growing Krylov subspace: the descent direction $\mathbf{p}_i$ and the residual $\mathbf{r}_i$. The descent direction represents the research vector along which the (iterative) solution is updated, and the residual is $\mathbf{T}$-orthogonal[iii] to the descent direction of previous iteration, thus allows the building of the next research subspace (see [13]).

Its initialization reads as follows:

$$\begin{cases} \boldsymbol{\mu}_0 = \boldsymbol{\alpha}\mathbf{E} \text{ or } 0 \\ \mathbf{r}_0 = \mathbf{E} - \mathbf{T}\boldsymbol{\mu}_0 \\ \mathbf{p}_0 = \mathbf{r}_0 \end{cases} \tag{2.23}$$

One can note that this initialization costs (in computational time) one matrix-vector product if a non-zero guess is used.

[iii] Let A denote a matrix. Matrix-orthogonality does not differ much, in its definition, from the vector orthogonality. Without overburdening the reader with details, we will simply note that, for two vectors u and v to be A-orthogonal, the scalar product $\langle u, Av \rangle$ must be equal to zero.

One iteration of the algorithm is then:

$$\begin{cases} \gamma_i = \frac{\langle \mathbf{r}_i, \mathbf{r}_i \rangle}{\langle \mathbf{p}_i, \mathbf{T}\mathbf{p}_i \rangle} \\ \boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i + \gamma_i \mathbf{p}_i \\ \mathbf{r}_{i+1} = \mathbf{r}_i - \gamma_i \mathbf{T}\mathbf{p}_i \\ \beta_{i+1} = \frac{\langle \mathbf{r}_{i+1}, \mathbf{r}_{i+1} \rangle}{\langle \mathbf{r}_i, \mathbf{r}_i \rangle} \\ \mathbf{p}_{i+1} = \mathbf{r}_{i+1} + \beta_{i+1} \mathbf{p}_i \end{cases} \tag{2.24}$$

$\langle \mathbf{u}, \mathbf{v} \rangle$ designates the scalar product between two vectors u and v. At iteration i,

- γ_i, the optimal distance to move along the search direction $\mathbf{p}_i$ is computed,
- the induced dipoles are updated ($\boldsymbol{\mu}_{i+1}$),
- the residual is computed, effectively increasing the Krylov subspace size,
- the next search direction is computed.

The costly part in terms of computation is the matrix-vector product ($\mathbf{T}\mathbf{p}_i$) necessary for each iteration.

As the optimal iterative solver when considering symmetric positive definite matrices, and given its guaranteed convergence, the Conjugate Gradient appears as the method of choice for treating our polarization problem.

2.1.4 Reach convergence faster

After choosing the Conjugate Gradient, since it is optimal for our problem, its convergence properties can be improved in several manners. This may seem counter-intuitive: if CG is optimal, how could we hope to improve it ?

The optimality of CG means that, for a given number of iterations, no other solver could yield a better result. Put in other words, to minimize the number of iterations required to reach a convergence

criterion, CG is the best choice. Further improving its performances can thus only be done by acting on the "input" of the algorithm: two vectors (**E** and $\boldsymbol{\mu}_0$) and the polarization matrix **T** (see eq. 2.23). Of course, the computational cost for this better "preparation" must stay negligible compared to the subsequent acceleration gained.

In the following, we describe two such leads, one by acting on the matrix to be inverted, the second by choosing a good starting point for the solver.

Preconditioning

When considering Krylov subspaces methods, one can show[15] that the convergence rate of an iterative solver depends on the condition number of the matrix to be inverted. This condition number κ is, in our symmetric, positive, definite case, is defined as

$$\kappa(\mathbf{T}) = \frac{\lambda_{\text{max}}}{\lambda_{\text{min}}} \tag{2.25}$$

where λ_{max} and λ_{min} are the biggest and smallest eigenvalues, respectively. The smaller κ is, the better the matrix is conditioned, and the faster the iterative solver will work.

Suppose one has access to a matrix **P** whose inverse is somewhat "close" to the matrix **T**. Then the conditioning of **PT** will be better than the one of **T** (we have $\kappa(\mathbf{P}^{-1}\mathbf{T}) \leq \kappa(\mathbf{T})$). One can then change the original linear problem $\mathbf{T}\boldsymbol{\mu} = \mathbf{E}$ into the better conditioned one

$$\mathbf{P}^{-1}\mathbf{T}\boldsymbol{\mu} = \mathbf{P}^{-1}\mathbf{E} \tag{2.26}$$

Using our iterative solver on this new problem will then yield a faster convergence rate, thus accelerating the dynamics.

The only drawback here is the **P** matrix itself: it has to be both close to the **T** matrix and easily inverted. An ideal candidate here is the block-diagonal matrix of inverse polarizabilities (usually noted

$\boldsymbol{\alpha}^{-1}$)

$$\boldsymbol{\alpha}^{-1} = \begin{pmatrix} \alpha_1^{-1} & & (0) \\ & \alpha_2^{-1} & \\ & & \ddots \\ (0) & & \alpha_N^{-1} \end{pmatrix} \tag{2.27}$$

Inverting this matrix is indeed trivial and negligible in cost. However, it is a really rough approximation of the original **T** matrix and its effect will be subsequently limited.

Among the other preconditioner candidates, Wang and Skeel[21] propose the following: starting from

$$\mathbf{T} = (\boldsymbol{\alpha}^{-1} - \mathcal{T}) = \boldsymbol{\alpha}^{-1}(\mathbf{I} - \boldsymbol{\alpha}\mathcal{T}) \tag{2.28}$$

one has

$$\mathbf{T}^{-1} = \boldsymbol{\alpha}(\mathbf{I} - \boldsymbol{\alpha}\mathcal{T})^{-1} \tag{2.29}$$

which can be approximated as

$$\mathbf{T}^{-1} \simeq \boldsymbol{\alpha}(\mathbf{I} + \boldsymbol{\alpha}\mathcal{T}) \tag{2.30}$$

This approximation can be combined with a cutoff on the interactions in the $\mathcal{T}$ matrix. Being closer than the previous matrix **P** presented to **T**, the effect on the convergence rate are better, but come with a higher cost for each iteration (see [22]).

Other names, such as the incomplete Cholesky preconditioner, can be cited here. The "pure" Cholesky decomposition designates the rewriting of a positive-definite matrix as product of a triangular matrix and its transpose ($\mathbf{M} = \mathbf{L}\mathbf{L}^T$), allowing for an efficient direct solution of linear systems. The *incomplete* one is an approximation of this decomposition that can be used as a preconditioner.[15]

Divide and Conquer

Nocito and Beran proposed a good block-Jacobi preconditioner based on a divide and conquer strategy[23] (see [15] for a description of the block-Jacobi method). One first breaks the polarization matrix in spatial blocks (or subclusters), following the expectation that atoms will polarize their closest coun-

terparts the strongest. This is done through the K-means,[24] a method that concentrates the largest matrix elements along the diagonal, based on a distance criterion. Provided that the subclusters are small enough, or equivalently that the breakdown is sufficiently fine, one can solve the polarization equations within each subcluster using direct methods (see the **Direct methods** subsection in 2.1.2).

The remaining interactions, between the induced dipoles in different subclusters, are computed in an iterative fashion using block-Jacobi iterations coupled with DIIS: the Divide and Conquer algorithm can thus be seen as a Jacobi method with a very efficient preconditioner.

This method was implemented within the Tinker-HP code and showed its good applicability to parallel computations,[25] allowing for acceleration of the simulations.

Using a guess

Accelerated convergence can also be obtained by considering the starting point of the algorithm: if one knows an estimate of the induced dipoles prior to the computation, one can use it as a beginning point for the solver, and the convergence criterion should be reached in a lower number of iterations. This would effectively speed the computation. Simply put, the guess is a way to start the computation closer to the final solution. As for the preconditioner however, computing the guess should remain a negligible cost compared to the actual iterations of the algorithm.

A first possible choice would be to use the first order induced dipoles, that is, the one given by the polarizability and the external electric field with no dipole-dipole interaction

$$\boldsymbol{\mu}_0 = \boldsymbol{\alpha}\mathbf{E} \tag{2.31}$$

More involved guesses have been proposed, amongst which the predictor guesses. Predictors use the information from previous iterations (*i.e.* the induced dipoles obtained at earlier time-steps) to build more stable and efficient guesses. The simplest idea here would be to use the dipoles obtained from previous iteration, but it has been shown[1] that it requires tight convergence to allow stable simulations.

Kolafa *et al.* proposed a systematically improvable predictor[6] exhibiting good stability and gain in computational time when used as a guess.[26] For the memory demands to stay reasonable, Kolafa suggested a sixth order predictor, and it is thus the one implemented in Tinker-HP. More recently, Nocito

and Beran showed that extending this up to the sixteenth order yielded substantial acceleration (see [27]), while the overhead and memory usage were not problematic.

Such approaches are very appealing, as speedups reaching 50% can be achieved for 1 and 2 fs timesteps. The use of memory in the dynamics integration however prevents proper time-reversibility and phase-space volume conservation.[26, 22] The ASPC also exhibits problems regarding volume preservation,[21] which can affect specific types of dynamics. Moreover, the improvements do not apply for timesteps larger than 2 fs, although this is a likely situation when considering RESPA integration schemes (as will be detailed in chap. 4).

2.1.5 The need for improved algorithms

Out of all the algorithms that were described so far, none verifies simultaneously all the properties we could hope for. We are looking for a polarization solver that would produce accurate polarization energies, conserve the energy, but also be computationally efficient.

Indeed, after choosing the most adapted solver for our linear problem, and despite the auxiliary efforts to improve it, two important drawbacks still undermine the polarizable molecular dynamics.

Firstly, the computational speed: although many refinements are used to improve the convergence rate, polarizable molecular dynamics using induced dipoles are still slower than their non-polarizable counterparts. As a rule of thumb, one can consider that polarizatble molecular dynamics simulations are around ten times slower.

The algorithm performances are of course strongly dependent on the level of accuracy requested. As usual when talking about simulations, a compromise has to be found between the computation precision and its cost (or equivalently, the time required to perform them). Lipparini *et al.* showed[1] that the convergence criterion should be carefully chosen (*e.g.* with an error threshold of 10^{-6} D) to ensure energy conservation. In the most conservative choices, the number of matrix-vector products needed to compute the induced dipoles can reach a dozen !

Secondly, the energy drift: let us recall that in molecular dynamics, the forces are defined as the (spatial) gradients of the energies. In particular, when considering the polarization energy

$$\vec{F}_{\text{pol}} = -\vec{\nabla} E_{\text{pol}} \tag{2.32}$$

Using a simple chain rule, this differentiation can be explicited as a sum over several terms. Let us consider a single component of the gradient to simplify notations:

$$\frac{dE_{\text{pol}}}{dr_i^\alpha} = \frac{\partial E_{\text{pol}}}{\partial r_i^\alpha} + \frac{\partial E_{\text{pol}}}{\partial \boldsymbol{\mu}} \frac{\partial \boldsymbol{\mu}}{\partial r_i^\alpha} \tag{2.33}$$

The second term in this equation involves partial derivatives of the induced dipoles. Since these are assumed to be converged to the value minimizing the energy, one could expect $\frac{\partial E_{\text{pol}}}{\partial \mu}$ to be equal to zero. This assumption is made in any polarizable molecular dynamics code, considering that the convergence criterion chosen is enough to ensure that property.

It is however not true, since the induced dipoles are computed using an iterative algorithm, not perfectly converged. As a consequence, a small discrepancy exists between the polarization energy and the forces arising from it in most polarizable simulations. This error will cumulate as a *drift* over time, and can lead to unstable simulation if carried for too long (see [1]). The only possible answer is then to choose a tighter convergence criterion on the polarization energy, which will reverberate on the computation time itself.

This section gave the reader a quick overview of the treatment of induced polarization in contemporary simulations. Despite the various methods, algorithms, refinements available, one conclusion still painfully hinders polarizable molecular dynamics: the cost of simulations is still high.

In the following section, we will present a new algorithm to tackle the problems evoked above.

2.2 Truncated Conjugate Gradient: a new polarization solver

Knowing the limits of the common polarization solvers, a truncation of the well-known Conjugate Gradient algorithm was proposed, introducing the *Truncated Conjugate Gradient* (TCG).[22] In this first part, the reader will find a presentation of the algorithm and its main features. Simulation results will be presented in 2.3.

Truncating the algorithm means that instead of verifying, at each iteration, the value of a numerical quantity (energy, norm of the residual...), the total number of iterations to be performed is fixed before-

hand. The polarization solver thus becomes non-iterative, and gains advantages as described below. Moreover, as any Krylov method, this builds a class of method that are systematically improvable.

2.2.1 Simulation stability

Using this method, the expression for the induced dipoles becomes analytical, only depending on the input vectors (the guess $\boldsymbol{\mu}_0$ and the residual $\mathbf{r}_0$). Let us write n_{TCG} the truncation order, then for $n_{\text{TCG}} = 1$, we have:

$$\boldsymbol{\mu}_{\text{TCG1}} = \boldsymbol{\mu}_0 + \frac{\langle \mathbf{r}_0, \mathbf{r}_0 \rangle}{\langle \mathbf{r}_0, \mathbf{T}\mathbf{r}_0 \rangle} \mathbf{r}_0 \tag{2.34}$$

The induced dipoles are now expressed as a linear combination of the Krylov basis vectors ($\mathbf{r}_0$, $\mathbf{T}\mathbf{r}_0$...), with scalar coefficients involving their scalar products. When n_{TCG} increases, this formula becomes more and more complex, and some notations were adopted "on the fly" during the derivation. To avoid overburdening this section, they are reproduced in the appendix.

$$\boldsymbol{\mu}_{\text{TCG1}} = \boldsymbol{\mu}_0 + t_4 \mathbf{r}_0 \tag{2.35}$$

$$\boldsymbol{\mu}_{\text{TCG2}} = \boldsymbol{\mu}_0 + (\gamma_1 t_2 + t_4)\mathbf{r}_0 - \gamma_1 t_4 \mathbf{P}_1 \tag{2.36}$$

$$\boldsymbol{\mu}_{\text{TCG3}} = \boldsymbol{\mu}_0 + (t_4 + \gamma_1 t_2 + \gamma_2 + \gamma_2\beta_2 t_2)\boldsymbol{r}_0 - (\gamma_1 t_4 + \gamma_2 t_4 + \gamma_2\beta_2 t_4)\mathbf{P}_1 - \gamma_1\gamma_2 \mathbf{P}_2 \tag{2.37}$$

Having access to the induced dipole analytical formula, the polarization energy

$$E_{\text{pol, TCG}n} = -\frac{1}{2}\langle \boldsymbol{\mu}_{\text{TCG}n}, \mathbf{E} \rangle \tag{2.38}$$

can thus also be expressed *analytically*, and more importantly, so does its gradient. This means that using the TCG, one has access to gradients – and subsequently, forces – that are exactly consistent with the computed induced dipoles, thus erasing the drift shown in section 2.1.5, and one can count on excellent energy conservation (this will be illustrated in the numerical results section). The derivation of these gradients is quite cumbersome and requires careful implementation to stay efficient, as will be discussed in section 2.2.4.

It is however important to note that this definition of the polarization energy corresponds to the

variational minimum of the functional $E_{\text{pol}}[\boldsymbol{\mu}] = \frac{1}{2}\langle \boldsymbol{\mu}, \mathbf{T}\boldsymbol{\mu} \rangle - \langle \boldsymbol{\mu}, \mathbf{E} \rangle$ (equation 2.4). Nevertheless, in our case, our induced dipole vector is not fully converged, since we use iterative methods, and we are thus using a slightly different definition of the polarization energy when using equation 2.38 above. This does not revoke the analytical consistency of our derivatives, and the claim on the energy conservation remains true. In fact, the difference between both polarization energy definitions is small if the dipoles are close to the variational minimum, a condition that should hold in our simulations. In addition, and from a more practical point of view, using the pure variational functional would yield very involved expressions. The missing term $\frac{1}{2}\langle \boldsymbol{\mu}, \mathbf{T}\boldsymbol{\mu} \rangle$ would be extremely cumbersome to differentiate, and even worse, would require an extra matrix-vector product, which would be the exact opposite of our objective. In practice, this additional effort would not bring any substantial improvement to the method.

2.2.2 Computational cost

The second main advantage the TCG gives us is a full control over the computational effort devoted to polarization. Indeed, by choosing a truncation order, one limits the number of matrix-vector products performed. Ideally, a small number of iterations would be sufficient to yield satisfying results, but we also know that a correctly converged Conjugate Gradient could require around 10 iterations. A correct middle ground has to be found – as it is always the case in numerical simulations – between precision and efficiency. We show in section 2.3 that a very limited n_{TCG} (one or two) is enough to produce very satisfying results, partially thanks to the refinement of the methods presented below.

In terms of parallel implementation, the method is equivalent to n iterations of the Conjugate Gradient solver, we will thus be able to reuse the machinery already developed and optimized to compute our truncation, and its associated gradients.

2.2.3 Refinements of the solver

Now that the iteration number is fixed, the objective is to improve the quality of the computed induced dipoles while respecting the fixed computational cost: increasing the truncation order would have the same effect as selecting a tighter convergence criterion for the Conjugate Gradient, but here our goal remains to accelerate dynamics, so we will refrain ourselves from this solution.

Preconditioner and guess

Exactly like in the Conjugate Gradient's case, using a better starting point for the solver or improving the problem's conditioning before starting the resolution are two good leads to improve the results. One could note that in this new situation, we won't reduce the number of iterations, as it is already fixed: we now want to improve the accuracy of the solution obtained with this $n_{\text{TCG}n}$ iterations.

The two solutions evoked earlier (2.1.4) thus still apply here: preconditioning the polarization matrix, as well as using an initial guess, will improve the precision on the final set of induced dipoles. However, one should keep in mind that the analytical formulae promised by the TCG come with a complexity cost. Let us imagine using a guess based on previous time-steps (such as the ASPC). Every vector quantity from the previous set of dipoles should be stored, and linearly combined. Although this should not be a problem memory-wise, the gradients will quickly become a nightmare to compute at each time-step. The same remark can be done on the advanced preconditioners, where any spatial dependency (as is it the case for Cholesky preconditioners) would imply heavy expressions for the gradients.

The choice to use a guess also implies that an extra matrix-vector product will be required to compute the initial residue, as seen in the initialization equations (2.23), since we have:

$$\mathbf{r}_0 = \mathbf{E} - \mathbf{T}\boldsymbol{\mu}_0 \tag{2.39}$$

Peek-step

Wang and Skeel,[21] noting that computing the initial residue costs an extra matrix-vector product when using a guess, proposed a method to save that extra cost using a *peek step* (also known as Picard step). As explained earlier, at each iteration i, the descent direction that will be followed at iteration $i+1$ is computed. Calculating the exact distance that should be travelled along this direction would require an extra matrix-vector product, but one can do a supplementary fixed-step iteration:

$$\boldsymbol{\mu}_{\text{TCG, peek},n} = \boldsymbol{\mu}_{\text{TCG}n} + \omega\boldsymbol{\alpha}\mathbf{r}_{n+1} = \boldsymbol{\mu}_{\text{TCG}n} + \omega\boldsymbol{\mu}_{\text{peek},n} \tag{2.40}$$

This is exactly equivalent to performing an extra Jacobi over-relaxation step after the CG iterations. The usefulness of this extra step directly depends on ω, whose choice is not trivial. ω_{opt} (as defined in 2.16) is optimal in the asymptotic limit, that is, when a large number of iterations of the JOR step are

computed. Since this is not our case, one should not expect this ω value to necessarily yield the best possible result. ω_{opt} is tested and compared to other possible ω values in 2.3.

To avoid having to compute the spectrum of the matrix, one can note that ω_{opt}, by improving the convergence on $\boldsymbol{\mu}_n$, would minimize both the RMS error on the induced dipoles, and thus (as a consequence) the polarization energy. Yet one can choose to *fit* ω so that the "peeked" dipoles obtained reproduce as closely as possible the polarization energy. To do so, one first needs to compute the fully converged dipoles (using for example the Conjugate Gradient solver with a very tight convergence criterion). Then, looking at the energy expression

$$E_{\text{pol}} = -\frac{1}{2}\langle \boldsymbol{\mu}_{\text{TCG}n\text{ peek}}, \mathbf{E}\rangle \tag{2.41}$$

$$= -\frac{1}{2}\langle \boldsymbol{\mu}_{\text{TCG}n}, \mathbf{E}\rangle - \frac{\omega}{2}\langle \boldsymbol{\alpha}\mathbf{r}_{n+1}, \mathbf{E}\rangle \tag{2.42}$$

one can simply define ω_{fit} as

$$\omega_{\text{fit}} = -\frac{2E_{\text{pol}}^{(\text{ref})} + \langle \boldsymbol{\mu}_{\text{TCG}n}, \mathbf{E}\rangle}{\langle \boldsymbol{\alpha}\mathbf{r}_{n+1}, \mathbf{E}\rangle} = \frac{\langle \boldsymbol{\mu}_{\text{CG,ref}} - \boldsymbol{\mu}_{\text{TCG}n}, \mathbf{E}\rangle}{\langle \boldsymbol{\alpha}\mathbf{r}_{n+1}, \mathbf{E}\rangle} \tag{2.43}$$

Where $E_{\text{pol}}^{(\text{ref})}$ designates the reference polarization energy obtained when using a tightly converged Conjugate Gradient, and $\boldsymbol{\mu}_{\text{CG,ref}}$ the induced dipoles obtained through this same procedure.

One could note that this can be rewritten as:

$$\omega_{\text{fit}} = \frac{\langle \mathbf{T}^{-1}\mathbf{r}_n, \mathbf{E}\rangle}{\langle \boldsymbol{\alpha}\mathbf{r}_{n+1}, \mathbf{E}\rangle} \tag{2.44}$$

Computing the induced dipoles using a fully, tightly converged Conjugate Gradient comes however at a high cost, and it is precisely what we were trying to avoid. This fitting procedure will hence be carried out every n_{fit} timesteps only. A more detailed study on this fitting will be presented in section 2.3.

A particular case: the orthogonal peek-step

Particular care has to be taken when trying to combine the refinements that were presented in 2.1.4, in particular when trying to use a peek-step with a preconditioner. When using a peek-step, the final

expression for the polarization energy is given by eq. 2.42. However, assuming that we use a diagonal preconditioner, then the vector quantities in TCG are multiplied by $\boldsymbol{\alpha}^{-1}$. This means that one can rewrite $\boldsymbol{\mu}_{\text{peek},n}$ as

$$\boldsymbol{\mu}_{\text{peek},n} = \boldsymbol{\alpha}^{-1}.\boldsymbol{\alpha}\mathbf{r}_{n+1} = \mathbf{r}_{n+1} \tag{2.45}$$

Besides, when using the Conjugate Gradient, the residuals vectors are computed iteratively in such a way that two different residuals vectors $\mathbf{r}_i$ and $\mathbf{r}_j$ are orthogonal:

$$\langle \mathbf{r}_i, \mathbf{r}_j \rangle = 0 \quad \forall j \neq i \tag{2.46}$$

Yet, looking at eq. 2.23, we have

$$\mathbf{r}_0 = \mathbf{E} - \mathbf{T}\boldsymbol{\mu}_0 \tag{2.47}$$

meaning that when no guess $\boldsymbol{\mu}_0$ is used, the initial residual vector $\mathbf{r}_0$ is equal to $\mathbf{E}$. In this case, the polarization energy arising from the peek term is

$$\langle \boldsymbol{\mu}_{\text{peek},n}, \mathbf{E} \rangle = \langle \mathbf{r}_{n+1}, \mathbf{r}_0 \rangle = 0 \tag{2.48}$$

Practically, this has a very simple but important consequence on the choice of a setup for the TCG: if no guess is used, then the peek-step *can not* be used in the computation.

2.2.4 Computation of the forces

As stated in 1.3, the forces driving our simulation are expressed as gradients of the energies, and the polarization terms are no exception. In this section, we will show how a naive implementation of the forces arising from polarization could in fact completely negate our efforts to accelerate the simulation. We then present a strategy to preserve our acceleration during computations.

Keeping the notations adopted earlier, the formal derivative of the induced dipoles for the first two orders of TCG are:

$$\boldsymbol{\mu}'_{\text{TCG1}} = \boldsymbol{\mu}'_0 + t_4\mathbf{r}'_0 + t'_4\mathbf{r}_0 \tag{2.49}$$

$$\boldsymbol{\mu}'_{\text{TCG2}} = \boldsymbol{\mu}'_0 + (t_4 + \gamma_1 t_2)\mathbf{r}'_0 + (t'_4 + \gamma'_1 t_2 + \gamma_1 t'_2)\mathbf{r}_0 + (\gamma'_1 t_4 + \gamma_1 t'_4)\mathbf{P}_1 + \gamma_1 t_4 \mathbf{P}'_1 \tag{2.50}$$

Let us recall that our collective vectors have a $3N$ size. If one were to differentiate one such vector with respect to $3N$ spatial coordinates, it would yield a $3N \times 3N$ matrix. Looking at two arbitrarily chosen atoms with indiced i and j, the force experienced by atom i arising from atom j reads

$$\vec{f_{j/i}} = \begin{pmatrix} -\frac{dE_{ij}}{dr_{ij}^{\alpha}} \\ -\frac{dE_{ij}}{dr_{ij}^{\beta}} \\ -\frac{dE_{ij}}{dr_{ij}^{\delta}} \end{pmatrix} \tag{2.51}$$

where r_{ij}^{α} is the α component of the vector $\vec{r_{ij}} = \vec{r_j} - \vec{r_i}$, and E_{ij} is the energy of interaction between i and j. Firstly, the total force experienced by each atom in the simulation is the sum of all the forces arising from the $(N-1)$ other atoms, or, to put it in other words, of all the $N-1$ derivatives of the interaction energies. With thus have a total of $N \times (N-1)$ forces coming into play (which is equivalent to N^2 for high N). Using Newton's third law, this number can actually be divided in two (we simply have $\vec{f_{i/j}} = -\vec{f_{j/i}}$), but the order of magnitude remains at $O(N^2)$ (when using Smooth Particle Mesh Ewald, this costs drops to $O(N \log N)$). Computing a set of forces is thus already a non-negligible part, with the cost arising from computing all the r_{ij} distances.

Secondly, storing such quantities would be heavy on the memory, and performing any operation (scalar products, combinations...) has to be carefully done, as it could considerably slow down the computations. In particular, using the analytical expressions in 2.49 and 2.50 without reconsideration will repeat the same operations several times, thus slowing down the computation for no good reason.

There is an even worse threat: in the explicit expression of the induced dipoles, differentiation (in order to compute the gradient), terms involving a squared polarization matrix exist (*e.g.* $\mathbf{T}^2\mathbf{r}_0$). Computing the derivative of such a term would involve a term of the shape $\mathbf{TT}'\mathbf{r}_0$ which effectively would boil down to a matrix-matrix product: this would be computationally too expensive.

A careful implementation for computing the forces has thus been proposed in [28]. It is based on a direct computation of the gradients of the energy, storing a minimal number of intermediate quantities, and specifically no intermediate vector or matrix derivative. Implementing such a method supposes a careful bookkeeping of the many terms involved to end up with the smallest computational effort

possible.

Let us consider the derivative of the polarization energy in order to extract general properties that would allow us to make its computation more efficient.

$$E'_{\text{pol, TCG}n} = -\frac{1}{2}\langle \boldsymbol{\mu}'_{\text{TCG}n}, \mathbf{E}\rangle - \frac{1}{2}\langle \boldsymbol{\mu}_{\text{TCG}n}, \mathbf{E}'\rangle \tag{2.52}$$

Each vector in equations 2.49 and 2.50 can be expressed as a combination of simpler vectors which are increasing powers of $\mathbf{T}$ applied to the initial residual $\mathbf{r}_0$, and the scalars (such as γ_1 or t_4) are in fact scalar products of these vectors as well. In other words, it means that every term in equations 2.49 and 2.50 can be expressed using $\mathbf{r}_0$, $\mathbf{T}\mathbf{r}_0$, $\mathbf{T}^2\mathbf{r}_0$, $\mathbf{T}^3\mathbf{r}_0$...

The general shape of these vectors can be written $\mathbf{T}^m\mathbf{r}_0$, with m a positive integer. A formal differentiation of such an expression would yield three types of terms:

- $\mathbf{T}^m\mathbf{r}'_0$,
- $\mathbf{T}'\mathbf{T}^{m-1}\mathbf{r}_0$ (if $m \geq A$)
- $\mathbf{T}^{m-k-1}\mathbf{T}'\mathbf{T}^k\mathbf{r}_0$ (with $m \geq k+1$)

One should recall that the polarization matrix is symmetric. As such, we have $\langle \mathbf{A}, \mathbf{TB}\rangle = \langle \mathbf{TA}, \mathbf{B}\rangle$. This means that scalar products involving $\mathbf{T}^m\mathbf{r}'_0$ can be rewritten in the shape of $\langle \mathbf{r}'_0, \mathbf{T}^m\mathbf{A}\rangle$, which will prove to be useful in the next paragraphs. The same rewriting can be done when looking at terms like $\langle \mathbf{T}^{m-k-1}\mathbf{T}'\mathbf{T}^k\mathbf{r}_0, \mathbf{A}\rangle$, who were initially impossible to compute in reasonable time since they would involve matrix-matrix products: one can express them as $\langle \mathbf{T}'\mathbf{T}^k\mathbf{r}_0, \mathbf{T}^{m-k-1}\mathbf{A}\rangle$.

If one were to develop this equation (using the expressions for $\boldsymbol{\mu}'_{\text{TCG}n}$ presented earlier), one could see that every single term in the overall sum involves a differentiated quantity, within a scalar product, which would either be the electric field itself ($\langle \mathbf{A}, \mathbf{E}'\rangle$) or a differentiated polarization matrix ($\langle \mathbf{A}, \mathbf{T}'\mathbf{B}\rangle$). This is also true when using a guess. Indeed, we explained earlier that for computational efficiency reasons, we would limit ourselves to using direct field as a guess. In this case,

$$\mathbf{r}_0 = \mathbf{E} - \mathbf{T}\boldsymbol{\mu}_0 \tag{2.53}$$

$$= \mathbf{E} - \mathbf{T}\boldsymbol{\alpha}\mathbf{E} \tag{2.54}$$

gives

$$\mathbf{r}'_0 = \mathbf{E}' - \mathbf{T}'\boldsymbol{\alpha}\mathbf{E} - \mathbf{T}\boldsymbol{\alpha}\mathbf{E}' \tag{2.55}$$

Here, as announced, the differentiated terms are either $\mathbf{E}'$ or a polarization matrix $\mathbf{T}'$.

By analogy, we can assimilate these two types of terms to forces, where $\langle \mathbf{A}, \mathbf{E}' \rangle$ would correspond to a force produced by the interaction of a set of dipoles $\mathbf{A}$ with the electric field, and $\langle \mathbf{A}, \mathbf{T}'\mathbf{B} \rangle$ would be the force arising from the interaction between two sets of dipoles $\mathbf{A}$ and $\mathbf{B}$. The computation of such quantities, as explained earlier, is expensive, as one needs the distance between every pair of atoms i-j ($i \neq j$) to compute the interaction. To minimize this cost, every force computation of this shape will be computed in a single double-loop: the $O(N^2)$ (or $O(N \log N)$ in SPME) is thus only counted once.

To minimize the number of operations involving differentiated terms, as their impact on the computation will be important, a simple gathering of terms that should be taken as scalar product with the same differentiated vector has to be done. For example, considering that $\mathbf{V}'$ is a differentiated vector (either $\mathbf{V}' = \mathbf{E}'$ or $\mathbf{V}' = \mathbf{T}'\mathbf{W}$), and $\mathbf{A}$ and $\mathbf{B}$ are two other vectors, if one needs to compute $\langle \mathbf{A}, \mathbf{V}' \rangle + \langle \mathbf{B}, \mathbf{V}' \rangle$, it is much more efficient to prepare a third vector $\mathbf{C} = \mathbf{A} + \mathbf{B}$, and then to perform the scalar product $\langle \mathbf{C}, \mathbf{V}' \rangle$. The idea is not complicated, but implies quite involved expressions that the reader can find in the Appendix. It could be noted that the choice to use a guess or a peek-step in the computation, as it adds terms to the final induced dipoles expressions, complicates these formula even further.

Ultimately, three strategies were followed to ensure an efficient implementation of the gradients calculations:

- over-expensive terms were avoided through smart scalar products and thanks to the symmetry of the polarization matrix,
- the most expensive ($O(N^2)$) operations were grouped an performed in a single loop,
- scalar products with differentiated vectors were organized so as to minimize the number of times they needed to be called.

To illustrate this strategy, we reproduce here the expression of the derivative of the first-order TCG polarization energy.

$$E'_{\text{TCG1}} = -\frac{1}{2}\left(\langle \mathbf{r}'_0, a^{(1)}_{1,0}\mathbf{E} + a^{(1)}_{1,1}\mathbf{r}_0 + a^{(1)}_{1,2}\mathbf{T}\mathbf{r}_0\rangle + \langle \mathbf{T}'\mathbf{r}_0, a^{(1)}_{2,1}\mathbf{r}_0\rangle\right) \tag{2.56}$$

- $a^{(1)}_{1,0} = t_4$
- $a^{(1)}_{1,1} = \frac{2sp_0}{t_1} + t_4$
- $a^{(1)}_{1,2} = -\frac{2sp_0n_0}{t_1^2}$
- $a^{(1)}_{2,1} = -\frac{sp_0n_0}{t_1^2}$

The expressions accounting for further orders, as well as the peek-step ones, can be found in the appendix.

Bearing in mind that future implementations of the method could pose the same problems to anyone interested, and that this bookkeeping task of bringing every terms together in a proper way should not be done over and over again by other innocent coders, this somewhat involved process was summed up in an article of the Journal of Chemical Physics reproduced hereafter.

The truncated conjugate gradient (TCG), a non-iterative/fixed-cost strategy for computing polarization in molecular dynamics: Fast evaluation of analytical forces

Félix Aviat,[1] Louis Lagardère,[1,2,a)] and Jean-Philip Piquemal[1,3,4,a)]
[1]*Laboratoire de Chimie Théorique, Sorbonne Universités, UPMC Université Paris 06, UMR 7616, F-75005 Paris, France*
[2]*Institut des Sciences du Calcul et des Données, Sorbonne Universités, UPMC Université Paris 06, F-75005 Paris, France*
[3]*Department of Biomedical Engineering, The University of Texas at Austin, Austin, Texas 78712, USA*
[4]*Institut Universitaire de France, Paris Cedex 05 75231, France*

(Received 1 June 2017; accepted 1 August 2017; published online 29 August 2017)

In a recent paper [F. Aviat *et al.*, J. Chem. Theory Comput. **13**, 180–190 (2017)], we proposed the Truncated Conjugate Gradient (TCG) approach to compute the polarization energy and forces in polarizable molecular simulations. The method consists in truncating the conjugate gradient algorithm at a fixed predetermined order leading to a fixed computational cost and can thus be considered "non-iterative." This gives the possibility to derive analytical forces avoiding the usual energy conservation (i.e., drifts) issues occurring with iterative approaches. A key point concerns the evaluation of the analytical gradients, which is more complex than that with a usual solver. In this paper, after reviewing the present state of the art of polarization solvers, we detail a viable strategy for the efficient implementation of the TCG calculation. The complete cost of the approach is then measured as it is tested using a multi-time step scheme and compared to timings using usual iterative approaches. We show that the TCG methods are more efficient than traditional techniques, making it a method of choice for future long molecular dynamics simulations using polarizable force fields where energy conservation matters. We detail the various steps required for the implementation of the complete method by software developers. *Published by AIP Publishing.* [http://dx.doi.org/10.1063/1.4985911]

INTRODUCTION

Polarizable force field simulations using point dipole models are not slow anymore. Indeed, in recent years, the computational cost of the explicit evaluation of the many-body polarization energy and associated forces has been significantly reduced using state of the art mathematical techniques. More precisely, the bottleneck of such approaches is the mandatory resolution of a large set of linear equations (i.e., requiring a matrix inversion) whose size depends on the number of polarizable sites, which is very large in practice (for example, up to several tens of thousands of atoms for medium sized proteins in water). Therefore, direct matrix inversion approaches are unfeasible, and one has to resort to iterative methods[1] such as the Preconditioned Conjugate Gradient (PCG) or the Jacobi/Direct Inversion of the Iterative Subspace (JI/DIIS). Both methods have the advantages to ensure convergence and to be compatible with a massively parallel implementation[2] coupled to Smooth Particle Mesh Ewald (SPME),[3] enabling the possibility to tackle large systems of interest that range from materials to biophysics. However, iterative techniques have to address two aspects simultaneously: a low computational cost and a high accuracy on both energy *and* forces. But the standard way of computing the forces assumes that the dipoles are fully converged and thus these forces are not the exact opposite of the gradient of the polarization energy. This means that to avoid energy drifts, users have to enforce the quality of the non-analytical forces by choosing a tighter convergence criterion of 10^{-5}–10^{-8} D for the dipoles, leading to a strong increase in the number of iterations required to reach convergence. This degrades the computational efficiency of the solvers, limiting the use of molecular dynamics with polarizable force fields. In that context, several strategies have been explored to prevent this drift while ensuring accurate results and a low computational overhead.

In this paper, we review the present status of the polarization solvers before introducing the truncated conjugate gradient (TCG), a method presented in Ref. 4 to propose an efficient solution to these challenges. We then address the issue of the fast computation of the analytical gradients for TCG by presenting a general way to formulate the TCG polarization forces. Analytical formulas are given for the TCG1 and the TCG2 methods, as well as for their refinements with the use of a preconditioner and peek steps.[4] Indeed as a preconditioner improves the convergence of the polarization computation, a peek step allows us to perform an additional but inexpensive Jacobi/Picard pseudo-iteration that does not require any matrix-vector product as it uses the available residual obtained from the TCG process. Finally,

a)Authors to whom correspondence should be addressed: louis.lagardere@upmc.fr and jpp@lct.jussieu.fr

timings to compute these forces in a production context of a reversible reference system propagator algorithm (RESPA) integrator are given and compared to the ones obtained with standard iterative solvers and different levels of convergence as well as different predictor guesses for these solvers.

POLARIZATION SOLVERS: PRESENT STATUS

Several iterative solvers applied to the polarization equations have been presented and tested, such as the Jacobi Over Relaxation (JOR) method, the (preconditioned) conjugate gradient method, the Jacobi/DIIS method (see Refs. 1 and 2), or the recently introduced potentially faster divide and conquer block-Jacobi/DIIS method.[5]

Considering an iterative solver, several techniques can be used to reduce the computational cost to reach convergence by reducing the number of necessary iterations. In the context of Krylov methods such as the conjugate gradient, it is, for example, possible to use a preconditioner. It consists in choosing a matrix $\mathbf{P}$ such that $\mathbf{P}^{-1}$ is close to $\mathbf{T}^{-1}$ (where $\mathbf{T}$ is the polarization matrix to be inverted, presented in the section titled TCG: Notations) and in applying the iterative method to the modified linear system where the matrix and the right hand side are multiplied by $\mathbf{P}^{-1}$. The convergence of the solver is then accelerated because of the clustering of the eigenvalues of the matrix $\mathbf{P}^{-1}\mathbf{T}$. Efficient preconditioners for the polarization equations have been designed, such as the ones proposed by Wang and Skeel,[6] which provide a reduction in the number of iterations to reach convergence up to 10%–20%, depending on the system (i.e., on the condition number of the matrix that one needs to invert).

Another way to improve convergence of an iterative solver is to choose an initial "predictor" guess as close as possible to the actual solution of the linear equations. This guess can be constructed using information from one or a few of the past values of the dipoles. The most naive way to do so is to choose the value of the dipoles at the previous time step (previous guess) but more elaborate and efficient strategies have been designed, such as Kolafa's Always Stable Predictor Corrector (ASPC)[7] or Skeel's Least Square Predictor Corrector (LSPC),[6] which can reduce the number of iterations required to reach convergence up to a factor two in a standard production context.[1] Nevertheless, these two ways to construct initial guesses lose their efficiency when one uses larger time steps, as it the case with the RESPA (Reversible reference System Propagator Algorithm) multiple time step integrator[8] (instabilities occur when such predictors are used with time steps larger than 2 fs).

Note that the two refinements (preconditioning and choosing the initial guess of the solver wisely) can be coupled without problem.

In the same spirit, it is also possible to speed up convergence by introducing an extended Lagrangian scheme to propagate a set of dipoles that are used as initial guess to standard iterative solvers (iEL/SCF or Extended Lagrangian Self-Consistent Field, see Ref. 9). This approach, derived from *ab initio* MD,[10,11] significantly reduces the number of iterations of the solver (by the same order of magnitude as the ASPC predictor) but requires using an additional thermostat in order to prevent energy flows between the degrees of freedom.

However, whatever the different speedup strategies applied to the popular iterative production methods such as PCG or JI/DIIS, they still suffer from an important drawback in link to the way the associated forces are computed. Indeed, they do not address the polarization energy drifting issues that will be encountered in long simulations of large non-homogeneous complexes, such as proteins in water or highly charged ionic liquids. In such a case, the mathematical problem, i.e., the matrix inversion, is costlier to solve as the polarization matrix itself is worse conditioned than in simple bulk water. Therefore, to ensure stability of very long time scale simulations towards microseconds where errors accumulate, they should all employ a tighter dipole convergence criterion (10^{-7}–10^{-8} D) leading to a higher number of iterations than usually discussed in benchmarks for short simulations, where the 10^{-5} D standard is employed, effectively causing really degraded real life performances.

Another set of methods address this issue by considering analytical formulas for the polarization energy.

The first idea in that direction was introduced by Wang,[12] who used Chebyshev polynomials to get analytical expressions of the polarization energy and its derivatives, which automatically ensures that the source of the energy drift previously evoked is removed. Unfortunately, the approach provided energy surfaces that were too far from the ones obtained with tightly converged iterative method and was thus not further investigated. Significant progresses were recently made in the same direction by Simmonett *et al.*[13] who proposed a revisitation of Wang's proposal through the ExPT (Extrapolated Perturbation Theory) perturbation approach, which is equivalent to the truncation of the Jacobi iterative method at a predetermined order combined with the use of a few parameters.

If the parametric aspect of their approach initially limited its global applicability to any type of system, the authors recently improved their method which is now denoted as OPT3 (OPT = Orders of Perturbation Theory)[14] by pushing it to higher order of perturbation and providing a systematic way for the parametrization, extending the applicability of the method. One advantage of the approach is its reduced cost compared to the best iterative approaches.

Alternatively, one can also consider the actual induced dipoles as new degrees of freedom and build an extended Lagrangian defining the way to propagate them during the dynamics without any SCF cycles.[15] The first results using this strategy are promising, and the method indeed does not require any iteration. On the performance side, one could argue that using a production PCG solver with a 10^{-5} D convergence threshold, a RESPA integrator with a 2 fs time step for the non-bonded forces coupled to Kolafa's ASPC is twice faster than the sequential iEL/0-SCF method with a 1 fs time step.[15] Nevertheless, this PCG speed advantage is only "apparent" as it does not solve the energy drift issue for long time scales whereas the iEl/0-SCF method has been shown to have improved energy conservation properties. This nice improvement is due to the use of thermostats and, therefore, iEL/0-SC

unfortunately suffers from the drawbacks of any extended Lagrangian approach that cannot use time steps larger than 1 fs.[6] As we stated before, if iterative methods do not have any theoretical upper limit to the time step they can be used with,[6] it requires not to use information from the past such as predictor-correctors, removing such speed advantage when using RESPA.

As we see from this discussion, the question of which method to adopt is complex as it appears difficult to combine all possible improvements.

In fact, we can state that reducing the computational cost of an iterative method to compute the polarization energy and forces always comes with degraded energy conservation. Energy conservation is tricky as it depends on the chemical nature of the system (charged or not, homogeneous or not). For example, polarization of bulk water systems requires less iterations to converge with PCG solvers. On the other hand, the ExPT method behaves poorly for the ionic liquid system that will be studied in the section titled "TCG: Notations,"[4] and the Jacobi method does not even converge in that case.

A major difficulty to compute the polarization energy and its gradient for future microsecond simulations is to offer a non-empirical strategy applicable to any kind of systems, embodying the following properties.

Indeed, such a method should be systematically improvable in order to allow the user to set the accuracy of the simulation depending on its goal. For example, the simple Jacobi method has been shown not to converge in several cases[2] and adding iterations would not improve the results. It should show good conservation of the total energy during a microcanonical simulation, ensuring good accuracy on the forces driving the dynamics. It should also be non-parametric to provide a close reproduction of any type of potential energy surface, without having to resort to force-field model reparametrization. In practice, a polarization scheme should also be affordable with a computational cost as reduced as possible. It should allow us to use larger time steps through multiple time step schemes such as RESPA. In the end, the selected criterion to compare computational efficiencies of the various schemes should be the global cost of computing both energy and derivatives with similar energy conservation capabilities for a given trajectory length.

TCG: CONTEXT

To address all these required features, we recently introduced a non-empirical and non-iterative strategy denoted as the Truncated Conjugate Gradient (TCG).[4] TCG is derived by explicitly writing down all numerical operations of a finite number of conjugate gradient cycles of iteration which can be user-chosen (be TCG-n, n = 1,3). As the number of operations in the TCG approach is fixed once and for all, it is possible to derive an exact analytical expression of the gradient of the energy like in ExPT/OPT3,[14] avoiding by construction any energy drift in microcanonical simulations and thus ensuring energy conservation in that context. The higher the TCG level is, the higher its accuracy is, as TCG inherits from the properties of the conjugate gradient and benefits from the fact that it is a Krylov method in which the associated error is monotonically reduced at each iteration. It can be shown in that context that the CG-method is mathematically optimal, meaning that it minimizes exactly the polarization energy on the so-called Krylov subspaces at each iteration and therefore guarantees that the number of the required matrix-vector products (1 per iteration in any iterative approach) is reduced to a minimum compared to other iterative methods. Moreover, the TCG accuracy can be improved at negligible costs (i.e., without any additional matrix-vector product) (i) by using preconditioners as presented above leading to the Truncated Preconditioned Conjugate Gradient (TPCG); (ii) by using the residue of the final CG step, available without any additional cost, to perform an additional "peek" iteration, equivalent to one step of Jacobi Over Relaxation (JOR) with a relaxation parameter which can be found adaptively.

Overall, the TCG approach was found to accurately reproduce energy surfaces at a reduced computational cost providing analytical forces. As it does not rely on history, it does not suffer from MD perturbations such as the ones arising when predictor guesses, which break the time-reversibility of the simulation, are used in polarization solvers. It is for the same reasons compatible with the use of a large time step with multi-time step integrators. Also, being based on the conjugate gradient and thus relying essentially on matrix vector products and computation of electric fields, it can replace standard solvers in a regular implementation including linear scaling ones using smooth particle mesh Ewald. Furthermore, it does not require additional advanced thermostating nor any additional parameter.

The purpose of this paper is to address one delicate point which is the main bottleneck of the TCG method: the complex derivation of its gradients. If TCG answers all the desired discussed properties for a polarization solver, a naive derivation of the energy gradients can lead to an undesired additional computational cost, while the method should remain analytical and accurate but cheap as well. The goal here is to detail a strategy enabling the fast computation of the analytical gradients that would allow developers to efficiently implement the TCG approach in the software of their choice. We will first present the technical aspect of TCG and its notations, and then we will detail the optimal computation of gradients in a form that could be implemented by developers.

TCG: NOTATIONS

We will place ourselves in the context of the AMOEBA force field[16] and consider a system of N atoms, each embodying a multipole expansion (up to quadrupoles) as permanent charge density and a polarizability tensor α_i. We will denote $\mathbf{E}$ as the $3N$ vector gathering all electric fields $\vec{E}_i$ created by the permanent charge density at atomic position i, and $\boldsymbol{\mu}$ is the equivalent $3N$ vector gathering the induced dipoles experienced at each atomic site. $\mathbf{T}$ is the $3N \times 3N$ polarization matrix, defined by blocks as follows. It bears the 3×3 polarizability tensors α_i along its diagonal block, and the interaction between the ith and jth dipoles is represented as the T_{ij}

tensor,

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

This matrix is symmetric and positive definite. Thanks to the Thole damping of the electric field at a short range, any polarization catastrophe is prevented. Indeed, the Thole damping acts on the eigenvalues; without Thole damping, negative eigenvalues could be found which is a problem for conjugate gradient methods.[1]

Using these notations, the total polarization energy can be expressed as follows:

$$E_{\text{pol}} = \frac{1}{2}\boldsymbol{\mu}^T \mathbf{T}\boldsymbol{\mu} - \boldsymbol{\mu}^T \mathbf{E}, \tag{1}$$

where $\boldsymbol{\mu}^T\mathbf{E}$ represents the scalar product of vectors $\boldsymbol{\mu}$ and $\mathbf{E}$ (also noted $\langle \boldsymbol{\mu}, \mathbf{E} \rangle$). One can easily see that the dipole vector $\boldsymbol{\mu}$ minimizing (1) verifies the following linear system:

$$\mathbf{T}\boldsymbol{\mu} = \mathbf{E} \tag{2}$$

giving the minimized polarization energy

$$E_{\text{pol}} = -\frac{1}{2}\boldsymbol{\mu}^T \mathbf{E}. \tag{3}$$

As explained earlier, the TCG method that we use to solve this equation derives from the conjugate gradient algorithm. It uses three vectors upon starting: the guess $\boldsymbol{\mu}_0$, the initial residual $\mathbf{r}_0 = \mathbf{T}\boldsymbol{\mu}_0 - \mathbf{E}$, and an initial descent direction $\mathbf{p}_0$ that we set to be equal to $\mathbf{r}_0$. It reads as follows:

$$\begin{cases} \gamma_i = \frac{\mathbf{r}_i^{\mathbf{T}} \mathbf{r}_i}{\mathbf{p}_i^T \mathbf{T} \mathbf{p}_i} \\ \boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i + \gamma_i \mathbf{p}_i \\ \mathbf{r}_{i+1} = \mathbf{r}_i - \gamma_i \mathbf{T}\mathbf{p}_i \\ \beta_{i+1} = \frac{\mathbf{r}_{i+1}^T \mathbf{r}_{i+1}}{\mathbf{r}_i^T \mathbf{r}_i} \\ \mathbf{p}_{i+1} = \mathbf{r}_{i+1} + \beta_{i+1}\mathbf{p}_i \end{cases}. \tag{4}$$

Instead of using a convergence criterion as a condition to stop iterating, as this is usually done, one can choose to arbitrarily fix the number of iterations and to unfold a finite number of computational operations that makes it fixed cost and non-iterative, as explained above. This defines our Truncated Conjugate Gradient (TCG) method. Besides the obvious advantage of drastically reducing the computational cost of each induced polarization calculation, it allows one to simulate perfectly stable molecular dynamics, without drift over time, as explained in Ref. 4. This advantage is not limited to MD and could be exploited in Monte Carlo simulations.

The exact, total derivative of the energy with respect to the nuclear position should be

$$\frac{\mathrm{d}E_{\text{pol}}}{\mathrm{d}r_i} = \frac{\partial E_{\text{pol}}}{\partial \boldsymbol{\mu}} \frac{\partial \boldsymbol{\mu}}{\partial r_i} + \frac{\partial E_{\text{pol}}}{\partial r_i}. \tag{5}$$

When using an iterative method, the provided solution $\boldsymbol{\mu}$ is inexact (approached only); thus the energy is not perfectly minimized with respect to the dipoles (the term $\partial E_{\text{pol}}/\partial \boldsymbol{\mu}$ is not zero). One usually still makes this erroneous assumption, giving $\mathrm{d}E_{\text{pol}}/\mathrm{d}r_i = \partial E_{\text{pol}}/\partial r_i$. This leads to computing forces that do not perfectly correspond to the system and thus to an unavoidable drift in the subsequent simulations.

If one fixes the number of iterations, it is however possible to "unroll" the analytical formula for the final polarization vector, expressed as a function of the starting quantities ($\boldsymbol{\mu}_0$, $\mathbf{r}_0$). Noting $\boldsymbol{\mu}_{\text{TCG}n}$ vector, with n the truncation order (i.e., the number of iterations of the algorithm), one obtains the TCGn family of methods that reads up to order three,

$$\boldsymbol{\mu}_{TCG1} = \boldsymbol{\mu}_0 + t_4 \mathbf{r}_0, \tag{6}$$

$$\boldsymbol{\mu}_{TCG2} = \boldsymbol{\mu}_0 + (\gamma_1 t_2 + t_4)\mathbf{r}_0 - \gamma_1 t_4 \mathbf{P}_1, \tag{7}$$

$$\boldsymbol{\mu}_{TCG3} = \boldsymbol{\mu}_0 + (t_4 + \gamma_1 t_2 + \gamma_2 + \gamma_2\beta_2 t_2)\mathbf{r}_0 - (\gamma_1 t_4 + \gamma_2 t_4 + \gamma_2\beta_2 t_4)\mathbf{P}_1 - \gamma_1\gamma_2 \mathbf{P}_2. \tag{8}$$

All quantities used in the previous equations are defined in the Appendix. In practice, we showed that one could stop as the TCG2 level, as it is accurate enough.

FAST COMPUTATION OF THE GRADIENTS

In this section, we first explain that computing the gradients of the energy, even though an analytical expression is at our disposal, is not straightforward. We then show how to pass the different hurdles encountered.

Having the analytical, exact expression of the dipoles allows one to differentiate them in an equally exact manner. A formal differentiation, with a prime "$'$" denoting it, would give for the first two orders,

$$\boldsymbol{\mu}'_{TCG1} = \boldsymbol{\mu}'_0 + t_4 \mathbf{r}'_0 + t'_4 \mathbf{r}_0, \tag{9}$$

$$\boldsymbol{\mu}'_{TCG2} = \boldsymbol{\mu}'_0 + (t_4 + \gamma_1 t_2)\mathbf{r}'_0 + (t'_4 + \gamma'_1 t_2 + \gamma_1 t'_2)\mathbf{r}_0 + \gamma'_1 t_4 \mathbf{P}_1 + \gamma_1 t'_4 \mathbf{P}_1 + \gamma_1 t_4 \mathbf{P}'_1. \tag{10}$$

However, the differentiation of a $3N$ vector with respect to $3N$ spatial coordinates would build a $3N \times 3N$ matrix. This leads to three obstacles that slow down the gradient computation:

- First, a scalar product of one such derivative $\mathbf{A}'$ with another vector $\mathbf{B}$ would lead to a $(3N)^2$ operation, which is a non-negligible cost, repeated for all products of this $\langle \mathbf{A}', \mathbf{B} \rangle$ form.
- Second, these products, when using the analytical expressions [Eqs. (9) and (10)] "as is," are repeated an unnecessary number of times, effectively making this slow-down a pure stop.
- Third, one can see that there are two types of vectors building $\boldsymbol{\mu}_{\text{TCG}n}$: the electric field $\mathbf{E}$, but also the product of the residue with successive powers of the polarization matrix ($\mathbf{r}_0$, $\mathbf{T}\mathbf{r}_0 = \mathbf{P}_1$, more generally $\mathbf{T}^m\mathbf{r}_0$, with m an integer). Differentiating $\boldsymbol{T}^m\boldsymbol{r}_0$ exhibits, amongst others, a $\mathbf{T}^p\mathbf{T}'\mathbf{T}^q\mathbf{r}_0$ term (with p and q two integers verifying $p + q + 1 = m$); computing such a $\mathbf{T} \cdot \mathbf{T}'\mathbf{A}$ product is equivalent to a matrix-matrix product, which is also computationally too expensive.

This makes a naive implementation of our method effectively unusable. Yet to run a classical simulation, one needs the

forces, i.e., the gradients of the polarization energy, rather than the derivatives of the dipoles themselves. What one really needs is thus the derivative of the following scalar product:

$$E_{\text{pol}} = \frac{1}{2}\langle \mathbf{E}, \boldsymbol{\mu}_{\text{TCG}n}\rangle, \tag{11}$$

that is, formally,

$$E'_{\text{pol}} = \frac{1}{2}\langle \mathbf{E}', \boldsymbol{\mu}_{\text{TCG}n}\rangle + \frac{1}{2}\langle \mathbf{E}, \boldsymbol{\mu}'_{\text{TCG}n}\rangle. \tag{12}$$

First, developing Eq. (12) shows that all scalar products involved a differentiated quantity: either a differentiated matrix (like $\langle \mathbf{A}, \mathbf{T}'\mathbf{B}\rangle$) or the derivative of the field itself ($\mathbf{E}'$). An analogy, or dimensional analysis, allows us to compare these terms to forces, with $\langle \mathbf{A}, \mathbf{E}'\rangle$ corresponding to a force produced by the interaction of the dipoles $\mathbf{A}$ with the electric field, and $\langle \mathbf{B}, \mathbf{T}'\mathbf{C}\rangle$ to a force arising from the interaction between two sets of dipoles $\mathbf{B}$ and $\mathbf{C}$. The expensive part of computing such quantities lies in the calculation of distances. All of these forces can be computed in a single double loop [whose cost is $O(N^2)$ for direct calculations and $O(N \log N)$ when using SPME] to minimize the computational cost and compute the said distances only once. This addresses the first hurdle evoked earlier.

We can also reorganize the gradient computation in order to minimize the number of the expensive scalar products involving a vector and a differentiated vector, by grouping all these scalar products and performing them all at once (given three vectors $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, if one needs to compute $\langle \mathbf{A}, \mathbf{B}'\rangle + \langle \mathbf{C}, \mathbf{B}'\rangle$, it is much more efficient to first prepare a vector $\mathbf{D} = \mathbf{A} + \mathbf{C}$ and then to compute $\langle \mathbf{D}, \mathbf{B}'\rangle$). This optimization, though quite simple in principle, actually requires quite involved expressions (see the Appendix). It is a simple solution to the second obstacle we listed.

Third, since $\mathbf{T}$ is a symmetric matrix, we have $\langle \mathbf{TA}, \mathbf{B}\rangle = \langle \mathbf{A}, \mathbf{TB}\rangle$ for any two vectors $\mathbf{A}$ and $\mathbf{B}$. In particular, for our generic vectors $\mathbf{T}^m\mathbf{r}_0$,

$$\langle \mathbf{T}^p\mathbf{T}'\mathbf{T}^q\mathbf{r}_0, \mathbf{A}\rangle = \langle \mathbf{T}'\mathbf{T}^q\mathbf{r}_0, \mathbf{T}^p\mathbf{A}\rangle. \tag{13}$$

Considering scalar products thus allows us to get rid of the matrix-matrix ($\mathbf{TT}'$) products, our third hurdle.

Overall, the solution to overcome our obstacles came from considering the polarization energy instead of the induced dipole themselves.

To illustrate our solution, one can write the analytical formulas as follows, for the TCG at order one and two, respectively:

$$E'_{\text{pol, TCG1}} = \frac{1}{2}\left(\langle \mathbf{r}'_0, a^{(1)}_{1,0}\mathbf{E} + a^{(1)}_{1,1}\mathbf{r}_0 + a^{(1)}_{1,2}\mathbf{T}\mathbf{r}_0\rangle + \langle \mathbf{T}'\mathbf{r}_0, a^{(1)}_{2,1}\mathbf{r}_0\rangle\right), \tag{14}$$

$$\begin{aligned} E'_{\text{pol, TCG2}} = \frac{1}{2}\Big(&\langle \mathbf{E}', \boldsymbol{\mu}_{\text{TCG2}}\rangle + \langle \boldsymbol{\mu}'_0, \mathbf{E}\rangle + \langle \mathbf{r}'_0, a^{(2)}_{1,0}\mathbf{E} + a^{(2)}_{1,-1}\mathbf{TE} \\ &+ a^{(2)}_{1,1}\mathbf{r}_0 + a^{(2)}_{1,2}\mathbf{T}\mathbf{r}_0 + a^{(2)}_{1,3}\mathbf{T}^2\mathbf{r}_0 + a^{(2)}_{1,4}\mathbf{T}^3\mathbf{r}_0\rangle \\ &+ \langle \mathbf{T}'\mathbf{r}_0, a^{(2)}_{2,0}\mathbf{E} + a^{(2)}_{2,1}\mathbf{r}_0 + a^{(2)}_{2,2}\mathbf{T}\mathbf{r}_0 + a^{(2)}_{2,3}\mathbf{T}^2\mathbf{r}_0\rangle \\ &+ \langle \mathbf{T}'\mathbf{T}\mathbf{r}_0, a^{(2)}_{3,1}\mathbf{r}_0 + a^{(2)}_{3,2}\mathbf{T}\mathbf{r}_0\rangle + \langle \mathbf{T}'\mathbf{T}^2\mathbf{r}_0, a^{(2)}_{4,1}\mathbf{r}_0\rangle\Big), \end{aligned} \tag{15}$$

where the coefficients $a^{(k)}_{i,j}$ are the result of the cumbersome derivation evoked earlier; their explicit expression can be found in the Appendix.

As stated earlier in this paper, the so-called peek-step is a supplementary JOR iteration based on the last obtained residual $\mathbf{r}_n$. It simply improves the solution to reach the following expression:

$$\boldsymbol{\mu}^{(peek)}_{\text{TCG}n} = \boldsymbol{\mu}_{\text{TCG}n} + \omega\boldsymbol{\alpha}\mathbf{r}_n, \tag{16}$$

where α is the relaxation parameter mentioned earlier; more precisions on its choice can be found in Ref. 4. Defining $\boldsymbol{\mu}_{\text{peek, TCG}n} = \omega\boldsymbol{\alpha}\mathbf{r}_n$, the supplementary contribution of the peek step can be also written as follows:

$$\begin{aligned} E'_{\text{peek, TCG1}} = &\langle \boldsymbol{\mu}_{\text{peek, TCG1}}, \mathbf{E}'\rangle + \langle \mathbf{r}'_0, a^{(1,p)}_{1,\alpha 0}\boldsymbol{\alpha}\mathbf{E} + a^{(1,p)}_{1,1\alpha}\mathbf{T}\boldsymbol{\alpha}\mathbf{E} \\ &+ a^{(1,p)}_{1,1}\mathbf{r}_0 + a^{(1,p)}_{1,2}\mathbf{T}\mathbf{r}_0\rangle \\ &+ \langle \mathbf{T}'\mathbf{r}_0, a^{(1,p)}_{2,1}\mathbf{r}_0 + a^{(1,p)}_{2,\alpha 0}\boldsymbol{\alpha}\mathbf{E}\rangle, \end{aligned} \tag{17}$$

$$\begin{aligned} E'_{\text{peek, TCG2}} = &\langle \boldsymbol{\mu}_{\text{peek, TCG2}}, \mathbf{E}'\rangle \\ &+ \langle \mathbf{r}'_0, a^{(2,p)}_{1,0\alpha}\boldsymbol{\alpha}\mathbf{E} + a^{(2,p)}_{1,1\alpha}\mathbf{T}\boldsymbol{\alpha}\mathbf{E} + a^{(2,p)}_{1,2\alpha}\mathbf{T}^2\boldsymbol{\alpha}\mathbf{E} \\ &+ a^{(2,p)}_{1,1}\mathbf{r}_0 + a^{(2,p)}_{1,2}\mathbf{T}\mathbf{r}_0 + a^{(2,p)}_{1,3}\mathbf{T}^2\mathbf{r}_0 + a^{(2,p)}_{1,4}\mathbf{T}^3\mathbf{r}_0\rangle \\ &+ \langle \mathbf{T}'\mathbf{r}_0, a^{(2,p)}_{2,\alpha 0}\boldsymbol{\alpha}\mathbf{E} + a^{(2,p)}_{2,1\alpha}\mathbf{T}\boldsymbol{\alpha}\mathbf{E} + a^{(2,p)}_{2,1}\mathbf{r}_0 \\ &+ a^{(2,p)}_{2,2}\mathbf{T}\mathbf{r}_0 + a^{(2,p)}_{2,3}\mathbf{T}^2\mathbf{r}_0\rangle \\ &+ \langle \mathbf{T}'\mathbf{T}\mathbf{r}_0, a^{(2,p)}_{3,\alpha 0}\boldsymbol{\alpha}\mathbf{E} + a^{(2,p)}_{3,1}\mathbf{r}_0 + a^{(2,p)}_{3,2}\mathbf{T}\mathbf{r}_0\rangle \\ &+ \langle \mathbf{T}'\mathbf{T}^2\mathbf{r}_0, a^{(2,p)}_{4,1}\mathbf{r}_0\rangle \end{aligned} \tag{18}$$

(the coefficients $a^{(k,p)}_{i,j}$, as well as an explicit formula for the $\boldsymbol{\mu}_{\text{peek}}$ vectors, are reproduced in the Appendix). One should then simply sum the corresponding terms to obtain the final expression for the polarization energy gradients in a computationally feasible way, for example, the scalar product $\langle \mathbf{r}'_0, \mathbf{r}_0\rangle$ should now be multiplicated by coefficient $a^{(1)}_{1,1} + a^{(1,p)}_{1,1}$ to get the correct gradients for TCG1.

All these formulas have been tested and validated against gradients obtained via finite differences. Such details could be useful to allow anyone to implement the fast evaluation of the forces necessary to the use of TCG. The source code of this method will be freely available in Tinker-HP version 1.1.[17]

To sum up, the implementation of the gradient calculation that we propose here follows these three steps: First, we compute the successive matrix-vector products to build the successive $\mathbf{T}^m\mathbf{r}_0$ vectors needed; second, we perform the various scalar products appearing in our analytical formulas, allowing us to assemble (through weighted sums) a second set of vectors; finally, we perform simultaneously on all these assembled vectors a "force-like" calculation. The choice to use—or not—a peek step only changes the assembled vectors on step two, through an extra set of coefficients as presented above.

NUMERICAL RESULTS

In this section, we report the timings of the implementation presented above for different systems as it has been added to the software Tinker-HP. More precisely, we report

the cost of the calculation of the polarization energy and the associated forces with different methods: a standard diagonally preconditioned conjugate gradient (PCG) with a 10^{-5} D convergence threshold, the same method with a tighter 10^{-8} D convergence threshold (that ensures energy conservation as explained above), and the TPCG1 and the TPCG2 methods with the "direct field"[1] $\boldsymbol{\alpha}\mathbf{E}$ as guess $\boldsymbol{\mu}_0$ with a Jacobi peek step ($\omega = 1$). For the two PCG solver settings, the average number of iterations is also reported in parentheses. Note that the computational cost of these two methods would be the same with any other kind of peek steps whose cost is negligible, as described in Ref. 4. For the PCG solvers, we report timings using the simple "direct field" as a guess [noted "PCG (10^{-x} D)" in Table I] and also timings using the ASPC predictor [noted "PCG (10^{-x} D, ASPC)"].[7] These methods are timed in the nowadays standard context of the RESPA integrator[8] used with a 2 fs time step for the non-bonded forces.

The systems that are tested here are the same than in our previous work:[4] three solvated protein droplets (the HIV nucleocapsid ncp7 made of 18 518 atoms, the ubiquitin made of 9737 atoms, and the dihydrofolate reductase, dhfr, with 23 558 atoms) and an ionic liquid, the dimethyl-imidazolium [dmim+][Cl−] (3672 atoms). No boundary conditions are used in these tests; therefore, each matrix-vector product and force computation involved in the PCG solvers and in the TCG formulas has a $O(N^2)$ computational cost. However, these matrix-vector products can be easily re-expressed following the possible choices for the boundary conditions that will give rise to slightly different forms of the polarization matrix. For example, TCG being really close to PCG, it can either be applied in the context of the particle mesh Ewald[2,18] method with a $O(N \ln N)$ cost, or using the fast multipole summation technique[19] with a $O(N)$ cost. These operations are by far the costliest in the computation of the dipoles and of the polarization forces. This is why we report the timings as their proportional cost compared to the PCG solver with a convergence threshold of 10^{-5} D and the direct field as a guess, as these proportions would be the same when using other boundary conditions. We chose these settings to be our reference.

All these (sequential) timings were obtained on an HP 620 Workstation made of Intel Xeon E5-2665 CPUs at 2.4 GHz and were averaged over 100 ps of NVT trajectories at 300 K for the protein droplets and at 425 K for the ionic liquid.

TABLE I. Average time for the computation of the polarization energy and the associated forces for different methods, using the PCG converged at 10^{-5} D as the reference, for a RESPA(2 fs) time step. In parentheses, mean number of iterations needed.

	Ubiquitin	ncp7	dhfr	[dmim+][Cl−]
PCG (10^{-5} D)	100% (8)	100% (8)	100% (8)	100% (8)
PCG (10^{-5} D, ASPC)	88% (6)	85% (6)	88% (6)	84% (5)
PCG (10^{-8} D)	136% (15)	138% (15)	143% (16)	138% (15)
PCG (10^{-8} D, ASPC)	125% (13)	127% (13)	125% (13)	117% (12)
TPCG1	43%	43%	44%	44%
TPCG2	61%	62%	63%	63%

We observe that both the TPCG methods are significantly faster compared to standard production settings (10^{-5} D). Compared to more strict settings using a convergence criterion of 10^{-8} D for the PCG solver, which guarantees energy conservation during the MD simulation, differences are even more striking because the computational cost of the TPCG1 and TPCG2 methods is found to be, respectively, more than three times faster and more than twice faster, respectively.

This means that using these methods with the implementation described in this paper enables not only to guarantee energy conservation but also to save a considerable amount of time during the computation of the polarization energy and the associated forces.

Concerning the use of ASPC, a striking result at a time step of 2 fs is the smaller reduction of iterations necessary to reach convergence compared to the reduction observed at 1 fs[1] where a 50% gain was observed for a 10^{-5} D threshold. In other words, ASPC guess is less efficient when using a bigger time step. Following intuition, the shorter the time step, the more efficient the ASPC is. Moreover, in line with our previous study,[1] we also observed that the proportional gain in that regard is even smaller for a tighter dipole convergence criterion (such as 10^{-8} D), making very long simulations a daunting challenge.

Another remark concerns the use of even larger time steps with the RESPA integrator. It has been indeed shown that one can use a 3 fs time step for the non-bonded forces, provided that masses of the hydrogen atoms of the system are appropriately redistributed among heavy atom carriers.[20] But such large time steps limit the use of predictor such as the ASPC, and no gain in the number of iterations can be obtained with these methods. On the contrary, the computational cost of the T(P)CG family of methods does not suffer from such a change as no history is taken into account. The computational cost at 3 fs would remain the same as that in the 2 fs context, offering an automatic 1.5 acceleration for the same trajectory length at no cost, increasing the global speedup offered by the use of T(P)CG.

CONCLUSION

As we have seen, one can reformulate the analytical expressions for the gradients of the truncated conjugate gradient using a clear strategy. We detailed for interested developers the various steps required for the implementation of the complete TCG method including fast force computations.

This strategy allows the implementation of these gradients to be fast enough for the computational cost of an evaluation of the polarization energy and the associated forces to be greatly reduced compared to standard production settings using iterative methods. The TPCG2 method is more than 1.6 times faster than the PCG solver with a 10^{-5} D convergence criterion and the direct field as a guess using a RESPA integrator with a 2 fs time step (1.4 when ASPC is used). Moreover, it is more than 2 times faster than a PCG with a convergence criterion of 10^{-8} D and the same predictor guess, such settings being mandatory to guarantee energy conservation with the standard PCG for long simulations. As the number of

operations in the TCG method is fixed and does not rely on history (i.e., no previous dipole guess nor predictor guess), it can be applied with larger time steps for the same fixed computational cost.

The TCG approach provides an accurate reproduction of energy surfaces[4] at a reduced computational cost, providing analytical forces that avoid by construction the drift issues without relying on complex parametrization nor adding extra degrees of freedom limiting the settings than one can use to integrate MD trajectories. That is why it should be a method of choice for long time scale and stable simulations using polarizable force fields. Since all TCG's analytical formulas involve the expressions of electric fields as well as matrix-vector products, these latter are easily and directly transposable in different boundary conditions. In particular, the extension to smooth particle mesh Ewald is straightforward. For the same reasons, the parallel implementation of these methods within the context of spatial decomposition follows any PCG one and will be described in a future paper dedicated to the massively parallel Tinker-HP package. In that context, capabilities of the AMOEBA force field using a TCG/SPME coupling will be tested by comparing various properties obtained with these methods.

ACKNOWLEDGMENTS

This work was supported in part by French state funds managed by CalSimLab and the ANR within the Investissements d Avenir program under Reference No. ANR-11-IDEX-0004-02. Jean-Philip Piquemal and Louis Lagardère are grateful for support by the Direction Générale de l'Armement (DGA) Maitrise NRBC of the French Ministry of Defense.

APPENDIX: COEFFICIENTS AND PEEK STEP FORMULAS

We introduce the following notations to express the analytical formulas of the induced dipoles, as well as their derivatives. Each term can be expressed using the starting vectors ($\mathbf{r}_0$ and $\boldsymbol{\mu}_0$) and the polarization matrix $\mathbf{T}$.

Vectors:

- $\mathbf{r}_0 = \mathbf{E} - \mathbf{T}\boldsymbol{\mu}_0$,
- $\mathbf{p}_0 = \mathbf{r}_0$,
- $\mathbf{P}_1 = \mathbf{T}\mathbf{r}_0$,
- $\mathbf{P}_2 = t_2\mathbf{P}_1 - t_4\mathbf{T}^2\mathbf{r}_0$,
- $\mathbf{P}_3 = (1+\beta_2 t_2)\mathbf{T}\mathbf{r}_0 - (t_4+\beta_2 t_4)\mathbf{T}\mathbf{P}_1 - \gamma_1\mathbf{T}\mathbf{P}_2$.

Scalars:

- $n_0 = \mathbf{r}_0^T\mathbf{r}_0$,
- $t_1 = \mathbf{r}_0^T\mathbf{P}_1$,
- $t_2 = \dfrac{n_0||\mathbf{P}_1||^2}{t_1^2}$,
- $t_3 = t_1\mathbf{P}_1^T\mathbf{P}_2$,
- $t_4 = \dfrac{n_0}{t_1}$,
- $t_5 = \mathbf{P}_1^T\mathbf{P}_2$,
- $t_8 = t_5 = t_2||\mathbf{P}_1||^2 - t_4 t_9$,
- $t_9 = \mathbf{r}_0^T\mathbf{T}^3\mathbf{r}_0$,
- $t_{10} = t_1^2 - n_0||\mathbf{P}_1||^2$,
- $\gamma_1 = \dfrac{t_1^2 - n_0||\mathbf{P}_1||^2}{t_3}$,
- $sp_0 = \mathbf{r}_0^T\mathbf{E}$,
- $sp_1 = \mathbf{P}_1^T\mathbf{E} = \mathbf{E}^T\mathbf{T}\mathbf{r}_0$,
- $b_1 = sp_0 - \gamma_1 sp_1$,
- $b_2 = sp_0 t_2 - t_4 sp_1$,
- $spp_1 = \langle\boldsymbol{\alpha}\mathbf{E},\mathbf{E}\rangle$,
- $spp_2 = \langle\boldsymbol{\alpha}\mathbf{T}\mathbf{r}_0,\mathbf{E}\rangle$,
- $\beta_2 = \dfrac{n_0 + t_4^2||\mathbf{P}_1||^2 + \gamma_1^2||\mathbf{P}_2||^2 - 2t_1t_4 - 2\gamma_1 t_4||\mathbf{P}_1||^2 + 2\gamma_1 t_4 t_5}{(t_2-1)n_0}$,
- $\gamma_2 = \dfrac{n_0 + t_4^2||\mathbf{P}_1||^2 + \gamma_1^2||\mathbf{P}_2||^2 - 2t_1t_4 - 2\gamma_1 t_4||\mathbf{P}_1||^2 + 2\gamma_1 t_4 t_5}{(1+\beta_2 t_2)\mathbf{r}_0^T\mathbf{P}_3 - (t_4+\beta_2 t_4)\mathbf{P}_1^T\mathbf{P}_3 + \gamma_1\mathbf{P}_2^T\mathbf{P}_3}$.

1. Peek-step formulas

$$\boldsymbol{\mu}_{\text{peek, TCG1}} = \omega\boldsymbol{\alpha}\mathbf{r}_0 - \omega t_4\boldsymbol{\alpha}\mathbf{P}_1, \tag{A1}$$

$$\boldsymbol{\mu}_{\text{peek, TCG2}} = \omega\boldsymbol{\alpha}\mathbf{r}_0 - \omega t_4\boldsymbol{\alpha}\mathbf{P}_1 - \omega\boldsymbol{\alpha}\gamma_1 t_2\mathbf{P}_1 - \omega\boldsymbol{\alpha}\gamma_1 t_4\mathbf{T}^2\mathbf{r}_0. \tag{A2}$$

2. Coefficients for the analytical expressions

The superscript number, between parentheses, indicates the truncation number (1 or 2). p indicates that the coefficient corresponds to the peek-step derivative and needs to be added to the energy derivative coefficient itself.

Derivation of $E_{\text{pol, TCG1}}$:

- $a^{(1)}_{1,0} = t_4$,
- $a^{(1)}_{1,1} = \frac{2sp_0}{t_1} + t_4$,
- $a^{(1)}_{1,2} = -\frac{2sp_0n_0}{t_1^2}$,
- $a^{(1)}_{2,1} = -\frac{sp_0n_0}{t_1^2}$.

Peek-step for TCG1:

- $a^{(1,p)}_{1,\alpha 0} = \omega$,
- $a^{(1,p)}_{1,1\alpha} = -t_4\omega$,
- $a^{(1,p)}_{1,1} = -\frac{2spp_1\omega}{t_1}$,
- $a^{(1,p)}_{1,2} = \frac{2n_0spp_1\omega}{t_1^2}$,
- $a^{(1,p)}_{2,\alpha 0} = -t_4\omega$,
- $a^{(1,p)}_{2,1} = \frac{n_0spp_1\omega}{t_1^2}$.

TCG2:

- $a^{(2)}_{1,0} = t_4 + \gamma_1 t_2$,
- $a^{(2)}_{1,-1} = -\gamma_1 t_4$,
- $a^{(2)}_{2,0} = -\gamma_1 t_4$,
- $a^{(2)}_{2,1} = -\frac{n_0b_1}{t_1^2} + 2\frac{t_1b_2}{t_3} - \frac{n_0t_9t_{10}b_2}{t_1t_3^2} + 2\frac{t_2np_1t_{10}b_2}{t_3^2} - \frac{t_8t_{10}b_2}{t_3^2} - 2\frac{n_0np_1sp_0\gamma_1}{t_1^3}$,
- $a^{(2)}_{1,1} = \frac{2b_1}{t_1} - \frac{2np_1b_2}{t_3} - 2\frac{np_1^2t_{10}b_2}{t_3^2t_1} + 2\frac{t_9t_{10}b_2}{t_3^2} + 2\frac{np_1sp_0\gamma_1}{t_1^2}$,
- $a^{(2)}_{1,2} = -2\frac{n_0b_1}{t_1^2} + 4\frac{t_1b_2}{t_3} - 2\frac{n_0t_9t_{10}b_2}{t_1t_3^2} + 4\frac{t_2np_1t_{10}b_2}{t_3^2} - 2\frac{t_8t_{10}b_2}{t_3^2} - 4\frac{4n_0np_1sp_0\gamma_1}{t_1^3}$,
- $a^{(2)}_{1,3} = -4\frac{t_1t_2t_{10}b_2}{t_3^2} - 2\frac{n_0b_2}{t_3} + 2\frac{n_0sp_0\gamma_1}{t_1^2}$,
- $a^{(2)}_{1,4} = 2\frac{t_1t_4t_{10}b_2}{t_3^2}$,
- $a^{(2)}_{2,2} = -\frac{n_0b_2}{t_3} - 2\frac{t_1t_2t_{10}b_2}{t_3^2} + \frac{n_0sp_0\gamma_1}{t_1^2}$,
- $a^{(2)}_{2,3} = \frac{t_1t_4t_{10}b_2}{t_3^2}$,
- $a^{(2)}_{3,1} = -\frac{n_0b_2}{t_3} - 2\frac{t_1t_2t_{10}b_2}{t_3^2} + \frac{n_0\gamma_1sp_0}{t_1^2}$,
- $a^{(2)}_{4,1} = \frac{t_1t_4t_{10}b_2}{t_3^2}$,
- $a^{(2)}_{3,2} = \frac{t_1t_4t_{10}b_2}{t_3^2}$.

Peek-step for TCG2:

- $a^{(2,p)}_{1,0\alpha} = \omega$,
- $a^{(2,p)}_{1,1\alpha} = -\omega(t_2\gamma_1 + t_4)$,
- $a^{(2,p)}_{1,1} = -\frac{2np_1}{t_1^2}\omega\gamma_1 spp_1 + (\omega t_2 spp_1 + \omega t_4 spp_2) \times \left(\frac{2np_1}{t_3} + \frac{2np_1^2t_{10}}{t_1t_3^2} - \frac{2t_9t_{10}}{t_3^2}\right) - \frac{2}{t_1}(\omega\gamma_1 spp_2 + \omega spp_1)$,
- $a^{(2,p)}_{1,2} = \frac{4n_0np_1}{t_1^3}\omega\gamma_1 spp_1 + (\omega t_2 spp_1 + \omega t_4 spp_2) \times \left(-\frac{4t_1}{t_3} + \frac{2n_0t_9t_{10}}{t_1t_3^2} - \frac{4np_1t_2t_{10}}{t_3^2} + \frac{2t_8t_{10}}{t_3^2}\right) + \frac{2n_0}{t_1^2}(\omega\gamma_1 spp_2 + \omega spp_1)$,
- $a^{(2,p)}_{1,3} = -\frac{2n_0}{t_1^2}\gamma_1\omega spp_1 + (\omega t_2 spp_1 + \omega t_4 spp_2)\left(\frac{4t_1t_2t_{10}}{t_3^2} + \frac{2n_0}{t_3}\right)$,
- $a^{(2,p)}_{1,4} = -(\omega t_2 spp_1 + \omega t_4 spp_2)\frac{2t_1t_4t_{10}}{t_3^2}$,
- $a^{(2,p)}_{2,\alpha 0} = -\omega(\gamma_1 t_2 + t_4)$,
- $a^{(2,p)}_{2,1} = \frac{2n_0np_1}{t_1^3}\omega\gamma_1 spp_1 + (\omega t_2 spp_1 + \omega t_4 spp_2) \times \left(-\frac{2t_1}{t_3} + \frac{n_0t_9t_{10}}{t_1t_3^2} - \frac{2np_1t_2t_{10}}{t_3^2} + \frac{t_8t_{10}}{t_3^2}\right) + \frac{n_0}{t_1^2}(\omega\gamma_1 spp_2 + \omega spp_1)$,
- $a^{(2,p)}_{2,2} = -\frac{n_0}{t_1^2}\omega\gamma_1 spp_1 + (\omega t_2 spp_1 + \omega t_4 spp_2)\left(\frac{n_0}{t_3} + \frac{2t_1t_2t_{10}}{t_3^2}\right)$,
- $a^{(2,p)}_{2,3} = -(\omega t_2 spp_1 + \omega t_4 spp_2)\frac{t_1t_4t_{10}}{t_3^2}$,
- $a^{(2,p)}_{3,1} = -\frac{n_0}{t_1^2}\omega\gamma_1 spp_1 + (\omega t_2 spp_1 + \omega t_4 spp_2)\left(\frac{n_0}{t_3} + \frac{2t_1t_2t_{10}}{t_3^2}\right)$,
- $a^{(2,p)}_{3,2} = -(\omega t_2 spp_1 + \omega t_4 spp_2)\frac{t_1t_4t_{10}}{t_3^2}$,
- $a^{(2,p)}_{1,2\alpha} = -\omega t_4\gamma_1$,
- $a^{(2,p)}_{2,1\alpha} = -\omega t_4\gamma_1$,
- $a^{(2,p)}_{3,\alpha 0} = -\omega\gamma_1 t_4$,
- $a^{(2,p)}_{4,1} = -(\omega t_2 spp_1 + \omega t_4 spp_2)\frac{t_1t_4t_{10}}{t_3^2}$.

[1]F. Lipparini, L. Lagardère, B. Stamm, E. Cancès, M. Schnieders, P. Ren, Y. Maday, and J.-P. Piquemal, J. Chem. Theory Comput. **10**, 1638–1651 (2014).
[2]L. Lagardère, F. Lipparini, E. Polack, B. Stamm, E. Cancès, M. Schnieders, P. Ren, Y. Maday, and J. P. Piquemal, J. Chem. Theory Comput. **11**, 2589–2599 (2015).
[3]U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. G. Pedersen, J. Chem. Phys. **103**, 8577–8593 (1995).
[4]F. Aviat, A. Levitt, B. Stamm, Y. Maday, P. Ren, J. W. Ponder, L. Lagardère, and J.-P. Piquemal, J. Chem. Theory Comput. **13**, 180–190 (2017).
[5]D. Nocito and G. J. O. Beran, J. Chem. Phys. **146**, 114103 (2017).
[6]W. Wang and R. D. Skeel, J. Chem. Phys. **123**, 164107 (2005).
[7]J. Kolafa, J. Comput. Chem. **25**, 335–342 (2004).
[8]M. Tuckerman, B. J. Berne, and G. J. Martyna, J. Chem. Phys. **97**, 1990–2001 (1992).
[9]A. Albaugh, O. Demerdash, and T. Head-Gordon, J. Chem. Phys. **143**, 174104 (2015).
[10]A. Niklasson, P. Steneteg, A. Odell, N. Bock, M. Challacombe, C. Tymczak, E. Holmström, G. Zheng, and V. Weber, J. Chem. Phys. **130**, 214109 (2009).
[11]D. Loco, L. Lagardère, S. Caprasecca, F. Lipparini, B. Mennucci, and J.-P. Piquemal, "Hybrid QM/MM molecular dynamics with AMOEBA polarizable embedding," J. Chem. Theory Comput. (published online).
[12]W. Wang, "Fast polarizable force field computation in biomolecular simulations," Ph.D. thesis, University of Illinois at Urbana-Champaign, 2013.
[13]A. C. Simmonett, F. C. Pickard IV, Y. Shao, T. E. Cheatham III, and B. R. Brooks, J. Chem. Phys. **143**, 074115 (2015).
[14]A. C. Simmonett, F. C. Pickard IV, J. W. Ponder, and B. R. Brooks, J. Chem. Phys. **145**, 164101 (2016).
[15]A. Albaugh, A. M. Niklasson, and T. Head-Gordon, J. Phys. Chem. Lett. **8**, 1714–1723 (2017).
[16]J. W. Ponder, C. Wu, P. Ren, V. S. Pande, J. D. Chodera, M. J. Schnieders, I. Haque, D. L. Mobley, D. S. Lambrecht, R. A. DiStasio, Jr., M. Head-Gordon, G. N. I. Clark, M. E. Johnson, and T. Head-Gordon, J. Phys. Chem. B **114**, 2564–2549 (2010).
[17]See http://www.ip2ct.upmc.fr/tinkerHP/ for Tinker-HP; accessed 31 May 2017.
[18]T. Darden, D. York, and L. Pedersen, J. Chem. Phys. **98**, 10089–10092 (1993).
[19]L. Greengard and V. Rokhlin, J. Comput. Phys. **73**, 325–348 (1987).
[20]C. W. Hopkins, S. Le Grand, R. C. Walker, and A. E. Roitberg, J. Chem. Theory Comput. **11**, 1864–1874 (2015).

2.3 Assessment of the TCG on static and dynamic properties

In this section, the reader will find results obtained with various options and orders of the TCG. The previous claim that a small order of truncation (or equivalently as small number of matrix-vector products) is enough to have satisfying simulations will be supported throughout the following paragraphs.

We will first present tests on static properties of various systems, computing various energies and verifying proper functioning of the simulations. Then we turn to the diffusion coefficient, in order to also be able to characterize the dynamical quality of our trajectories.

Finally, timings of the various TCG orders and setups are presented, and we propose a synthesis of these numerical tests. All results presented in the following were obtained using AMOEBA forcefield.[29] AMOEBA encompasses point-multipoles up to quadrupoles to represent the permanent density of charges, and is of course polarizable. It was calibrated for water,[30] nucleic acids and other various molecules.[29]

2.3.1 Static properties

What we call static properties designates properties that can be computed as ensemble averages, using integrals as presented in chapter 1 (eq. 1.5). These are thus quantities that will give us insight on how well the phase space is explored and whether configurations of the system are properly probed. We will report various energies and radial distribution functions to illustrate in this work. The movement of particles (molecules, atoms) is not under scrutiny here, and only their positions matter: no conclusion on the quality of our systems dynamics should be drawn from these results.

Various systems of interest were subject to tests here: liquid water boxes of various sizes, a small set of solvated proteins, but also ionic liquids.

Preliminar studies: CG dynamics

We will firstly present a set of results obtained using post-treatment of dynamics using the Conjugate Gradient solver. 100 ps simulations were carried out using a tightly converged Conjugate Gradient. One configuration frame was extracted every picosecond. The induced dipole vector, as well as the subsequent polarization energy, were then recalculated using the TCG with various settings (different orders and refinements) for each frame. An average was then performed over the hundred values, and

presented below.

The purpose of this "preliminary" study is to test whether a truncation as we proposed for TCG is indeed viable, and worthy of more developments. Simulations using the exact gradients (as derived in 2.2.4) will be presented after these first results.

Let us first look at the polarization energies obtained on three water systems (S1 containing 27 water molecules, S2 containing 216 water molecules and S3 4000). All were done in the NVT ensemble, using a Bussi thermostat, thermalizing the system at 300 K. We present the polarization energies obtained for various preconditioners and truncation orders in table 2.1.

Solver	Prec.	S1 (27 w.m.)	S2 (216 w.m.)	S3 (4000 w.m.)
Ref	-	-81.03	-803.33	-15229.87
TCG1	-	-73.50 (+8%)	-728.73 (+9%)	13813.35 (+9%)
TCG1	diag	-74.98 (+7%)	-741.91 (+8%)	-14028.18 (+8%)
TCG1	Skeel	-78.63 (+3%)	-779.17 (+3%)	-14743.48 (+3%)
TCG2	-	-80.69 (+0.4%)	-800.32 (+0.3%)	-15173.15 (+0.3%)
TCG2	diag	-80.81 (+0.2%)	-801.61 (+0.2%)	-15194.87 (+0.2%)
TCG2	Skeel	-81.03 (<0.1%)	-803.11(<0.1%)	-15222.53 (<0.1%)
TCG3	-	-81.24 (-0.2%)	-805.20 (-0.2%)	-15265.65 (-0.2%)
TCG3	diag	-81.20 (-0.2%)	-805.26 (-0.2%)	-15268.43 (-0.2%)
TCG3	Skeel	-81.06 (-0.4%)	-803.64 (<0.1%)	-15236.03 (<0.1%)

Table 2.1: **Preconditioners - Polarization energies** in kcal/mol for water systems, with different TCG and preconditioners, using a direct field guess ($\boldsymbol{\mu}_0 = \boldsymbol{\alpha}\mathbf{E}$). The reference results were obtained using CG with a 10^{-8} convergence criterion on the norm of the residual. "Prec" stands for the preconditioner, with "diag" being the diagonal one and "Skeel" the one proposed by Wang and Skeel.[21] Percentages given in brackets are the relative error with respect to the reference.

Looking at table 2.1, and focusing on the TCG computations were no preconditioner was used, the first order of truncation (TCG1) shows a decent agreement with the reference Conjugate Gradient, with less than 10% error overall, eventhough it may appear as a quite rough approximation. Errors drop under 1% when moving to the second order (TCG2), which is already behind the statistical uncertainty from our simulations: this already seems to confirm the early allegation that a low-order truncation is enough to correctly account for the induced polarization. In fact, precision obtained using simply

the second truncation order are accurate enough – and this will be further probed and verified using the various refinements of section 2.2.3 in the next paragraphs – for us to consider that the third order (TCG3) is not even necessary.

On a side note, one could notice that the TCG sometimes gives energies that are lower than the reference, *i.e.* that appear to be more converged (this can be observed for TCG3). This is an artifact coming from the fact that the functional that is minimized during the Conjugate Gradient procedure ($E_{\text{pol}}[\boldsymbol{\mu}] = \frac{1}{2}\langle\boldsymbol{\mu}, \mathbf{T}\boldsymbol{\mu}\rangle - \langle\boldsymbol{\mu}, \mathbf{E}\rangle$) is not exactly the same as what we compute as polarization energy ($-\frac{1}{2}\langle\boldsymbol{\mu}, \mathbf{E}\rangle$).

Table 2.1 also gives a particular insight on the effect of the *preconditioner*. For the first two orders of truncation, the use of the diagonal preconditioner slightly improves the accuracy on the energies[iv]. Skeel's preconditioner, more involved, yields better improvements, as it is most obvious when looking at the TCG1 energies. This is quite expected, as the diagonal preconditioner is a quite poor approximation of the inverse of the polarization matrix, whereas Skeel's version is more complex and precise (see section 2.1.4, equation 2.30).

TCG order	Prec.	S1	S2	S3
TCG1	-	6.3×10^{-3}	7.0×10^{-3}	7.1×10^{-3}
TPCG1	diag	4.9×10^{-3}	5.6×10^{-3}	5.8×10^{-3}
TPCG1	Skeel	2.2×10^{-3}	2.6×10^{-3}	2.7×10^{-3}
TCG2	-	1.7×10^{-3}	1.9×10^{-3}	1.9×10^{-3}
TPCG2	diag	9.2×10^{-4}	1.1×10^{-3}	1.1×10^{-3}
TPCG2	Skeel	3.0×10^{-4}	3.9×10^{-4}	4.2×10^{-4}
TCG3	-	4.7×10^{-4}	5.4×10^{-4}	5.5×10^{-4}
TPCG3	diag	3.8×10^{-4}	3.8×10^{-4}	3.9×10^{-4}
TPCG3	Skeel	6.6×10^{-5}	9.5×10^{-5}	1.0×10^{-4}

Table 2.2: **Preconditioners - Induced dipoles RMS.** A direct field guess was used ($\boldsymbol{\mu}_0 = \boldsymbol{\alpha}\mathbf{E}$). "Prec" stands for the preconditioner, "diag" for the diagonal one, "Skeel" for the one proposed by Skeel[21] .

To provide another measure of the accuracy of the method, we also calculated the RMS (Root Mean Square) error on the induced dipoles. For each frame of the 100 ps simulation, the RMS error between

[iv] This effect should also be the same for the third order (TCG3), but the errors become so small that they fall beyond the statistical uncertainty.

the CG and the TCG induced dipole vector was computed. This RMS error was then averaged over all hundred values, and presented in tables 2.2 and 2.4 for the water systems. In table 2.2, the effect of the preconditioner is even easier to see: for every order of truncation in the TCG, activating a simple preconditioner (the diagonal one) sligltly reduces the RMS error, and this effect is strengthened when using better preconditioners such as Skeel's one.

TCG order	Prec.	Peek	S1	S2	S3
Ref	-	-	-81.03	-803.33	-15229.87
TCG1	-	-	-73.50 (+8%)	-728.73 (+9%)	13813.35 (+9%)
TCG1	-	$\omega = 1$	-81.41 (-0.4%)	-806.83 (-0.4%)	-15315.13 (-0.5%)
TCG1	diag	$\omega = 1$	-79.88 (+1%)	-791.51 (+1%)	-15001.40 (+1%)
TCG1	diag	ω_{opt}	-78.98 (+3%)	-780.94 (+3%)	-14789.04 (+3%)
TCG1	diag	ω_{fit}	-81.06 (<1%)	-803.42 (<0.1%)	-15230.10 (<0.1%)
TCG2	-	-	-80.69 (+0.4%)	-800.32 (+0.3%)	-15173.15 (+0.3%)
TCG2	-	$\omega = 1$	-80.23 (+1%)	-794.49 (+1%)	-15061.22 (+0.1%)
TCG2	diag	$\omega = 1$	-80.98 (<0.1%)	-802.74 (<0.1%)	-15218.27 (<0.1%)
TCG2	diag	ω_{opt}	-80.95 (<1%)	-802.50 (<0.1%)	-15213.17 (+0.1%)
TCG2	diag	ω_{fit}	-81.02 (<1%)	-803.06 (<0.1%)	-15231.14 (<0.1%)
TCG3	-	-	-81.24 (-0.2%)	-805.20 (-0.2%)	-15265.65 (-0.2%)
TCG3	-	$\omega = 1$	-80.78 (+0.3%)	-800.83 (+0.3%)	-15181.55 (+0.3%)
TCG3	diag	$\omega = 1$	-81.03 (<0.1%)	-803.27 (<0.1%)	-15228.74 (<0.1%)

Table 2.3: **Peek-step – Polarization Energies** in kcal/mol, presented for various peek-steps, using a direct field guess ($\boldsymbol{\mu}_0 = \boldsymbol{\alpha}\mathbf{E}$). The reference simulations were done using PCG with a tight convergence criterion. "Prec" stands for the preconditioner, "diag" for the diagonal one. "Peek" designates the use of a peek-step, using a scaling ω reproduced in the corresponding column. Percentage given in brackets are the relative error with respect to the reference.

Table 2.3 focuses on the influence of the *peek-step* (as presented in section 2.2.3). The RMS error on the induced dipole is also reproduced in table 2.4. The choice of a scaling factor ω for the peek-step is far from trivial, as will be discussed in the following paragraphs.

Firstly, results obtained with peek-steps comfort the idea that a low order truncation enable good results, as once again, the polarization energies obtained with TCG1 and TCG2 are in very good agreement

with the PCG reference. The RMS errors, related in table 2.4, also show decreases when using a peek-step.

The use of a Jacobi Over-Relaxation step (JOR), *i.e.* the choice $\omega = 1$ yields improvements in the energies for the first truncation order (TCG1), but this is not the case anymore for TCG2 and 3. Its effect on the RMS, however, is always positive (the error on the induced dipoles is lowered).

The more complex choices ω_{fit} and ω_{opt} seem to have two different roles to play there. ω_{opt} being aimed at a better convergence *asymptotically*, it improves the results on the RMS error, but does not systematically give energies that are better than a JOR step would yield. On the other hand, ω_{fit} is designed for the reproduction of energies, and as such, reproduces them with a high accuracy. However, this comes with a slight cost on the RMS error on the dipoles, that can in fact be higher than when using a $\omega = 1$ or ω_{opt}.

TCG order	Prec	Peek	S1	S2	S3
TCG1	-	-	6.3×10^{-3}	7.0×10^{-3}	7.1×10^{-3}
TCG1	-	$\omega = 1$	3.6×10^{-3}	3.9×10^{-3}	3.7×10^{-3}
TCG1	diag	$\omega = 1$	2.2×10^{-3}	2.6×10^{-3}	2.7×10^{-3}
TCG1	diag	ω_{opt}	2.3×10^{-3}	2.7×10^{-3}	2.8×10^{-3}
TCG1	diag	ω_{fit}	2.6×10^{-3}	3.0×10^{-3}	3.0×10^{-3}
TCG2	-	-	1.7×10^{-3}	1.9×10^{-3}	1.9×10^{-3}
TCG2	-	$\omega = 1$	1.5×10^{-3}	1.7×10^{-3}	1.8×10^{-3}
TCG2	diag	$\omega = 1$	4.1×10^{-4}	5.0×10^{-4}	5.2×10^{-4}
TCG2	diag	ω_{opt}	3.9×10^{-4}	4.6×10^{-4}	4.7×10^{-4}
TCG2	diag	ω_{fit}	5.3×10^{-4}	7.0×10^{-4}	1.0×10^{-3}
TCG3	-	-	4.7×10^{-4}	5.4×10^{-4}	5.5×10^{-4}
TCG3	-	$\omega = 1$	4.6×10^{-4}	4.9×10^{-4}	4.8×10^{-4}
TCG3	diag	$\omega = 1$	1.3×10^{-4}	1.5×10^{-4}	1.6×10^{-4}

Table 2.4: **Peek-step – RMS of the induced dipole vector** compared to the reference for water systems. The peek-step is multiplied by an ω stated in the "Peek" column.

Extension to other systems

This study was also applied to several other systems, much more heterogeneous, in order to investigate on the versatility of the TCG solver. Three solvated proteins were used as test cases, namely

- the nucleocapsid ncp7, containing two Zn^{II} cations, totaling 18 515 atoms (including solvating water molecules as well as counter-ions);
- the dihydrofolate reductase (dhfr), consisting in about 2 500 atoms and surrounded in a seven thousand water molecules droplet;
- the ubiquitin protein, whose nucleic acid chain is made of 1233 atoms, solvated in a 2835 water molecules droplet.

An exemple of ionic liquid was also investigated, the dimethylimidazolium with chlorine counter-ions ($[dmim^+][Cl^-]$). While all other simulations were run at 300 K, this system was thermalized at 415 K, following recommendations in ref. [31].

As observed earlier on water systems, the non-refined TCG1 (*i.e.* the first order TCG without any refinement), or TCG1 with a simple preconditioner, yield errors that can reach 10%. For the other TCG setups tested (higher truncation orders, use of peek-step), the polarization energies presented in table 2.5 show an excellent agreement with the reference values, obtained with a tightly converged Conjugate Gradient. This is a first proof of the *adaptability* of the Truncated Conjugate Gradient.

This good behaviour, very similar to the one obtained on water, is also reproduced when focusing on the RMS errors on the dipoles, as shown in tables 2.6 and 2.7. As observed earlier, the RMS error diminishes as one uses more advanced preconditioners; the distinction made between ω_{fit} and ω_{opt} still holds (ω_{opt} systematically improves the RMS, while ω_{fit} is more suited for enhancing the energy results).

A few preliminary conclusions can be drawn from these simulations. The Truncated Conjugate Gradient appears to give good results, no matter the system under study. Polarization energies stay within a 10% error limit with the first order of truncation, and drop under a few percents when moving to the second order. As one could have expected, a higher truncation order also means a systematic decrease

Solver	Prec.	Peek	ncp7	ubiquitin	dhfr	$[dmim^+][Cl^-]$
Ref	-	-	-24202.54	-11154.87	-28759.01	-1476.79
TCG1	-	-	-21733.63	-9897.22	-25583.50	-1428.35
TCG1	-	$\omega = 1$	-24481.14	-11231.35	-28986.08	-1477.08
TCG1	diag	$\omega = 1$	-23532.73	-10829.84	-27972.41	-1493.58
TCG1	diag	ω_{opt}	-22773.65	-10513.24	-27079.47	-1484.24
TCG1	diag	ω_{fit}	-24161.11	-11162.02	-28766.40	-1479.06
TCG2	-	-	-23922.79	-11031.67	-28463.51	-1420.00
TCG2	-	$\omega = 1$	-23965.96	-11009.06	-28384.49	-1465.73
TCG2	diag	$\omega = 1$	-24123.65	-11128.14	-28683.52	-1471.34
TCG2	diag	ω_{opt}	-23938.70	-11066.44	-28504.96	-1468.29
TCG2	diag	ω_{fit}	-24205.30	-11154.21	-28753.60	-1475.08
TCG3	-	-	-24262.87	-11174.93	-28812.99	-1450.22
TCG3	-	$\omega = 1$	-24121.02	-11105.78	-28635.73	-1441.95
TCG3	diag	$\omega = 1$	-24194.37	-11150.95	-28749.68	-1478.83

Table 2.5: **Polarization Energies** of protein droplet and ionic liquids, using various peek-steps. A direct field guess $\boldsymbol{\mu}_0$ is used here. "Prec" stands for "Preconditioner", "diag" designates the diagonal one. "Peek" deignates the peek-step, using a scaling ω reproduced in the corresponding columns.

of the RMS error on the induced dipoles vector. Lastly, refinements derived in 2.2.3 proved their worth, as the peek-step in particular, appearing as a very efficient and flexible tool.

Dynamics using TCG forces

After these preliminary tests, essential to measure the viability of the Truncated Conjugate Gradient algorithm, we developed the full machinery that allows the computation of the exact forces, as derived in 2.2.4. This implementation was somewhat cumbersome, given the complexity and the number of terms involved, as well as the specificities of our highly parallel framework. Adaptation to the PME framework (sec. 1.5.2) in particular proved to be time-consuming.

Nevetheless, thanks to these efforts, the Truncated Conjugate Gradient became a fully useable method for polarizable molecular dynamics, and we propose here a more extensive assessments of its capacities based on the study of a fourth water system (S4), containing 500 water molecules, for

System	Prec.	ncp7	ubiquitin	dhfr	[dmim+][Cl-]
TCG1	-	8.9×10^{-3}	8.8×10^{-3}	8.8×10^{-3}	1.1×10^{-2}
TCG1	diag	8.6×10^{-3}	8.0×10^{-3}	8.1×10^{-3}	6.9×10^{-3}
TCG1	Skeel	5.5×10^{-3}	4.4×10^{-3}	4.5×10^{-3}	5.6×10^{-3}
TCG2	-	3.5×10^{-3}	3.2×10^{-3}	3.2×10^{-3}	7.2×10^{-3}
TCG2	diag	2.5×10^{-3}	2.0×10^{-3}	2.2×10^{-3}	3.4×10^{-3}
TCG2	Skeel	9.0×10^{-4}	7.7×10^{-4}	7.8×10^{-4}	1.5×10^{-3}
TCG3	-	2.1×10^{-3}	1.7×10^{-3}	1.7×10^{-3}	5.3×10^{-3}
TCG3	diag	7.1×10^{-4}	6.5×10^{-4}	7.2×10^{-4}	7.9×10^{-4}
TCG3	Skeel	2.1×10^{-4}	1.8×10^{-4}	1.9×10^{-4}	3.2×10^{-4}

Table 2.6: **Influence of the preconditioner – RMS of the dipole vector** for our second set of systems. Notations are identical to the previous RMS tables.

which dynamics were carried out using the proper *analytical forces* of the TCG.

The first notable thing here is that the error on the energies is higher than what was obtained using PCG dynamics. This is consistent with the fact that we are not exploring exactly the same potential energy surface. This can also be linked to the various other methods discussed earlier, as some of them were not able to reproduce the reference fully-converged PCG potential energy surface. Nevertheless, the TCG remains viable, as this error drops to a few percents quite easily, either by using the second order TCG, or by using refinements such as the peek-step. A TCG2 using a fitted peek-step (ω_{fit}) yields indeed excellent results, as does the TCG1 when all refinements are activated.

It can also be noted that the error on the fully refined TCG (using a diagonal preconditioner, a direct-field guess and a peek-step), although it remains under 1.5%, is slightly worse for the ω_{fit} peeking. This may be explained by ω_{fit}'s computation which was done, in earlier versions of the code, through a dichotomy and thus with a possible slight loss of precision. Both errors nevertheless stay within the standard deviation, such that it is complicated to seize the importance of this effect. As of now, we have no full certainty and we need to further explore this issue.

Conclusion The first objective behind these computations was to show the applicability of the TCG method, and more specifically for a low truncation order. Here, TCG1 gives a first approximation that is already good given the very cheap computational price paid. TCG2 provides results in very good agree-

Water Box	Prec.	Peek	ncp7	ubiquitin	dhfr	[dmim+][Cl-]
TCG1	diag	$\omega = 1$	4.4×10^{-3}	3.9×10^{-3}	4.1×10^{-3}	3.2×10^{-3}
TCG1	diag	ω_{opt}	5.1×10^{-3}	4.7×10^{-3}	4.8×10^{-3}	3.8×10^{-3}
TCG1	diag	ω_{fit}	4.9×10^{-3}	4.5×10^{-3}	4.6×10^{-3}	4.5×10^{-3}
TCG2	diag	$\omega = 1$	1.7×10^{-3}	1.4×10^{-3}	1.7×10^{-3}	1.6×10^{-3}
TCG2	diag	ω_{opt}	1.3×10^{-3}	1.0×10^{-3}	1.1×10^{-3}	1.9×10^{-3}
TCG2	diag	ω_{fit}	2.2×10^{-3}	1.7×10^{-3}	2.1×10^{-3}	2.0×10^{-3}
TCG3	diag	$\omega = 1$	4.3×10^{-4}	3.8×10^{-4}	4.8×10^{-4}	4.5×10^{-4}

Table 2.7: **Influence of the peek-step – RMS of the dipole vector** compared to the reference, for the inhomogeneous systems, using a peek-step. Notations from previous RMS tables were kept.

ment with the reference ones, and one should note that this remains true when considering systems renowned for being complicated to simulate, such as ionic liquids. TCG3 further refines the results, as one should expect. Yet, given the level of precision already reached using TCG2, especially when using some of the available refinements (preconditioning, peek-step...), the third order does not appear to be necessary.

This versatility of the TCG, exhibiting convergence for any kind of system, is an inheritance from the Conjugate Gradient method: using this algorithm to perform our truncation proves to be the right choice.

Vaporization enthalpies of water

Continuing our tests, and staying with static properties, we will now present *vaporization enthalpy* computations.

The enthalpy of vaporization, also called heat of vaporization, measures the energy required to vaporize a quantity of a substance, that is, to change its state from liquid to gas. It is usually written ΔH_{vap}.

Being a *state function*, the enthalpy difference of a compound between his liquid and gas phase can be calculated following any path we want. If we suppose that the liquid must first break the intermolecular interaction keeping it together, then expand by exerting a pressure on its environment, two terms

Solver	Setup			Energies	
order	Prec.	Guess	Peek-step	Polarization	Potential
Ref	-	-	-	-2198.81	-4490.46
TCG1	-	-	-	-1634.53 (-34%)	-4036.61 (-11%)
TCG1	diag	-	-	-1928.79 (-14%)	-4303.58 (-4.3%)
TCG1	diag	●	-	-1897.15 (-16%)	-4288.86 (-4.7%)
TCG1	-	●	$\omega = 1$	-2083.40 (-5.4%)	-4410.06 (-1.8%)
TCG1	-	-	ω_{fit}	-2032.14 (-8.1%)	-4378.03 (-2.6%)
TCG1	diag	●	ω_{fit}	-2160.03 (-1.7%)	-4459.29 (-0.7%)
TCG2	-	-	-	-2018.17 (-8.8%)	-4384.94 (-2.4%)
TCG2	diag	-	-	-2152.34 (-2.0%)	-4461.86 (-0.6%)
TCG2	-	●	-	-2230.36 (1.5%)	-4521.98 (0.7%)
TCG2	diag	●	-	-2210.11 (0.6%)	-4504.63 (0.3%)
TCG2	-	-	ω_{fit}	-2188.46 (-0.4%)	-4488.46 (<0.1%)
TCG2	-	●	$\omega = 1$	-2186.38 (-0.5%)	-4483.44 (-0.2%)
TCG2	diag	●	$\omega = 1$	-2194.71 (-0.1%)	-4489.88 (<0.1%)
TCG2	diag	●	ω_{fit}	-2169.09 (-1.3%)	-4463.73 (-0.6%)

Table 2.8: **TCG dynamics – 500 water molecules.** This table compiles polarization and potential energies obtained for various setups of the TCG (order, refinements used are varying here). "Prec": preconditioner. Guess: use of the direct field guess ($\boldsymbol{\alpha}\mathbf{E}$). Energies are given in kcal/mol.

arise. Using classical thermodynamic notations, it can be written as

$$\Delta H_{\text{vap}} = \Delta U_{\text{vap}} - p\Delta V \tag{2.57}$$

with ΔU_{vap} measuring the internal energy necessary to overcome the intermolecular interactions

$$\Delta U_{\text{vap}} = U_{\text{gaz}} - U_{\text{liq}} \tag{2.58}$$

and $p\Delta V$ is the work exerted against the external pressure.

Let us come back to our microscopic realm, and now consider quantities per unit (from now on, we

assume that ΔH_{vap} is in energy/mol or energy/molecule). The internal energy of a gas water molecule U_{gaz} can be computed using a single molecule in vacuum, thus not interacting with anything else.

On the other hand, U_{liq} can be computed using a simulation of bulk liquid water (which is the case for our water systems S1-4). We only have to divide the obtained potential energy by the total number of water molecules in the simulation, to be able to compare it with U_{gaz}.

Finally, assuming an ideal gas behaviour for the gas phase water gives us $pV = nRT$, which allows us to rewrite eq. 2.57 as

$$\Delta H_{\text{vap}} = \langle U_{\text{gas}} \rangle - \frac{\langle U_{\text{liq}} \rangle}{N} + RT \tag{2.59}$$

This formula was applied, with $\langle U_{\text{gas}} \rangle$ obtained after a one nanosecond of a single water molecule in gas phase. Table 2.9 presents the heat of vaporization obtained with various TCG options. Simulations were done in the NPT ensemble (we want the pression to remain constant here) using a Monte-Carlo barostat, with at least 100 ps of equilibration and 300 ps of effectively analyzed run.

Method	Prec.	Guess	Peek	Potential energy	ΔH_{vap}	Relative error vs. ref.
Ref				-4628.97	10.74 (± 0.05)	–
TCG1	-	-	-	-4014.6	9.51 (± 0.23)	11%
TCG1	●	-	-	-4442.77	10.36 (± 0.17)	3%
TCG1	-	-	ω_{fit}	-4437,25	10.36 (± 0.18)	3%
TCG1	●	●	$\omega = 1$	-4568.16	10.62 (± 0.05)	1%
TCG1	●	●	ω_{fit}	-4568.79	10.62 (± 0.1)	1%
TCG2	-	-	-	-4557.3	10.60 (± 0.1)	1%
TCG2	●	-	-	-4572.39	10.63 (± 0.12)	1%
TCG2	-	-	ω_{fit}	-4601.38	10.68 (± 0.12)	0,4%
TCG2	●	●	$\omega = 1$	-4628.05	10,73 (± 0,07)	< 0.1 %
TCG2	●	●	ω_{fit}	-4571.78	10.63 (± 0.1)	1%

Table 2.9: **Vaporization enthalpies of water.** All energies are expressed in kcal/mol. Reference is a 400 ps simulation, including 100 ps of equilibration, using the PCG solver for polarization, converged with a (tight) criterion of 10^{-8} on the residual norm. "Prec","Guess" and "Peek" respectively show whether the diagonal preconditioner, the direct field guess, and/or the Peek-step were used or not. The ω value given in the peek column deisgnates the scalar value which multiplies the peek-step (as seen in 2.2.3).

Although the method seems to always underestimate the reference results, agreement becomes re-

ally good, as for previous properties, when considering second order TCG, or when using the refinements such as the peek-step and preconditioner as developed earlier.

One can also notice a slight deterioration of the values when using a peek-step with a fitted ω.

Radial distribution functions

Radial distribution functions (g) measure the variation of density at any distance from a reference particle. It can be simply understood as follow: given a particle i, $g(r)$ gives the probability to find another particle at a distance r of i. Water exhibits strong intermolecular interactions (including hydrogen bonds), which causes strong ordering of the molecules. When considering liquid water, this ordering translates into an equivalent of a solvation radius for water molecules themselves: if one looks at the g_{OO} radial distribution function, measuring the proability to find an oxygen at any distance from another oxygen, a very distinct peak signals that space between neighbouring water molecules is quite well defined. Peaks can be seen at multiples of this average distance, though they quickly vanish, showing that this structural ordering persists over a few molecules. $g_{HH}(r)$, measuring probability to find a hydrogen atom at distance r from another one, also shows a – perhaps even more obvious – peak corresponding to the average distance between the two hydrogen atoms within a water molecules. Being able to reproduce accurately radial distribution functions is thus a good insight on the simulation accuracy.

We reproduced the Oxygen-Oxygen radial distribution functions obtained with TCG1 in figure 2.2, and the other ones, obtained with TCG2, in figure 2.3. In both case, a reference $g(r)$ function was computed using the Conjugate Gradient and reproduced on the figures. For the first truncation order, discrepancies with the reference curves can be observed in particular for the non-refined[v] TCG. The additional use of a diagonal preconditioner brings the curve closer to the reference, yet it is not sufficient to be exact. There again, the peek-step plays a key role, as it reduces almost completely the gap. The "full-option" TCG1, finally, yields a curve which is basically confounded with the reference one.

Looking at TCG2, we now have excellent agreement with the PCG reference, excepted for the non-refined version, seemingly too rough to yield a perfect radial distribution.

[v] with no refinement, *i.e.* no preconditioner, no guess, and no peek-step.

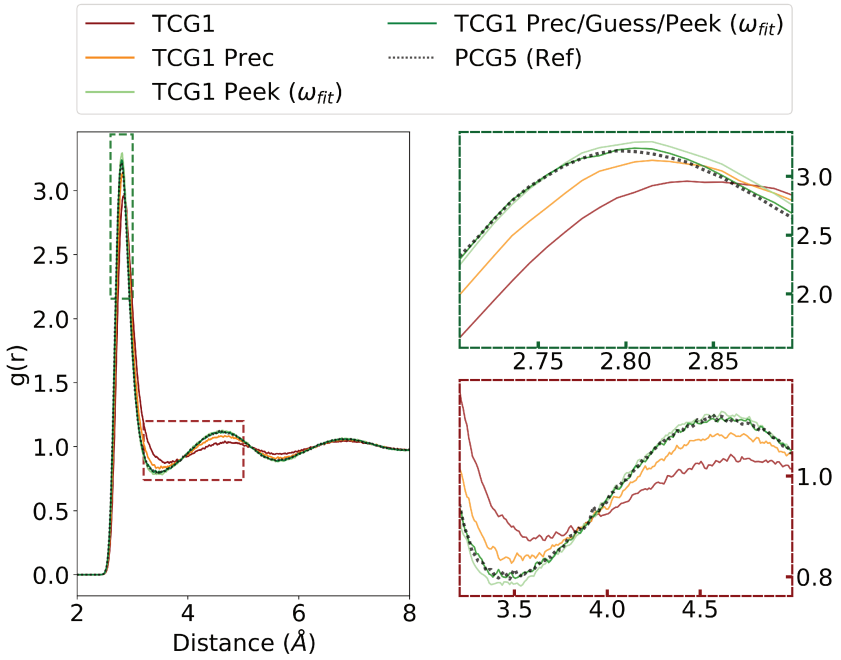


Figure 2.2: **Oxygen-Oxygen radial distribution fonction of water, using various TCG1 setups.**

2.3.2 A dynamical property: the diffusion constant

In its most general definition, the diffusion constant, or diffusion coefficient, is a measure of the drifting of particles in a system under the influence of an external force or gradient.

Starting from the classical (macroscopic) Fick's first law of diffusion, one can describe the flux of particles $\vec{j}$ as

$$\vec{j}(\vec{r},t) = -D\vec{\nabla}n(\vec{r},t) \tag{2.60}$$

with D the diffusion coefficient and n the density.

The conservation of matter links the time variation of the concentration with the exchange of matter with the environment:

$$\frac{\partial n(\vec{r},t)}{\partial t} + \vec{\nabla}.\vec{j}(\vec{r},t) = 0 \tag{2.61}$$

Combining equations 2.60 and 2.61 gives

$$\frac{\partial n(\vec{r},t)}{\partial t} = D\nabla^2 n(\vec{r},t) \tag{2.62}$$

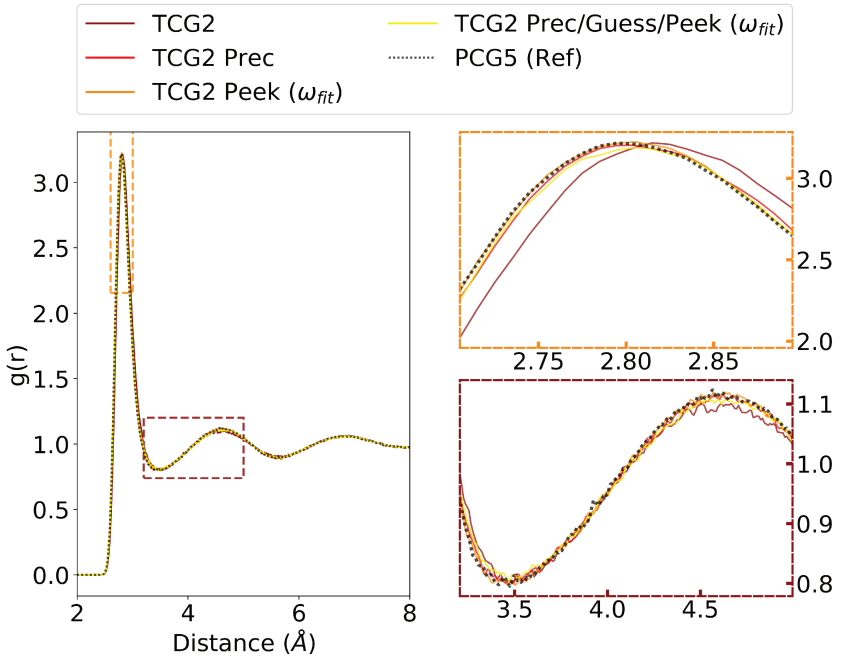


Figure 2.3: **Oxygen-Oxygen radial distribution fonction of water, using various TCG2 setups.**

Noting that the mean square displacement can be computed following

$$\langle \left(\vec{r}(t) - \vec{r}(t_0)\right)^2 \rangle = \int \mathrm{d}^3 r \ \left(\vec{r}(t) - \vec{r}(t_0)\right)^2 n(\vec{r}, t) \tag{2.63}$$

integrating both sides of 2.62 and using Green's theorem, one gets (in three dimensions) the Einstein relationship:

$$D = \frac{1}{6} \frac{\mathrm{d}}{\mathrm{d}t} \langle \left(\vec{r}(t) - \vec{r}(t_0)\right)^2 \rangle \tag{2.64}$$

A different integration leads us to the Green-Kubo expression, reading

$$D = \frac{1}{3} \int_{t_0}^{\infty} \mathrm{d}t \ \langle \vec{v}(t).\vec{v}(t_0) \rangle \tag{2.65}$$

When considering a pure water system, the question essentially boils down to measuring the mean square displacement of the molecules. It can also be called, in this case, a *self-diffusivity* coefficient. A correct diffusion constant is indicative of a proper behaviour of the simulation dynamic-wise. On the opposite, a diffusion coefficient lower than experimental data shows that the movement of particles is dampened, as if the system was more viscous than expected.

Using Einstein's formula, diffusion constants were computed using eq. 2.64, with positions extracted from dynamics. We computed diffusion constants for water *molecules*, as the interesting value here concerns molecule as whole entities, without the intra-atomic movements. Tinker package comes with a set of analysis executables, including one designed for diffusion constant calculations. It uses molecules' centers of mass to compute the average squared displacement.

Diffusion constant were computed for our 500 water molecule system at 300 K, in the NVT ensemble. Maginn[32] reported that using NVE simulations was a better practice, given the possible effect of thermal (and/or pressure) thermostats on the dynamics. The diffusion constants computed, in our case, did not differ in both ensembles.

Solver	Prec	Guess	Peek	D	Error (%)
Ref				1.96 ±0.02	0.0
TCG1	-	-	-	3.29 ±0.04	40.4
TCG1	Diag	-	-	2.37 ±0.02	17.3
TCG1	Diag	●	-	2.22 ±0.03	11.7
TCG1	-	-	ω_{fit}	1.65 ±0.00	-18.7
TCG1	-	●	$\omega = 1$	2.15 ±0.03	8.8
TCG1	Diag	●	ω_{fit}	1.89 ±0.04	-3.7
TCG2	-	-	-	2.16 ±0.02	9.2
TCG2	Diag	-	-	2.01 ±0.01	2.2
TCG2	-	●	-	1.77 ±0.04	-11.1
TCG2	Diag	●	-	1.81 ±0.01	-8.5
TCG2	-	●	$\omega = 1$	1.94 ±0.04	-1.3
TCG2	-	-	ω_{fit}	1.95 ±0.02	-0.4
TCG2	Diag	●	$\omega = 1$	1.91 ±0.01	-2.9
TCG2	Diag	●	ω_{fit}	1.997 ±0.02	1.8

Table 2.10: **Water self-diffusivity constants.** D are given in $\times 10^{-5}$ cm^2/s . "Ref." is the reference, it was computed using a PCG solver, converged with a criterion of 10^{-5} on the residual norm. "Prec","Guess" and "Peek" respectively show whether the diagonal preconditioner, the direct field guess, and/or the Peek-step were used or not. The ω value given in the peek column designates the scalar value which multiplies the peek-step (as seen in 2.2.3).

Simulations were run for 2 ns in total, including 400 ps of equilibration and 1.6 ns of production used to compute the mean square displacements.

Focusing on the results obtained by the TCG, the trend that appeared earlier on static properties seem to be confirmed here: first orders of the TCG method show decent results when used with refinements, and TCG2 show improvements lowering errors to a few percents. However, first order truncation when not refined enough seems to struggle much more when it comes to reproducing dynamic variables. A diffusion constant of 3.3 would indeed vouch for a very "liquid" and mobile water, above the experimental observations $D_{\text{exp}} \simeq 2.29$ (see, amongst many others, [33]).

Nevertheless, a properly used TCG proves to be a method able reproduce correctly not only the static properties of a system, but also the dynamic ones, making it a versatile polarization solver not only in terms of the variety of possible systems, but also the type of studies to be carried out.

Strikingly, when using a preconditioner and a guess, the ω_{fit} peek-step now gives better results than the $\omega = 1$ one, which seems in contradiction with our previous observations regarding these setups.

2.3.3 Parametrization of the peek-step, variations of ω_{fit}

The use of a peek-step scaled with a ω_{fit} proved to be an excellent method to reproduce very accurate energy, as it is specially fitted in this regard. While it is a quite simple tool to use, it deserves a slightly more involved study to better understand its capabilities.

For one, one should monitor the range to which the peek-step is allowed to go. More precisely, it should be noted that definition 2.43 will always give a value for ω_{fit}, including if, for any reason, the error on the induced dipoles were to be important, ω_{fit} could theoretically reach any real value, as large as it may be. One should however remember that these values will have a direct influence on the dipoles as well: should the correction, say, exceed the magnitude of the original $\boldsymbol{\mu}_{\text{TCG}n}$ ones, then the peek-step "correction" would turn into a straight error.

For several systems (the S2 water system containing 216 molecules, the solvated ubiquitin, and the GAG protein[vi]), we plotted the evolution of the RMS error on the induced dipoles (using a tightly converged PCG as a reference) as a function of the value of ω, as well as the subsequent value of the

[vi] GAG is a very small protein; when solvated it builds a system containing about 8000 atoms.

polarization energy. We reprodyce the ubiquitin graph in figure 2.4 (figures 5 and 6 in the Appendix present the water and gag ones). The TCG solver here had no preconditioner, nor guess, activated.

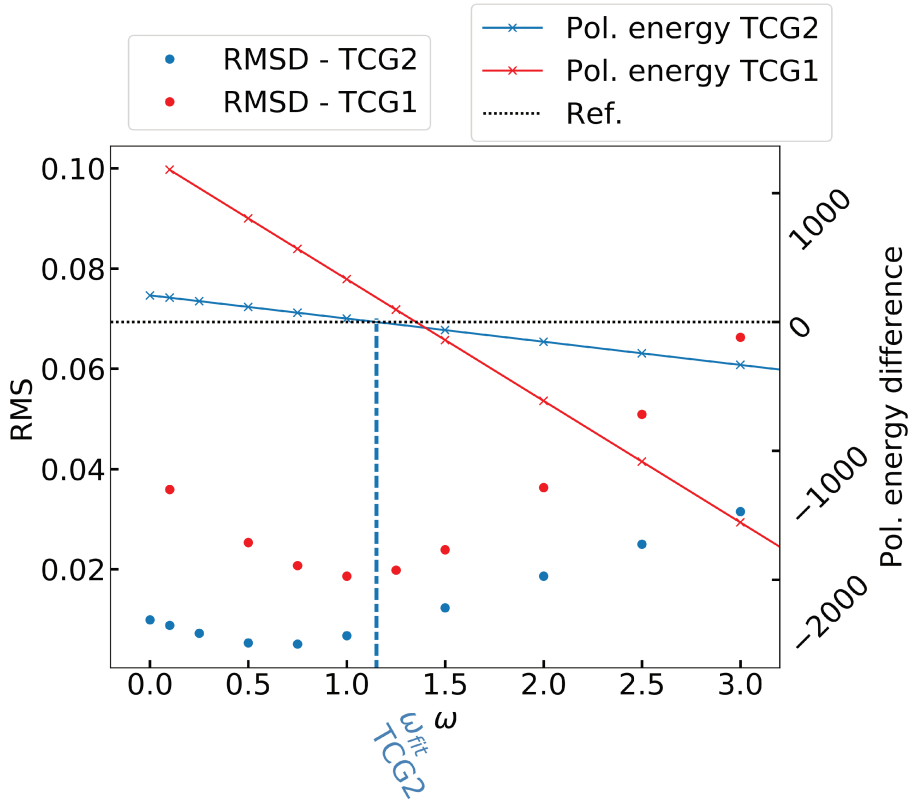


Figure 2.4: **Ubiquitin – Influence of** ω. Polarization energy and RMS on the induced dipoles plotted as a function of ω.

Let us first look at the evolution of the RMS error. For all systems, and each TCG order, a common behaviour can be observed: the general shape of the curve exhibits a minimum located at a particular value ω_{min}.

Meanwhile, if we look at the energy, we have on one hand a black dashed line representing the reference (PCG) polarization energy, and on the other hand, one line for each TCG order that corresponds to the peeked polarization energy. This line represents the affine function ($y = ax + b$) defined by function 2.42, with y the total polarization energy E_{pol}, ω as x, $\frac{1}{2}\langle \boldsymbol{\mu}_{\text{TCG}n}, \mathbf{E} \rangle$ for y-intercept, and $\frac{1}{2}\langle \boldsymbol{\alpha}\mathbf{r}_{n+1}, \mathbf{E} \rangle$ for slope. We can search for the intersection of the black dashed line, corresponding to the reference polarization energy, with a colored line, in order to understand the fitting of the peek-step (at the intersection, $E_{\text{pol, ref}} = E_{\text{pol, peeked}}$). Otherwise put, the ω corresponding to the abscissa of this interaction will be ω_{fit}, as illustrated on figure 2.4.

Comparing the position of ω_{min} with ω_{fit} firstly shows that these two values are different, as the results obtained earlier foreshadowed. But it also shows that these two value are quite close to each other, on both well- and ill-conditioned systems (water as well as solvated proteins). This shows us that ω_{fit} fitting does not yield crazily high (or low) values, at least for the various tested systems.

During these simulations, ω_{fit} was refitted every 100 time-step, in order to evaluate the variation it undergoes over the course of the simulation. We observed that these variations depend on the level of convergence that the polarization solver reached before the peek-step is applied. Explicitly, if the difference between TCG and the reference is substantial regarding the polarization energy, as it would be the case for a first order TCG with little to no refinement, ω_{fit} remains very stable along the simulation. On the contrary, when using a very accurate TCG version (second order using all refinements for example), ω_{fit} has a rather important range of variation that can reach ± 0.6.

This could be explained by the importance of the correction required: when using a TCG1 with no refinement, the error between TCG's polarization energy and the fully converged CG reference is rather important, such that the scaling of the contribution to the energy from the peek-step remains stable. On the other hand, when using a fully-refined TCG2, the error between TCG's polarization energy (before the peek-step is used) and the reference polarization energy is much smaller (see for example the 0.6% of error in table 2.8 for the TCG2 using a preconditioner and a guess). The correction required from the peek-step is thus much smaller, and hence more sensitive to the small fluctuations of TCG's polarization energy. This could explain the higher fluctuation of ω_{fit}.

This reasoning is only possible providing that there is no brutal change to the system itself, that is to say that the peek-step energy contribution $\langle \boldsymbol{\mu}_{\text{peek-step}}, \mathbf{E} \rangle$ does not change much along the simulation.

This analysis is empirical and would need further tests, and is only here as a proposed explanation.

An excellent precision on the energies can be reached by the use of a fitted peek-step (with ω_{fit}), as showed in the previous sections. Monte-Carlo simulation methods are based on non-physical moves whose likeliness are controlled by the change of energy they would cause. As such, their accuracy are very closely depending on the precision on the evaluation of the energy of a configuration. For this reason, TCG refined with a fitted peek-step thus appears as a method of choice for such simulations.

This becomes even clearer when considering the timings, given that no gradients have to be computed to carry out Monte-Carlo simulations. The TCG would thus be an ideal candidate for such exper-

iments, especially when considering badly conditioned systems (such as ionic liquids), since they yield the most impressive speedups.

This adaptability to the Monte-Carlo scheme comes from the fact that TCG does not need to stay within a Molecular Dynamics framework. Other methods such as the extended Lagrangian ones (as seen in section 2.1.2) do require a full MD framework to function, and can thus not be used as Monte-Carlo polarization solvers.

2.3.4 Timings

We are left with one of the most important of TCG's properties to assess: its computational speed. To that end, we performed simulations of a thousand time-steps using various setups of the solver, and measured for each the production in ns/day. By fixing the timestep length to 1 fs for all the simulations, the number of ns/day is directly the number of timestep multiplied by the time-step length δt. We thus have a direct measurement of the computational time spent per time/step, or equivalently, of the speed of the computation.

Simulations on the system S4, containing 500 water molecules, were run on a node of 24 processors. Simulations for the ubiquitin and [dmim$^+$] systems were run on two nodes of 24 processors, *i.e.* 48 processors total. Results were compiled in table 2.11, with a PCG solver converged to 10^{-5} as reference. This PCG reference was computed *without* using the ASPC guess presented in section 2.1.4, in order to compare methods with good time-reversibility and volume preserving properties.

Solver	Refinements			Water (S4)		Ubiquitin		$[dmim^+][Cl^-]$	
order	Prec.	Guess	Peek	ns/day	diff.	ns/day	diff.	ns/day	diff.
Ref.				3.06	0.0	0.58	0.0	1.57	0.0
PCG8				2.25	-26.6	0.40	-31.6	1.11	-29.1
TCG1	-	-	-	4.68	52.9	1.16	96.3	3.21	104.2
TCG1	●	-	-	4.65	51.8	1.16	97.8	3.21	104.2
TCG1	-	●	-	3.67	19.9	0.8	37.0	2.15	36.9
TCG1	-	-	ω_{fit}	4.01	31.1	0.84	43.8	2.24	42.5
TCG1	●	●	$\omega = 1$	3.53	15.4	0.79	35.6	2.13	35.7
TCG1	●	●	ω_{fit}	3.57	16.6	0.77	32.6	2.08	32.5
TCG2	-	-	-	3.89	27.2	0.89	52.8	2.47	57.5
TCG2	●	-	-	3.86	26.1	0.89	52.0	2.45	55.8
TCG2	-	●	-	2.98	-2.6	0.63	7.9	1.68	7.0
TCG2	●	●	-	3.03	-0.9	0.63	7.6	1.68	7.2
TCG2	-	-	ω_{fit}	3.35	9.6	0.64	9.6	1.72	9.6
TCG2	●	●	$\omega = 1$	2.88	-5.9	0.59	1.7	1.57	0.1
TCG2	●	●	ω_{fit}	2.83	-7.6	0.58	-0.5	1.55	-1.6

Table 2.11: **Timings for various TCG setups.** For each simulation, 1000 1 fs time-steps were run. Simulations on the water system S4 were performed using 24 cores; ubiquitin and the dimethylimidazolium solution with 48 cores. All timings are given in ns/day. The reference timings were obtained using a PCG solver with a 10^{-5} convergence criterion. "PCG8" stands for the Preconditioned Conjugate Solver with a 10^{-8} convergence criterion. "Diff." designates the relative difference, in percents, with respect to the reference.

Firstly, one can note that the simpler the method is, the faster the acceleration obtained: the best speedups are obtained when using non-refined TCG1, while the fully-refined TCG2 can yield slower dynamics than the PCG reference.

This may seem counter-intuitive, but it arises from the expressions of the gradients that were derived in the previous section. The rewriting of the derivatives, a vital point in the feasability of the TCG, can indeed require up to two extra matrix-vector products (one accounting for the use of a guess, as showed by equation 2.39, another for the supplementary terms that the peek-step requires).

A second general observation concerns the variation of performance depending on the systems in consideration. Indeed, one can note that the performance gains are systematically higher when looking at the most heterogeneous systems (for example, TCG1 accelerates the water system's simulation by slightly more than 50%, and the ionic liquid's by more than a hundred).

This can be explained by the conditioning of the polarization matrices (see eq. 2.25) which would be better when looking at a homogeneous system. This means that, for a given level of convergence, a greater number of Conjugate Gradient iterations would be required when calculating the induced dipoles vector on a heterogeneous system. When compared to the *fixed* number of iterations that TCG requires, the difference will thus be more important when looking at the inhomogeneous systems; the speedup provided by TCG will hence be larger.

From a more practical point of view, several simple guidelines can be drawn from these results. The diagonal preconditioner that was implemented has a really limited impact on the timings, while it allows a substantial improvement of the simulation's qualities (as shown by the diffusion coefficient and the computed energies). Thus, if one wishes to use a really cheap method, the best way to go is to use the diagonal preconditioner, without any guess or peek-step, baring in mind that not using the preconditioner would barely mean any savings in computation time.

On another hand, the fully refined TCG offers no effective speedup as is; and for simple simulations, we would suggest to use a peek-step with a fitted ω. We showed in the previous sections the quality of the simulations obtained with this options, and we now observe that this still ensures a roughly 10% speedup.

The timings presented were computed using a quite standard 1 fs time-step. Sensibly equivalent

results can be observed when using a RESPA integrator with a 2 fs time-step (see Appendix). However, one should keep in mind that this comes with an important improvement regarding the polarization forces, which are now exactly consistent with the polarization energy. This means, as explained in 2.2.1, that there should be no drift arising from polarization in the simulation.

This increase in stability is a direct advantage to explore integration using *larger time-steps*, and this becomes even more critical as usual accelerations, such as the ASPC, are not viable when exceeding the 2 fs limit. The applicability of the Truncated Conjugate Gradient to large time-step methods will be explored extensively in chapter 4.

2.3.5 TCG: a first conclusion

To sum up this first study on the performances and viability of the Truncated Conjugate Gradient, a few main points should be put forth:

- As always in computational experiment, a good balance has to be found between numerical accuracy and computation time. Thanks to the many different setups of the TCG, the user can finely tune the solver to his will in order to control this balance.
- The diversity of solver setups is more generally a good sign that TCG could be used in various roles, as the following chapter will illustrate.
- The stability of the method allows one to consider larger time-steps,[34] which could yield decisive accelerations, while the commonly used method are still limiting in this regard.

Looking at the various results, the TCG2 using a fitted peek-step (with ω_{fit}) appears as a setup of choice, with a very good cost effectiveness. It indeed provides speedups and a reasonably easy implementation, avoiding the use of a guess and its supplementary matrix-vector product, and ensures very good precision.

Finally, it is worth noting that the reference we used throughout this chapter (a Preconditioned Conjugate Gradient with a 10^{-5} convergence threshold on the residual norm) is known to yield drift in the total energy of the simulation, thus distorting the dynamics. If we wanted to reach the stability allowed by the TCG, this criterion should be tightened to 10^{-8} (as demonstrated by Lipparini *et al.*[1]).

Indeed, the drift produced by a simulation where the convergence criterion is fixed at 10^{-5} disappears in the short term, but will accumulate on larger time-scales, effectively preventing the simulation of long dynamics. In the 10^{-8} case, more iterations of the Conjugate Gradient are required, effectively slowing down the computations. Tables 2.11 (and 10 in Appendix) show this slowdown. The speedups allowed by TCG are much more interesting compared to this tighter reference: even the fully refined version becomes 25 to 45% faster (depending on the system) ! When considering long simulations, TCG is thus definitely a much faster choice.

Perspectives

Amongst the possible improvements proposed to further exploit TCG's potential, one could propose a splitting of the physical system in "zones" of different conditioning of the **T** matrix, that should be treated with different precision levels (as for QM-MM, where a subset is treated using quantum mechanics while its environment is treated classically). The fact that several orders of the TCG have been developed could indeed lead to a divisions of the system with a more precise polarization treatment of the most sensitive zones. One could for example imagine a protein treated using TCG2 while the bulk water is described using TCG1 only.

The question of linking these two (or more) domains, especially given the complexity of the equations driving the polarization, can be answered using a mathematical tool called the *Schur complement*. By rewriting the polarization matrix, it allows for an approximation of its inverse under certain conditions (see [15] for more details). One could compute the induced polarization using TCG1, then, using the Schur complement, obtain an approached value of the TCG2 order for the particles of interest (*e.g.* a protein's active site). However, the possible gains, in terms of accuracy and computation speed, remain marginal. It was thus decided not to spend time implementing it, in order to focus on more efficient acceleration strategies.

The reader could note that, more than a single algorithm, the Truncated Conjugate Gradient actually defines a family of algorithms with increasing computational cost and accuracy. Up to now, only the first few order have been extensively used and studied, and proved to be adequate for a wide variety of systems. Yet, if there was ever the need for a higher order truncation, its derivation should remain straightforward, although quite involved in terms of formulae. Such further developments may

be implemented using automatic differentiation[35] in order to ease the coding task, as the gradients expression will progressively get more and more complex.

Moreover, this family of algorithms and its various setups span a large choice of polarization solvers with specific acceleration and accuracy properties, such that one could imagine having various "golden" TCG standards for different systems.

The Truncated Conjugate Gradient concept was first introduced in an article of the Journal of Chemical Theory and Computation, reproduced hereafter.

JCTC Journal of Chemical Theory and Computation

Article

pubs.acs.org/JCTC

Truncated Conjugate Gradient: An Optimal Strategy for the Analytical Evaluation of the Many-Body Polarization Energy and Forces in Molecular Simulations

Félix Aviat,[†] Antoine Levitt,[‡] Benjamin Stamm,[¶,§] Yvon Maday,[∥,⊥,#] Pengyu Ren,[■] Jay W. Ponder,[△] Louis Lagardère,[*,▽,†] and Jean-Philip Piquemal[*,†,⊥]

[†]Laboratoire de Chimie Théorique, UPMC Université Paris 06, UMR 7617, F-75005, Paris, France

[‡]Inria Paris, F-75589 Paris Cedex 12, Université Paris-Est, CERMICS (ENPC), Marne-la-Vallée, F-77455, France

[¶]MATHCCES, Department of Mathematics, RWTH Aachen University, Schinkelstraße 2, D-52062 Aachen, Germany

[§]Computational Biomedicine, Institute for Advanced Simulation IAS-5 and Institute of Neuroscience and Medicine INM-9, Forschungszentrum Jülich, Jülich, 52425, Germany

[∥]Laboratoire Jacques-Louis Lions, UPMC Université Paris 06, UMR 7598, F-75005, Paris, France

[⊥]Institut Universitaire de France, Paris Cedex 05, 75231, France

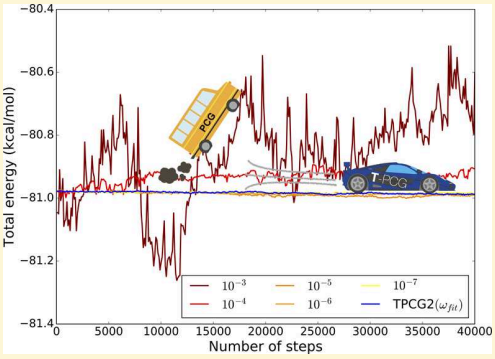
[#]Division of Applied Maths, Brown University, Providence, Rhode Island 02912, United States

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

[△]Department of Chemistry, Washington University in Saint Louis, Campus Box 1134, One Brookings Drive, Saint Louis, Missouri 63130, United States

[▽]Institut du Calcul et de la Simulation, UPMC Université Paris 06, F-75005, Paris, France

ABSTRACT: We introduce a new class of methods, denoted as Truncated Conjugate Gradient(TCG), to solve the many-body polarization energy and its associated forces in molecular simulations (i.e. molecular dynamics (MD) and Monte Carlo). The method consists in a fixed number of Conjugate Gradient (CG) iterations. TCG approaches provide a scalable solution to the polarization problem at a user-chosen cost and a corresponding optimal accuracy. The optimality of the CG-method guarantees that the number of the required matrix-vector products are reduced to a minimum compared to other iterative methods. This family of methods is non-empirical, fully adaptive, and provides analytical gradients, avoiding therefore any energy drift in MD as compared to popular iterative solvers. Besides speed, one great advantage of this class of approximate methods is that their accuracy is systematically improvable. Indeed, as the CG-method is a Krylov subspace method, the associated error is monotonically reduced at each iteration. On top of that, two improvements can be proposed at virtually no cost: (i) the use of preconditioners can be employed, which leads to the Truncated Preconditioned Conjugate Gradient (TPCG); (ii) since the residual of the final step of the CG-method is available, one additional Picard fixed point iteration ("peek"), equivalent to one step of Jacobi Over Relaxation (JOR) with relaxation parameter ω, can be made at almost no cost. This method is denoted by TCG-n(ω). Black-box adaptive methods to find good choices of ω are provided and discussed. Results show that TPCG-3(ω) is converged to high accuracy (a few kcal/mol) for various types of systems including proteins and highly charged systems at the fixed cost of four matrix-vector products: three CG iterations plus the initial CG descent direction. Alternatively, T(P)CG-2(ω) provides robust results at a reduced cost (three matrix-vector products) and offers new perspectives for long polarizable MD as a production algorithm. The T(P)CG-1(ω) level provides less accurate solutions for inhomogeneous systems, but its applicability to well-conditioned problems such as water is remarkable, with only two matrix-vector product evaluations.



1. INTRODUCTION

In recent years, the development of polarizable force fields has led to new methodologies incorporating more physics. Therefore, higher accuracy in the evaluation of energies can be achieved.[1] Indeed, the explicit inclusion of the many-body polarization energy offers a better treatment of intermolecular interactions, with immediate applications in various fields of application ranging from biomolecular simulations to material science. However, adding polarization to a force field is

Received: October 6, 2016
Published: November 22, 2016

DOI: 10.1021/acs.jctc.6b00981
J. Chem. Theory Comput. 2017, 13, 180−190

associated with a significant increase of the overall computational cost. In that context, various strategies have been introduced, including Drude oscillators,[2] fluctuating charges,[3] Kriging methods,[4] and induced dipoles.[1,5] Among them, the induced dipole approach has been shown to provide a good balance between accuracy and computational efficiency, and can be implemented in a scalable fashion.[6]

One issue with this approach is the mandatory resolution of a set of linear equations the size of which depends on the number of atoms (or polarizable sites). In practice, for the large systems of interest of force fields methods, a direct matrix inversion approach using the LU or Cholesky decomposition is not computationally feasible because of its cubic cost in the number of atoms. Luckily, iterative methods provide a remedy. We showed in a recent paper[6,7] that techniques such as the Preconditioned Conjugate Gradient (PCG) or the Jacobi/Direct Inversion of the Iterative Subspace (JI/DIIS) were efficient for large scale simulations as they offer the possibility of a massively parallel implementation coupled to fast summation techniques such as the Smooth Particle Mesh Ewald (SPME).[8] The overall cost is then directly proportional to the number of iterations necessary to achieve a good convergence. In that context, predictor-corrector strategies have been introduced to reduce this number using the information on the previous time-steps.[9,10] Extended Lagrangian formulations inspired by efficient ab initio methods have also been introduced in order to limit the computational cost, but they require additional thermostats.[11] In practice, iterative methods are now standard but suffer from energy conservation issues due to their nonanalytical evaluation of the forces. Moreover, force fields are optimized to reach a precision for 10^{-1} to 10^{-2} kcal/mol in the polarization energy. Such a precision can easily be reached using a convergence threshold of 10^{-3} to 10^{-4} Debye on the induced dipoles. However, when using iterative schemes, one needs to enforce the quality of the nonanalytical forces in order to guarantee the energy conservation. Hence, a tighter convergence criterion of 10^{-5} to 10^{-7} Debye must be used for its computation. This leads to a very significant increase of the number of iterations. Overall, this additional computational cost is not linked to the accuracy of the polarization energy but only ensures the numerical stability of the MD scheme. In that context, in their 2005 seminal paper[12] (see also ref. 13), Wang and Skeel postulated that another strategy would be possible if one could offer a method allowing analytical derivatives and therefore avoiding by construction the risk of loss of energy conservation (i.e. the drift). Such a method would be associated with a fixed number of iterations and could extend the applicability of polarizable simulations. Wang explored such strategies based on modified Chebyshev polynomials but noticed that even if the intended analytical expression was obtained, it offered little accuracy compared to fully converged iterated results. In that context, Simmonett et al.[14,15] recently proposed to revisit this assumption of a perturbation approach evaluating an approximated polarization energy denoted as ExPT. They proposed a parametric equation offering analytical derivatives and a good accuracy for some class of systems. However, the parametric aspect of the approach limits its global applicability to other types of systems. The purpose of this paper is to introduce a nonempirical strategy based on the same goals: analytical derivatives in order to guarantee energy conservation, limited number of iterations and reasonable accuracy.

We will first present the variational formulation of the polarization energy and the associated linear system. Then, we will look at iterative methods that are commonly used to solve it and discuss how they can cause problems in molecular simulations. Following this, we will describe two classes of iterative methods, the fixed point methods and the Krylov methods, and see how one can compute the polarization energy and its associated forces analytically (therefore avoiding the energy drift mentioned above). Finally, we will show some numerical results and discuss the accuracy of the new methods.

2. CONTEXT AND NOTATIONS

In the context of force fields, several techniques are used in order to take polarization into account. Everything that will be presented in this paper concerns the widely used induced dipole model. In this model, each or some of the atomic sites are associated with a 3×3 polarizability tensor, so that induced dipoles appear on these sites because of the electric fields created by the permanent charge density and by the other induced dipoles.

2.1. Notations. In the rest of the paper, we will assume that the studied system consists of N atoms, each of them bearing an invertible 3×3 polarizability tensor α_i. We will denote by $\vec{E}_i$ the electric field created by the permanent density of charge on site i, and by $\vec{\mu}_i$ the induced dipole on site i. The $3N$ vectors collecting these vectors will respectively be noted $\mathbf{E}$ and $\boldsymbol{\mu}$. Furthermore, for $i \neq j$, we will denote by T_{ij} the 3×3 tensor representing the interaction between the ith and the jth induced dipole, so that $T_{ij}\vec{\mu}_j$ is the (possibly damped) electric field created by $\vec{\mu}_j$ on site i. We are now able to define by blocks the so-called polarization matrix of the system block by block:

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

This matrix is symmetric and we assume that it is also positive definite (thanks to the damping of the electric fields at short-range) so that the energy functional defined below has a minimum and "the polarization catastrophe"[16] is avoided.

2.2. Variational Formulation of the Polarization Energy and the Associated Linear System. Given these notations, one can express the polarization energy of the studied system in the context of an induced dipole polarizable force field as follows:

$$E_{\text{pol}} = \frac{1}{2}\boldsymbol{\mu}^T \mathbf{T}\boldsymbol{\mu} - \boldsymbol{\mu}^T \mathbf{E} \tag{1}$$

The dipoles $\boldsymbol{\mu}$ of the quadratic energy functional are determined by the first optimality condition in the form of the following linear system:

$$\mathbf{T}\boldsymbol{\mu} = \mathbf{E} \tag{2}$$

so that finally:

$$E_{\text{pol}} = -\frac{1}{2}\boldsymbol{\mu}^T \mathbf{E} \tag{3}$$

for the minimizing dipoles $\boldsymbol{\mu}$. The linear system expressed above has to be solved at each time step of a MD trajectory, so that a computationally efficient technique must be used to solve it. Two kinds of methods exist to solve a linear system, the direct ones and the iterative ones. The first approaches, such as the LU or

DOI: 10.1021/acs.jctc.6b00981
J. Chem. Theory Comput. 2017, 13, 180−190

Cholesky decomposition, yield exact results (up to round-off errors) but their computational cost grows proportionally to N^3 and their memory requirements proportionally to N^2, making them hardly usable for large systems of biological interest.

3. ITERATIVE METHODS

In contrast, iterative techniques yield approximate results depending on a convergence criterion, but their computional cost is proportional to the number of iterations times the cost of one iteration (dominated by the cost of a matrix-vector product). This implies that the iterative techniques can be used in conjunction with an efficient matrix-vector multiplication method such as the Smooth Particle Mesh Ewald or the Fast Multipoles.[8,17]

Several issues arise when using an iterative method to solve the polarization energy. The first one is related to the way the associated forces are computed. Indeed, the polarization energy is a function of the induced dipoles and of the atomic positions, so that one can rely on the chain rule to express the total derivative of this energy with respect to the atomic positions. The induced dipoles are then assumed to be completely minimizing E_{pol} so that $\frac{\partial E_{\mathrm{pol}}}{\partial \boldsymbol{\mu}}$ is assumed to be zero, yielding the following expression (that is analogous to the Hellman−Feynman theorem in quantum mechanics):

$$\frac{\mathrm{d}E_{\mathrm{pol}}}{\mathrm{d}r_i} = \frac{\partial E_{\mathrm{pol}}}{\partial \boldsymbol{\mu}}\frac{\partial \boldsymbol{\mu}}{\partial r_i} + \frac{\partial E_{\mathrm{pol}}}{\partial r_i} = \frac{\partial E_{\mathrm{pol}}}{\partial r_i} \tag{4}$$

As the iterative method for the resolution of the induced dipoles is never perfectly converged, the previous assumption is never perfectly satisfied. Consequently, the forces calculated using this method are not exactly the negative of the first derivative of E_{pol} (eq 3) with respect to the nuclear positions, potentially giving rise to an energy drift in a MD simulation. This is illustrated by the following graph (Figure 1) representing the evolution of the total energy for a water box of 27 molecules, using the (diagonally) PCG with different convergence thresholds, namely 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, and 10^{-7}. An initial guess not issued from the past iterations was used, for a short MD simulation of 10 ps, using a time step of 0.25 fs. Such a small time step was used in order to minimize the numerical error coming from time-integration. One can directly observe the relation between the convergence threshold and the energy conservation.

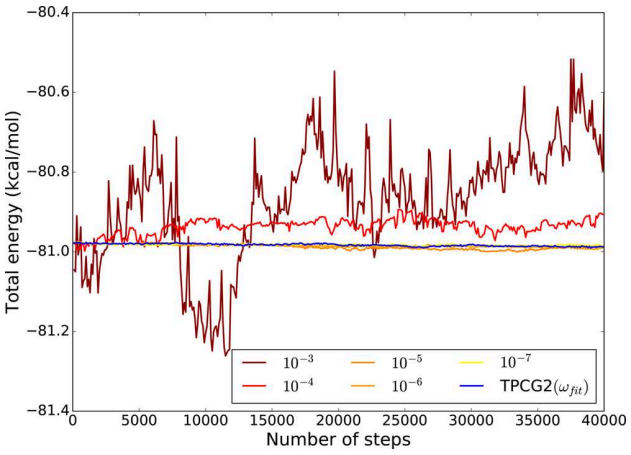


Figure 1. Evolution of the total energy of a water box of 27 molecules computed with PCG and different convergence thresholds (AMOEBA), and with the TPCG2(ω_{fit}) method, with P = diag. The drift on the total energy is fully related to the polarization contribution, the TPCG2(ω_{fit}) converges to the 10^{-7} PCG results without any drift.

The second issue is the computational cost of the iterative methods, proportional to the number of iterations times the cost of one iteration. Solving the polarization equations costs usually (depending on the settings of the simulation) more than 60% of the total cost of an MD step. It has already been shown that carefully choosing the iterative techique employed and taking an initial guess $\boldsymbol{\mu}_0$ using information from the past (by using predictor guesses[9,10]) can yield an important reduction of the number of iterations required to reach a satisfactory convergence threshold. Nevertheless, some limitations exist due to the imperfect time reversibilty and volume preservation that they imply. Furthermore, the ability to parallelize the method efficiently also influences the choice of the optimal method.[6,7]

These issues motivate the derivation of a computationally cheap analytical approximation of the polarization energy in polarizable MD. We aim also for such an approximation to be as close as possible to the exact results (or at least within the accuracy of the force field) so that it would not require a reparametrization of the force fields. For the forces to be analytical, the computation of the induced dipoles must be history free and should therefore avoid the use of predictors.

4. FIXED POINT METHODS AND RELATION WITH ExPT

This first class of methods regroups the fixed point methods, also called stationary methods. In this case, one splits the matrix into two parts in order to re-express the solution of the linear system as a fixed point of a mapping associated with the splitting. For the polarization matrix one can re-express $\mathbf{T}$ as the sum of its (block-) diagonal and off-diagonal part:

$$\mathbf{T} = \boldsymbol{\alpha}^{-1} - \mathcal{T} \tag{5}$$

yielding the following expression of the solution $\boldsymbol{\mu}$:

$$\boldsymbol{\mu} = \boldsymbol{\alpha}(\mathbf{E} + \mathcal{T}\boldsymbol{\mu}) \tag{6}$$

where $\boldsymbol{\mu}$ appears as the fixed point of a mapping. Then, Picard's fixed point theorem[18] tells us that starting from any guess $\boldsymbol{\mu}_0$ and computing the following sequence of dipoles (denoted by $\mathbf{r}_n$ the residual associated with $\boldsymbol{\mu}_n$):

$$\boldsymbol{\mu}_{n+1} = \boldsymbol{\alpha}\mathbf{E} + \boldsymbol{\alpha}\mathcal{T}\boldsymbol{\mu}_n = \boldsymbol{\mu}_n + \boldsymbol{\alpha}\mathbf{r}_n \tag{7}$$

we converge toward the solution if $\rho(\boldsymbol{\alpha}\mathcal{T}) < 1$, with $\rho(\mathbf{M})$ denoting the spectral radius of a given matrix $\mathbf{M}$. The method that was described above is the Jacobi method and if we had split the matrix between its upper triangular part and the remaining terms, we would have obtained the Gauss−Seidel method.

A direct refinement of the Jacobi method consists in choosing a "relaxation" parameter ω and the following (relaxed) scheme:

$$\boldsymbol{\mu}_{n+1} = (1-\omega)\boldsymbol{\mu}_n + \omega(\boldsymbol{\mu}_n + \boldsymbol{\alpha}\mathbf{r}_n) = \boldsymbol{\mu}_n + \omega\boldsymbol{\alpha}\mathbf{r}_n \tag{8}$$

which is convergent if $\rho(I_d - \omega\boldsymbol{\alpha}\mathbf{T}) < 1$. In the rest of the text we will denote this method as JOR (Jacobi over Relaxation).[19,20]

One way to get analytical approximations of the polarization energy is to truncate one of these methods at a fixed order. One could for example choose to truncate the Jacobi method at some order n to obtain an analytical approximation to the solutions of the induced dipoles which we rearrange as:

$$\boldsymbol{\mu}_n = \boldsymbol{\mu}_{(0)} + \boldsymbol{\mu}_{(1)} + \ldots + \boldsymbol{\mu}_{(n)} \tag{9}$$

with

DOI: 10.1021/acs.jctc.6b00981
J. Chem. Theory Comput. 2017, 13, 180−190

$$\boldsymbol{\mu}_{(n)} = \boldsymbol{\alpha}(\mathcal{T}\boldsymbol{\alpha})^n\mathbf{E} \tag{10}$$

which is exactly the formulation of the perturbation theory (PT) method proposed by Simmonett et al.,[14] even though they follow another reasoning related to perturbation theory. The ExPT method that they propose is then made by truncating this expansion at order two and by using a linear combination of $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_3$

$$\boldsymbol{\mu}_{\mathrm{ExPT}} = c_0\boldsymbol{\mu}_0 + c_1\boldsymbol{\mu}_3 \tag{11}$$

in order to provide the following expression for the approximation of the polarization energy:

$$E_{\mathrm{pol,ExPT}} = -\frac{1}{2}\boldsymbol{\mu}_{\mathrm{ExPT}}^{T}\mathbf{E} \tag{12}$$

The computational cost of this method is then equivalent to making three matrix-vector multiplication and its accuracy is good in many cases but it has the disadvantage of using two parameters that need to be fitted. Simmonett and co-workers recently extended the ExPT technique to higher orders, giving the OPTn class of methods,[15] that lead to improved results but require even more empirical parameters.

5. KRYLOV METHODS: PRECONDITIONED CONJUGATE GRADIENT

The point of view of the Krylov methods is completely different.[21] It consists in minimizing some energy functional at each iteration over some growing subspaces.

Starting from some initial guess $\boldsymbol{\mu}_0$, let us define the associated residual:

$$\mathbf{r}_0 = \mathbf{E} - \mathbf{T}\boldsymbol{\mu}_0 \tag{13}$$

We are now able to define the so-called Krylov subspaces of order $p \in \mathbb{N}$:

$$K_p = \mathrm{span}(\mathbf{r}_0, \mathrm{Tr}_0, ..., \mathbf{T}^{p-1}\mathbf{r}_0) \tag{14}$$

We clearly have the following inclusion of spaces:

$$K_1 \subseteq K_2 \subseteq ... \tag{15}$$

Then, μ_n is determined as the minimum of the energy functional over $\boldsymbol{\mu}_0 + K_p$. As one is minimizing at each iteration the energy functional over some increasing sequence of embedded spaces, the error (as measured by the functional) is necessarily decreasing. One can show that there exists a $p \leq 3N$ such that the exact solution $\boldsymbol{\mu}$ belongs to $\boldsymbol{\mu}_0 + K_p$, meaning that these methods always converge and even provide the exact solution after a finite number of steps, the worst case being when they converge in $3N$ iterations. In practice, however, only very few iterations are needed to obtain accurate solutions.

The different Krylov methods are determined by the quantity that is minimized over the Krylov subspaces: if one minimizes E_{pol} then one obtains the conjugate gradient (given the assumption that $\mathbf{T}$ is symmetric definite positive), if one minimizes $\|\mathbf{r}_n\|_{l^2}$ then one gets the GMRES method[21] (which is equivalent to some version of the JI/DIIS[22]). But many other methods exist, such as the Minres algorithm[23] or the BiCG method[21] for nonsymmetric matrices.

The conjugate gradient algorithm updates three vectors at each iteration: a descent direction, a dipole vector, and the associated residual. These vectors are updated using three scalars that are obtained by making some scalar products over these vectors. After the following initialization (using here the direct field $\boldsymbol{\alpha}\mathbf{E}$ as an initial guess):

$$\begin{cases} \boldsymbol{\mu}_0 = \boldsymbol{\alpha}\mathbf{E} \\ \mathbf{r}_0 = \mathbf{E} - \mathbf{T}\boldsymbol{\mu}_0 \\ \mathbf{p}_0 = \mathbf{r}_0 \end{cases} \tag{16}$$

the algorithm reads as follows:

$$\begin{cases} \gamma_i = \dfrac{\mathbf{r}_i^{\mathbf{T}}\mathbf{r}_i}{\mathbf{p}_i^{T}\mathbf{T}\mathbf{p}_i} \\ \boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i + \gamma_i\mathbf{p}_i \\ \mathbf{r}_{i+1} = \mathbf{r}_i - \gamma_i\mathbf{T}\mathbf{p}_i \\ \beta_{i+1} = \dfrac{\mathbf{r}_{i+1}^{T}\mathbf{r}_{i+1}}{\mathbf{r}_i^{T}\mathbf{r}_i} \\ \mathbf{p}_{i+1} = \mathbf{r}_{i+1} + \beta_{i+1}\mathbf{p}_i \end{cases} \tag{17}$$

where $\mathbf{p}_i$ is the descent direction at iteration i, $\boldsymbol{\mu}_i$ the associated dipole vector, and $\mathbf{r}_i$ the associated residual. The magic of the conjugate gradient algorithm is that this simple recursion scheme still guarantees $\boldsymbol{\mu}_i$ to be the optimum over the entire Krylov-subspace of order i.

There are several techniques to accelerate the convergence of this algorithm. A widely used strategy is to use preconditioners. Indeed, one can show that the convergence rate of the conjugate gradient, and more generally of Krylov subspace methods, depends on the condition number of the matrix that is being inverted: the lower this number is (it is always greater than 1), the faster the conjugate gradient will converge. In the case of symmetric positive definite (s.p.d.) matrices as the polarization matrix, this number can be expressed such as

$$\kappa(\mathbf{T}) = \frac{\lambda_{\max}}{\lambda_{\min}} \tag{18}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest eigenvalues of the polarization matrix. A preconditioner is then a matrix P that is "close" to the inverse of $\mathbf{T}$, such that the matrix P is easily applied to a vector, $\kappa(P\mathbf{T}) \leq \kappa(\mathbf{T})$, and $\kappa(P\mathbf{T})$ is close to 1. The conjugate gradient algorithm is then applied to the matrix $P\mathbf{T}$ with $P\mathbf{E}$ as a right-hand side. We first chose to use one of the simplest forms of preconditioner: the diagonal or Jacobi preconditioner, in which P is the inverse of the (block-)diagonal part of the polarization matrix. The advantage of this choice in our context is that multiplying a matrix by a diagonal matrix is computationaly almost free and that the diagonal of $\mathbf{T}$ does not depend on the positions of the atoms of the system that is studied. As a consequence, this choice does not complicate the expression of the gradients very much. On the down side, the diagonal of $\mathbf{T}$ is not a perfect approximation of it, so that we do not expect the acceleration of convergence to be the highest among the possible choices of preconditioners. This is why we also considered a more efficient preconditioner designed for the polarization problem which we will present below. Wang and Skeel[12] start from the expression

$$\mathbf{T}^{-1} = \boldsymbol{\alpha}(I_d - \boldsymbol{\alpha}\mathcal{T})^{-1} \tag{19}$$

giving the first approximation

$$\mathbf{T}^{-1} \approx \boldsymbol{\alpha}(I_d + \boldsymbol{\alpha}\mathcal{T}) \tag{20}$$

DOI: 10.1021/acs.jctc.6b00981
J. Chem. Theory Comput. 2017, 13, 180−190

which is in fact equivalent to one Jacobi iteration. A second approximation is then made by only considering the interactions within a certain cutoff in the matrix $\mathcal{T}$. A typical value for this cutoff is 3 Å, a value that we also used for our numerical tests presented below. This preconditioner has a bigger impact on reducing the condition number of the polarization matrix than the Jacobi one but it also has a higher computational cost per iteration. This cost is typically a bit less than half a real space matrix-vector product within a Particle Mesh Ewald simulation with usual settings for the value we chose (7 Å cutoff). The parallel implementation of this preconditioner would require additional communications before and after the application of this preconditioner.[6] Finally, as it depends on the atoms positions, the expression of the gradients of the associated dipoles would be very involved (therefore in the following sections we will only retain the diagonal preconditionner). To illustrate the different rates of convergence of these iterative methods we plotted in Figure 2 their convergence as well as the

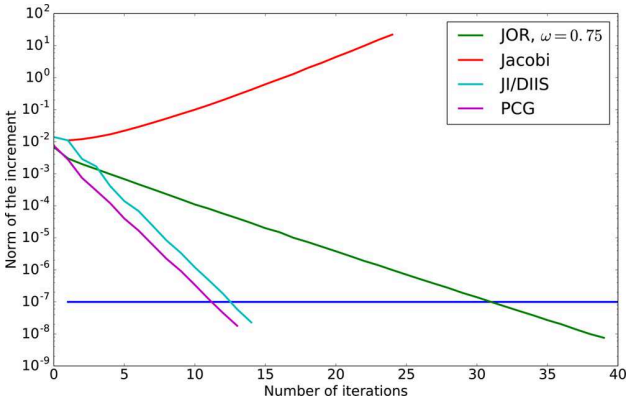


Figure 2. Norm of the increment as a function of the number of iterations for different iterative methods (AMOEBA), computed on ubiquitin.

one of JI/DIIS wich is described in ref. 7 (represented by the norm of the increment) as a function of the number of iterations in the context of the AMOEBA polarizable force field for the ubiquitin protein in water. Note that the Jacobi iterations are not convergent in this case and that both the JI/DIIS and the Preconditioned Conjugate Gradient converge twice as fast as the JOR (as supported by the theory, as the convergence rate of JOR depends on the condition number, while the rate of Krylov methods depends on its square root).

We will now explain how to truncate the preconditioned conjugate gradient to get analytical expressions that approximate the polarization energy.

6. TRUNCATED PRECONDITIONED CONJUGATE GRADIENT

Let us define $\boldsymbol{\mu}_{\mathrm{TCG}n}$, the approximation of the induced dipoles obtained by truncating the conjugate gradient at order n. We immediately have the result that $E_{\mathrm{pol}}(\boldsymbol{\mu}) \leq E_{\mathrm{pol}}(\boldsymbol{\mu}_{\mathrm{TCG}n}) \leq E_{\mathrm{pol}}(\boldsymbol{\mu}_{\mathrm{TCG}m})$ if $n \geq m$, with E_{pol} written as in eq 1, and $\boldsymbol{\mu}$ being the exact solution of the linear system. In other words, the quality of the approximation is systematically improvable. One can then unfold the algorithm at order one and two. Using the following notations:

$$\begin{aligned} n_0 &= \mathbf{r}_0^T\mathbf{r}_0 \\ \mathbf{P}_1 &= \mathbf{T}\mathbf{r}_0 \\ t_1 &= \mathbf{r}_0^T\mathbf{P}_1 \\ t_2 &= \frac{n_0\|\mathbf{P}_1\|^2}{t_1^2} \\ \mathbf{P}_2 &= \mathbf{T}\mathbf{p}_1 \\ t_3 &= t_1\mathbf{P}_1^T\mathbf{P}_2 \\ t_4 &= \frac{n_0}{t_1} \\ \gamma_1 &= \frac{t_1^2 - n_0\|\mathbf{P}_1\|^2}{t_3} \end{aligned} \tag{21}$$

one obtains the cumbersome but analytical approximations for the dipoles corresponding to the conjugate gradient truncated at order one and two, thus allowing the derivation of analytical forces that are the exact negative of the gradients of the energy:

$$\boldsymbol{\mu}_{\mathrm{TCG1}} = \boldsymbol{\mu}_0 + t_4\mathbf{r}_0 \tag{22}$$

$$\boldsymbol{\mu}_{\mathrm{TCG2}} = \boldsymbol{\mu}_0 + (t_4 + \gamma_1 t_2)\mathbf{r}_0 - \gamma_1 t_4\mathbf{P}_1 \tag{23}$$

As in the ExPT approach, one can take the following expression as approximation of the polarization energy:

$$E_{\mathrm{pol,TCG}n} = -\frac{1}{2}\boldsymbol{\mu}_{\mathrm{TCG}n}^T\mathbf{E} \tag{24}$$

Note that both these expressions would be exact if the dipole vectors were exact and that the closer these vectors are to the fully converged dipoles, the closer these energies will be to the actual polarization energy.

Indeed, we have:

$$|E_{\mathrm{pol}}(\boldsymbol{\mu}) - E_{\mathrm{T(P)CG}n}(\boldsymbol{\mu})| = \left|\frac{1}{2}\boldsymbol{\mu}^T\mathbf{T}\boldsymbol{\mu} - \boldsymbol{\mu}^T E + \frac{1}{2}\boldsymbol{\mu}^T E\right| \tag{25}$$

$$|E_{\mathrm{pol}}(\boldsymbol{\mu}) - E_{\mathrm{T(P)CG}n}(\boldsymbol{\mu})| = \frac{1}{2}|\boldsymbol{\mu}^T(\mathbf{T}\boldsymbol{\mu} - E)| \tag{26}$$

leading to the following inequatlity:

$$|E_{\mathrm{pol}}(\boldsymbol{\mu}) - E_{\mathrm{T(P)CG}n}(\boldsymbol{\mu})| \leq \frac{1}{2}\|\boldsymbol{\mu}\|_{l^2}\cdot\|\mathbf{r}_n\|_{l^2} \tag{27}$$

These energies are not the expression minimized over the Krylov subspaces at each iteration of the conjugate gradient (CG) algorithm (see eq 1), but they coincide at convergence which should almost be the case if our method is accurate.

These results are naturally extended to the preconditioned conjugate gradient (PCG). One can of course also choose to truncate the (P)CG at a superior order and still be analytical to obtain a more accurate approximation, at the price however of additional computational time, and the analytical expression of the energy and its derivatives will be incrementely more complex, thus cumbersome to implement. In the following section, where numerical results are presented, we will limit ourselves to TCG3 as the highest order of truncation.

Moreover, having chosen an order of truncation of the (P)CG, one can exploit the residual (if it is computed to monitor the accuracy) of the last iteration of the truncated algorithm in order to get closer to the converged value by computing one step of a

DOI: 10.1021/acs.jctc.6b00981
J. Chem. Theory Comput. 2017, 13, 180−190

fixed point iterative method. As did Wang and Skeel,[12] we will call this operation a peek step. Indeed, many fixed point iterative methods such as the Jacobi and more generally the JOR only require to know a starting value of the dipoles and the associated residual to be applied at each iteration. Note that the Jacobi method can be seen as a particular case of the JOR method with $\omega = 1$ and that this operation is not computationally expensive, as it only requires a matrix-vector product with a diagonal matrix if the residual is known. As for any fixed-point method of a linear system, the asymptotic convergence of the JOR method depends on the spectral radius of the iteration matrix. More precisely, the condition

$$\rho(I_d - \omega\boldsymbol{\alpha}\mathbf{T}) < 1 \tag{28}$$

guarantees that the JOR method is convergent. Asymptotically, the best convergence rate is obtained with the value of ω that minimizes this spectral radius. One can show that if $\mathbf{T}$ is symmetric positive definite, this value is

$$\omega_{\text{opt}} = \frac{2}{\lambda_{\min} + \lambda_{\max}} \tag{29}$$

$\lambda_{\min}$ and $\lambda_{\max}$ being, respectively, the smallest and largest eigenvalue of $\boldsymbol{\alpha}\mathbf{T}$.

As these results are asymptotic, one cannot necessarily expect the associated methods to give the best results if only the so-called peek step is applied, as this depends on the composition of the current approximation (which is in our case provided by the T(P)CG) in the eigenvector-basis of $\mathbf{T}$.

Since we cannot rely on asymptotic results for one iteration, we also explored another strategy that can be of use in cases in which one is particularly interested in the values of the energies, as for example in Monte Carlo simulations. The ω_{opt} based on the spectrum intends to further optimize all the modes of the polarization matrix after the (P)CG steps (independently of the actual approximation) and should therefore asymptotically improve both the energy and the RMS. However, other values of ω that take into account the actual approximation can be used to further improve the accuracy. This explains why we tried, starting from one or two iterations of (P)CG, to choose the value of ω that gave the closest approximate polarization energy to the fully converged one. Since the optimal parameter (in this new sense) requires another matrix-vector multiplication, we tried to obtain values of this parameter on the fly by fitting one or several energies against the energies obtained with the fully converged dipoles or a superior truncation of (P)CG. It will be called ω_{fit}.

Starting for example from $\boldsymbol{\mu}_{\text{TCG2}}$, and noting:

$$\boldsymbol{\mu}_{\text{TCG2,peek}} = \boldsymbol{\mu}_{\text{TCG2}} + \omega\boldsymbol{\alpha}\mathbf{r}_2 \tag{30}$$

one can see this procedure as a line search: given the starting point $\boldsymbol{\mu}_2$, one further tries to optimize the energies along the parametrized line $\boldsymbol{\mu}_2 + \omega\boldsymbol{\alpha}\mathbf{r}_2$ for $\omega \in \mathbb{R}$.

Once one of these methods is chosen, analytical expressions of the associated forces can be naturally obtained, thus ensuring that the forces are (up to round off errors) the opposite of the exact gradients of the polarization energy, and thus avoiding an energy drift. Gradients of the energies have been derived and are presented in a technical appendix at the end of the manuscript.

7. NUMERICAL RESULTS

7.1. Energy Conservation of the T(P)CG*n* Approaches.

We first emphasize that Figure 1 already displays an important result: the TCG*n* methods ensure total energy conservation as they use analytical evaluation of the forces. All further refinements discussed in section 6 lead to the same behavior, incrementally closer to the reference energy.

7.2. Stability of the Spectrum.

Before presenting the complete numerical tests, we analyze here the spectrum of the polarization matrix during a MD simulation. Indeed, as pointed out in the theory section, some refinements of the TCG rely on the extreme eigenvalues of $\mathbf{T}$ and $\boldsymbol{\alpha}\mathbf{T}$. We followed the evolution of these eigenvalues during 100 ps of MD. Those tests were made with the Lanczos algorithm since all the matrices we are interested in are symmetric. Indeed, one great advantage of the Lanczos method is that it reduces the computational cost compared to direct methods (such as the one available in the LAPACK library). That way, if direct eigenvalue solvers force the user to compute the full spectrum (i.e all the eigenvalues), Lanczos method allows rapid access to the extreme eigenvalues by constructing a much smaller tridiagonal matrix whose spectrum is close to the one of the original matrix, leading to almost identical extreme eigenvalues that can then be obtained in a few iterations. We observed that in all cases these values are stable over the 100 ps of the MD trajectories as pointed out by Skeel.[12] This can be seen for S3 and the ubiquitin system in Figures 3 and 4. This result motivated our choice to compute ω_{opt}

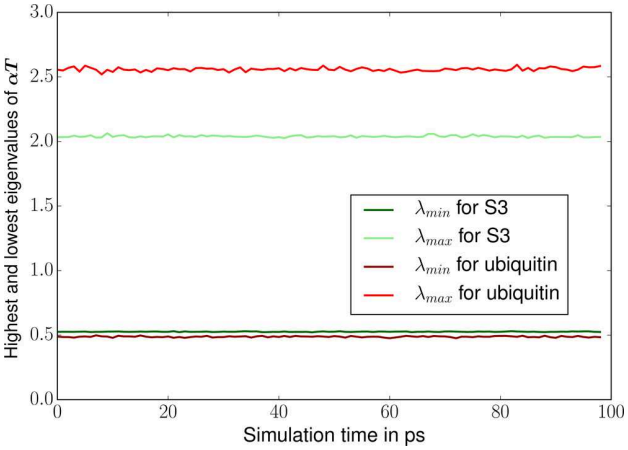


Figure 3. Evolution of the extreme eigenvalues of $\boldsymbol{\alpha}\mathbf{T}$ for S3 and ubiquitin.

and ω_{fit} for the first geometry of our equilibrated systems and to keep this value for all the others geometries. Both our Lanczos

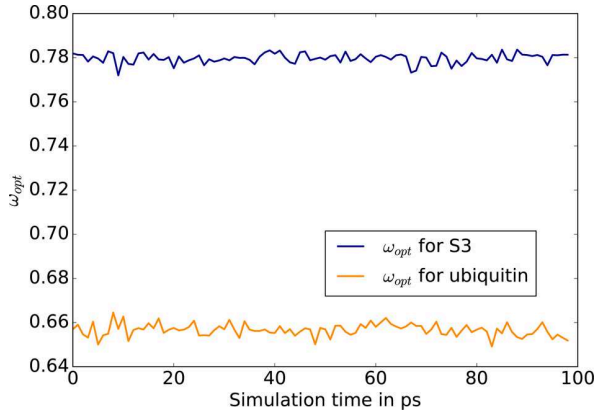


Figure 4. Evolution of ω_{opt} for S3 and ubiquitin.

DOI: 10.1021/acs.jctc.6b00981
J. Chem. Theory Comput. 2017, 13, 180−190

implementation and the energy fitting procedure are fast enough to be used on the fly while being negligeable over a 100 ps MD simulation. In our tests, the adaptive reevaluation of the ω's was nevertheless never required.

7.3. Computational Details and Notations. In this section, we will present some numerical results from the methods presented above. All the tests presented here were made using the AMOEBA polarizable force field[24,25] implemented in our Tinker-HP code.[26] The proposed benchmarks deal with homogeneous and inhomogeneous systems: water clusters and protein in water droplets as well as an ionic liquid system. The three water systems will be called S1, S2, and S3 and contain respectively 27 molecules (81 atoms), 216 molecules (648 atoms), and 4000 molecules (12000 atoms). We chose difficult systems ranging from usual proteins to metalloproteins and highly charged ionic liquids.[27] The protein droplets are, respectively, a metalloprotein containing two Zn(II) cations (nucleocapsid protein ncp7) with water (18515 atoms including counterions), the ubiquitin protein with water (9737 atoms), and the dhfr protein with water (23558 atoms). The ionic liquid system is made of [dmim+][Cl−] (1−3 dimethylimidazolium) ions, making it a highly charged system of 3672 atoms with a very different regime of polarization interactions. All the results presented in this section were averaged over 100 geometries that were extracted from a 100 ps MD NVT trajectory (one geometry was saved every picosecond) at 300 K for all systems, except the [dmim+][CL−] at 425 K. The results, that will give indications about the accuracy of the approximate methods compared to the fully converged iterative results, will give two different and complementary aspects of this accuracy. We will first compare the polarization energies (in kcal/mol) obtained with dipoles converged with a very tight criterion (RMS of the residual under 10^{-9}) to the ones obtained with T(P)CG. We will then present the RMS of the difference between the fully converged dipole vectors and the approximate methods. This RMS is a good indicator of the quality of T(P)CG forces compared to the reference: the smaller this RMS is, the closer the approximated but analytical forces will be to the reference force.

Tables 1 to 4 describe the water systems and Tables 5 to 8 describe the protein droplets and ionic liquid. We will denote by "ref" the results obtained with dipoles converged up to 10^{-9} in the RMS of the residual; by "ExPT" the results obtained with the method of Simmonnet et al. presented in section 3; by "TCGn" (with n = 1, 2, and 3) the results obtained with the CG truncated at order 1, 2, and 3; by "TPCGn" (P = diag) (with n = 1, 2, and 3) the results obtained with the preconditioned (with the diagonal) CG truncated at order 1,2 and 3; by "TPCGn" (P = Skeel) (with n = 1, 2, and 3) the results obtained with the preconditioned (using Wang and Skeel's preconditioner) CG truncated at order 1, 2, and 3.

Furthermore, we will present results obtained with different kinds of peek steps. We will first denote by TCGn(ω = 1) (with n = 1, 2, and 3) the results obtained with the CG truncated at different orders when a Jacobi peek step is made after the last conjugate gradient iteration. We will also denote by TPCGn (P = diag) the results where the same peek step is made after different numbers of iterations of the PCG with a diagonal preconditioner.

We will also denote by TPCGn(P = diag)(ω_{opt}) (with n = 1 and 2) the results obtained with 1 and 2 iterations of diagonally preconditioned CG and a JOR peek step with an "optimal" ω_{opt} in the sense of section 6 (that depends whether a preconditioner is used or not).

As explained in the previous section we also explored a strategy where the damping parameter of the JOR step is fitted to reproduce energy values. In the following tables, the damping parameter will be denoted by ω_{fit}.

7.4. Numerical Results. A first observation to make is that given a particular matrix (preconditioned or not) and with or whithout a JOR peek step, the results are always better in terms of energy and RMS when one performs more matrix-vector products, that is, going to a higher order of truncation. This is naturally explained in the context of the Krylov methods: an additional matrix-vector product increases the dimension of the Krylov subspace on which the polarization functional (see eq 1) is minimized, and thus systematically improves the associated results. We should also recall here that the functional that is minimized over growing subspaces is not exactly the same as the one we are taking as the polarization energy and that this explains the nonvariationality of some of our results: there are many cases where the energy associated TCG3 is slightly lower than the one associated with the fully converged dipoles (see discussion in section 6).

Table 1. Polarization Energies of Water Systems

water box	S1	S2	S3
ref	−81.03	−803.33	−15229.87
ExPT	−69.54	−660.95	−12822.79
TCG1	−73.50	−728.73	−13814.35
TCG2	−80.69	−800.32	−15173.15
TCG3	−81.24	−805.20	−15265.65
TPCG1 (P = diag)	−74.98	−741.91	−14028.18
TPCG2 (P = diag)	−80.81	−801.61	−15194.87
TPCG3 (P = diag)	−81.20	−805.26	−15268.43
TPCG1 (P = Skeel)	−78.63	−779.17	−14743.48
TPCG2 (P = Skeel)	−81.03	−803.11	−15222.53
TPCG3 (P = Skeel)	−81.06	−803.64	−15236.03

Table 2. Polarization Energies of Water Systems, Using a Peek-Step

water box	S1	S2	S3
ref	−81.03	−803.33	−15229.87
TCG1(ω = 1)	−81.41	−806.83	−15315.13
TCG2(ω = 1)	−80.23	−794.49	−15061.22
TCG3(ω = 1)	−80.78	−800.83	−15181.55
TPCG1 (P = diag)(ω = 1)	−79.88	−791.51	−15001.40
TPCG2 (P = diag)(ω = 1)	−80.98	−802.74	−15218.27
TPCG3 (P = diag)(ω = 1)	−81.03	−803.27	−15228.74
TPCG1 (P = diag)(ω_{opt})	−78.98	−780.94	−14789.04
TPCG2 (P = diag)(ω_{opt})	−80.95	−802.50	−15213.17
TPCG1 (P = diag)(ω_{fit})	−81.06	−803.42	−15230.10
TPCG2 (P = diag)(ω_{fit})	−81.02	−803.06	−15231.14

We can also see on the numerical tests that using a preconditioner systematically reduces the associated RMS. Concerning the energy, the improvement is less systematic and depends on the type of preconditioner: the diagonal is less accurate than the one described by Wang et al.,[12] a result that was anticipated.

Nevertheless, preconditioning is important when coupled with a peek step: a combination of any preconditioner with the peek is better than the peek alone. However, concerning the peek itself, one observes a systematic improvement of both RMS and energy

DOI: 10.1021/acs.jctc.6b00981
J. Chem. Theory Comput. 2017, 13, 180−190

Table 3. RMS of the dipole vector compared to the reference for water systems

water box	S1	S2	S3
ExPT	1.4×10^{-2}	2.5×10^{-2}	2.6×10^{-2}
TCG1	6.3×10^{-3}	7.0×10^{-3}	7.1×10^{-3}
TCG2	1.7×10^{-3}	1.9×10^{-3}	1.9×10^{-3}
TCG3	4.7×10^{-4}	5.4×10^{-4}	5.5×10^{-4}
TPCG1 (P = diag)	4.9×10^{-3}	5.6×10^{-3}	5.8×10^{-3}
TPCG2 (P = diag)	9.2×10^{-4}	1.1×10^{-3}	1.1×10^{-3}
TPCG3 (P = diag)	3.8×10^{-4}	3.8×10^{-4}	3.9×10^{-4}
TPCG1(P = Skeel)	2.2×10^{-3}	2.6×10^{-3}	2.7×10^{-3}
TPCG2 (P = Skeel)	3.0×10^{-4}	3.9×10^{-4}	4.2×10^{-4}
TPCG3 (P = Skeel)	6.6×10^{-5}	9.5×10^{-5}	1.0×10^{-4}

Table 4. RMS of the Dipole Vector Compared to the Reference for Water Systems, Using a Peek-Step

water box	S1	S2	S3
TCG1($\omega = 1$)	3.6×10^{-3}	3.9×10^{-3}	3.7×10^{-3}
TCG2($\omega = 1$)	1.5×10^{-3}	1.7×10^{-3}	1.8×10^{-3}
TCG3($\omega = 1$)	4.6×10^{-4}	4.9×10^{-4}	4.8×10^{-4}
TPCG1(P = diag)($\omega = 1$)	2.2×10^{-3}	2.6×10^{-3}	2.7×10^{-3}
TPCG2 (P = diag)($\omega = 1$)	4.1×10^{-4}	5.0×10^{-4}	5.2×10^{-4}
TPCG3 (P = diag)($\omega = 1$)	1.3×10^{-4}	1.5×10^{-4}	1.6×10^{-4}
TPCG1 (P = diag)(ω_{opt})	2.3×10^{-3}	2.7×10^{-3}	2.8×10^{-3}
TPCG2 (P = diag)(ω_{opt})	3.9×10^{-4}	4.6×10^{-4}	4.7×10^{-4}
TPCG1 (P = diag)(ω_{fit})	2.6×10^{-3}	3.0×10^{-3}	3.0×10^{-3}
TPCG2 (P = diag)(ω_{fit})	5.3×10^{-4}	7.0×10^{-4}	1.0×10^{-3}

Table 5. Polarization Energies of Protein Droplet and Ionic Liquids

system	ncp7	ubiquitin	dhfr	[dmim+][Cl−]
ref	−24202.54	−11154.87	−28759.01	−1476.79
ExPT	−27362.70	−10919.77	−28076.62	−5841.73
TCG1	−21733.63	−9897.22	−25583.50	−1428.35
TCG2	−23922.79	−11031.67	−28463.51	−1420.00
TCG3	−24262.87	−11174.93	−28812.99	−1450.22
TPCG1 (P = diag)	−21438.14	−9907.09	−25588.07	−1465.66
TPCG2 (P = diag)	−23613.31	−10948.32	−28206.73	−1462.22
TPCG3 (P = diag)	−24219.49	−11164.62	−28775.53	−1469.89
TPCG1 (P = Skeel)	−22489.55	−10458.44	−27030.86	−1424.49
TPCG2 (P = Skeel)	−24056.53	−11090.36	−28637.35	−1469.05
TPCG3 (P = Skeel)	−24208.22	−11144.53	−28763.55	−1477.02

with and without preconditioning. In particular this is the case when $\omega = 1$ (Jacobi peek step).

As the spectrum is stable (see section 7.2), one can use an adaptive ω_{opt} coefficient computed on one geometry using a few iterations of the Lanczos method. In that case, the energies are slighlty less accurate than the ones obtained with $\omega = 1$. Concerning the RMS, we observe a systematic reduction by a factor 2 for TPCG2 and TPCG3 but not for TPCG1. This occurs because, if the asymptotic coefficient ω_{opt} is the same, the starting point of the peek step is different and is significantly better for TPCG2 and TPCG3 as additional matrix-vector products have been computed.

The results obtained with ω_{fit} after 1, 2, or 3 iterations of PCG show that it is possible to stay close to the converged value of the polarization energy with only one or two matrix-vector products and a ω parameter that is only fitted once during a 100 ps dynamic. But we can also see that this is made at the cost of slightly degrading the RMS compared to the results obtained with ω_{opt} or with $\omega = 1$. Overall, these RMS are of the same order of magnitude than the ones obtained with ω_{opt} and $\omega = 1$. This balance between RMS and energy depending on the choice of ω as the relaxation parameter for a JOR peek step can be seen as the choice to favor the minimization of the error along some modes of the polarization matrix: the energy is more sensitive to modes corresponding to large eigenvalues whereas the RMS is sensitive to all of them. Overall, a $\omega = 1$ Jacobi peek step tends to improve both RMS and the energy whereas ω_{opt} favors RMS and ω_{fit} favors energies. As we showed, TPCG1 should not be used with a ω_{opt} peek step but with one corresponding to $\omega = 1$ and ω_{fit}, but all options are open for TPCG2 and TPCG3.

A choice can then be made depending on the simulation that one wants to run. For a Monte Carlo simulation it is essential to have accurate energies: the strategy of using an adaptative parameter (refittable at a negligeable cost) that allows the correct reproduction of the energies with only one or two iterations of the (P)CG would hence produce excellent results. On the other side, during a MD simulation, one wants to get the dynamics right; in this case, choosing the method that minimizes the RMS and thus the error made on the forces may produce improved results. For example, using TPCG2(P = diag)(ω_{opt}) is a good strategy to fulfill this purpose. However, the procedure leading to ω_{fit} only slightly degrades the RMS and provides RMS far beyond the usual values for which the force field models are parametrized. One has also to keep in mind that other source of errors exist in MD, such as the ones due to the PME discretization or van der Waals cutoffs, that are larger than the error discussed in this section. Nevertheless, none of the refinements will compete with a full additional matrix-vector product because an additional CG step is optimal. We see clearly that TPCG3(ω_{fit}) reaches high accuracy on both RMS and energies.

Concerning preconditioning, we confirm the very good behavior of the Skeel preconditioner. However, its cost is non-negligible in terms of computations, in terms of necessary communications arising when running in parallel, and in terms of complexity of implementation. We recommend therefore the use of the simpler yet efficient diagonal preconditioner. Overall, possibilities of tayloring TCG approaches are infinite. In practice, one could design more adapted preconditioners combining accuracy and low computational cost.

To conclude, a striking result is obtained for well conditioned systems such as water: computations show that they will require a smaller order of truncation than the proteins to obtain the same level of accuracy.

8. CONCLUSION

We proposed a general way to derive an analytical expression of the many-body polarization energy that approximates the inverse of $\mathbf{T}$ using a truncated preconditioned conjugate gradient approach. The general method gives analytical forces, guaranteeing that they are the opposite of the exact gradients of the energies, parameter free, and can replace the usual many-body polarization solvers in popular codes with little effort. The proposed technique allows by construction a true energy conservation as it is based on analytical derivatives. The method

DOI: 10.1021/acs.jctc.6b00981
J. Chem. Theory Comput. 2017, 13, 180−190

Table 6. Polarization Energies of Protein Droplet and Ionic Liquids, Using a Peek-Step

system	ncp7	ubiquitin	dhfr	[dmim+][Cl-]
ref	−24202.54	−11154.87	−28759.01	−1476.79
TCG1(ω = 1)	−24481.14	−11231.35	−28986.08	−1477.08
TCG2(ω = 1)	−23965.96	−11009.06	−28384.49	−1465.73
TCG3(ω = 1)	−24121.02	−11105.78	−28635.73	−1441.95
TPCG1 (P = diag)(ω = 1)	−23532.73	−10829.84	−27972.41	−1493.58
TPCG2 (P = diag)(ω = 1)	−24123.65	−11128.14	−28683.52	−1471.34
TPCG3 (P = diag)(ω = 1)	−24194.37	−11150.95	−28749.68	−1478.83
TPCG1 (P = diag)(ω_{opt})	−22773.65	−10513.24	−27079.47	−1484.24
TPCG2 (P = diag)(ω_{opt})	−23938.70	−11066.44	−28504.96	−1468.29
TPCG1 (P = diag)(ω_{fit})	−24161.11	−11162.02	−28766.40	−1479.06
TPCG2 (P = diag)(ω_{fit})	−24205.30	−11154.21	−28753.60	−1475.08

Table 7. RMS of the Dipole Vector Compared to the Reference for Protein Droplets and Ionic Liquids

water box	ncp7	ubiquitin	dhfr	[dmim+][Cl-]
ExPT	8.1×10^{-2}	5.2×10^{-2}	5.4×10^{-2}	1.3×10^{-1}
TCG1	8.9×10^{-3}	8.8×10^{-3}	8.8×10^{-3}	1.1×10^{-2}
TCG2	3.5×10^{-3}	3.2×10^{-3}	3.2×10^{-3}	7.2×10^{-3}
TCG3	2.1×10^{-3}	1.7×10^{-3}	1.7×10^{-3}	5.3×10^{-3}
TPCG1 (P = diag)	8.6×10^{-3}	8.0×10^{-3}	8.1×10^{-3}	6.9×10^{-3}
TPCG2 (P = diag)	2.5×10^{-3}	2.0×10^{-3}	2.2×10^{-3}	3.4×10^{-3}
TPCG3 (P = diag)	7.1×10^{-4}	6.5×10^{-4}	7.2×10^{-4}	7.9×10^{-4}
TPCG1 (P = Skeel)	5.5×10^{-3}	4.4×10^{-3}	4.5×10^{-3}	5.6×10^{-3}
TPCG2 (P = Skeel)	9.0×10^{-4}	7.7×10^{-4}	7.8×10^{-4}	1.5×10^{-3}
TPCG3 (P = Skeel)	2.1×10^{-4}	1.8×10^{-4}	1.9×10^{-4}	3.2×10^{-4}

minimizes the energy over the (preconditioned) Krylov space which leads to superior accuracy than fixed point inspired methods such as ExPT and associated methods. It does not use any history of the previous steps and is therefore fully time reversible and is compatible with multitimestep integrators.[28] The best compromise between accuracy and speed appears to be the TPCG-2 approach that consists of two iterations of PCG with a computational cost of three matrix vector multiplications for the energy (one for the descent direction plus two for the iterations). The analytical derivatives have a cost equivalent to an additional matrix vector product. The overall computational cost is therefore identical to that of the ExPT. We showed that the method allows the computation of potential energy surfaces very close to the exact ones and that it is systematically improvable using a final peek step. Strategies for adaptative JOR coefficients have been discussed and allows an improvement of the desired quantities at a negligeable cost. Overall, among all the derived methods, TPCG3(ω_{fit}) provides high accuracy in both energy and RMS. Concerning the future improvements of the accuracy of the method, one could find dedicated preconditionners improving the efficiency of the CG steps. Nevertheless, the final choice of ingredients will be a trade-off between accuracy, computational cost, and communication cost when running in parallel. We will address this issue in the context of the Tinker-HP package. The TPCG*n* approaches will be coupled to a domain decomposition infrastructure with linear scaling capabilities, thanks to a SPME[8] implementation, which is straightforward in link with our previous work on PCG. Future work will then include validation of the methods by comparing condensed-phase properties obtained using different orders of TCG. Given the level of accuracy already obtained on induced dipoles and energies, we expect the majority of these properties to be conserved by using T(P)CG2 and higher-order methods.

TECHNICAL APPENDIX

A.1. Analytical Gradients and Polarization Energies for TCG

In this section, we will present the analytical derivatives of the polarization energies associated with the polarization energies $E_{pol,TCG1}$ and $E_{pol,TCG2}$ with respect to the positions of the atoms of the system. The extension to $E_{pol,(P=diag)TCG1}$ and $E_{pol,(P=diag)TCG2}$ is straightforward, as is the expressions including a final JOR peek step. We don't report here the expression of the analytical gradients of $E_{pol,TCG3}$ as it follows the same logic but is just incremently complex.

These gradients have been validated against the ones obtained with finite differences of the energies and an implementation of these equations will be accessible through the Tinker-HP

Table 8. RMS of the Dipole Vector Compared to the Reference for Protein Droplets and Ionic Liquids, Using a Peek-Step

water box	ncp7	ubiquitin	dhfr	[dmim+][Cl-]
TCG1(ω = 1)	4.6×10^{-3}	4.4×10^{-3}	4.5×10^{-3}	7.0×10^{-3}
TCG2(ω = 1)	2.9×10^{-3}	2.5×10^{-3}	2.5×10^{-3}	5.5×10^{-3}
TCG3(ω = 1)	1.6×10^{-3}	1.1×10^{-3}	1.1×10^{-3}	4.1×10^{-3}
TPCG1 (P = diag)(ω = 1)	4.4×10^{-3}	3.9×10^{-3}	4.1×10^{-3}	3.2×10^{-3}
TPCG2 (P = diag)(ω = 1)	1.7×10^{-3}	1.4×10^{-3}	1.7×10^{-3}	1.6×10^{-3}
TPCG3 (P = diag)(ω = 1)	4.3×10^{-4}	3.8×10^{-4}	4.8×10^{-4}	4.5×10^{-4}
TPCG1 (P = diag)(ω_{opt})	5.1×10^{-3}	4.7×10^{-3}	4.8×10^{-3}	3.8×10^{-3}
TPCG2 (P = diag)(ω_{opt})	1.3×10^{-3}	1.0×10^{-3}	1.1×10^{-3}	1.9×10^{-3}
TPCG1 (Jacobi)(ω_{fit})	4.9×10^{-3}	4.5×10^{-3}	4.6×10^{-3}	4.5×10^{-3}
TPCG2 (Jacobi)(ω_{fit})	2.2×10^{-3}	1.7×10^{-3}	2.1×10^{-3}	2.0×10^{-3}

DOI: 10.1021/acs.jctc.6b00981
J. Chem. Theory Comput. 2017, 13, 180−190

software public distribution. Since we are in the context of the AMOEBA force field, we will consider that each atom site embodies a permanent multipole expansion up to quadrupoles. For site i, the components of this expansion will be denoted by $q_i, \vec{\mu}_{p,i}, \theta_i$.

Furthermore, since the permanent dipoles and quadrupoles are expressed in a local frame that depends on the positions of neighboring atoms, they are rotated in the lab frame with rotation matrices depending on these positions, so that we now have to deal with partial derivatives of the dipole and quadrupole components: the "torques". Therefore, the derivative of the polarization energy ϵ, written as $\frac{1}{2}\boldsymbol{\mu}^T\mathbf{E}$ for $\boldsymbol{\mu} = \boldsymbol{\mu}_{\mathrm{TCG1}}$ or $\boldsymbol{\mu}_{\mathrm{TCG2}}$, with respect to the β-component of the kth site is given by

$$\frac{\mathrm{d}\epsilon}{\mathrm{d}r_k^\beta} = \frac{\partial\epsilon}{\partial r_k^\beta} + \sum_{i=1,N}\sum_{\alpha=1,3}\sum_{\gamma=1,3}\frac{\partial\epsilon}{\partial\theta_{p,i}^{\alpha,\gamma}}\frac{\partial\theta_{p,i}^{\alpha,\gamma}}{\partial r_k^\beta} + \sum_{i=1,N}\sum_{\alpha=1,3}\frac{\partial\epsilon}{\partial\mu_{p,i}^\alpha}\frac{\partial\mu_{p,i}^\alpha}{\partial r_{k^\beta}} \tag{31}$$

Formally, these derivatives can be written as

$$\epsilon' = -\frac{1}{2}(\boldsymbol{\mu}'^T\mathbf{E} + \boldsymbol{\mu}^T\mathbf{E}') \tag{32}$$

Hence different types of derivatives are involved:

- derivatives of the rotated permanent multipoles
- derivatives of the permanent electric field with respect to the spatial components of the different atoms
- derivatives of the permanent electric field with respect to the permanent multipoles
- derivatives of the induced dipole vector ($\boldsymbol{\mu}$) with respect to spatial components
- derivatives of the induced dipole vector with respect to the permanent multipole components

As these quantities are standard except for the ones concerning the approximate dipole vector, these are the only one we will express here.

Using the same notation as before we have

$$\begin{aligned}
\mathbf{r}_0 &= \mathbf{E} - \mathbf{T}\boldsymbol{\mu}_0\\
\mathbf{p}_0 &= \mathbf{r}_0\\
n_0 &= \mathbf{r}_0^T\mathbf{r}_0\\
\mathbf{P}_1 &= \mathbf{T}\mathbf{r}_0\\
t_1 &= \mathbf{r}_0^T\mathbf{P}_1\\
t_2 &= \frac{n_0\|\mathbf{P}_1\|^2}{t_1^2}\\
\mathbf{P}_2 &= \mathbf{T}\mathbf{p}_1\\
t_3 &= t_1\mathbf{P}_1^T\mathbf{P}_2\\
t_4 &= \frac{n_0}{t_1}\\
\gamma_1 &= \frac{t_1^2 - n_0\|\mathbf{P}_1\|^2}{t_3}\\
t_5 &= \mathbf{P}_1^T\mathbf{P}_2\\
\beta_2 &= \frac{n_0 + t_4^2\|\mathbf{P}_1\|^2 + \gamma_1^2\|\mathbf{P}_2\|^2 - 2t_1t_4 - 2\gamma_1t_4\|\mathbf{P}_1\|^2 + 2\gamma_1t_4t_5}{(t_2-1)n_0}\\
\mathbf{P}_3 &= (1+\beta_2t_2)\mathbf{T}\mathbf{r}_0 - (t_4+\beta_2t_4)\mathbf{T}\mathbf{P}_1 - \gamma_1\mathbf{T}\mathbf{P}_2\\
\gamma_2 &= \frac{n_0 + t_4^2\|\mathbf{P}_1\|^2 + \gamma_1^2\|\mathbf{P}_2\|^2 - 2t_1t_4 - 2\gamma_1t_4\|\mathbf{P}_1\|^2 + 2\gamma_1t_4t_5}{(1+\beta_2t_2)\mathbf{r}_0^T\mathbf{P}_3 - (t_4+\beta_2t_4)\mathbf{P}_1^T\mathbf{P}_3 + \gamma_1\mathbf{P}_2^T\mathbf{P}_3}
\end{aligned} \tag{33}$$

So that

$$\boldsymbol{\mu}_{\mathrm{TCG1}} = \boldsymbol{\mu}_0 + t_4\mathbf{r}_0 \tag{34}$$

$$\boldsymbol{\mu}_{\mathrm{TCG2}} = \boldsymbol{\mu}_0 + (\gamma_1t_2 + t_4)\mathbf{r}_0 - \gamma_1t_4\mathbf{P}_1 \tag{35}$$

$$\boldsymbol{\mu}_{\mathrm{TCG3}} = \boldsymbol{\mu}_0 + (t_4 + \gamma_1t_2 + \gamma_2 + \gamma_2\beta_2t_2)\mathbf{r}_0 - (\gamma_1t_4 + \gamma_2t_4 + \gamma_2\beta_2t_4)\mathbf{P}_1 - \gamma_1\gamma_2\mathbf{P}_2 \tag{36}$$

We then need to differentiate these expressions with respect to space and multipole components, respectively. Using the following formal development for the spatial derivative:

DOI: 10.1021/acs.jctc.6b00981
J. Chem. Theory Comput. 2017, 13, 180−190

$$
\begin{aligned}
\mathbf{r}_0' &= \mathbf{E}' - \mathbf{T}'\boldsymbol{\mu}_0 - \mathbf{T}\boldsymbol{\mu}_0' \\
n_0' &= 2\mathbf{r}_0^T\mathbf{r}_0' \\
\mathbf{P}_1' &= \mathbf{T}'\mathbf{r}_0 + \mathbf{T}\mathbf{r}_0' \\
(\|\mathbf{P}_1\|^2)' &= 2\mathbf{P}_1^T\mathbf{P}_1 \\
t_1' &= \mathbf{r}_0^T\mathbf{P}_1' + \mathbf{P}_1^T\mathbf{r}_0' \\
t_2' &= \frac{(n_0'\|\mathbf{P}_1\|^2 + n_0(\|\mathbf{P}_1\|^2)')t_1^2 - (n_0\|\mathbf{P}_1\|^2)2t_1t_1'}{t_1^4} \\
\mathbf{P}_2' &= \mathbf{T}'\mathbf{P}_1 + \mathbf{T}\mathbf{P}_1' \\
t_3' &= t_1'\mathbf{P}_1^T\mathbf{P}_2 + t_1\mathbf{P}_2^T\mathbf{P}_1' + t_1\mathbf{P}_1^T\mathbf{P}_2' \\
t_4' &= \frac{n_0't_1 - n_0t_1'}{t_1^2} \\
\gamma_1' &= \frac{1}{t_3^2}((2t_1t_1' - n_0'\|\mathbf{P}_1\|^2 - n_0(\|\mathbf{P}_1\|^2)')t_3 \\
&\quad - (t_1^2 - n_0\|\mathbf{P}_1\|^2)t_3')
\end{aligned}
\tag{37}
$$

we obtain

$$
\boldsymbol{\mu}_{\mathrm{TCG1}}' = \boldsymbol{\mu}_0' + t_4\mathbf{r}_0' + t_4'\mathbf{r}_0 \tag{38}
$$

$$
\begin{aligned}
\boldsymbol{\mu}_{\mathrm{TCG2}}' = \boldsymbol{\mu}_0' &+ (t_4 + \gamma_1t_2)\mathbf{r}_0' + (t_4' + \gamma_1't_2 + \gamma_1t_2')\mathbf{r}_0 \\
&+ \gamma_1't_4\mathbf{P}_1 + \gamma_1t_4'\mathbf{P}_1 + \gamma_1t_4\mathbf{P}_1'
\end{aligned}
\tag{39}
$$

AUTHOR INFORMATION

Corresponding Authors

*E-mail: louis.lagardere@upmc.fr.

*E-mail: jpp@lct.jussieu.fr.

ORCID

Jean-Philip Piquemal: 0000-0001-6615-9426

Funding

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. Funding from French CNRS through a PICS grant between UPMC and UT Austin is acknowledged. Jean-Philip Piquemal is grateful for support by the Direction Générale de l'Armement (DGA) Maitrise NRBC of the French Ministry of Defense. J.W.P. and P.R. thank the National Institutes of Health (R01GM106137 and R01GM114237) for support.

Notes

The authors declare no competing financial interest.

REFERENCES

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

(2) Lopes, P. E.; Huang, J.; Shim, J.; Luo, Y.; Li, H.; Roux, B.; MacKerell, A. D., Jr *J. Chem. Theory Comput.* **2013**, *9*, 5430−5449.

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

(4) Mills, M. J.; Popelier, P. L. *Comput. Theor. Chem.* **2011**, *975*, 42−51.

(5) Ren, P.; Ponder, J. W. *J. Phys. Chem. B* **2003**, *107*, 5933−5947.

(6) Lagardère, L.; Lipparini, F.; Polack, É.; Stamm, B.; Cancès, É.; Schnieders, M.; Ren, P.; Maday, Y.; Piquemal, J.-P. *J. Chem. Theory Comput.* **2015**, *11*, 2589−2599.

(7) 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.

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

(9) Kolafa, J. *J. Comput. Chem.* **2004**, *25*, 335−342.

(10) Gear, C. W. *Commun. ACM* **1971**, *14*, 176−179.

(11) Albaugh, A.; Demerdash, O.; Head-Gordon, T. *J. Chem. Phys.* **2015**, *143*, 174104.

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

(13) Wang, W. Fast Polarizable Force Field Computation in Biomolecular Simulations. Ph.D. Thesis, University of Illinois at Urbana-Champaign, 2013.

(14) Simmonett, A. C.; Pickard, F. C., IV; Shao, Y.; Cheatham, T. E., III; Brooks, B. R. *J. Chem. Phys.* **2015**, *143*, 074115.

(15) Simmonett, A. C.; Pickard, F. C., IV; Ponder, J. W.; Brooks, B. R. *J. Chem. Phys.* **2016**, *145*, 164101.

(16) Thole, B. T. *Chem. Phys.* **1981**, *59*, 341−350.

(17) Cheng, H.; Greengard, L.; Rokhlin, V. *J. Comput. Phys.* **1999**, *155*, 468−498.

(18) Picard, E. *J. Math. Pures Appl.* **1890**, *6*, 145−210.

(19) Ryaben'kii, V. S.; Tsynkov, S. V. *A Theoretical Introduction to Numerical Analysis*; CRC Press, 2006.

(20) Quarteroni, A.; Sacco, R.; Saleri, F. In *Numerical Mathematics*; Springer Science & Business Media, 2010; Vol. *37*.

(21) Saad, Y. In *Iterative Methods for Sparse Linear Systems*; Siam, 2003.

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

(23) Paige, C. C.; Saunders, M. A. *SIAM journal on numerical analysis* **1975**, *12*, 617−629.

(24) 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.

(25) Shi, Y.; Xia, Z.; Zhang, J.; Best, R.; Wu, C.; Ponder, J. W.; Ren, P. *J. Chem. Theory Comput.* **2013**, *9*, 4046−4063.

(26) Tinker-HP. http://www.ip2ct.upmc.fr/tinkerHP/, 2015.

(27) Starovoytov, O. N.; Torabifard, H.; Cisneros, G. A. s. *J. Phys. Chem. B* **2014**, *118*, 7156−7166.

(28) Tuckerman, M.; Berne, B. J.; Martyna, G. J. *J. Chem. Phys.* **1992**, *97*, 1990−2001.

DOI: 10.1021/acs.jctc.6b00981
J. Chem. Theory Comput. 2017, 13, 180−190

As of now, we have a new polarization solver that enables simulations both with improved stability and increased computational efficiency. To provide a more severe test, we will now focus on the computations of free energies, trying to evaluate TCG's ability for this sensitive task.

Bibliography

[1] F. Lipparini, L. Lagardère, B. Stamm, E. Cancès, M. Schnieders, P. Ren, Y. Maday, and J.-P. Piquemal, "Scalable Evaluation of Polarization Energy and Associated Forces in Polarizable Molecular Dynamics: I. Toward Massively Parallel Direct Space Computations," *Journal of Chemical Theory and Computation*, vol. 10, pp. 1638–1651, Apr. 2014.

[2] N. J. Higham, *Accuracy and stability of numerical algorithms*. Philadelphia: Soc. for Industrial and Applied Math, 2. ed ed., 2002. OCLC: 265000938.

[3] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes - The Art of Scientific Computing*. Cambridge University Press, 3rd ed., 2007.

[4] P. Ahlström, A. Wallqvist, S. Engström, and B. Jönsson, "A molecular dynamics study of polarizable water," *Molecular Physics*, vol. 68, pp. 563–581, Oct. 1989.

[5] J. Kolafa, "Numerical Integration of Equations of Motion with a Self-Consistent Field given by an Implicit Equation," *Molecular Simulation*, vol. 18, pp. 193–212, Oct. 1996.

[6] J. Kolafa, "Time-Reversible Always Stable Predictor-Corrector Method for Molecular Dynamics of Polarizable Molecules," *Journal of Computational Chemistry*, vol. 25, no. 3, pp. 335–342, 2004.

[7] D. V. Belle, M. Froeyen, G. Lippens, and S. J. Wodak, "Molecular dynamics simulation of polarizable water by an extended Lagrangian method," *Molecular Physics*, Aug. 2006.

[8] R. Car and M. Parrinello, "Unified Approach for Molecular Dynamics and Density-Functional Theory," *Physical Review Letters*, vol. 55, pp. 2471–2474, Nov. 1985.

[9] M. Tuckerman, *Statistical Mechanics: Theory and Molecular Simulation*. OUP Oxford, Feb. 2010. Google-Books-ID: Lo3JqcOpgrcC.

[10] A. M. N. Niklasson, "Extended Born-Oppenheimer Molecular Dynamics," *Physical Review Letters*, vol. 100, p. 123004, Mar. 2008.

[11] A. Albaugh, O. Demerdash, and T. Head-Gordon, "An efficient and stable hybrid extended Lagrangian/self-consistent field scheme for solving classical mutual induction," *The Journal of Chemical Physics*, vol. 143, p. 174104, Nov. 2015.

[12] E. Picard, "Mémoire sur la théorie des équations aux dérivées partielles et la méthode des approximations successives," *Journal de mathématiques pures et appliquées*, vol. 6, pp. 145–210, 1890.

[13] J. R. Shewchuk, "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain," Aug. 1994.

[14] L. Lagardère, F. Lipparini, E. Polack, B. Stamm, E. Cancès, M. Schnieders, P. Ren, Y. Maday, and J.-P. Piquemal, "Scalable evaluation of polarization energy and associated forces in polarizable molecular dynamics: II. toward massively parallel computations using smooth particle mesh ewald," *Journal of Chemical Theory and Computation*, vol. 11, pp. 2589–2599, June 2015.

[15] Y. Saad, *Iterative methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2nd ed., 2003.

[16] A. C. Simmonett, F. C. Pickard, Y. Shao, T. E. Cheatham, and B. R. Brooks, "Efficient treatment of induced dipoles," *The Journal of Chemical Physics*, vol. 143, p. 074115, Aug. 2015.

[17] A. C. Simmonett, F. C. Pickard, J. W. Ponder, and B. R. Brooks, "An empirical extrapolation scheme for efficient treatment of induced dipoles," *The Journal of Chemical Physics*, vol. 145, p. 164101, Oct. 2016.

[18] P. Pulay, "Convergence acceleration of iterative sequences. the case of scf iteration," *Chemical Physics Letters*, vol. 73, pp. 393–398, July 1980.

[19] P. Pulay, "Improved SCF convergence acceleration," *Journal of Computational Chemistry*, vol. 3, no. 4, pp. 556–560, 1982.

[20] T. Rohwedder and R. Schneider, "An analysis for the DIIS acceleration method used in quantum chemistry calculations," *Journal of Mathematical Chemistry*, vol. 49, pp. 1889–1914, Oct. 2011.

[21] W. Wang and R. D. Skeel, "Fast evaluation of polarizable forces," *The Journal of Chemical Physics*, vol. 123, p. 164107, Oct. 2005.

[22] F. Aviat, A. Levitt, B. Stamm, Y. Maday, P. Ren, J. W. Ponder, L. Lagardère, and J.-P. Piquemal, "Truncated Conjugate Gradient: An Optimal Strategy for the Analytical Evaluation of the Many-Body Polarization Energy and Forces in Molecular Simulations," *Journal of Chemical Theory and Computation*, vol. 13, pp. 180–190, Jan. 2017.

[23] D. Nocito and G. J. O. Beran, "Fast divide-and-conquer algorithm for evaluating polarization in classical force fields," *The Journal of Chemical Physics*, vol. 146, no. 11, p. 114103, 2017.

[24] H. Steinhaus, "Sur la division des corps matériels en parties," *Cl.III*, vol. IV, no. 12, pp. 801–804, 1956.

[25] D. Nocito and G. J. O. Beran, "Massively Parallel Implementation of Divide-and-Conquer Jacobi Iterations Using Particle-Mesh Ewald for Force Field Polarization," *Journal of Chemical Theory and Computation*, vol. 14, pp. 3633–3642, July 2018.

[26] F. Lipparini, L. Lagardère, C. Raynaud, B. Stamm, E. Cancès, B. Mennucci, M. Schnieders, P. Ren, Y. Maday, and J.-P. Piquemal, "Polarizable Molecular Dynamics in a Polarizable Continuum Solvent," *Journal of Chemical Theory and Computation*, vol. 11, pp. 623–634, Feb. 2015.

[27] D. Nocito and G. J. O. Beran, "Reduced computational cost of polarizable force fields by a modification of the always stable predictor-corrector," *The Journal of Chemical Physics*, vol. 150, p. 151103, Apr. 2019.

[28] F. Aviat, L. Lagardère, and J.-P. Piquemal, "The truncated conjugate gradient (TCG), a non-iterative/fixed-cost strategy for computing polarization in molecular dynamics: Fast evaluation of analytical forces," *The Journal of Chemical Physics*, vol. 147, p. 161724, Aug. 2017.

[29] J. W. Ponder, C. Wu, P. Ren, V. S. Pande, J. D. Chodera, M. J. Schnieders, I. Haque, D. L. Mobley, D. S. Lambrecht, R. A. DiStasio, M. Head-Gordon, G. N. I. Clark, M. E. Johnson, and T. Head-Gordon, "Current Status of the AMOEBA Polarizable Force Field," *The Journal of Physical Chemistry B*, vol. 114, pp. 2549–2564, Mar. 2010.

[30] P. Ren and J. W. Ponder, "Polarizable Atomic Multipole Water Model for Molecular Mechanics Simulation," *The Journal of Physical Chemistry B*, vol. 107, pp. 5933–5947, June 2003.

[31] O. N. Starovoytov, H. Torabifard, and G. A. Cisneros, "Development of AMOEBA Force Field for 1,3-Dimethylimidazolium Based Ionic Liquids," *The Journal of Physical Chemistry B*, vol. 118, pp. 7156–7166, June 2014.

[32] E. J. Maginn, R. A. Messerly, D. J. Carlson, D. R. Roe, and J. R. Elliott, "Best Practices for Computing Transport Properties 1. Self-Diffusivity and Viscosity from Equilibrium Molecular Dynamics [Article v1.0]," *Living Journal of Computational Molecular Science*, vol. 1, p. 6324, Dec. 2018.

[33] M. Holz, S. R. Heil, and A. Sacco, "Temperature-dependent self-diffusion coefficients of water and six selected molecular liquids for calibration in accurate 1h NMR PFG measurements," *Physical Chemistry Chemical Physics*, vol. 2, pp. 4740–4742, Jan. 2000.

[34] A. Albaugh, M. E. Tuckerman, and T. Head-Gordon, "Combining Iteration-Free Polarization with Large Time Step Stochastic-Isokinetic Integration," *Journal of Chemical Theory and Computation*, vol. 15, pp. 2195–2205, Apr. 2019.

[35] R. Neidinger, "Introduction to Automatic Differentiation and MATLAB Object-Oriented Programming," *SIAM Review*, vol. 52, pp. 545–563, Jan. 2010.

Chapter 3

Towards faster free energies

Contents

Beyond the properties we have computed so far (energies, radial distribution functions, diffusion constants), there is the free energy. The free energy is a quantity of prime interest in a number of fields that requires complex methods and careful attention to be computed.

In this chapter, we will firstly give a small presentation of the free energy itself, then explain some of the common methods used to compute it. We then move on to exploit the Truncated Conjugate Gradient built in the previous chapter to observe its applicability when facing such a difficult task.

3.1 Free energy

Like the entropy or the internal energy, free energy is a thermodynamic state function. Strictly speaking, a difference of free energy is defined as the amount of work needed to perform a thermodynamic transformation of a system from one state to another, provided that this transformation follows a reversible path.

It is a useful quantity, as it indicates whether the given transformation is thermodynamically favorable (if the free energy difference decreases) or not (if the free energy difference increases). Its magnitude is linked to the probability for the transformation to occur. For example, the binding of a ligand by a protein's active site can be characterized with a free energy difference between the "unbounded" and the "bound" states. If this free energy difference is smaller for a ligand 1 than for a ligand 2, then it will indicate that ligand 1 will bind more easily with the protein. This quantity is therefore very relevant in biochemistry, or even drug-design.

Two free energy state functions actually exist, namely the Helmholtz and the Gibbs one. *Helmholtz* free energy, noted A, is defined as

$$A(N, V, T) = E(N, V, S) - TS(N, V, T) \tag{3.1}$$

with T the temperature, E the internal energy, S the entropy. It is used when temperature and volume are fixed (in the canonical ensemble).

Gibbs free energy is the equivalent for the isobaric ensemble, with

$$G(N, P, T) = A(N, V(P), T) + PV(P) \tag{3.2}$$

It measures the work needed to perform a transformation this time at fixed pressure and temperature.

It should also be noted that the free energy is function of the internal energy and so includes a potential energy. As such, the free energy of one single system depends on the "zero" chosen, and one will rather look at free energy *differences* (usually denoted ΔA or ΔG), more meaningful.

Another important point should be made about free energy. Any state function obeys the following property: considering any thermodynamic transformation, the change of a state function only depends on the *initial and final states*.

This will hence also be the case for free energy, meaning that the calculation of a ΔG (or ΔA) only requires knowledge of the beginning and final state. As we will see in the next sections however, the computation may be numerically impeded if these two state are too different. In this case, the process can be divided in several short stages, changing a transformation $A \longrightarrow B$ into $A \longrightarrow \alpha_1 \longrightarrow \alpha_2 \longrightarrow \ldots \longrightarrow B$. Any such sequence of intermediate states starting from the initial and finishing at the final state can thus be used to compute free energy differences. Such a sequence is called a *thermodynamic path* (or sometimes, perhaps slightly imprecisely, a thermodynamic cycle).

3.1.1 Calculation method

We stated in the beginning of section 3.1 that, since it carries a potential energy contribution, free energy is usually computed as differences between two states.

Assuming we have two well defined states A and B, we will present in the following sections a few of the methods that can be used to compute ΔA_{AB}, the free energy difference between states A and B.

We then quickly evoke another type of free energy calculation, designed for dealing with rare events.

Free Energy Perturbation (FEP)

Focusing on the Helmholtz free energy for the remainder of this chapter, we will first introduce a few notations. We will first focus on partition functions. In the canonical ensemble, the partition function of a system is usually written as

$$Q(N,V,T) = \frac{1}{N!h^{3N}} \int \mathrm{d}^N\mathbf{r}\, \mathrm{d}^N\mathbf{p} \mathrm{e}^{-\beta H(\mathbf{r},\mathbf{p})} \tag{3.3}$$

where h is Planck's constant, and H is the system's Hamiltonian

$$H(\mathbf{r},\mathbf{p}) = \sum_{i=1}^{N} \frac{\vec{p_i}^2}{2m_i} + U(\mathbf{r}) \tag{3.4}$$

(U designates the potential energy). Integration of the momenta allows one to rewrite the partition function as

$$Q(N,V,T) = \frac{1}{N!\lambda_{dB}^{3N}} \int \mathrm{d}^N\mathbf{r}\ \mathrm{e}^{-\beta U(\mathbf{r})} = \frac{Z(N,V,T)}{N!\lambda_{dB}^{3N}} \tag{3.5}$$

where λ_{dB} is the de Broglie thermal wavelength. This defines a new quantity Z called the configurational partition function, which depends on the positions and the potential energy: $Z = \int \mathrm{d}^N\mathbf{q}\ \mathrm{e}^{-\beta U(\mathbf{q})}$

One can show that the free energy can be written as

$$A(N,V,T) = -kT\ \ln(Q(N,V,T)) \tag{3.6}$$

equivalently, a free energy difference between two states A and B would yield:

$$\Delta A_{AB} = A_A - A_B = -kT\ \ln\left(\frac{Q_B}{Q_A}\right) = -kT\ \ln\left(\frac{Z_B}{Z_A}\right) \tag{3.7}$$

This rewriting does however not yield a straightforward quantity to be computed, as molecular dynamics do not give access directly to partition functions.

Yet one can rewrite one of the configurational partition functions as

$$\begin{aligned} Z_B &= \int \mathrm{d}^N\mathbf{r}\ \mathrm{e}^{-\beta U_B(\mathbf{r})}\mathrm{e}^{-\beta U_A(\mathbf{r})}\mathrm{e}^{\beta U_A(\mathbf{r})} && (3.8)\\ &= \int \mathrm{d}^N\mathbf{r}\ \mathrm{e}^{-\beta U_A(\mathbf{r})}\mathrm{e}^{-\beta(U_B(\mathbf{r})-U_A(\mathbf{r}))} && (3.9) \end{aligned}$$

The ratio Z_B/Z_A becomes

$$\begin{aligned} \frac{Z_B}{Z_A} &= \frac{1}{Z_A}\int \mathrm{d}^N\mathbf{r}\ \mathrm{e}^{-\beta U_A(\mathbf{r})}\mathrm{e}^{-\beta(U_B(\mathbf{r})-U_A(\mathbf{r}))} && (3.10)\\ &= \left\langle \mathrm{e}^{-\beta(U_B(\mathbf{r})-U_A(\mathbf{r}))}\right\rangle_A && (3.11) \end{aligned}$$

where $\langle\ldots\rangle_A$ stands for an average with respect to the distribution of the state A. The free energy difference becomes

$$\Delta A_{AB} = -kT\ \ln\left\langle \mathrm{e}^{-\beta(U_B(\mathbf{r})-U_A(\mathbf{r}))}\right\rangle_A \tag{3.12}$$

Note that the inverse pathway is also possible, and leads to the mirror formula

$$\Delta A_{AB} = kT \ln \left\langle e^{-\beta(U_A(\mathbf{r})-U_B(\mathbf{r}))} \right\rangle_B \tag{3.13}$$

with 3.12 usually referred to as "forward FEP" and 3.13 as "backward FEP".

Computing an average such as the one in 3.10 supposes to first simulate the system in state A, using the potential correctly describing this state and storing the positions adopted by the system along this trajectory. Then, using the stored configurations, compute $\exp(U_B(\mathbf{r}))$ (and $\exp(U_A(\mathbf{r}))$, if that was not already saved during the dynamics) for each of them. Finally, a simple statistical average has to be done.

The mathematical shape of the average taken here raises an important caveat. For any configuration such that the potential energy difference $U_A - U_B$ is large, then $e^{-\beta(U_A-U_B)}$ will become very small. The concerned configuration will thus have a very little weight in the total average.

This method thus requires the probed states to be reasonably close, such that there is a sufficient overlap in the potential energy surfaces (PES). If this is not the case, one possible solution is to divide the thermodynamic path $A \longrightarrow B$ in several smaller steps $A \longrightarrow \alpha_1 \longrightarrow \ldots \longrightarrow \alpha_s \longrightarrow B$, where the difference between two intermediate steps α_i and α_{i+1} potential energy surfaces would be smaller (and thus the overlap of the potential energy surfaces better).

One can then compute, for each pair of states, the associated free energy difference $\Delta A_{\alpha_i \alpha_{i+1}}$, to finally sum them all as

$$\Delta_{AB} = \Delta A_{A\alpha_1} + \sum_{i=1}^{s-1} \Delta A_{\alpha_j \alpha_{j+1}} + \Delta A_{\alpha_s B} \tag{3.14}$$

(s represents the number of intermediates steps). This assumes that a simple procedure can be found for designing this intermediate steps. The computation of hydration free energies in this work will present one such possibility.

Thermodynamical integration

The Free Energy Perturbation approach, as seen above, is based on a discrete sequence of intermediate states (from α_1 to α_s), built so that the overlap between two successive potential energy surface is important enough for the average to be computed smoothly.

Yet one may want to find a more continuous approach, for which there would not be a set of intermediate states but where the system evolves continuously from state A to state B.

To implement this idea using the framework we defined in the previous section, one can introduce a λ parameter, varying between 0 and 1. A new potential can then be defined as follows

$$U(\mathbf{r}, \lambda) = (1 - \lambda)U_A(\mathbf{r}) + \lambda U_B(\mathbf{r}) \tag{3.15}$$

such that, when $\lambda = 0$, $U = U_A$ (in other words, the system is in state A) and when $\lambda = 1$, $U = U_B$ (the system is in state B).

Let us then start by differentiating equation 3.6 with respect to our parameter λ:

$$\begin{aligned}
\frac{\partial A}{\partial \lambda} &= -\frac{kT}{Z}\frac{\partial Z}{\partial \lambda} && (3.16)\\
&= -\frac{kT}{Z}\int \mathrm{d}^N\mathbf{r}\ \frac{\partial}{\partial \lambda}\left[\mathrm{e}^{-\beta U(\mathbf{r},\lambda)}\right] && (3.17)\\
&= -\frac{1}{Z}\int \mathrm{d}^N\mathbf{r}\ \frac{1}{\beta} \times (-\beta)\frac{\partial U}{\partial \lambda}\mathrm{e}^{-\beta U(\mathbf{r},\lambda)} && (3.18)\\
&= \left\langle \frac{\partial U}{\partial \lambda} \right\rangle && (3.19)
\end{aligned}$$

Using this result in the simple following relation

$$\Delta A_{AB} = \int_0^1 \mathrm{d}\lambda\ \frac{\partial A}{\partial \lambda} \tag{3.20}$$

one gets

$$\Delta A_{AB} = \int_0^1 \mathrm{d}\lambda\ \left\langle \frac{\partial U}{\partial \lambda} \right\rangle_\lambda \tag{3.21}$$

Here, $\langle \ldots \rangle_\lambda$ stands for an average over the ensemble whose probability distribution is $\exp\left(U(\mathbf{r}, \lambda)\right)$. Given the very simple shape of our potential (3.15), this equation simply becomes

$$\Delta A_{AB} = \int_0^1 \mathrm{d}\lambda\ \langle U_B - U_A \rangle_\lambda \tag{3.22}$$

However, the potential U can be changed to use more complex switching functions of the λ parameter

$$U(\mathbf{r}, \lambda) = f_1(\lambda)U_A(\mathbf{r}) + f_2(\lambda)U_B(\mathbf{q}) \tag{3.23}$$

provided that $f_1(0) = 1$, $f_1(1) = 0$, $f_2(0) = 0$ and $f_2(1) = 1$. This allows for more efficient integration schemes.[1]

In practice, one defines a set of n_λ values of λ. For each λ_i, a simulation is carried out, and $\left\langle \frac{\partial U}{\partial \lambda_i} \right\rangle_{\lambda_i}$ is computed. The final result is then obtained by numerical integration.

Bennett Acceptance Ration (BAR)

In order to improve the precision of the Free Energy Perturbation method, C. Bennett[2] proposed a method using simulations of both states A and B (instead of one state only for the FEP). Inspired from the Monte-Carlo jumps, the method imagines a different kind of move, where the configuration is kept but the potential is switched from U_A to U_B (or vice-versa).

Such a "jump" would change the energy by a quantity ΔU, and the probability for it to be accepted would be $M(\beta \Delta U)$, with the Metropolis function $M(x) = \min[1, \exp(-x)]$.

The detailed balance condition that must be respected reads

$$M\left[\beta U_B(\mathbf{r}) - \beta U_A(\mathbf{r})\right] \mathrm{e}^{-\beta U_A(\mathbf{r})} = M\left[\beta U_A(\mathbf{r}) - \beta U_B(\mathbf{r})\right] \mathrm{e}^{-\beta U_B(\mathbf{r})} \tag{3.24}$$

Integrating over phase space, and multiplying by $\frac{Z_A}{Z_A}$ on the left hand side and by $\frac{Z_B}{Z_B}$ on the right hand side gives:

$$Z_A \frac{\int \mathrm{d}^N\mathbf{r}\; M\left[\beta U_B(\mathbf{r}) - \beta U_A(\mathbf{r})\right] \mathrm{e}^{-\beta U_A(\mathbf{r})}}{Z_A} = Z_B \frac{\int \mathrm{d}^N\mathbf{r}\; M\left[\beta U_A(\mathbf{r}) - \beta U_B(\mathbf{r})\right] \mathrm{e}^{-\beta U_B(\mathbf{r})}}{Z_B} \tag{3.25}$$

Recognizing configurational averages on both sides of the equation above, we can rewrite it as

$$\frac{Z_A}{Z_B} = \frac{\left\langle M\left[\beta(U_A(\mathbf{r}) - U_B(\mathbf{r}))\right]\right\rangle_B}{\left\langle M\left[\beta(U_B(\mathbf{r}) - U_A(\mathbf{r}))\right]\right\rangle_A} \tag{3.26}$$

which gives us access to the $\frac{Z_A}{Z_B}$ ratio, and thus to the free energy through equation 3.7.

Bennett extended this to any weighting function $W(\mathbf{q})$ to replace the Metropolis function, and showed that the optimal (most accurate) choice of weighting function changed equation 3.26 into

$$\frac{Z_A}{Z_B} = \frac{\left\langle f\left(U_A - U_B + C\right)\right\rangle_B}{\left\langle f\left(U_B - U_A - C\right)\right\rangle_A} \exp(C) \tag{3.27}$$

with f the Fermi function $f(x) = 1/(1 + \exp(x))$, and

$$C = \ln(Z_B n_A / Z_A n_B) \tag{3.28}$$

a constant, n_A and n_B the number of statistically independent configurations from state A and B.

This shapes a procedure to compute free energies as follows.

1. Carry out a simulation of the system in state A, another in state B. Store both trajectories.
2. Compute U_A and U_B for each configuration in trajectory A, then B.
3. Calculate the C value as the fixed point of an iterative sequence.

Computation of C

Let us write C as an iterative sequence C_n. Starting with $C_0 = 0$, one can calculate the left-hand side of equation 3.27, namely $\frac{Z_A}{Z_B}$. Given the definition of C (eq. 3.28), this gives

$$C_1 = \ln \frac{Z_B n_A}{Z_A n_B} \tag{3.29}$$

C_1 can then be reintroduced in 3.27, to yield a new value for $\frac{Z_A}{Z_B}$. The general sequence reads

$$\begin{cases} C_0 = 0 \\ C_{n+1} = \ln \left[\frac{n_A}{n_B} \times \frac{Z_B}{Z_A} \right] \end{cases} \tag{3.30}$$

$\frac{Z_B}{Z_A}$ being a function of C_n. This sequence will converge towards the value C verifying 3.27.

The conditions for a FEP calculation to converge are more stringent, and one expects the BAR method to converge easier. This can be explained by the necessity of overlap between potential energy surfaces (PES) that drives both methods; with BAR, the use of trajectories produced using both PES minimizes the overlap problem. The BAR was hence the method we chose to evaluate free energies in this work.

Nevertheless, a separation in intermediates steps is still possible, as explained in 3.1.1 (eq. 3.14) if the overlap between potential energy surfaces of states A and B was still not sufficient.

Free energy bootstrap

In order to improve the estimation of statistical quantities, one can use the *bootstrap* method. Although no "extra" information can be gained from a finite set of data through this method, it allows to refine the quality of the statistical quantities that were computed.
The bootstrap procedure, in its simplest form[a], consists firstly in repeating the following two steps k times:

1. extract a random subset of the whole dataset one is analyzing;
2. compute and store statistical quantities (averages, standard deviations).

Using the k averages and standard deviations obtained, one can then compute an "average of averages", which improves the statistical uncertainty.

[a] A wide variety of bootstrap types exist, depending on the type of problem being studied. We will narrow our description to the simplest case, which we are effectively using.

3.1.2 Sampling rare events

Let us suppose that we are interested in the unfolding of a protein. Keeping our notations, A would be the folded state and B the unfolded one. The complexity of proteins raises an important problem if one wants to use the methods presented above: how does one define the intermediate hamiltonians used to describe the evolution from A to B ? Besides, the time scale on which a protein folds or unfolds is generally unreachable for typical simulations, as it is usually in the microsecond range.

To measure whether the folding is happening, one can define *reaction coordinates* (or "collective variables"), which is a function of a subset of the particles coordinates. As a (very simple) example, in our protein folding case, it could be the distance between the nucleic acids at each extremity of the proteic chain ($r_{CR} = ||r_1 - r_n||$). When this distance is, say r_f, the protein is considered to be folded, while when it reaches a certain distance $r_u > r_f$, it is considered to be unfolded.

One possible solution to force the slow process to happen at a much faster pace than normal would then be to restrain the system, forcing him to follow the reaction coordinate. This is the objective of the *umbrella sampling* technique, which adds a bias to the potential. To that end, one can define a sequence of progressing values s_i of the reaction coordinate, such that s_1 accounts for the value of the reaction coordinate in the initial state, and s_n is its equivalent in the final state. The bias is then chosen as a harmonic potential whose equilibrium values is the value s_i, so that the movements of the reaction coordinate is restrained between the s values.

$$U_{\text{bias}}(r_{\text{CR}}, s_i) = \frac{1}{2}\kappa(r_{\text{CR}} - s_i)^2 \tag{3.31}$$

One can then perform a sequence of molecular dynamics simulation, using the biasing $U_{\text{bias}}(r_{\text{CR}}, s_1)$, then $U_{\text{bias}}(r_{\text{CR}}, s_1)$ and so on until $U_{\text{bias}}(r_{\text{CR}}, s_n)$. It gives access to the probability distribution of the value of the reaction coordinate knowing that a bias was added to restrain it around s_i.

This method enforces the system to undergo the required transformation. However, it uses a modified potential, and the results of the simulations can not be directly analyzed to produce the free energies, as one needs to *unbias* them. This is usually done using the WHAM (Weighted Histogram Analysis Model),[3] allowing one to extract the most accurate free-energy differences values and minimizing statistical errors. Without entering further in details, several other rare-events sampling methods exist, although they may appear as being more complex (*e.g.* the Blue Moon ensemble,[4] Steered Molecular Dynamics,[5] OSRW[6]).

3.2 TCG vs Free energies

3.2.1 Hydration free energy

In this work, we computed *hydration* free energies. These are the energy difference between a system in the vacuum and a system solvated in water. A hydration free energy provides two informations on the hydration (or solvation in water):

- firstly, the energy difference arising from the switching from solute-solute and solvent-solvent interactions to solute-solvent ones, which could be designated as dissolution energy;

- secondly, the entropy difference between the ordering of the pure solvent and the disorder introduced by the solute, that we could designate as dissolution entropy.

A much broader interest can be seen in these computations, thanks to the state function nature of the free energy. The use of thermodynamic paths is a very powerful tool to use when considering complicated processes. Let us imagine for example a protein and a ligand, for which we want to evaluate the binding free energy. The binding process can be complicated and usually occurs at time-scales that are very long. This complexity is worsened by the presence of the water solvent, which occupy the active sites where the ligand should connect with the protein, on top of being an important computational additional cost for the simulation.

Instead of having to perform very long simulations, hoping for the binding to occur, a smarter path can be followed, as illustrated in figure 3.1. One could first compute the free energy corresponding to

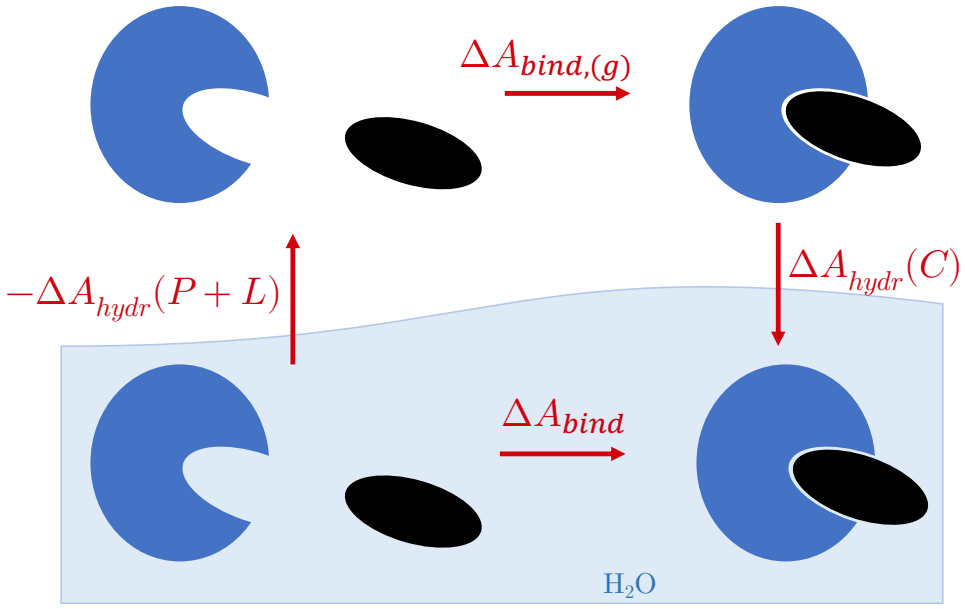


Figure 3.1: Thermodynamic pathway for the computation of the binding free energy of a protein P with a ligand L, forming the complex C.

the desolvation of both protagonists (the protein and the ligand), which would be the opposite of the hydration free energy ($-\Delta A_{\text{hydr}}(P+L)$ on the figure). On the other end, the hydration free energy for the *complex* (where the ligand has attached to the protein) can be computed ($\Delta A_{\text{hydr}}(C)$ on the figure). The missing step is finally the binding one, which will be much easier to carry out in gas phase, without

solvent molecules ($\Delta A_{\text{bind,(g)}}$). The total pathway can be summarized as

$$\Delta A_{\text{bind}} = -\Delta A_{\text{hydr}}(P + L) + \Delta A_{\text{bind,(g)}} + \Delta A_{\text{hydr}}(C) \tag{3.32}$$

Hydration free energies are thus a convenient tool in order to simplify complex free energy calculations.

3.2.2 Hydration free energies of ions

In order to test our Truncated Conjugate Gradient algorithm, we computed hydration free energies for the sodium cation. Ions are a good starting point for free energies, as they have been extensively studied in the past. Their small size is also helpful as it limits the computational effort, and will allow us to run a good amount of tests in order to assess the TCG's behaviour. Lastly, they represent a model validation before switching to more important and interesting systems like proteins.

The hydration process was decomposed in a three-steps thermodynamical path. Firstly, the solute is discharged in a vacuum, meaning that its charge is reduced to zero. Then, the solute is placed in the solvent, and the van der Waals interactions are turned on, while electrostatic terms are still null. Finally, the electrostatic interactions are reactivated.

The first step of the thermodynamical path supposes no energy change, as there is no interaction involved, and thus no energy difference between the charged ion in the vacuum (initial state) and the non-charged ion in the vacuum (final state).

The challenge comes from the second and third steps. Focusing on the activation of the van der Waals force, the strategy used was the following. A scaling parameter λ_{VdW} was used, with initial value $\lambda_{\text{VdW}} = 0$ (where there is no van der Waals interaction) and final value $\lambda_{\text{VdW}} = 1$ (where van der Waals interactions are fully taken into account). Seven intermediate steps between these boundaries in the thermodynamic path were done ($\lambda_{\text{VdW}} = \{0.4, 0.5, 0.6, 0.65, 0.7, 0.8, 0.9\}$).

The same progressive approach was followed during the third step for reactivating the charges: a scaling parameter λ_{elec} was switched from 0 to 1 progressively in eight steps ($\lambda_{\text{elec}} = \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$). Overall, we thus considered 19 states, whose

potential depended on these two parameters $U(\lambda_{\text{VdW}}, \lambda_{\text{elec}})$.

For each of these states, also commonly referred to as "windows", a 2 ns simulation in the NVT ensemble was performed. The first 400 ps of each trajectory were discarded as thermalization and equilibration, and BAR calculations were carried out using the last 1.6 fs.

Solver	δt	Prec	Guess	Peek	ΔA_{hydr} (kcal/mol)	Error (%)
pcg5A	2				-91.67	0.0
TCG1	2	-	-	-	-58.07	-57.8
TCG1	2	●	-	-	-80,37	-14
TCG1	2	●	●	-	-74.06	-23.7
TCG1	2	-	-	$\omega_{\text{fit}} = 1.153$	-68.86	-33.1
TCG1	2	●	●	$\omega = 1$	-86.94	-5.4
TCG1	2	●	●	$\omega_{\text{fit}} = 1.234$	-90.86	-0.9
TCG2	1	-	-	-	-91.75	0.1
TCG2	2	-	-	-	-83.27	-10.1
TCG2	2	●	-	-	-89.32	-2.6
TCG2	2	●	-	$\omega = 1$	-89.17	-2.8
TCG2	2	-	-	$\omega_{\text{fit}} = 1.153$	-91.95	0.3
TCG2	2	●	●	$\omega = 1$	-91.46	-0.2
TCG2	2	●	●	$\omega_{\text{fit}} = 1.518$	-92.97	1.4

Table 3.1: **Free energy values computed with BAR** calculation method, using different polarization solvers[ii]. A = ASPC for the PCG ones

Note that, in this table, ω_{fit} is presented as a fixed value. Computation of ω_{fit} was indeed done prior to the sequence of thermodynamic windows, in order to prevent problems in the free energy integration (a detailed explanation is provided in the next section).

At first glance, the behaviour that was observed earlier regarding TCG performances seems to be reproduced here, with TCG1 allowing for a good first approximation for a low computational price, while TCG2 provides results much more accurate, comparable to the reference (PCG) values, and come with a

[ii]To be able to compare these values to experiments and previous computations,[7] one needs to account for the standard state difference: in simulation, it is 1 mol/L, where in experiment it is one atom only. A free energy difference thus ensues: $-RT \ln\left(\frac{V_{1\,atom}}{V_{1\,mol/L}}\right) = 1.84$ kcal/mol, which needs to be added to the simulation results.

slightly higher price.

However, one can observe that free energy computations are very sensitive. Where relative errors were rather small when looking at potential or polarization energies, larger deviations are measured here. Even more striking, the "naked" first order (TCG1 without any refinement) yielded trajectories on which BAR or FEP calculations could simply not converge. This could be fixed by adding extra steps in the scaling parameters switching (λ_{VdW} and λ_{elec}), in order to reduce the difference between two successive potential energy surfaces (*i.e.* obtaining a better overlap), and improve the precision of the BAR and FEP calculations.

This would however mean more simulations, and it might be wiser to simply choose a better precision version of the TCG, agreeing to pay the extra-price it supposes, so as to ultimately reduce the total computation price. Nevertheless, TCG appears as a perfectly viable method to compute hydration free energies, provided that we use refinement – and using a proper ω_{fit}, as we will discuss in the next session.

It is also interesting to note the difference of behaviour regarding the integration time-step: for the non-refined TCG2, we computed the whole set of simulations (switching λ_{VdW} then λ_{elec}) once using a 1 fs time-step, and another time using a 2 fs time-step (using the RESPA integration, as presented in next chapter). The results were considerably improved, and this is quite easily explained: the configuration obtained after integrating over a large time-step will carry a bigger error, and a "rougher" TCG (such as TCG2 without any refinement) will have a harder time projecting back on the correct potential energy surface. Results show how helpful the peek-step can be in this regard. On the contrary, with a smaller (1 fs) time-step, the error accumulated is smaller, and it is therefore easier for a less advanced version of the TCG to correct it.

As observed in the previous chapter, we also note here that, when using a fully refined TCG2, there seems to be better results when the peek-step is scaled by $\omega = 1$ than when it is scaled by ω_{fit}. This may be partly caused by the fixing of ω_{fit} that we will explain in next section, but this is also consistent with previous observation.

Again, the very good results obtained by a TCG2 with no other refinement than the peek-step using ω_{fit} is clear here, which confirms its usefulness (foreseen in 2.3.5) as a fast and accurate solver.

3.2.3 The ω_{fit} question

In chapter 2, we saw the usefulness of the peek-step regarding the improvement of TCG results. In particular, the use of a fitted ω scalar allows one to obtain excellent agreement on the energies, and appears as the method of choice. One should however be very cautious when trying to compute free energies using this refinement.

Indeed, as was shown in 3.1.1, the free energy calculation methods (whether we talk about Free Energy Perturbation, Bennett Acceptance Ratio or Thermodynamic Integration) all rest on the computation of ensemble averages that are defined by a potential. It is therefore of predominant importance that this potential is well defined. This is the case when using the Truncated Conjugate Gradient. As detailed in the previous chapter, the polarization energy follows an analytical formula, and is thus very precisely defined.

However, the implementation presented earlier for the peek-step using fitting is based on a regular recomputation of the ω_{fit} scalar to reproduce as closely as possible the polarization energy. While this is the key element for very accurate reproduction of energies, this also means that, for every time ω_{fit} is recomputed, the total potential energy changes, and the potential energy surface being explored is also different. To put it otherwise, the definition of the polarization energy changes.

As a consequence, when computing the free energy difference using values of ω_{fit} refitted "on the fly", the resulting free energy does not only account for the apparition of a charged solute in a solvent, but also... for the change in the polarization energy, even though it does not correspond to any physically meaning transformation. The final extracted result would thus be polluted with an artifact arising from the ongoing change in the polarization energy definition.

It appears clearly that one should decide on a fixed value for ω beforehand, and keep this value over the course of the numerical simulations. For the same reason, it should also be the same for every "window" (every value of the λ_{VdW} and λ_{elec} parameter).

Choosing this fixed value is not an obvious question. To explore the different values that ω_{fit} adopts over a free energy calculation, we focused on a TCG2 computation. For each windows, we started from a configuration equilibrated over 2 ns using a tightly converged PCG solver. We then switched to the Truncated Conjugate Gradient, using a diagonal preconditioner, the direct field guess, and a peek-step

using an ω_{fit} refitted every 1000 steps. After a 400 ps run, successive ω_{fit} values were extracted and averaged.

Distinct differences arise when comparing thermodynamic windows with each other: during the activation of the van der Waals terms, ω_{fit} doesn't change, as it account only for the polarization between the water molecules. Nevertheless, it drops from 1.8 to 1.5 during the activation of the charges terms, as polarization involving the ion is progressively more and more important.

The choice was then made to use the value of ω_{fit} for every thermodynamic window, in order to fit as well as possible the energy when the solvated ion is fully recharged in the solvent. Admittedly, this won't give optimal results for all the other windows, but it will allow us to preserve a constant definition of the polarization energy over the course of the computation, ultimately ensuring that the resulting free-energy values are properly defined.

3.2.4 BAR reweighting: making more with less (a glimpse on the next step)

Pursuing on the objective of obtaining better results using cheaper computational methods, the *reweighting* method[8] (also named *importance sampling*) was proposed as a solution to use trajectories performed at low accuracy level to extract informations corresponding to higher level dynamics.

The idea is to carry out a simulation using a fast model (such as for example TCG1), and then to use the successive configurations extracted from this simulation to re-compute the energies using a more accurate model (such as TCG2). As configurations are only saved every n_{save} time-steps, the amount of calculation required to post-treat the trajectory will be negligible compared to the initial simulation. Besides, the (expensive) gradients that would be needed for an accurate simulations are not computed here, which is also a source of computation savings.

In this section, we will explain the principles of the reweighting. Let us imagine that one is trying to compute the average of a function $b(\mathbf{r})$. Let us note H_1 and H_2 two possible Hamiltonians to describe the system, with H_2 being more accurate than H_2. The best ensemble average of function b would be obtained using the most precise Hamiltonian, and thus reads

$$\langle b(\mathbf{r}) \rangle_{H_2} = \frac{\int \mathrm{d}^N\mathbf{r}\ b(\mathbf{r}) \mathrm{e}^{-\beta H_2}}{\int \mathrm{d}^N\mathbf{r}\ \mathrm{e}^{-\beta H_2}} = \frac{\int \mathrm{d}^N\mathbf{r}\ b(\mathbf{r}) \mathrm{e}^{-\beta U_2}}{\int \mathrm{d}^N\mathbf{r}\ \mathrm{e}^{-\beta U_2}} \tag{3.33}$$

A simple multiplication by the neutral term $\exp(\beta H_1 - \beta H_1)$ then yields

$$\frac{\int \mathrm{d}^N\mathbf{r}\; b(\mathbf{r})\mathrm{e}^{-\beta U_2}\mathrm{e}^{\beta U_1}\mathrm{e}^{-\beta U_1}}{\int \mathrm{d}^N\mathbf{r}\; \mathrm{e}^{-\beta U_2}\mathrm{e}^{\beta U_1}\mathrm{e}^{-\beta U_1}} = \frac{\int \mathrm{d}^N\mathbf{r}\; b(\mathbf{r})\mathrm{e}^{-\beta(U_2-U_1)}\mathrm{e}^{-\beta U_1}}{\int \mathrm{d}^N\mathbf{r}\; \mathrm{e}^{-\beta(U_2-U_1)}\mathrm{e}^{-\beta U_1}} \tag{3.34}$$

By multiplying by the ratio Z_1/Z_1 (with Z_1 the configuration partition function associated to the potential U_1) and defining $\Delta U = U_2 - U_1$, one can recognize ensemble averages as

$$\frac{\int \mathrm{d}^N\mathbf{r}\; b(\mathbf{r})\mathrm{e}^{-\beta\Delta H}\mathrm{e}^{-\beta U_1}}{Z_1}\frac{Z_1}{\int \mathrm{d}^N\mathbf{r}\; \mathrm{e}^{-\beta\Delta U}\mathrm{e}^{-\beta U_1}} = \frac{\left\langle b(\mathbf{r})\mathrm{e}^{-\beta\Delta U}\right\rangle_{U_1}}{\left\langle \mathrm{e}^{-\beta\Delta U}\right\rangle_{U_1}} \tag{3.35}$$

Finally, one has a formula that gives an ensemble average over H_2 distribution as a ratio of ensemble averages... over H_1 distributions.

$$\langle b(\mathbf{r})\rangle_{U_2} = \frac{\left\langle b(\mathbf{r})\mathrm{e}^{-\beta\Delta U}\right\rangle_{U_1}}{\left\langle \mathrm{e}^{-\beta\Delta U}\right\rangle_{U_1}} \tag{3.36}$$

A practical implementation of this method would thus work as follows

1. perform a simulation using the U_1 hamiltonian, storing potential energy values and the trajectory (list of successive configurations of the system);

2. compute U_2 for all configurations in the trajectory file;

3. compute the averages at the right-hand side of equation 3.36.

As a first validation test, reweighting was applied to the computation of polarization and potential energies. Trajectories of 200 ps were computed, with one frame extracted every ps.

Table 3.2 summarizes these first results. The reference simulation was carried out using a Conjugate Gradient solver with a 10^{-8} convergence criterion. The "fast" trajectory was computed using a non-refined TCG1. Energies were then computed from this trajectory using the reweighting procedure. The accurate Hamiltonian (H_2) used here was the reference one (CG with a 10^{-8} convergence criterion). The objective was thus to measure how well we could reproduce the CG results using TCG1 with the reweighting.

	E_{pol}	Error	E_{pot}	Error
Ref.	-2163	-	-4470	-
TCG1 (non-refined)	-1599	29%	-4008	10%
Reweighting	-1912	12%	-4289	4%

Table 3.2: **Reweighting applied to polarization and potential energies.** Here, the "reweighting" gives energy values obtained using the reweighting method, applied to a TCG1 simulation (with no refinement used), reweighted following a tightly converged CG solver. Reference is a Conjugate Gradient simulation converged with a 10^{-8} criterion. Error is computed with respect to the reference value. Energies are given in kcal/mol.

For both the polarization and potential energies, the error observed is divided by more than two when using the reweighting scheme. The reweighting thus seems to allow for a good correction, even-though it does not manage to fully reproduce the reference energies.

This reweighting can be applied in a straightforward way to the FEP (eq. 3.12) in order to compute the average $\langle \exp[-\beta(U_A - U_B)] \rangle_B$. Reintroducing this in equation 3.36 simply gives

$$\langle \exp[-\beta(U_A - U_B)] \rangle_{B,2} = \frac{\left\langle e^{-\beta(U_A-U_B)} e^{-\beta \Delta U} \right\rangle_{B,1}}{\left\langle e^{-\beta \Delta U} \right\rangle_{B,1}}, \qquad \Delta U = U_2 - U_1 \tag{3.37}$$

Here, $\langle \ldots \rangle_{B,1}$ designates an ensemble average performed over the distribution generated by Hamiltonian $H_{B,1}$, that is, the Hamiltonian describing state B with the "low" accuracy level. $\langle \ldots \rangle_{B,2}$ follows the same notation, with $H_{B,2}$ the "high" accuracy level Hamiltonian describing state B.

The reweighting was tested on the FEP computation method on two thermodynamical windows (*i.e* firstly for $\lambda_{\text{VdW}} = 0.7$ to $\lambda_{\text{VdW}} = 0.8$, secondly for $\lambda_{\text{elec}} = 0.9$ to 1). Both forward and backward FEP were computed (see equations 3.12 and 3.13).

These windows were randomly selected, and serve here as a simple proof of principle for the applicability of the reweighting to free energies. Of course, the long-term objective is rather to use this process on the BAR computation method. Nevertheless, these preliminary results are very encouraging, as they show good improvement of the free energy values on different thermodynamic windows, and with a reduced computational time consumed.

	$\lambda_{\text{VdW}} = 0.7 \rightarrow 0.8$		$\lambda_{\text{elec}} = 0.9 \rightarrow 1$	
	Backward FEP	Forward FEP	Backward FEP	Forward FEP
Ref.	-0.3979	-0.3446	-19.62	-19.53
TCG2 (non-refined)	-0.2849 (28%)	-0.3628 (5%)	-19.01 (3%)	-19.07 (2%)
Reweighting	-0.3720 (7%)	-0.3413 (1%)	-19.55 (<1%)	-19.47 (<1%)

Table 3.3: **Reweighting applied to FEP.** Here, the "reweighting" gives energy values obtained using the reweighting method, applied to a TCG1 simulation (with no refinement used), reweighted following a tightly converged CG solver. Reference is a Conjugate Gradient simulation converged with a 10^{-8} criterion. Next to the TCG2 and reweighted energies, the relative error with respect to the reference value is given. Energies are given in kcal/mol.

Application to the BAR follows the same idea, though BAR already uses two trajectories (one following Hamiltonian H_A, the other following H_B). Adding the reweighting will thus require a total of four sets of energy values: $U_{A,1}$, $U_{A,2}$, $U_{B,1}$ and $U_{B,2}$, with A and B denoting the thermodynamic states defining the transformation, and 1 and 2 being the two levels of precision.

Two ensemble averages are needed in order to compute the BAR free energies: $\langle f\,(U_A - U_B + C)\rangle_B$ and $\langle f\,(U_B - U_A - C)\rangle_B$ (see eq. 3.27). We're trying to extract the best available precision there (corresponding to Hamitonian H_2) while using trajectories produced using the – less accurate – Hamiltonian H_1. The two ensemble averages of interest are thus $\langle f\,(x)\rangle_{B,2}$ and $\langle f\,(-x)\rangle_{A,2}$, with $x = U_A - U_B + C$

Using the reweighting technique on each, one obtains:

$$\begin{cases} \langle f\,(U_A - U_B + C)\rangle_{B,2} &= \dfrac{\left\langle f(U_A - U_B + C)\exp\left[-\beta(U_{B,2} - U_{B,1})\right]\right\rangle_{B,1}}{\left\langle \exp\left[-\beta(U_{B,2} - U_{B,1})\right]\right\rangle_{B,1}} \\ \langle f\,(U_B - U_A - C)\rangle_{A,2} &= \dfrac{\left\langle f(U_B - U_A - C)\exp\left[-\beta(U_{A,2} - U_{A,1})\right]\right\rangle_{A,1}}{\left\langle \exp\left[-\beta(U_{A,2} - U_{A,1})\right]\right\rangle_{A,1}} \end{cases} \tag{3.38}$$

which effectively changes the final ratio to

$$\frac{Z_A}{Z_B} = \frac{\left\langle f\,(U_A - U_B + C)\exp\left[-\beta(U_{B,2} - U_{B,1})\right]\right\rangle_{B,1}}{\left\langle f\,(U_B - U_A - C)\exp\left[-\beta(U_{A,2} - U_{A,1})\right]\right\rangle_{A,1}} \frac{\left\langle \exp\left[-\beta(U_{A,2} - U_{A,1})\right]\right\rangle_{A,1}}{\left\langle \exp\left[-\beta(U_{B,2} - U_{B,1})\right]\right\rangle_{B,1}} \exp(C) \tag{3.39}$$

Using this ratio, the BAR method can be computed as previously: using $C_0 = 0$, one can compute the ratio $\frac{Z_A}{Z_B}$, which can then be used to compute an improved value C_1 for the C constant. This iteration

is then repeated until the $C_{(n)}$ sequence converges. The only major difficulty here is the treatment of all different energetic values.

This method is rather representing the very near future of our computations, as we are confident that it will yield good results while saving precious computational time.

We could also mention that this is an *a posteriori* solution to accelerate free energy computations. Indeed, all the improvement presented in this work (both regarding polarization solver and integrators) are all devoted to a more efficient handling of the Molecular Dynamics, whereas the reweighting technique is based on a better, smarter analysis of the data after their production, which was less at the heart of our focus.

The reweighting procedure finally appears as a good method to accelerate accuracy demanding computations such as free energies estimations. It however provides slightly degraded values. It should therefore be used in situations were "quick and cheap" estimations are require without the need of high precision. Such an approach could be used for example in high-throughput screening studies of large numbers of compounds where an initial assessment at reasonable (but not full) accuracy of free energies, aiming to eliminate low affinity compounds, is sufficient.

3.2.5 Conclusion

To summarize, two strategies were proposed in this chapter to compute free energies. The first one is the use of the Bennett Acceptance Ratio method, where TCG2 proved to be a perfectly viable candidate. Care has to be taken if adding a peek-step, as ω_{fit} has to be well controlled.

The second one allows the use of less accurate polarization solvers, but comes with a high gain in terms of computational time. More tests have to be undertaken, but preliminary results are encouraging. This appears as a good solution for anybody trying to compute fast free energies without requiring a very high precision.

We can now affirm that TCG solvers are perfectly viable when it comes to the sensitive computation of free energies. We also have very convincing tests aiming at accelerated computation of free energies using reweighting. Nevertheless, the amount of computation required to simulate all thermodynamic windows in order to properly compute free energies is still quite heavy. Aggregating the various features

that were presented so far, we will thus explore another way to further accelerate our computations: the integrators.

Bibliography

[1] M. Tuckerman, *Statistical Mechanics: Theory and Molecular Simulation*. OUP Oxford, Feb. 2010. Google-Books-ID: Lo3Jqcopgrc C.

[2] C. H. Bennett, "Efficient estimation of free energy differences from Monte Carlo data," *Journal of Computational Physics*, vol. 22, pp. 245–268, Oct. 1976.

[3] S. Kumar, J. M. Rosenberg, D. Bouzida, R. H. Swendsen, and P. A. Kollman, "THE weighted histogram analysis method for free-energy calculations on biomolecules. I. The method," Oct. 1992.

[4] M. Sprik and G. Ciccotti, "Free energy from constrained molecular dynamics," *The Journal of Chemical Physics*, vol. 109, pp. 7737–7744, Nov. 1998.

[5] B. Isralewitz, M. Gao, and K. Schulten, "Steered molecular dynamics and mechanical functions of proteins," *Current Opinion in Structural Biology*, vol. 11, pp. 224–230, Apr. 2001.

[6] L. Zheng, M. Chen, and W. Yang, "Random walk in orthogonal space to achieve efficient free-energy simulation of complex systems," *Proceedings of the National Academy of Sciences*, vol. 105, pp. 20227–20232, Dec. 2008.

[7] M. J. Schnieders, J. Baltrusaitis, Y. Shi, G. Chattree, L. Zheng, W. Yang, and P. Ren, "The structure, thermodynamics, and solubility of organic crystals from simulation with a polarizable force field," *Journal of Chemical Theory and Computation*, vol. 8, no. 5, pp. 1721–1736, 2012. PMID: 22582032.

[8] J. Zhang, Y. Shi, and P. Ren, "Polarizable Force Fields for Scoring Potein-Ligand Interactions," in *Protein-Ligand Interactions*, vol. 53, Wiley-VCH, holger gohlke ed., 2012.

Chapter 4

Towards faster free energies, another strategy: improved integrators

Contents

4.1 Advanced integrators: multi-timestepping

After our developments regarding polarization solvers, we now focus on the role of molecular dynamics *integrators*, and we will more specifically look into the case of Langevin dynamics. In this chapter, the

reader will find a brief summary of the multi-timestep methods, then a step-by-step description of the implementation of optimal integration algorithms.

4.1.1 Classical integrator

In order to study integrator shapes and properties, let us first introduce a few notations related to classical mechanics.

We will firstly write the Hamiltonian $\mathcal{H}$ as

$$\mathcal{H}(\mathbf{r}, \mathbf{p}) = E_{\text{kin}} + E_{\text{pot}} = \sum_{i=1}^{N} \frac{||\vec{p}_i||^2}{2m_i} + U(\mathbf{r}) \tag{4.1}$$

It is simply the total energy of a system. The position and momenta of all particles in our system define a point in the $6N$ dimensional domain called the *phase space*, that we could note $x = (\mathbf{r}, \mathbf{p})$. The generic properties we considered in chapter 1, just like the Hamiltonian, are all functions of the phase space vector x.

The time derivative of such a property, who has no *explicit* dependence in time ($\frac{\partial b}{\partial t} = 0$), can be expressed using the chain rule as:

$$\frac{\mathrm{d}b}{\mathrm{d}t} = \sum_{i=1}^{N} \left(\frac{\partial b}{\partial \vec{r}_i} \frac{\partial \vec{r}_i}{\partial t} + \frac{\partial b}{\partial \vec{p}_i} \frac{\partial \vec{p}_i}{\partial t} \right) \tag{4.2}$$

Hamilton's equation[1] give $\frac{\partial \vec{r}_i}{\partial t} = \frac{\partial \mathcal{H}}{\partial \vec{p}_i}$ and $\frac{\partial \vec{p}_i}{\partial t} = -\frac{\partial \mathcal{H}}{\partial \vec{r}_i}$, which leads to a final expression for this time derivative:

$$\frac{\mathrm{d}b}{\mathrm{d}t} = \sum_{i=1}^{N} \left(\frac{\partial b}{\partial \vec{r}_i} \frac{\partial \mathcal{H}}{\partial \vec{p}_i} + \frac{\partial b}{\partial \vec{p}_i} \frac{\partial \mathcal{H}}{\partial \vec{r}_i} \right) \tag{4.3}$$

This can be said more simply

$$\frac{\mathrm{d}b}{\mathrm{d}t} = iLb \tag{4.4}$$

by defining the Liouville operator L as

$$iL = \sum_{i=1}^{N} \frac{\partial H}{\partial \vec{p}_i} \frac{\partial}{\partial \vec{r}_i} - \frac{\partial H}{\partial \vec{r}_i} \frac{\partial}{\partial \vec{p}_i} \tag{4.5}$$

A formal solution of eq. 4.4 gives us $b(x_t) = \mathrm{e}^{iLt} b(x_t)$. When choosing the phase-space vector itself for our $b(x_t)$ function, we get

$$x(t) = \mathrm{e}^{iLt} x(0) \tag{4.6}$$

which is a formal expression to write the time integration of our system, based on the so-called *classical propagator* e^{iLt}. This simple expression hides the complexity of integration schemes: we have no possibility to exactly express the result of eq. 4.6 in general (and especially for our N-particles systems).

Starting from this point, we will try to derive approximations to help us designing dynamics integration schemes.

4.1.2 Trotter theorem

The Liouville operator is the sum of two terms L_1 and L_2.

$$iL_1 = \sum_{i=1}^{N} \frac{\partial H}{\partial \vec{p}_i} \frac{\partial}{\partial \vec{r}_i} = \sum_{i=1}^{N} \dot{\vec{r}}_i \frac{\partial}{\partial \vec{r}_i} = \sum_{i=1}^{N} \frac{\vec{p}_i}{m} \frac{\partial}{\partial \vec{r}_i} \tag{4.7}$$

corresponds to the kinetic part of the Liouville operator, and

$$iL_2 = \sum_{i=1}^{N} -\frac{\partial H}{\partial \vec{r}_i} \frac{\partial}{\partial \vec{p}_i} = \sum_{i=1}^{N} -\dot{\vec{p}}_i \frac{\partial}{\partial \vec{p}_i} = \sum_{i=1}^{N} F(\mathbf{r}) \frac{\partial}{\partial \vec{p}_i} \tag{4.8}$$

corresponds to the forces.

However, these two operators do not commute and as such, we can't express the classical propagator in 4.6 as a product of exponential:

$$\mathrm{e}^{iLt} \neq \mathrm{e}^{iL_1 t} \mathrm{e}^{iL_2 t} \tag{4.9}$$

This would allow us to evaluate these operator sequentially (computing first the result of the action of $\mathrm{e}^{iL_2 t}$ on our phase-space vector, then of $\mathrm{e}^{iL_2 t}$), which we could do exactly !

One thus needs to resort to an approximation of the classical propagator. For that purpose, the *Trotter theorem*[2] (or Strang splitting[3] formula) states that for two non-commuting operators A and B,

$$\mathrm{e}^{A+B} = \lim_{P \to \infty} \left(\mathrm{e}^{\frac{B}{2P}} \mathrm{e}^{\frac{A}{P}} \mathrm{e}^{\frac{B}{2P}} \right)^P \tag{4.10}$$

Applying this to our classical propagator, we have

$$e^{iLt} = \lim_{P\to\infty} \left(e^{iL_2 \frac{t}{2P}} e^{iL_1 \frac{t}{P}} e^{iL_2 \frac{t}{2P}} \right)^P \tag{4.11}$$

By defining a time-step length $\Delta t = t/P$, and taking the $1/P$ power of both sides, we finally get an approximation that will prove useful in the next steps of this work.

$$e^{iL\Delta t} = e^{iL_2 \frac{\Delta t}{2}} e^{iL_1 \Delta t} e^{iL_2 \frac{\Delta t}{2}} + O(\Delta t^2) \tag{4.12}$$

4.1.3 The RESPA splits

If we consider the energies that have to be computed to integrate the dynamics at each time-step, as shown in 1.3, we can see that two families are distinguished: the bonded terms (intramolecular interactions) on one hand , and the non-bonded ones (intermolecular interactions) on the other. The non-bonded energy terms are varying fast, given the stiffness of the mathematical shape they follow, and the physical reality they try to reproduce. The forces deriving from these energies also vary with high frequency. If we want to properly observe these movements, we will need a fine time-step δt.

Meanwhile, the non-bonded terms (van der Waals, electrostatics, polarization) vary much slower (especially so at long distance), such that computing their derivatives as often as we compute the bonded-term forces – as it is the case for example in the velocity-Verlet scheme – leads to considerable amount of time spent on non-necessary calculations. Choosing to use a bigger time-step for all the simulation would not allow to reproduce the fastly varying terms, and would lead to accumulation of energy in these modes. It is thus not a viable solution. This loss is worsened by the complexity of the non-bonded terms compared to the bonded ones: the Lennard-Jones potential and the electrostatic interactions are computed between all pairs of atoms, and chapter 2 may have already convinced the reader of the high price of polarization energy calculations.

An ideal integrator would thus juggle between two time-step lengths, one for the high-frequency energy terms, the second for the low-frequency ones. The framework presented in sections 4.1.1 and 4.1.2 allows to design it in a rigorous manner.

If we decide to explicitly distinguish the slow and fast evolving forces in L_2, with:

$$iL_2 = iL_{\text{f,slow}} + iL_{\text{f,fast}} = \sum_{i=1}^{N} \left[F_{\text{slow}}(\mathbf{r}) + F_{\text{fast}}(\mathbf{r})\right] \frac{\partial}{\partial \vec{p}_i} \tag{4.13}$$

Then it is possible to redefine two operators (whose sum would still give the total Liouville operator)

$$iL_{\text{fast}} = \sum_{i=1}^{N} \left[F_{\text{fast}}(\mathbf{r}) \frac{\partial}{\partial \vec{p}_i} + \frac{\vec{p}_i}{m} \frac{\partial}{\partial \vec{r}_i}\right] = iL_{\text{f,fast}} + iL_1 \tag{4.14}$$

$$iL_{\text{slow}} = \sum_{i=1}^{N} \left[F_{\text{slow}}(\mathbf{r}) \frac{\partial}{\partial \vec{p}_i}\right] = iL_{\text{f,slow}} \tag{4.15}$$

Applying Trotter theorem here reads:

$$e^{iL\Delta t} \simeq e^{iL_{\text{slow}}\frac{\Delta t}{2}} e^{iL_{\text{fast}}\Delta t} e^{iL_{\text{slow}}\frac{\Delta t}{2}} \tag{4.16}$$

But the Trotter theorem can also be applied to the $\exp(iL_{\text{fast}}\delta t)$ term. If one defines a smaller time-step $\delta t = \frac{\Delta t}{n}$, the same pathway leads to

$$e^{iL_{\text{fast}}\Delta t} = \left(e^{iL_{\text{f,fast}}\frac{\delta t}{2}} e^{iL_1\delta t} e^{iL_{\text{f,fast}}\frac{\delta t}{2}}\right)^n \tag{4.17}$$

Finally, by reusing expression 4.17 in 4.16, we get

$$e^{iL\Delta t} = e^{iL_{\text{slow}}\frac{\Delta t}{2}} \left(e^{iL_{\text{f,fast}}\frac{\delta t}{2}} e^{iL_1\delta t} e^{iL_{\text{f,fast}}\frac{\delta t}{2}}\right)^n e^{iL_{\text{slow}}\frac{\Delta t}{2}} \tag{4.18}$$

The final expression here shows an operator to integrate our equations that is more finely adapted to the specifics of the dynamics: one can choose a Δt that is adapted to describe the slowly varying terms and a δt for the high-frequency ones, under the condition that $\Delta t = n \times \delta t$, with n an integer.

For a total simulation time T, this scheme also allows us to compute the most expensive terms belonging to L_{slow} $n/2$ times less often than the fast ones, which will translate into substantial accelerations of the computation, as we will see in 4.2.1.

The time-step choice will become an important question to properly use this method. A few trials and errors, monitoring the simulation properties, gives us an effective answer. Yet, following a more

rigorous approach, one could also calculate the Fourier transform of the energy in a reference simulation in order to extract the typical times that should be chosen.

RESPA1 – One step further

Quite naturally, one may want to design integrators which marry the simulation frequencies even better. A more refined splitting could be designed for that purpose, where force terms would divide between fast, slow, and intermediate ones.

Introducing an extra splitting is quite straightforward, and would yield a propagator of the following shape

$$e^{iL\Delta T} = e^{iL_{\text{f,slow}}\frac{\Delta T}{2}}\left[e^{iL_{\text{f,interm}}\frac{\Delta t}{2}}\left(\mathrm{e}^{iL_{\text{f,fast}}\frac{\delta t}{2}}\mathrm{e}^{iL_1\delta t}\mathrm{e}^{iL_{\text{f,fast}}\frac{\delta t}{2}}\right)^n e^{iL_{\text{f,interm}}\frac{\Delta t}{2}}\right]^m e^{iL_{\text{f,slow}}\frac{\Delta T}{2}} \tag{4.19}$$

This would require three time-steps, here noted δt, Δt and ΔT, with the conditions that

$$\begin{cases} \Delta T = m\Delta t \\ \Delta t = n\delta t \end{cases} \tag{4.20}$$

The splitting of the forces that was adopted will be discussed in section 4.2.2.

4.2 An iterative search for the optimal integrator

Using the splittings proposed earlier, this section details the designing of an optimal integrator. The objective is to be able to use the biggest possible time-steps (hence accelerating the computations) while preserving the correctness of the simulations.

As a reference, we use a Velocity-Verlet integrator as presented in 1.2, with a 0.5 fs time-step. It is a widely used algorithm giving us results we can trust. This time-step choice is quite small, to ensure that we're considering conservative dynamics. In general practices, however, the common choice for the time-step of the Velocity-Verlet is rather 1 fs. As a consequence, while we will compare the simulation accuracy with the 0.5 fs reference, we will calculate the *speedup* with respect to the 1 fs Velocity-Verlet. This allows us to remain certain of the quality of the simulations, while not overselling the gains.

Lastly, the ASPC belong to the predictor-corrector family and can be seen as a particular guess

involved in the polarization solver that accelerates the computation. Usually, such algorithm are used at the cost of time-reversibility (see sec. 2.1.2). In this particular case, ASPC is designed to preserve as much as possible time-reversibility making it a production algorithm for standard polarizable MD simulations. Therefore, speedups are compared to a standard 1 fs Velocity-Verlet but but the algorithm is not effective for timesteps larger than 2fs where computational gains disappear and instabilities greatly increase.

For a better understanding and comparison of the algorithms, we shall give pseudocode expressions detailing the integrators presented. For the Velocity-Verlet reference, this reads

Algorithm 1 : Velocity-Verlet

$\vec{p}_i \leftarrow \vec{p}_i + \delta t/2 \times \vec{f}_i$
$\vec{q}_i \leftarrow \vec{q}_i + \delta t \times \vec{p}_i/m_i$
Compute forces using updated $\vec{q}_i$'s.
$\vec{p}_i \leftarrow \vec{p}_i + \delta t/2 \times \vec{f}_i$

4.2.1 V-RESPA: a first splitting of the forces

Applying the RESPA logic detailed in 4.1, we split the terms of our AMOEBA force field between the bonded and non-bonded ones, in order to integrate less frequently the slowly evolving terms. The Velocity-Verlet shape is kept, as one can see in the algorithm hereafter, and we thus denoted this integrator as *V-RESPA*, where "V" stands for Velocity-Verlet. The integration algorithm will read

Algorithm 2 : V-RESPA – Using $\Delta t = n\delta t$

$\vec{p}_i \leftarrow \vec{p}_i + \Delta t/2 \times \vec{f}_{i,\text{SLOW}}$
for i = 1, n **do**
  $\vec{p}_i \leftarrow \vec{p}_i + \delta t/2 \times \vec{f}_{i,\text{FAST}}$
  $\vec{q}_i \leftarrow \vec{p}_i + \delta t \times \vec{p}_i/m_i$
  Compute FAST forces using updated $\vec{q}_i$'s.
  $\vec{p}_i \leftarrow \vec{p}_i + \delta t/2 \times \vec{f}_{i,\text{FAST}}$
end for
Compute SLOW forces using updated $\vec{q}_i$'s.
$\vec{p}_i \leftarrow \vec{p}_i + \Delta t/2 \times \vec{f}_{i,\text{SLOW}}$

The advantage of using such a scheme clearly appears, as we directly see that the fast evolving forces will be computed n times more than the slow ones.

System	Prop.	Verlet 0.5fs	0.25/2	0.25/3
Water (S3)	E_{pot}	-4459 ±39	-4447 ±36	-4408 ±39
	E_{pol}	-2169 ±41	-2145 ±39	-2108 ±39
	D	2.08 ±0.03	2.13 ±0.02	2.25 ±0.02
Ubiquitin	E_{pot}	-27894 ±102	-27850 ± 105	-27628 ±95
	E_{pol}	-13052 ±98	-12917 ±99	-12708 ±95
Speedup	(vs. ASPC)	–	1.75	2.37
	(vs. no ASPC)	–	2.53	3.42

Table 4.1: **V-RESPA integrators.** E_{pot} and E_{pol} designate potential and polarization energy respectively, both given in kcal/mol. Error computed with respect to the reference (Verlet 0.5 fs). D is the diffusion constant (expressed $\times 10^{-5}$ cm^2/s) [ii]. Speedup are computed vs. a 1 fs Velocity-Verlet reference, both with and without ASPC.

In order to assess the various integrators qualities, we performed computations on the system S3 introduced in chap. 2, a cubic box containing 500 water molecules, as well as on the solvated ubiquitin protein (1233 atoms for the protein chain, plus 2835 water molecules). Simulations were run over 2 ns, in the NVT ensemble with $T = 300$ K. Potential and polarization energies were calculated for both systems. To check dynamical and structural properties, diffusion constants and radial distribution functions were also calculated.

Table 4.2.1 presents the results obtained with V-RESPA integrators. Each is noted as $\delta t/\Delta t$, following notations previously defined. For example, the "0.25/2" integrator means that the time-step for the integration of the fast varying forces is 0.25 fs, and 2 fs for the slowly varying forces.

The errors obtained on the energies are very satisfactory, remaining under 3%. Diffusion coefficient also show the reliability of this integrator. In both case, the errors are maximal for the largest outer time-step (Δt) of 3 fs, indicating that the larger this time-step gets, the more inaccurate integration becomes. The radial distribution functions, in figure 4.1 also yield convincing results, in the shape of a very good agreement with the reference curves. The first and second peaks, displayed with a larger scale, are very well reproduced.

[ii] The astute reader will remark a difference between the reference diffusion coefficient given in this table and the one given in chap. 2. These two sets of computation (chap. 2 and chap. 4) were in fact carried out using different force-field parameters (two different versions of AMOEBA's water). As this value is only useful as a reference point, this has no influence on our reasoning.

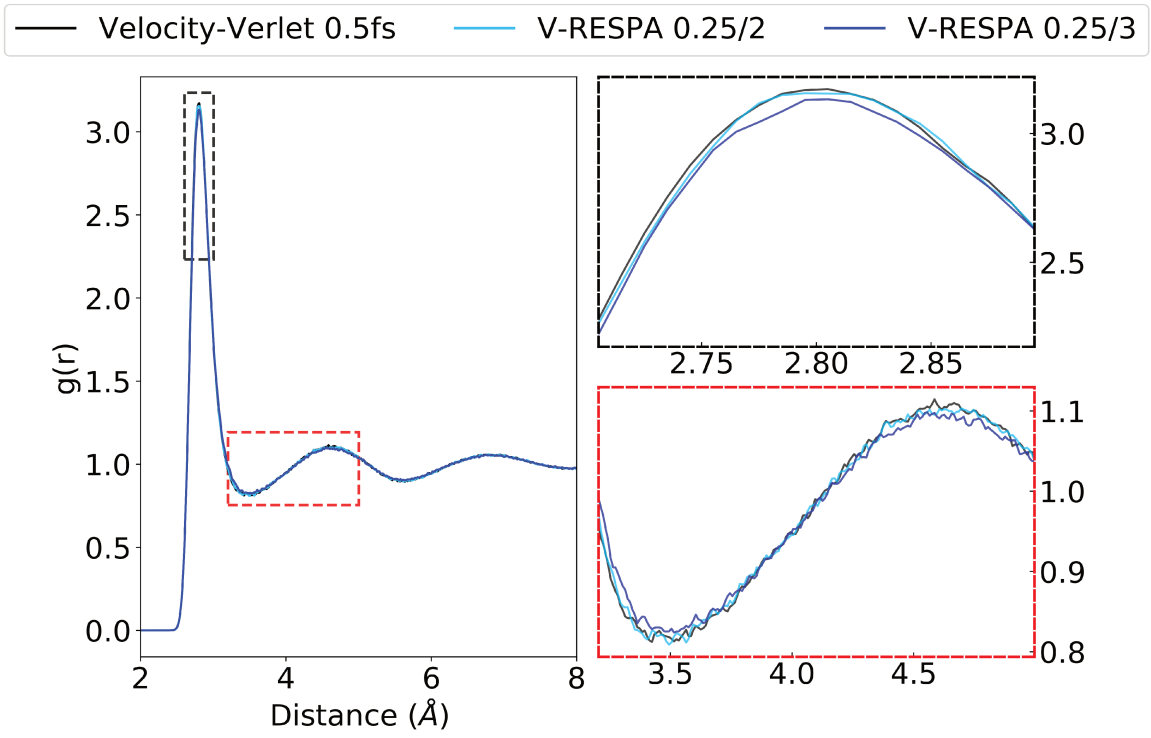


Figure 4.1: **V-RESPA integrators.** Oxygen-oxygen radial distribution function, comparerd to the Velocity-Verlet reference using $\delta t = 0.5$ fs. Top- and bottom-right panels show a magnification of the black and red boxes drawn on the left figure, corresponding to the first two peaks of the radial distribution function.

The first integrator (0.25/2) rewards the user with a speedup of 1.75 (2.53 when comparing to the – slower – non-ASPC Velocity-Verlet); the second (0.25/3) yields speedups of 2.37 (3.42 with a non-ASPC reference). So far, we thus have faster dynamics while preserving accuracy.

4.2.2 RESPA1: pushing the splitting

To further pursue the integration acceleration, the splitting logic can be pushed in order to distinguish three time-steps, following the RESPA1 logic. In the previous section, we splitted forces between the bonded and non-bonded ones. Focusing on the latter, one could argue that their typical evolution times, although longer than the intramolecular terms, span quite a wide range. Indeed, imagining two atoms i and j, the closer they are together, the more a small variation of r_{ij} (corresponding to a displacement over one time-step) will impact the interaction. At long range, the value of $\frac{1}{r_{ij}^n}$ does not vary much, while they do at short range.

We thus splitted the non-bonded potential terms using a distance cutoff. It was smoothed out by

a switching function, so that the transition between the domains would not be brutal. For example, the electrostatic terms between atoms separated by 5 Angstroms or less are treated as real *short-range* terms, by 5 to 7 Angstroms as real *long-range* terms, and above 7 Angstroms as reciprocal terms in the PME framework.

Switching function

The switching function S was defined as follows by Margul *et al.*[4]

$$S(r, r_c, \lambda) = \begin{cases} 1 & \text{if } r \leq r_c - \lambda \\ g(r, r_c, \lambda) & \text{if } r_c - \lambda \leq r \leq r_c \\ 0 & \text{if } r \leq r_c \end{cases} \tag{4.21}$$

where r is the distance between two atoms, r_c the short vs. long-range cutoff distance, and λ is a switching parameter controlling the smoothing length (we used $\lambda = 0.5$ Angstroms). The function g is defined as

$$g(r, r_c, \lambda) = 1 + u^3(15u - 6u^2 - 10) \tag{4.22}$$

$$u = \frac{1}{\lambda}(r - r_c + \lambda) \tag{4.23}$$

The corresponding pseudocode algorithm is reproduced hereunder; the "SLOW" forces are the long-range intermolecular ones, the "INTERM" forces are the short-range intermolecular ones, and the "FAST" forces are the intramolecular ones.

Algorithm 3 : V-RESPA1 – Using $\Delta t = n\delta t$ and $\Delta T = m\Delta t$

$\vec{p}_i \leftarrow \vec{p}_i + \Delta T/2 \times \vec{f}_{i,\text{SLOW}}$

for i = 1, m **do**

$\vec{p}_i \leftarrow \vec{p}_i + \Delta t/2 \times \vec{f}_{i,\text{INTERM}}$

for j = 1, n **do**

$\vec{p}_i \leftarrow \vec{p}_i + \delta t/2 \times \vec{f}_{i,\text{FAST}}$

$\vec{q}_i \leftarrow \vec{q}_i + \delta t \times \vec{p}_i/m_i$

Compute FAST forces using updated $\vec{q}_i$'s.

$\vec{p}_i \leftarrow \vec{p}_i + \delta t/2\times$

end for

$\vec{q}_i \leftarrow \vec{q}_i + \Delta t \times \vec{p}_i/m_i$

Compute INTERM forces using updated $\vec{q}_i$'s.

$\vec{p}_i \leftarrow \vec{p}_i + \Delta t/2 \times \vec{f}_{i,\text{INTERM}}$

end for

Compute SLOW forces using updated $\vec{q}_i$'s.

$\vec{p}_i \leftarrow \vec{p}_i + \Delta T/2 \times \vec{f}_{i,\text{SLOW}}$

Three V-RESPA1 integrators were tested: 0.25/2/4, 0.25/2.5/5 and 0.25/2/6. These notations stand for $\delta t/\Delta t/\Delta T$, *i.e.* the timesteps used for the integration of the fast/intermediate/slow evolving forces. Note that, as required by equations 4.20, they are integer multiple of each other.

The results are compiled within table 4.2 and figure 4.2. As one could expected, this method accelerates computation, further than the simple split of RESPA could, and one can reach a 3.7 speedup (2.53 vs. ASPC reference).

Looking at the water system's energies, all three setups seem to give correct result. However, the same can not be said when looking at the ubiquitin, particularly its polarization energy, for the 0.25/2.5/5 integrator, where the error rises to 7.5%. The better results obtained by the 0.25/2/6 integrator (the relative error on the polarization energy drops back to 3.1%) should be subject to caution, as they were obtained using an even larger outer timestep (ΔT). They may obtained through the refinement of the inner timestep, but could also be a simple result of error compensation, as we can't expect results to improve in accuracy if we increase time-step size.

System	Prop.	Verlet 0.5fs	0.25/2/4	0.25/2.5/5	0.25/2/6
Water (S_3)	E_{pot}	-4459 ±39	-4433 ±37	-4517 ±39	-4522 ±37
	E_{pol}	-2169 ±41	-2135 ±41	-2201 ±41	-2228 ±40
	D	2.08 ±0.03	2.18 ±0.03	1.6 ±0.02	1.34 ±0.01
Ubiquitin	E_{pot}	-27894 ±102	-27766 ±91	-28876 ±110	-28333 ±98
	E_{pol}	-13052 ±98	-12893 ±98	-14009 ±120	-13457 ±1000
Speedup	(vs. ASPC)	–	1.72	2.43	2.53
	(vs. no ASPC)	–	2.5	3.5	3.7

Table 4.2: **V-RESPA1 integrators.** E_{pot} and E_{pol} designate potential and polarization energy respectively, both given in kcal/mol. Error computed with respect to the reference (Verlet 0.5 fs). D is the diffusion constant (expressed $\times 10^{-5}$ cm^2/s). Speedup are computed vs. a 1 fs Velocity-Verlet reference, both with and without ASPC.

Much more troubling are the radial distributions and the diffusion coefficient: figure 4.2 clearly shows a discrepancy in the peaks with the 1 fs reference for the 0.25/2.5/5 and 0.25/2/6 integrators. The diffusion constants measured are even worse, as they drop down by 36%. These two elements indicate that the dynamics are poorly reproduced in the simulations, making this V-RESPA1 scheme non-viable in our pursuit for dynamics acceleration.[iii]

We thus needed a solution to improve the quality of our integration without sacrificing the speed gains obtained so far.

4.2.3 Reconsidering Langevin dynamics integration with BAOAB

Let us focus, starting from this point, on simulations performed in the NVT ensemble, describing systems in contact with a thermostat. Although we briefly presented the Molecular Dynamics framework in the beginning of this work, some details were voluntarily left aside, including the functioning of thermostats. In recent years, developments in Molecular Dynamics integration mostly come from mathematics, and where mostly built unpon Langevin dynamics, which are better understood from a mathematical point of view. Following these developments, let us present some of the ensuing derivation in line with our TCG research.

Thermostatting a system is not a trivial task, and several different algorithms can handle it. The

[iii] The first integrator (0.25/2/4) yields acceptable results, but the obtained speedup is not interesting enough.

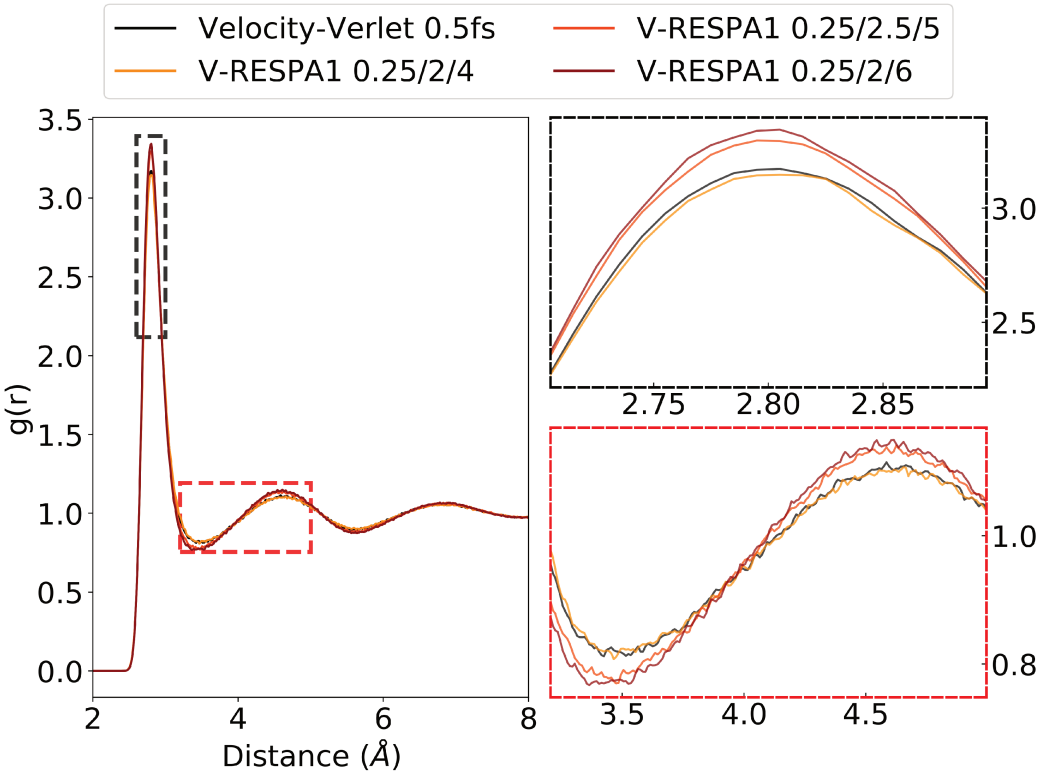


Figure 4.2: **V-RESPA1 integrators.** Oxygen-oxygen radial distribution function, comparerd to the Velocity-Verlet reference using $\delta t = 0.5$ fs. "V-RESPA" designates Velocity-Verlet-RESPA integrators. A difference with the Velocity-Verlet reference becomes apparent for the V-RESPA1 0.25/2.5/6 and 0.25/2/6 integrators.

most straightforward of them, albeit not the most physical, simply periodically rescales all velocities, such that the kinetic energy obtained would correspond to the chosen temperature.

Another possibility would be to set aside the formalism used so far, to express the equations of motion of our thermostatted system using Langevin equations:

$$\mathrm{d}\mathbf{q} = \frac{1}{\mathbf{M}}\mathbf{p}\mathrm{d}t \tag{4.24}$$

$$\mathrm{d}\mathbf{p} = -\nabla U(\mathbf{q})\mathrm{d}t - \gamma\mathbf{v}\mathrm{d}t + \sigma\mathbf{M}^{\frac{1}{2}}\mathrm{d}\mathbf{W} \tag{4.25}$$

Equation 4.24 contains the 2$^{\text{nd}}$ law of Newton, while 4.25 encompasses a *friction* term controlled by a friction coefficient γ, and $\mathrm{d}\mathbf{W}$ represents a vector infinitesimal Wiener process[iv]. This is a stochastic thermalization (given the Brownian intervention), wich acts locally (each degree of freedom is thermalized

[iv] A Wiener process is a stochastic process, with each step being uncorrelated to the previous one, whose increments follow a Gaussian distribution. It yields Brownian movement.

through its own noise). Of course, the intervention of a friction means that the dynamics are modified, if we compare to the Hamiltonian framework, which does not have friction.

Leimkuhler and Matthews[5, 6] proposed a method to integrate these equations starting on the separation hereafter:

$$\begin{bmatrix} \mathrm{d}\mathbf{q} \\ \mathrm{d}\mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{M}^{-1}\mathbf{p} \\ 0 \end{bmatrix} \mathrm{d}t + \begin{bmatrix} 0 \\ -\nabla U(\mathbf{q}) \end{bmatrix} \mathrm{d}t + \begin{bmatrix} 0 \\ -\gamma \mathbf{p} \mathrm{d}t + \sigma \mathbf{M}^{\frac{1}{2}} \mathrm{d}\mathbf{W} \end{bmatrix} \tag{4.26}$$

The three elements colored on eq. 4.26 are designated as A (in red), B (in blue) and O (in green), respectively.

Taking each of these elements separated, one can note that they build equations that can be exactly solved. This is quite obvious for A and B, as they correspond to usual terms:

- A gives the update of the position given the velocities as

$$\begin{cases} \mathbf{q}(t+\delta t) = \mathbf{q}(t) + \mathbf{p}(t)\mathbf{M}^{-1}\delta t \\ \mathbf{p}(t+\delta t) = \mathbf{p}(t) \end{cases} \tag{4.27}$$

- B updates the vitesses uing Newton's second law

$$\begin{cases} \mathbf{q}(t+\delta t) = \mathbf{q}(t) \\ \mathbf{p}(t+\delta t) = \mathbf{p}(t) - \nabla U(\mathbf{q})\delta t \end{cases} \tag{4.28}$$

The final term O has the following solution (see [6]):

$$\begin{cases} \mathbf{q}(t+\delta t) = \mathbf{q}(t) \\ \mathbf{p}(t+\delta t) = \mathrm{e}^{-\gamma\delta t}\mathbf{p}(t) + \frac{\sigma}{\sqrt{2\gamma}}\sqrt{1-\mathrm{e}^{-2\gamma\delta t}}\mathbf{M}^{\frac{1}{2}}\mathbf{R}_{\delta t} \end{cases} \tag{4.29}$$

where $R_{\delta t}$ follows a normal distribution. Given the presence of a stochastic process, an "exact" solution means that the solution proposed here effectively yields the correct probability distribution.

Several integration schemes could be chosen here. The most simple – maybe naively so – would be to solve the A part first, with a time-step δt, then the B part with the same time-step, then the O part with the same time-step. This would define the "ABO" method.

Yet by splitting some of these elements in two, using half time-step integration, one can define the "BAOAB" method. If we note L_A, L_B and L_O the operators associated with each piece presented previously, the splitting (using Trotter theorem as presented earlier) gives:

$$e^{iL\delta t} = e^{iL_B\frac{\delta t}{2}} e^{iL_A\frac{\delta t}{2}} e^{iL_O\delta t} e^{iL_A\frac{\delta t}{2}} e^{iL_B\frac{\delta t}{2}} \tag{4.30}$$

This defines a new algorithm, reproduced hereunder.

Algorithm 4 : BAOAB (formally)

(**B**) $\vec{p}_i \leftarrow \vec{p}_i + \delta t/2 \times \vec{f}_i$
(**A**) $\vec{q}_i \leftarrow \vec{q}_i + \delta t \times \vec{p}_i/m_i$
(**O**) $\vec{p}_i \leftarrow \exp(-\gamma\delta t) \times \vec{p}_i + \sigma\sqrt{\frac{m_i}{2\gamma}(1-\exp(-2\gamma\delta t))}\vec{R}_{\delta t}$
(**A**) $\vec{q}_i \leftarrow \vec{q}_i + \delta t \times \vec{p}_i/m_i$
Compute forces using updated $\vec{q}_i$'s.
(**B**) $\vec{p}_i \leftarrow \vec{p}_i + \delta t/2 \times \vec{f}_i$

This integrator scheme allowed Matthews and Leimkuhler to use bigger time-steps in their simulation. Since this is precisely our objective, we will thus try a BAOAB implementation of our RESPA1 splitting. The algorithms that we built using both BAOAB and RESPA1 will be named "B-RESPA1" (as opposed to "V-RESPA1" for the Velocity-Verlet ones). Here is a typical such algorithm:

Algorithm 5 : B-RESPA1 – Using $\Delta t = n\delta t$ and $\Delta T = m\Delta t$

$\vec{p}_i \leftarrow \vec{p}_i + \Delta T/2 \times \vec{f}_{i,\text{SLOW}}$

for i = 1, m **do**

$\vec{p}_i \leftarrow \vec{p}_i + \Delta t/2 \times \vec{f}_{i,\text{INTERM}}$

for j = 1, n **do**

(**B**) $\vec{p}_i \leftarrow \vec{p}_i + \delta t/2 \times \vec{f}_{i,\text{FAST}}$

(**A**) $\vec{q}_i \leftarrow \vec{q}_i + \delta t \times \vec{p}_i/m_i$

(**O**) $\vec{p}_i \leftarrow \exp(-\gamma\delta t) \times \vec{p}_i + \sigma\sqrt{\frac{m_i}{2\gamma}(1-\exp(-2\gamma\delta t))}\vec{R}_{\delta t}$

(**A**) $\vec{q}_i \leftarrow \vec{q}_i + \delta t \times \vec{p}_i/m_i$

Compute FAST forces using updated $\vec{q}_i$'s.

(**B**) $\vec{p}_i \leftarrow \vec{p}_i + \delta t/2\times$

end for

$\vec{q}_i \leftarrow \vec{q}_i + \Delta t \times \vec{p}_i/m_i$

Compute INTERM forces using updated $\vec{q}_i$'s.

$\vec{p}_i \leftarrow \vec{p}_i + \Delta t/2 \times \vec{f}_{i,\text{INTERM}}$

end for

Compute SLOW forces using updated $\vec{q}_i$'s.

$\vec{p}_i \leftarrow \vec{p}_i + \Delta T/2 \times \vec{f}_{i,\text{SLOW}}$

This algorithm was tested using the same conditions, and we present the results in table 4.3 and 4.4, and in figure 4.3.

System	Prop.	Verlet 0.5fs	0.25/2/4	0.25/2.5/5	0.25/2/6
Water (S_3)	E_{pot}	-4459 ±39	-4454 ±45	-4436 ±43	-4415 ±45
	E_{pol}	-2169 ±41	-2147 ±40	-2125 ±40	-2126 ±40
Ubiquitin	E_{pot}	-27894 ±102	-27891 ±105	-27708 ±114	-27662 ±108
	E_{pol}	-13052 ±98	-12932 ±98	-12810 ±98	-12819 ±95
Speedup	(vs. ASPC)	–	1.72	2.43	2.53
	(vs. no ASPC)	–	2.5	3.5	3.7

Table 4.3: **V-RESPA1 integrators.** E_{pot} and E_{pol} designate potential and polarization energy respectively, both given in kcal/mol. Error computed with respect to the reference (Verlet 0.5 fs). Speedup are computed vs. a 1 fs Velocity-Verlet reference, both with and without ASPC.

Errors obtained on the energies are below the 2% bar, even for the 0.25/2.5/5 integrator who yielded incorrect polarization energies. This shows that the static properties are now back to a correct representation.

	BAOAB 1fs	0.25/2/4	0.25/2.5/5	0.25/2/6
D	1.82 ±0.01	1.84 ±0.02	1.92 ±0.03	1.96 ±0.01

Table 4.4: Diffusion coefficients obtained using BAOAB-RESPA1 integrators. D is the diffusion constant (expressed $\times 10^{-5}$ cm^2/s). Note that the reference is now a pure BAOAB one (no RESPA splitting).

As explained previously, the dynamics are modified since we are now in the Langevin framework, and more specifically given the use of a friction term. One should thus expect to have a change in the dynamical behaviour of the systems. This will reflect on the diffusion constants values, and it would make no sense to compare them to a Velocity-Verlet reference. As a consequence, we computed a new reference value, using the BAOAB scheme but no splitting (no RESPA or RESPA1) of the forces. Diffusion coefficients obtained with the various splits are then compared to this value, in order to stay consistent. As shown in table 4.4, the agreement is now much better (the biggest error is 8%).

This new reference is only necessary when trying to account for dynamic properties, as static and structural ones are not influenced by the movement of the particles. Hence the BAOAB 1 fs reference is required only here.

The same type of conclusions can be drawn from the radial distribution functions (fig. 4.3), where while V-RESPA1 integrators exhibited discrepancies, the reference curves are now much better reproduced.

These various results show that the implementation of a BAOAB scheme in association with the RESPA1 splitting allowed to solve the issues observed previously with the Verlet integrators. As claimed by Leimkuhler and Matthews, the dynamics are stabilized and we can aim for higher time-steps.

Note that the speedup are exactly identical to the ones obtained with Velocity-Verlet. This is due to the algorithms shapes: B-RESPA1 and V-RESPA1 are different when it comes to the treatment of *fast* parts only. The computations of the intermediate and slow terms, corresponding to the long range forces, is the most expensive part, and it is treated equally in both algorithms.

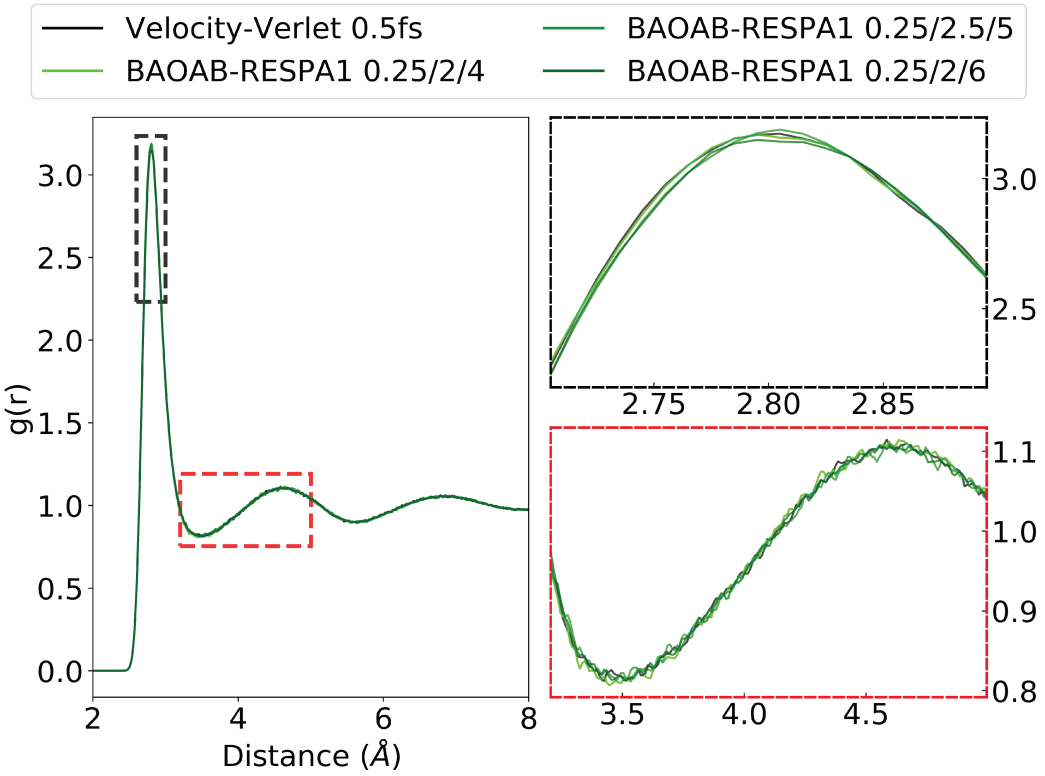


Figure 4.3: **B-RESPA1 integrators.** Oxygen-oxygen radial distribution function, comparerd to the Velocity-Verlet reference, using $\delta t = 0.5$ fs.

4.2.4 A closer look at polarization: TCG strikes back

Let us now have a closer look at the RESPA1 splitting, namely the separation between the short and long-range of the non-bonded forces. If we recall the shape of the energy terms when explicited within the SPME framework, they become a sum of three terms: E_{real} for direct space interactions , E_{recip} in reciprocal space, and a self-correction term (arising from the PME derivation) E_{self}. Within the RESPA1 framework, a supplementary separation based on the distance is added. It affects the real term such that

$$E_{\text{real}} = E_{\text{real, short}} + E_{\text{real, long}} \tag{4.31}$$

Recalling that we work with the AMOEBA force field, let us then gather all the involved terms, separating them depending on their typical times of evolution.

$$E_{\text{INTERM}} = E_{\text{elec, real, short}} + E_{\text{VdW, short}} + E_{\text{pol, real, short}} \tag{4.32}$$

$$E_{\text{SLOW}} = E_{\text{elec, real, long}} + E_{\text{elec, recip}} + E_{\text{elec, self}} + E_{\text{VdW, long}} + E_{\text{pol, real, long}} + E_{\text{pol, recip}} + E_{\text{pol, self}} \tag{4.33}$$

("elec", "VdW" and "pol" stand for electrostatic, Van der Waals and polarization, respectively).

Within Tinker-HP, Van der Waals terms are not treated using PME, but only with a cutoff. Therefore, they don't decompose in reciprocal and self-correction terms. Let us focus on the polarization terms here, coloured in the previous equations. We have

$$E_{\text{pol, total}} = E_{\text{pol, real, short}} + E_{\text{pol, real, long}} + E_{\text{pol, recip}} + E_{\text{pol, self}} \tag{4.34}$$

which can easily be rewritten as

$$E_{\text{pol, real, long}} + E_{\text{pol, recip}} + E_{\text{pol, self}} = E_{\text{pol, total}} - E_{\text{pol, real, short}} \tag{4.35}$$

The INTERM and SLOW terms can thus be changed into

$$E_{\text{INTERM}} = E_{\text{elec, real, short}} + E_{\text{VdW, short}} + E_{\text{pol, real, short}} \tag{4.36}$$

$$E_{\text{SLOW}} = E_{\text{elec, real, long}} + E_{\text{elec, recip}} + E_{\text{elec, self}} + E_{\text{VdW, long}} + \left[E_{\text{pol, total}} - E_{\text{pol, real, short}}\right] \tag{4.37}$$

This simple rewriting highlights that essentially two polarization terms are involved here: one for the short range of the real part, and another for the whole energy term without range separation. This means that one could chose to use *two polarization solvers*, one for each of these terms.

For the short-range real part $E_{\text{pol, real, short}}$, the ideal solver would have to be fast (as it is evaluated more often than the other), yet still exhibiting good properties, and in particular would ensure stability of the dynamics. It would be further accelerated thanks to the use of a polarization matrix $\mathbf{T}_{\mathbf{real,short}}$ taking into account the range cutoff.

On the other hand, the long-range real part $E_{\text{pol, real, long}}$ has to be computed precisely, as it encompasses all the missing terms. This means that computation will be more expensive than for the first solver. Fortunately, since this solver is only present in the "outer" time-step integration (it belongs to the SLOW terms), which is computed less frequently, the impact of a more expensive computation will hence be minimized.

We thus experimented using various solvers, and came up with the following pair.

- For the solution of the short-range real part, we used *TCG1*. It was developed precisely to compute polarization at a lower computational cost, while ensuring good energy conservation: it is an

ideal candidate. We used a preconditioner, but no guess or peek, as this allows for the fastest computations while still making use of the cheapest refinement we have.

- For the solution of the total polarization energy, we used a well converged *PCG*. As expected, it is slower, but is a good solution to retrieve the small deviation to the exact solution that TCG1 would incur.

In terms of implementation, we thus designed a short-range version of the TCG solvers to compute the $E_{\text{pol,real,short}}$ term, and the $E_{\text{pol,total}}$ term was computed as it used to be with no need for further developments. To sum up, our strategy is the following: using an approximate but fast solver for the high frequency terms, we hope to *recover the low frequency terms* by using a more precise solver at the outer time-teps.

Numerical results obtained using these "BAOAB-RESPA+TCG1" integrators are reported in tables 4.5, 4.6.

System		Verlet 0.5fs	0.25/2/4	0.25/2.5/5	0.25/2/6
Water (S_3)	E_{pot}	-4459 ±39	-4457 ±44	-4433 ±43	-4411 ±45
	E_{pol}	-2169 ±41	- 2146 ±40	-2124 ±39	-2119 ±40
Ubiquitin	E_{pot}	-27894 ±102	-27869 ±113	-27741 ±117	-27575 ±114
	E_{pol}	-13052 ±98	-12919 ±98	-12819 ±98	-12746 ±102
Speedup	(vs. ASPC)	–	2.32	2.7	2.9
	(vs. no ASPC)	–	3.4	3.9	4.2

Table 4.5: **B-RESPA1+TCG1 integrators.** E_{pot} and E_{pol} designate potential and polarization energy respectively, both given in kcal/mol. Error computed with respect to the reference (Verlet 0.5 fs). Speedup are computed vs. a 1 fs Velocity-Verlet reference, both with and without ASPC.

We can see that the energies are well conserved, as they remain beyond 3% of relative error. Given the way the polarization terms were splitted, this is not an obvious results. It is an *a posteriori* justification for the resplitting of the terms we wrote in equation 4.35, effectively ensuring that our strategy works.

Results shown on the diffusion constants (table 4.6) are also very satisfying, the error compared to a BAOAB 1 fs reference remain roughly the same as the ones obtained previously without this particular

	BAOAB 1fs	0.25/2/4	0.25/2.5/5	0.25/2/6
D	1.82 ±0.01	1.92 ±0.02	1.94 ±0.02	1.88 ±0.04

Table 4.6: Diffusion constants obtained with B-RESPA1+TCG/HMR integrators. D is the diffusion constant (expressed $\times 10^{-5}$ cm^2/s). Note that the reference is now a pure BAOAB one (no RESPA splitting).

treatment of the polarization.

Thanks to this rewriting of the polarization energy terms, we could make use of our previous developments regarding polarization solvers, and more precisely of the rapidity and robustness of the TCG1 solver. This also allowed effective speedups to progress and reach a new maximum (4.2 when comparing with a no-ASPC reference).

4.2.5 Hydrogen Mass Repartitioning: the final blow ?

So far, we have described several methods to improve the maximal time-step that one can use, whether it lies in the integration scheme itself, or in a refined treatment of the forces.

The bottleneck in this time-step race will always be the high frequency terms. Indeed, in order to correctly reproduce a phenomenon occurring at a very short time period, one needs to select a time-step several times smaller than this period. Looking at our classical molecular dynamics, it appears quite logically that these terms will involve hydrogen atoms, as they are (by far) the lightest. Procedures such as[7] or LINCS[8] work towards constraining these motions, but do not allow for time-steps bigger than 2 fs.[9]

In order to eliminate these motions, Feenstra *et al.*[9] proposed to redistribute the mass of heavy atoms onto the hydrogen ones, such that the high-frequency terms disappear from the dynamics[v]. This method is named *Hydrogen Mass Repartitioning* (HMR). This allows for much bigger time-steps (their 2 fs limit is pushed to a 7 fs one), thus accelerating considerably the simulations.

Notwithstanding, by redistributing masses, the systems dynamics are modified. While the authors claim that this has little effect on the motion of the main protein chains, they report bigger consequences when looking at the movement of water molecules. Diffusion (and subsequently viscosity) is

[v]Another version of the HMR method is implemented using *dummy* atoms in lieu of hydrogen ones, but we did not explore this possibility, because it is much more complicated to implement and yields results that are further away from the usual properties of the studied system.

found to be largely different.

One should therefore not expect to reproduce *every* property to a high level of accuracy, and in particular not the dynamic ones.

We implemented the HMR on an integrator using PCG for polarization solver, and on a second integrator using TCG1 as the solver for the short range real polarization as shown in previous section. Tables 4.7, 4.8 and figure 4.4 present the results obtained using these integrators.

System		Verlet 0.5fs	$1/\frac{10}{3}/10$	$1/\frac{10}{3}$(TCG1)/10(PCG)
Water (S_3)	E_{pot}	-4459 ±39	-4442 ±44	-4442 ±43
	E_{pol}	-2169 ±41	-2133 ±39	-2131 ±39
Ubiquitin	E_{pot}	-27894 ±102	-27611 ±114	-27523 ±118
	E_{pol}	-13052 ±98	-12837 ±97	-12792 ±98
Speedup	(vs. ASPC)	–	4	4.72 (4.91*)
	(vs. no ASPC)	–	5.8	6.8 (7.0*)

Table 4.7: **B-RESPA1(+TCG) integrators, using HMR.** E_{pot} and E_{pol} designate potential and polarization energy respectively, both given in kcal/mol. Error computed with respect to the reference (Verlet 0.5 fs). Speedup are computed vs. a 1 fs Velocity-Verlet reference, both with and without ASPC. *: numbers obtained using a PCG for the largest time-step, refined with an advanced preconditioner (Cholesky).

Energies obtained using this ultimate acceleration method are still in excellent agreement with the reference value, as attested in table 4.8. We can deduce that the modification on the dynamics still allow for proper ensemble averages, as configurational space seems to still be well explored.

Radial distribution functions, which can also be quite sensitive when looking at the peaks of the function, are also very close to the reference (Verlet) ones, confirming that the HMR addition has no dreadful impact on the static properties.

The diffusion constants, however, suffer from a significant drop. This is expected, as we are now using a method which modifies the dynamics by redistributing the atomic masses (we will discuss this in more details in the following paragraphs).

The speedup obtained here are the maximal values we will present in this work. The combination of all the previous technique, namely RESPA1 splitting, BAOAB integration scheme, TCG/PCG polarization treatment and HMR, yields speedups close to 5 *times* (7 if we compare to reference simulations using

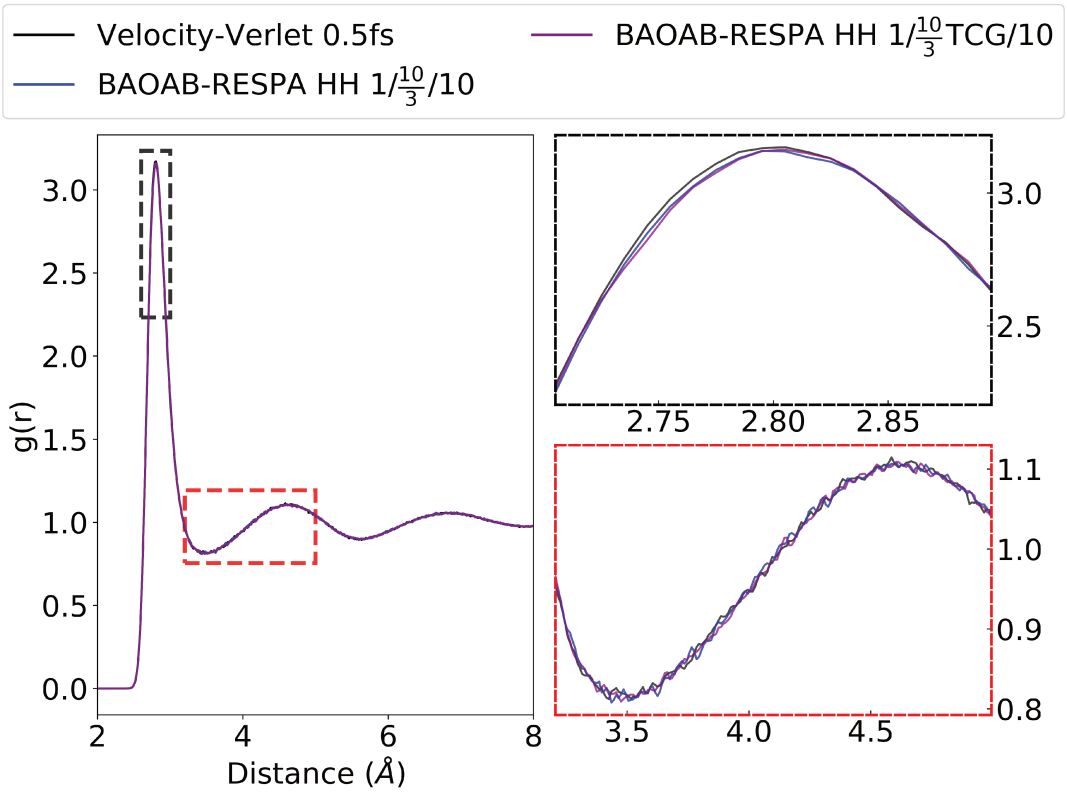


Figure 4.4: **B-RESPA1+TCG1 integrators.** Oxygen-oxygen radial distribution function, comparerd to the Velocity-Verlet reference, using $\delta t = 0.5$ fs.

	$1/\frac{10}{3}/10$	$1/\frac{10}{3}$(TCG1)/10(PCG)
D	1.6 ±0.01	1.6 ±0.02

Table 4.8: **Diffusion constant using B-RESPA1+HMR integrators**. D is the diffusion constant (expressed $\times 10^{-5}$ cm^2/s).

no ASPC).

One could however wonder how far it is advisable to go in disturbing the dynamical behaviour of the system. Naturally, if one wants to study diffusion coefficients, HMR should not be used. This is also true for the BAOAB integration and, more generally, the use of Langevin dynamics. As we saw in previous section, this already has a substantial effect.

To broaden the discussion, Berendsen[9] as well as Morrone[10] pointed out that the diffusion coefficient (or inversely, the viscosity) would have an incidence on the sampling efficiency of a method.

Indeed, if one wants to sample the configurational space of a very viscous system, the molecules

will be moving at a very low pace, undergoing strong friction. It will thus take a considerably longer time to correctly explore all possible configurations with consistent probabilities. This is the risk behind modifying the dynamics used in our simulation, and the last results obtained should be taken cautiously.

A middle ground between the simulation acceleration and the degradation of the dynamics quality has to be found there. The maximum decrease of the diffusion constant observed with the $1/\frac{10}{3}$(TCG)/10(PCG) remains quite acceptable if we compare it to other integration methods aiming at large time-step integration. Albaugh *et al.*[11] indeed proposed a very efficient integration combining extended Lagrangian and large time-step integration, but reported diffusion constants off by a factor 5.

Assessment on free energy computations

As we did for the Truncated Conjugate Gradient, we close this study on the integrators by assessing their capability to produce free energies. We tested the fastest methods that we derived in this chapter (BAOAB-RESPA1 using TCG and HMR). We have seen how sensitive this kind of computation could be, and will thus use it as an ultimate test for our integrators. The computation method is the same as the one presented in chap. 3 (see section 3.2.2), relying on successive thermodynamical windows that progressively activate Van der Waals, then electrostatic interactions.

	Ref.	$1/\frac{10}{3}$/10-HMR	$1/\frac{10}{3}$(TCG1)/10-HMR
ΔA_{hydr} Na^+ (kcal/mol)	-91.5 ($\pm$0.13)	-91.5 ($\pm$0.13)	-91.4 ($\pm$0.13)

Table 4.9: Hydration free energies for the Na^+ cation. "Ref" designates the reference integrator, namely a 0.5 fs Verlet. HMR denotes the use of Hydrogen Mass Repartitioning.

The results displayed in table 4.9 demonstrate the excellent adaptability of our newly designed integrators with the highest speedup gains. The complexity and sensitivity usually to be worried about when computing free energies demonstrates the stability and applicability of these integrators, even-though these final versions alter the dynamics.

Note that the non-refined ("naked") TCG1 was used here, *i.e.* the fastest TCG and "roughest" version. The objective was to stay as simple as possible to create a proof of principle. Any other setup could nevertheless be chosen here, and this represents another direction in which we could expand our tests.

Last but not least, we limited the range of our time-step increase to values that were preserving our

computed properties. One could however choose to extend this range at the price of a certain accuracy on the properties – essentially, this is once again a tradeoff between computational speed and accuracy, and different applications may have different ideal balance between these two.

Through an iterative search, we assembled several pieces of the MD integration's arsenal, as well as our own TCG algorithm, in order to build the most efficient integrators possible. Speedup was increased through forces splitting, via RESPA then RESPA1; stabilization of the dynamics was obtained thanks to the BAOAB integration scheme; through the use of TCG and careful consideration of the polarization terms; and finally via the Hydrogen Mass Repartitioning technique.

This work was published as a letter in the Journal of Physical Chemistry Letters, and is reproduced hereafter.

THE JOURNAL OF PHYSICAL CHEMISTRY Letters

Letter

Cite This: J. Phys. Chem. Lett. 2019, 10, 2593−2599

pubs.acs.org/JPCL

Pushing the Limits of Multiple-Time-Step Strategies for Polarizable Point Dipole Molecular Dynamics

Louis Lagardère,*[,†,‡] Félix Aviat,[¶,‡] and Jean-Philip Piquemal*[,¶,§,⊥]

[†]Institut Parisien de Chimie Physique et Theorique, Sorbonne Université, FR2622 CNRS, F-75005 Paris, France

[‡]Institut des Sciences du Calcul et des Données, Sorbonne Université, F-75005 Paris, France

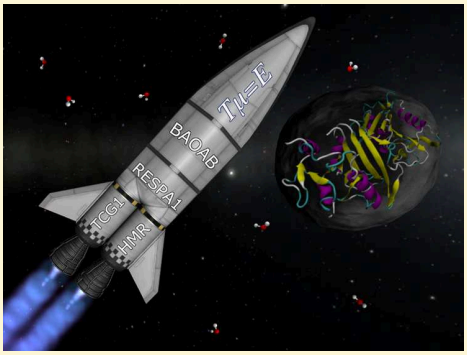
[¶]Laboratoire de Chimie Théorique, Sorbonne Université, UMR 7616 CNRS, F-75005 Paris, France

[§]Institut Universitaire de France, F-75005 Paris, France

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

Supporting Information

ABSTRACT: We propose an incremental construction of multi-time-step integrators to accelerate polarizable point dipole molecular dynamics while preserving sampling efficiency. We start by building integrators using frequency-driven splittings of energy terms and a Velocity-Verlet evaluation of the most rapidly varying forces and compare a standard bonded/nonbonded split to a three-group split dividing nonbonded forces (including polarization) into short- and long-range contributions. We then introduce new approaches by coupling these splittings to Langevin dynamics and to Leimkuhler's BAOAB integrator in order to reach larger time steps (6 fs) for long-range forces. We further increase sampling efficiency by (i) accelerating the polarization evaluation using a fast/noniterative truncated conjugate gradient (TCG-1) as a short-range solver and (ii) pushing the outer time step to 10 fs using hydrogen mass repartitioning. The new BAOAB-RESPA1 integrators demonstrate up to a 7-fold acceleration over standard 1 fs (Tinker-HP) integration and reduce the performance gap between polarizable and classical force fields while preserving static and dynamical properties.



The most straightforward way to speed up molecular dynamics (MD)[1,2] is to use larger time steps. In this context, multi-time-step schemes emerged,[3] but the largest usable time step is limited by resonance effects.[4,5] As pointed out by various authors, it is possible to overcome these effects by using modified dynamics that still sample the correct measure, but these solutions alter the dynamical properties (generalized Langevin equation (GLE);[6] stochastic isokinetic extended phase-space algorithm[7−9]). However, in practice, one would like to accelerate MD while also preserving the dynamics.[6,10] This Letter addresses this problem in the particular context of polarizable force fields (PFFs).[11,12] This class of methods is more computationally expensive than classical force fields (FFs) because of the need to evaluate a many-body polarizable energy.[13,14] Multi-time-stepping is therefore essential. The general consensus to ensure conserved properties is to limit the use of reversible Reference System Propagator Algorithm (**RESPA**) integrators[3] to a bonded/nonbonded force split and to use a 2 fs time step for the nonbonded forces. Further splitting of the nonbonded forces is not straightforward because of the many-body nature of polarization but has been shown to be applicable.[8,15] Indeed, one can define a short-range polarization energy and evaluate, at an outer time step, the slowly varying difference between the actual polarization energy (and forces) and the short-range ones. More precisely, one has to evaluate both the short-range and total polarization terms at these outer time steps. The reduced computational cost of the short-range polarization contribution and the less frequent evaluation of the total one effectively reduce the computational effort. Because the upper limits of these strategies have not yet been evaluated by the community, we will, in this Letter, assess this frontier to improve simulation performances while respecting two important constraints: (i) the mandatory need to preserve static and dynamical properties and (ii) the possibility of a black-box implementation allowing for strong computational speedups without dependence to the studied system. In everything that follows, tests have been made using the AMOEBA PFF[16] and the Tinker-HP software.[17] Technical details as well as various algorithmic setups are provided in the Supporting Information (see section S1). A summary of our incremental strategy is depicted in Figure 1. Interested developers can also look at the code that will be available on the Tinker-HP Web site[18] and later on Github.[19]

A Popular Integrator: V-RESPA. Let us first evaluate the limits of the bonded/nonbonded RESPA integrators for which

Received: March 29, 2019
Accepted: May 3, 2019
Published: May 3, 2019

DOI: 10.1021/acs.jpclett.9b00901
J. Phys. Chem. Lett. 2019, 10, 2593−2599

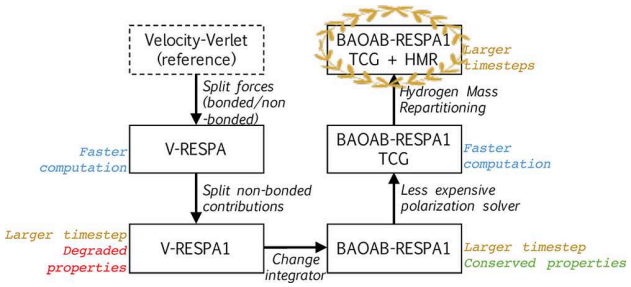


Figure 1. "V-" (or "BAOAB-") indicates that the numerical integration scheme is Velocity-Verlet[20] (or BAOAB[21,22]). "RESPA" and "RESPA1" respectively mean the RESPA single-split (bonded vs. nonbonded) strategy[3] and the RESPA1 double-split (bonded, short-range nonbonded, long-range nonbonded) one.[23] "TCG" is the acronym for truncated conjugate gradient, a fixed-cost noniterative polarization solver.[24] "HMR" stands for hydrogen mass repartitioning,[10] implemented to avoid high-frequency motions.

all of the bonded terms are evaluated within a Velocity-Verlet[20] loop (denoted as **V-RESPA** in the rest of the text) every 0.25 fs and for which the nonbonded terms (van der Waals, electrostatics, and polarization) are evaluated first at 2 and then at 3 fs. To assess the accuracy of the associated integrators, we ran simulations of 2 ns in the *NVT* ensemble at 300 K with two test systems: a cubic box of 500 water molecules, with a 24.66 Å edge, and a 9737 atoms box with edges of 54.99 × 41.91 × 41.91 Å containing a solvated protein (the ubiquitin). In both cases, periodic boundary conditions for electrostatics and polarization were evaluated with Smooth Particle Mesh Ewald (SPME)[17,25,26] with standard parameters (see the SI) as we chose a preconditioned conjugate gradient (PCG) polarization solver using a diagonal preconditioner and a 10^{-5} convergence threshold.[13,14] For each of these systems and for each integrator, we computed various static observables: average potential energy, average polarization energy, and for the bulk water system, radial distribution functions. In this last case, we also computed the self-diffusion coefficient, a dynamical property evaluated with the Einstein formula by averaging over a series of time origins.[27] The self-diffusion coefficient is known to have a size dependency vanishing at the infinite size limit,[27] but here, these values are only used as a means of comparison between integrators; hence, these corrections were not applied. These tests were performed in the canonical ensemble for which the choice of the thermostat impacts the dynamics of the system. We ran these tests using the global velocity rescaling thermostat developed by Bussi et al. with a relaxation constant of 0.2 ps, for which the dynamical properties are close to the one obtained with pure Hamiltonian dynamics.[28] These values have been compared to the ones obtained with a Velocity-Verlet integrator used at a 0.5 fs time step, which can be

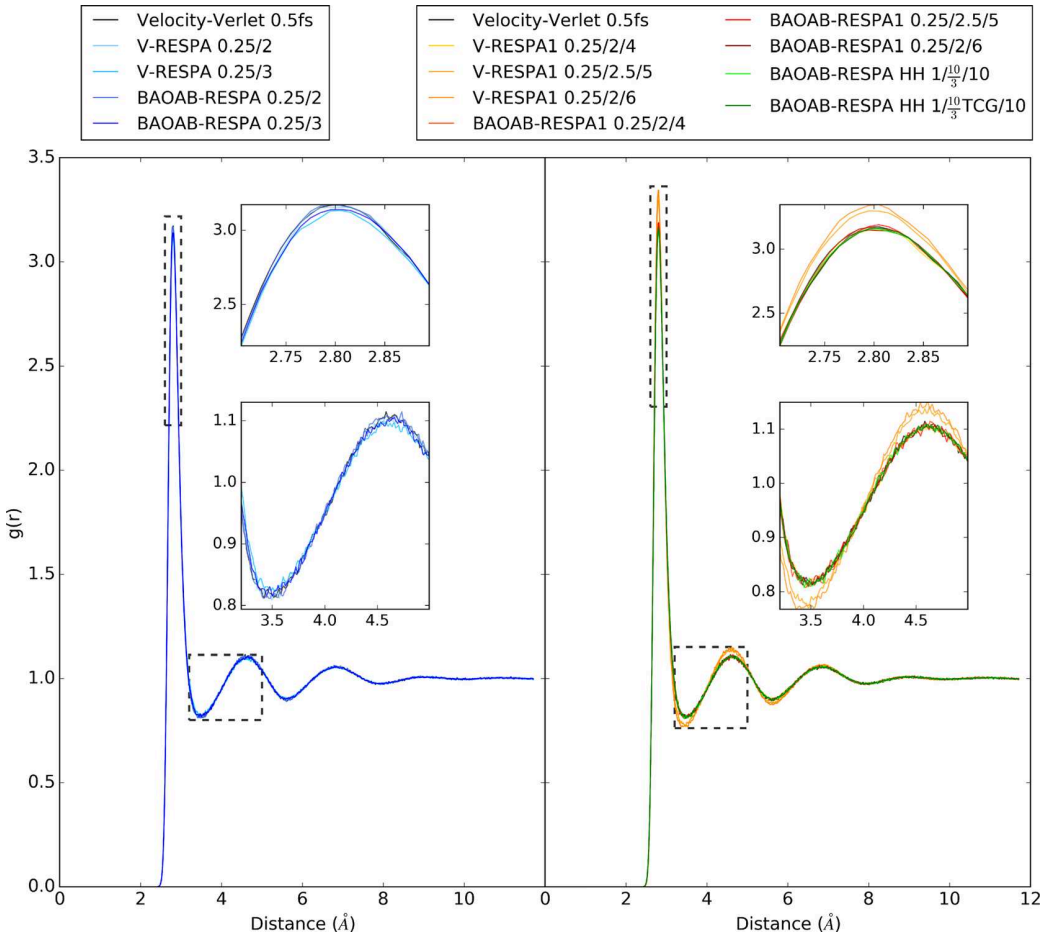


Figure 2. Oxygen−oxygen radial distribution function for various integrators (see the text for notation). Radial distributions appear correct with most of the setups. However, degraded results are obtained with the V-RESPA1 integrators using large outer time steps beyond 4 fs.

DOI: 10.1021/acs.jpclett.9b00901
J. Phys. Chem. Lett. 2019, 10, 2593−2599

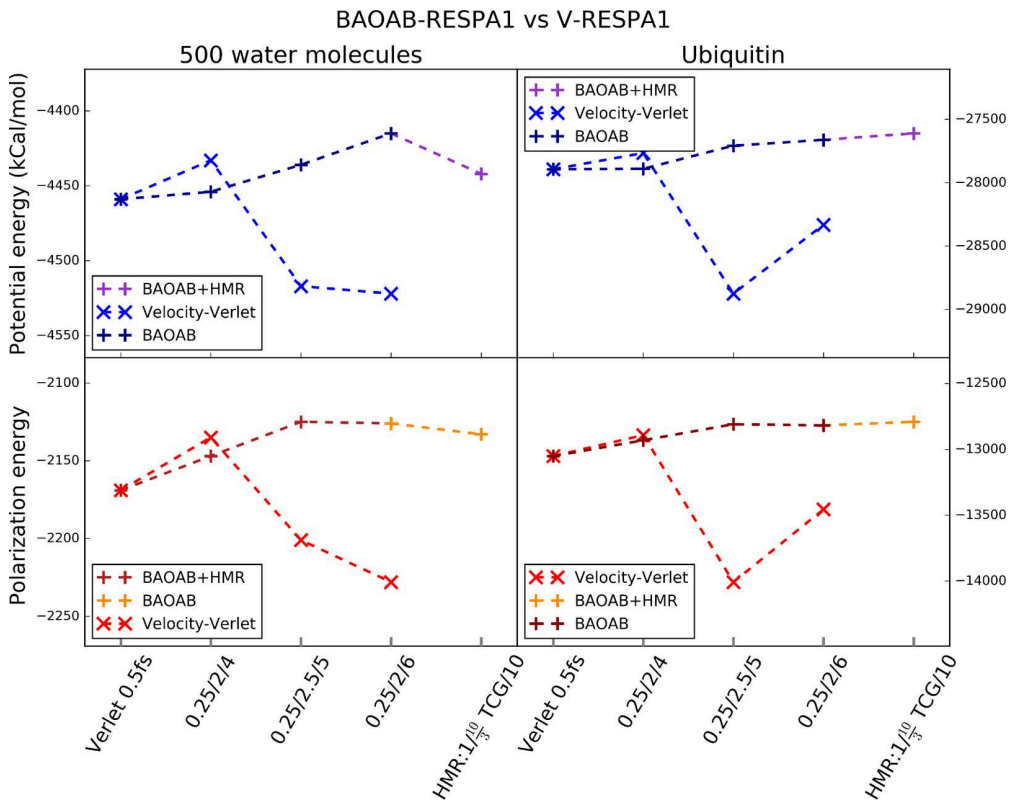


Figure 3. Average potential and polarization energies (in kcal/mol) for a 500 molecule water box and solvated ubiquitin computed using various integrators.

considered as a reference. In the rest of the text, we will be denoting the different time step lengths as a/b, a being the bonded terms one and b the nonbonded terms one (both in fs).

For both systems, the V-RESPA integrator where nonbonded forces are evaluated at 2 fs gives similar results as the reference (within statistical uncertainty), with a difference of less than 2% for average energies (see Tables 1−3 of the SI). With an outer time step of 3 fs, the error on the total potential energy is still satisfactory (around 1%), but the error on the polarization energy grows significantly (more than 2.5%). This advocates for careful use of this setup.

Concerning O−O radial distribution function for water, no significant differences with the reference Velocity-Verlet (0.5 fs) ones can be observed among these different methods (see Figure 2).

The self-diffusion coefficients of water are also nicely preserved for the V-RESPA integrator with a 2 fs outer time step, though it is slightly off (around 8%) with a 3 fs outer time step (see Table 1, SI).

Further Range Separation in Polarizable Force Fields: V-RESPA1. As a second part, we will now evaluate the limits of other RESPA integrators, for which the nonbonded terms are further split in two parts, the short- and long-range. We are now considering three terms: the bonded, the nonbonded short-range, and the nonbonded long-range terms. Regarding the split of the electrostatics and the polarization energies, we chose to use the **RESPA1** logic,[23] where the short-range part of the electrostatic (and polarization) energy is defined as the short-range part of the real space SPME energy. In this case, it has been shown that the stability of the integrator is less dependent on the smoothing parameters used to switch between short- and long-range.[29] Details of the definition of these short- and long-range forces as well as these smoothing parameters can be found in the SI. We test various setups within this context: the bonded forces are always evaluated every 0.25 fs, but the short-range nonbonded ones are evaluated either every 2 or 2.5 fs, and the time step of the long-range forces (that has to be a multiple of the previous one) is either 4, 5, or 6 fs. In the rest of the text, these integrators will be denoted as **V-RESPA1**. We will be denoting the different time step lengths of the integrators as $a/b/c$, a being the bonded term time step length, b the short-range nonbonded, and c the long-range ones (all in fs).

For the bulk water system (see Table 4 in the SI), we observe that both the average potential and polarization energies are preserved within 2% of the reference value for the 0.25/2/4 and the 0.25/2.5/5 setups but that the average polarization energy is more than 2% off for the 0.25/2/6 setup. Concerning the radial distribution functions of water, it is clear that only the 0.25/2/4 integrator gives satisfactory results as other choices diverge from the reference, as can be seen in Figure 2. Furthermore, if the self-diffusion coefficient is stable for the 0.25/2/4 integrator (see Table 5, SI), it exhibits a dramatic decrease for the other ones (falling at 1.34 instead of 2.08 for the 0.25/2/6 setup). This shows not only that the dynamical properties are not well preserved with these setups but also that the computational gains expected due to the use of a larger time step are counterbalanced by a lower sampling rate.[6,10]

Indeed, as pointed out by Berendsen,[6,10] such a decrease in the self-diffusion coefficient is expected to reduce the sampling

DOI: 10.1021/acs.jpclett.9b00901
J. Phys. Chem. Lett. 2019, 10, 2593−2599

efficiency by a similar amount because it is associated with an increase of water viscosity and thus a slowing down of large-scale motions. For the solvated ubiquitin, it is also clear that only the 0.25/2/4 setup corresponds to satisfactory accuracy as the other ones give average potential and polarization energies off by more than 3% (see Table 6, SI).

Recovering Accuracy through Langevin Dynamics: The New BAOAB-RESPA1 Integrator for Polarizable Force Fields. Thirdly, another way to sample the canonical ensemble is Langevin dynamics, where coupling to a heat bath is made through additional local friction and dissipative stochastic terms. The dynamics is then known to be altered compared to the pure Hamiltonian one, but this impact is expected to be small for a relatively small friction constant (less than 1 ps^{-1}). In this context, Leimkuhler and collaborators proposed an integrator for Langevin dynamics based on operator splitting and a particular ordering of the terms of the equations of motion named **BAOAB**.[21,22] They showed in various contexts[21] that this integrator has improved accuracy for configurational sampling compared to other ones. We thus also tested the previously presented splittings using this new integrator, with a 1 ps^{-1} friction constant (they will be denoted **BAOAB-RESPA** and **BAOAB-RESPA1** in the rest of the text ; see the SI for additional description of these integrators), and noted a significant improvement in terms of accuracy (reported in Tables 7−9 of the SI).

Indeed, for the bulk water system (Table 7 of the SI), the errors for the BAOAB-RESPA integrator with a bonded/nonbonded split and a 3 fs outer time step are limited to less than 1% for the average total potential energy and 2% for the average polarization energy, in both cases staying within statistical error, compared to 1.1 and 2.8% for the Velocity-Verlet-based integrator. Figure 2 also shows improved agreement with the reference for the water radial distribution compared to RESPA. The same behavior can be observed on the solvated ubiquitin, for which both of these values stay respectively below 1 and 2%, within statistical error (see Table 9, SI). Because the dynamics is modified when running *NVT* trajectories with Langevin, even if the differences are expected to be small for a small friction, comparing values of the self-diffusion coefficients obtained with these integrators only makes sense by taking as a reference a numerical scheme integrating Langevin dynamics with conservative parameters. This is why we chose, as a reference for these values, the ones obtained with a 1 fs BAOAB integrator and 1 ps^{-1} friction, which as expected gives self-diffusion close to the reference Velocity-Verlet one (1.82 versus 2.08). For the BAOAB-RESPA integrators, we see that errors in the self-diffusion coefficients (see Table 8, SI) are limited to 6% with a 3 fs outer time step compared to 8% with a similar time step and a Velocity-Verlet inner loop. The better performances of BAOAB-based integrators with respect to the Velocity-Verlet ones becomes obvious within a RESPA1 split. Indeed, we computed the same observables for the BAOAB-RESPA1 integrators (see Tables 10−12, SI, and Figures 3 and 4), equivalent to the V-RESPA1 integrators, and we see that the average potential and polarization energies are strikingly more stable and always within 2% error with respect to their reference value, that is to say, within statistical uncertainty. Similar comments can be made for the radial distribution functions: unlike the Velocity-Verlet integrators, they almost perfectly overlap with their references even with a 5 or 6 fs outer time step. When using the Velocity-Verlet-based

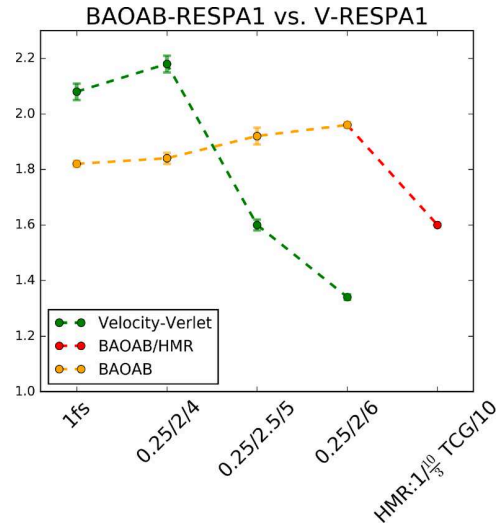


Figure 4. Self-diffusion (D) coefficient for various integrators (in 10^{-5} cm^2 s^{-1}).

RESPA1 integrators with an outer time step larger than the most conservative (4 fs) one, the diffusion coefficient showed a dramatic decrease (see Figure 3 and Table 5 in the SI). Yet, in the BAOAB-RESPA1 case, this dynamical observable remains far more stable for all integrator setups: even for the choice of evaluating long-range nonbonded forces every 6 fs, the error is less than 8%, whereas it exceeds 35% in the equivalent V-RESPA1 setup. This highlights again that BAOAB-RESPA1 integrators are not only more accurate but also ensure a conserved sampling rate, which is not the case for the V-RESPA1 ones.

Concerning the effective speedups in our implementation, Table 1 displays the gains obtained for the BAOAB-RESPA and the BAOAB-RESPA1 integrators (compared to a regular 1 fs Velocity-Verlet). They are the same as the one obtained for the V-RESPA and the V-RESPA1 integrators. We show two entries in Table 1: one where a guess based on Kolafa's Always Stable Predictor Corrector[30] (ASPC) is used for the induced dipoles (standard Tinker-HP setting), but only at short-range for the RESPA1 integrators, and one where the "direct field" guess is used,[13,14] showing that up to a 2.53 speedup (3.7 without ASPC) is achieved. Note that for the RESPA1 schemes, additional gains are made in the long-range polarization solvers by using, at the same time step, the short-range dipoles obtained as a guess for the long-range ones, effectively reducing the number of iterations required to converge. For the BAOAB-based integrators, the benefits of using the RESPA1 splitting are clearly demonstrated as the 0.25/2.5/5 and 0.25/2/6 frameworks are both faster than the 0.25/3 RESPA integrator for a similar accuracy.

Speeding Up BAOAB-RESPA1: TCG-1 Solver for the Short-Range Polarization and Hydrogen Mass Repartioning. When using a RESPA1 multi-time-step integrator and a PFF, the sole purpose of the short-range polarization energy is to eliminate the high-frequency part of the total polarization energy.[8,31] This is why an approximate but less computationally expensive and nonparametric expression of the polarization energy can be used to fill this role and provide an additional speedup. In that context, we decided to use the recently introduced truncated conjugate gradient (**TCG**)[24,32] as a short-range solver. TCG can be chosen to be minimal in cost (TCG-1 with

DOI: 10.1021/acs.jpclett.9b00901
J. Phys. Chem. Lett. 2019, 10, 2593−2599

Table 1. Speedup of BAOAB-RESPA and BAOAB-RESPA1 Integrators Calculated with Respect to the Velocity-Verlet Integrator at 1 fs[a]

splits	0.25/2	0.25/3	0.25/2/4	0.25/2.5/5	0.25/2/6	0.25/2(TCG)/4	0.25/2.5(TCG)/5	0.25/2(TCG)/6	$1/\frac{10}{3}/10$	$1/\frac{10}{3}$(TCG)/10
ASPC	1.75	2.37	1.72	2.43	2.53	2.32	2.7	2.9	4	4.72 (4.91*)
No ASPC	2.53	3.42	2.5	3.5	3.7	3.4	3.9	4.2	5.8	6.8 (7.0*)
RESPA-type	R	R	R1	R1	R1	R1	R1	R1	R1(HMR)	R1(HMR)

[a]The types of RESPA integrators are defined by R1 = RESPA1 and R = RESPA. Speedups obtained with V-RESPA and V-RESPA1 integrators are identical. * = replacement of the PCG diagonal preconditioner by an improved technique;[33,34] see the text.

a diagonal preconditioner but without any guess and without a peak step) to be the fastest possible. This coupling provides an additional computational gain at a conserved accuracy (see Tables 13−15, SI), corresponding to a final speedup of more than 4 times compared to a regular MD of 1 fs with a Velocity-Verlet integrator (see Table 1). Such a polarization setup offers full energy conservation, and the static and dynamical properties are marginally affected by this choice.[13,14,24]

At this point, we reached the performance limits if one wants to preserve tight accuracy on the dynamics. One of the most natural ways to further increase the size of the usable time step when simulating a large biological system is to redistribute the mass of the heavy atoms on the hydrogens that they are carrying (a method named hydrogen mass repartitioning (**HMR**)[10]), thus limiting the high-frequency stretching motions of these atoms while keeping the same configurational potential energy surface. In the following, we show that this redistribution allows one to use even larger time steps while maintaining satisfactory accuracy with a BAOAB-RESPA1 integrator and the same TCG-1 short-range polarization solver as before (Tables 16−18, SI). As can be seen in Table 1, the approach appears to be a very good compromise: large speedups can be obtained by pushing the bonded force time step to 1 fs, the short-range nonbonded forces time step up to $\frac{10}{3}$ fs, and the outer one up to 10 fs. Details on how the mass repartitioning is done can be found in SI (section S1). A resulting acceleration of 4.72 (6.8 without ASPC) is obtained, keeping the errors on the average energies below 2%, maintaining an accurate evaluation of radial distribution functions and a good enough evaluation of the self-diffusion coefficient so that sampling efficiency is preserved. Because PCG, as a Krylov method, is systematically improvable,[13] additional small speedups can be obtained by focusing on the long-range PCG solver performances. For example, besides using a diagonal preconditioner, one could use more advanced techniques such as those proposed by Skeel[33] or by Beran.[34] Improved performances of 3−4% are observed, reaching a global acceleration of more than 7-fold, with the same accuracy as a 1 fs Velocity-Verlet scheme without ASPC. Finally, beside the net acceleration, another advantage of the TCG use lies in the absence of use of a dipole history (as in predictor-correctors such as ASPC), leading to a method free of time-reversibilty and volume preservation issues.[24] Finally, the small decrease (8%) of the self-diffusion constant (see Figure 4) observed in the most aggressive setup has to be compared with actual available large step methods[9] that, despite their qualities, are not able to maintain accuracy on dynamical properties, providing diffusion constants reduced by a factor of 5.[9] Our approach does not suffer from these problems: it remains operational, maintaining sampling efficiency.

To illustrate the robustness of these approaches, we performed several tests taking advantage of our massively parallel AMOEBA production implementation in Tinker-HP.[17] First, we checked the stability of the dynamics using the fastest available setup. We provide a 15 ns simulation of ubiquitin (see SI, section S3): the potential and polarization energies normally fluctuate around their mean values, demonstrating the stability of the approach. Furthermore, we computed the average molecular dipole moments for the bulk water systems and confirmed their full stability (see SI, section S4). Second, we ran simulations on large systems of biological interest, namely, the solvated dihydrofolate reductase protein (DHFR, 23358 atoms) and the solvated Satellite Tobacco Virus (STMV, 1066628 atoms). The discussed speedup of 7-fold (vs a 1 fs/Velocity-Verlet/PCG-10^{-5}) is conserved as we obtained a production of 22.2 ns/day on 680 cores for DHFR and 1.2 ns/day on 10800 cores for STMV. Such results are of major interest, as a 7-fold acceleration will enable one to save millions of hours of computing time while enabling long and accurate polarizable molecular dynamics studies on large systems. Finally, we computed a more involved property that is of key importance in biological simulations: hydration free energies. We applied the Bennett acceptance ratio method,[35] a commonly used approach to compute free energy differences[36] to evaluate the solvation free energy of a sodium cation in water. Results are shown in Table 2, and practical details on

Table 2. Hydratation Free Energies for the Na^+ Cation

	Velocity-Verlet 0.5 fs	$1/\frac{10}{3}/10$-HMR	$1/\frac{10}{3}$(TCG1)/10-HMR
ΔG_{hydrat} Na^+ (kcal/mol)	89.7(±0.13)	89.7(±0.13)	89.6(±0.13)

the choice of alchemical free energy difference windows can be found in the SI (section S1). Even for the fastest setup, the values obtained are within 0.1 kcal/mol of the 89.7 kcal/mol for the reference,[37] demonstrating the validity of these acceleration schemes and their capability to preserve accuracy.

To conclude, after examining the limits of a standard Velocity-Verlet integrator for PFFs used in combination with a RESPA1 split, we introduced new BAOAB-RESPA1 Langevin integrators coupled to a fast short-range noniterative TCG-1 polarization solver and HMR, achieving all together large computational speedups. Two optimal BAOAB-RESPA1 setups were presented and compared to a 1 fs Velocity-Verlet reference: (i) one (namely, 0.25/2(TCG)/6) for which all properties are preserved while providing a global speedup of more than 4-fold and (ii) a second ($1/\frac{10}{3}$(TCG)/10+HMR) for which dynamical properties are slightly affected but where sampling remains efficient, offering a strong acceleration up to 7-fold. As accuracy is maintained and sampling efficiency is preserved while being system-independent, the proposed methodology can be used as a black-box in our Tinker-HP framework, benefiting from its massive parallel implementation

DOI: 10.1021/acs.jpclett.9b00901
J. Phys. Chem. Lett. 2019, 10, 2593−2599

and offering therefore further computational gains.[17] Such findings are game-changing as they extend the applicability of polarizable MD to longer-time-scale simulations and larger systems. In practice, the resulting performance gain helps reduce the computational gap between point dipole PFFs such as AMOEBA and more tractable models such as Drude[38] or even nonpolarizable force fields such as CHARMM[39] or AMBER.[40]

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acs.jpclett.9b00901.

Methods and additional results, including average potential and polarization energies and self-diffusion coefficients (PDF)

■ AUTHOR INFORMATION

Corresponding Authors

*E-mail: louis.lagardere@sorbonne-universite.fr.

*E-mail: jean-philip.piquemal@sorbonne-universite.fr.

ORCID

Jean-Philip Piquemal: 0000-0001-6615-9426

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

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. We thank GENCI for computer time within the Grand Challenge on the Irène Joliot-Curie (TGCC, Bruyères le Châtel, France).

■ REFERENCES

(1) Allen, M. P.; Tildesley, D. J. *Computer Simulation of Liquids*; Clarendon Press: New York, 1989.

(2) McCammon, J. A.; Harvey, S. C. *Dynamics of Proteins and Nucleic Acids*; Cambridge University Press, 1987.

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

(4) Biesiadecki, J. J.; Skeel, R. D. Dangers of Multiple Time Step Methods. *J. Comput. Phys.* **1993**, *109*, 318−328.

(5) Ma, Q.; Izaguirre, J.; Skeel, R. Verlet-I/R-RESPA/Impulse is Limited by Nonlinear Instabilities. *SIAM Journal on Scientific Computing* **2003**, *24*, 1951−1973.

(6) Morrone, J. A.; Markland, T. E.; Ceriotti, M.; Berne, B. J. Efficient multiple time scale molecular dynamics: Using colored noise thermostats to stabilize resonances. *J. Chem. Phys.* **2011**, *134*, No. 014103.

(7) Leimkuhler, B.; Margul, D. T.; Tuckerman, M. E. Stochastic, resonance-free multiple time-step algorithm for molecular dynamics with very large time steps. *Mol. Phys.* **2013**, *111*, 3579−3594.

(8) Margul, D. T.; Tuckerman, M. E. A Stochastic, Resonance-Free Multiple Time-Step Algorithm for Polarizable Models That Permits Very Large Time Steps. *J. Chem. Theory Comput.* **2016**, *12*, 2170−2180.

(9) Albaugh, A.; Tuckerman, M. E.; Head-Gordon, T. Combining Iteration-Free Polar-ization with Large Time Step Stochastic-Isokinetic Integration. *J. Chem. Theory Comput.* **2019**, *15*, 2195−2205.

(10) Feenstra, K. A.; Hess, B.; Berendsen, H. J. C. Improving efficiency of large time-scale molecular dynamics simulations of hydrogen-rich systems. *J. Comput. Chem.* **1999**, *20*, 786−798.

(11) Shi, Y.; Ren, P.; Schnieders, M.; Piquemal, J.-P. *Reviews in Computational Chemistry*; John Wiley and Sons, Inc, 2015; Vol. *28*; pp 51−86.

(12) Gresh, N.; Cisneros, G. A.; Darden, T. A.; Piquemal, J.-P. Anisotropic, Polarizable Molecular Mechanics Studies of Inter- and Intramolecular Interactions and Ligand Macromolecule Complexes. A Bottom-Up Strategy. *J. Chem. Theory Comput.* **2007**, *3*, 1960−1986.

(13) 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.

(14) 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.

(15) Masella, M. The multiple time step r-RESPA procedure and polarizable potentials based on induced dipole moments. *Mol. Phys.* **2006**, *104*, 415−428.

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

(17) 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.

(18) *Tinker-HP's Website*. http://tinker-hp.ip2ct.upmc.fr/ (Accessed Apr 24, 2019).

(19) *Tinker-HP's Github*. https://github.com/TinkerTools/Tinker-HP (Accessed Apr 24, 2019).

(20) Swope, W. C.; Andersen, H. C.; Berens, P. H.; Wilson, K. R. A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *J. Chem. Phys.* **1982**, *76*, 637−649.

(21) Leimkuhler, B.; Matthews, C. Rational Construction of Stochastic Numerical Methods for Molecular Sampling. *Applied Mathematics Research eXpress* **2012**, *2013*, 34−56.

(22) Leimkuhler, B.; Matthews, C. Robust and efficient configurational molecular sampling via Langevin dynamics. *J. Chem. Phys.* **2013**, *138*, 174102.

(23) Zhou, R.; Harder, E.; Xu, H.; Berne, B. J. Efficient multiple time step method for use with Ewald and particle mesh Ewald for large biomolecular systems. *J. Chem. Phys.* **2001**, *115*, 2348−2358.

(24) 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.

(25) Essmann, U.; Perera, L.; Berkowitz, M. L.; Darden, T.; Lee, H.; Pedersen, L. G. A smooth particle mesh Ewald method. *J. Chem. Phys.* **1995**, *103*, 8577−8593.

(26) Stamm, B.; Lagardère, L.; Polack, E.; Maday, Y.; Piquemal, J.-P. A coherent derivation of the Ewald summation for arbitrary orders of multipoles: The self-terms. *J. Chem. Phys.* **2018**, *149*, 124103.

(27) Maginn, E. J.; Messerly, R. A.; Carlson, D. J.; Roe, D. R.; Elliott, J. R. Best Practices for Computing Transport Properties 1. Self-Diffusivity and Viscosity from Equilibrium Molecular Dynamics [Article v1.0]. *LiveCoMS* **2019**, *1*, 6324.

(28) Bussi, G.; Donadio, D.; Parrinello, M. Canonical sampling through velocity rescaling. *J. Chem. Phys.* **2007**, *126*, No. 014101.

(29) Morrone, J. A.; Zhou, R.; Berne, B. J. Molecular Dynamics with Multiple Time Scales: How to Avoid Pitfalls. *J. Chem. Theory Comput.* **2010**, *6*, 1798−1804.

(30) Kolafa, J. Time-reversible always stable predictor−corrector method for molecular dynamics of polarizable molecules. *J. Comput. Chem.* **2004**, *25*, 335−342.
(31) Liberatore, E.; Meli, R.; Rothlisberger, U. A Versatile Multiple Time Step Scheme for Efficient ab Initio Molecular Dynamics Simulations. *J. Chem. Theory Comput.* **2018**, *14*, 2834−2842.
(32) 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.
(33) Wang, W.; Skeel, R. D. Fast evaluation of polarizable forces. *J. Chem. Phys.* **2005**, *123*, 164107.
(34) Nocito, D.; Beran, G. J. O. Fast divide-and-conquer algorithm for evaluating polarization in classical force fields. *J. Chem. Phys.* **2017**, *146*, 114103.
(35) Bennett, C. H. Efficient estimation of free energy differences from Monte Carlo data. *J. Comput. Phys.* **1976**, *22*, 245−268.
(36) Rackers, J. A.; Laury, M. L.; Lu, C.; Wang, Z.; Lagardère, L.; Piquemal, J.-P.; Ren, P.; Ponder, J. W.; Schnieders, M. J. TINKER 8: A Modular Software Package for Molecular Design and Simulation. *J. Comput. Chem.* **2018**, *14*, 5273.
(37) Schnieders, M. J.; Baltrusaitis, J.; Shi, Y.; Chattree, G.; Zheng, L.; Yang, W.; Ren, P. The Structure, Thermodynamics, and Solubility of Organic Crystals from Simulation with a Polarizable Force Field. *J. Chem. Theory Comput.* **2012**, *8*, 1721−1736.
(38) Jiang, W.; Hardy, D. J.; Phillips, J. C.; MacKerell, A. D.; Schulten, K.; Roux, B. High- Performance Scalable Molecular Dynamics Simulations of a Polarizable Force Field Based on Classical Drude Oscillators in NAMD. *J. Phys. Chem. Lett.* **2011**, *2*, 87−92.
(39) Huang, J.; Rauscher, S.; Nawrocki, G.; Ran, T.; Feig, M.; de Groot, B. L.; Grub-mueller, H.; MacKerell, A. D., Jr. CHARMM36m: an improved force field for folded and intrinsically disordered proteins. *Nat. Methods* **2017**, *14*, 71−73.
(40) Maier, J. A.; Martinez, C.; Kasavajhala, K.; Wickstrom, L.; Hauser, K. E.; Simmerling, C. ff14SB: Improving the Accuracy of Protein Side Chain and Backbone Parameters from ff99SB. *J. Chem. Theory Comput.* **2015**, *11*, 3696−3713.

DOI: 10.1021/acs.jpclett.9b00901
J. Phys. Chem. Lett. 2019, 10, 2593−2599

Bibliography

[1] Mark Tuckerman. *Statistical Mechanics: Theory and Molecular Simulation*. OUP Oxford, February 2010. Google-Books-ID: Lo3Jqc0pgrcC.

[2] H. F. Trotter. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society*, 10(4):545–551, 1959.

[3] Gilbert Strang. On the Construction and Comparison of Difference Schemes. *SIAM Journal on Numerical Analysis*, 5(3):506–517, 1968.

[4] Daniel T. Margul and Mark E. Tuckerman. A Stochastic, Resonance-Free Multiple Time-Step Algorithm for Polarizable Models That Permits Very Large Time Steps. *Journal of Chemical Theory and Computation*, 12(5):2170–2180, May 2016.

[5] Benedict Leimkuhler and Charles Matthews. Rational Construction of Stochastic Numerical Methods for Molecular Sampling. *Applied Mathematics Research eXpress*, 2013(1):34–56, January 2013. BAOAB.

[6] Benedict Leimkuhler and Charles Matthews. Robust and efficient configurational molecular sampling via Langevin dynamics. *The Journal of Chemical Physics*, 138(17):174102, May 2013.

[7] Jean-Paul Ryckaert, Giovanni Ciccotti, and Herman J.C Berendsen. Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. *Journal of Computational Physics*, 23(3):327–341, March 1977.

[8] Berk Hess, Henk Bekker, Herman J. C. Berendsen, and Johannes G. E. M. Fraaije. LINCS: A linear constraint solver for molecular simulations. *Journal of Computational Chemistry*, 18(12):1463–1472, 1997.

[9] K. Anton Feenstra, Berk Hess, and Herman J. C. Berendsen. Improving efficiency of large time-scale molecular dynamics simulations of hydrogen-rich systems. *Journal of Computational Chemistry*, 20(8):786–798, 1999.

[10] Joseph A. Morrone, Thomas E. Markland, Michele Ceriotti, and B. J. Berne. Efficient multiple time scale molecular dynamics: Using colored noise thermostats to stabilize resonances. *The Journal of Chemical Physics*, 134(1):014103, January 2011.

[11] Alex Albaugh, Mark E. Tuckerman, and Teresa Head-Gordon. Combining Iteration-Free Polarization with Large Time Step Stochastic-Isokinetic Integration. *Journal of Chemical Theory and Computation*, 15(4):2195–2205, April 2019.

Conclusion

Molecular dynamics is an essential tool for understanding the microscopic world, but requires the development of meaningful, accurate models correctly accounting for the complexity of atomic and molecular behaviours. From simple chemical reactions to biological systems interacting, a wide range of methods, based on quantum and classical mechanics, have been studied and perfected.

Through careful numerical simulations of the systems of interest, MD gives access to invaluable thermodynamical quantities, including free energies or phase change enthalpies, but also to dynamical and structural ones, characterizing the movement of particles and their organization. Accuracy of the simulations is directly dependent on the models chosen to reproduce physical interactions, namely the force fields. The family of force fields is large, and comprises several generations, where the oldest ones only mimick the strongest coupling terms using simple approximations. *Polarizable* force fields, on the other end, are the youngest generation. On top of using refined (second, third order) approximations that were the mainspring of their predecessors, polarizable force fields aim at taking the electronic density into account, without spending the (very expensive) cost of quantum calculations. A few models were developed in order to explain the behaviour of this highly complex contributions as simply as possible. Each comes with its own tools and calculation process, its perks and drawbacks.

This work focused on the *induced dipole* polarization model, assimilating the polarized electronic density around atoms to a point-dipole vector. This model show the best accuracy amongs its fellows, as well as being the most efficient computationally-wise. However, using this approach requires the implementation of a self-consistent solver which considerably slows down the simulation.

We thus proposed a new algorithm, the Truncated Conjugate Gradient (TCG), based on a truncation of the state-of-the-art polarization solver. This new solver has several advantages over the previous one: since the user can choose beforehand the number of solver iterations to be performed, the scheme

becomes non-iterative, and its cost can be chosen to be low. Various simulations show that even at very low order, the TCG yields very good results, which validates its interest as a faster polarization solver.

The non-iterative nature of the method also means that one can derive exact formula for the induced dipole values, their associated interaction energies, and the forces they elicit. As a consequence, one can compute forces that are exactly consistent with the induced polarization model, ensuring a much better energy conservation. Essentially, TCG dynamics can be described as very stable simulations whose potential energy surface is offsetted by a small quantity.

Refinements can be implemented to further improve the TCG's performance in terms of accuracy. The peek-step, in particular, is a parametrizable extra step in the polarization solving process allowing to obtain very accurate polarization energies (and even to cancel the offset evoked earlier).

These developments are conceptually quite simple, as they are merely the consequences of a truncation of a very well studied algorithm, the Conjugate Gradient. However, they are quite involved in terms of implementation, and producing a usable program is not trivial. We thus proposed a strategy to properly code the TCG and its associated forces.

The new algorithm was tested on many different systems such as liquid water boxes of different sizes, solvated proteins, and even ionic liquids. Static properties, such as polarization and potential energies, a dynamical property (the diffusion coefficient), radial distribution functions were computed and compared to references. In every instance, TCG performed well, with the first order of truncation (TCG1) appearing as a good choice for a very cheap polarization solver, while the second order (TCG2) establishes itself as a perfectly viable alternative to the slower, state-of-the art solvers, exhibiting very small errors in the results.

To assess the behaviour of TCG on more advanced properties, hydration free energies of Na^+ cations were computed. Free energies are very important thermodynamical quantities from the purely physical problems to biochemical systems, but their estimation requires very fine computations. TCG robustness is reaffirmed, as the second order of truncation yields almost perfect results. The first order of truncation plays again the role of a very cheap estimator carrying a more important error. Reconsidering on the free energy calculation schemes, a reweighting technique to post-treat low-precision trajectory using high-precision parameters allow to extract better values for a cheaper price was presented.

Nevertheless, we observed that an important number of different simulations had to be carried out in order to extract reliable free energy values. Keeping our initial objective of acceleration of the polarizable simulations, we then turned towards a cornerstone of molecular dynamics: the integrator.

Simple (Euler) approximations can give stable integrator schemes. Yet an interesting zoology of methods can be explored in order to improve these "primary" integrators. We presented step-by-step improvements of the computational speed, following the developments hereafter. Using a slightly more involved classical mechanics framework, splitting strategies such as RESPA and RESPA1 emerged. These splitting treat motions differently depending on their typical frequency, so that adapted time-steps can be used. Stabilization of the accelerated dynamics can be achieved thanks to the Langevin equations and the BAOAB integration scheme. By reconsidering the splitting of the polarization forces, we put our TCG to good use by exploiting its stability and rapidity properties. As a last effort, the Hydrogen Mass Repartitioning, preventing motions of the highest frequency (involving hydrogen atoms), was added to our new integration strategy.

Numerical tests were performed throughout this design process, allowing us to finally propose two optimal integrators. The first one is tightly conservative and reaches speedup up to five times the standard (1 fs) integrators. The second one is slightly less precise when it comes to the dynamical properties, but allows seven-fold speedups of the simulations, while maintaining very good accuracy on static properties and even free energies.

The work presented in this thesis, initially started on the simple truncation of a polarization solver, proved to surpass our initial expectations. Indeed, the Truncated Conjugate Gradient reshapes the landscape of polarizable molecular dynamics, by making it much more affordable as well as more stable. Through a lot of efforts in implementation and testing, we now achieved a very good understanding of its behaviour and its possible refinements. The computational bottleneck that induced polarization represented is now solved. Polarizable Molecular Dynamics can not be considered as "slow" anymore.

The numerical experiments already carried out using TCG proved its versatility and applicability, and it was incorporated into the development of new MD integrators, who benefited from its acceleration and even more strikingly from its improvements on steadiness of the dynamics with respect to large time-steps. These new developments have already been used as production codes by several members of our laboratory, who benefited from their steadiness and improved performances.

Furthermore, all this work was done with a high-performance oriented mindset, and is highly parallel. It is implemented within Tinker-HP as one of the engines of the 1.2.0 release. Our seven-fold acceleration and its excellent free energy results will thus be available for all and for all computational resources. Thanks to these improvements, we hope to open the door to complex biological problems, such as protein-protein interactions and docking processes, and to unlock very long scale simulations aiming at the microsecond frontier. The proposed accelerations could greatly benefit fields such as pharmacology, where computing accurate – and fast ! – free energies is important, enabling better tools for predictions and drug design.

As a conclusion, this work is now growing branches in the MD integrators domain, in free energy computations, and in Monte-Carlo simulations, overcoming the initial polarization problem. The mathematical basis on which the TCG was built makes it systematically improvable, while providing us with well controlled methods; many more refinements could be implemented (better guesses, new preconditioners...) that have yet to be explored. Next implementation steps could benefit from automatic differentiation to avoid their complexity and therefore further increase accuracy through inclusion of more advanced mathematics.

TCG opened the door to a whole family of polarization solvers, that we can only hope to explore in the future.

Appendix

Notations for the Truncated Conjugate Gradient expressions

Vectors

- $\mathbf{r}_0 = \mathbf{E} - \mathbf{T}\boldsymbol{\mu}_0$
- $\mathbf{p}_0 = \mathbf{r}_0$
- $\mathbf{P}_1 = \mathbf{T}\mathbf{r}_0$
- $\mathbf{P}_2 = t_2\mathbf{P}_1 - t_4\mathbf{T}^2\mathbf{r}_0$
- $\mathbf{P}_3 = (1 + \beta_2 t_2)\mathbf{T}\mathbf{r}_0 - (t_4 + \beta_2 t_4)\mathbf{T}\mathbf{P}_1 + \gamma_1\mathbf{T}\mathbf{P}_2$

Scalars

- $n_0 = \mathbf{r}_0^{\mathbf{T}}\mathbf{r}_0$
- $t_1 = \mathbf{r}_0^T\mathbf{P}_1$
- $t_2 = \dfrac{n_0||\mathbf{P}_1||^2}{t_1^2}$
- $t_3 = t_1\mathbf{P}_1^T\mathbf{P}_2$
- $t_4 = \dfrac{n_0}{t_1}$
- $t_5 = \mathbf{P}_1^T\mathbf{P}_2$
- $t_8 = t_2||\mathbf{P}_1||^2 - t_4 t_9$
- $t_9 = \mathbf{r}_0^T\mathbf{T}^3\mathbf{r}_0$
- $t_{10} = t_1^2 - n_0||\mathbf{P}_1||^2$
- $\gamma_1 = \dfrac{t_1^2 - n_0||\mathbf{P}_1||^2}{t_3}$
- $sp_0 = \mathbf{r}_0^T\mathbf{E}$
- $sp_1 = \mathbf{P}_1^T\mathbf{E} = \mathbf{E}^T\mathbf{T}\mathbf{r}_0$
- $b_1 = sp_0 - \gamma_1 sp_1$
- $b_2 = sp_0 t_2 - t_4 sp_1$
- $spp_1 = \langle\boldsymbol{\alpha}\mathbf{E}, \mathbf{E}\rangle$
- $spp_2 = \langle\boldsymbol{\alpha}\mathbf{T}\mathbf{r}_0, \mathbf{E}\rangle$

- $\beta_2 = \dfrac{n_0 + t_4^2||\mathbf{P}_1||^2 + \gamma_1^2||\mathbf{P}_2||^2 - 2t_1t_4 - 2\gamma_1 t_4||\mathbf{P}_1||^2 + 2\gamma_1 t_4 t_5}{(t_2-1)n_0}$
- $\gamma_2 = \dfrac{n_0 + t_4^2||\mathbf{P}_1||^2 + \gamma_1^2||\mathbf{P}_2||^2 - 2t_1t_4 - 2\gamma_1 t_4||\mathbf{P}_1||^2 + 2\gamma_1 t_4 t_5}{(1+\beta_2 t_2)\mathbf{r}_0^T\mathbf{P}_3 - (t_4+\beta_2 t_4)\mathbf{P}_1^T\mathbf{P}_3 + \gamma_1\mathbf{P}_2^T\mathbf{P}_3}$

Analytical expression of the gradients

Second-order TCG (no peek-step):

$$E'_{\text{pol, TCG2}} = \frac{1}{2}\Big(\langle \mathbf{E}', \boldsymbol{\mu}_{\text{TCG2}}\rangle + \langle \boldsymbol{\mu}'_0, \mathbf{E}\rangle + \langle \mathbf{r}'_0, a^{(2)}_{1,0}\mathbf{E} + a^{(2)}_{1,-1}\mathbf{TE} + a^{(1)}_{1,1}\mathbf{r}_0 + a^{(2)}_{1,2}\mathbf{Tr}_0 + a^{(2)}_{1,3}\mathbf{T}^2\mathbf{r}_0 + a^{(2)}_{1,4}\mathbf{T}^3\mathbf{r}_0\rangle + \langle \mathbf{T}'\mathbf{r}_0, a^{(2)}_{2,0}\mathbf{E} + a^{(2)}_{2,1}\mathbf{r}_0 + a^{(2)}_{2,2}\mathbf{Tr}_0 + a^{(2)}_{2,3}\mathbf{T}^2\mathbf{r}_0\rangle + \langle \mathbf{T}'\mathbf{T}\mathbf{r}_0, a^{(2)}_{3,1}\mathbf{r}_0\rangle + \langle \mathbf{T}'\mathbf{T}^2\mathbf{r}_0, a^{(2)}_{4,1}\mathbf{r}_0\rangle\Big) \tag{38}$$

- $a^{(2)}_{1,0} = t_4 + \gamma_1 t_2$
- $a^{(2)}_{1,-1} = -\gamma_1 t_4$
- $a^{(2)}_{1,1} = \frac{2b_1}{t_1} - \frac{2np_1b_2}{t_3} - 2\frac{np_1^2t_{10}b_2}{t_3^2t_1} + 2\frac{t_9t_{10}b_2}{t_3^2} + 2\frac{np_1sp_0\gamma_1}{t_1^2}$
- $a^{(2)}_{1,2} = -2\frac{n_0b_1}{t_1^2} + 4\frac{t_1b_2}{t_3} - 2\frac{n_0t_9t_{10}b_2}{t_1t_3^2} + 4\frac{t_2np_1t_{10}b_2}{t_3^2} - 2\frac{t_8t_{10}b_2}{t_3^2} - 4\frac{4n_0np_1sp_0\gamma_1}{t_1^3}$
- $a^{(2)}_{1,3} = -4\frac{t_1t_2t_{10}b_2}{t_3^2} - 2\frac{n_0b_2}{t_3} + 2\frac{n_0sp_0\gamma_1}{t_1^2}$
- $a^{(2)}_{1,4} = 2\frac{t_1t_4t_{10}b_2}{t_3^2}$
- $a^{(2)}_{2,0} = -\gamma_1 t_4$
- $a^{(2)}_{2,1} = -\frac{n_0b_1}{t_1^2} + 2\frac{t_1b_2}{t_3} - \frac{n_0t_9t_{10}b_2}{t_1t_3^2} + 2\frac{t_2np_1t_{10}b_2}{t_3^2} - \frac{t_8t_{10}b_2}{t_3^2} - 2\frac{n_0np_1sp_0\gamma_1}{t_1^3}$
- $a^{(2)}_{2,2} = -\frac{n_0b_2}{t_3} - 2\frac{t_1t_2t_{10}b_2}{t_3^2} + \frac{n_0sp_0\gamma_1}{t1_2}$
- $a^{(2)}_{2,3} = \frac{t_1t_4t_10b_2}{t_3^2}$
- $a^{(2)}_{3,1} = -\frac{n_0b_2}{t_3} - 2\frac{t_1t_2t_{10}b_2}{t_3^2} + \frac{n_0\gamma_1sp_0}{t_1^2}$
- $a^{(2)}_{4,1} = \frac{t_1t_4t_{10}b_2}{t_3^2}$

Gradients of the TCG polarization energies using a peek-step

At first order:

$$E'_{\text{peek, TCG1}} = \langle \boldsymbol{\mu}_{\text{peek, TCG1}}, \mathbf{E}'\rangle + \langle \mathbf{r}'_0, a^{(1,p)}_{1,\alpha 0}\boldsymbol{\alpha}\mathbf{E} + a^{(1,p)}_{1,1\alpha}\mathbf{T}\boldsymbol{\alpha}\mathbf{E} + a^{(1,p)}_{1,1}\mathbf{r}_0 + a^{(1,p)}_{1,2}\mathbf{Tr}_0\rangle + \langle \mathbf{T}'\mathbf{r}_0, a^{(1,p)}_{2,1}\mathbf{r}_0 + a^{(1,p)}_{2,\alpha 0}\boldsymbol{\alpha}\mathbf{E}\rangle \tag{39}$$

using

- $a_{1,\alpha 0}^{(1,p)} = \omega$
- $a_{1,1\alpha}^{(1,p)} = -t_4$
- $a_{1,1}^{(1,p)} = -\frac{2spp_1\omega}{t_1}a$
- $a_{1,2}^{(1,p)} = \frac{2n_0spp_1\omega}{t_1^2}$
- $a_{2,\alpha 0}^{(1,p)} = -t_4\omega$
- $a_{2,1}^{(1,p)} = \frac{n_0spp_1\omega}{t_1^2}$

At second order:

$$\begin{aligned} E'_{\text{peek, TCG2}} &= \langle \boldsymbol{\mu}_{\text{peek, TCG2}}, \mathbf{E}' \rangle \\ &+ \langle \mathbf{r}'_0, a_{1,1\alpha}^{(2,p)}\mathbf{T}\boldsymbol{\alpha}\mathbf{E} + a_{1,2\alpha}^{(2,p)}\mathbf{T}^2\boldsymbol{\alpha}\mathbf{E} + a_{1,1}^{(2,p)}\mathbf{r}_0 + a_{1,2}^{(2,p)}\mathbf{T}\mathbf{r}_0 + a_{1,3}^{(2,p)}\mathbf{T}^2\mathbf{r}_0 + a_{1,4}^{(2,p)}\mathbf{T}^3\mathbf{r}_0 \rangle \\ &+ \langle \mathbf{T}'\mathbf{r}_0, a_{2,\alpha 0}^{(2,p)}\boldsymbol{\alpha}\mathbf{E} + a_{2,1\alpha}^{(2,p)}\mathbf{T}\boldsymbol{\alpha}\mathbf{E} + a_{2,1}^{(2,p)}\mathbf{r}_0 + a_{2,2}^{(2,p)}\mathbf{T}\mathbf{r}_0 + a_{2,3}^{(2,p)}\mathbf{T}^2\mathbf{r}_0 \rangle \\ &+ \langle \mathbf{T}'\mathbf{T}\mathbf{r}_0, a_{3,\alpha 0}^{(2,p)}\boldsymbol{\alpha}\mathbf{E} + a_{3,1}^{(2,p)}\mathbf{r}_0 + a_{3,2}^{(2,p)}\mathbf{T}\mathbf{r}_0 \rangle + \langle \mathbf{T}'\mathbf{T}^2\mathbf{r}_0, a_{4,1}^{(2,p)}\mathbf{r}_0 \rangle \end{aligned} \tag{40}$$

- $a_{1,1\alpha}^{(2,p)} = -\omega t_2\gamma_1$
- $a_{1,2\alpha}^{(2,p)} = -\omega t_4\gamma_1$
- $a_{1,1}^{(2,p)} = -\frac{2np_1\omega\gamma_1spp_1}{t_1^2} + (\omega t_2spp_1 + \omega t_4spp_2)\left(\frac{2np_1}{t_3} + \frac{2np_1^2t_{10}}{t_1t_3^2} + \frac{2t_9t_{10}}{t_3^2}\right) - \frac{2\omega\gamma_1spp_2}{t_1}$
- $a_{1,2}^{(2,p)} = \frac{4n_0np_1\omega\gamma_1spp_1}{t_1^3} + (\omega t_2spp_1 + \omega t_4spp_2)\left(-\frac{4t_1}{t_3} + \frac{2n_0t_9t_{10}}{t_1t_3^2} - \frac{4np_1t_2t_{10}}{t_3^2} + \frac{2t_8t_{10}}{t_3^2}\right) + \frac{2n_0\omega\gamma_1spp_2}{t_1^2}$
- $a_{1,3}^{(2,p)} = -\frac{2n_0\gamma_1\omega spp_1}{t_1^2} + (\omega t_2spp_1 + \omega t_4spp_2)\left(\frac{4t_1t_2t_{10}}{t_1t_3^2} + \frac{2n_0}{t_3}\right)$
- $a_{1,4}^{(2,p)} = -(\omega t_2spp_1 + \omega t_4spp_2)\frac{2t_1t_4t_{10}}{t_3^2}$
- $a_{2,\alpha 0}^{(2,p)} = -\omega\gamma_1t_2$
- $a_{2,1\alpha}^{(2,p)} = -\omega t_4\gamma_1$
- $a_{2,1}^{(2,p)} = \frac{2n_0np_1\gamma_1\omega spp_1}{t_1^2} + (\omega t_2spp_1 + \omega t_4spp_2)\left(-\frac{2t_1}{t_3} + \frac{n_0t_9t_{10}}{t_1t_3^2} - \frac{2np_1t_2t_{10}}{t_3^2} + \frac{t_8t_{10}}{t_3^2}\right) + \frac{n_0\omega\gamma_1spp_1}{t_1^2}$
- $a_{2,2}^{(2,p)} = -\frac{n_0\omega\gamma_1spp_1}{t_1^2} + (\omega t_2spp_1 + \omega t_4spp_2)\left(\frac{n_0}{t_3} + \frac{2t_1t_2t_{10}}{t_3^2}\right)$
- $a_{2,3}^{(2,p)} = (\omega t_2spp_1 + \omega t_4spp_2)\frac{t_1t_4t_{10}}{t_3^2}$
- $a_{3,\alpha 0}^{(2,p)} = -\omega t_4\gamma_1$
- $a_{3,1}^{(2,p)} = -\frac{n_0\gamma_1\omega spp_1}{t_1^2} + (\omega t_2spp_1 + \omega t_4spp_2)\left(\frac{n_0}{t_3} + \frac{2t_1t_2t_{10}}{t_3^2}\right)$
- $a_{3,2}^{(2,p)} = -(\omega t_2spp_1 + \omega t_4spp_2)\frac{t_1t_4t_{10}}{t_3^2}$
- $a_{4,1}^{(2,p)} = -(\omega t_2spp_1 + \omega t_4spp_2)\frac{t_1t_4t_{10}}{t_3^2}$

TCG timings using a 2 fs time-step

Solver	Refinements			Water (S4)		Ubiquitin		$[dmim^+][Cl^-]$	
order	Prec.	Guess	Peek	ns/day	diff.	ns/day	diff.	ns/day	diff.
Ref.				3.06	0.0	0.58	0.0	1.57	0.0
PCG8				2.25	-26.6	0.40	-31.6	1.11	-29.1
Ref.	-	-	-	5.19	0.0	0.97	0.0	1.50	0.0
PCG8	-	-	-	4.02	-22.6	0.71	-27.6	1.07	-28.1
TCG1	-	-	-	7.51	44.7	1.67	71.0	2.86	91.5
TCG1	-	●	-	6.17	18.9	1.15	18.0	3.63	142.8
TCG1	●	-	-	7.40	42.6	1.67	71.3	2.73	82.9
TCG1	-	-	ω_{fit}	6.61	27.4	1.32	35.2	2.08	38.9
TCG1	●	●	$\omega = 1$	6.12	18.0	1.26	28.9	1.98	32.5
TCG1	●	●	ω_{fit}	5.82	12.2	1.23	26.4	1.91	27.9
TCG2	-	-	-	6.37	22.7	1.34	37.8	2.26	51.4
TCG2	●	-	-	6.38	23.0	1.38	41.2	2.27	51.5
TCG2	●	●	-	5.29	1.9	1.04	7.0	1.57	5.3
TCG2	-	●	-	5.13	-1.1	1.02	4.7	2.96	98.0
TCG2	-	-	ω_{fit}	5.59	7.7	1.06	8.3	1.62	8.3
TCG2	●	●	$\omega = 1$	5.10	-1.8	1.00	2.2	1.50	0.1
TCG2	●	●	ω_{fit}	5.01	-3.5	0.97	-0.1	1.46	-2.2

Table 10: **Timings for various TCG setups.** For each simulation, 1000 2 fs time-steps were run using the RESPA integrator. Simulations on the water system S4 were performed using 24 cores; ubiquitin and the dimethylimidazolium solution with 48 cores. All timings are given in ns/day. The reference timings were obtained using a PCG solver with a 10^{-5} convergence criterion. "PCG8" stands for the Preconditioned Conjugate Solver with a 10^{-8} convergence criterion. "Diff." designates the relative difference, in percents, with respect to the reference.

Parametrization of the peek-step, variations of ω

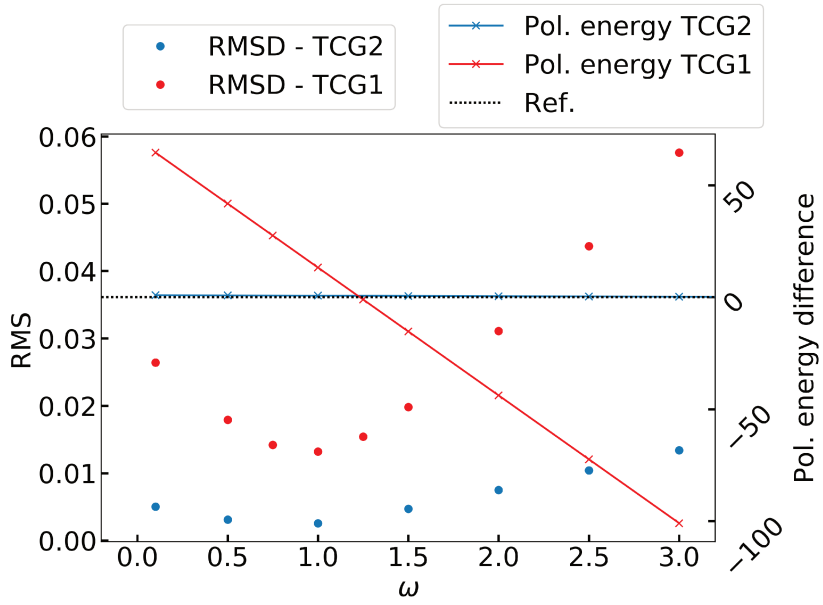


Figure 5: **Water – Influence of ω.** Polarization energy and RMS on the induced dipoles plotted as a function of ω, for 216 water molecules. The dashed black line represents the reference polarization energy, obtained with PCG.

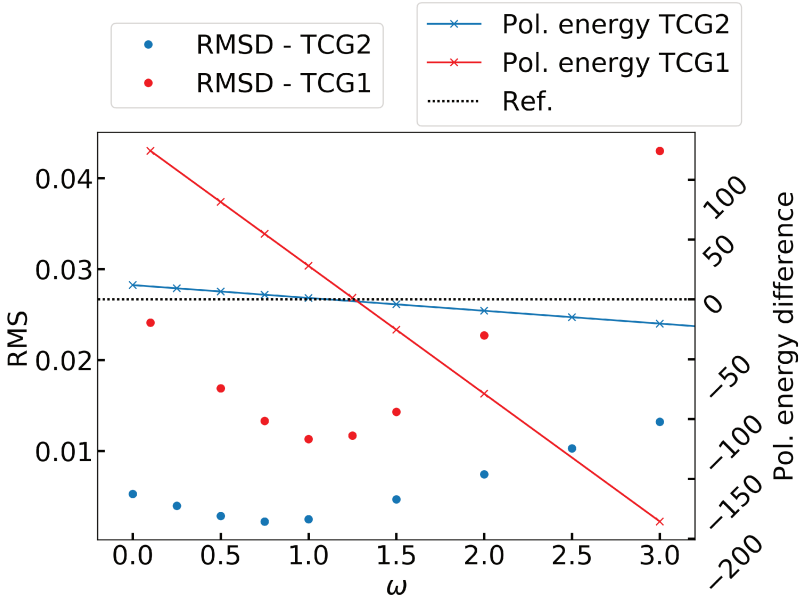


Figure 6: **GAG – Influence of ω.** Polarization energy and RMS on the induced dipoles plotted as a function of ω, computed on the GAG protein.

Chemical Science

EDGE ARTICLE

View Article Online
View Journal | View Issue

Cite this: *Chem. Sci.*, 2018, **9**, 956

Tinker-HP: a massively parallel molecular dynamics package for multiscale simulations of large complex systems with advanced point dipole polarizable force fields†

Louis Lagardère,[abc] Luc-Henri Jolly,[b] Filippo Lipparini,[d] Félix Aviat,[c] Benjamin Stamm,[e] Zhifeng F. Jing,[f] Matthew Harger,[f] Hedieh Torabifard,[g] G. Andrés Cisneros,[h] Michael J. Schnieders,[i] Nohad Gresh,[c] Yvon Maday,[jkl] Pengyu Y. Ren,[f] Jay W. Ponder[m] and Jean-Philip Piquemal*[cfk]

We present Tinker-HP, a massively MPI parallel package dedicated to classical molecular dynamics (MD) and to multiscale simulations, using advanced polarizable force fields (PFF) encompassing distributed multipoles electrostatics. Tinker-HP is an evolution of the popular Tinker package code that conserves its simplicity of use and its reference double precision implementation for CPUs. Grounded on interdisciplinary efforts with applied mathematics, Tinker-HP allows for long polarizable MD simulations on large systems up to millions of atoms. We detail in the paper the newly developed extension of massively parallel 3D spatial decomposition to point dipole polarizable models as well as their coupling to efficient Krylov iterative and non-iterative polarization solvers. The design of the code allows the use of various computer systems ranging from laboratory workstations to modern petascale supercomputers with thousands of cores. Tinker-HP proposes therefore the first high-performance scalable CPU computing environment for the development of next generation point dipole PFFs and for production simulations. Strategies linking Tinker-HP to Quantum Mechanics (QM) in the framework of multiscale polarizable self-consistent QM/MD simulations are also provided. The possibilities, performances and scalability of the software are demonstrated *via* benchmarks calculations using the polarizable AMOEBA force field on systems ranging from large water boxes of increasing size and ionic liquids to (very) large biosystems encompassing several proteins as well as the complete satellite tobacco mosaic virus and ribosome structures. For small systems, Tinker-HP appears to be competitive with the Tinker-OpenMM GPU implementation of Tinker. As the system size grows, Tinker-HP remains operational thanks to its access to distributed memory and takes advantage of its new algorithmic enabling for stable long timescale polarizable simulations. Overall, a several thousand-fold acceleration over a single-core computation is observed for the largest systems. The extension of the present CPU implementation of Tinker-HP to other computational platforms is discussed.

Received 18th October 2017
Accepted 24th November 2017

DOI: 10.1039/c7sc04531j

rsc.li/chemical-science

1 Introduction

Over the last 60 years, classical Molecular Dynamics (MD) has been an intense field of research with a high rate growth. Indeed, solving Newton equations of motion to resolve the time-dependent dynamics of atoms within large molecules allows to perform simulations in various fields of research ranging from materials to biophysics and chemistry. For example, in the

[a]*Sorbonne Université, Institut des Sciences du Calcul et des Données, Paris, France*
[b]*Sorbonne Université, Institut Parisien de Chimie Physique et Théorique, CNRS, FR 2622, Paris, France*
[c]*Sorbonne Université, Laboratoire de Chimie Théorique, UMR 7616, CNRS, Paris, France. E-mail: jpp@lct.jussieu.fr*
[d]*Universita di Pisa, Dipartimento di Chimica e Chimica Industriale, Pisa, Italy*
[e]*MATHCCES, Department of Mathematics, RWTH Aachen University, Aachen, Germany*
[f]*The University of Texas at Austin, Department of Biomedical Engineering, TX, USA*
[g]*Department of Chemistry, Wayne State University, Detroit, MI 48202, USA*
[h]*Department of Chemistry, University of North Texas, Denton, TX 76202, USA*
[i]*The University of Iowa, Department of Biomedical Engineering, Iowa City, IA, USA*
[j]*Sorbonne Université, Laboratoire Jacques-Louis Lions, UMR 7598, CNRS, Paris, France*
[k]*Institut Universitaire de France, Paris, France*
[l]*Brown University, Division of Applied Maths, Providence, RI, USA*
[m]*Washington University in Saint Louis, Department of Chemistry, Saint Louis, MI, USA*

† Electronic supplementary information (ESI) available. See DOI: 10.1039/c7sc04531j

community of protein simulations, classical force fields (FF) such as CHARMM,[1] AMBER,[2] OPLS,[3] GROMOS[4] and others,[5] enabled large scale simulations on complex systems thanks to the low computational cost of their energy function. In that context, various simulation packages appeared, often associated to these FF such as the popular CHARMM,[6] GROMOS[7] and AMBER softwares.[8] Among these, Tinker (presently version 8 (ref. 9)) was introduced in 1990 with the philosophy of being both user friendly and to provide a reference toolbox for developers. Later on, the evolution of computer systems enabled the emergence of massively parallel softwares dedicated to molecular simulations such as LAMMPS,[10] NAMD,[11] Gromacs,[12] AMBER (PME-MD),[13] DLPOLY,[14] Genesis[15] or Desmond.[16] As they were granted the use of large computational resources, access to million atoms systems and biological time scales became possible.[17] Nevertheless, up to now, such simulations are mainly limited to first-generation molecular mechanics (MM) models that remains confined to a lower resolution approximation of the true quantum mechanical Born–Oppenheimer potential energy surfaces (PES). However, beside these methods, more advanced second generation "polarizable" force fields (PFF) emerged in the last 30 years.[18–28] Grounded on Quantum Mechanics (QM) and usually calibrated on the basis of Energy Decomposition Analysis (EDA),[29] they go beyond pairwise approximation by including explicit many-body induction effects such as polarization and in some cases charge-transfer. Fluctuating charges, classical Drude approaches or point dipole coupled to point-charge models using distributed polarizabilities are among the most studied techniques aiming to include polarization effects.[28] On the accuracy side, some PFF go beyond the point charge approximation incorporating a more detailed representation of the permanent and induced charge distributions using QM-derived distributed multipoles and polarizabilities.[18,19,24,26] Recently, a third-generation PFF using distributed frozen electronic densities in order to incorporate short-range quantum effects[30] appeared. In term of PES, these advanced force fields clearly tend to offer improved accuracy, better transferability and therefore are hoped to be more predictive. Unfortunately, everything has a cost: such elegant methods are more complex by design, and are therefore computationally challenging. Until recently the more advanced point dipole polarizable approaches were thought to be doomed for the simulation of realistic systems due to the evaluation cost of the polarization energy. Large scale polarizable production MD simulations were limited to the use of the Drude-type/point-charge model (using an extended Lagrangian propagation scheme)[31] that was found to be more tractable than point dipole models (using iterative solvers) coupled to multipolar representation of the permanent charge distribution. Nevertheless, despite this scalability issue, time was not lost and accurate models were developed such as the Tinker package, original home of the multipolar AMOEBA PFF,[24] specialized in offering a dedicated development platform with all required advanced algorithms for these accurate techniques. Moreover, ten years ago, a hardware technical revolution in the field of High Performance Computing (HPC), had a profound impact on MD simulations with classical FF.[32] Indeed, the introduction of Graphical Processing Units (GPUs) offered a brute force hardware acceleration to MD packages thanks to simple- or mixed-precision implementations.[33] Tinker benefited from the availability of this low cost but powerful type of hardware. It led to a GPU version of the code denoted Tinker-OpenMM.[34] The code is based both on Tinker and on the OpenMM library (now version 7 (ref. 35)) which pioneered the use of GPUs with polarizable force fields. Tinker-OpenMM offers a 200-fold acceleration compared to a regular single core CPU computation giving access to accurate free energy simulations. However, when one considers the need for biophysical simulations, this acceleration remains not sufficient.

The purpose of the present work is to push the scalability improvements of Tinker through new algorithms to explore strategies enabling a 1000-fold and more speedup. These new developments aim towards modern "big Iron" petascale supercomputers using distributed memory and the code design also offers consequent speedups on laboratory clusters and on multicore desktop stations. The philosophy here is to build a highly scalable double precision code, fully compatible and consistent with the canonical reference Tinker and Tinker-OpenMM codes. As the new code remains a part of the Tinker package, it is designed to keep its user-friendliness offered to both developers and users but also to provide an extended access to larger scale/longer timescale MD simulations on any type of CPU platforms. The incentive to produce such a reference double precision code is guided by the will to also perform scalable hybrid QM/MM MD simulations where rounding errors must be eliminated. This will bring us not to cut any corners in our numerical implementation with the key mantra that one should not scale at any cost, as the algorithms developed in this interdisciplinary project should be based on solid mathematical grounds.

The paper is organized as follows. First, we will present the newly developed extension of 3D spatial decomposition and memory distribution to polarizable point dipole models that is at the heart of Tinker-HP for short-range interactions. Then we will detail the handling of long-range electrostatic and polarization interactions with a new framework coupling Smooth Particle Ewald to Krylov iterative and non iterative polarization solvers. We will then introduce the possibilities of the software and show benchmarks for selected applications in the context of the AMOEBA PFF.[24,36] Finally, we will present functionalities of Tinker-HP that go beyond MD simulations in periodic boundary conditions as we conclude by drawing some perspectives about evolutions of the code towards next HPC platforms.

2 Accelerating polarizable molecular dynamics using massively parallel 3D spatial decomposition

In this section, we describe the first extension of 3D spatial decomposition to polarizable point dipoles models dedicated to production simulations. Indeed, in the past, point dipole model

implementations in parallel have been limited to the use of a few dozen processors.[37] In this section, we detail the parallelization strategy used in Tinker-HP to overcome this problem and to deal with local interactions, including the direct-space part of electrostatic and polarization interactions. The long-range, reciprocal field part of such interactions, is discussed in Section 3.

2.1 State of the art in massively parallel implementation of classical molecular dynamics simulations

Several strategies[10–12,16] have been devised in order to treat short-range interactions on large-scale parallel computers using distributed memory parallelism. In Tinker-HP, we have implemented a spatial decomposition (or domain decomposition) method. In this approach, the simulation domain is decomposed in 3D blocks and each block is assigned to a processor. Each processor then handles the computation of the forces and the update of the coordinates for the atoms assigned to the block at each time-step. This strategy is motivated by the fact that the interactions considered are short-range, and that the positions of the atoms do not change much between two consecutive time-steps. An example of such a decomposition with $3 \times 3 \times 3 = 27$ blocks is given in Fig. 1. One can show[10] that if the cutoff (r_c) involved in the short-range interactions is superior to the size of an edge of a block, which is the case with a high number of processors, the amount of data to be communicated in and out of each processor at each time step (the so-called communication volume) scales like $\mathcal{O}({r_c}^3)$ (if the density of the system is uniform) independently of the number of processors. As a consequence, the communications are local which is an advantage of this method over the other ones. However, achieving a good load-balancing is harder using this strategy when the density of the system is not uniform or when the simulation box is not a parallelepiped.

Let us give more details about the algorithm and the main steps required to perform a time step of MD using this method. We assume that the simulated system resides in a box that has been divided in as many 3D blocks as the number of processors used. Let us focus on a processor that has been assigned a 3D block and let us assume that this processor knows the current positions, velocities and accelerations of the atoms currently belonging to this block. In integrator schemes such as velocity Verlet, the first integration step consists of an update of the local positions and a first update of the velocities. Because of these position changes, some atoms may cross the local block boundaries and need to be reassigned to neighboring blocks. This step, that we will call "reassign step" only requires local communications between a small number of neighboring processes.

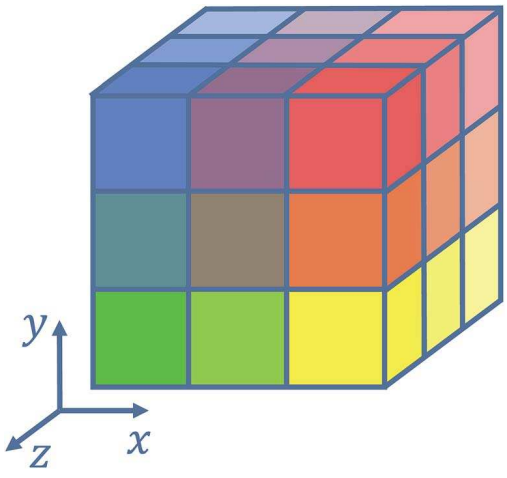


Fig. 1 Example of 3D spatial decomposition.

In the second step, the forces are computed and used for the second update of the velocities. This requires the processor to know the positions of all atoms within the interaction cutoff, that have to be communicated from the processors assigned to the blocks that are at distance inferior or equal to the cutoff. We will call this step, which also involves local communications (but that may involve more distant processors than the previous one) "position comm" step. Once this is done, the algorithm loops back to the first step.

The communication volume involved in the position comm step can be reduced by taking into account the pairwise nature of the fundamental operations needed to compute the forces. Given a pair of atoms, in fact, one needs to choose which processor will perform the elementary force computation. This can be done on the basis of a geometrical argument. Among the various methods, that are also called neutral territory approaches,[38] we choose the one presented by Shaw *et al.*,[16] known as the midpoint method.[38] This method picks out the processor that computes an interaction between two atoms as the one assigned to the subdomain where the center of the segment between the two atoms lies. As a consequence, each processor only needs to import information about atoms located at less than $\frac{r_c}{2}$ from its block: one can show that the communication volume is then, with d being the size of an edge of a subdomain, $V_{MP} = 3d^2 r_c + \frac{3}{4} d\pi {r_c}^2 + \frac{1}{6}\pi {r_c}^3$ as represented schematically in Fig. 2. This is a significant reduction with respect to the naive method,[38] especially at a high processors count. Note, however, that within this scheme, a processor might need to compute the elementary interaction between atoms that do not belong to its block.

Furthermore, once the elementary pairwise computation has been done, we can take advantage of Newton's third law and communicate the force back to both processors from which the positions originated ("force comm" step). This additional communication cost is in general negligible compared to the computational gain represented by the reduction of the computations of the forces by half.

Additionally, the midpoint approach is simple enough not to complicate too much the implementation, which is ideal for a platform like Tinker-HP, meant to be shared with a community of developers. Nevertheless, more elaborate techniques are interesting and have been shown to reduce asymptotically the amount of data that need to be communicated in the "position comm" step and in the "forces comm" step. We are currently studying these methods in the context of PFF to compare them

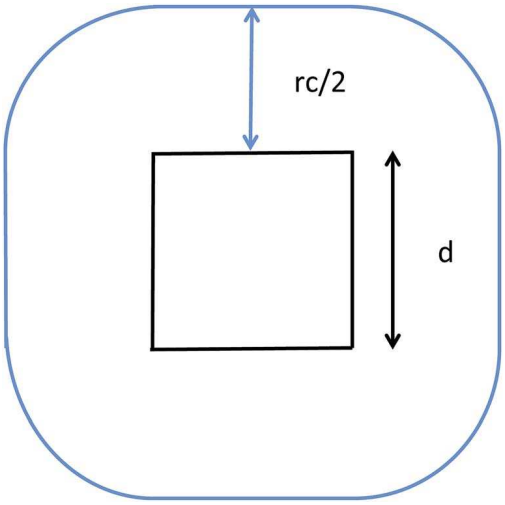


Fig. 2 Illustration of the midpoint rule in 2D: the square of edge *d* represents a subdomain assigned to a process and the blue line delimits the area that has to be imported by the latter.

to the present midpoint approach. Some of them should appear in future releases of our code.

The algorithmic structure of a parallel (short-range) MD step with spatial decomposition is shown in Fig. 3.

To address load balancing issues that may appear in non-homogeneous systems (when equally sized subdomains contain a very different number of atoms), a procedure in which the size of the subdomains is iteratively changed has been implemented.

2.2 Distributed memory for simulations using point dipole models

Distributed memory parallelism allows one to scatter the memory among the processors and thus to run simulations that would not be possible because of memory limitations. In Tinker-HP, almost all data are distributed, this being possible by reallocation of dynamically allocated arrays at regular intervals. For example, during the computation of the non-bonded forces at a O(N) computational cost using the linked-cell method,[39] the neighbor lists used, that are the most memory-consuming data structures of the program, are reallocated at the same frequency as they are updated. This is an important aspect allowing Tinker-HP to remain efficient on small computer clusters and desktop stations as the list builder will adapt to the situation.

Unfortunately, some data structures such as the arrays containing the global parameters are allocated once and for all and cannot be distributed. This is especially problematic for PFFs such as AMOEBA, that require more parameters than the classical ones: replicating these arrays for each processor would be prohibitive. This issue can be circumvented by using shared memory segments that can be managed with MPI (3.X) directives. This means that these data are allocated only once per node and are accessible by every processor within the node, reducing thus memory requirements by the number of processors of the node.

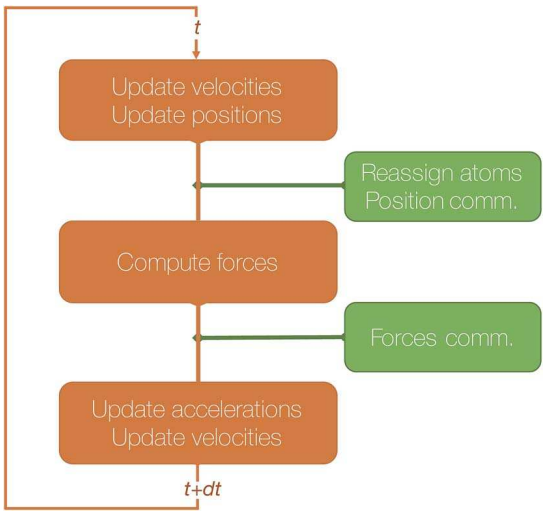


Fig. 3 Schematic representation of a velocity Verlet step.

2.3 Adaptation of the 3D spatial decomposition to point dipole polarizable force fields

In this section, we will explain how the global concepts of 3D spatial decomposition can be adapted to the special case of the computation of the polarization energy and forces in PFFs. To our knowledge this is the first functional production implementation of such a technique in that context. Indeed, some of us proposed recently a 1D spatial decomposition[40] implementation for AMOEBA. Here we propose a full extension to a 3D spatial decomposition to benefit from further performance enhancements. We will limit ourselves to the induced dipole model that is used in AMOEBA and that is the one implemented in Tinker-HP but the methodology is general and can be applied to various types of point dipole models.

The computation of the polarization energy in PFFs using the induced dipole formulation consists of two steps. First, a set of $3N$ (N being the number of polarizable sites) induced dipoles has to be computed by minimizing the functional

$$E_{\mathrm{pol}} = \frac{1}{2}\boldsymbol{\mu}^T \boldsymbol{T}\boldsymbol{\mu} - \boldsymbol{\mu}^T \boldsymbol{E},$$

where $\boldsymbol{E}$ is a $3N$ vector representing the electric field produced by the permanent density of charge at the polarizable sites. This is equivalent to solving the $3N \times 3N$ linear system

$$\boldsymbol{T}\boldsymbol{\mu} = \boldsymbol{E}, \quad (1)$$

where $\boldsymbol{T}$ is the polarization matrix. A detailed analysis of the polarization matrix and of the iterative methods that can be used to efficiently solve the linear system in eqn (1) can be found in ref. 41. Tinker-HP relies on Krylov approaches such as the Preconditioned Conjugate Gradient (PCG) and the Jacobi/Direct Inversion of the Iterative Subspace (JI/DIIS) algorithms. Their scalability and robustness have been discussed in previous works.[40,41] Additionally, we recently introduced a powerful non-iterative Krylov solver with analytical derivatives named the Truncated Conjugate Gradient[42,43] (TCG). Such a method has the same scalability as PCG but offers a reduced

cost with conserved precision as it does not suffer from the typical drift observed in polarizable MD scheme based on iterative techniques. For all these iterative methods, the building blocks are matrix-vector products and scalar products. Focusing on the short-range, direct space part of the computation, each matrix vector product (MVP) is analogous to a force computation (as described in the previous section). Indeed, each MVP is analogous to computing a set of electric fields due to a set of dipoles so that in the context of a parallel MD with 3D spatial decomposition, communications of the "neighboring" dipoles are mandatory before each matrix-vector product: this is equivalent to the "position comm" step previously described. Since Newton's third law is used, symmetrical communications of some electric fields created by the local dipoles have to be communicated after the matrix-vector product computation: this is equivalent to the "forces comm" described above. The scalar products require a global reduction and are naturally distributed among the processors independently of the parallelization strategy.

The computation of the induced dipoles by iterative methods represents not only an important additional computational cost, but also an important communication cost, as at each iteration two of the three communication steps described in Section 2 are required.

An essential part of our parallelization strategy is masking communication by computation in the parallel solvers whenever possible. This is achieved by using non-blocking MPI routines and by starting the receptions and the sendings of data as soon as possible, and, at the same time, verifying that the communications are finished as late as possible in the code, so that computations are being made between these two states. A schematic representation of a typical iteration of a polarization solver is shown in Fig. 4.

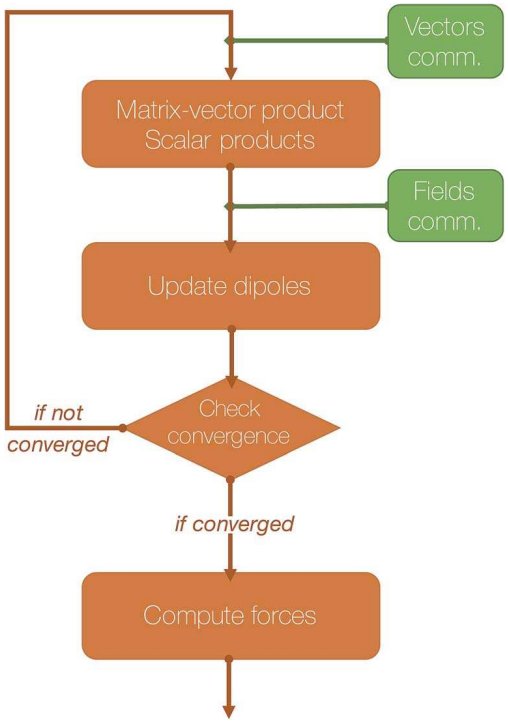


Fig. 4 Schematic representation of an iteration of a polarization solver.

3 Parallel algorithm for point dipoles using smooth particle mesh Ewald

We present here new developments concerning the use of SPME (Smooth Particle Mesh Ewald) using distributed multipole electrostatics and polarizable point dipole models. Building on our previous work[40] where we proposed a 1D decomposition of the distributed SPME grids, we now extend this formalism to the use of 2D pencil decomposition. Such an approach offers strongly improved performances especially when coupled to efficient iterative and non-iterative Krylov polarization solvers. In the previous section we focused the discussion on the parallelization strategy for short-range interactions. These include the bonded and van der Waals interactions, as well as the short range portion of the electrostatic and polarization interactions. The long-range part of such interactions needs to be handled separately, with a strategy that depends on the boundary conditions used for the simulation. Two main strategies exist in this regard: explicit solvent in periodic boundary conditions (PBC) and implicit solvation models. In this section, we focus on PBC. The additional possibility offered by Tinker-HP of treating the boundary with a polarizable continuum solvation model, namely, the Conductor-like Screening Model[44–46] (COSMO), is presented in Section 6.

As we stated before, the method that we adopt for PBC is the Smooth Particle-Mesh Ewald[47] (SPME). It has become a standard algorithm in modern biomolecular simulations to compute electrostatic interactions in periodic boundary conditions, thanks to its advantageous $\mathcal{O}(N \log N)$ scaling. The method has been extended to PFFs[48] as well as to multipolar interactions,[49] possibly including penetration effects.[50]

Let us explain the steps that are followed during a SPME computation for the electrostatic potential produced by distributed multipoles. The exact same considerations apply to the computation of the electrostatic and polarization forces and during a MVP computation during the iterative solution of the polarization equations. The electrostatic interactions are divided into two parts, one of which is short-range and is treated in the direct space, while the other is long-range and is treated in Fourier space. For the first, short-range part, the consideration made in Section 2 apply: we focus here on the reciprocal space computation. Such a computation requires the definition of a 3D grid and the use of Fast Fourier Transforms, which requires a significantly different parallelization strategy. The most standard one uses a 1D or 2D decomposition of the 3D grid and has been described elsewhere[12,40] in detail. Let us summarize its main steps and analyze the parallelization strategy employed in Tinker-HP.

The SPME computation requires to distribute the multipoles on the grid using a B-spline interpolation and then to solve Poisson's equation in the reciprocal space. The distribution of the 3D grid is therefore what drives the parallelization strategy.

In Tinker-HP, the grid is decomposed into 2D pencils, and each pencil is assigned to a processor. The first step of SPME consists into assigning charges or higher order multipoles to the grid-points. As explained in our previous work,[40] this operation requires local communications between processors assigned to neighboring portions of the grid.

The second step consist into switching to Fourier space by applying a forward FFT to the grid that has just been filled. In Tinker-HP, this is entirely handled by the 2DECOMP&FFT library.[51,52]

Then, the convolution with a pair potential[47] is done in Fourier space, which is a simple multiplication that is naturally distributed among the processors without any necessary communication.

Finally, the result of this multiplication is transformed back to real space by applying a backward FFT, which is also taken care of by 2DECOMP&FFT in Tinker-HP.

A final local set of communications between processors responsible for neighboring portions of the grid is done, followed by local multiplication with B-splines. A schematic representation of these steps is shown in Fig. 5.

Naturally, because the Fourier space decomposition of the grid may not fit exactly the 3D spatial decomposition, additional communications of positions are required before starting the reciprocal part of a SPME computation. Furthermore, when electrostatic or polarization forces are computed in this way, or after a matrix-vector multiplication in an iteration of a polarization solver, communication of some of these forces or dipoles are required.

Lagardère *et al.* showed[40] that the reciprocal part of SPME presented just above does not scale as well as the direct part with the number of processors, because of the relatively poor parallel scaling of the FFTs. Furthermore, because reciprocal space and direct space computations are independent and because reciprocal space is usually computationally cheaper, a usual strategy is to assign a smaller group of processors to reciprocal space and the rest to the direct space. This strategy can be used in Tinker-HP for both permanent electrostatics and polarization.

In that case, a difficulty arises in PFF computations. The load balancing between direct and reciprocal space computations is in fact essential to achieve a good scalability. However, the relative cost of direct and reciprocal computations is different for permanent electrostatics and MVP required for the computation of the induced dipoles. At this moment, only heuristic strategies have been implemented in Tinker-HP to handle this problem.

4 Software possibilities

Tinker-HP is part of the Tinker 8 package and consequently it is fully compatible with the canonical Tinker and the Tinker-OpenMM (GPU) codes. Therefore, all Tinker's analysis and visualization tools are available with Tinker-HP. Details about these possibilities are not described here and can be accessed on the Tinker community website (http://tinkertools.org). The Tinker-HP source code is freely available to the academic community: details and downloading informations can be found on the Tinker-HP website (http://www.ip2ct.upmc.fr/tinkerHP). In the following section, we detail the possibilities of the code that will be contained in the incoming public releases.

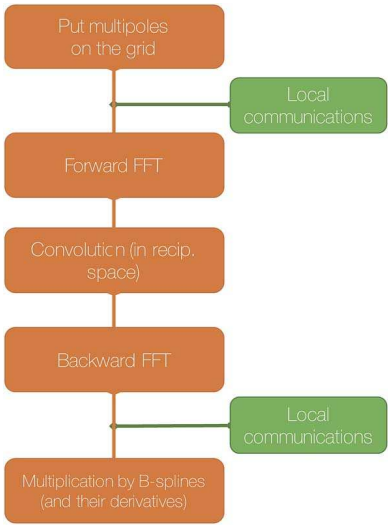


Fig. 5 Schematic representation of the computation of the reciprocal part of the electrostatic energy and forces with SPME.

4.1 Polarizable molecular dynamics engine features

List builder. As we stated in the method section, Tinker-HP is designed to be used on all types of CPU-based computer systems ranging from desktop computer to supercomputers. To do so, the package embodies a fast O(N) massively parallel list builder that is designed for both an extensive use of a large number of cores and to enable also an efficient treatment on a small number of cores.

Polarization solvers. Massively parallel implementation of various polarization Krylov solvers are present and includes iterative methods such as PCG, JI/DIIS. Both approaches can be used in connection with Kolafaś Alway Stable Predictor (ASPC)[53] that reduces significantly the iteration numbers for 1 fs and 2 fs timesteps simulations (see ref. 43 for discussion). An efficient non-iterative/fixed cost approach is also available: the Truncated Conjugate Gradient (TCG). TCG is implemented at the TCG1 and TCG2 levels with various refinements.[42,43] The TCG approaches are a strong asset of Tinker-HP as they accurately reproduce energy surfaces at a reduced computational cost and provide analytical forces. Such an approach avoids numerical drifts appearing with iterative methods and therefore brings enhanced energy conservation for long simulations. It is also fully time-reversible and compatible with the use of larger timesteps.

It is important to point out that an important choice in the Tinker-HP strategy is to keep accuracy to the maximum by retaining a double-precision approach. By definition, GPUs have the strong advantage of using mixed precision which has been shown to produce more stability than simple precision computations. The strategy here is to build on the availability of

the double precision to use algorithms that should/could not be used in mixed precision but are expected to be fully operational and faster in our case. For example, any CG methods are sensitive to precision (the symmetry of matrices being lost) as is the case for predictor-correctors such as the ASPC. Tinker-HP offers a full use of these strategies and compensates for the extra computational cost of double precision by more dependable algorithms.

Integrators. Most of the integrators available in Tinker have been implemented including, namely velocity Verlet, Beeman and RESPA[54] which allows production MD simulations with 2 fs time steps, and 3 fs timesteps using H mass repartitioning.

Simulation ensembles and associated tools. NVE, NVT and NPT simulations are possible. Bussi and Berendsen thermostats are available. NPT simulations are also implemented with a Berendsen barostat.

Restraints and soft cores van der Waals. Position, distance, angle, torsions and centroid based harmonic restraints as well as softcore van der Waals and scaled electrostatics for free energy calculations are available.

Geometry optimization. To prepare large systems encompassing millions of atoms through geometry optimization, Tinker-HP offers a massively parallel version of Tinker's limited memory BFGS quasi-newton nonlinear optimization routine (LBFGS).

4.2 Available force fields

Advanced point dipole polarizable force fields. Tinker-HP allows for electrostatics to range from point charges to fully distributed multipoles (up to quadrupoles), point dipole polarization approaches using distributed polarizabilities[41] coupled to Thole (or dual Thole) damping approaches as well as van der Waals interactions using the Lennard-Jones or the Halgren functions. This choice was motivated as these functional forms have been extensively used by various research groups that could therefore easily use Tinker-HP with their own parametrizations. Presently, two polarizable force field models, both relying on the Thole/point dipole polarization model, are available. The first model is the AMBER f99 polarizable model. It is limited to point charges to compute the permanent electrostatics and uses a 6–12 Lennard Jones for the van der Waals.[20,55] The second is the AMOEBA polarizable model which has been shown to have a wide applicability for systems ranging from liquids to metals ions, including heavy ones, in solution and to proteins and to DNA/RNA.[24,36,37,56–58] A major difference compared to the AMBER model is the replacement of the fixed partial charge model with polarizable distributed atomic multipoles till quadrupoles moments, allowing accurate reproduction of molecular electrostatic potentials, and higher resolution rendering of difficult directional effects in hydrogen bonding and other interactions. van der Waals interactions are also different and use the Halgren buffered 14–7 function.[59] The AMOEBA polarizable model responds to changing or heterogeneous molecular environments and its parameterization was performed against gas phase experimental data and high-level quantum mechanical results. The AMOEBA model includes high accuracy water model as well as parametrization for organic molecules, proteins,[60] ions and DNA/RNA complexes.

Classical force fields. By construction, the software is able to perform classical force field simulations following the canonical Tinker initial implementation of the AMBER, CHARMM and OPLS potentials. Such force fields also benefit from the speed up of the massively parallel framework but our objective is to reach comparable performance to the AMBER and CHARMM (Domdec[61]) CPU implementations. The detailed analysis of such code capabilities being beyond the scope of this paper, fully dedicated to polarizable models, and it will be discussed elsewhere. However, it can be noted that classical MM that requires much less work than PFFs allows for a 5–8 acceleration of the production per day over AMOEBA (depending on the use of TCG *vs.* PCG solvers) on the same computational platform, and will be used for hybrid simulations with PFFs coupled to non-polarizable parts of the system. For higher performances using Tinker, one could use the Tinker-OpenMM access to the OpenMM library implementation of such classical FF. For example, it is possible to produce 305 ns per day for DHFR with the same GTX 1080 card (mixed precision) and settings used in this work using the AMBER force field.

5 Benchmarks and applications using the AMOEBA polarizable force field

The present implementation has been extensively tested and reaches exactly the same accuracy as the canonical Tinker for polarizable force field when considering analogous algorithms, allowing Tinker-HP to produce reference computations. All the proposed benchmarks use the AMOEBA force field. We tested the performances of Tinker-HP on various systems. We studied the scalability of the code dealing with homogeneous systems such as bulk water, and inhomogeneous systems ranging from ionic liquids to proteins. Finally we tested our approach on very large biosystems.

5.1 Computer platforms

All tests have been performed on the Occigen machine at GENCI (CINES, Montpellier, France) and at CYFRONET (Krakow, Poland) on the Prometheus machine. Occigen is a Bullx DLC with Intel Xeon E5-2690 v3 (24 Haswell cores at 2.6 GHz per node) and Intel Xeon E5-2690 v4 (28 Broadwell cores at 2.6 GHz per node), Infiniband FDR and 128 Go of memory per node. Prometheus is a HP Apollo 8000 with Intel Xeon E5-2680 v3 (24 Haswell cores at 2.5 GHz per node), Infiniband and 128 Gb of memory per node. For consistency, all results are given for Haswell processors. We observed an average four per cent gain in speed on the Broadwell configuration, especially for a suboptimal number of cores, *i.e.* before the scaling limit. Some timings have been obtained using Tinker-OpenMM on GPU cards (NVIDIA GTX 970 and GTX 1080), the best GPU results (GTX 1080) can be found in Table 3 below, the GTX 970 productions being roughly half of the GTX 1080 ones.

5.2 Simulations setup

Benchmark simulations (except free energies) were made in the NVT ensemble with a Verlet/RESPA multi-time step integrator with either a 2 fs or a 3 fs time-step (using hydrogen mass repartitioning in the latter case) for the non-bonded forces and half of this value for the bonded forces. Two Krylov solvers were considered here: iterative PCG and non-iterative TPCG, both using a diagonal preconditioner.[41,42] Note that we report here the first results ever using TCG coupled to SPME. The convergence criterion for the PCG iterative solver was set to 10^{-5} D. Electrostatics and polarization interactions were treated using the PME algorithm with a real space Ewald cutoff of 7.0 Å. The van der Waals cutoff was set 9.0 Å without any long-range correction.

5.3 Homogeneous systems: water boxes and ionic liquids

Water boxes. We first benchmarked the code on cubic water boxes of increasing size: from 96 000 atoms up to 23.3 millions atoms. Table 1 summarizes the characteristics of these boxes: their size in Angstroms, the number of atoms they contain, the size of the associated PME grid and the name with which they will be referenced in the rest of the paper.

Fig. 6 show the detailed scalability up to almost 1 million atoms.

A very good scalability is observed in the three cases. Table 3 displays the best production timings in ns per day. The code appears to be competitive with the GPU numbers extracted from Tinker-OpenMM even for a system such as the smallest water-box test (Puddle, 96 000 atoms). In this case, Tinker-HP is already 1.5 faster than a GTX 1080 card (3 times for a GTX 970) but with double precision compared to mixed precision arithmetics used by GPUS. As we will discuss later in the case of proteins, the newly introduced 3D domain decomposition algorithmic for polarizable FF becomes more beneficial when the size of the system grows and a first advantage of Tinker-HP is to be able to use the distributed memory system of the CPU platform. Also for such large systems numerical instabilities of the polarization solvers that result in energy drifts[40–43] are a key error that must be contained. Double precision is highly preferable when one wants to use advanced conjugate gradient solvers (and Krylov approaches in general). Tinker-HP has an advantage as it affords mathematically robust solutions for "drift-free" polarization solvers (Truncated Conjugate Gradient, TCG[42,43]) with analytic forces. Such techniques allow for (very) long simulations. A stable adaptation of these methods to mixed precision hardware (*i.e.* GPUs) is underway but is mathematically non-trivial. Note that for short to medium simulations of a few dozen ns, the discussion is without object as the drifting issue will remain negligeable offering a full applicability of GPUs acceleration. However, towards and beyond the microsecond, the analytical forces polarization solvers will be key for stable polarizable simulations. For the other benchmark cases, the speedup increases to a 5 and 6-fold over a GTX970 (2 and 3-fold over a GTX1080) for 288 000 atoms (Pond) and 864 000 atoms (Lake) water boxes respectively. For the Lake box, a detailed analysis of the scaling against ideal scaling is provided in ESI S2.† We then pushed the code towards its limits by testing very large systems including 7 776 000 and 23 300 000 atoms respectively. At these levels, GPUs have memory limitations that makes such simulations impossible, which is not the case with supercomputers relying on distributed memory. These "computational experiments" took place on the Prometheus supercomputer (CYFRONET, Krakow, Poland) and enabled us to test for the validity of the code on a very large scale. Results show that Tinker-HP is still operational beyond 20 million atoms. Of course, the production really slows down to a few dozen ps per day but the performance is noticeable as it would be quite enough to compute properties such as electrostatic potentials or even a short ns-scale molecular dynamics. Thus, one can expect, depending on the machine used, to produce a ns in a few weeks on the largest Ocean water box using TCG2/RESPA (3 fs). It is worth noticing that the largest computation was limited only by the computer system availability and that presently larger systems are potentially accessible with more computational resources. However, such very large computations require a different setup than the others due to memory limitations and communication issues. Indeed, for such a large number of atoms, FFTs really become severely time limiting and intra-node communications strongly affect performances. One solution that was used for Ocean was to only use a fraction of the cores of a node to take advantage of the node memory without suffering from excessive communications. That way, if the Ocean test ran on 12 288 cores on 512 nodes, we used only 6 cores/node (on 24) to actually perform the computation. This gave us the possibility to better use the bandwidth of the interconnect cards (by reducing contention in MPI transfers between cores and cards), a strategy that compensates for the lack of active cores and that can be used for any system size. We used the same strategy to a lower extent for Sea as 17 cores out of 24 were active. Overall, a rough estimate for the fastest Broadwell CPU configuration (Occigen) is that using a RESPA (3 fs)/TCG2 setup, a routine production of 1 ns per day is reachable for a million atoms. Such a value is a combination of various hardware setups that are not only dependent on the CPU speed (and numbers), as the interconnection cards have a strong influence on the final results (Fig. 7).

Table 1 Water boxes used for benchmark purposes

System	Puddle	Pond	Lake	Sea	Ocean
Number of atoms	96 000	288 000	8 640 000	7 776 000	23 328 000
Size (of an edge) in Angstroms	98.5	145	205.19	426.82	615.57
Size (of an edge) of the PME grid	120	144	250	432	648

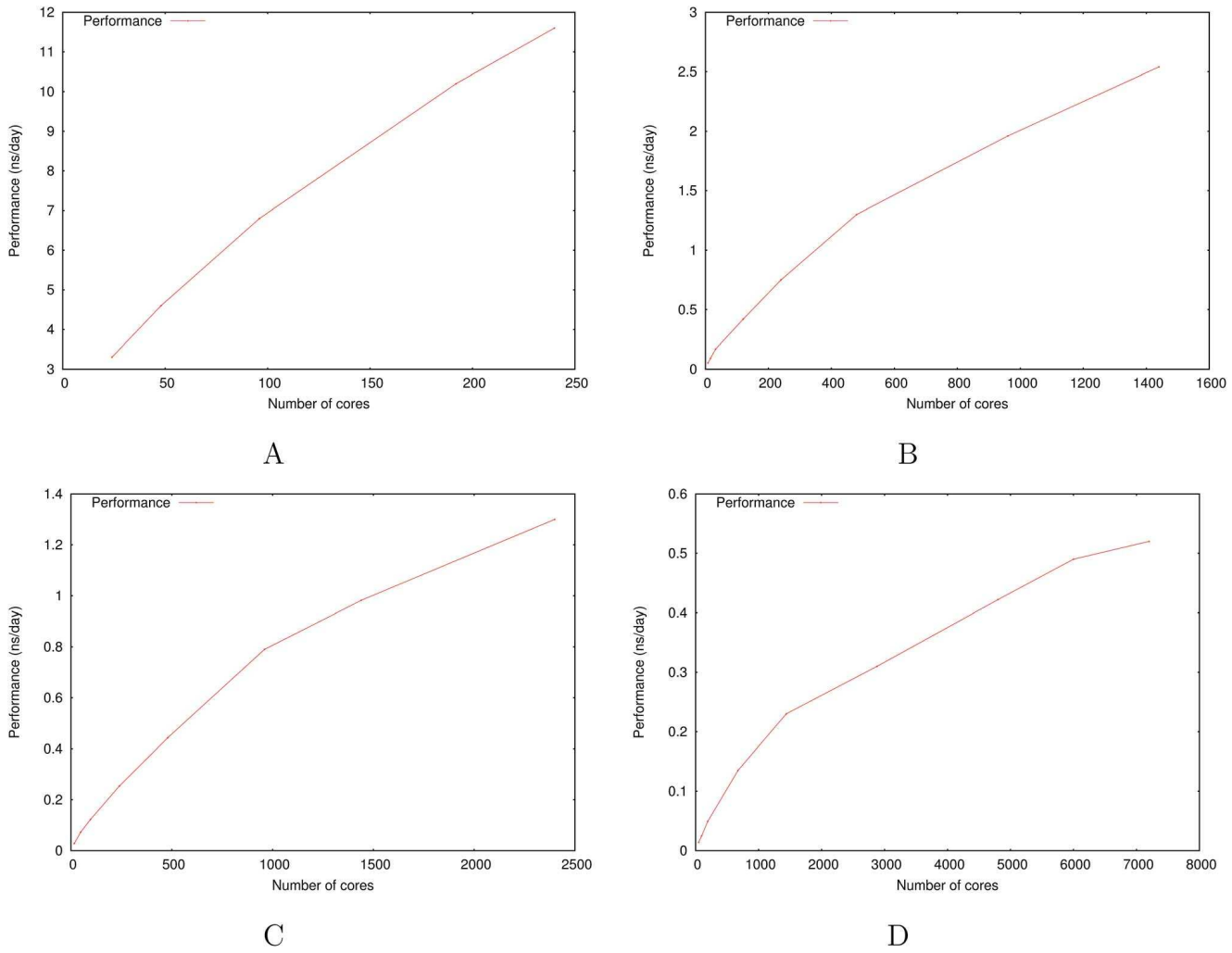


Fig. 6 Performance gain for the [dmim$^+$][cl$^-$] ionic liquid system (A) and the Puddle (B), Pond (C) and Lake (D) water boxes.

Ionic liquids. Room temperature ionic liquids (ILs) are molten salts at room temperature that are formed by the combination of organic or inorganic cations with (generally) inorganic anions. These compounds exhibit a wide variety of useful properties that led to their use in numerous applications.[62–65] The unusual properties observed in ILs arise from the inter- and intra-molecular interactions of the constituent ions. Thus, the computational simulations of these systems greatly benefit from the use of highly accurate potentials. Recently AMOEBA parameters for several ILs have been developed and applied for various systems.[66–68] It is known that polarization effects result in better reproduction of transport properties.[69–72] In addition, ILs are viscous fluids and it is thus necessary to perform relatively long MD simulations. Therefore, Tinker-HP is an ideal platform for these systems given its HPC capabilities and implementation of significantly more accurate and efficient algorithms for the evaluation of the polarization component. Indeed, ILs usually require a lot more iterations than standard molecules with standard solvers such as JOR (Jacobi Over Relaxation, see ref. 41), which is not the case with Krylov solvers such as PCG or TCG, with which such systems have been tested.[42]

As a first example, simulations were performed for 1,3-dimethylimidazolium imidazolium/chloride ([dmim$^+$][cl$^-$]) for 200 ns using the parameters reported by Starovoytov *et al.*[66] The results calculated with Tinker-HP are in very good agreement with the previously reported results, with the added advantage that Tinker-HP provides excellent scaling, with production runs for a system of 216 ion pairs (in a cubic box of 35.275 Å, a PME grid of $48 \times 48 \times 48$ and a 7 Å real space cutoff) of more than 11.5 ns per day on 240 cores. Therefore, Tinker-HP enables simulations of IL systems in the hundreds of ns up to μs timescales.

5.4 Speeding up free-energy computations: assessing large water box hydration free energies computations

The observed speed-up on water boxes led us to test the performance AND the accuracy of free energy computations using large water boxes to compare them to initial works using AMOEBA and the canonical Tinker software. The hydration free energies for water, benzene, K^+ and Na^+ were calculated by summing up the free energies of three thermodynamic steps, solute discharging in a vacuum, solute van der Waals coupling with solvent, and solute recharging in solvent. For K^+ and Na^+,

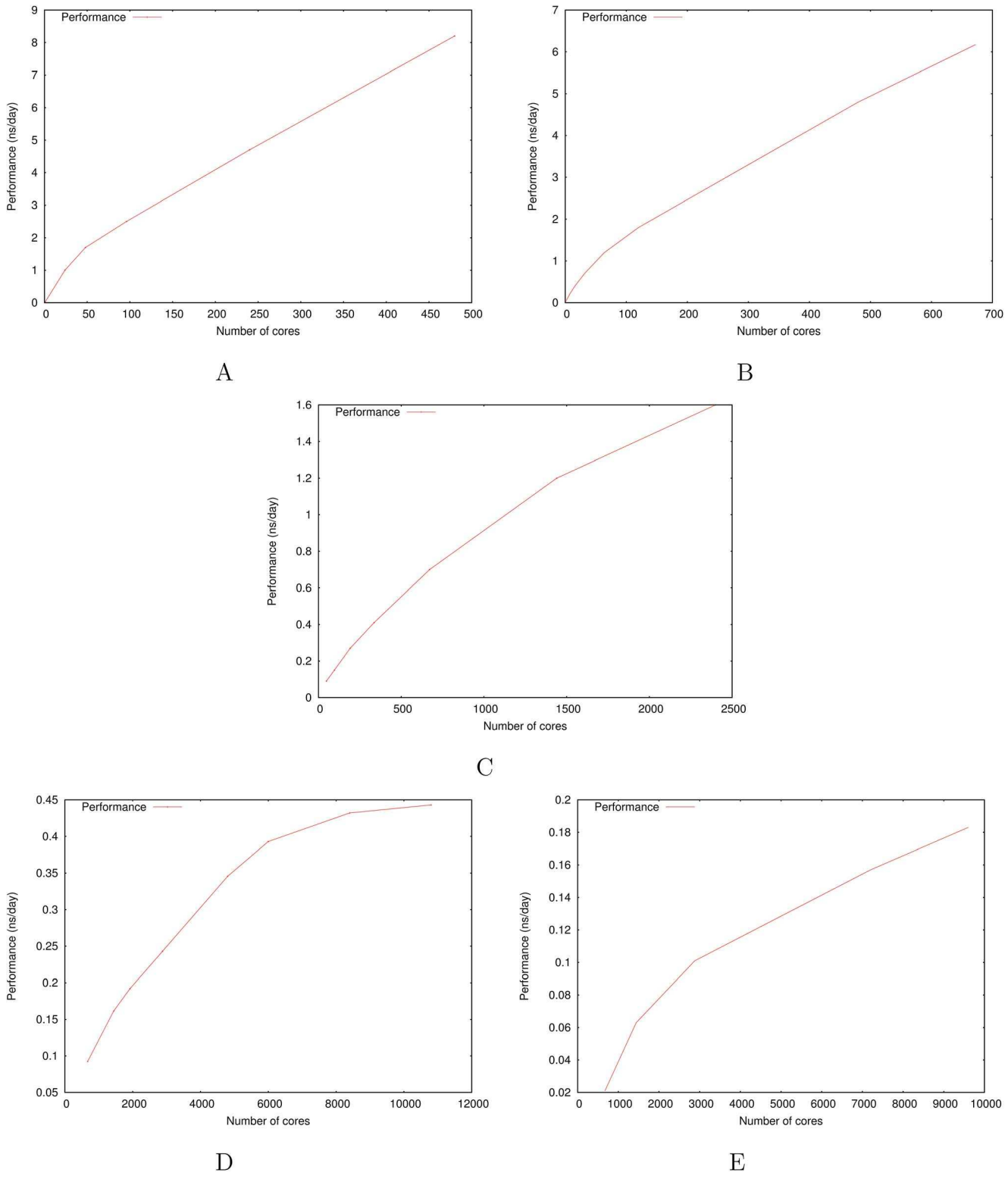


Fig. 7 Performance gain for the ubiquitin protein in water (A), the dihydrofolate reductase protein (dhfr) in water (B), the COX-2 system in water (C), the satellite tobacco mosaic virus in water (D) and the ribosome in water (E).

since the standard state in simulation was 1 mol L^{-1} and the standard state in experiment was 1 atom, the free energy difference between the two states of 1.87 kcal mol^{-1} was added to the final results. The softcore van der Waals potential was used as in our latest work with Tinker. A total of 21 alchemical states were considered, and a 2 ns NVT simulation was performed at each state. The RESPA (2 fs) integrator was employed as the temperature was maintained at 298 K by the Bussi thermostat. The vdW interaction was truncated at 12.0 Å as SPME used a real-space cutoff of 8.0 Å and a 72 × 72 × 72 grid. The

Bennet Acceptance Ratio (BAR)[73] method was used to extract the free energies between states. In order to test the computation efficiency, the solute molecule was immersed in a large cubic simulation box of 6000 water molecules. The length of the box was 56 Å and 192 cores were used for each simulation with 48 dedicated cores for PME. This number of core is suboptimal but already provides a very good speedup as all windows were launched simultaneously on a total of 4032 cores, a computer resource that is commonly accessible in modern computer centers. Each total free energy evaluation took 18 hours to complete using a PCG coupled to Kolafa's predictor-corrector (ASPC) algorithm with a 10^{-5} convergence threshold. The hydration free energies for water, benzene, sodium and potassium are listed in the table of the ESI S1,† together with results from previous work. For all four solute molecules, there is excellent agreement between Tinker-HP and previous simulations using either BAR or OSRW (Orthogonal Space Random Walk) method.[74] The values converge at 2 ns with a statistical error of around 0.1 kcal mol^{-1}. The hydration free energies for potassium obtained from Tinker-HP and the Tinker published results are slightly different because the Tinker historical work did not use the softcore van der Waals potential at that time, but appears fully consistent with the present canonical Tinker result. Overall, Tinker-HP appears reliable and very efficient for the calculation of solvation free energies with huge gain in terms of computation time. Of course, further tests on more complex free energy computations are required to test all the possible combinations of TCG and RESPA algorithms. If TCG2 is really accurate and fast, TCG1 is significantly faster but these procedures have not been extensively tested yet and their evaluation concerning their applicability to free energy computations will be the subject of a larger study. In any case, TCG2 would lead to a computing time reduction of the same computations to roughly 14.5 hours and TCG1 to 12.5 hours. Such studies will benefit from the computational platform introduced in Tinker-OpenMM that allows computing absolute binding and relative alchemical approach as well as relative binding affinities of ligands to the same host. As an immediate other perspective, the OSRW results extracted from the canonical Tinker are presented in the table. This approach leads to very similar results to the BAR approach but requires up to 5 times less computer time. OSRW is currently under implementation in Tinker-HP. These results give an idea about the new possibilities offered by massive parallelism for free energies evaluations: the discussed simulations that initially took months are now possible within half a day and soon in a couple of hours with OSRW within Tinker-HP.

5.5 From proteins to realistic biosystems

To study the scalability and applicability of the Tinker-HP platform to complex non homogeneous systems, we tested various systems starting from the "small" ubiquitin protein (9737 atoms), and prototypic dihydrofolate reductase (dhfr, 23 558 atoms) which is the reference protein test case extracted from the joint AMBER/CHARMM benchmark (http://ambermd.org/amber10.bench1.html). We push the code towards the simulation of very large biosystems tackling the COX-2 dimer, the Satellite Tobacco Mosaic Virus (STMV) and the ribosome full structures in polarizable water. All timings are obtained for equilibrated systems.

The characteristics of the inhomogeneous systems simulations boxes used for benchmark are summed up in Table 2.

Small proteins: ubiquitin and DHFR. We started our study by testing Tinker-HP on small proteins were 3D domain decomposition is expected not be fully efficient (our water boxe study started at 96 000 atoms, which is 4 times the size of DHFR and 10 times that of Ubiquitin). Surprisingly, results remain competitive with GPUs which are fully taking advantage of their computing power for such a range of systems with low memory requirements. DHFR allows to study in depth the code behavior in that system size range. Indeed, the best production time for a use of all cores of a node brings us to a 7.69 ns per day using TCG2. This production time is really close to the 8.29 ns per day exhibited by Tinker-OpenMM on a GTX1080 (see Table 3). If we used the same number of cores distributed on more nodes, to use the same technique we used on the large ocean and sea water boxes, the performance extends to 8.79 ns per day. These numbers make Tinker-HP very competitive for these small systems on a reasonable number of cores that is easily accessible on modern supercomputers. In addition, one can note that most of the recent machines use Broadwell Xeon that gives slightly better performances by a few percents. In other words, Tinker-HP is able to compensate for the computational burden of the use of double precision thanks to its new algorithmics compared to the accelerated mixed precision GPUs thus reaching both speed and accuracy. A detailed analysis of the DHFR scaling against ideal scaling is provided in ESI S2.† As one could expect, the deviation to the ideal scaling is higher than in the case of the previously larger Lake water box: larger the system is, closer to the ideal scaling we get.

Larger systems: COX-2, STMV and ribosome solvated in water. For larger systems, as it was shown for the water boxes, the 3D domain decomposition speedup is taking full effect and the distributed memory approach offers an access to systems that were up to now restricted to classical non-polarizable force fields implemented in HPC packages. The benchmarks of Table 3 show that the discussion is fully transferable to non-homogeneous systems as realistic simulation times on a reasonable number of cores are reachable for the COX-2, STMV and ribosome systems allowing for meaningful simulations. The table displays a test for the COX-2 dimer (part of the Tinker benchmark suite, see https://dasher.wustl.edu/tinker/distribution/bench/) for which 1.6 ns per day are possible on 2400 cores, a computer resource that is easily accessible in supercomputer centers. If one wants to push the performances, one ns simulation can be achieved in a little more than a day on the STMV structure (taken from the NAMD website: http://www.ks.uiuc.edu/Research/namd/) which is not accessible to our GPU implementation due to memory requirements. Such a result is really extremely promising, considering that STMV encompasses more than a million atoms within the full virus structure including its full genetic materials, the whole system being fully solvated in water. Such simulations are indeed

Table 2 Biosystems used for benchmark purposes

Systems	Ubiquitin	Dhfr	COX-2	STMV	Ribosome
Number of atoms	9732	23 558	174 219	1 066 228	3 484 755
Size (of an edge) in Angstroms	54.99 × 41.91 × 41.91	62.23	120	223	327.1
Size (of an edge) of the PME grid	72 × 54 × 54	64	128	270	360

Table 3 Best production time (ns per day) for the different test systems (AMOEBA force field) using various methods. Number of atoms and optimal number of cores are given for each systems. All timings are given for Intel Haswell processors. Reference canonical Tinker CPU times are given for Open-MP computations using 8 cores. All computations were performed using a RESPA (2 fs) integrator if not specified otherwise. ASPC = Always Stable Predictor Corrector.[53] N.A. = Non Applicable due to memory limitations. GPU production times were obtained using the Tinker-OpenMM software[34] (CUDA 7.5), the JI/DIIS solver and a GTX 1080 NVIDIA card

Systems	Ubiquitin	DHFR	COX-2	STMV	Ribosome
Number of atoms	9737	23 558	174 219	1 066 628	3 484 755
Tinker-HP number of CPU cores	480	680(960)	2400	10 800	10 800
PCG (10^{-5} D, ASPC)	8.4	6.3(7.2)	1.6	0.45	0.18
TPCG2	10.42	7.81(8.93)	1.98	0.56	0.22
TPCG2/RESPA (3 fs)	15.62	11.71(13.39)	2.98	0.84	0.34
CPU OPEN-MP	0.43	0.21	0.024	0.0007	N.A.
GPU (GTX 1080)	10.97	7.85	1.15	N.A.	N.A.
Systems (water boxes)	Puddle	Pond	Lake	Sea	Ocean
Number of atoms	96 000	288 000	864 000	7 776 000	23.3×10^6
Tinker-HP number of CPU cores	1440	2400	7200	7104	12 288
PCG (10^{-5} D, ASPC)	2.54	1.3	0.52	0.062	0.0077
TPCG2	3.10	1.59	0.63	0.076	0.01
TPCG2/RESPA (3 fs)	4.65	2.38	0.95	0.11	0.014
CPU OPEN-MP	0.050	0.014	0.003	N.A.	N.A.
GPU (GTX 1080)	2.06	0.80	0.21	N.A.	N.A.

relatively recent even for classical force fields as the Schulten group only produced the first studies 10 years ago.[75] The present extension of the simulation capabilities to advanced multipolar polarizable force fields opens new routes to the understanding of complex biosystems. Indeed, as we have seen, Tinker-HP is able to go far beyond the million atom scale and studies on the ribosome become possible following early studies (see ref. 76 and references therein). We built a model for benchmark purposes for the 70 s ribosome from *Thermus thermophilus* containing nearly 5000 nucleotides and over 20 proteins, with over 4100 sodium ions to neutralize the nucleic acid, and about a million water molecules for a total of 3 484 755 atoms. Presently, three days are necessary to produce a ns allowing for a very detailed study of such an important structure. We expect even free energy studies to be feasible. Various incoming studies will analyze more in-depth the use of PFFs to such mostly important biosystems.

6 Beyond classical MD simulations in periodic boundary conditions

So far, we have presented the capabilities of Tinker HP in the context of PBC classical molecular dynamics simulations. We have discussed the parallelization strategy and showed benchmark results that demonstrate the scalability and performances of the code. While Tinker-HP is mainly a molecular dynamics code, it is not limited to PBC classical simulations and can be used for different applications. In particular, Tinker-HP offers the possibility of performing non-periodic MD simulation with a polarizable force field such as AMOEBA using a polarizable continuum solvation model as a boundary. This possibility is not our main choice for MD simulation and, as a consequence, has not been as thoroughly optimized as the PBC code. Furthermore, it involves a few computational steps that scale quadratically with respect to the size of the system, making it not suitable for the very large systems presented in Section 5. However, the possibility of computing the energy and forces with non-periodic boundary conditions and with a continuum boundary opens the way for using Tinker-HP as a module to handle the classical part in a polarizable QM/MM(/continuum) calculations,[77–81] including the computation of molecular properties and *ab initio* multiscale QM/MM MD simulations. These calculations are usually dominated in computational cost by the QM part, making the quadratic scaling of the classical part a minor issue. Nevertheless, the scalability of Tinker-HP paves the way to large-scale polarizable multiscale simulations.

In this section, we will describe the non-periodic code in Tinker-HP, based on the recently proposed ddCOSMO,[45,46,82,83]

a domain decomposition (dd) discretization of the conductor-like screening model.[44] We will then discuss two complementary QM/MM strategies that can be used to couple Tinker-HP to a quantum-mechanical code.

6.1 Implicit solvent: ddCOSMO

Continuum solvation models[84,85] (CSM) are a well-established technology in both quantum chemistry and MD. The CSM developed for MD are usually based on the Generalized Born (GB) Ansatz, or its multipolar generalization, which approximate the solution to the electrostatics equations in the presence of a continuum with an additive energy term. Methods developed in quantum chemistry rely, on the other hand, on a rigorous numerical solution of Poisson's equation. Such models are much more expensive than the GB counterpart; however, since these models have been developed for quantum mechanical calculations, and therefore for up to medium-sized systems, their computational cost is not a real limitation in QM calculations. Nevertheless, it has always prevented their application to MD simulations. The use of a polarizable CSM is of particular interest when a PFF is used due to the natural consistency between the two approaches. Recently, a new discretization to COSMO has been proposed. Such a new discretization, named ddCOSMO, has been developed when the molecular cavity is made of interlocking spheres (*i.e.*, van der Waals cavity) and has been extensively described elsewhere.[46] The dd approach offers huge advantages since the matrix to be inverted to solve the model at each time step is highly sparse: as a consequence, the model scales naturally linearly with the size of the system and the iterative solution to the ddCOSMO equations is perfectly suited for a parallel implementation in which the spheres that constitute the cavity are distributed among cores.

The parallelization strategy adopted for the ddCOSMO implementation follows the spatial decomposition logic discussed in Section 2. Again, we divide the space occupied by the system into blocks and assign a block to each CPU. The CPU is then responsible for updating the positions, speeds and accelerations of the atoms belonging to it block. However, there are two important differences compared to the spatial decomposition discussed for short-range interactions. First, the space occupied by the solute is not a cube or a regular geometrical configuration but rather a cavity whose shape depends on the configuration of the solute. Second, the cavity is not fixed during the simulation as it evolves with the solute.

To address the first issue, we define the blocks by enclosing the solute in the smallest parallelepiped containing it and we divide this parallelepiped into smaller ones. This strategy presents the advantage of allowing us to reuse the whole machinery that has been described in Section 2. However, such a strategy can imply potential load balancing issues that require to be addressed, especially when a high number of processors is used. Again, an iterative procedure has been implemented to determine the optimal sizes of the sub-domains.

To solve the second issue, one should in principle recompute the enclosing parallelepiped at each time step. To avoid the cost of performing such an operation, we build a slightly larger parallelepiped and recompute its geometry only once every few MD steps ($n = 20$ for example).

In Tinker-HP, the solution to the ddCOSMO linear equations is computed by using the JI/DIIS iterative solver also used for the polarization eqn (1). The iterative procedure requires to compute MVP with the sparse ddCOSMO matrix, which can be done both very efficiently and involving only local communications. However, the right-hand side of the ddCOSMO equations depends on the electrostatic potential created by the solute's permanent and induced multipoles. In the current implementation, the potential is computed *via* a double loop, which implies a $\mathcal{O}(N^2)$ computational cost. Furthermore, an "all to all" communication of the positions of the system is required prior to this computation.

Thus, the computational bottleneck in terms of both computational complexity and parallel efficiency lies in the computation of the right-hand side. If AMOEBA/ddCOSMO MD simulations have been shown to be possible,[46] this kind of boundary is not competitive with SPME in term of pure polarizable MD production. However, as we stated at the beginning of this section, the advantage of the ddCOSMO implementation is to provide a boundary condition for multiscale simulations. In particular, having non-periodic boundary conditions is ideal when working with localized basis functions in QM computations.

Detailed benchmark results of the current parallel implementation are presented in ESI S3.†

6.2 Multiscale modeling and polarizable QM/MM

The PFF/ddCOSMO framework described in this section is a starting point for multiscale, polarizable QM/MM simulations. This is a fundamental direction for Tinker-HP as PFFs such as AMOEBA provide a high-quality embedding strategy for QM systems with various potential applications. For instance, in a recent publication, some of us showed how a DFT-based QM/AMOEBA description is able to model electronic excitations in aqueous solution[80] for systems that interact in a specific and structured way with the environment. An *ab initio* QM/MM MD strategy has also been recently proposed.[81]

The present QM/MM possibilities of Tinker-HP follow two complementary strategies. Tinker-HP can be used as an external embedding tool, or can be directly coupled to a QM code in order to obtain a fully self-consistent polarizable QM/MM implementation.

The first strategy is the one followed in LICHEM[77] (Layered Interacting CHEmical Model), that provides a QM/MM interface with unmodified quantum chemistry software suites such as Gaussian,[86] PSI4,[87] and NWChem[88] to perform QM/MM calculations using the AMOEBA force field. This is done by approximating AMOEBA's multipolar distribution, with a set of point charges,[89] which can then be read by the QM code. This choice is motivated by the idea of developing an interface with existing QM codes with non-native multipolar QM/MM capabilities. LICHEM extracts forces and energies from unmodified QM packages to perform a variety of calculations for non-bonded and bonded QM/MM systems, the latter by using the

pseudobond formalism explicitly extended for QM/MM with PFFs.[77,90,91] The calculations available in LICHEM include geometry and reaction path optimizations, single-point energy calculations, Monte Carlo, PIMD, *etc.*

Currently, the polarization component for the QM/MM interaction term in LICHEM is not fully self-consistent due to the use of unmodified QM codes. This is because only the field from the permanent multipoles from the MM subsystem is included in the effective Hamiltonian for the polarization component of the QM/MM interaction. However, as has been shown previously, this approximation, coupled with the fact that the QM and MM subsystem polarization is fully considered results into the recovery of over 80% of the total QM/MM self-consistent polarization.[77,92]

For the computation of electronic properties and full hybrid MD simulations, a second QM/MM approach can be pursued. This approach proposes a fully self-consistent treatment of the electronic density and the MM polarization and requires a modification of the QM self-consistent field routines. A QM/AMOEBA implementation that couples Tinker-HP to a locally modified version of the Gaussian suite of programs[86] has been recently introduced.[80,81] Such a strategy enables to use a DFT/AMOEBA based polarizable QM/MM strategy to compute the energy and response properties of an embedded system, as well as to perform Born–Oppenheimer (BO) hybrid QM/MM MD. The latter is accelerated through the use of an extended BO Lagrangian approach (XL-BO),[93] which provides enhanced guess for the electronic density at every time step and allows for a stable hybrid MD with enhanced energy conservation.

In short, Tinker-HP offers additional advanced QM/MM functionalities with polarizable force fields. The continuous investigation efforts in our groups have the objective to bring sampling capabilities in a multiscale polarizable environment dedicated to electronic structure as sampling has been shown to be a key issue for predictive studies.[80]

7 Conclusion and perspectives

Our results demonstrate that molecular dynamics simulations with advanced point dipole polarizable force fields using distributed multipoles should no longer be qualified as slow anymore. The Tinker-HP software offers an efficient environment that enables one to perform large scale relatively long MD simulations on various complex systems encompassing several million atoms thanks to the new extension of 3D spatial decomposition to polarizable models coupled to advanced Krylov polarization solvers. It is able to ensure accuracy and speed as it exploits double precision, thanks to its new algorithmics able to circumvent the computational burden providing both additional speedups and mathematical robustness. For small systems, Tinker-HP is competitive with the present GPU implementation of Tinker (Tinker-OpenMM) whereas strong gains are observed for medium systems offering several thousand-fold acceleration compared to single core computations. For large systems, Tinker-HP remains the only operational Tinker code as it is able to efficiently distribute memory among nodes. We believe that this new tool will be of interest for the community of modelers, who will be able to perform meaningful simulations to test the applicability and discuss advantages of polarizable potentials. Of course, such developments will first find an echo in the field of chemistry where extreme accuracy matters, for example using embeddings of QM methods by PFFS that are beneficial to compute properties and where double precision is mandatory. For biophysics, where extreme sampling is required, the full application of PFFs remains a daunting task as present AMOEBA simulations, despite the discussed acceleration on large systems, still require weeks of computation. However, a few microseconds simulations are now technically possible and some applications such as free energy computations are completely accessible. In some way, PFFs are now able to produce simulations that classical force fields were able to generate a few years ago on similar platforms. The one-order of magnitude difference in speed of PFFs compared to classical FFs (when one considers the same computational platform, *i.e.* CPU or GPU), will remain due to the lower functional form complexity of the latter. However, the acceleration gains observed in optimal timings for codes like AMBER, NAMD, GROMACS or equivalent, are all obtained using GPU accelerators and through many years of optimization. Still, an important point to evaluate the future of PFF simulations is the fact that we have been really conservative in our present discussed benchmarks and optimization is only starting. Issues of precision, cutoffs, convergence criteria and vectorization will be addressed and will generate strongly improved performances. Note that the Tinker-HP methodology is not limited to CPUs. Indeed, the Tinker-HP FORTRAN legacy code will benefit from GPU acceleration as FORTRAN portability strategies exist and are under investigation (Hybrid-Fortran[94] and OpenACC[95]). For CPUs, we also expect strong performance gains on new generation "big core" Xeon (Skylake and successors) and "small core" Xeon-Phi (Knight Landings) processors thanks to vectorization efforts exploiting AVX512 instructions without sacrificing double precision. Finally, Tinker-HP will be synchronized with Tinker-OpenMM[34] opening our developments to the OpenMM community. Various method developments, already present in the Tinker community, will be integrated in the next version of the code, keeping in mind the mandatory philosophy to include only well-understood and scalable techniques. The high-performance implementation of additional multipolar polarizable force fields will be performed including the SIBFA[26] (in progress), MPID[96] (multipole and induced dipoles, the mapping of the CHARMM Drude polarizable force field on induced dipoles) and AMOEBA 2 models. Efforts will also be devoted to the porting of the third generation GEM (Gaussian Electrostatic Model) polarizable force field that relies on frozen distributed densities.[30,97,98] The present technology will be complemented by massively parallel Monte-Carlo approaches, Langevin, constant-pH and various types of accelerated molecular dynamics. Advanced sampling techniques such as OSRW[74] and replica exchange will be added. Concerning multiscale QM/MM simulations, studies towards coupling with linear scaling QM approaches will be pursued to continue to speed up hybrid MD simulations.

Conflicts of interest

There are no conflicts to declare.

Acknowledgements

This work was made possible thanks to the French state funds managed by the CalSimLab LABEX and the ANR within the Investissements d'Avenir program (reference ANR-11-IDEX-0004-02) and through support of the Direction Générale de l'Armement (DGA) Maîtrise NRBC of the French Ministry of Defense. Funding from French Centre National de la Recherche (CNRS) is also acknowledged (PICS international collaboration grant (Sorbonne U.-UT AUstin) and Infiniti fund), as well as support by the National Institutes of Health (R01GM106137, R01GM114237 and R01GM108583), CPRIT (RP160657) and the Robert A. Welch Foundation (F-1691). This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1610403. The authors are also indebted to the Texas Advanced Center (TACC, Austin, USA) and to the ISCD (Institut des Sciences du Calcul et des Données, Sorbonne U., France) for providing a development access for the initial design of this code. Production computations have been performed at GENCI (grant no. A0010707671) on the Occigen supercomputer (CINES, Montpellier, France). The authors are grateful to the CYFRONET computer center in Krakow (Poland) which provided help and resources to perform some of the large scale simulations of the paper on the Prometheus machine. LL, LHJ and JPP thank Cédric Bourasset and David Guibert (Centre for Excellence in Parallel Programming, ATOS-Bull) for fruitful discussions.

References

1 J. Huang, S. Rauscher, G. Nawrocki, T. Ran, M. Feig, B. L. de Groot, H. Grubmueller and A. D. MacKerell Jr, *Nat. Methods*, 2017, **14**, 71–73.
2 J. A. Maier, C. Martinez, K. Kasavajhala, L. Wickstrom, K. E. Hauser and C. Simmerling, *J. Chem. Theory Comput.*, 2015, **11**, 3696–3713.
3 M. J. Robertson, J. Tirado-Rives and W. L. Jorgensen, *J. Chem. Theory Comput.*, 2015, **11**, 3499–3509.
4 M. M. Reif, P. H. Honenberger and C. Oostenbrink, *J. Chem. Theory Comput.*, 2012, **8**, 3705–3723.
5 J. W. Ponder and D. A. Case, *Adv. Protein Chem.*, 2003, **66**, 27–85, Protein Simulations.
6 B. R. Brooks, C. L. Brooks, A. D. Mackerell, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R. W. Pastor, C. B. Post, J. Z. Pu, M. Schaefer, B. Tidor, R. M. Venable, H. L. Woodcock, X. Wu, W. Yang, D. M. York and M. Karplus, *J. Comput. Chem.*, 2009, **30**, 1545–1614.
7 A.-P. E. Kunz, J. R. Allison, D. P. Geerke, B. A. C. Horta, P. H. Hunenberger, S. Riniker, N. Schmid and W. F. van Gunsteren, *J. Comput. Chem.*, 2012, **33**, 340–353.
8 R. Salomon-Ferrer, D. A. Case and R. C. Walker, *Wiley Interdiscip. Rev.: Comput. Mol. Sci.*, 2013, **3**, 198–210.
9 J. A. Rackers, M. L. Laury, C. Lu, Z. Wang, L. Lagardère, J.-P. Piquemal, P. Ren and J. W. Ponder, *J. Comput. Chem.*, 2018, in preparation.
10 S. Plimpton, *J. Comput. Phys.*, 1995, **117**, 1–19.
11 J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale and K. Schulten, *J. Comput. Chem.*, 2005, **26**, 1781–1802.
12 M. J. Abraham, T. Murtola, R. Schulz, S. Pall, J. C. Smith, B. Hess and E. Lindahl, *SoftwareX*, 2015, **1–2**, 19–25.
13 R. Salomon-Ferrer, A. W. Gotz, D. Poole, S. Le Grand and R. C. Walker, *J. Chem. Theory Comput.*, 2013, **9**, 3878–3888.
14 W. Smith and I. T. Todorov, *Mol. Simul.*, 2006, **32**, 935–943.
15 C. Kobayashi, J. Jung, Y. Matsunaga, T. Mori, T. Ando, K. Tamura, M. Kamiya and Y. Sugita, *J. Comput. Chem.*, 2017, **38**, 2193–2206.
16 K. J. Bowers, E. Chow, H. Xu, R. O. Dror, M. P. Eastwood, B. A. Gregersen, J. L. Klepeis, I. Kolossvary, M. A. Moraes, F. D. Sacerdoti, J. K. Salmon, Y. Shan and D. E. Shaw, Scalable Algorithms for Molecular Dynamics Simulations on Commodity Clusters, *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, New York, NY, USA, 2006.
17 D. E. Shaw, R. O. Dror, J. K. Salmon, J. P. Grossman, K. M. Mackenzie, J. A. Bank, C. Young, M. M. Deneroff, B. Batson, K. J. Bowers, E. Chow, M. P. Eastwood, D. J. Ierardi, J. L. Klepeis, J. S. Kuskin, R. H. Larson, K. Lindorff-Larsen, P. Maragakis, M. A. Moraes, S. Piana, Y. Shan and B. Towles, Millisecond-scale Molecular Dynamics Simulations on Anton, *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, New York, NY, USA, 2009, pp 39:1–39:11.
18 N. Gresh, P. Claverie and A. Pullman, *Theor. Chim. Acta*, 1984, **66**, 1–20.
19 A. Stone, *The Theory of Intermolcular Forces*, Oxford Scholarship, 2013.
20 W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gould, K. M. Merz, D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. W. Caldwell and P. A. Kollman, *J. Am. Chem. Soc.*, 1996, **118**, 2309.
21 M. S. Gordon, M. A. Freitag, P. Bandyopadhyay, J. H. Jensen, V. Kairys and W. J. Stevens, *J. Phys. Chem. A*, 2001, **105**, 293–307.
22 O. Engkvist, P.-O. Atrand and G. Karlstromm, *Chem. Rev.*, 2000, **100**, 4087–4108.
23 M. S. Gordon, L. Slipchenko, H. Li and J. H. Jensen, in *Chapter 10 The Effective Fragment Potential: A General Method for Predicting Intermolecular Interactions*, ed. D. Spellmeyer and R. Wheeler, Annual Reports in Computational Chemistry Supplement C, Elsevier, 2007, vol. 3, pp. 177–193.
24 P. Ren and J. W. Ponder, *J. Phys. Chem. B*, 2003, **107**, 5933–5947.

25 G. Lamoureux and B. Roux, *J. Chem. Phys.*, 2003, **119**, 3025–3039.
26 N. Gresh, G. A. Cisneros, T. A. Darden and J.-P. Piquemal, *J. Chem. Theory Comput.*, 2007, **3**, 1960–1986.
27 W. L. Jorgensen, *J. Chem. Theory Comput.*, 2007, **3**, 1877.
28 Y. Shi, P. Ren, M. Schnieders and J.-P. Piquemal, *Reviews in Computational Chemistry*, John Wiley and Sons, Inc, 2015, vol. 28, pp. 51–86.
29 N. Gresh, K. E. Hage, E. Goldwaser, B. de Courcy, R. Chaudret, D. Perahia, C. Narth, L. Lagardère, F. Lipparini and J.-P. Piquemal, in *Quantum Modeling of Complex Molecular Systems*, ed. J.-L. Rivail, M. Ruiz-Lopez and X. Assfeld, Springer International Publishing, Cham, 2015, pp. 1–49.
30 J.-P. Piquemal and G. A. Cisneros, *Many-Body Effects and Electrostatics in Biomolecules*, Pan Standford, 2016, pp. 269–299.
31 W. Jiang, D. J. Hardy, J. C. Phillips, A. D. MacKerell, K. Schulten and B. Roux, *J. Phys. Chem. Lett.*, 2011, **2**, 87–92.
32 J. E. Stone, J. C. Phillips, P. L. Freddolino, D. J. Hardy, L. G. Trabuco and K. Schulten, *J. Comput. Chem.*, 2007, **28**, 2618–2640.
33 S. L. Grand, A. W. Gotz and R. C. Walker, *Comput. Phys. Commun.*, 2013, **184**, 374–380.
34 M. Harger, D. Li, Z. Wang, K. Dalby, L. Lagardère, J.-P. Piquemal, J. Ponder and P. Ren, *J. Comput. Chem.*, 2017, **38**, 2047–2055.
35 P. Eastman, J. Swails, J. D. Chodera, R. T. McGibbon, Y. Zhao, K. A. Beauchamp, L.-P. Wang, A. C. Simmonett, M. P. Harrigan, C. D. Stern, R. P. Wiewiora, B. R. Brooks and V. S. Pande, *PLoS Comput. Biol.*, 2017, **13**, 1–17.
36 J. W. Ponder, C. Wu, P. Ren, V. S. Pande, J. D. Chodera, M. J. Schnieders, I. Haque, D. L. Mobley, D. S. Lambrecht, R. A. J. DiStasio, M. Head-Gordon, G. N. I. Clark, M. E. Johnson and T. Head-Gordon, *J. Phys. Chem. B*, 2007, **114**, 2549–2564.
37 J.-P. Piquemal, L. Perera, G. A. Cisneros, P. Ren, L. G. Pedersen and T. A. Darden, *J. Chem. Phys.*, 2006, **125**, 054511.
38 K. J. Bowers, R. O. Dror and D. E. Shaw, *J. Chem. Phys.*, 2006, **124**, 184109.
39 M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*, Clarendon Press, New York, NY, USA, 1989.
40 L. Lagardère, F. Lipparini, E. Polack, B. Stamm, E. Cancès, M. Schnieders, P. Ren, Y. Maday and J. P. Piquemal, *J. Chem. Theory Comput.*, 2015, **11**, 2589–2599.
41 F. Lipparini, L. Lagardère, B. Stamm, E. Cancès, M. Schnieders, P. Ren, Y. Maday and J.-P. Piquemal, *J. Chem. Theory Comput.*, 2014, **10**, 1638–1651.
42 F. Aviat, A. Levitt, B. Stamm, Y. Maday, P. Ren, J. W. Ponder, L. Lagardère and J.-P. Piquemal, *J. Chem. Theory Comput.*, 2017, **13**, 180–190.
43 F. Aviat, L. Lagardère and J.-P. Piquemal, *J. Chem. Phys.*, 2017, **147**, 161724.
44 A. Klamt and G. Schuurmann, *J. Chem. Soc., Perkin Trans. 2*, 1993, 799–805.
45 F. Lipparini, B. Stamm, E. Cancès, Y. Maday and B. Mennucci, *J. Chem. Theory Comput.*, 2013, **9**, 3637–3648.
46 F. Lipparini, L. Lagardère, C. Raynaud, B. Stamm, E. Cancès, B. Mennucci, M. Schnieders, P. Ren, Y. Maday and J.-P. Piquemal, *J. Chem. Theory Comput.*, 2015, **11**, 623–634.
47 U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee and L. G. Pedersen, *J. Chem. Phys.*, 1995, **103**, 8577–8593.
48 A. Toukmaji, C. Sagui, J. Board and T. Darden, *J. Chem. Phys.*, 2000, **113**, 10913–10927.
49 C. Sagui, L. G. Pedersen and T. A. Darden, *J. Chem. Phys.*, 2004, **120**, 73–87.
50 C. Narth, L. Lagardère, É. Polack, N. Gresh, Q. Wang, D. R. Bell, J. A. Rackers, J. W. Ponder, P. Y. Ren and J.-P. Piquemal, *J. Comput. Chem.*, 2016, **37**, 494–506.
51 N. Li and S. Laizet, *Cray User Group 2010 conference*, Edinburgh, 2010.
52 M. Frigo and S. G. Johnson, *Proc. IEEE*, 2005, **93**, 216–231, Special issue on "Program Generation, Optimization, and Platform Adaptation".
53 J. Kolafa, *J. Comput. Chem.*, 2004, **25**, 335–342.
54 M. Tuckerman, B. J. Berne and G. J. Martyna, *J. Chem. Phys.*, 1992, **97**, 1990–2001.
55 J. Wang, P. Cieplak, Q. Cai, M.-J. Hsieh, J. Wang, Y. Duan and R. Luo, *J. Phys. Chem. B*, 2012, **116**, 7999–8008.
56 Y. Shi, Z. Xia, J. Zhang, R. Best, C. Wu, J. W. Ponder and P. Ren, *J. Chem. Theory Comput.*, 2013, **9**, 4046–4063.
57 A. Marjolin, C. Gourlaouen, C. Clavaguéra, P. Y. Ren, J. C. Wu, N. Gresh, J.-P. Dognon and J.-P. Piquemal, *Theor. Chem. Acc.*, 2012, **131**, 1198.
58 A. Marjolin, C. Gourlaouen, C. Clavaguéra, P. Y. Ren, J.-P. Piquemal and J.-P. Dognon, *J. Mol. Model.*, 2014, **20**, 2471.
59 T. A. Halgren, *J. Am. Chem. Soc.*, 1992, **114**, 7827–7843.
60 Y. Shi, Z. Xia, J. Zhang, R. Best, C. Wu, J. W. Ponder and P. Ren, *J. Chem. Theory Comput.*, 2013, **9**, 4046–4063.
61 A.-P. Hynninen and M. F. Crowley, *J. Comput. Chem.*, 2014, **35**, 406–413.
62 F. Endres and S. Z. E. Abedin, *Phys. Chem. Chem. Phys.*, 2006, **8**, 2101–2116.
63 R. P. Swatloski, S. K. Spear, J. D. Holbrey and R. D. Rogers, *J. Am. Chem. Soc.*, 2002, **124**, 4974–4975.
64 H. Zhang, J. Wu, J. Zhang and J. He, *Macromolecules*, 2005, **38**, 8272–8277.
65 D. Li, M. Wang, J. Wu, Q. Zhang, Y. Luo, Z. Yu, Q. Meng and Z. Wu, *Langmuir*, 2009, **25**, 4808–4814.
66 O. N. Starovoytov, H. Torabifard and G. A. Cisneros, *J. Phys. Chem. B*, 2014, **118**, 7156–7166.
67 Y.-J. Tu, M. J. Allen and G. A. Cisneros, *Phys. Chem. Chem. Phys.*, 2016, **18**, 10323–30333.
68 H. Torabifard, L. Reed, M. T. Berry, J. E. Jein, E. Menke and G. Cisneros, *J. Chem. Phys.*, 2017, **147**, 161731.
69 T. Yan, C. J. Burnham, M. G. D. Pópolo and G. A. Voth, *J. Phys. Chem. B*, 2004, **108**, 11877–11881.
70 A. Bagno, F. D'Amico and G. Saielli, *J. Mol. Liq.*, 2007, **131–132**, 17–23.
71 O. Borodin and G. D. Smith, *J. Phys. Chem. B*, 2006, **110**, 6279–6292.

72 V. V. Chaban and O. V. Prezhdo, *Phys. Chem. Chem. Phys.*, 2011, **13**, 19345–19354.
73 C. H. Bennett, *J. Comput. Phys.*, 1976, **22**, 245–268.
74 L. Zheng, M. Chen and W. Yang, *Proc. Natl. Acad. Sci. U. S. A.*, 2008, **105**, 20227–20232.
75 P. L. Freddolino, A. S. Arkhipov, S. B. Larson, A. McPherson and K. Schulten, *Structure*, 2006, **14**, 437–449.
76 J. R. Perilla, B. C. Goh, C. K. Cassidy, B. Liu, R. C. Bernardi, T. Rudack, H. Yu, Z. Wu and K. Schulten, *Current Opinion in Structural Biology*, 2015, vol. 31, pp. 64–74, Theory and simulation/Macromolecular machines and assembliesTheory and simulation/Macromolecular machines and assemblies.
77 E. G. Kratz, A. R. Walker, L. Lagardère, F. Lipparini, J.-P. Piquemal and G. Andrés Cisneros, *J. Comput. Chem.*, 2016, **37**, 1019–1029.
78 C. Curutchet, A. Muñoz-Losa, S. Monti, J. Kongsted, G. D. Scholes and B. Mennucci, *J. Chem. Theory Comput.*, 2009, **5**, 1838–1848.
79 S. Caprasecca, S. Jurinovich, L. Lagardère, B. Stamm and F. Lipparini, *J. Chem. Theory Comput.*, 2015, **11**, 694–704.
80 D. Loco, E. Polack, S. Caprasecca, L. Lagardère, F. Lipparini, J.-P. Piquemal and B. Mennucci, *J. Chem. Theory Comput.*, 2016, **12**, 3654–3661.
81 D. Loco, L. Lagardère, S. Caprasecca, F. Lipparini, B. Mennucci and J.-P. Piquemal, *J. Chem. Theory Comput.*, 2017, **13**, 4025–4033.
82 F. Lipparini, L. Lagardère, G. Scalmani, B. Stamm, E. Cancès, Y. Maday, J.-P. Piquemal, M. J. Frisch and B. Mennucci, *J. Phys. Chem. Lett.*, 2014, **5**, 953–958.
83 F. Lipparini, G. Scalmani, L. Lagardère, B. Stamm, E. Cancès, Y. Maday, J.-P. Piquemal, M. J. Frisch and B. Mennucci, *J. Chem. Phys.*, 2014, **141**, 184108.
84 J. Tomasi, B. Mennucci and R. Cammi, *Chem. Rev.*, 2005, **105**, 2999–3093.
85 C. Cramer and D. Truhlar, *Chem. Rev.*, 1999, **99**, 2161–2200.
86 M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, G. A. Petersson, H. Nakatsuji, X. Li, M. Caricato, A. V. Marenich, J. Bloino, B. G. Janesko, R. Gomperts, B. Mennucci, H. P. Hratchian, J. V. Ortiz, A. F. Izmaylov, J. L. Sonnenberg, D. Williams-Young, F. Ding, F. Lipparini, F. Egidi, J. Goings, B. Peng, A. Petrone, T. Henderson, D. Ranasinghe, V. G. Zakrzewski, J. Gao, N. Rega, G. Zheng, W. Liang, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, K. Throssell, J. A. Montgomery Jr, J. E. Peralta, F. Ogliaro, M. J. Bearpark, J. J. Heyd, E. N. Brothers, K. N. Kudin, V. N. Staroverov, T. A. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A. P. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, J. M. Millam, M. Klene, C. Adamo, R. Cammi, J. W. Ochterski, R. L. Martin, K. Morokuma, O. Farkas, J. B. Foresman and D. J. Fox, *Gaussian 16 Revision A.03*, Gaussian Inc, Wallingford CT, 2016.
87 R. M. Parrish, L. A. Burns, D. G. A. Smith, A. C. Simmonett, A. E. DePrince, E. G. Hohenstein, U. Bozkaya, A. Y. Sokolov, R. Di Remigio, R. M. Richard, J. F. Gonthier, A. M. James, H. R. McAlexander, A. Kumar, M. Saitow, X. Wang, B. P. Pritchard, P. Verma, H. F. Schaefer, K. Patkowski, R. A. King, E. F. Valeev, F. A. Evangelista, J. M. Turney, T. D. Crawford and C. D. Sherrill, *J. Chem. Theory Comput.*, 2017, **13**, 3185–3197.
88 M. Valiev, E. Bylaska, N. Govind, K. Kowalski, T. Straatsma, H. V. Dam, D. Wang, J. Nieplocha, E. Apra, T. Windus and W. de Jong, *Comput. Phys. Commun.*, 2010, **181**, 1477–1489.
89 M. Devereux, S. Raghunathan, D. G. Fedorov and M. Meuwly, *J. Chem. Theory Comput.*, 2014, **10**, 4229–4241.
90 Y. Zhang, T.-S. Lee and W. Yang, *J. Chem. Phys.*, 1999, **110**, 46–54.
91 J. M. Parks, H. Hu, A. J. Cohen and W. Yang, *J. Chem. Phys.*, 2008, **129**, 154106.
92 L.-P. Wang, T. Head-Gordon, J. W. Ponder, P. Ren, J. D. Chodera, P. K. Eastman, T. J. Martinez and V. S. Pande, *J. Phys. Chem. B*, 2013, **117**, 9956–9972.
93 A. M. N. Niklasson, P. Steneteg, A. Odell, N. Bock, M. Challacombe, C. J. Tymczak, E. Holmstrom, G. Zheng and V. Weber, *J. Chem. Phys.*, 2009, **130**, 214109.
94 M. Müller and T. Aoki, arXiv preprint arXiv:1710.08616 2017.
95 openacc, https://www.openacc.org, accessed: 2017-05-31.
96 J. Huang, A. C. Simmonett, F. C. Pickard IV, A. D. MacKerell Jr and B. R. Brooks, *J. Chem. Phys.*, 2017, **147**, 161702.
97 J.-P. Piquemal, G. A. Cisneros, P. Reinhardt, N. Gresh and T. A. Darden, *J. Chem. Phys.*, 2006, **124**, 104101.
98 R. E. Duke, O. N. Starovoytov, J.-P. Piquemal and G. A. Cisneros, *J. Chem. Theory Comput.*, 2014, **10**, 1361–1365.

List of Figures

List of Tables