FAST POLARIZABLE FORCE FIELD COMPUTATION IN BIOMOLECULAR SIMULATIONS

BY

WEI WANG

B.S., Tsinghua University, 1993
M.S., Peking University, 1996
M.S., University of Illinois at Urbana-Champaign, 2001

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2005

Urbana, Illinois

FAST POLARIZABLE FORCE FIELD COMPUTATION IN BIOMOLECULAR SIMULATIONS

Wei Wang,
Department of Computer Science
University of Illinois at Urbana-Champaign, 2005
Robert D. Skeel, Advisor

Polarizable force fields are considered to be the single most significant development in the next-generation force fields used in biomolecular simulations. The self-consistent computation of induced atomic dipoles in a polarizable force field is expensive due to the cost of solving a large dense linear system at each timestep in molecular dynamics simulations. Methods are developed that reduce the cost of computing the electrostatic energy and force of a polarizable model from about 7.5 times the cost of computing those of a non-polarizable model to less than twice the cost. The reduction is achieved by an efficient implementation of the particle–mesh Ewald method, an accurate and robust predictor based on least squares fitting, and two non-stationary iterative methods whose fast convergence is empowered by a simple preconditioner. Furthermore, with these methods, we show that the self-consistent approach with a larger timestep is faster than the extended Lagrangian approach. The use of dipole moments from previous timesteps to calculate an accurate initial guess for iterative methods leads to an energy drift and compromises the volume-preserving property of the integration. Iterative methods with zero initial guess do not lead to perceptible energy drift if a reasonably strict convergence criterion for the iteration is imposed and the numerical integrator is volume-preserving. The approximate solution computed by an iterative method ruins the symplectic property of the integrator. To address this problem, a non-iterative method has been developed based on an approximation to the electrostatic potential energy and has been efficiently implemented. The method preserves the symplecticness of the integrator and is suitable for long time simulations. The research will help polarizable force fields modeling and computation to become a routine part of molecular dynamics simulations for biomolecular systems.

This dissertation is dedicated to my parents, my wife, and my daughter.

Acknowledgments

I would like to express my profound gratitude to my advisor, Prof. Robert D. Skeel, for his deep insight and guidance, as well as his patience, high standards, and serious and disciplined attitude toward work and life. The thesis would not come into existence without his great help. I would like to thank Prof. Klaus Schulten for hosting me in his productive group. I wish I could have his insight and broad vision some day. I would like to thank Prof. Stephen Bond, who demonstrated me many wonders in molecular dynamics simulations in his class. He also helped me a lot through many discussions. I wish him the best luck in his career and life.

My sincere thanks go to David J. Hardy, for his help, for stimulating discussions, and for his MDX software, from which started my research code. In the Theoretical and Computational Biology Group, Markus Dittrich, Chalermpol Kanchanawarin (Tik), and Barry Isralewitz lent me their generous help and encouragement, which are highly appreciated. They also kindly allowed me to run jobs on their desktops when the normal computer job queues were full. John Stone kindly helped me improve the code execution speed. In the Numerical Analysis group in Siebel Center, Eric Cyr, David Alber and Rebecca Hartman-Baker helped me improve my slides. Eric also helped me in many other aspects. I want to thank Gang Zou, Yuzhou Tang, Xuemin Gu for their help in work as well as in life.

Finally I should thank my wife, Yongquan Yuan, and my parents for their support, sacrifice, and patience.

I would like to acknowledge the financial support awarded by National Institutes of Health (Grant P41RR05969) and by National Science Foundation (Grant DMS-020442).

Table of Contents

List of Tables

List of Figures

Chapter 1

Polarizable force fields

Polarization refers to the electron density redistribution due to the electric field. Current generation non-polarizable force fields for biomolecular simulations, such as OPLS [72], CHARMM [87], AMBER [33], MMFF [61], and GROMOS [15, 124], include the polarization *implicitly* in the charge–charge and Lennard-Jones parameterizations [62, 114, 108]. They have serious theoretical and practical limitations because the polarization is treated only in an average sense [62, 108, 104, 52]. The treatment cannot reflect the dependency of the electron density on the positions of atoms, nor can it respond dynamically to different environment, which varies from almost non-conductive inside the protein cavities to very conductive on the protein–water interface. The explicit inclusion of polarization can significantly improve a force field's (i) *accuracy*, when being compared to quantum computation or experimental results [139], and (ii) *transferability*, when applying the same force field to a wide range of temperature and pressure [122]. Much research has shown promising results for polarizable force fields [25, 42, 12, 59, 146, 66, 69, 145, 56, 122, 11]. In fact, the inclusion of polarizability is considered the single most significant development in the next-generation force fields [62, 108, 86] in biomolecular modeling and simulations. Polarizable models have the prospect to enable accurate computation of the binding energies of proteins and ligands in drug design [108], an application of huge industrial importance. What is more, polarizability is indispensable for studies of interfaces [37, 123], and of some ionic or hydrophobic solvation processes [127, 17, 28, 137, 106, 73, 69, 147].

Reference [114] gives a thorough description of polarizable force field modeling principles and their computational costs; reviews [108] and [86] focus on polarizable force fields within the bigger picture of classical force field development, while an earlier paper [62] summarizes the growing effort

on polarizable force field modeling in studies of water, solvation, and some small biomolecules.

The next two sections present the "big picture" into which the current dissertation is embedded. This is followed by an overview of the dissertation.

1.1 Polarization models

Polarization models can be roughly divided into two categories: point dipole models and fluctuating charge models. A variation of the point dipole model is the shell (Drude) model; a variation of the fluctuating charge model is the semi-empirical model [114].

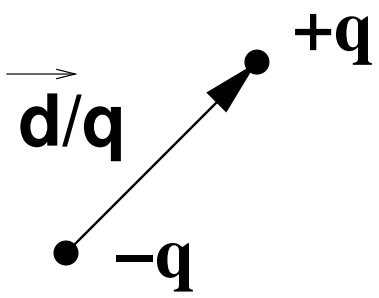

Figure 1.1: A point dipole $\vec{d}$ can be regarded as two charges of opposite signs in the limit of $q \to \infty$.

A point dipole model represents the charge distribution of an atom by a charge and an induced dipole. The model can be considered as a natural extension of the point charge model by including the next term in the multipole expansion [134][70, chapter4]. In a point dipole model [6, 5, 133, 116, 130, 1, 24, 17, 117, 25, 26, 21, 31, 38], the pairwise potential energy between atoms at $\vec{r}_i$ and $\vec{r}_j$, with charges q_i and q_j and dipole moments $\vec{d}_i$ and $\vec{d}_j$, respectively, is

$$(q_i + \vec{d}_i \cdot \nabla_i)(q_j + \vec{d}_j \cdot \nabla_j)\frac{1}{|\vec{r}_i - \vec{r}_j|}\,. \tag{1.1}$$

In general there is a pre-factor K/ϵ_s, where K is a constant and ϵ_s is the relative permittivity of the medium. Here we use CGS units, for which $K = 1$, and assume a vacuum medium, for which

$\epsilon_s = 1$. The total electrostatic energy of an N-atom system can be represented in matrix form as

$$E(\boldsymbol{r}, \boldsymbol{d}) \;=\; \frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\boldsymbol{G}_0(\boldsymbol{r})\boldsymbol{q} + \boldsymbol{d}^{\mathsf{T}}\boldsymbol{G}_1(\boldsymbol{r})\boldsymbol{q} + \frac{1}{2}\boldsymbol{d}^{\mathsf{T}}\boldsymbol{G}_2(\boldsymbol{r})\boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^{\mathsf{T}}\boldsymbol{D}_\alpha^{-1}\boldsymbol{d}\,, \tag{1.2}$$

where $\boldsymbol{q}$ is the collection of charges, $\boldsymbol{d}$ the collection of dipole moments, $\boldsymbol{G}_0$, $\boldsymbol{G}_1$, and $\boldsymbol{G}_2$ are the charge–charge, charge–dipole, and dipole–dipole interaction matrices defined through (1.1), and $\boldsymbol{D}_\alpha$ is a block diagonal matrix incorporating the polarizability of each atom. The last term is essentially the "polarization energy," which atoms must overcome to have nonzero dipole values. Induced dipoles assume values that minimize the energy (1.2):

$$\frac{\partial}{\partial \boldsymbol{d}} E(\boldsymbol{r}, \boldsymbol{d}) = 0 \quad \Rightarrow \quad (\boldsymbol{D}_\alpha^{-1} + \boldsymbol{G}_2)\boldsymbol{d} = -\boldsymbol{G}_1\boldsymbol{q}\,. \tag{1.3}$$

Once the dipole is known, the energy and force can be computed subsequently.

Eq. (1.3) reflects the "non-additive" nature of a polarizable force field: adding one more atom would change the dipole value on *each* atom, thus changing the interaction between *every* atom pair. So a total re-computation is needed, instead of simply adding the interaction of this new atom with all other atoms. An important observation is that $\boldsymbol{D}_\alpha^{-1} + \boldsymbol{G}_2$ must be positive definite since the total electrostatic energy, a quadratic in $\boldsymbol{d}$, must be lower-bounded. However, $\boldsymbol{D}_\alpha^{-1} + \boldsymbol{G}_2$ can become indefinite when two dipoles are too close to each other. This is a shortcoming of the model, known as a "polarization catastrophe" [6, 138], where unreasonably large dipole values are computed from (1.3), which do not correspond to the minimum of the electrostatic energy. The Lennard-Jones potential can keep atoms from being too close. But when it is not enough, model designers add damping terms to the dipole–dipole interaction at small distances [138, 17, 112], or choose particularly small polarizability values [6, 5] so that it is unlikely the polarization catastrophe occurs in molecular dynamics simulations. Notice that the second approach may not be able to avoid the catastrophe in Monte Carlo simulations, where moves are unphysical.

In shell (Drude) models [143, 47, 82, 81, 148, 149, 129, 122], an induced dipole is emulated by two charges of opposite signs, as is shown in Fig. 1.2. Each polarizable atom is represented by a pair of point charges bound by a stiff spring. Shell models can be easily implemented in a computer program since the charge–charge and bonding interactions are already present in current

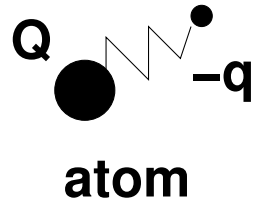

Figure 1.2: In a shell (Drude) model, a positively charged atom is represented by two charges ($Q > q$) bound together by a stiff spring.

force fields. However, the number of charges is doubled in a shell model and the computational cost is doubled at least.

Polarization can also be modeled by allowing the value of partial charges to change in response to the local electric field. A fluctuating charge model [8, 84, 9, 29, 136, 29, 99, 100, 125] approximates the energy required to create a charge q by a Taylor expansion in q to second order:

$$U(q) = \chi q + \frac{1}{2} J q^2 \,,$$

where χ is the "electronegativity" and $\frac{1}{2}J > 0$ is the "hardness" of the atom. The total energy of the system is the summation of these single-atom energies and the electrostatic interactions between the charges:

$$U(\boldsymbol{q}) \;=\; \sum_i (\chi_i q_i + \frac{1}{2} J_{ii} q_i^2) + \sum_{i>j} J_{ij}(r_{ij}) q_i q_j \,, \tag{1.4}$$

where $J_{ij}(r_{ij})$ is equal to $1/r_{ij}$ at large distances but can be different at short distances to account for the finite spatial distribution of each charge. The charges take values to minimize the total electrostatic energy with the constraint that the total net charge is a constant [114]:

$$\boldsymbol{q} \;=\; \arg\min_{\boldsymbol{q}} \left[U(\boldsymbol{q}) + \mu (\sum_i q_i - q_{\rm tot}) \right] , \tag{1.5}$$

where μ is the Lagrangian multiplier. This gives

$$-\frac{\partial U(\boldsymbol{q})}{\partial q_i} = \mu \,, \qquad \text{for } i = 1, 2, \ldots, N \,. \tag{1.6}$$

The left hand side of the above equation is called the "chemical potential" of atom i, so (1.6) means all atoms should have the same chemical potential. In many cases, the charge flow is confined artificially inside each molecule to avoid model deficiencies [114]. Computationally speaking, the fluctuating charge model is probably the most efficient model: it does not have dipole interactions, so it is simpler than point dipole models; its number of charges remains the same as the number of atoms, a clear advantage over shell models.

However, the fluctuating charge model has some limitations. Probably the most significant one is due to a geometry effect. For example, water, a planar three-atom molecule, can only have a planar polarizability according to this model, although experiments show a water molecule's polarizability is nearly isotropic [62, 114]. A current trend is to combine fluctuating charge models with the point dipole model [131, 132, 77, 76].

Semi-empirical models [50, 51, 23, 22, 58, 71] are based on systematic approximations to quantum mechanics descriptions and can be considered as an "advanced" type of fluctuating charge model [114] in the sense that the fluctuating charge is represented by atomic orbitals. They generally have better agreement with experiments but computationally are even more expensive than other polarizable force fields. For large systems and long simulations, these models are still too expensive to use.

In this dissertation, we choose to compute the point dipole models, since they have less modeling limitations and are widely used in literature.

1.2 Computational methods

Forces are computed in various contexts: deterministic dynamics, stochastic dynamics, Monte Carlo simulations, and energy minimization. The context has important implications for the polarizable force computation. In deterministic dynamics, energy conservation is important. This generally requires the computed force to be the exact negative gradient of a potential to a very high accuracy. On the other hand, the continuity of molecular dynamics simulation enables accurate predictions (initial guesses) for the point dipoles, which can help greatly in solving the self-consistent equation, though the use of history undermines badly the symplecticness of the integrator. In stochastic dynamics, e.g., integrating the Langevin equation, damping terms in the equation of motion gen-

erally enhance the stability, so the force can be computed with approximations which destroy its conservative property as long as it is accurate. The flip side is the initial guess of dipoles generally is not as good as that in deterministic dynamics. Monte Carlo simulation can be very expensive for polarizable force fields since a trial move of one atom or molecule results in a total re-computation of the energy and force if treated rigorously. Current Monte Carlo methods either compromise the detailed balance for efficiency [90, 121, 102], or use techniques similar to the extended Lagrangian method [88, 30].

We are mainly concerned with deterministic molecular dynamics (MD) simulations, for which two methods are widely used for polarizable force field computation [62, 114]: the self-consistent method and the extended Lagrangian method. At each timestep, the self-consistent method overtly minimizes the electrostatic energy with respect to the polarizable degrees of freedom, which are induced dipoles in a point dipole model, auxiliary charge positions in a shell model, and charge values in a fluctuating charge model. In other words, Eq. (1.3) is solved for the point dipole model, and Eq. (1.5) is solved for the fluctuating charge model. This is generally very expensive. The extended Lagrangian method [27, 130] treats the polarizable degrees of freedom as dynamic variables, which are assigned kinetic energies with fictitious masses. Their masses serve for computational convenience and do not have any physical meaning. Starting from a configuration with the electrostatic potential minimized, the method simply does normal dynamics with a small timestep. Although it does not explicitly minimize the electrostatic energy at each timestep, it can keep the electrostatic energy close to its minimal values for a certain time. The length of the time depends on the coupling of the fictitious subsystem with the rest of the system: the weaker the coupling, the longer the time. It has been proven that, under certain conditions, the atom position error is proportional to the square-root of the fictitious mass [19]. Small fictitious mass values are required to keep the fictitious kinetic energy (temperature) low as well as to reduce the error [111, 98].

The extended Lagrangian method is faster than the self-consistent method since it only approximately minimizes the electrostatic energy. But it has the following drawbacks:

- The fictitious mass must be small to reduce the dynamic coupling between the polarizable degrees of freedom and the atomic coordinates so that a low temperature of the polarizable degrees of freedom is maintained [111]. This, in turn, requires a smaller integration timestep.

For example, timesteps of 0.2 and 1 fs (1 fs = 10^{-15} second) have to be used in references [129] and [140], respectively. This effectively reduces the efficiency.

- It introduces artifacts. First, the physical system's linear momentum is no longer conserved. Reference [92] further points out when coupling different subsystems to different reservoirs by Nosé–Hoover method, no properly defined linear momentum can be defined, and a large heat flow is observed between a reservoir and a subsystem. Secondly, since the polarizable degrees of freedom are, in fact, at a much lower temperature than that of the rest of the system, the system is in a metastable state [111, 101]. The heat flow from other degrees of freedom to the polarizable degree of freedom is undesirable, yet unavoidable. In a recent paper [63], even though the timestep is limited to 0.75–1 fs, the system has to have a full energy minimization every 300 ps (1 ps = 10^3 fs), making the dynamics irreversible.

In this dissertation, we choose the self-consistent approach because it is suitable for kinetic as well as thermodynamic calculations and because it is a standard for induced dipole calculation against which other more compromised approaches can be compared.

The self-consistent computation of point dipole models has been very expensive. In fact, the slow adoption of polarizable force fields is partly due to its much higher computation cost [86, 122]. Compared with the charge-only models, references [116] and [97] reports respectively a nine- and eight-fold increase in the computational cost with the standard Ewald sum implementation, while [140] reports a more than six-fold increase with the particle–mesh Ewald [39, 44] implementation. Review [114] says the computation of the Ewald sum of a point dipole model is, as "a widely used rule of thumb", four times more expensive than that of a charge-only model. The next two chapters of this dissertation are devoted to improving the result of reference [140]. We are able to compute the self-consistent solution for an induced dipole model with less than 100% extra computation cost.

1.3 Dissertation outline

The fundamental problem is solving the dipole equation efficiently in MD simulations. Our effort on this is presented in Chapter 3. The problem is intimately related to two other topics in MD: fast electrostatic solvers and dynamics. The first topic is handled quite satisfactorily in Chapter 2.

The second topic is tackled in Chapters 4 and 5.

The slow decay $(1/r)$ of the charge–charge interaction makes the computation of the electrostatic interaction *the* most expensive task in biomolecular simulations. A simple cut-off treatment can have serious unphysical effects [57, 85, 2, 46, 97, 80, 16, 89, 113]. On the other hand, the Ewald summation [3] is considered a reliable way for describing the electrostatic interaction [3], although it could lead to bias in free energy computation [16, 78] and to artificial stability if the cell size is too small [144]. The straightforward implementation of the Ewald sum is $O(N^{3/2})$ at best [49, page 304-306] where N is the number of the atoms. So the computation should be carried out with a fast electrostatic solver such as the particle–mesh Ewald (PME) method [39, 44], whose asymptotic cost is $O(N \log N)$.

Chapter 2 provides our matrix formulation and implementation of the Ewald sum and the PME method for a point dipole model. PME is widely used in biomolecular simulation software packages, such as NAMD [74] and AMBER [33, 103]. Our formulation reveals that PME can be easily extended to multipole computations [120] and can be implemented very efficiently. In our implementation, the energy/force computation with the dipole moments given incurs only about one quarter extra cost compared to a charge-only model. The most significant extra cost of the implementation in reference [140] comes from solving the dipole equation (1.3) iteratively: one iteration is almost as expensive as one full energy/force evaluation for a charge-only model. The major contribution of this chapter is an algorithm that reduces the cost to one third. The implementation based on our formulation selectively stores intermediate values to avoid most re-computations.

Chapter 3 presents new methods that reduce the number of iterations from six in reference [140] to two when solving the dipole equation to a given accuracy level. Our improvement comes from two contributions: a more accurate prediction and a faster converging iteration method. We first show that polynomial extrapolation of degree from four to six can give very accurate predictions, even though the underlying integrator is only second order accurate. However, polynomial prediction suffers from the numerical instability and possibly from the Runge phenomena [35, §4.3.4]. So we propose and implement a new predictor based on a least squares fitting of previous values of the dipole moment. The new predictor is accurate, because its predictions are as good as or better

than those from polynomial extrapolations, and robust, because the prediction quality does not degrade as more dipoles are used. To accelerate convergence for the iteration process, we propose and implement the Chebyshev semi-iterative method [41] and a modified conjugate gradient (CG) method, both methods taking advantage of the matrix $\boldsymbol{D}_\alpha^{-1}+\boldsymbol{G}_2$ being symmetric positive definite. A disadvantage of the standard CG implementation is that two matrix–vector multiplications are needed to get the first update to the solution. Although the residual is obtained simultaneously as the converged solution is obtained, it cannot be used. So one matrix-vector multiplication is wasted. We make use of this last matrix-vector multiplication by a suboptimal last step so that CG becomes competitive with the Chebyshev method. The performance of the two non-stationary iterative methods depends on the condition number of the matrix in the linear system being solved. For this purpose, we design and implement a simple but efficient preconditioner based on a local approximation to the dipole–dipole interaction matrix and, furthermore, a polynomial approximation to the inverse of the local approximation [141]. The comparison of the computational cost between a polarizable point-dipole model and a charge-only model shows that our implementation incurs less than 100% extra cost. The following table summarizes the computational costs of our implementation. Costs are expressed in work units. One work unit is the computation cost for a full energy/force evaluation of the charge-only model.

	cost of computing dipoles (cost per iteration × iterations)	cost of computing energy and force	overall cost
reference [140]	$(\approx 1) \times 6$	1.25–1.30	≈ 7.5
our implementation	0.34×2	1.28	1.94

Moreover, our implementation with a larger timestep, e.g., 2 fs, is faster than the extended Lagrangian method.

Chapter 4 discusses the effect of dipole moment quality on dynamics. Foremost, we discuss the the energy drift problem of the self-consistent computation. For Hamiltonian system simulations, the conservation of the Hamiltonian is strongly assisted by the symplectic integration. However, self-consistent computation compromises the symplecticness in two ways: (1) inexact solution makes the force nonconservative, and (2) prediction from previous values makes the self-consistent solution history-dependent. By examining each effect separately, we find non-conservativeness alone does not

cause significant energy drift if the computed dipole moments are reasonably accurate. But history dependence is more detrimental and for tolerable energy drift, it requires the computed dipole moments to be two orders of magnitude more accurate than what is suitable in MD simulations. In Chapter 4, we also discuss the always-stable-predictor-corrector (ASPC) method [79], which can maintain a constant energy for a long time with only one iteration if the timestep is 1 fs. The method has a poor accuracy and does not have a direct accuracy control mechanism. We improve the method by combining its quasi-time-reversible prediction with the least squares prediction and requesting the iteration not to stop until the solution is accurate enough. The improvement maintains accuracy with a small amount of extra cost. Furthermore, we observe that the method fails to conserve energy when a larger timestep is used. Preservation of phase space volume is a property weaker than the symplecticness, but still desirable for a numerical integration. We point out that the self-consistent computation is volume-preserving if it does not use history.

Our recommendations of method and parameters depend on quality and cost. If a short timestep ($\Delta t = 1, 2$ fs) is used, use the least squares predictor with 8 or more previous dipoles and require high accuracy for the dipole solution (to keep the energy drift tolerable). If a longer timestep ($\Delta t > 2$ fs) is used, e.g., in a multiple-time-stepping method, prediction helps very little to obtain an accurate initial guess. So use zero as an initial guess and declare convergence for a relatively low accuracy. This is still good enough to compute the energy and force with an accuracy suitable for MD simulations and maintain the energy at a constant level for long time simulations.

Chapter 5 details a non-iterative method to avoid the secular energy drift problem. The electrostatic energy is (re)defined through an appropriate polynomial which approximates the matrix inverse accurately. Then the force is computed as the exact negative gradient of the energy. By doing this, the energy drift problem is eliminated while we still maintain a reasonable accuracy for the computed dipoles.

Appendix A introduces the physical models and discusses some implementation issues. We have designed many nontrivial numerical tests to ensure implementation correctness. We also point out a pitfall when the constraint enforcement is combined with velocity rescaling methods.

In summary, our major contributions include the following:

- The Ewald sum and the PME method for induced point-dipole model computations are

formulated in a clear and concise matrix form. The PME method is implemented efficiently.

- A least squares predictor is designed which is more accurate than polynomial extrapolation predictor and more robust against numerical instability.

- Efficient iteration algorithms are developed for solving the dipole equation. We improve the CG method by peeking ahead one step to get an accurate solution sooner. We also design a simple and efficient preconditioner based on a local approximation to the dipole–dipole interaction matrix and furthermore a polynomial approximation to the inverse of the preconditioner. With the efficient algorithms we have developed, we demonstrate that the self-consistent computation with 1 fs timestep incurs less than 100% computational cost, and the self-consistent approach with a larger timestep is faster than the extended Lagrangian approach.

- The energy drift problem is clarified. Two sources in the self-consistent computation lead to the energy drift: a nonconservative force and the use of information from previous timesteps. The second source is more detrimental in leading to the energy drift. We also point out that the computation without using history is volume-preserving.

- The ASPC method is improved by an accuracy control mechanism with a small extra cost. which is achieved by the efficient iteration algorithm as well as the combination of the quasi-time-reversible predictor in the ASPC method with the least squares predictor.

- A novel non-iterative method based on an accurate polynomial approximation to the matrix inverse is proposed and an efficient implementation is carried out. The idea is to make it feasible to compute the force as the negative gradient of the potential. The method eliminates the energy drift problem and is suitable for long time simulations.

Chapter 2

Computation of the Ewald sum

Ewald summation [45] is a *description* of the long-range electrostatic interactions. The simulated systems are generally of limited size due to limited computing resources, leading to unwanted surface effects. To avoid them, periodic boundary conditions are applied [3], namely, the system is replicated infinitely to fill the space, as is illustrated in Fig. 2.1. Unfortunately, the resulting infinite sum of the charge–charge interaction converges only conditionally, if the total charge is 0, and diverges otherwise. Ewald summation corresponds to a special summation order specified in Eq. (2.1) [40, 3].

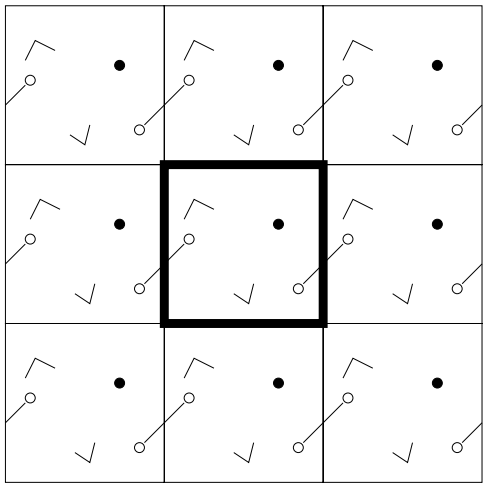

Figure 2.1: Periodic boundary conditions in two dimensions.

Computing the Ewald sum is *the* most expensive task in biomolecular simulations. For a given accuracy, direct implementation of the Ewald sum is $O(N^{3/2})$ at best [49, page 304-306]. Fast electrostatic solvers reduce the cost significantly. Grid-based methods, such as particle–mesh Ewald

(PME) [39, 44] and particle–particle particle–mesh (P^3M) [67] methods, map charges at arbitrary positions to the nodes of a uniform grid using a "restriction operator," thus enabling use of the fast-Fourier transform (FFT) to compute the reciprocal sum efficiently at a cost of $O(N \log N)$. Tree-code based methods, such as the fast multipole method [55, 54], recursively divide the space into cells and each cell into sub-cells, thus forming a tree structure. Close-range pairs are computed exactly, while charges far away are grouped by their cell ID, and the corresponding interactions are approximated by the multipole expansion. Although the cost is $O(N)$, the coefficient of N is so large that it is actually slower than PME or P^3M method for systems of tens of thousands of atoms [107]. Furthermore, when a charge moves and switches its role from "nearby" to "far away" relative to another charge, their interaction changes discontinuously from being treated exactly to being treated approximately. This leads to energy drift within the range of a few picoseconds [126] unless very high accuracy approximation is used. A promising $O(N)$ method is the multilevel summation method [126, 20]. The method recursively employs the separation of length scale and the approximation of the smoother ("far away") interactions on a coarser grid. For a non-periodic system, the multilevel summation method is four times faster than the fast multipole method for an accuracy suitable for molecular dynamics simulations; for a periodic system, its speed is comparable to the PME method. We choose the PME method because it is fast, efficient, and widely used in biomolecular simulation software packages, such as NAMD [74] and AMBER [33, 103].

Sections 2.1 and 2.2 present the matrix formulation of the Ewald sum of an induced point dipole model and the PME method, respectively. Compared to other formulations [96, 140], our approach is concise and simple. It promotes a high-level understanding of computation and implementation, and it is particularly suitable for representing the dipole equation. It reveals why PME can be easily generalized to high-order multipole computations [120] and why iteratively solving the dipole equation can be expensive.

Some data are repeatedly used in the matrix–vector multiplications when solving a dipole equation, while a single use is enough for computing a charge-only model. Section 2.2 shows by selectively storing intermediate data to avoid re-computations, we reduce the computational cost for a matrix–vector multiplication from about 100% reported in reference [140] to about 34%.

2.1 Ewald sum for a point dipole model

A simulated physical system is generally a parallelepiped box with edges given by three linearly independent basis vectors $\vec{a}_1$, $\vec{a}_2$, $\vec{a}_3$, and its volume $V = \det(\vec{a}_1, \vec{a}_2, \vec{a}_3)$. The corresponding reciprocal lattice basis vectors are $\vec{b}_1$, $\vec{b}_2$, and $\vec{b}_3$, which are defined so that for $\alpha, \beta = 1, 2, 3$, $\vec{a}_\alpha \cdot \vec{b}_\beta = \delta_{\alpha\beta}$, where $\delta_{\alpha\beta}$ is 1 if $\alpha = \beta$, and 0 otherwise.

For a system of atoms, each having a point charge and a point dipole, the electrostatic potential with periodic boundary conditions is given as

$$E^{\mathrm{el}} = \lim_{R\to\infty} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} {\sum_{|\vec{n}_r|<R}}' (q_i + \vec{d}_i \cdot \nabla_i)(q_j + \vec{d}_j \cdot \nabla_j) \frac{1}{|\vec{r}_i - \vec{r}_j + \vec{n}_r|}, \tag{2.1}$$

where q_i, $\vec{d}_i$, and $\vec{r}_i$ are the charge, dipole and position of atom i respectively, ∇_i is the gradient with respect to $\vec{r}_i$, and $\vec{n}_r$ is a lattice vector defined as $\vec{n}_r = n_1\vec{a}_1 + n_2\vec{a}_2 + n_3\vec{a}_3$, where n_1, n_2, and n_3 are integers. The prime on the summation over $\vec{n}_r$ means some terms are excluded: if $i = j$, the $\vec{n}_r = 0$ term is excluded; or if $j \in \chi(i)$, where $\chi(i)$ is the list of excluded atoms of atom i, then the interaction of the closest distance is excluded. For example, the 1–2 pairs (two atoms connected directly by a bond) and 1–3 pairs (two atoms connected to the same third atom) are often excluded. The coefficient $\frac{1}{2}$ corrects the double counting in i- and j-sum for $\vec{n}_r = 0$ terms and counts half of the Coulomb potential for $\vec{n}_r \neq 0$ terms.

Reference [40] proves that Eq. (2.1) can be written as a sum of four terms:

$$E^{\mathrm{el}} = E^{\mathrm{dir}} + E^{\mathrm{rec}} + E^{\mathrm{self}} + E^{\mathrm{surface}}, \tag{2.2}$$

where the first two terms are computationally expensive:

$$\begin{aligned} E^{\mathrm{dir}} = {} & \frac{1}{2} \sum_{i,j=1}^{N} {\sum_{\vec{n}_r}}' (q_i + \vec{d}_i \cdot \nabla_i)(q_j + \vec{d}_j \cdot \nabla_j) \frac{\mathrm{erfc}(\beta|\vec{r}_i - \vec{r}_j + \vec{n}_r|)}{|\vec{r}_i - \vec{r}_j + \vec{n}_r|} \\ & - \frac{1}{2} \sum_{i=1}^{N} \sum_{j\in\chi(i)} (q_i + \vec{d}_i \cdot \nabla_i)(q_j + \vec{d}_j \cdot \nabla_j) \frac{\mathrm{erf}(\beta|\vec{r}_i - \vec{r}_j + \vec{n}_r|)}{|\vec{r}_i - \vec{r}_j + \vec{n}_r|}, \end{aligned} \tag{2.3}$$

$$E^{\mathrm{rec}} = \frac{1}{2\pi V} \sum_{\vec{m}\neq 0} \frac{\exp(-\pi^2|\vec{m}|^2/\beta^2)}{|\vec{m}|^2} \left| \sum_{i=1}^{N} (q_i + \vec{d}_i \cdot \nabla_i) \exp\left(2\pi \mathrm{i} \vec{m} \cdot \vec{r}_i\right) \right|^2, \tag{2.4}$$

while the other two terms are computationally trivial:

$$E^{\text{self}} = -\frac{\beta}{\sqrt{\pi}}\sum_{i=1}^{N} q_i^2 - \frac{2\beta^3}{3\sqrt{\pi}}\sum_{i=1}^{N}\vec{d}_i\cdot\vec{d}_i\,, \tag{2.5}$$

$$E^{\text{surface}} = \frac{2\pi}{3V}\left|\sum_{i=1}^{N} q_i\vec{r}_i + \vec{d}_i\right|^2. \tag{2.6}$$

Here $\vec{m} = m_1\vec{b}_1 + m_2\vec{b}_2 + m_3\vec{b}_3$, where m_1, m_2, m_3 are integers, and β is a parameter adjusting the workload distributions on direct and reciprocal sums. The overall Ewald sum, with or without the surface term, is independent of β. The error function $\text{erf}(x)$ and the complementary error functions $\text{erfc}(x)$ in Eq. (2.3) are defined as

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\mathrm{d}t = \frac{2}{\sqrt{\pi}}(x - \frac{1}{3}x^3 + \cdots)\,, \tag{2.7}$$

$$\text{erfc}(x) = 1 - \text{erf}(x)\,. \tag{2.8}$$

In Eq. (2.1), it is implicitly assumed that the spherical set of boxes is surrounded by vacuum. In general, if the surrounding's relative permittivity is ϵ_{s}, the Ewald sum still has the same direct, reciprocal, and self energy terms, but the surface term is changed to [40]

$$E^{\text{surface}} = \frac{2\pi}{(2\epsilon_{\text{s}}+1)V}\left|\sum_{i=1}^{N} q_i\vec{r}_i + \vec{d}_i\right|^2. \tag{2.9}$$

For example, vacuum has $\epsilon_s = 1$, while metal has $\epsilon_s = \infty$. The surface energy is zero if $\epsilon_s = \infty$ is assumed (the "tin-foil" boundary condition). In most biomolecular simulations, the environment is water with $\epsilon_{\text{water}} \approx 80 \gg 1$, so the surface term is considered negligibly small. The surface potential is not continuous when a particle crosses the boundary, so including this term requires special care. For example, reference [115] treats $\vec{r}_i$ in the surface term as "itinerant" positions, instead of those confined in the box. Generally, the Ewald sum does not include the surface term, and in this dissertation, the surface term is neglected.

Before we represent the Ewald sum in a matrix form, we need to define some basic functions

$$g^{\mathrm{dir}}(\vec{r},\vec{r}\prime) = \sum_{\vec{n}_{\vec{r}}} \frac{\mathrm{erfc}(\beta|\vec{r}-\vec{r}\prime+\vec{n}_r|)}{|\vec{r}-\vec{r}\prime+\vec{n}_r|}\,, \tag{2.10}$$

$$g^{\mathrm{excld}}(\vec{r},\vec{r}\prime) = \begin{cases} \frac{\mathrm{erf}(\beta|\vec{r}-\vec{r}\prime+\vec{\nu}|)}{|\vec{r}-\vec{r}\prime+\vec{\nu}|} & \vec{r}\neq\vec{r}\prime \\ \frac{2\beta}{\sqrt{\pi}} & \vec{r}=\vec{r}\prime \end{cases} \tag{2.11}$$

$$g^{\mathrm{dirx}}(\vec{r},\vec{r}\prime) = \sum_{\vec{n}_{\vec{r}}\neq\vec{\nu}} \frac{\mathrm{erfc}(\beta|\vec{r}-\vec{r}\prime+\vec{n}_r|)}{|\vec{r}-\vec{r}\prime+\vec{n}_r|} - g^{\mathrm{excld}}(\vec{r},\vec{r}\prime) \tag{2.12}$$

$$g^{\mathrm{rec}}(\vec{r},\vec{r}\prime) = \frac{1}{\pi V}\sum_{\vec{m}\neq 0} \frac{\exp(-\pi^2|\vec{m}|^2/\beta^2)}{|\vec{m}|^2}\exp\left(2\pi\mathrm{i}\vec{m}\cdot(\vec{r}-\vec{r}\prime)\right). \tag{2.13}$$

Here $\vec{\nu}=\vec{\nu}(\vec{r}-\vec{r}\prime)$ is the lattice vector such that $|\vec{r}-\vec{r}\prime+\vec{\nu}|$ is minimal among all possible lattice vectors. Note that g^{excld} has no singularities. Next, we define the "direct sum" $\boldsymbol{G}$ matrices as

$$(\boldsymbol{G}_0^{\mathrm{dir}})_{ij} = \begin{cases} g^{\mathrm{dirx}}(\vec{r}_i,\vec{r}_j) & (i,j)\in\chi\,,\text{or } i=j, \\ g^{\mathrm{dir}}(\vec{r}_i,\vec{r}_j) & \text{otherwise,} \end{cases}, \tag{2.14}$$

$$(\boldsymbol{G}_1^{\mathrm{dir}})_{ij} = \begin{cases} \nabla g^{\mathrm{dirx}}(\vec{r}_i,\vec{r}_j) & (i,j)\in\chi\,,\text{or } i=j, \\ \nabla g^{\mathrm{dir}}(\vec{r}_i,\vec{r}_j) & \text{otherwise,} \end{cases}, \tag{2.15}$$

$$(\boldsymbol{G}_2^{\mathrm{dir}})_{ij} = \begin{cases} \nabla(\nabla\prime)^{\mathsf{T}} g^{\mathrm{dirx}}(\vec{r}_i,\vec{r}_j) & (i,j)\in\chi\,,\text{or } i=j, \\ \nabla(\nabla\prime)^{\mathsf{T}} g^{\mathrm{dir}}(\vec{r}_i,\vec{r}_j) & \text{otherwise,} \end{cases}, \tag{2.16}$$

where ∇ and $\nabla\prime$ are the gradients with respect to the first and second arguments of the target functions. Define the reciprocal $\boldsymbol{G}$ matrices as

$$(\boldsymbol{G}_0^{\mathrm{rec}})_{ij} = g^{\mathrm{rec}}(\vec{r}_i,\vec{r}_j)\,, \tag{2.17}$$

$$(\boldsymbol{G}_1^{\mathrm{rec}})_{ij} = \nabla g^{\mathrm{rec}}(\vec{r}_i,\vec{r}_j)\,, \tag{2.18}$$

$$(\boldsymbol{G}_2^{\mathrm{rec}})_{ij} = \nabla(\nabla\prime)^{\mathsf{T}} g^{\mathrm{rec}}(\vec{r}_i,\vec{r}_j)\,, \tag{2.19}$$

and define the overall $\boldsymbol{G}$ matrices as the sum of the direct and reciprocal $\boldsymbol{G}$ matrices:

$$\boldsymbol{G}_0 = \boldsymbol{G}_0^{\rm dir} + \boldsymbol{G}_0^{\rm rec}\,, \tag{2.20}$$

$$\boldsymbol{G}_1 = \boldsymbol{G}_1^{\rm dir} + \boldsymbol{G}_1^{\rm rec}\,, \tag{2.21}$$

$$\boldsymbol{G}_2 = \boldsymbol{G}_2^{\rm dir} + \boldsymbol{G}_2^{\rm rec}\,. \tag{2.22}$$

$\boldsymbol{G}_1$ is an $N \times N$ matrix of 3×1 blocks, each block being the Jacobian of the corresponding $\boldsymbol{G}_0$ element; $\boldsymbol{G}_2$ is an $N \times N$ matrix of 3×3 blocks, each block being the Hessian of the corresponding $\boldsymbol{G}_0$ element. So the dimension is $3N \times N$ for $\boldsymbol{G}_1$, and $3N \times 3N$ for $\boldsymbol{G}_2$. Note that $(\boldsymbol{G}_0^{\rm dir})_{ij} = (\boldsymbol{G}_0^{\rm dir})_{ji}$ and $(\boldsymbol{G}_2^{\rm dir})_{ij} = (\boldsymbol{G}_2^{\rm dir})_{ji}$. So $\boldsymbol{G}_0^{\rm dir}$ is symmetric and $\boldsymbol{G}_2^{\rm dir}$ is "block symmetric." Since $(\boldsymbol{G}_1^{\rm dir})_{i\alpha,j} = -(\boldsymbol{G}_1^{\rm dir})_{j\alpha,i}$, $\boldsymbol{G}_1^{\rm dir}$ is "block skew symmetric." The same is true for the $\boldsymbol{G}^{\rm rec}$ and $\boldsymbol{G}$ matrices. As shown in Eq. (A.13), the diagonal blocks of $\boldsymbol{G}_2$ are not zero, although they are for non-periodic systems.

For induced dipoles, the polarization energy is an energy expense that atoms must pay to have nonzero dipoles:

$$E^{\rm polar} = \frac{1}{2}\sum_{i=1}^{N} \vec{d}_i^{\,\mathsf{T}} \boldsymbol{\alpha}_i^{-1} \vec{d}_i\,. \tag{2.23}$$

Here the dipole $\vec{d}_i$ is a column vector of size 3 and the polarizability $\boldsymbol{\alpha}_i$ of atom i is a 3×3 matrix. It is diagonal for a simple model, but can be more general for complicated models [134].

Recall that $\boldsymbol{q} = [q_1, q_2, \ldots, q_N]^{\mathsf{T}}$ is the charge array, $\boldsymbol{d} = [\vec{d}_1^{\,\mathsf{T}}, \vec{d}_2^{\,\mathsf{T}}, \ldots, \vec{d}_N^{\,\mathsf{T}}]^{\mathsf{T}}$ is the dipole array, and $\boldsymbol{D}_\alpha = \mathrm{diag}(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_N)$ is the block diagonal polarizability matrix. If we add in the polarization energy and neglect the surface term, the Ewald sum is

$$E^{\rm Ewald}(\boldsymbol{r}, \boldsymbol{d}) = \frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\boldsymbol{G}_0(\boldsymbol{r})\boldsymbol{q} + \boldsymbol{d}^{\mathsf{T}}\boldsymbol{G}_1(\boldsymbol{r})\boldsymbol{q} + \frac{1}{2}\boldsymbol{d}^{\mathsf{T}}\boldsymbol{G}_2(\boldsymbol{r})\boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^{\mathsf{T}}\boldsymbol{D}_\alpha^{-1}\boldsymbol{d}\,, \tag{2.24}$$

Note that the self energy $E^{\rm self}$ has been absorbed into the direct sum.

In practice, the summation over the direct lattice ($\vec{n}_r$) and the reciprocal lattice ($\vec{m}$) will be truncated when the terms are negligibly small. For a given accuracy requirement $\epsilon\,\text{Å}^{-1}$, the cutoff

criteria used in this dissertation are

$$1\text{Å} \times \frac{\operatorname{erfc}(\beta r_{\mathrm{c}})}{r_{\mathrm{c}}} \quad \leq \quad \epsilon\,, \tag{2.25}$$

$$\exp\left(-\frac{\pi^2 m_{\mathrm{c}}^2}{\beta^2}\right) \quad \leq \quad \epsilon\,. \tag{2.26}$$

The first appears in reference [140], and the second in reference [48]. For the reciprocal sum, assuming the simulation box is cubic of size $L \geq 1\,\text{Å}$, then

$$1\text{Å} \times \frac{1}{2\pi V m_{\mathrm{c}}^2} \exp\left(-\frac{\pi^2 m_{\mathrm{c}}^2}{\beta^2}\right) \quad \leq \quad 1\text{Å} \times \frac{1}{L} \exp\left(-\frac{\pi^2 m_{\mathrm{c}}^2}{\beta^2}\right) \leq \exp\left(-\frac{\pi^2 m_{\mathrm{c}}^2}{\beta^2}\right) \leq \epsilon \tag{2.27}$$

So Eq. (2.26) is a conservative criterion.

The induced dipoles take the values that minimizes the total electrostatic energy,

$$\frac{\partial}{\partial \boldsymbol{d}} E^{\mathrm{Ewald}}(\boldsymbol{r}, \boldsymbol{d}) \quad = \quad 0\,, \tag{2.28}$$

which gives the equation for the dipole moments

$$(\boldsymbol{D}_\alpha^{-1} + \boldsymbol{G}_2)\boldsymbol{d} \quad = \quad -\boldsymbol{G}_1 \boldsymbol{q}\,. \tag{2.29}$$

To make things clear, we define $\boldsymbol{d}^*(\boldsymbol{r}) = (\boldsymbol{D}_\alpha^{-1} + \boldsymbol{G}_2(\boldsymbol{r}))^{-1}(-\boldsymbol{G}_1(\boldsymbol{r})\boldsymbol{q})$, which shows the dependency of $\boldsymbol{d}$ on $\boldsymbol{r}$ explicitly, and $\tilde{E}(\boldsymbol{r}) = E^{\mathrm{Ewald}}(\boldsymbol{r}, \boldsymbol{d}(\boldsymbol{r}))$. In the following force derivation, $\boldsymbol{d}$ is the value while $\boldsymbol{d}^*$ is the function:

$$\begin{aligned}
F_{k\sigma} &= -\frac{\partial}{\partial r_{k\sigma}} \tilde{E}(\boldsymbol{r}) \\
&= -\frac{\partial}{\partial r_{k\sigma}} E^{\mathrm{Ewald}}(\boldsymbol{r}, \boldsymbol{d}) - \left(\frac{\partial}{\partial \boldsymbol{d}} E^{\mathrm{Ewald}}(\boldsymbol{r}, \boldsymbol{d})\right)^{\mathsf{T}} \frac{\partial \boldsymbol{d}^*(\boldsymbol{r})}{\partial r_{k\sigma}} \\
&= -\frac{\partial}{\partial r_{k\sigma}} E^{\mathrm{Ewald}}(\boldsymbol{r}, \boldsymbol{d}) \\
&= -\frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_0\right)\boldsymbol{q} - \boldsymbol{d}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_1\right)\boldsymbol{q} - \frac{1}{2}\boldsymbol{d}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_2\right)\boldsymbol{d}\,.
\end{aligned} \tag{2.30}$$

The three terms in Eq. (2.30) are the charge–charge, charge–dipole, and dipole–dipole force, respectively. With the optimal dipole vector satisfying (2.29), the energy given by (2.24) simplifies

to

$$E^{\text{Ewald}}(\boldsymbol{r}) \;=\; \frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\boldsymbol{G}_0(\boldsymbol{r})\boldsymbol{q} + \frac{1}{2}\boldsymbol{d}(\boldsymbol{r})^{\mathsf{T}}\boldsymbol{G}_1(\boldsymbol{r})\boldsymbol{q}\,. \tag{2.31}$$

2.2 Particle–mesh Ewald method

In this dissertation, the particle–mesh Ewald (PME) method refers to the Smooth PME (SPME) method [44], which is preferable to the original PME [39], not only because it is faster, but also because the force computed by SPME is exactly the negative gradient of the (approximated) potential. For Hamiltonian system simulations, this property along with symplectic integration will make sure the long-time drift of the Hamiltonian is minimal.

The Ewald sum has two major parts: direct sum and reciprocal sum. Correspondingly, the $\boldsymbol{G}$ matrices, energy, and force can be split as follows according to Eq. (2.31):

$$\begin{aligned}
E^{\text{Ewald}} &= E^{\text{dir}} + E^{\text{rec}}\,, && (2.32)\\
E^{\text{dir}} &= \frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\boldsymbol{G}_0^{\text{dir}}\boldsymbol{q} + \frac{1}{2}\boldsymbol{d}^{\mathsf{T}}\boldsymbol{G}_1^{\text{dir}}\boldsymbol{q}\,, && (2.33)\\
E^{\text{rec}} &= \frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\boldsymbol{G}_0^{\text{rec}}\boldsymbol{q} + \frac{1}{2}\boldsymbol{d}^{\mathsf{T}}\boldsymbol{G}_1^{\text{rec}}\boldsymbol{q}\,, && (2.34)\\
\boldsymbol{F} &= \boldsymbol{F}^{\text{dir}} + \boldsymbol{F}^{\text{rec}}\,, && (2.35)\\
\boldsymbol{F}_{k\sigma}^{\text{dir}} &= -\frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_0^{\text{dir}}\right)\boldsymbol{q} - \boldsymbol{d}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_1^{\text{dir}}\right)\boldsymbol{q} - \frac{1}{2}\boldsymbol{d}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_2^{\text{dir}}\right)\boldsymbol{d}\,, && (2.36)\\
\boldsymbol{F}_{k\sigma}^{\text{rec}} &= -\frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_0^{\text{rec}}\right)\boldsymbol{q} - \boldsymbol{d}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_1^{\text{rec}}\right)\boldsymbol{q} - \frac{1}{2}\boldsymbol{d}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_2^{\text{rec}}\right)\boldsymbol{d}\,. && (2.37)
\end{aligned}$$

Note that in PME, the Ewald parameter β is intentionally chosen large so that for direct sum, a small cut-off radius is needed for a given accuracy (see Eq. (2.25)). This leads to sparse $\boldsymbol{G}^{\text{dir}}$ matrices and an $O(N)$ computational cost for the direct sum. For the reciprocal sum computation, fast Fourier transforms (FFTs) are used and the computation cost is $O(N \log N)$. The implementation of the direct sum in PME is described in Section 2.2.1, and here we focus on the reciprocal sum.

2.2.1 Direct sum implementation

The computation cost of the direct sum can be surprisingly high for the PME method in which the major work is expected to be done in the reciprocal sum. One reason is that the Ewald sum and the Lennard Jones interactions are often computed simultaneously, and the latter requires a large cutoff radius.

From Eqs. (2.30) and (2.35), we have

$$\begin{aligned} F_{k\sigma}^{\rm dir} &= -\sum_{j=1}^{N} q_k q_j (\boldsymbol{G}_1^{\rm dir})_{k\sigma,j} - \sum_{j=1}^{N} \sum_{\alpha=x,y,z} (d_{j\alpha} q_k - d_{k\alpha} q_j)(\boldsymbol{G}_2^{\rm dir})_{k\sigma,j\alpha} \\ &\quad - \sum_{j=1}^{N} \sum_{\alpha,\beta=x,y,z} d_{k\alpha} d_{j\beta} G_{3,k\sigma\alpha,j\beta}^{\rm dir} \,, \end{aligned} \tag{2.38}$$

where

$$G_{3,k\sigma\alpha,j\beta}^{\rm dir} = \frac{\partial}{\partial r_{k\sigma}} (\boldsymbol{G}_2^{\rm dir})_{k\alpha,j\beta} \,. \tag{2.39}$$

In most Ewald sum computations, the direct sum cutoff radius is less than half of the system size. This means the summation in Eqs. (2.10) and (2.12) has only one term. Following [128, 140], we define

$$B_0(r) = \frac{\operatorname{erfc}(\beta r)}{r} \,, \tag{2.40}$$

$$B_k(r) = -\frac{1}{r} \frac{\mathrm{d}B_k(r)}{\mathrm{d}r} \,, \qquad k = 1, 2, 3, \tag{2.41}$$

$$\bar{B}_0(r) = B_0(r) - \frac{1}{r} \,, \tag{2.42}$$

$$\bar{B}_k(r) = -\frac{1}{r} \frac{\mathrm{d}\bar{B}_k(r)}{\mathrm{d}r} \,, \qquad k = 1, 2, 3. \tag{2.43}$$

It can see shown

$$B_k(r) = \frac{1}{r^2} \big[(2k-1) B_{k-1}(r) + \frac{(2\beta^2)^k}{\beta\sqrt{\pi}} \exp(-\beta^2 r^2) \big] \qquad k = 1, 2, 3, \tag{2.44}$$

$$\bar{B}_k(r) = \frac{1}{r^2} \big[(2k-1) \bar{B}_{k-1}(r) + \frac{(2\beta^2)^k}{\beta\sqrt{\pi}} \exp(-\beta^2 r^2) \big] \qquad k = 1, 2, 3. \tag{2.45}$$

If (i,j) is not an excluded pair, then

$$
\begin{aligned}
(\boldsymbol{G}_0^{\mathrm{dir}})_{ij} &= B_0(r)\,, && (2.46)\\
(\boldsymbol{G}_1^{\mathrm{dir}})_{ij} &= -B_1(r)\cdot\vec{r}, && (2.47)\\
(\boldsymbol{G}_2^{\mathrm{dir}})_{ij} &= B_1(r)\cdot\boldsymbol{I} - B_2(r)\cdot\vec{r}\cdot\vec{r}^{\mathsf{T}}\,, && (2.48)\\
(\boldsymbol{G}_3^{\mathrm{dir}})_{i\sigma\alpha,j\beta} &= B_3(r)\cdot r_\sigma r_\alpha r_\beta - B_2(r)\cdot(r_\sigma\delta_{\alpha\beta} + r_\alpha\delta_{\beta\sigma} + r_\beta\delta_{\alpha\sigma})\,. && (2.49)
\end{aligned}
$$

For clarity, we have dropped the subscript ij of $\vec{r}$, so $\vec{r}$ is actually $\vec{r}_{ij} = \vec{r}_i - \vec{r}_j + \vec{\nu}$, a column vector of size 3 in Eqs. (2.46)–(2.49). Also, r represents the 2-norm of $\vec{r}$.

If (i,j) is an excluded pair, we have similar expressions:

$$
\begin{aligned}
(\boldsymbol{G}_0^{\mathrm{dir}})_{ij} &= \bar{B}_0(r)\,, && (2.50)\\
(\boldsymbol{G}_1^{\mathrm{dir}})_{ij} &= -\bar{B}_1(r)\cdot\vec{r}, && (2.51)\\
(\boldsymbol{G}_2^{\mathrm{dir}})_{ij} &= \bar{B}_1(r)\cdot\boldsymbol{I} - \bar{B}_2(r)\cdot\vec{r}\cdot\vec{r}^{\mathsf{T}}\,, && (2.52)\\
(\boldsymbol{G}_3^{\mathrm{dir}})_{i\sigma\alpha,j\beta} &= \bar{B}_3(r)\cdot r_\sigma r_\alpha r_\beta - \bar{B}_2(r)\cdot(r_\sigma\delta_{\alpha\beta} + r_\alpha\delta_{\beta\sigma} + r_\beta\delta_{\alpha\sigma})\,. && (2.53)
\end{aligned}
$$

Unlike the permanent multipole (including charge-only model) computations, the matrix $\boldsymbol{G}_2$ is used repeatedly when solving the dipole equation iteratively. Re-computation of $\boldsymbol{G}_2$ for each iteration is expensive, so we need to store it in some way to save time and cost.

In an implementation, each atom has a list of neighboring atoms whose distance is less than the direct sum cut-off. For each neighboring pair, the corresponding B_1, B_2, and B_3 are computed and stored for later use. We do not store $\vec{r}_{ij}$s, but compute them when needed. In this way, most re-computations are avoided while memory is used judiciously.

The above implementation requires an extra storage of approximately $\frac{1}{2}\cdot 3N\frac{4}{3}\pi\rho r_{\mathrm{c}}^3$ doubles for storing B_1, B_2 and B_3 for each pair, and $\frac{1}{2}\cdot N\frac{4}{3}\pi\rho r_{\mathrm{c}}^3$ integers for maintaining a neighbor list, where ρ is the average number of atoms per unit volume. The coefficient $\frac{1}{2}$ appears because we count each pair only once. A typical double-type datum takes 8 bytes, while an integer-type datum takes

4 bytes. So typical storage in bytes is

$$\frac{1}{2}\cdot\frac{4}{3}\pi\rho r_{\rm c}^3\cdot 3N\cdot 8+\frac{1}{2}\cdot\frac{4}{3}\pi\rho r_{\rm c}^3 N\cdot 4=\frac{56}{3}\pi\rho r_{\rm c}^3\cdot N\,. \tag{2.54}$$

The typical density for a biomolecular system is $\rho\approx 0.1$ atoms/Å^3. For $r_{\rm c}=8\,$Å, the memory requirement is $3003N$ bytes, and for $r_{\rm c}=10\,$Å, the memory requirement is less than $6000N$ bytes. For a system with N =10,000 atoms, this requires only 30 or 60 megabytes of memory, respectively. Current personal computers have memory in the range of several hundred megabytes. For larger systems, parallel computation is needed. For a typical parallel computation setting, each processor has about 1,000–10,000 atoms. For example, NAMD recommends 1,000 atoms per CPU [105] for optimal efficiency. The above memory requirement is not a problem .

To avoid the "polarization catastrophe" [6, 138], some point dipole models have a short-range damping term which takes effect only when the distance of two dipoles is less than a small screening distance. The implementation of this term in the direct sum computation is straightforward and similar to the handling of "excluded pairs." Since our test models do not suffer the polarization catastrophe in practice, we do not concern ourselves about it in this dissertation.

2.2.2 Reciprocal sum

Two approximations are made in PME for the reciprocal sum computation. The first approximation truncates the infinite sum in the reciprocal sum to a summation over a cubic region of the reciprocal lattice, not a sphere as in the standard Ewald summation. The second approximation interpolates sines and cosines from a uniform grid, which we can do because the range of wave numbers is restricted.

The *first* approximation of PME is the following cubic truncation:

$$g^{\rm rec}(\vec{r},\vec{r}\,')\approx\frac{1}{\pi V}\sum_{m_1=-K_1/2}^{K_1/2-1}\sum_{m_2=-K_2/2}^{K_2/2-1}\sum_{m_3=-K_3/2}^{K_3/2-1}\frac{\exp(-\pi^2|\vec{m}|^2/\beta^2)}{|\vec{m}|^2}\exp\left(2\pi{\rm i}\vec{m}\cdot(\vec{r}-\vec{r}\,')\right), \tag{2.55}$$

where $\vec{m}=m_1\vec{b}_1+m_2\vec{b}_2+m_3\vec{b}_3$, m_1, m_2, and m_3 are integers, K_1, K_2, and K_3 are some large even integers so that the truncation error is negligible.

Corresponding to the truncated reciprocal lattice, we have a grid in the real space and there are

K_1, K_2, and K_3 grid points along each dimension. This motivates the following "u-representation" for a position vector $\vec{r}$:

$$\vec{r} \;=\; \vec{r}_{\rm corner} + [\vec{a}_1\ \vec{a}_2\ \vec{a}_3] \begin{bmatrix} u_1/K_1 \\ u_2/K_2 \\ u_3/K_3 \end{bmatrix}, \tag{2.56}$$

where $\vec{r}_{\rm corner}$ is the position of a corner of the simulation box, which is chosen so that $0 \le u_\alpha < K_\alpha$ for $\alpha = 1, 2, 3$ constitutes the simulation box. Equivalently,

$$\vec{u} \;=\; \begin{bmatrix} K_1\vec{b}_1^{\mathsf{T}} \\ K_2\vec{b}_2^{\mathsf{T}} \\ K_3\vec{b}_3^{\mathsf{T}} \end{bmatrix} (\vec{r} - \vec{r}_{\rm corner}) = \boldsymbol{T}(\vec{r} - \vec{r}_{\rm corner})\,. \tag{2.57}$$

where $\vec{b}_1$, $\vec{b}_2$, and $\vec{b}_3$ are the reciprocal lattice basis vectors.

To present the second approximation made by the PME method, we first introduce the B-spline functions and the periodic B-spline functions. The B-spline function $M_p(u)$ is defined recursively as

$$M_1(u) \;=\; \begin{cases} 1\,, & 0 \le u < 1\,, \\ 0\,, & \text{otherwise}\,, \end{cases} \tag{2.58}$$

$$M_p(u) \;=\; \frac{u}{p-1} M_{p-1}(u) + \frac{p-u}{p-1} M_{p-1}(u-1)\,, \qquad \text{for } p > 1. \tag{2.59}$$

The first few B-spline functions are shown in Fig. 2.2.

The B-spline functions have the following properties [44]:

$$(1) \qquad M_p(u) > 0\,, \text{ if } 0 < u < p, \quad M_p(u) = 0\,, \text{ otherwise,} \tag{2.60}$$

$$(2) \qquad \sum_{n=-\infty}^{\infty} M_p(u-n) = 1\,, \tag{2.61}$$

$$(3) \qquad M_p'(u) = M_{p-1}(u) - M_{p-1}(u-1), \quad \text{for } p > 1\,. \tag{2.62}$$

The first two properties imply B-spline functions can be considered as discrete weight functions.

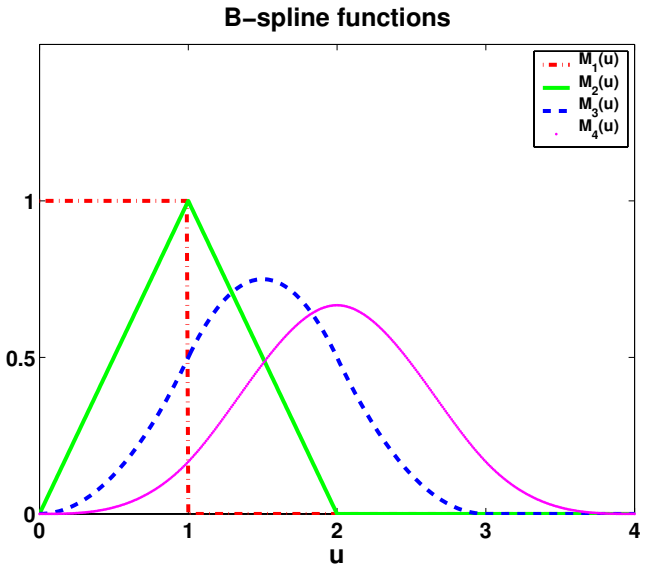

Figure 2.2: The first few B-spline functions.

The third property enables us to get the analytical force expression.

The periodic B-spline functions are defined as

$$\Phi_K(u) \;=\; \sum_{i=-\infty}^{\infty} M_p(u - iK)\,. \tag{2.63}$$

The function has period K because $\Phi_K(u+K) = \Phi(u)$. The local support of M_p reduces the seemingly infinite sum in (2.63) to a finite sum of no more than p terms. In fact, real applications have $K \gg p$, and the summation over i has at most one nonzero term.

The basic interpolation idea is the following:

$$\begin{aligned}
\exp\left(2\pi \mathrm{i}\frac{mu}{K}\right) &\approx b_K(m) \sum_{n=-\infty}^{\infty} M_p(u-n)\exp\left(2\pi \mathrm{i}\frac{mn}{K}\right) \\
&= b_K(m) \sum_{n=0}^{K-1} \Phi_K(u-n)\exp\left(2\pi \mathrm{i}\frac{mn}{K}\right), \qquad (2.64) \\
b_K(m) &= \frac{\exp\left(2\pi \mathrm{i} m(p-1)/K\right)}{\sum_{n=0}^{p-2} M_p(n+1)\cdot \exp\left(2\pi \mathrm{i} mn/K\right)}, \qquad (2.65)
\end{aligned}$$

where m is an integer, $0 \le m < K$, and $b_K(m)$ is a normalization constant. The B-spline functions are generally not nodal basis functions for spline interpolation, but the factor of $b_K(m)$ makes possible the interpolation of the sines and cosines by the B-spline functions. In Eq. (2.64), the

summation has about p nonzero terms.

The *second* approximation made by PME is the following interpolation:

$$\begin{aligned}\exp\left(2\pi \mathrm{i}\vec{m}\cdot(\vec{r}-\vec{r}_{\mathrm{corner}})\right) &= \exp\left(2\pi\mathrm{i}\left[\frac{u_1 m_1}{K_1}+\frac{u_2 m_2}{K_2}+\frac{u_3 m_3}{K_3}\right]\right) \\ &\approx b_{\vec{K}}(\vec{m})\sum_{\vec{n}}\varphi_{\vec{n}}(\vec{u})\exp(2\pi\mathrm{i}[\frac{m_1 n_1}{K_1}+\frac{m_2 n_2}{K_2}+\frac{m_3 n_3}{K_3}])\,, \end{aligned} \quad (2.66)$$

where

$$b_{\vec{K}}(\vec{m}) = b_{K_1}(m_1)b_{K_2}(m_2)b_{K_3}(m_3)\,, \quad (2.67)$$

$$\varphi_{\vec{n}}(\vec{u}) = \Phi_{K_1}(u_1-n_1)\Phi_{K_2}(u_2-n_2)\Phi_{K_3}(u_3-n_3)\,. \quad (2.68)$$

So

$$\exp\left(2\pi\mathrm{i}\vec{m}\cdot(\vec{r}_i-\vec{r}_{\mathrm{corner}})\right) \approx (\boldsymbol{I}_h^0\boldsymbol{F}\boldsymbol{B})_{i\vec{m}}\,, \quad (2.69)$$

with

$$(\boldsymbol{I}_h^0)_{i\vec{n}} = \varphi_{\vec{n}}(\vec{u}_i)\,, \quad (2.70)$$

$$\boldsymbol{F}_{\vec{n},\vec{m}} = \exp\left(2\pi\mathrm{i}\left[\frac{m_1 n_1}{K_1}+\frac{m_2 n_2}{K_2}+\frac{m_3 n_3}{K_3}\right]\right), \quad (2.71)$$

$$\boldsymbol{B}_{\vec{m},\vec{m}} = b_{\vec{K}}(\vec{m})\,. \quad (2.72)$$

Here, $\boldsymbol{I}_h^0$ is a matrix of size $N\times K$, where $K=K_1K_2K_3$, $\boldsymbol{F}$ is the Fourier transform matrix of size $K\times K$, and $\boldsymbol{B}$ is a diagonal matrix of size $K\times K$. $\boldsymbol{I}_h^0$ is a prolongation operator which interpolates grid values to particle positions. Correspondingly, $(\boldsymbol{I}_h^0)^{\mathsf{T}}$ restricts data at particle positions onto the grid (see Fig. 2.3). Reference [39] and [44] point out that the grid size should be proportional to the average distance between atoms to have a reasonable accuracy for molecular dynamics simulations. This means $K=K_1K_2K_3=O(N)$, which generally guarantees that the accuracy requirement in Eq. (2.26) is satisfied.

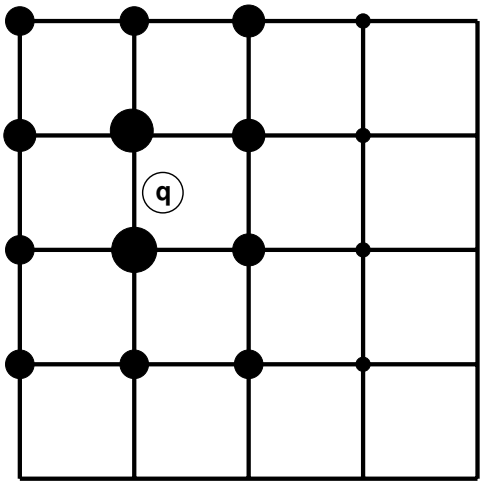

Figure 2.3: In PME, a charge, represented by an empty circle with name q, is mapped (restricted) to the nodes of a uniform grid.

$\boldsymbol{I}_h^0$ is *sparse* due to the local support of the B-spline basis functions. From Eq. (2.61), we see

$$\begin{aligned}\sum_{\vec{n}}(\boldsymbol{I}_h^0)_{i\vec{n}} &= \sum_{n_1}\Phi_{K_1}(u_{i1}-n_1)\cdot\sum_{n_2}\Phi_{K_2}(u_{i2}-n_2)\cdot\sum_{n_3}\Phi_{K_3}(u_{i3}-n_3)\\ &= \sum_{i_1=-\infty}^{\infty}M_p(u_{i1}-i_1)\sum_{i_2=-\infty}^{\infty}M_p(u_{i2}-i_2)\sum_{i_3=-\infty}^{\infty}M_p(u_{i3}-i_3)\\ &= 1\cdot 1\cdot 1 = 1\,. \end{aligned} \tag{2.73}$$

$(\boldsymbol{I}_h^0)_{i\vec{n}}$ is the weight function, representing how much of the charge q_i is distributed to grid node $\vec{n}$. The above expression just means the sum of the weights is equal to 1.

From Eqs. (2.17), (2.13), and (2.69), we have

$$(\boldsymbol{G}_0^{\rm rec})_{ij} \approx \sum_{m_1'=-K_1/2}^{K_1/2-1}\sum_{m_2'=-K_2/2}^{K_2/2-1}\sum_{m_3'=-K_3/2}^{K_3/2-1}(\boldsymbol{I}_h^0\boldsymbol{F}\boldsymbol{B})_{i,\vec{m}'}U(\vec{m}')(\boldsymbol{I}_h^0\boldsymbol{F}\boldsymbol{B})^*_{j\vec{m}'}\,, \tag{2.74}$$

where

$$U(\vec{m}) = \frac{1}{\pi V}\frac{\exp(-\pi^2|\vec{m}|^2/\beta^2)}{|\vec{m}|^2}\,, \qquad \vec{m}\neq 0\,, \tag{2.75}$$

$$U(0) = 0\,, \qquad \vec{m} = 0\,. \tag{2.76}$$

The summation in Eq. (2.74) is not $[0:K_1-1]\times[0:K_2-1)\times[0:K_3-1]$ needed by the FFT.

The discrepancy is easily fixed by the following: For each $\vec{m}'$, we define a corresponding vector $\vec{m}$, the components of which are

$$m_\alpha \;=\; \begin{cases} m'_\alpha\,, & m'_\alpha \geq 0\,, \\ m'_\alpha + K_\alpha\,, & m'_\alpha < 0\,. \end{cases} \tag{2.77}$$

So we have $0 \leq m_\alpha < K_\alpha$, $\boldsymbol{F}_{\vec{m}',\vec{n}} = \boldsymbol{F}_{\vec{m},\vec{n}}$, $\boldsymbol{B}_{\vec{m}',\vec{m}'} = \boldsymbol{B}_{\vec{m},\vec{m}}$, and $(\boldsymbol{I}_h^0\boldsymbol{F}\boldsymbol{B})_{j\vec{m}'} = (\boldsymbol{I}_h^0\boldsymbol{F}\boldsymbol{B})_{j\vec{m}}$. Define diagonal $\boldsymbol{D}$ as

$$\boldsymbol{D} \;=\; |\boldsymbol{B}|^2 U(\vec{m}')\,. \tag{2.78}$$

Eq. (2.74) gives

$$(\boldsymbol{G}_0^{\rm rec})_{ij} \approx (\boldsymbol{I}_h^0\boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}(\boldsymbol{I}_h^0)^{\mathsf{T}})_{ij}, \quad \text{or} \quad \boldsymbol{G}_0^{\rm rec} \approx \boldsymbol{I}_h^0\boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}(\boldsymbol{I}_h^0)^{\mathsf{T}}\,. \tag{2.79}$$

Similarly, Eq. (2.55) gives

$$g^{\rm rec}(\vec{r},\vec{r}\prime) \;=\; \sum_{\vec{n}}\sum_{\vec{n}\prime} \varphi_{\vec{n}}(\vec{u})(\boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{T}})_{\vec{n}\vec{n}\prime}\varphi_{\vec{n}\prime}(\vec{u}\prime)\,. \tag{2.80}$$

To get the expression for $\boldsymbol{G}_1^{\rm rec}$, $\boldsymbol{G}_2^{\rm rec}$, and later, energy and force, define

$$(\boldsymbol{I}_h^1)_{i\vec{n}} \;=\; \nabla_i(\boldsymbol{I}_h^0)_{i\vec{n}} = \boldsymbol{T}^{\mathsf{T}}(D\varphi_{\vec{n}}(\vec{u}_i))\,, \tag{2.81}$$

$$(\boldsymbol{I}_h^2)_{i\vec{n}} \;=\; \nabla_i\nabla_i{}^{\mathsf{T}}(\boldsymbol{I}_h^0)_{i\vec{n}} = \boldsymbol{T}^{\mathsf{T}}(D^2\varphi_{\vec{n}}(\vec{u}_i))\boldsymbol{T}\,, \tag{2.82}$$

where the second equalities follow from Eqs. (2.70) and (2.57), and $D\varphi_{\vec{n}}$ and $D^2\varphi_{\vec{n}}$ are the Jacobian

vector and the Hessian matrix of the basis function $\varphi_{\vec{n}}$, which are given by (see Eq. (2.68))

$$D\varphi_{\vec{n}}(\vec{u}) = \begin{bmatrix} \Phi_1'\Phi_2\Phi_3 \\ \Phi_1\Phi_2'\Phi_3 \\ \Phi_1\Phi_2\Phi_3' \end{bmatrix}, \tag{2.83}$$

$$D^2\varphi_{\vec{n}}(\vec{u}) = \begin{bmatrix} \Phi_1''\Phi_2\Phi_3 & \Phi_1'\Phi_2'\Phi_3 & \Phi_1'\Phi_2\Phi_3' \\ \Phi_1'\Phi_2'\Phi_3 & \Phi_1\Phi_2''\Phi_3 & \Phi_1\Phi_2'\Phi_3' \\ \Phi_1'\Phi_2\Phi_3' & \Phi_1\Phi_2'\Phi_3' & \Phi_1\Phi_2\Phi_3'' \end{bmatrix}, \tag{2.84}$$

where

$$\Phi_\alpha'(x) = \sum_{i=-\infty}^{\infty} M_p'(x - iK_\alpha) = \sum_{i=-\infty}^{\infty} \big(M_{p-1}(x - iK_\alpha) - M_{p-1}(x - iK_\alpha - 1)\big), \tag{2.85}$$

$$\Phi_\alpha''(x) = \sum_{i=-\infty}^{\infty} \big(M_{p-2}(x - iK_\alpha) - 2M_{p-2}(x - iK_\alpha - 1) + M_{p-2}(x - iK_\alpha - 2)\big). \tag{2.86}$$

So each "element" of $\boldsymbol{I}_h^1$ is a 3×1 block, and each "element" of $\boldsymbol{I}_h^2$ is a 3×3 block.

From Eqs. (2.79), (2.70), (2.81), (2.21), (2.22), and (2.80), we have

$$\boldsymbol{G}_0^{\mathrm{rec}} \approx \boldsymbol{I}_h^0(\boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}})(\boldsymbol{I}_h^0)^{\mathsf{T}}, \tag{2.87}$$

$$\boldsymbol{G}_1^{\mathrm{rec}} \approx \boldsymbol{I}_h^1(\boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}})(\boldsymbol{I}_h^0)^{\mathsf{T}}, \tag{2.88}$$

$$\boldsymbol{G}_2^{\mathrm{rec}} \approx \boldsymbol{I}_h^1(\boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}})(\boldsymbol{I}_h^1)^{\mathsf{T}}. \tag{2.89}$$

The matrix–vector multiplication for each $\boldsymbol{G}_i^{\mathrm{rec}}$ is simple and direct from the above expressions. For example, to compute $\boldsymbol{G}_2^{\mathrm{rec}}\boldsymbol{d}$, we first restrict (distribute) $\boldsymbol{d}$ to grid nodes, then do a backward FFT, then multiply by the diagonal matrix $\boldsymbol{D}$, then do a forward FFT, and, in the end, interpolate back to real space.

We define the following quantities:

$$\text{grid charges}: \qquad \boldsymbol{q}_h = (\boldsymbol{I}_h^0)^{\mathsf{T}} \boldsymbol{q}\,, \tag{2.90}$$

$$\text{grid dipoles}: \qquad \boldsymbol{d}_h = (\boldsymbol{I}_h^1)^{\mathsf{T}} \boldsymbol{d}\,, \tag{2.91}$$

$$\text{sum of both}: \qquad \boldsymbol{s}_h = \boldsymbol{q}_h + \boldsymbol{d}_h\,, \tag{2.92}$$

$$\text{grid potential}: \qquad \boldsymbol{v}_h = \boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}} \boldsymbol{s}_h\,. \tag{2.93}$$

Here, $\boldsymbol{q}_h$, $\boldsymbol{d}_h$, $\boldsymbol{s}_h$, and $\boldsymbol{v}_h$ are vectors of size K. From Eq. (2.73), the summations of the components of $\boldsymbol{q}_h$ and $\boldsymbol{d}_h$ are 0:

$$\sum_{\vec{n}} (\boldsymbol{q}_h)_{\vec{n}} = \sum_{\vec{n}} \sum_i (\boldsymbol{I}_h^0)_{i\vec{n}} q_i = \sum_i q_i \sum_{\vec{n}} (\boldsymbol{I}_h^0)_{i\vec{n}} = \sum_i q_i \cdot 1 = 0\,, \tag{2.94}$$

$$\sum_{\vec{n}} (\boldsymbol{d}_h)_{\vec{n}} = \sum_{\vec{n}} \sum_{i\alpha} (\boldsymbol{I}_h^1)_{i\alpha\vec{n}} d_{i\alpha} = \sum_{i\alpha} d_{i\alpha} \frac{\partial}{\partial r_{i\alpha}} \sum_{\vec{n}} (\boldsymbol{I}_h^0)_{i\vec{n}} = \sum_{i\alpha} d_{i\alpha} \frac{\partial}{\partial r_{i\alpha}} 1 = 0\,. \tag{2.95}$$

The first one is true because the system is charge-neutral.

From Eqs. (2.34), (2.87), (2.88), (2.90)–(2.93), the reciprocal energy is

$$\begin{aligned} E^{\mathrm{rec}} &= \frac{1}{2}\boldsymbol{q}^{\mathsf{T}} \boldsymbol{G}_0^{\mathrm{rec}} \boldsymbol{q} + \frac{1}{2}\boldsymbol{d}^{\mathsf{T}} \boldsymbol{G}_1^{\mathrm{rec}} \boldsymbol{q} \\ &= \frac{1}{2}\boldsymbol{q}^{\mathsf{T}} \boldsymbol{G}_0^{\mathrm{rec}} \boldsymbol{q} + \frac{1}{2}\boldsymbol{q}^{\mathsf{T}} (\boldsymbol{G}_1^{\mathrm{rec}})^{\mathsf{H}} \boldsymbol{d} \\ &\approx \frac{1}{2}\boldsymbol{q}^{\mathsf{T}} \boldsymbol{I}_h^0 (\boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}) (\boldsymbol{I}_h^0)^{\mathsf{T}} \boldsymbol{q} + \frac{1}{2}\boldsymbol{q}^{\mathsf{T}} \boldsymbol{I}_h^0 (\boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}) (\boldsymbol{I}_h^1)^{\mathsf{T}} \boldsymbol{d} \\ &= \frac{1}{2}\boldsymbol{q}_h{}^{\mathsf{T}} \boldsymbol{v}_h\,. \end{aligned} \tag{2.96}$$

Next, we derive the force expression. First, from Eqs. (2.24), (2.17), (2.18), and (2.19), we have

$$\begin{aligned} \boldsymbol{F}_k^{\mathrm{rec}} &= -\nabla_k \left(\frac{1}{2}\boldsymbol{q}^{\mathsf{T}} \boldsymbol{G}_0^{\mathrm{rec}} \boldsymbol{q} + \boldsymbol{d}^{\mathsf{T}} \boldsymbol{G}_1^{\mathrm{rec}} \boldsymbol{q} + \frac{1}{2}\boldsymbol{d}^{\mathsf{T}} \boldsymbol{G}_2^{\mathrm{rec}} \boldsymbol{d}\right) \\ &= -\nabla_k \frac{1}{2} \sum_{ij} (q_i + \vec{d}_i \cdot \nabla)(q_j + \vec{d}_j \cdot \nabla') g^{\mathrm{rec}}(\vec{r}_i, \vec{r}_j) \\ &= -\nabla_k (q_k + \vec{d}_k \cdot \nabla) \sum_j (q_j + \vec{d}_j \cdot \nabla') g^{\mathrm{rec}}(\vec{r}_k, \vec{r}_j)\,. \end{aligned} \tag{2.97}$$

Then, from Eqs. (2.87), and (2.90)–(2.93),

$$\begin{aligned}
\boldsymbol{F}_k^{\mathrm{rec}} &\approx -\nabla_k(q_k+\vec{d}_k\cdot\nabla_k)\sum_j(q_j+\vec{d}_j\cdot\nabla_j)(\boldsymbol{I}_h^0\boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}(\boldsymbol{I}_h^0)^{\mathsf{T}})_{kj}\\
&= -\nabla_k(q_k+\vec{d}_k\cdot\nabla_k)\sum_{\vec{n}\vec{n}'}(\boldsymbol{I}_h^0)_{k\vec{n}}(\boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}})_{\vec{n}\vec{n}'}\sum_j(q_j+\vec{d}_j\cdot\nabla_j)(\boldsymbol{I}_h^0)_{j\vec{n}'}\\
&= -\nabla_k(q_k+\vec{d}_k\cdot\nabla_k)\sum_{\vec{n}}(\boldsymbol{I}_h^0)_{k\vec{n}}\sum_{\vec{n}'}(\boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}})_{\vec{n}\vec{n}'}(\boldsymbol{s}_h)_{\vec{n}'}\\
&= -\sum_{\vec{n}}\Big(\nabla_k(q_k+\vec{d}_k\cdot\nabla_k)(\boldsymbol{I}_h^0)_{k\vec{n}}\Big)(\boldsymbol{v}_h)_{\vec{n}}\,. \qquad (2.98)
\end{aligned}$$

In the end, from Eqs. (2.81) and (2.82),

$$\boldsymbol{F}_k^{\mathrm{rec}} \approx -\sum_{\vec{n}}\Big(q_k(\boldsymbol{I}_h^1)_{k\vec{n}}+(\boldsymbol{I}_h^2)_{k\vec{n}}\boldsymbol{d}_k\Big)\cdot(\boldsymbol{v}_h)_{\vec{n}}\,. \qquad (2.99)$$

The cost for computing the energy and force, assuming the dipole is known, is counted as follows: computing $\boldsymbol{q}_h$, $\boldsymbol{d}_h$, and $\boldsymbol{s}_h$ from (2.90)–(2.92) costs $O(N)$ since $\boldsymbol{I}_h^0$ and $\boldsymbol{I}_h^1$ are sparse; computing $\boldsymbol{v}_h$ from (2.93) needs two FFTs; computing the energy from (2.96) and force from (2.99) costs $O(N)$ since for a given value of k, $(\boldsymbol{I}_h^1)_{k\vec{n}}\neq 0$ and $(\boldsymbol{I}_h^2)_{k\vec{n}}\neq 0$ for only p^3 values of $\vec{n}$.

A very nice feature of PME shows up in Eqs. (2.87)–(2.89): the $\boldsymbol{G}$ matrices are decomposed in such a way that the dependence on position is separated from the other part of the reciprocal sum computation. Taking the gradient of the $\boldsymbol{G}$ matrices becomes easy and straightforward. This is why generalizing PME to high-level multipole computation does not incur much extra cost. In PME, the force computation (computing the gradient) and the higher-degree multipole interaction (also computing the gradient) can be properly arranged so that additional computation is minimal.

Notice from Eqs. (2.87)–(2.89) that although $\boldsymbol{G}_0^{\mathrm{rec}}$ and $\boldsymbol{G}_2^{\mathrm{rec}}$ are still symmetric after the interpolation approximation, $\boldsymbol{G}_1^{\mathrm{rec}}$ is not "block skew symmetric" anymore. The consequence is the loss of linear momentum conservation. Fig. 2.4 shows that the energy and the linear momentum cannot both be conserved by a simulation employing the PME method. The momentum drift is measured against the thermal momentum of water molecules. The thermal momentum p is computed by the equipartition theorem $\frac{1}{2}p^2/m=\frac{3}{2}k_{\mathrm{B}}T$, where m is the mass of a water molecule, and $T=300\,\mathrm{K}$. Because the PME method makes an interpolation error in Eq. (2.66), the sum of all the forces con-

tributed by the reciprocal sum is not zero but a small number instead. If this small extra force is subtracted out [140] to conserve linear momentum, then the force is not the exact negative gradient of the potential any more, and this leads to a small energy drift for long simulations. Other fast electrostatic solvers, including the multilevel summation method [126, 20] and the particle–particle particle–mesh method [67], cannot conserve energy and momentum simultaneously either.

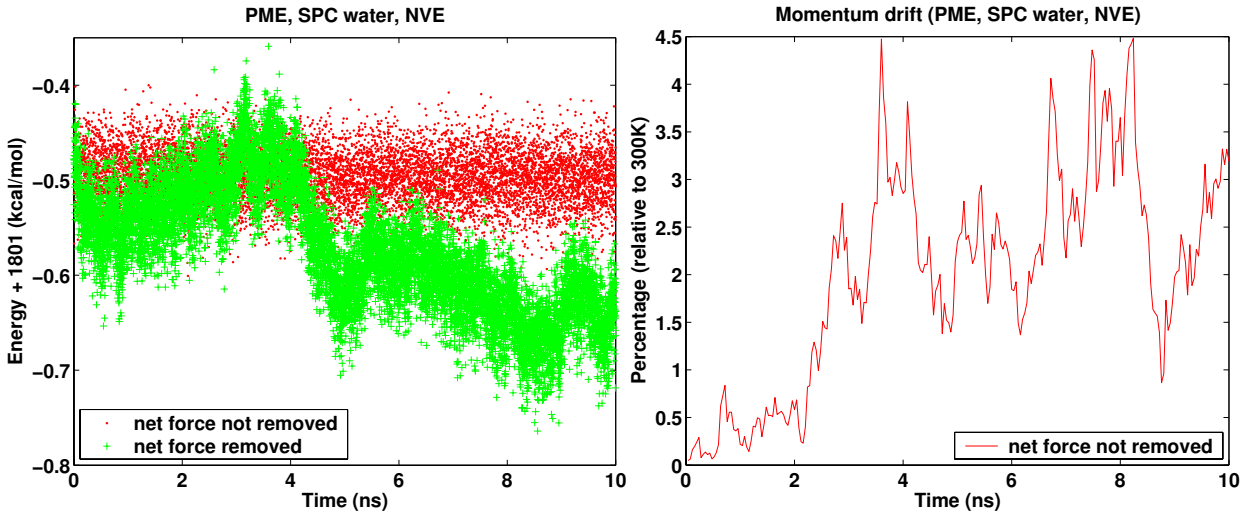

Figure 2.4: The energy and momentum of a system of 216 SPC [14] water molecules.

2.2.3 Overall computation sequence

The computation has three major steps: preparation, solving the dipole equation iteratively, and computing the electrostatic energy and the electrostatic force.

The iteration scheme should be formulated carefully to avoid unnecessary cost. The Picard iteration

$$\boldsymbol{d}_{n+1} \quad = \quad \boldsymbol{D}_\alpha \big[-\boldsymbol{G}_1 \boldsymbol{q} - \boldsymbol{G}_2 \boldsymbol{d}_n \big] \,, \tag{2.100}$$

for example, is formulated as follows to save two FFTs by not explicitly computing $-\boldsymbol{G}_1^{\mathrm{rec}} \boldsymbol{q}$:

$$\boldsymbol{d}_{n+1} \quad = \quad -\boldsymbol{D}_\alpha \big(\boldsymbol{G}_1^{\mathrm{dir}} \boldsymbol{q} + \boldsymbol{G}_2^{\mathrm{dir}} \boldsymbol{d}_n + \boldsymbol{I}_h^1 F \boldsymbol{D} \boldsymbol{F}^{\mathsf{H}} [(\boldsymbol{I}_h^0)^{\mathsf{T}} \boldsymbol{q} + (\boldsymbol{I}_h^1)^{\mathsf{T}} \boldsymbol{d}_n] \big) \,, \tag{2.101}$$

or for $n > 1$,

$$\boldsymbol{d}_{n+1} \;=\; \boldsymbol{d}_n - \boldsymbol{D}_\alpha\big[\boldsymbol{G}_2^{\rm dir}(\boldsymbol{d}_n - \boldsymbol{d}_{n-1}) + \boldsymbol{I}_h^1\boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}(\boldsymbol{I}_h^1)^{\mathsf{T}}(\boldsymbol{d}_n - \boldsymbol{d}_{n-1})\big]\,. \tag{2.102}$$

The sequencing of the computation is summarized in the following:

1. preparation

   (a) direct sum: compute and store $\frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\boldsymbol{G}_0^{\rm dir}\boldsymbol{q}$ (scalar), $-\boldsymbol{G}_1^{\rm dir}\boldsymbol{q}$ (vector). Compute and store the B_1, B_2, and B_3. Since $\boldsymbol{G}_0^{\rm dir}$ is sparse, the operation and storage cost is $O(N)$.

   (b) reciprocal sum: compute and store $\boldsymbol{q}_h = (\boldsymbol{I}_h^0)^{\mathsf{T}}\boldsymbol{q}$ (vector), and partially compute and store $\boldsymbol{I}_h^1$ and $\boldsymbol{I}_h^2$: for each atom and each dimension, maintain arrays of size $3p$ storing M_p, M_p', and M_p''. This accounts for $9pN$ double-type data, and requires $72pN$ bytes on most platforms. Since p is 4 or 6 in most MD simulations, the memory requirement is affordable. M_p is computed recursively from order 1 up to p. According to Eq. (2.62), M_p', and M_p'' are computed simultaneously as M_p is computed and the computation incurs little extra cost. Since $\boldsymbol{I}_h^0$, $\boldsymbol{I}_h^1$, and $\boldsymbol{I}_h^2$ are sparse, the computation cost is $O(N)$.

2. solving the dipole equation

   (a) use some predictor to compute the initial guess. The cost is $O(N)$.

   (b) use some iterative method to solve the dipole equation. For example, for Picard iteration (2.100), use Eq. (2.101). The cost for each step is 2 FFTs.

3. computing the electrostatic energy and the electrostatic force

   (a) direct sum: compute $\frac{1}{2}\boldsymbol{d}^{\mathsf{T}}\boldsymbol{G}_1^{\rm dir}\boldsymbol{q}$, add it to $\frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\boldsymbol{G}_0^{\rm dir}\boldsymbol{q}$ to get $E^{\rm dir}$. Compute force $\boldsymbol{F}^{\rm dir}$ according to Eq. (2.30). The cost is $O(N)$.

   (b) reciprocal sum: compute $\boldsymbol{d}_h = (\boldsymbol{I}_h^1)^{\mathsf{T}}\boldsymbol{d}$, $\boldsymbol{s}_h = \boldsymbol{q}_h + \boldsymbol{d}_h$, and $\boldsymbol{v}_h = \boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}\boldsymbol{s}_h$ (2 FFTs). Then compute $E^{\rm rec}$ and $\boldsymbol{F}^{\rm rec}$ according to Eqs. (2.96) and (2.99).

   (c) the electrostatic energy is $E^{\rm dir} + E^{\rm rec}$, the electrostatic force is $\boldsymbol{F}^{\rm dir} + \boldsymbol{F}^{\rm rec}$.

The total number of FFTs is $2 + 2\times$ number of iterations.

Chapter 3

Self-consistent solution

Given an accuracy (convergence) requirement, how can we solve the dipole equation (2.29) as fast as possible? This chapter tackles two aspects of the problem: an accurate initial guess and quickly converging iterations.

An accurate initial guess is pursued in Section 3.1. The continuity of a molecular dynamics trajectory provides the foundation for making a good initial guess. Previous predictors [53, 1, 117, 21, 31, 135, 140] are mostly based on the polynomial extrapolation. Since the dipole is a continuous function of time through position, polynomial extrapolation predictor assumes a Taylor expansion in time exists at the current timestep.

Given the fact that the numerical integrator, the velocity-Verlet method, is only second order accurate, it is natural to ask if the prediction can be good if the polynomial degree is higher than two. However, as backward error analysis shows, the velocity-Verlet integrator solves a nearby Hamiltonian system to a very high degree [110, 43]. The numerical solution we are computing can be regarded as a smooth function of time restricted at discrete timesteps, so it can be interpolated by a polynomial.

Although the reference [140] uses only degree-1 polynomial prediction, we demonstrate that predictions using polynomials of degree four or five provide a much more accurate initial guess. However, high-degree polynomial extrapolation has two problems: (1) it is ill conditioned, meaning a small error in the previous dipoles is magnified by the prediction, and (2) the quality can degrade significantly due to the "Runge phenomenon" [35, §4.3.4]. In practice, the accuracy of the polynomial predictor degrades when the degree reaches five or six. It is also undesirable that the optimal degree depend significantly on the iteration method and related parameters. So we propose a new

predictor which uses least squares fitting of previous data to predict the dipole. We show that it is more accurate and it is numerically more stable than polynomial prediction.

Quickly converging iteration methods are proposed in Section 3.2. As has been pointed out in Chapter 1, we should take advantage of the matrix $\boldsymbol{D}_\alpha^{-1} + \boldsymbol{G}_2$ being symmetric positive definite. By using the conjugate gradient (CG) or Chebyshev semi-iterative method with a novel preconditioner, we can reduce the number of iterations significantly.

Timing result shows, for a 1 fs timestep, the self-consistent approach leads to an extra cost of 94% compared to a charge-only model. When a longer timestep is used, the self-consistent approach is more efficient than the extended Lagrangian approach.

The widely used root-mean-square (RMS) error for an iteration method is defined as

$$\text{RMS error} \quad = \quad \frac{1}{\sqrt{N}} \|\boldsymbol{d}_m - \boldsymbol{d}_{m-1}\|_2 \,, \tag{3.1}$$

where N is the number of atoms and m is the iteration step. Computing the RMS error is an inexpensive $O(N)$ calculation. For a fast converging iteration, the RMS error is probably a better estimation of the RMS norm of the *previous* solution error than the current one since

$$\|\boldsymbol{d}_{m-1} - \boldsymbol{d}_{\text{exact}}\|_2 \leq \|\boldsymbol{d}_{m-1} - \boldsymbol{d}_m\|_2 + \|\boldsymbol{d}_m - \boldsymbol{d}_{\text{exact}}\|_2 \approx \|\boldsymbol{d}_{m-1} - \boldsymbol{d}_m\|_2 \,. \tag{3.2}$$

For this reason, RMS error can badly over-estimate the true error. For example, if the initial guess is not accurate enough, at least two iterations are needed in general, no matter how fast the iterative method converges.

In this chapter, convergence is claimed if the RMS error is less than 10^{-6} or 10^{-7} Debye (1 Debye $= 3.33564 \times 10^{-30}$ Coulomb $\times$ meter). But the relative error is probably easier to interpret. Since the average dipole moment of an atom in the RPOL [36] water system is 0.25 Debye from Eq. (A.3), the above convergence criteria is 4 or 0.4 parts per million (ppm).

3.1 Initial guess

Degree-$(k-1)$ polynomial extrapolation uses k ($k \geq 1$) previous dipole moments to predict the dipole moment at timestep n:

$$\boldsymbol{d}_0^n = \sum_{i=1}^{k} (-1)^{i+1} \binom{k}{i} \boldsymbol{d}^{n-i} . \tag{3.3}$$

The superscripts represent timesteps. The first few formulas for polynomial extrapolation are as follows:

Degree	$\boldsymbol{d}_0^n$
0	$\boldsymbol{d}^{n-1}$
1	$2\boldsymbol{d}^{n-1} - \boldsymbol{d}^{n-2}$
2	$3\boldsymbol{d}^{n-1} - 3\boldsymbol{d}^{n-2} + \boldsymbol{d}^{n-3}$
3	$4\boldsymbol{d}^{n-1} - 6\boldsymbol{d}^{n-2} + 4\boldsymbol{d}^{n-3} - \boldsymbol{d}^{n-4}$
4	$5\boldsymbol{d}^{n-1} - 10\boldsymbol{d}^{n-2} + 10\boldsymbol{d}^{n-3} - 5\boldsymbol{d}^{n-4} + \boldsymbol{d}^{n-5}$
$\vdots$	$\vdots$

We examine the accuracy of the predictions by their relative errors. Define

$$\delta d = \frac{\|\boldsymbol{d} - \boldsymbol{d}_{\text{exact}}\|_2}{\|\boldsymbol{d}_{\text{exact}}\|_2} . \tag{3.4}$$

Table 3.1 shows the prediction error in terms of δd computed for some snapshots drawn from a molecular dynamics simulation. The numbers change from case to case, but the trend remains about the same.

degree	0	1	2	3	4	5	6	7	8
δd (ppm)	1.6e4	2.2e3	4.9e2	1.4e2	58	26	29	53	99

Table 3.1: The polynomial extrapolation error.

A good initial guess reduces the computational cost significantly (see Fig. 3.1 and 3.2; the various iterative methods will be described in Section 3.2). However, numerical instability degrades the results when the degree is greater than five. A rough estimate of the error magnification effect is

given as follows:

$$\|\boldsymbol{\epsilon}_0^n\| = \left\| \sum_{i=1}^{k} (-1)^{i+1} \binom{k}{i} \boldsymbol{\epsilon}^{n-i} \right\| \quad \leq \quad \max_{1 \leq i \leq k} \|\boldsymbol{\epsilon}^{n-i}\| \cdot \sum_{i=1}^{k} \binom{k}{i} = (2^k - 1) \max_{1 \leq i \leq k} \|\boldsymbol{\epsilon}^{n-i}\| \,, \qquad (3.5)$$

where $\|\cdot\|$ denotes some norm. A degree–5 ($k = 6$) polynomial prediction could amplify the error by a factor of 63. Since the convergence factor is about 0.34 (see Fig. 3.3), four Picard iterations would be needed to compensate such error growth.

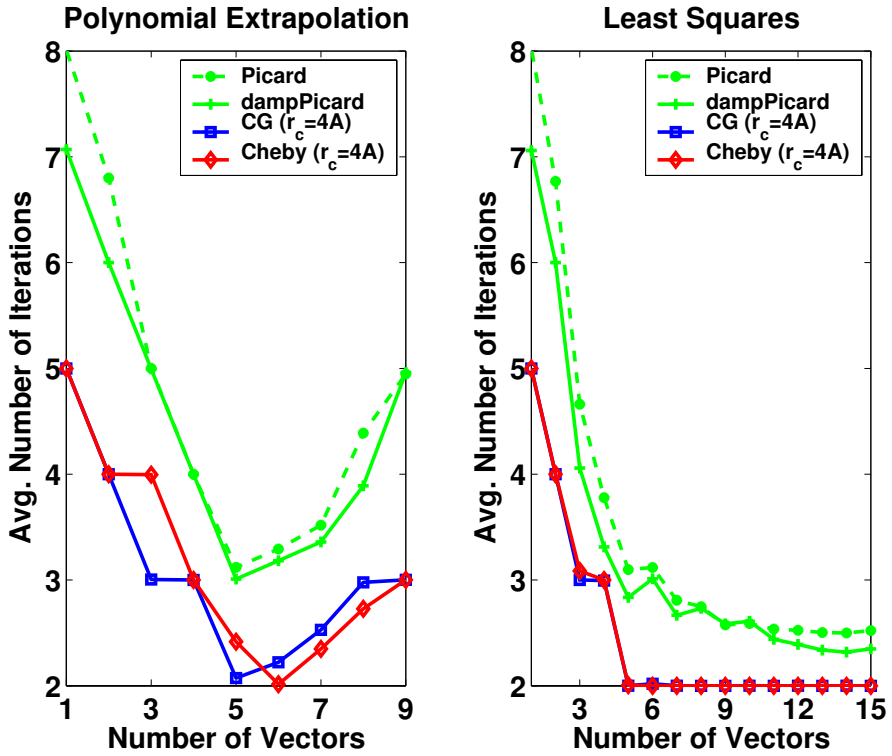

Figure 3.1: Average number of iterations for different methods. The superfluous RMS convergence criteria is 4 ppm.

The least squares prediction arises by asking the following question after the dipole $\boldsymbol{d}^n$ is computed: what is the best prediction of $\boldsymbol{d}^n$ we could have made from a linear combination of $\boldsymbol{d}^{n-1}, \ldots, \boldsymbol{d}^{n-k}$? The answer is obtained by choosing coefficients $c_1, c_2, \ldots, c_k$ that minimize the objective function

$$T \quad = \quad \|\boldsymbol{d}^n - \sum_{i=1}^{k} c_i \boldsymbol{d}^{n-i}\|_2^2 \,. \qquad (3.6)$$

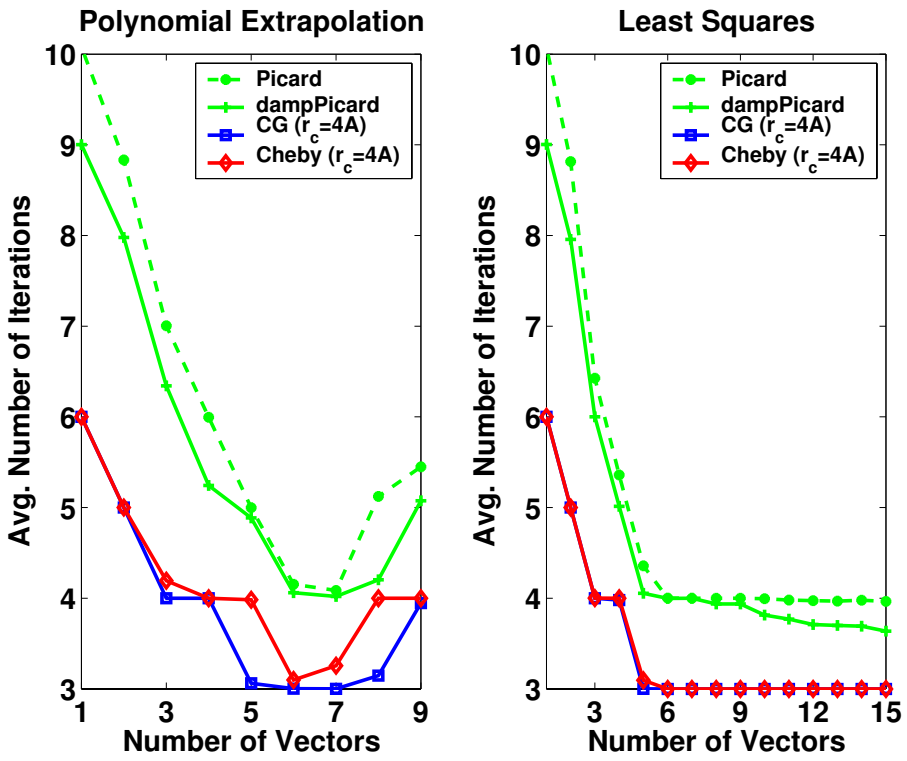

Figure 3.2: Average number of iterations for different methods. The superfluous RMS convergence criteria is 0.4 ppm.

Numerical methods solving this least squares problem has been discussed in textbooks such as [141]. The fastest method is the normal equation method, in which we obtain the optimal coefficients $c_1, \ldots, c_k$ by solving the equations

$$\frac{\partial T}{\partial c_i} = 0 \quad \Rightarrow \quad \sum_{j=1}^{k} (\boldsymbol{d}^{n-i})^{\mathsf{T}} \boldsymbol{d}^{n-j} c_j = (\boldsymbol{d}^{n-i})^{\mathsf{T}} \boldsymbol{d}^{n} \,, \tag{3.7}$$

Eq. (3.7) is k equations for k unknowns $(c_1, \ldots, c_k)$, and can be solved by Gaussian elimination. After we compute these coefficients, we use them to predict the dipole at the next timestep:

$$\boldsymbol{d}^{n+1}_{0,\text{least squares}} \quad = \quad \sum_{i=1}^{k} c_i \boldsymbol{d}^{n+1-i} \,. \tag{3.8}$$

Other more numerically stable methods for solving the least squares problem include the QR factorization and the singular value decomposition methods [141]. We have also implemented the QR factorization method with column pivoting, but find that numerical instability is not a serious

issue in most cases.

A closer look at the data shows that least squares prediction is a little better than the optimal polynomial prediction:

$$\delta d_{\text{optimal, polynomial}} \approx 2 \times 10^{-5}\,, \tag{3.9}$$

$$\delta d_{\text{optimal, least squares}} \approx 1 \times 10^{-5}\,. \tag{3.10}$$

Least squares prediction incurs a negligibly small amount of extra work at each step: in our test cases, the prediction cost is less than 1% of the overall electrostatic computation. we need to do k inner products of vectors, and solve a $k \times k$ linear system, where k is a small number, generally no larger than 10. Extra memory to store $k-1$ previous vectors is also needed.

3.2 Iteration method

Iterative methods can be classified into two categories: stationary iterative methods, such as the Jacobi, Picard, and Gauss-Seidel method; and non-stationary iterative methods, such as the conjugate gradient (CG) method and the Chebyshev semi-iterative method.

Stationary methods split the matrix into two matrices, one of which is easy to invert. For our problem, there is little choice but to split the left-hand-side matrix into $\boldsymbol{D}_\alpha^{-1}$ and $\boldsymbol{G}_2$, and use the following Picard iteration:

$$\boldsymbol{d}_{m+1} = \boldsymbol{D}_\alpha(-\boldsymbol{G}_1\boldsymbol{q} - \boldsymbol{G}_2\boldsymbol{d}_m)\,. \tag{3.11}$$

We do not call it Jacobi iteration because the diagonal elements of $\boldsymbol{G}_2$, although small, are not zero (see Eq. (A.13)). The natural breakup into a simple dominant part plus a lesser part leads us to call it Picard iteration. This iteration is used widely in the literature to solve the dipole equation probably for two reasons. First, it is simple and has a clear intuitive meaning: the dipole is proportional to the electric field with the linear "coefficient" $\boldsymbol{D}_\alpha$, while the electric field comes from charges ($-\boldsymbol{G}_1\boldsymbol{q}$) and other dipoles ($-\boldsymbol{G}_2\boldsymbol{d}$). Second, the eigenvalues of the iteration matrix $\boldsymbol{D}_\alpha\boldsymbol{G}_2$ cluster around zero, as shown in Fig. 3.3. If we decompose the error vector into components

each parallel to an eigenvector, those components whose corresponding eigenvalues are close to zero are damped away after one Picard iteration. So the method is fairly efficient for solving the dipole equation.

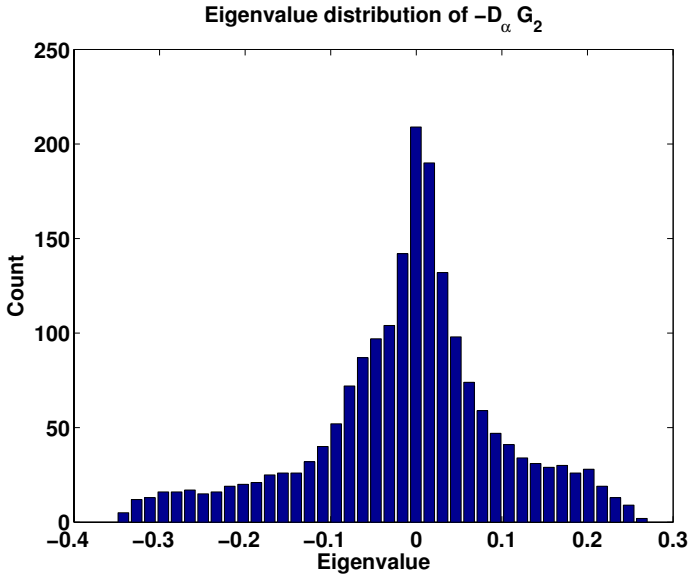

Figure 3.3: Eigenvalue distribution of matrix $-\boldsymbol{D}_\alpha \boldsymbol{G}_2$.

The convergence factor of a stationary method is determined by the spectral radius of the iteration matrix [65]. For Picard iteration, the spectral radius $\rho(-\boldsymbol{D}_\alpha \boldsymbol{G}_2)$ is about 0.34 (see Fig. 3.3). If the spectrum of the iteration matrix is not symmetric about zero, a damping scheme can be used:

$$\boldsymbol{d}_{m+1} \quad = \quad \omega \boldsymbol{D}_\alpha(-\boldsymbol{G}_1\boldsymbol{q} - \boldsymbol{G}_2\boldsymbol{d}_m) + (1-\omega)\boldsymbol{d}_m \,, \tag{3.12}$$

where ω is the damping factor, whose optimal value can be determined by the largest and smallest eigenvalues of the iteration matrix:

$$\omega_{\rm opt} \quad = \quad \frac{2}{2-(\lambda_{\rm max}+\lambda_{\rm min})} \,. \tag{3.13}$$

The convergence factor is

$$r \quad = \quad \frac{\lambda_{\rm max}-\lambda_{\rm min}}{2-(\lambda_{\rm max}+\lambda_{\rm min})} \,. \tag{3.14}$$

Given a typical value of $\lambda_{\max} \approx 0.26$, $\lambda_{\min} \approx -0.34$, the convergence factor for damped Picard method with optimal damping factor is 0.29: damping helps only a little.

Direct inversion in the iterative subspace (DIIS) method [109] is a popular acceleration method in *ab initio* molecular dynamics simulations. For a stationary iteration method, after a few iterations, the error is dominated by components corresponding to largest absolute eigenvalues, and the convergence slows down. At this point in the computation, DIIS accelerates the convergence by making a least squares approximation to zero in the linear space $\{x^i\}_{i=0}^n$. However, we find that by the time that DIIS does better than the Picard method, after about five iterations, the solution is already good enough to be considered converged. So we do not explore DIIS further.

The conjugate gradient (CG) [41, §8.3] or Chebyshev semi-iterative method [41, §8.2] are two non-stationary methods for solving symmetric positive-definite linear systems. CG is a popular method that minimizes the energy norm of the error, while the Chebyshev method, with optimal parameters, is targeted at minimizing the 2-norm of the error. The error is bounded by [41]

$$\frac{\|\boldsymbol{d}_m - \boldsymbol{d}_{\text{exact}}\|}{\|\boldsymbol{d}_0 - \boldsymbol{d}_{\text{exact}}\|} \quad \leq \quad 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^m, \tag{3.15}$$

where $\kappa = \lambda_{\max}/\lambda_{\min}$ is the condition number of the preconditioned matrix and $\|\cdot\|$ is the energy norm for CG and the 2-norm for the Chebyshev method.

It takes two matrix–vector multiplications for the standard CG method to get the first update of the solution: one for computing the residual to determine the search direction, another for computing the optimal distance to move along the search direction. After that, each iteration requires one matrix–vector multiplication. So for the same number of updates of the solution, CG does one more (expensive) matrix–vector multiplication than other methods. This extra cost can be saved by a "peek" step which uses the available residual to do one Picard iteration:

$$\boldsymbol{d}' \quad = \quad \boldsymbol{d} + \boldsymbol{D}_\alpha \boldsymbol{r} = \boldsymbol{D}_\alpha(-\boldsymbol{G}_1\boldsymbol{q} - \boldsymbol{G}_2\boldsymbol{d})\,, \tag{3.16}$$

where $\boldsymbol{r}$ is the residual. The following pseudo code for the modified CG method follows reference [41,

§8.3]:

$$
\begin{aligned}
\boldsymbol{r} &:= -\boldsymbol{G}_1\boldsymbol{q} - (\boldsymbol{D}_\alpha^{-1} + \boldsymbol{G}_2)\boldsymbol{d};\\
\text{solve } \boldsymbol{M}\boldsymbol{s} &= \boldsymbol{r} \text{ for } \boldsymbol{s};\\
c &:= \boldsymbol{r}^\mathsf{T}\boldsymbol{s};\\
\text{FOR } \mathit{iter} &:= 1, 2, \ldots, \mathit{maximum\_iteration}\\
&\quad \boldsymbol{u} := (\boldsymbol{D}_\alpha^{-1} + \boldsymbol{G}_2)\boldsymbol{s};\\
&\quad a := \frac{c}{\boldsymbol{s}^\mathsf{T}\boldsymbol{u}};\\
&\quad \boldsymbol{d} := \boldsymbol{d} + a \cdot \boldsymbol{s}; \quad \boldsymbol{r} := \boldsymbol{r} - a \cdot \boldsymbol{u};\\
/* \text{ peek } */ &\quad \boldsymbol{d}' := \boldsymbol{d} + \boldsymbol{D}_\alpha\boldsymbol{r}; \quad \text{IF } (\|\boldsymbol{d}' - \boldsymbol{d}\|_2^2 < \epsilon^2) \text{ BREAK};\\
&\quad \text{solve } \boldsymbol{M}\boldsymbol{t} = \boldsymbol{r} \text{ for } \boldsymbol{t};\\
&\quad c_{\text{new}} := \boldsymbol{r}^\mathsf{T}\boldsymbol{t}; \quad b := \frac{c_{\text{new}}}{c}; \quad c := c_{\text{new}};\\
&\quad \boldsymbol{s} := \boldsymbol{t} + b \cdot \boldsymbol{s};\\
\text{END } \mathit{iter};&\\
\text{claim } \boldsymbol{d}' &\text{ as the solution.}
\end{aligned}
$$

Here $\boldsymbol{s}$ is the search direction, $\boldsymbol{M}$ is a preconditioner, $\boldsymbol{t}$ and $\boldsymbol{u}$ are temporary vectors, ϵ is the convergence criteria, a is a scalar marking the optimal position along the direction $\boldsymbol{s}$, and b, c, and c_{new} are scalars. Compared to the standard CG implementation, the only change is the added "peek" step. This inexpensive $O(N)$ computation does not alter the CG search path, but allows us to find a converged solution one step earlier than the standard CG method in most cases. It does not mean to peek at the next solution computed by the CG method. In fact, we replace the last CG step by a suboptimal but acceptable solution.

The straightforward implementation of the Chebyshev method follows the description in reference [41, §8.2]. The method requires a good estimation of the spectrum range, which, according to our experience, changes little during MD simulations. So this expensive computation can be done once for all, and the long-time performance suffers little.

3.3 Preconditioner

A preconditioner is an easy-to-invert approximation to the left-hand-side matrix and is used to reduce the condition number of that matrix, since Eq. (3.15) indicates that reducing the condition number accelerates the convergence. This section first presents a preconditioner constructed by the "local approximation" idea [141, page 317] and then provides a "polynomial approximation" [141, page 318] to the inverse of the local approximation preconditioner, which effectively solves the preconditioned problem $\boldsymbol{M}\boldsymbol{t} = \boldsymbol{r}$ by a single matrix–vector multiplication.

For Eq. (2.29), an obvious local approximation to $\boldsymbol{D}_\alpha^{-1} + \boldsymbol{G}_2$ is the matrix $\boldsymbol{M} = \boldsymbol{D}_\alpha^{-1} + \boldsymbol{N}$ with

$$\boldsymbol{N}_{ij} \quad = \quad \begin{cases} \boldsymbol{T}(\vec{r}_{ij}) & i \neq j, |\vec{r}_{ij}| < r_\mathrm{c}, \text{ and } (i,j) \notin \chi \\ 0 & \text{otherwise} \end{cases} , \tag{3.17}$$

where χ is the set of excluded pairs used in Chapter 2, the cutoff radius r_c is a parameter, and $\boldsymbol{T}(r)$ is the dipole–dipole interaction tensor:

$$\boldsymbol{T}(\vec{r}) \quad = \quad \frac{1}{r^3}(\boldsymbol{I} - 3\frac{\vec{r}\vec{r}^\mathsf{T}}{r^2}) , \tag{3.18}$$

where $\vec{r}$ is a column vector of size 3, r is the 2-norm of $\vec{r}$, and $\boldsymbol{I}$ is the identity matrix of size 3. For a dipole, the above preconditioner considers only nearby dipoles whose distance is within the cutoff radius, since they have the most significant influence. We can consult the radial distribution functions to determine r_c. From Fig. A.2 , we see 3 Å is just after the first peak of the O–H, H–H, and O–O radial distribution functions. Table 3.2 provides the convergence factor for a typical matrix arising from molecular dynamics simulations.

Method	condition number	convergence factor
Picard	$--$	0.34
vanilla CG,Chebyshev (Cheby)	4.31	0.35
CG,Cheby preconditioned by $\boldsymbol{M}$ with $r_\mathrm{c} = 0$ Å	1.80	0.15
CG,Cheby preconditioned by $\boldsymbol{M}$ with $r_\mathrm{c} = 3$ Å	1.44	0.09
CG,Cheby preconditioned by $\boldsymbol{M}$ with $r_\mathrm{c} = 4$ Å	1.38	0.08

Table 3.2: Convergence factors of different iteration methods.

On average, each atom has less than 6 neighbors whose distances are less than 3 Å for the

RPOL water system. Since the intra-molecular electrostatic interaction is excluded, we only need to maintain a list whose average length is less than 4 for each atom. If the cutoff is 4 Å, the list has an average length of less than 12.

It is important that the effectiveness of the preconditioner should not depend on the system size, in other words, that the preconditioner "scales." For a cutoff radius larger than 4 Å, the gain in reducing the number of iteration is less significant, but the cost in solving the preconditioner problem, which is proportional to r_c^3, increases significantly.

Long distance dipole effects can be incorporated into the preconditioner through reaction field approximations [94]: all atoms outside a cutoff radius are approximated by a continuum media, and are represented as a dielectric constant. A dipole induces dipoles in the continuum, which interacts with other dipoles whose distances are within the cutoff sphere. The dipole–dipole interaction tensor is modified to [94]:

$$\boldsymbol{T}_{\mathrm{RF}}(r) \;=\; \boldsymbol{T}(r) + \frac{2(\epsilon_{\mathrm{RF}}-1)}{2\epsilon_{\mathrm{RF}}+1}\frac{1}{|r_c|^3}\boldsymbol{I}\,, \qquad r < r_c\,, \tag{3.19}$$

where ϵ_{RF} is the dielectric constant. For water, its value is about 80. However, this correction does not give perceptible improvement.

The next question is how to quickly solve $\boldsymbol{M}\boldsymbol{s} = \boldsymbol{r}$ for $\boldsymbol{s}$. We do not have to solve it exactly and solving it approximately is equivalent to using another preconditioner close to $\boldsymbol{M}$. The following expansion allows us use a polynomial to approximate $\boldsymbol{M}^{-1}$ directly

$$\boldsymbol{M}^{-1} \;=\; (\boldsymbol{I}+\boldsymbol{D}_\alpha\boldsymbol{N})^{-1}\boldsymbol{D}_\alpha = \boldsymbol{D}_\alpha - \boldsymbol{D}_\alpha\boldsymbol{N}\boldsymbol{D}_\alpha + (\boldsymbol{D}_\alpha\boldsymbol{N})^2\boldsymbol{D}_\alpha - \cdots\,. \tag{3.20}$$

We can truncate at a certain point and include the terms before that. More terms being included means better approximation to $\boldsymbol{M}^{-1}$. The truncation matrix is symmetric, a desirable feature of our design. The above preconditioners are simple and easy to implement since only matrix–vector multiplications are needed. In practice, for $r_c = 3$ Å, or 4 Å, including the first two terms is as good as $\boldsymbol{M}^{-1}$ in reducing the number of iterations. Eq. (3.20) can be improved by better expansions used in Chapter 5. But the improvement makes the parameters model dependent.

Reference [93] also uses preconditioners when solving the self-consistent equation by the CG

method for a fluctuating charge model when the underlying fast electrostatic solver is PME and FMM. But these preconditioners limit themselves to the use of existing software modules. For the PME method, reference [93] uses $\boldsymbol{D}_\alpha^{-1} + \boldsymbol{G}_2^{\mathrm{dir}}$ as the preconditioner. But in our tests, $\boldsymbol{D}_\alpha^{-1} + \boldsymbol{G}_2^{\mathrm{dir}}$ is not effective in reducing the condition number of the left-hand-side matrix, nor it is computationally efficient to solve $(\boldsymbol{D}_\alpha^{-1} + \boldsymbol{G}_2^{\mathrm{dir}})\boldsymbol{s} = \boldsymbol{r}$. For the FMM method, the preconditioner used by reference [93] is similar to ours. But the cutoff radius in [93] is 6 Å, the size of a leaf cell in the fast multipole method. Our tests show using a cutoff radius of larger than 4 Å does not reduce the number of iterations for the CG method but increases the computation cost significantly for solving the preconditioned problem. Reference [93] solves the preconditioner problem also by the CG method, which needs global synchronization in a parallel environment and can slow down the computation.

Since the preconditioner problem $\boldsymbol{M}\boldsymbol{t} = \boldsymbol{r}$ is not easy to solve approximately, we have considered several widely used techniques other than the polynomial approximation. Incomplete Cholesky factorizations is not scalable [10] for parallelism. The "approximate inverse" method [10] tries to find an approximate inverse of the left-hand-side matrix by reducing the Frobenius norm. But it may not be easy to make the approximate inverse symmetric in a parallel computing environment. The block diagonal inverse method might be applicable after a reordering of $\boldsymbol{M}$ by a standard method such as the reverse Cuthill-McKee method [119, §3.3]. In a few test cases for $r_{\mathrm{c}} = 2$ Å, the maximum block size is 45. But the size of the block quickly increases with the cutoff radius. So an upper limit on the block size should be implemented: if a block size is larger than, say, n_{c}, then the block is split into sub-blocks of size at most n_{c} large, and only the diagonal sub-blocks are inverted. The method needs complicated data structure for bookkeeping, and it is not as effective as the polynomial approximation in reducing the number of iterations in our tests.

3.4 Timing results

Table 3.3 shows a comparison between the computation of the polarizable RPOL water model and the charge-only SPC [14] water model. For both systems, there are 216 water molecules. The Ewald sum accuracy (ϵ in Eq. (2.25)) is set to 10^{-6}, the direct sum cutoff radius is 8 Å, and the grid size for the reciprocal sum computation is $18 \times 18 \times 18$. β is chosen so that the equal sign in Eq. (2.25) is satisfied. The time is averaged over 1000 MD steps with a 1 fs timestep. To solve the dipole

equation for the RPOL water model, the least squares prediction is used with 10 previous dipoles and the peek-CG method is used with the preconditioner designed in Section 3.3 having a 4 Å cutoff. The machine on which we test our code has an Intel Pentium 4 CPU of 3.06 GHz, the compiler is icc 8.0 with flags "-fast -unroll -xN." We define the cost for computing the charge-only model to be 1 work unit. Then the cost of one iteration is about 0.33 work units, much faster than about 1 unit in [140]. It is not clear how they implement the matrix–vector multiplication. One possible explanation is that they do not use the neighbor list nor store the $\boldsymbol{B}$ arrays (see Section 2.2.1). Note also that with dipole moments given, the polarizable model incurs only about 28% overhead with respect to the nonpolarizable model computations. This is consistent with reference [140], in which the corresponding cost is 25–30%. Detailed timing results can be found in Fig. 3.4.

Time (second)	SPC	RPOL	increase
direct sum	0.02116	0.02651	25%
reciprocal sum	0.00204	0.00331	62%
solving dipole	− − −	0.01511	−−
overall	0.02320	0.04494	94%

Table 3.3: The cost for computing the electrostatic energy and force of the RPOL and SPC models.

The prediction cost is negligible. The worst relative cost happens when the largest number of previous dipole moments are used, and fastest convergence is achieved. In our cases, this is when 15 previous dipole moments are used and two iterations lead to convergence. When this happens, the cost for prediction is only 0.88% of the total electrostatic computation.

The current implementation constructs the preconditioner before the iteration at each timestep. The construction has an estimated cost of 8% of a working unit. This step could have been integrated into the "preparation" phase thereby saving time even further. But for the sake of implementation simplicity, we do not do it. The result is, for methods that use preconditioner, there is a relatively high start up cost. In fact, estimated from table 3.4, each iteration costs less than 0.3 work unit.

Table 3.4 provides the timing results presented in work units for the peek-CG method with 4 Å cutoff when the initial guess is zero. The data gives the worst case scenario for the peek-CG

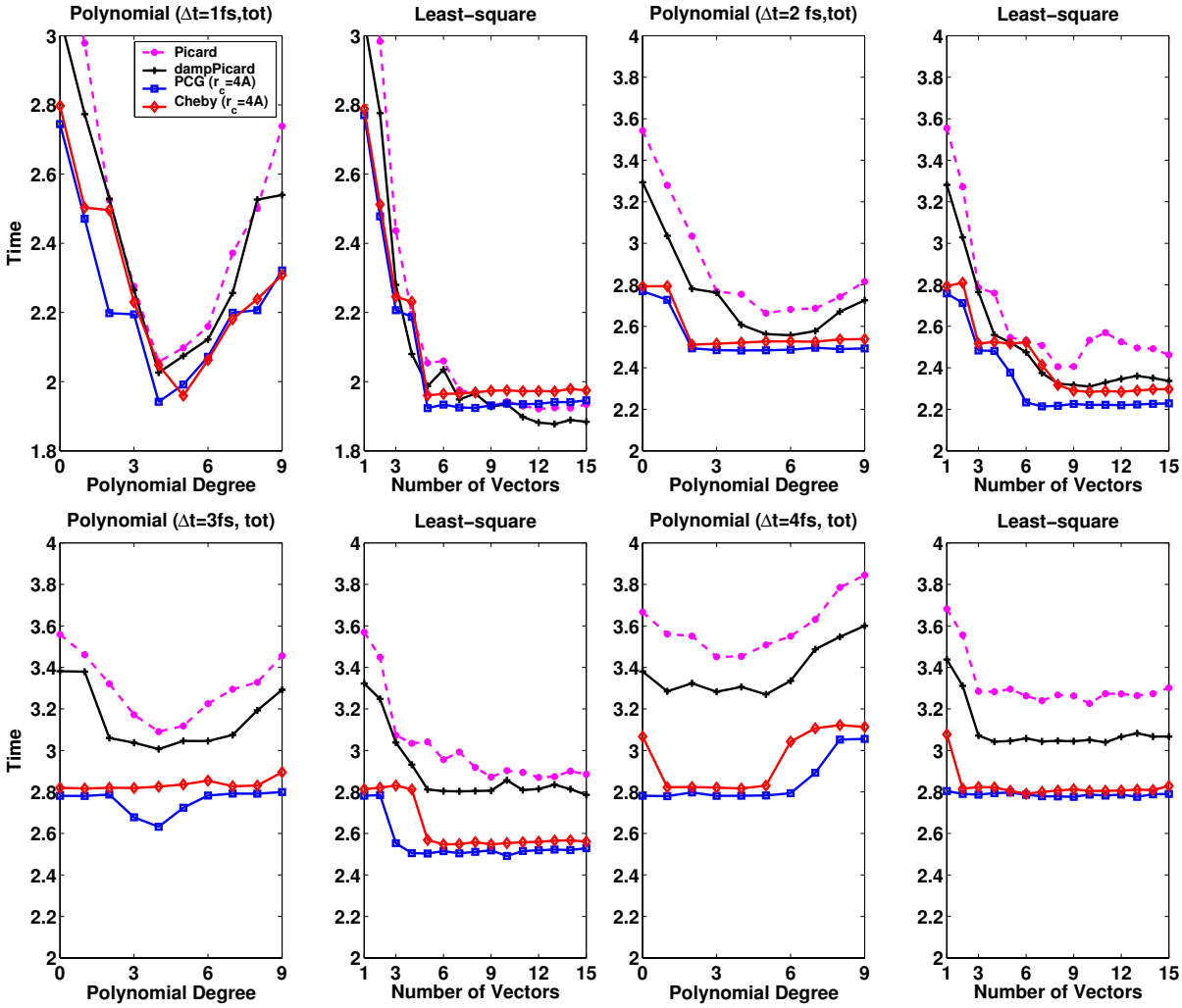

Figure 3.4: Computational cost in work units for different timesteps. The relative RMS convergence criteria is 4 ppm.

method when prediction is not used: a little more than three times as much as the charge-only computation. Note when we do not predict, the computational cost is independent of the timestep. The table is discussed further in Chapter 4.

3.5 Comparison to the extended Lagrangian approach

The extended Lagrangian approach is considered faster than the self-consistent approach since the former does not solve the dipole equation. However, the longest timestep an extended Lagrangian method can take is 1 fs [140, 129], while the self-consistent approach does not pose an upper limit on the possible timestep and the cost increase is modest when a larger timestep is used. In molecular

relative convergence criteria	400 ppm	40 ppm	4 ppm
average number of iterations	4.000	5.000	6.000
computational cost	2.50	2.77	3.09

Table 3.4: The cost of the self-consistent computation if the iteration starts from $\boldsymbol{d}_0 = 0$.

dynamics simulations, the timestep for computing the electrostatic energy/force can vary from 1 fs for velocity-Verlet method with fully flexible bonds, to 2 fs when covalent hydrogen bonds are rigid, to as large as 6 fs using multiple-time-stepping method [142, 105]. We carry out simulations with longer timesteps up through 4 fs, the largest timestep one can take without incurring significant energy drift with the velocity-Verlet method for the RPOL and SPC water systems.

Table 3.5 tells us that the self-consistent computation with a timestep no less than 2 fs is faster than the extended Lagrangian approach. The dipole equation is solved by the least squares prediction with 10 previous dipoles and the peek-CG method whose preconditioner is constructed with a 4 Å cutoff radius. Detailed timing results are summarized in Fig. 3.4.

timestep	1 fs	2 fs	3 fs	4 fs	Extended Lagrangian [140]
computational cost per fs	1.94	1.11	0.83	0.70	1.25–1.30

Table 3.5: The cost in work units for computing the electrostatic energy and force per femtosecond.

The average number of iterations needed for different timesteps are given in Fig. 3.5. The upper left figure is essentially the same as Fig. 3.1. It is repeated here for comparison purpose. The relative RMS convergence tolerance is 4 ppm, and all graphs in the figure have the same legend. We see the least squares predictor is consistently better than, or at least as good as, the polynomial predictor for all timesteps. For larger timesteps, prediction helps less, while the iteration method is more important. Empirically, we have optimal number of iterations is one more than the timestep in femtoseconds.

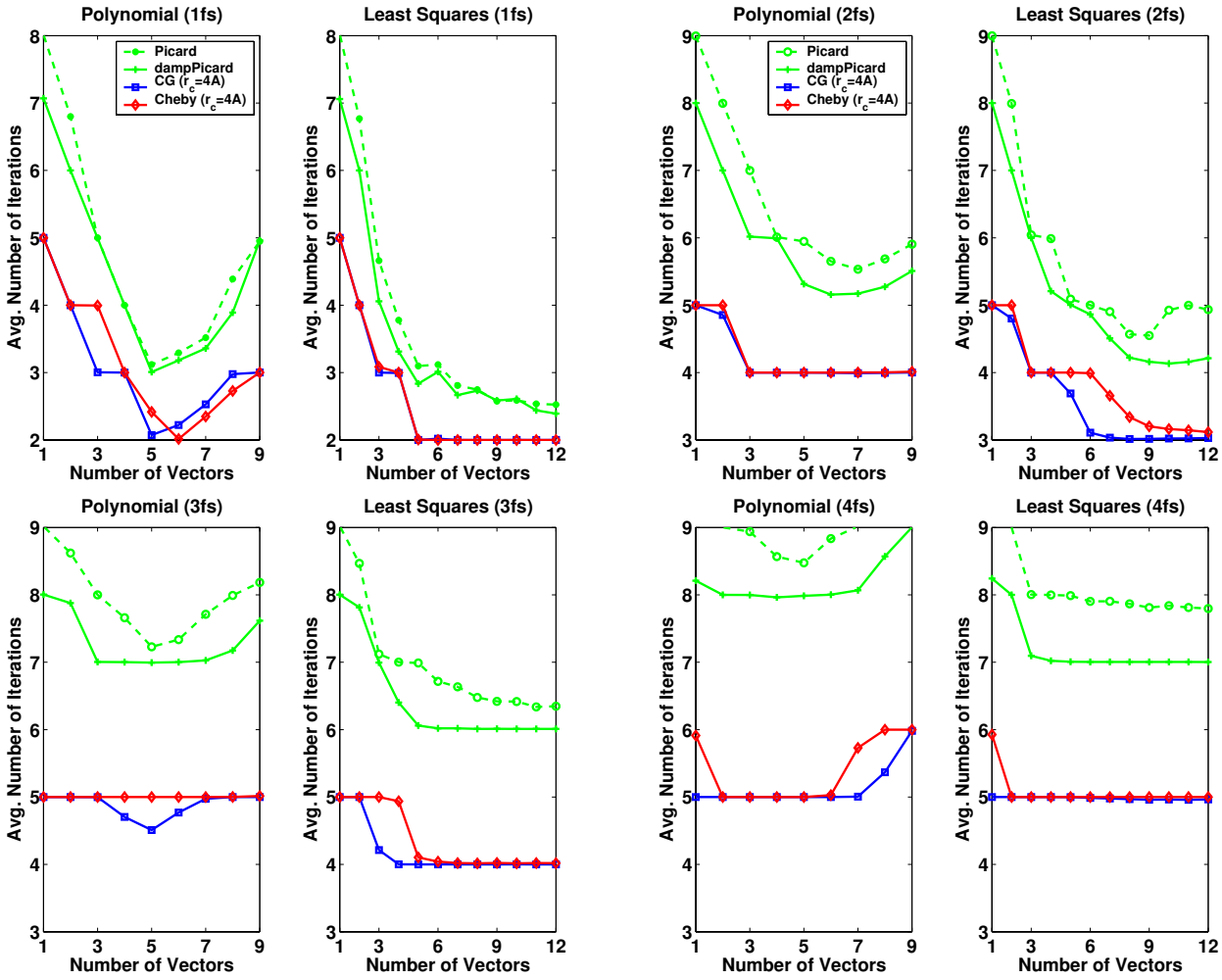

Figure 3.5: Average number of iterations for different timesteps.

Chapter 4

Energy drift

For deterministic simulations, conservation of energy or equivalent conserved quantity is very important. But Fig. 4.1 shows that self-consistent computations can lead to significant energy drift unless fully converged. This chapter discusses the origin of the energy drift and methods to control it.

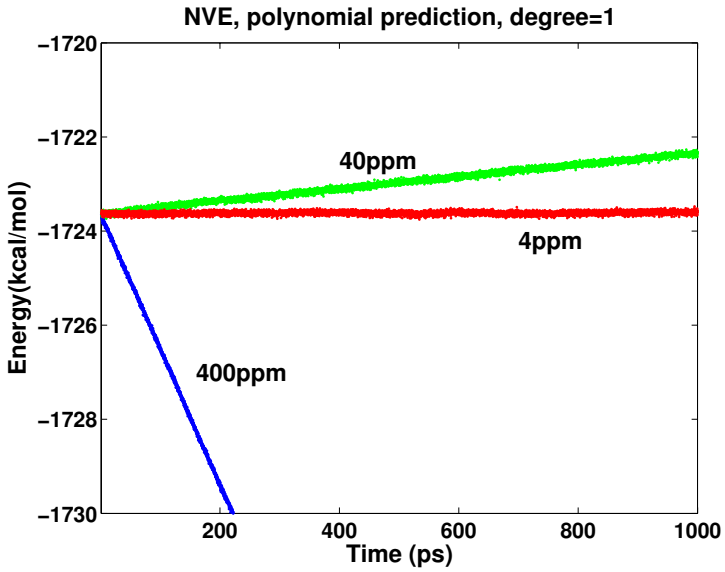

Figure 4.1: Energy drifts from 1 ns simulations for a RPOL water system.

For numerical integration of a Hamiltonian system, the energy conservation is strongly assisted by the numerical integrator being symplectic [60, Theorem IX.8.1]. However, for self-consistent computations, the symplecticness of the numerical integrator is compromised in two ways: (i) the computed force is not conservative due to the iterative solution being not exact, and (ii) the solution is history dependent due to the prediction. A conclusion of this chapter is that history dependency

is more detrimental than non-conservativeness in causing the energy drift.

Section 4.1 first analyzes the non-conservative effect on the self-consistent solution by excluding history from the self-consistent computation and then analyzes the history effect by examining the energy drift for different predictors. We determine that the dipole solution needs only 400 ppm RMS convergence criteria for a suitably accurate energy/force evaluation. Then, we consider long-time NVE simulations with the dipole equation solved by starting from $\boldsymbol{d}_0 = 0$ (no history). The energy drift is hardly perceptible even if the convergence criteria is 400 ppm. This is in sharp contrast with Fig. 4.1, where the same convergence criteria leads to significant energy drift. When history (prediction) is used, the energy drift strongly depends on the type of the prediction.

Section 4.2 discusses the always-stable-prediction-corrector (ASPC) method proposed in [79]. With a timestep of 1 fs, the ASPC method can maintain a constant energy level with only one damped Picard iteration by using a quasi-time-reversible predictor and a proper damping factor. The method is very fast since only one iteration is needed, but it fails to conserve energy when the timestep is 2 fs. Another drawback of the method is its low accuracy and lack of direct accuracy control. To improve its accuracy, we employ the time-reversible requirements as constraints into least squares fitting. This combination can have benefits from both approaches: stability with a low convergence accuracy criteria from quasi-time-reversible prediction and accuracy from least squares prediction. We demonstrate the improved approach has better accuracy and maintains a constant energy level for long time simulations with an average iteration of less than 1.5 only.

Section 4.3 shows that symplecticness is compromised in self-consistent computations but the volume-preserving property is maintained if history is not used. Integrators which do not preserve volume can lead to serious problems, such as the flying-ice cube phenomena [64].

Since energy drift is sometimes unavoidable for MD simulations, a practical energy drift criterion is needed when we evaluate a method. Currently, the simulation length is tens of nanoseconds, so a reasonable criterion is

the total energy drift in a 20 ns simulation be no more than
a 5 Kelvin change of the system temperature.

For our system of 216 RPOL water molecules, the energy drift should be no more than 0.321 kcal/mol/ns.

Accuracy needs and cost decide which method to use. Although symplecticness is not preserved if the dipole solution is not exact, phase-space volume-preservation and energy conservation probably suffice. So with respect to quality, zero-guess self-consistent computation is better. However, as Table 3.4 shows, the extra cost will be about 150% compared to the charge-only computation. On the other hand, if we use accurate prediction, we can obtain very accurate dipole moments with about 94% extra cost compared with the charge-only computation and keep the energy drift negligibly small. When the computed dipole moment has a small error, we would expect the phase space volume change is small.

If a short timestep ($\Delta t = 1, 2$ fs) is used, we should use the least squares predictor using 8 or more previous dipoles, and require high accuracy (4 ppm) for the dipole solution. If longer timesteps ($\Delta t > 2$ fs) are used (e.g., in multiple-time-stepping method), prediction helps very little to obtain an accurate initial guess, so we should use zero-guess and claim convergence for a relatively low accuracy (400 ppm) dipole solution, which is still good enough to accurately compute the energy and force and maintain the energy at a constant level for long time simulations.

4.1 Accuracy and history

We first determine a suitable convergence criterion for the iteration based on PME accuracy. For this purpose, we look at the relative errors in the 2-norm of the dipole moment, the electrostatic energy, and the electrostatic force. Exact values are computed by the standard Ewald sum method, not PME, with the Ewald sum error tolerance (ϵ in Eq. (2.25)) set to 10^{-20} and the relative dipole convergence criteria set to 4×10^{-15}.

Table 4.1 shows the error introduced by PME as well as by iteration with different convergence criteria. The quantity δd is defined in Eq. (3.4), δF^{el} is defined similarly, and

$$\delta E^{\mathrm{el}} = \frac{|E^{\mathrm{el}} - E^{\mathrm{el}}_{\mathrm{exact}}|}{E_{\mathrm{k}}} \tag{4.1}$$

where E_{k} is the kinetic energy of the system. We do not use $|E^{\mathrm{el}}_{\mathrm{exact}}|$ in the denominator because the potential energy can be redefined by adding an arbitrary constant without affecting the dynamics. The Ewald sum accuracy (ϵ in Eq. (2.25)) is set to 10^{-6}, the error used routinely in our simulations.

For the "0 ppm" column, the relative dipole convergence criteria is set to 4×10^{-15}, so the error dominantly comes from the PME method. For other columns, each corresponds to a specified convergence error. Observe that the relative error introduced by PME is at the level of 10^{-4}. Also, observe that the relative RMS convergence tolerance can be as high as 400 ppm without introducing any significant error.

	0 ppm	4 ppm	40 ppm	400 ppm	4000 ppm
δd (ppm)	153	153	153	167	689
δF^{el} (ppm)	136	136	136	138	240
δE^{el} (ppm)	0.209	0.209	0.209	0.210	0.240

Table 4.1: The error of the dipole, the electrostatic force, and the electrostatic energy of the PME method and of different convergence criteria.

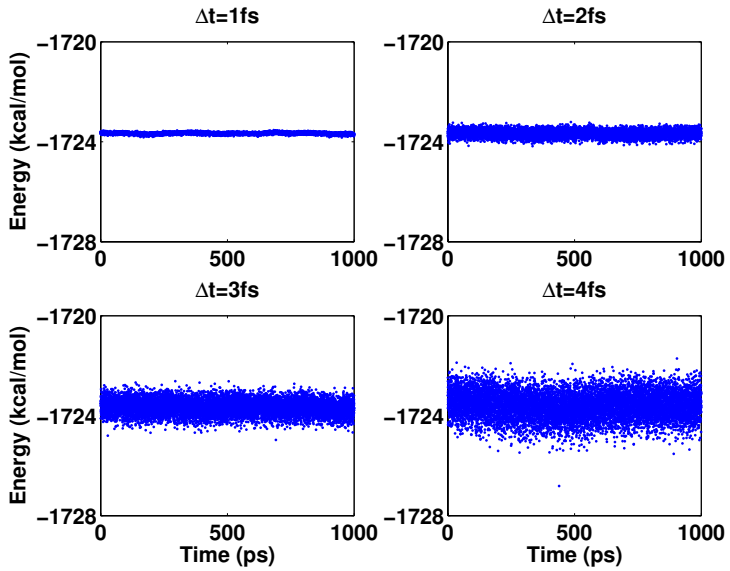

Figure 4.2: Energies in NVE ensemble simulations when iteration starts from $\boldsymbol{d}_0 = 0$. The relative RMS error is 400ppm.

To study the energy drift dependency on predictions, we carefully exclude other factors which can cause the energy drift. The van der Waals potential is smoothed and is a C^1 function (see Appendix A), constraints are enforced by the SETTLE method [91], and the net force generated by the PME method is not subtracted out (see Fig. 2.4).

Fig. 4.2 shows the energies of NVE ensemble simulations using 400ppm as the relative RMS convergence criteria with 0 initial guess. The integrator is the velocity Verlet method. The iteration

method is the peek-CG method, and the preconditioner is constructed with a 4 Å cutoff. The energy drift is hardly perceptible. This means an inexact solution alone, and hence a nonconservative force, does not necessarily cause significant energy drift.

On the other hand, when history is used, the drift can be significant as is seen from Fig. 4.3. The iteration method is peek-CG with an $r_c = 4$ Å cutoff preconditioner. The line with legend "400ppm, 0" is the energy drift from a simulation in which the dipole equation is solved by 0 initial guess and a relative RMS convergence criteria of 400 ppm. We have a few observations from Fig. 4.3:

- The energy drift is approximately proportional to the RMS convergence error.
- For polynomial extrapolation, the energy drift strongly depends on the polynomial degree. Higher degree leads to less drift in general. For least squares prediction, the dependence on the number of previous dipole moments is less significant.

4.2 ASPC method

The ASPC method computes the dipole moment at step n by two steps

$$\text{predict}: \quad \boldsymbol{d}_0^n = \sum_{i=1}^{k} c_i \boldsymbol{d}^{n-i}, \tag{4.2}$$

$$\text{iterate once}: \quad \boldsymbol{d}^n = \omega \boldsymbol{D}_\alpha(-\boldsymbol{G}_2\boldsymbol{d}_0^n - \boldsymbol{G}_1\boldsymbol{q}) + (1-\omega)\boldsymbol{d}_0^n, \tag{4.3}$$

where c_i are chosen in such a way that if we assume $\boldsymbol{d}$ is a smooth function of time and do a Taylor expansion at $t = t^n$, then we will get

$$\boldsymbol{d}_0^n = \boldsymbol{d}^n + \tilde{c}_2\Delta t^2 + \tilde{c}_4\Delta t^4 + \cdots + \tilde{c}_{2k-4}\Delta t^{2k-4} + O(\Delta t^{2k-2}). \tag{4.4}$$

where vectors $\{\tilde{c}_{2i}\}_{i=1}^{k-2}$ are some uninteresting coefficients. All the odd power terms of Δt up through Δt^{2k-3} are eliminated. The intention is to improve the "time-reversibility" of the prediction, since when $\Delta t \to -\Delta t$, odd power terms change sign, while even power terms remain the same. The

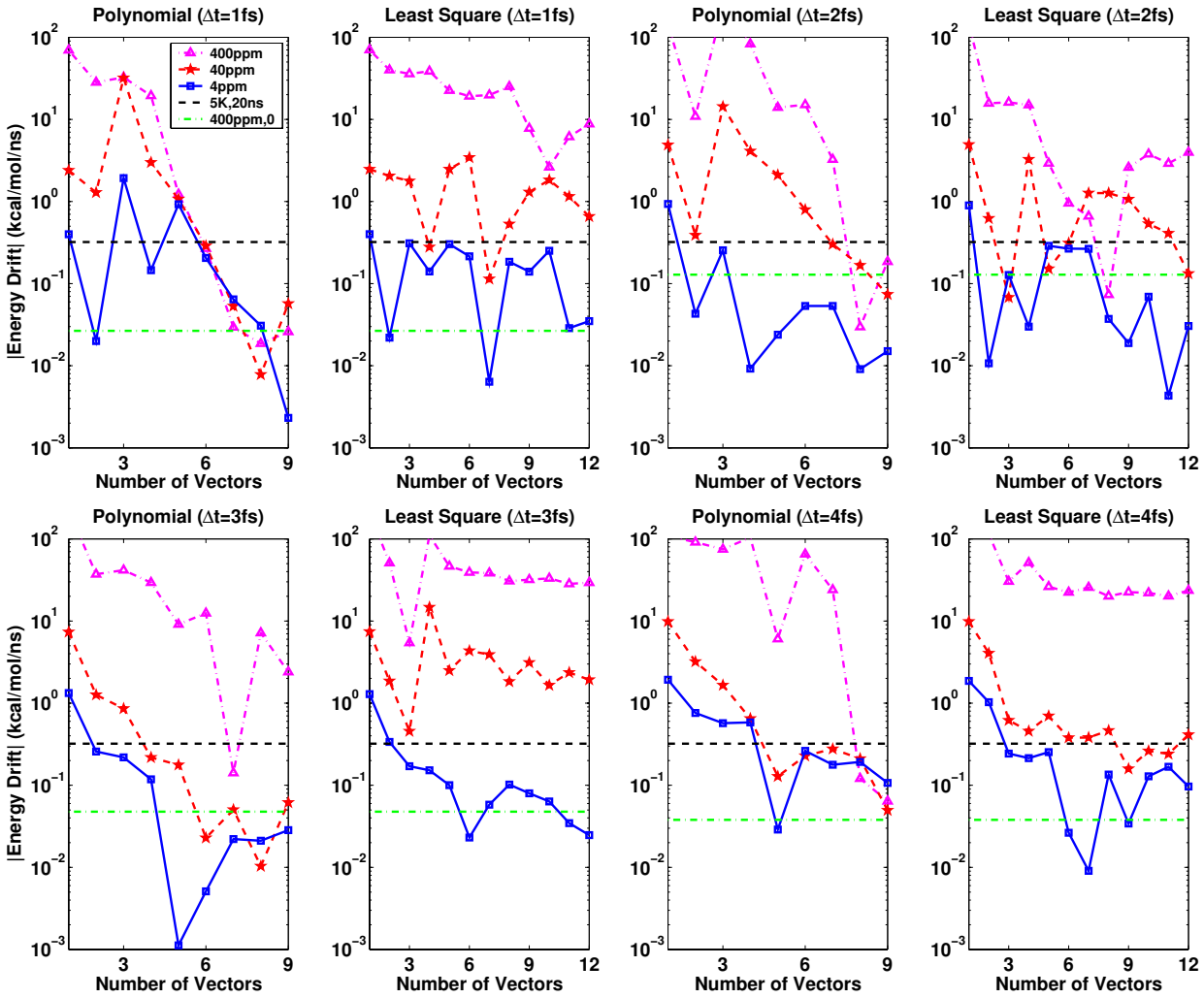

Figure 4.3: Energy drifts when dipoles from previous timesteps are used for an accurate initial guess.

ASPC method does only one damped Picard iteration. The damping factor ω in Eq. (4.3) is chosen by a frozen coefficient analysis in which $\boldsymbol{G}_2$ is assumed constant and $\boldsymbol{G}_1\boldsymbol{q}$ is assumed 0. By requiring that the dipole moments converge to 0 as the timestep increases, an upper limit on the value of ω can be obtained. The paper points out that using

$$\omega \quad = \quad \frac{k+1}{2k+1} \tag{4.5}$$

guarantees the dynamics to be stable. The energies of the NVE ensemble simulations using the ASPC method are presented in Fig. 4.4. When $\Delta t = 1$ fs, the energy drift is tolerable for $k = 5, 6, 7$.

If the damping factors ω is too large, the energy quickly drifts away or even jumps: the dynamics is unstable. The optimal damping factor is obtained by a trial-and-error process: reduce the value ω gradually until the dynamics is stable for a few picoseconds. The results are $\omega = 0.82$ for $k = 5$, $\omega = 0.81$ for $k = 6$, $\omega = 0.80$ for $k = 7$, and $\omega = 0.79$ for $k = 8$. When $\Delta t = 2$ fs, the energy drift is unavoidable even with those "safe" damping factors defined in Eq. (4.5). Note the energy and time scale are different for the two figures.

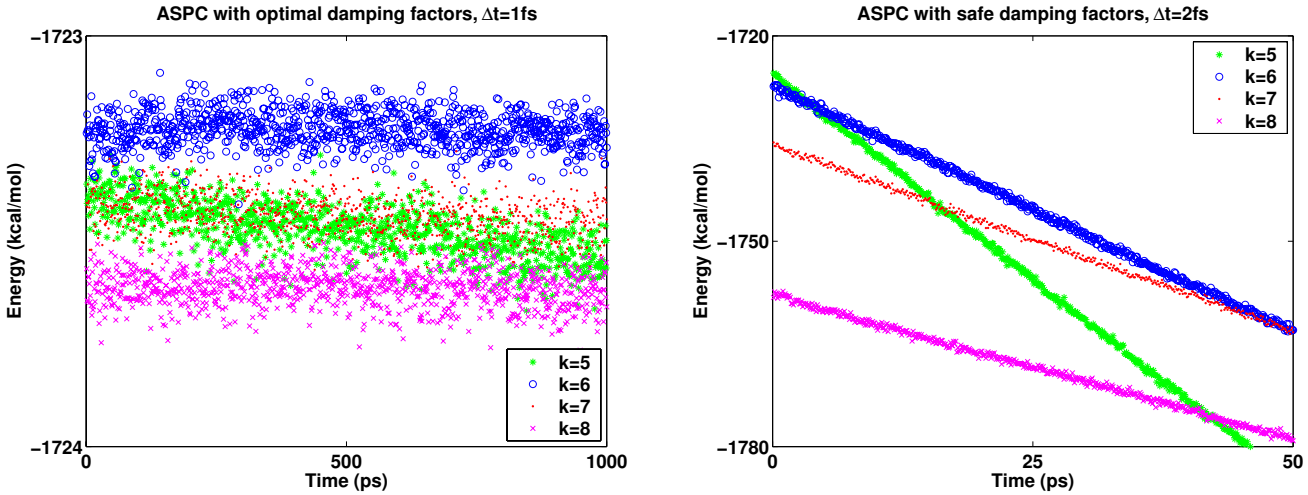

Figure 4.4: MD simulations using the ASPC method.

The dynamics of many physical systems can be approximated by oscillations near its potential minimum. This motivates us to study a one-dimensional toy problem with unit mass and the following potential energy:

$$U(x,d) \quad = \quad \frac{1}{2}x^2 + \frac{1}{2}(2+x^2)d^2 - d\,. \tag{4.6}$$

where d is an auxiliary variable, whose value is determined by minimizing the potential

$$\frac{\partial U(x,d)}{\partial d} = 0 \quad \Rightarrow \quad d = \frac{1}{2+x^2}\,. \tag{4.7}$$

The system is integrated by the velocity-Verlet method while the dipole is computed by four methods:

1. Exact: solve d exactly at each step according to Eq. (4.7).

2. Polynomial: use degree-3 polynomial extrapolation to compute the initial value, then iterate by $d_{m+1} = \frac{1}{2}(1 - x^2 d_m)$ until the RMS error (Eq. (3.1))is less than 0.5.

3. Zero-guess: start the same iteration with $d = 0$ until the RMS error is less than 0.5.

4. ASPC method: use $k = 6$ and the "safe" damping factor as given by Eq. (4.5).

Fig. 4.5 shows the trajectories in phase space spanned by x and momentum p. 10000 steps are integrated. To be "fair" to all the methods, we choose the RMS criteria to be 0.5. The average number of iterations for polynomial prediction ("Poly." in the figure) is 2.0 and the average number of iterations for zero prediction ("Zero" in the figure) is 1.0 for both timesteps. For small timesteps, ASPC is very good. But for a larger timestep, the ASPC trajectory collapses to the center. Note that the zero-guess method is excellent for both timesteps. The reason is that the method preserves phase space volume, a topic discussed in detail in Section 4.3.

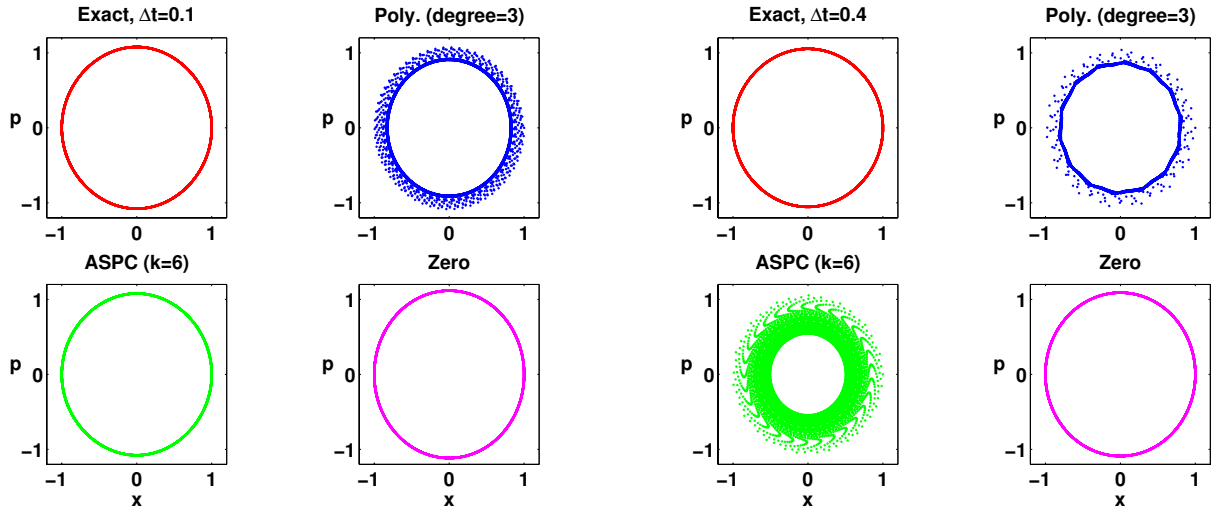

Figure 4.5: Phase space trajectories for different prediction methods.

For molecular dynamics, dipole moments predicted by Eq. (4.2) of the ASPC method have a relatively poor accuracy compared to polynomial extrapolation and least squares prediction, and the ASPC method does not have a direct control over the solution accuracy. In our simulations, the errors of the ASPC method are

$$\delta \boldsymbol{d} \approx 770 \text{ ppm}\,, \qquad \delta \boldsymbol{F} \approx 250 \text{ ppm}\,. \tag{4.8}$$

	Energy Drift (kcal/mol/ns)				Average Iterations			
$m\backslash k$	12	13	14	15	12	13	14	15
5	0.12	−0.07	−0.21	0.07	1.48	1.50	1.51	1.53
6	0.18	0.16	0.19	−0.01	1.47	1.49	1.57	1.65
7	0.41	0.06	−0.05	0.02	1.59	1.63	1.64	1.63

Table 4.2: Energy drifts and average number of iterations for the quasi-time-reversible-least-square predictor.

After we enforce an accuracy control by requesting the iteration stops only if the RMS error is small enough, we find at least two Chebyshev semi-iterative iterations are needed even if the convergence criteria is 400 ppm. To do better, we need to improve prediction accuracy. For this purpose, we combine the ASPC method with the least squares predictor to achieve both accuracy and stability. Suppose the new predictor is

$$\boldsymbol{d}_0^n \;=\; \sum_{i=1}^{k} c_i \boldsymbol{d}^{n-i}\,, \tag{4.9}$$

and we have $m+1$ ($k \geq m+1 \geq 1$) time-reversibility constraints

$$\sum_{i=1}^{k} c_i = 1\,, \quad \sum_{i=1}^{k} c_i i = 0\,, \quad \sum_{i=1}^{k} c_i i^3 = 0\,, \quad \ldots, \quad \sum_{i=1}^{k} c_i i^{2m-1} \;=\; 0\,. \tag{4.10}$$

The objective function is

$$T_{k,m} \;=\; \|\boldsymbol{d}^n - \boldsymbol{d}_p^n\|^2 + \lambda_0(\sum_{i=1}^{k} c_i - 1) + \lambda_1 \sum_{i=1}^{k} c_i i + \lambda_2 \sum_{i=1}^{k} c_i i^3 + \cdots + \lambda_m \sum_{i=1}^{k} c_i i^{2m-1}\,. \tag{4.11}$$

where $\lambda_0, \lambda_1, \ldots, \lambda_m$ are Lagrangian multipliers. Minimizing it gives the coefficients we need to predict the dipole at timestep $n+1$. The dipole solution error is controlled by requesting the RMS error to be less than a certain tolerance.

Tables 4.2 summarizes the simulation results using this quasi-time-reversible-least-square predictor. Each row corresponds to a set number of quasi-time-reversible (TR) constraints (5–7), each column corresponds to a set number of previous dipole moments (12–15) used in the least squares prediction. The Chebyshev semi-iterative method with preconditioner constructed with 4 Å cut-off is used to iterate to 400 ppm. The computational cost is further reduced, the energy drift is

tolerable, and we have suitable accuracy.

4.3 Prediction undermines the volume-preserving property

Energy preservation during the numerical integration is strongly assisted by the symplectic property. A weaker property of a numerical integration method is the volume-preserving property, which is corresponding to the Liouville theorem [7]. This section shows that the volume-preserving property is conserved if the iteration starts with the zero initial guess, but compromised by prediction. This section also reveals that the symplecticness is compromised in the self-consistent computation.

For positions $\boldsymbol{r}$ and momenta $\boldsymbol{p}$, both being $3N$-vectors, define

$$\Phi(\begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{p} \end{bmatrix}) = \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{p} + \frac{\Delta t}{2}\boldsymbol{F}(\boldsymbol{r}, \boldsymbol{d}(\boldsymbol{r})) \end{bmatrix}, \tag{4.12}$$

$$\Psi(\begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{p} \end{bmatrix}) = \begin{bmatrix} \boldsymbol{r} + \Delta t M^{-1}\boldsymbol{p} \\ \boldsymbol{p} \end{bmatrix}, \tag{4.13}$$

where M is the diagonal mass matrix and $\boldsymbol{F}$ is the forces. Here we assume prediction is not used, so $\boldsymbol{d}$ is a function of $\boldsymbol{r}$ only. The velocity-Verlet method can be written as

$$\begin{bmatrix} \boldsymbol{r}^{n+1} \\ \boldsymbol{p}^{n+1} \end{bmatrix} = \Phi \circ \Psi \circ \Phi(\begin{bmatrix} \boldsymbol{r}^{n} \\ \boldsymbol{p}^{n} \end{bmatrix}) \tag{4.14}$$

Define $z = [\boldsymbol{r}^{\mathsf{T}} \ \ \boldsymbol{p}^{\mathsf{T}}]^{\mathsf{T}}$. A $\boldsymbol{R}^{6N} \to \boldsymbol{R}^{6N}$ mapping $\phi(z)$ is symplectic [60] if

$$(\frac{\partial \phi}{\partial z})^{\mathsf{T}} \boldsymbol{J} \frac{\partial \phi}{\partial z} = \boldsymbol{J}, \tag{4.15}$$

where $\boldsymbol{J}$ is

$$\boldsymbol{J} = \begin{bmatrix} 0 & \boldsymbol{I} \\ -\boldsymbol{I} & 0 \end{bmatrix}, \tag{4.16}$$

and $\boldsymbol{I}$ is the identity matrix. A map ϕ is volume-preserving if it preserves the phase space volume:

$$\det(\frac{\partial \phi}{\partial z}) \quad = \quad 1 \tag{4.17}$$

From these definitions, we see a symplectic map is volume-preserving. We can verify that Ψ is symplectic, Φ is symplectic if $\boldsymbol{F}$ is a gradient of a scalar potential which depends only on $\boldsymbol{r}$. Since the composition of symplectic maps is still symplectic [60], the velocity-Verlet method is symplectic.

In the self-consistent computation, the inexact dipole solution makes the force not a gradient of the potential anymore and the symplecticness of the velocity-Verlet integrator is compromised. But

$$\det(\frac{\partial \Phi}{\partial z}) \quad = \quad \det(\begin{bmatrix} \boldsymbol{I} & 0 \\ \frac{\Delta t}{2}(\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{r}} + \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{d}} \frac{\partial \boldsymbol{d}}{\partial \boldsymbol{r}}) & \boldsymbol{I} \end{bmatrix}) = 1 \,. \tag{4.18}$$

The volume-preserving property is still conserved.

When prediction is used, the state vector becomes $(\boldsymbol{r}^n, \boldsymbol{p}^n, \boldsymbol{d}^n, \ldots, \boldsymbol{d}^{n-k+1})$. It is a complicated task to determine if the molecular dynamics simulation preserves phase space volume and in general there is no reason to believe that the volume is preserved.

Chapter 5

A non-iterative method

In this chapter, we use a fixed polynomial approximation to $(\boldsymbol{I} + \boldsymbol{D}_\alpha \boldsymbol{G}_2)^{-1}$

$$(\boldsymbol{I} + \boldsymbol{D}_\alpha \boldsymbol{G}_2)^{-1} \approx P_n(\boldsymbol{D}_\alpha \boldsymbol{G}_2) \tag{5.1}$$

where $P_n(\boldsymbol{D}_\alpha \boldsymbol{G}_2)$ is a degree-n polynomial in $\boldsymbol{D}_\alpha \boldsymbol{G}_2$ designed to have optimal accuracy. The electrostatic energy, written as

$$E^{\text{Ewald}} = \frac{1}{2}\boldsymbol{q}^\mathsf{T}\boldsymbol{G}_0\boldsymbol{q} - \frac{1}{2}(\boldsymbol{G}_1\boldsymbol{q})^\mathsf{T}(\boldsymbol{I} + \boldsymbol{D}_\alpha \boldsymbol{G}_2)^{-1}\boldsymbol{D}_\alpha \boldsymbol{G}_1\boldsymbol{q}\,, \tag{5.2}$$

is therefore approximated by

$$E^{\text{el}} = \frac{1}{2}\boldsymbol{q}^\mathsf{T}\boldsymbol{G}_0\boldsymbol{q} - \frac{1}{2}(\boldsymbol{G}_1\boldsymbol{q})^\mathsf{T}P_n(\boldsymbol{D}_\alpha \boldsymbol{G}_2)\boldsymbol{D}_\alpha \boldsymbol{G}_1\boldsymbol{q}\,. \tag{5.3}$$

Then it is feasible to define and compute the force to be the exact negative gradient of the potential energy. This ensures the symplecticness of the integrator, thus eliminating the energy drift problem.

The dipole, if needed, is

$$\boldsymbol{d}_n = P_n(\boldsymbol{D}_\alpha \boldsymbol{G}_2)(-\boldsymbol{D}_\alpha \boldsymbol{G}_1\boldsymbol{q})\,. \tag{5.4}$$

A possible drawback of the non-iterative method is that it may change the physical process. For example, computed by this method, the dynamics is free of polarization catastrophes even if the physical model is badly designed and leads to a catastrophe if the exact matrix inverse is used.

A degree-2 polynomial of $\boldsymbol{D}_\alpha \boldsymbol{G}_2$ is used in [75] in a Monte Carlo simulation of water and methanol. But the polynomial is not optimal since it is from a Neumann expansion.

5.1 Polynomial approximation

We start with degree-1 polynomial to demonstrate the process:

$$\boldsymbol{d} \approx (a\boldsymbol{D}_\alpha \boldsymbol{G}_2 + b\boldsymbol{I})(-\boldsymbol{D}_\alpha \boldsymbol{G}_1 \boldsymbol{q}), \tag{5.5}$$

where a and b are two parameters. The error introduced by the above approximation is

$$\Delta \boldsymbol{d} = \left[(a\boldsymbol{D}_\alpha \boldsymbol{G}_2 + b\boldsymbol{I}) - (\boldsymbol{I} + \boldsymbol{D}_\alpha \boldsymbol{G}_2)^{-1}\right](-\boldsymbol{D}_\alpha \boldsymbol{G}_1 \boldsymbol{q}). \tag{5.6}$$

Because $\boldsymbol{D}_\alpha^{-1}$ is positive definite, we define the α-norm of a vector $\boldsymbol{v}$ to be $\|\boldsymbol{v}\|_\alpha = \boldsymbol{v}^\mathsf{T} \boldsymbol{D}_\alpha^{-1} \boldsymbol{v}$. Then

$$\begin{aligned} \|\Delta \boldsymbol{d}\|_\alpha^2 &= (-\boldsymbol{D}_\alpha^{1/2} \boldsymbol{G}_1 \boldsymbol{q})^\mathsf{T} [a\boldsymbol{D}_\alpha^{1/2} \boldsymbol{G}_2 \boldsymbol{D}_\alpha^{1/2} + b\boldsymbol{I} - (\boldsymbol{I} + \boldsymbol{D}_\alpha^{1/2} \boldsymbol{G}_2 \boldsymbol{D}_\alpha^{1/2})^{-1}]^2 \\ &\quad \cdot (-\boldsymbol{D}_\alpha^{1/2} \boldsymbol{G}_1 \boldsymbol{q}). \end{aligned} \tag{5.7}$$

Since $\boldsymbol{D}_\alpha^{1/2} \boldsymbol{G}_2 \boldsymbol{D}_\alpha^{1/2}$ is symmetric, it is unitarily diagonalizable [141, Theorem 24.7] and has the following decomposition:

$$\boldsymbol{D}_\alpha^{1/2} \boldsymbol{G}_2 \boldsymbol{D}_\alpha^{1/2} = \sum_{i=1}^{n} \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^\mathsf{T}, \tag{5.8}$$

$$\sum_{i=1}^{n} \boldsymbol{u}_i \boldsymbol{u}_i^\mathsf{T} = \boldsymbol{I}, \quad \boldsymbol{u}_i^\mathsf{T} \boldsymbol{u}_j = \delta_{ij}, \tag{5.9}$$

where the λ_i and $\boldsymbol{u}_i$ are the eigenvalues and eigenvectors of the matrix $\boldsymbol{D}_\alpha^{1/2} \boldsymbol{G}_2 \boldsymbol{D}_\alpha^{1/2}$. So

$$\begin{aligned} \|\Delta \boldsymbol{d}\|_\alpha^2 &= \sum_{i=1}^{n} (a\lambda_i + b - \frac{1}{1+\lambda_i})^2 [\boldsymbol{u}_i^\mathsf{T} (-\boldsymbol{D}_\alpha^{1/2} \boldsymbol{G}_1 \boldsymbol{q})]^2 \\ &\le \max_{\lambda_\mathrm{m} \le \lambda \le \lambda_\mathrm{M}} (a\lambda + b - \frac{1}{1+\lambda})^2 \sum_{i=1}^{n} [\boldsymbol{u}_i^\mathsf{T} (-\boldsymbol{D}_\alpha^{1/2} \boldsymbol{G}_1 \boldsymbol{q})]^2 \\ &= \max_{\lambda_\mathrm{m} \le \lambda \le \lambda_\mathrm{M}} (a\lambda + b - \frac{1}{1+\lambda})^2 \| - \boldsymbol{G}_1 \boldsymbol{q}\|_\alpha^2, \end{aligned} \tag{5.10}$$

where $\lambda_{\rm m}$ and $\lambda_{\rm M}$ are the least and greatest eigenvalues of $\boldsymbol{D}_\alpha \boldsymbol{G}_2$, which has the same eigenvalues as $\boldsymbol{D}_\alpha^{1/2} \boldsymbol{G}_2 \boldsymbol{D}_\alpha^{1/2}$ does. The goal then is to find a and b so that the maximum of $|a\lambda + b - 1/(1+\lambda)|$ is minimized over the range $[\lambda_{\rm m}, \lambda_{\rm M}]$, i.e., we are to use a linear polynomial $a\lambda + b$ to do a uniform approximation to the function $1/(1+\lambda)$. The equioscillation theorem [83, Theorem2.19] is readily applied:

$$
\begin{aligned}
a\lambda_{\rm m} + b - \frac{1}{1+\lambda_{\rm m}} &= -\delta\,, && (5.11)\\
a\lambda_{*} + b - \frac{1}{1+\lambda_{*}} &= \delta\,, && (5.12)\\
a\lambda_{M} + b - \frac{1}{1+\lambda_{\rm M}} &= -\delta\,, && (5.13)\\
\frac{\rm d}{{\rm d}\lambda}(a\lambda + b - \frac{1}{1+\lambda})\Big|_{\lambda_*} &= 0\,. && (5.14)
\end{aligned}
$$

where δ is the maximum error, and $\lambda_* \in (\lambda_{\rm m}, \lambda_{\rm M})$. The solution is

$$
\begin{aligned}
a &= -\frac{1}{(1+\lambda_{\rm m})(1+\lambda_{\rm M})}\,, && (5.15)\\
b &= \frac{\sqrt{(1+\lambda_{\rm m})(1+\lambda_{\rm M})} + (\lambda_{\rm m}+\lambda_{\rm M})/2}{(1+\lambda_{\rm m})(1+\lambda_{\rm M})}\,, && (5.16)\\
\lambda_* &= \sqrt{(1+\lambda_{\rm M})(1+\lambda_{\rm m})} - 1\,, && (5.17)\\
\delta &= \frac{(\sqrt{\kappa}-1)^2}{2(1+\lambda_{\rm M})}\,, && (5.18)\\
\kappa &= \frac{1+\lambda_{\rm M}}{1+\lambda_{\rm m}}\,. && (5.19)
\end{aligned}
$$

The value κ is the condition number of $\boldsymbol{I} + \boldsymbol{D}_\alpha \boldsymbol{G}_2$. For a RPOL water model, $\lambda_{\rm m} \approx -0.34$, $\lambda_{\rm M} \approx 0.26$, so $\delta \approx 3.1\%$.

The degree-0 polynomial $\boldsymbol{d}_0 = a(-\boldsymbol{D}_\alpha \boldsymbol{G}_1 \boldsymbol{q})$ should have the following parameter value and error estimate:

$$
\begin{aligned}
a &= \frac{1}{2}(\frac{1}{1+\lambda_{\rm m}} + \frac{1}{1+\lambda_{\rm M}})\,, && (5.20)\\
\delta &= \frac{1}{2}(\frac{1}{1+\lambda_{\rm m}} - \frac{1}{1+\lambda_{\rm M}})\,. && (5.21)
\end{aligned}
$$

The degree-2 polynomial can be constructed in a similar way. This time, for the sake of clarity,

we use $\mu = 1 + \lambda$, $\mu_{\rm m} = 1 + \lambda_{\rm m}$, $\mu_{\rm M} = 1 + \lambda_{\rm M}$, and so on. Assume the optimal polynomial is $a\mu^2 + b\mu + c$, or $a\lambda^2 + (2a + b)\lambda + (a + b + c)$, then

$$\begin{aligned}
\frac{1}{\mu_{\rm m}} - (a\mu_{\rm m}^2 + b\mu_{\rm m} + c) &= \delta\,, && (5.22)\\
\frac{1}{\mu_1} - (a\mu_1^2 + b\mu_1 + c) &= -\delta\,, && (5.23)\\
\frac{1}{\mu_2} - (a\mu_2^2 + b\mu_2 + c) &= \delta\,, && (5.24)\\
\frac{1}{\mu_{\rm M}} - (a\mu_{\rm M}^2 + b\mu_{\rm M} + c) &= -\delta\,, && (5.25)\\
\frac{1}{\mu_1^2} + (2a\mu_1 + b) &= 0\,, && (5.26)\\
\frac{1}{\mu_2^2} + (2a\mu_2 + b) &= 0\,, && (5.27)
\end{aligned}$$

where $\mu_1, \mu_2 \in (\mu_{\rm m}, \mu_{\rm M})$. We can first solve for μ_1 and μ_2:

$$\begin{aligned}
\mu_1 &= \frac{1}{2}\mu_{\rm m}(\sqrt{\kappa} + 1)\,, && (5.28)\\
\mu_2 &= \sqrt{\kappa} \cdot \mu_1\,, && (5.29)
\end{aligned}$$

and then

$$\begin{aligned}
a &= \frac{1}{\mu_1^3}\frac{\sqrt{\kappa} + 1}{2\kappa}\,, && (5.30)\\
b &= -\frac{1}{\mu_1^2}\frac{\kappa + \sqrt{\kappa} + 1}{\kappa}\,, && (5.31)\\
c &= \frac{1}{\mu_{\rm m}}\frac{\kappa + 4\sqrt{\kappa} + 1}{2\kappa}\,, && (5.32)\\
\delta &= \frac{1}{\mu_{\rm m}}\frac{\kappa\sqrt{\kappa} - 3\kappa + 3\sqrt{\kappa} - 1}{2\kappa(\sqrt{\kappa} + 1)}\,. && (5.33)
\end{aligned}$$

For the given practical values ($\mu_{\rm m} = 1 - 0.34 = 0.66$, $\mu_{\rm M} = 1 + 0.26 = 1.26$), we have $\delta \approx 0.93\%$. From degree-1 to degree-2, the accuracy is improved from 3.1% to 0.93%. The optimal degree-2 dipole is

$$\boldsymbol{d}_2 = \left[a\boldsymbol{D}_\alpha\boldsymbol{G}_2\boldsymbol{D}_\alpha\boldsymbol{G}_2 + (2a + b)\boldsymbol{D}_\alpha\boldsymbol{G}_2 + (a + b + c)\boldsymbol{I}\right] \cdot (-\boldsymbol{D}_\alpha\boldsymbol{G}_1\boldsymbol{q})\,. \quad (5.34)$$

5.2 Efficient implementation

The key to the efficient implementation is to carefully define the intermediate quantities and reuse them to avoid re-computations.

For the optimal degree-0 polynomial approximation, we have, from Eq. (5.3),

$$E_0^{\mathrm{el}}(\boldsymbol{r}) = \frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\boldsymbol{G}_0(\boldsymbol{r})\boldsymbol{q} - \frac{a}{2}(\boldsymbol{G}_1(\boldsymbol{r})\boldsymbol{q})^{\mathsf{T}}\boldsymbol{D}_\alpha\boldsymbol{G}_1(\boldsymbol{r})\boldsymbol{q}\,, \tag{5.35}$$

$$F_{0,k\sigma}^{\mathrm{el}} = -\frac{\partial}{\partial r_{k\sigma}}E_0^{\mathrm{el}}(\boldsymbol{r}) = -\frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_0\right)\boldsymbol{q} + a(\boldsymbol{G}_1\boldsymbol{q})^{\mathsf{T}}\boldsymbol{D}_\alpha\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_1\right)\boldsymbol{q}\,, \tag{5.36}$$

where a is the constant determined by Eq. (5.20). We can define $d_0 = -a\boldsymbol{D}_\alpha\boldsymbol{G}_1\boldsymbol{q}$ and have

$$E_0^{\mathrm{el}} = \frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\boldsymbol{G}_0(\boldsymbol{r})\boldsymbol{q} + \frac{1}{2}\boldsymbol{d}_0{}^{\mathsf{T}}\boldsymbol{G}_1(\boldsymbol{r})\boldsymbol{q}\,, \tag{5.37}$$

$$F_{0,k\sigma}^{\mathrm{el}} = -q_k(\boldsymbol{G}_1\boldsymbol{q})_{k\sigma} - \boldsymbol{d}_0{}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_1\right)\boldsymbol{q}\,. \tag{5.38}$$

Since each $\boldsymbol{G}$ matrix is a sum of its direct sum and reciprocal sum parts, as shown in Eqs. (2.20)–(2.22), we look at each part separately. The direct sum contribution is straightforward:

$$E_0^{\mathrm{dir}} = \frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\boldsymbol{G}_0^{\mathrm{dir}}\boldsymbol{q} + \frac{1}{2}\boldsymbol{d}_0{}^{\mathsf{T}}(\boldsymbol{G}_1^{\mathrm{dir}}\boldsymbol{q})\,, \tag{5.39}$$

$$\boldsymbol{F}_{0,k\sigma}^{\mathrm{dir}} = -q_k(\boldsymbol{G}_1^{\mathrm{dir}}\boldsymbol{q})_{k\sigma} + \sum_{i=1}^{N}\sum_{\alpha=\mathrm{x,y,z}}(\boldsymbol{G}_2^{\mathrm{dir}})_{k\sigma,i\alpha}[(\boldsymbol{d}_0)_{k\alpha}q_i - (\boldsymbol{d}_0)_{i\alpha}q_k] \tag{5.40}$$

Note that $\boldsymbol{G}_1^{\mathrm{dir}}$ is "block skew symmetric" and $\boldsymbol{G}_2^{\mathrm{dir}}$ is block symmetric.

From Eqs. (2.87), (2.88), and (2.90), the reciprocal sum contribution is

$$E_0^{\mathrm{rec}} = \frac{1}{2}\boldsymbol{q}_h{}^{\mathsf{T}}\boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}\boldsymbol{q}_h + \frac{1}{2}\boldsymbol{d}_0{}^{\mathsf{T}}\boldsymbol{I}_h^1\boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}\boldsymbol{q}_h\,. \tag{5.41}$$

$$\boldsymbol{F}_{0,k\sigma}^{\mathrm{rec}} = -q_k(\boldsymbol{I}_h^1\boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}\boldsymbol{q}_h)_{k\sigma} - \boldsymbol{d}_0{}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{I}_h^1\right)\boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}\boldsymbol{q}_h - \boldsymbol{d}_0{}^{\mathsf{T}}\boldsymbol{I}_h^1\boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{I}_h^0\right)^{\mathsf{T}}\boldsymbol{q}\,, \tag{5.42}$$

It is natural to define two intermediate vectors:

$$\boldsymbol{e}_0 = \boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}(\boldsymbol{I}_h^0)^{\mathsf{T}}\boldsymbol{q}\,, \tag{5.43}$$

$$\boldsymbol{e}_1 = \boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}(\boldsymbol{I}_h^1)^{\mathsf{T}}\boldsymbol{d}_0\,, \tag{5.44}$$

so we have (also from Eq. (2.81), and (2.82))

$$E_0^{\mathrm{rec}} = \frac{1}{2}\boldsymbol{q}_h{}^{\mathsf{T}}(\boldsymbol{e}_0+\boldsymbol{e}_1)\,, \tag{5.45}$$

$$\begin{aligned}\boldsymbol{F}_{0,k\sigma}^{\mathrm{rec}} &= -q_k(\boldsymbol{I}_h^1\boldsymbol{e}_0)_{k\sigma} - \boldsymbol{d}_0{}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{I}_h^1\right)\boldsymbol{e}_0 - \boldsymbol{e}_1{}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{I}_h^0\right)^{\mathsf{T}}\boldsymbol{q} \\ &= -q_k\sum_{\vec{n}}(\boldsymbol{I}_h^1)_{k\sigma,\vec{n}}(\boldsymbol{e}_0+\boldsymbol{e}_1)_{\vec{n}} - \sum_{\alpha}(\boldsymbol{d}_0)_{k\alpha}\sum_{\vec{n}}(\boldsymbol{I}_h^2)_{k\sigma\alpha,\vec{n}}(\boldsymbol{e}_0)_{\vec{n}}\,,\end{aligned} \tag{5.46}$$

For the reciprocal sum , we first compute compute $\boldsymbol{e}_0$ by Eq. (5.43), then compute

$$\boldsymbol{d}_0 = -a\boldsymbol{D}_\alpha\boldsymbol{G}_1\boldsymbol{q} = -a\boldsymbol{D}_\alpha(\boldsymbol{G}_1^{\mathrm{dir}}\boldsymbol{q} + \boldsymbol{I}_h^1\boldsymbol{e}_0)\,, \tag{5.47}$$

then compute $\boldsymbol{e}_1$ by Eq. (5.44). In the end, we compute the energy according to Eq. (5.45) and the force according to Eq. (5.46). After the direct sum and the reciprocal sum are computed, we sum up the two parts to get the total electrostatic energy and force. Overall, 4 FFTs (when computing $\boldsymbol{e}_0$ and $\boldsymbol{e}_1$) are needed.

For the optimal degree-1 polynomial approximation, we can do similar analysis. Here we only present the results. First we have

$$\boldsymbol{e}_0 = \boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}(\boldsymbol{I}_h^0)^{\mathsf{T}}\boldsymbol{q}\,, \tag{5.48}$$

$$\boldsymbol{d}_0 = -\boldsymbol{D}_\alpha\boldsymbol{G}_1\boldsymbol{q} = -\boldsymbol{D}_\alpha(\boldsymbol{G}_1^{\mathrm{dir}}\boldsymbol{q} + \boldsymbol{I}_h^1\boldsymbol{e}_0)\,, \tag{5.49}$$

$$\boldsymbol{e}_1 = \boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}(\boldsymbol{I}_h^1)^{\mathsf{T}}\boldsymbol{d}_0\,, \tag{5.50}$$

$$\boldsymbol{d}_1 = (a\boldsymbol{D}_\alpha\boldsymbol{G}_2 + b\boldsymbol{I})\boldsymbol{d}_0 = a\boldsymbol{D}_\alpha(\boldsymbol{G}_2^{\mathrm{dir}}\boldsymbol{d}_0 + \boldsymbol{I}_h^1\boldsymbol{e}_1) + b\boldsymbol{d}_0\,, \tag{5.51}$$

$$\boldsymbol{e}_2 = \boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}(\boldsymbol{I}_h^1)^{\mathsf{T}}\boldsymbol{d}_1\,. \tag{5.52}$$

where a and b are computed from Eqs. (5.15) and (5.16). The energy is

$$E_1^{\rm el} = \frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\boldsymbol{G}_0\boldsymbol{q} + \frac{1}{2}\boldsymbol{d}_1{}^{\mathsf{T}}\boldsymbol{G}_1\boldsymbol{q} = \frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\boldsymbol{G}_0^{\rm dir}\boldsymbol{q} + \frac{1}{2}\boldsymbol{d}_1{}^{\mathsf{T}}(\boldsymbol{G}_1^{\rm dir}\boldsymbol{q}) + \frac{1}{2}\boldsymbol{q}_h{}^{\mathsf{T}}(\boldsymbol{e}_0+\boldsymbol{e}_2)\,. \tag{5.53}$$

Accordingly, the force is

$$\begin{aligned}
\boldsymbol{F}_{1,k\sigma} &= -\frac{\partial}{\partial r_{k\sigma}}E_1^{\rm el} = -\frac{\partial}{\partial r_{k\sigma}}\left[\frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\boldsymbol{G}_0\boldsymbol{q} - \frac{1}{2}(\boldsymbol{G}_1\boldsymbol{q})^{\mathsf{T}}(a\boldsymbol{D}_\alpha\boldsymbol{G}_2\boldsymbol{D}_\alpha + b\boldsymbol{D}_\alpha)\boldsymbol{G}_1\boldsymbol{q}\right], \\
&= -q_k(\boldsymbol{G}_1\boldsymbol{q})_{k\sigma} + \left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_1\boldsymbol{q}\right)^{\mathsf{T}}(a\boldsymbol{D}_\alpha\boldsymbol{G}_2\boldsymbol{D}_\alpha + b\boldsymbol{D}_\alpha)\boldsymbol{G}_1\boldsymbol{q} + \frac{a}{2}(\boldsymbol{G}_1\boldsymbol{q})^{\mathsf{T}}\boldsymbol{D}_\alpha\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_2\right)\boldsymbol{D}_\alpha\boldsymbol{G}_1\boldsymbol{q} \\
&= -q_k(\boldsymbol{G}_1\boldsymbol{q})_{k\sigma} - \boldsymbol{d}_1{}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_1\boldsymbol{q}\right) + \frac{a}{2}\boldsymbol{d}_0{}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_2\right)\boldsymbol{d}_0\,.
\end{aligned} \tag{5.54}$$

The direct sum contribution is

$$\begin{aligned}
\boldsymbol{F}_{1,k\sigma}^{\rm dir} &= -q_k(\boldsymbol{G}_1^{\rm dir}\boldsymbol{q})_{k\sigma} + \sum_{i\alpha}(\boldsymbol{G}_2^{\rm dir})_{k\sigma,i\alpha}[(\boldsymbol{d}_1)_{k\alpha}q_i - (\boldsymbol{d}_1)_{i\alpha}q_k] \\
&\quad + a\sum_{i\alpha\beta}(\boldsymbol{d}_0)_{k\alpha}(\boldsymbol{G}_3^{\rm dir})_{k\sigma\alpha,i\beta}(\boldsymbol{d}_0)_{i\beta}\,,
\end{aligned} \tag{5.55}$$

and the reciprocal sum contribution is

$$\begin{aligned}
\boldsymbol{F}_{1,k\sigma}^{\rm rec} &= -q_k\sum_{\vec{n}}(\boldsymbol{I}_h^1)_{k\sigma,\vec{n}}(\boldsymbol{e}_0+\boldsymbol{e}_2)_{\vec{n}} - \sum_\alpha(\boldsymbol{d}_1)_{k\alpha}\sum_{\vec{n}}(\boldsymbol{I}_h^2)_{k\sigma\alpha,\vec{n}}(\boldsymbol{e}_0)_{\vec{n}} \\
&\quad + a\sum_\alpha(\boldsymbol{d}_0)_{k\alpha}\sum_{\vec{n}}(\boldsymbol{I}_h^2)_{k\sigma\alpha,\vec{n}}(\boldsymbol{e}_1)_{\vec{n}}\,.
\end{aligned} \tag{5.56}$$

For the optimal degree-2 polynomial approximation, we start with

$$\begin{aligned}
\boldsymbol{e}_0 &= \boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}(\boldsymbol{I}_h^0)^{\mathsf{T}}\boldsymbol{q}\,, && (5.57)\\
\boldsymbol{d}_0 &= -\boldsymbol{D}_\alpha\boldsymbol{G}_1\boldsymbol{q} = -\boldsymbol{D}_\alpha(\boldsymbol{G}_1^{\mathrm{dir}}\boldsymbol{q} + \boldsymbol{I}_h^1\boldsymbol{e}_0)\,, && (5.58)\\
\boldsymbol{e}_1 &= \boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}(\boldsymbol{I}_h^1)^{\mathsf{T}}\boldsymbol{d}_0\,, && (5.59)\\
\boldsymbol{d}_1 &= (\boldsymbol{I} - \boldsymbol{D}_\alpha\boldsymbol{G}_2)\boldsymbol{d}_0 = \boldsymbol{d}_0 - \boldsymbol{D}_\alpha(\boldsymbol{G}_2^{\mathrm{dir}}\boldsymbol{d}_0 + \boldsymbol{I}_h^1\boldsymbol{e}_1)\,, && (5.60)\\
\boldsymbol{e}_2 &= \boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}(\boldsymbol{I}_h^1)^{\mathsf{T}}\boldsymbol{d}_1\,, && (5.61)\\
\boldsymbol{d}_2 &= (a\boldsymbol{D}_\alpha\boldsymbol{G}_2\boldsymbol{D}_\alpha\boldsymbol{G}_2 + b\boldsymbol{D}_\alpha\boldsymbol{G}_2 + c\boldsymbol{I})\boldsymbol{d}_0\\
&= -a\boldsymbol{D}_\alpha\boldsymbol{G}_2\boldsymbol{d}_1 - (a+b)\boldsymbol{d}_1 + (a+b+c)\boldsymbol{d}_0\,, && (5.62)\\
\boldsymbol{e}_3 &= \boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}(\boldsymbol{I}_h^1)^{\mathsf{T}}\boldsymbol{d}_2\,. && (5.63)
\end{aligned}$$

where a, b, and c are computed by Eqs. (5.30)–(5.32). The definition of $\boldsymbol{d}_1$ is motivated by Neumann expansion (see Eq. (5.68)). We have

$$\begin{aligned}
E_2^{\mathrm{el}} &= \frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\boldsymbol{G}_0\boldsymbol{q} + \frac{1}{2}\boldsymbol{d}_2{}^{\mathsf{T}}\boldsymbol{G}_1\boldsymbol{q} = \frac{1}{2}\boldsymbol{q}^{\mathsf{T}}\boldsymbol{G}_0^{\mathrm{dir}}\boldsymbol{q} + \frac{1}{2}\boldsymbol{d}_2{}^{\mathsf{T}}(\boldsymbol{G}_1^{\mathrm{dir}}\boldsymbol{q}) + \frac{1}{2}\boldsymbol{q}_h{}^{\mathsf{T}}(\boldsymbol{e}_0+\boldsymbol{e}_3)\,, && (5.64)\\
\boldsymbol{F}_{2,k\sigma} &= -q_k(\boldsymbol{G}_1\boldsymbol{q})_{k\sigma} - \boldsymbol{d}_2{}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_1\right)\boldsymbol{q} + (a+\frac{b}{2})\boldsymbol{d}_0{}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_2\right)\boldsymbol{d}_0\\
&\quad -a\boldsymbol{d}_0{}^{\mathsf{T}}\left(\frac{\partial}{\partial r_{k\sigma}}\boldsymbol{G}_2\right)\boldsymbol{d}_1\,, && (5.65)\\
\boldsymbol{F}_{2,k\sigma}^{\mathrm{dir}} &= -q_k(\boldsymbol{G}_1^{\mathrm{dir}}\boldsymbol{q})_{k\sigma} + \sum_{i\alpha}(\boldsymbol{G}_2^{\mathrm{dir}})_{k\sigma,i\alpha}[(\boldsymbol{d}_2)_{k\alpha}q_i - (\boldsymbol{d}_2)_{i\alpha}q_k]\\
&\quad + \sum_{i\alpha\beta}(\boldsymbol{G}_3^{\mathrm{dir}})_{k\sigma\alpha,i\beta}\big[(2a+b)(\boldsymbol{d}_0)_{k\alpha}(\boldsymbol{d}_0)_{i\beta} - a(\boldsymbol{d}_0)_{k\alpha}(\boldsymbol{d}_1)_{i\beta} - a(\boldsymbol{d}_0)_{i\beta}(\boldsymbol{d}_1)_{k\alpha}\big]\,, && (5.66)\\
\boldsymbol{F}_{2,k\sigma}^{\mathrm{rec}} &= -q_k\sum_{\vec{n}}(\boldsymbol{I}_h^1)_{k\sigma,\vec{n}}(\boldsymbol{e}_0+\boldsymbol{e}_3)_{\vec{n}} - \sum_\alpha(\boldsymbol{d}_2)_{k\alpha}\sum_{\vec{n}}(\boldsymbol{I}_h^2)_{k\sigma\alpha,\vec{n}}(\boldsymbol{e}_0)_{\vec{n}}\\
&\quad + \sum_\alpha\big[(2a+b)(\boldsymbol{d}_0)_{k\alpha} - a(\boldsymbol{d}_1)_{k\alpha}\big]\sum_{\vec{n}}(\boldsymbol{I}_h^2)_{k\sigma\alpha,\vec{n}}(\boldsymbol{e}_1)_{\vec{n}}\\
&\quad -a\sum_\alpha(\boldsymbol{d}_0)_{k\alpha}\sum_{\vec{n}}(\boldsymbol{I}_h^2)_{k\sigma\alpha,\vec{n}}(\boldsymbol{e}_2)_{\vec{n}}\,. && (5.67)
\end{aligned}$$

Note that the above expressions reuse partial computation results as much as possible. The computation sequence is (i) compute $\boldsymbol{e}$, $\boldsymbol{d}$ sequence up to $\boldsymbol{d}_k$ and $\boldsymbol{e}_{k+1}$, (ii) compute the energy, (iii) compute the force. Overall, for a degree-k polynomial approximation, $2(k+2)$ FFTs are needed to compute the $\boldsymbol{e}$ vectors. Other parts of the computation are $O(N)$.

5.3 Results

The reason to use approximate dipoles is to avoid secular energy drift. It is evident in Fig. 5.1 that this goal is achieved. For the momentum drift graph, the y-axis shows the ratio of the magnitude of the total momentum to the magnitude of the thermal momentum at 300 K, a measure which is discussed in presenting Fig. 2.4. We do not zero out the reciprocal sum contribution to the electrostatic force in PME in Fig. 5.1. If we do, we see a small energy drift (Fig. 5.2).

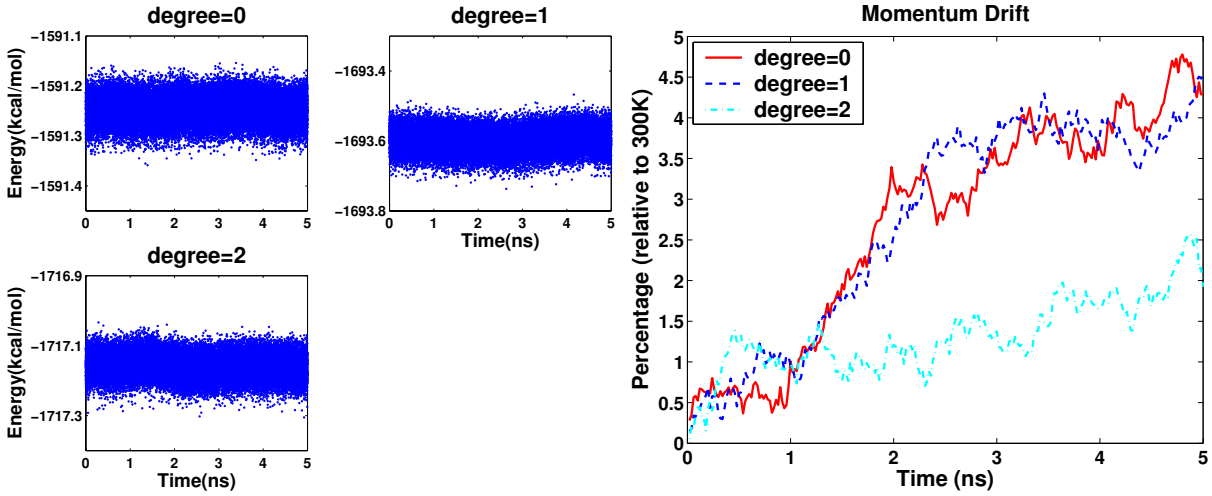

Figure 5.1: Energies and momenta for the non-iterative methods.

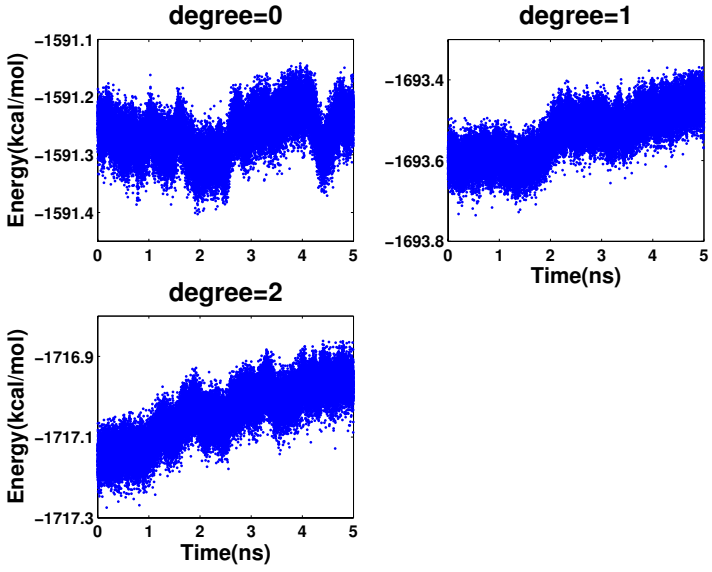

Figure 5.2: Small energy drifts for the non-iterative methods caused by PME.

The accuracy of the non-iterative method is low. The errors and computational cost are shown in Table 5.1 and Table 5.2 respectively. These errors are one order of magnitude larger than those in Table 4.1, which are obtained with more computational effort (see Table 3.4).

degree	0	1	2
δd (ppm)	10877	2787	3949
δF (ppm)	5222	6765	1197
δE (ppm)	5.85	1.66	2.52

Table 5.1: Errors of non-iterative methods.

degree	0	1	2
computational cost in work units	1.28	1.62	2.05

Table 5.2: Computational costs of non-iterative methods.

Table 5.3 summarizes the result for computing various physical quantities of RPOL water models. Percentages enclosed in parentheses are relative errors compared to those in the first row which are obtained from much more accurate self-consistent computations. The Neumann polynomial refers to the expansion used in [75]:

$$(\boldsymbol{D}_\alpha^{-1} + \boldsymbol{G}_2)^{-1} \quad = \quad \boldsymbol{D}_\alpha - \boldsymbol{D}_\alpha \boldsymbol{G}_2 \boldsymbol{D}_\alpha + (\boldsymbol{D}_\alpha \boldsymbol{G}_2)^2 \boldsymbol{D}_\alpha + \cdots . \tag{5.68}$$

For the Neumann expansion, a polynomial of at least degree 2 is needed to have an acceptable accuracy. The optimal expansion used in our computation is more accurate and a degree-1 optimal polynomial is probably accurate enough.

Polynomial Type	degree	Potential Energy (kcal/mol)	Diff. Const. (10^{-5} cm^2/s)	Dielectric Constant	Mol. Dipole (Debye)
self-consistent		-9.88	2.44	119.7	2.604
Neumann	0	-8.92 (-9.7%)	4.20 (72.1%)	101.2 (-15.4%)	2.440 (-6.3%)
	1	-9.62 (-2.6%)	2.92 (19.7%)	111.7 ($-$ 6.7%)	2.560 (-1.7%)
	2	-9.82 (-0.6%)	2.58 (5.7%)	118.8 ($-$ 0.8%)	2.593 (-0.4%)
Optimal	0	-9.39 (-5.0%)	3.44 (41.0%)	102.8 (-14.5%)	2.520 (3.2%)
	1	-9.97 (-0.9%)	2.40 (1.6%)	127.5 (6.5%)	2.620 (0.6%)
	2	-9.90 (-0.2%)	2.45 (0.3%)	122.0 (1.9%)	2.607 (0.1%)

Table 5.3: Physical quantities and errors of the RPOL model computed by the non-iterative methods.

Appendix A

Miscellaneous

Section A.1 presents two physical water models studied extensively in our research: the polarizable RPOL water model, and the non-polarizable SPC water model, with the emphasis on the former. Section A.2 presents the tests we have designed and implemented to debug the code. Section A.3 provides some β-independent quantities and an equality for the $\boldsymbol{G}$ matrices. Section A.4 describes a pitfall related to the misuse of velocity when the constraints are present.

A.1 Mathematical models

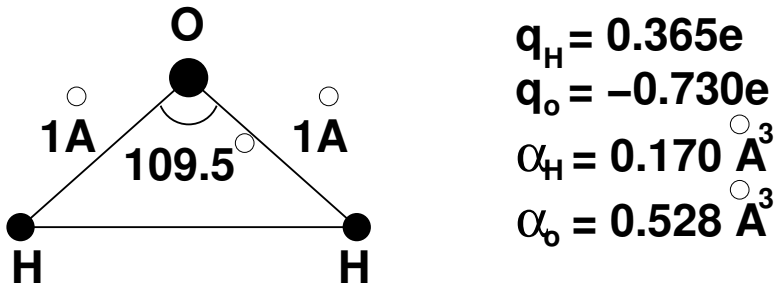

Figure A.1: RPOL water geometry

In the revised polarizable (RPOL) water model [36], water molecules are rigid. The distance between the oxygen atom and a hydrogen atom is 1 Å and the H–O–H angle is 109.5°. The Lennard-Jones interaction exists only between oxygen atoms:

$$E_{\mathrm{LJ}}(r) \quad = \quad 4\epsilon\big[(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6}\big]\,, \tag{A.1}$$

with $\sigma = 3.196\,\text{Å}$ and $\epsilon = 0.160\,\text{kcal/mol}$. Effective charges are $q_\text{O} = -0.730\,\text{e}$ and $q_\text{H} = 0.365\,\text{e}$, where e is the charge of a proton. The isotropic polarizabilities are $\alpha_\text{O} = 0.52\text{Å}^3$ and $\alpha_\text{H} = 0.170\text{Å}^3$. Electrostatic interactions between atoms in the same molecule are excluded.

Some typical values for an RPOL water system are helpful in understanding our results:

$$\frac{\|\boldsymbol{F}^\text{dipole}\|_2}{\|\boldsymbol{F}^\text{el}\|_2} \approx 28\%, \qquad \frac{|E^\text{dipole}|}{|E^\text{el}|} \approx 22\%, \tag{A.2}$$

$$\langle|\vec{d}_i|\rangle \approx 0.25 \text{ Debye}, \quad \langle|\vec{d}_\text{O}|\rangle \approx 0.38 \text{ Debye}, \quad \langle|\vec{d}_\text{H}|\rangle \approx 0.14 \text{ Debye}, \tag{A.3}$$

where $\boldsymbol{F}$ are forces, E is energy, E^dipole includes charge–dipole and dipole–dipole interactions, and $\langle|\vec{d}_i|\rangle$ stands for average magnitude of a dipole

$$\langle|\vec{d}_i|\rangle \;=\; \sqrt{\frac{1}{N}\sum_{i=1}^{N} \vec{d}_i \cdot \vec{d}_i}\,. \tag{A.4}$$

The simulated system has 216 RPOL water molecules. With density 0.99 g/cm^3, the system size is 18.688Å. The Lennard-Jones potential has an 8 Å cutoff. For constant energy simulations, we apply a switching function to make the potential C^1. So for $r_\text{s} < r < r_\text{c}$, where $r_\text{s} = 6\,\text{Å}$ is the switching radius, and $r_\text{c} = 8\,\text{Å}$ is the cutoff radius, the potential is multiplied by a switching function $s(r)$, with

$$s(r) \;=\; \frac{(r_\text{c}^2 - r^2)^2(r_\text{c}^2 + 2r^2 - 3r_\text{s}^2)}{(r_\text{c}^2 - r_\text{s}^2)^3}\,. \tag{A.5}$$

$s(r)$ is chosen to be a function of r^2 for fast computation and it satisfies

$$s(r_\text{s}) = 1\,, \qquad \frac{\mathrm{d}}{\mathrm{d}r}s(r_\text{s}) = 0\,, \tag{A.6}$$

$$s(r_\text{c}) = 0\,, \qquad \frac{\mathrm{d}}{\mathrm{d}r}s(r_\text{c}) = 0\,. \tag{A.7}$$

Because the constant temperature simulations are meant to be compared to [140], Berendsen's rescaling method [13] is used, a long range correction [3, §2.8] for the Lennard-Jones potential is included, and the switching is turned off. The major result is summarized in Table A.1. Our self-consistent computation uses a relative RMS convergence criteria of 4 ppm.

	Potential (kcal/mol)	Diff. Const. (10^{-5} cm^2/s)	Dielectric Constant	Mol. Dipole (Debye)
[140]	-9.88	2.4 ± 0.2	115	2.604
Ours	-9.88	2.5 ± 0.2	117	2.604

Table A.1: Physical quantities of the RPOL water model.

The radial distribution functions are shown in Fig. A.2. The result is obtained from a 10 ps simulation, positions are stored every 0.1 ps. They are almost the same as those in [140]. The only observable difference is the peak value of $g_{\mathrm{OO}}(r)$, Ours is a little smaller, but we are consistent with [127].

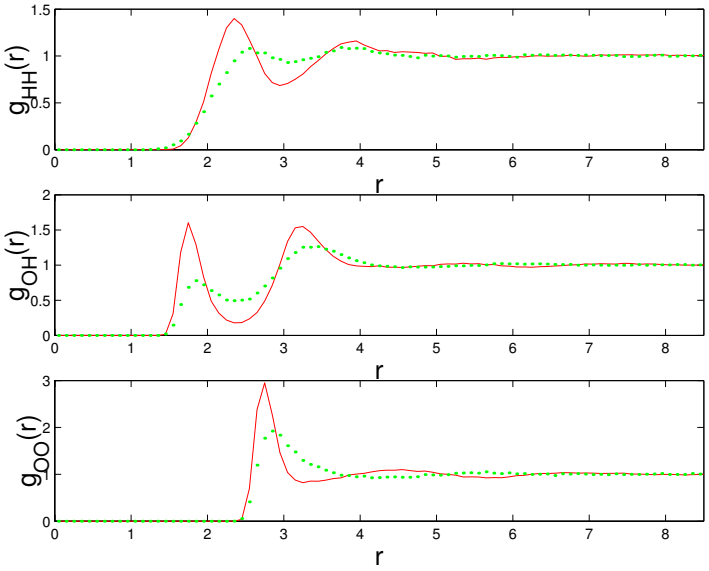

Figure A.2: Radial distribution function of RPOL water. The solid line is for 300K, the dotted line is for 573K.

In the non-polarizable simple point charge (SPC) water model [14], water molecules are rigid. The O–H bond length is 1 Å and the H–O–H bond angle is 109.28°. The hydrogen atom has charge 0.41 e, and the oxygen atom has charge -0.82 e; the Lennard-Jones interaction is represented as

$$E_{\mathrm{LJ}}(r) \quad = \quad -(\frac{A}{r})^6 + (\frac{B}{r})^{12}\,, \tag{A.8}$$

with $A = 2.924\,\text{Å}(\text{kcal/mol})^{1/6}$ and $B = 3.043\,\text{Å}(\text{kcal/mol})^{1/12}$.

A.2 Tests

Tests are important to ensure implementation correctness. The tests we have carried out are summarized in the following list:

1. Madelung constant test

   The system has 8 charges, sitting on vertices of a cube centered in the simulation box. The cubic side length is half of that of the simulation box. The nearest neighbors of each +e charge are −e charges. For such a system, the total electrostatic energy can be computed by

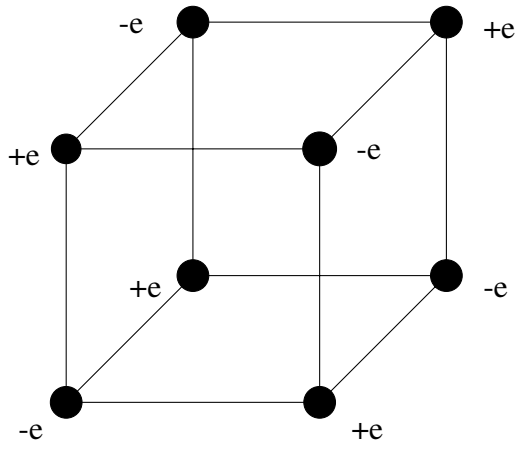

Figure A.3: Madelung system, the cubic has side length half of that of the simulation box.

   the "Madelung constant" [34, page 73–79], a physical constant computed in the same way as the Ewald sum, namely, summation over each box, then summation over spherical shells of boxes to infinity. The Madelung constant has been computed to a very precise level, so it can be used to represent the exact value. The results are (the unit is $e^2/\text{Å}$)

Ewald sum	-0.19423898606067433
Exact value	-0.19423898606067430

   The Ewald sum value is computed from C code, with accuracy, (ϵ in Eqs. (2.25) and (2.26)) set to be 1×10^{-20}. By symmetry, the force acting on each atom should be zero, while the C code output gives $\|F\|_\infty < 2 \times 10^{-19}$, smaller than machine ϵ.

2. reimplementation of the energy and force computation by Matlab scripts

Although much slower to execute, the script can be written down much easier, and in a relatively high-level way. Then a comparison of energy and force is made between the two implementations for the same set of atom positions. For the Ewald sum, the comparison is made for a 408 artificial atom system, which takes 12 hours for the scripts to finish. The differences between the Matlab scripts output and the C code implementation are

$$\|\boldsymbol{d}_{\mathrm{c}} - \boldsymbol{d}_{\mathrm{m}}\|_\infty = 3.3 \times 10^{-16}, \qquad \frac{\|\boldsymbol{d}_{\mathrm{c}} - \boldsymbol{d}_{\mathrm{m}}\|_2}{\|\boldsymbol{d}_c\|_2} = 2.8 \times 10^{-15}\,, \tag{A.9}$$

$$|E_{\mathrm{c}} - E_{\mathrm{m}}| = 1.0 \times 10^{-15}, \qquad \frac{|E_{\mathrm{c}} - E_{\mathrm{m}}|}{|E_{\mathrm{c}}|} = 2.1 \times 10^{-14}\,, \tag{A.10}$$

$$\|\boldsymbol{F}_{\mathrm{c}} - \boldsymbol{F}_{\mathrm{m}}\|_\infty = 1.1 \times 10^{-16}, \qquad \frac{\|\boldsymbol{F}_{\mathrm{c}} - \boldsymbol{F}_{\mathrm{m}}\|_2}{\|F_{\mathrm{c}}\|_2} = 2.6 \times 10^{-15}\,, \tag{A.11}$$

where the subscript "c" means the result is from C code, "m" means the result is from the Matlab script. Comparison is also made for every element of the $\boldsymbol{G}$ matrices for a three particle system. The artificial system has side length 1, and the three particles have charge e, -0.7e and -0.3e. Their locations are random. The accuracy (ϵ in Eqs. (2.25) and (2.26)) for the Ewald sum is 1×10^{-6}. The differences are summarized in Table A.2.

Difference	$\boldsymbol{G}_0$	$\boldsymbol{G}_1$	$\boldsymbol{G}_2$	$\boldsymbol{G}_2^{\mathrm{dir}}$	$\boldsymbol{G}_3^{\mathrm{dir}}$
absolute	3.9×10^{-16}	8.9×10^{-16}	7.1×10^{-16}	8.9×10^{-15}	8.5×10^{-14}
relative	1.9×10^{-15}	3.4×10^{-16}	5.1×10^{-16}	6.4×10^{-16}	9.8×10^{-16}

Table A.2: Comparison between the C code and the Matlab script implementations.

3. β-independence tests

The electrostatic energy should be independent of β. In the following table, E is computed with Ewald parameters set for MD simulation, E_{exact} is computed with accuracy of 1×10^{-20}.

β	E	E_{exact}
0.1	-1.166737	-1.16673608068544
0.2	-1.166736	-1.16673608068545
0.3	-1.166736	-1.16673608068546
0.4	-1.166738	-1.16673608068546
0.5	-1.166738	-1.16673608068548
0.6	-1.166736	-1.16673608068549
0.7	-1.166738	-1.16673608068551
0.8	-1.166738	-1.16673608068556

The relative change of E with respect to β is about 2×10^{-6}, while that of E_{exact} is 1×10^{-13}.

4. the Ewald energy and force of a system of only one dipole are 0, by Eq. (A.14).

   A test shows they are 0 within the machine ϵ.

5. $\alpha \to 0$ test

   Since $\boldsymbol{d} = \boldsymbol{D}_\alpha(-\boldsymbol{G}_1\boldsymbol{q} - \boldsymbol{G}_2\boldsymbol{d})$, as $\boldsymbol{D}_\alpha \to 0$, the energy cost for polarization ($\boldsymbol{d}^{\mathsf{T}}\boldsymbol{D}_\alpha^{-1}\boldsymbol{d}/2$) drives $\boldsymbol{d}$ to be smaller and smaller, so that $\boldsymbol{G}_2\boldsymbol{d} \ll \boldsymbol{G}_1\boldsymbol{q}$, and $\boldsymbol{d} \approx -\boldsymbol{D}_\alpha\boldsymbol{G}_1\boldsymbol{q}$. A test shows this is truly the case: a linear fit between $\|\boldsymbol{d}\|$ and $\| - \boldsymbol{D}_\alpha\boldsymbol{G}_1\boldsymbol{q}\|$ gives a slope of 1.003.

A.3 β-independence

The Ewald sum is independent of the parameter β. Here we present three more β-independent quantities and one equality. The three β-independent quantities are

$$(\boldsymbol{G}_0)_{ii} - (\boldsymbol{G}_0)_{ij}\,, \quad \boldsymbol{G}_1\,, \quad \text{and } \boldsymbol{G}_2. \tag{A.12}$$

The total electrostatic energy for a two-charge system (charges are q and $-q$) with the periodic boundary conditions, given by (2.2) with $\vec{d} = 0$, is independent of β. This means $(\boldsymbol{G}_0)_{11} - (\boldsymbol{G}_0)_{12}$ is independent of β. Since the result should not depend on the choice of charges, it must be true for arbitrary pairs. Taking the derivative of $(\boldsymbol{G}_0)_{ii} - (\boldsymbol{G}_0)_{ij} - (2\beta)/\sqrt{\pi}$ with respect to the position, we see all non-diagonal blocks of $\boldsymbol{G}_1$ are independent of β, because $(\boldsymbol{G}_0)_{ii}$ is independent of the

position. The diagonal blocks of $\boldsymbol{G}_1$ are zero. Looking it in another way, the quantity $-\boldsymbol{G}_1\boldsymbol{q}$ is the electric field generated by the charge. So it must be independent of β. Because $\boldsymbol{q}$ can be an arbitrary vector, $\boldsymbol{G}_1$ must be independent of β. If a system has no charges, but only dipoles, the total electrostatic energy, which is $\frac{1}{2}\boldsymbol{d}^{\mathsf{T}}\boldsymbol{G}_2\boldsymbol{d}+\frac{2\pi}{3V}d^{\mathsf{T}}\boldsymbol{d}$ according to Eq. (2.2), should be independent of β. Since $\boldsymbol{d}$ can be arbitrary, $\boldsymbol{G}_2$ must be independent of β.

The one equality is

$$(\boldsymbol{G}_2)_{i\alpha,i\alpha} \;=\; -\frac{4\pi}{3V}\,, \tag{A.13}$$

where V is the volume of the simulation box. Consider a cubic system having only one dipole $\vec{d}$ having the same component along each axis ($d_x = d_y = d_z$) in periodic boundary conditions. The total electrostatic energy, summed in the same order as the Ewald summation shown in Eq. (2.1), is [70, (4.20)]

$$\begin{aligned}
E^{\mathrm{el}} &= \lim_{R\to\infty}{\sum_{|\vec{n}|<R}}'\frac{1}{2}\left[\frac{\vec{d}\cdot\vec{d}}{|\vec{n}|^3}-\frac{3(\vec{n}\cdot\vec{d})^2}{|\vec{n}|^5}\right]\\
&= \frac{1}{2}\lim_{R\to\infty}\left[|\vec{d}|^2{\sum_{|\vec{n}|<R}}'\frac{1}{|\vec{n}|^3}-3(d_x^2{\sum_{|\vec{n}|<R}}'\frac{n_x^2}{|\vec{n}|^5}+d_y^2{\sum_{|\vec{n}|<R}}'\frac{n_y^2}{|\vec{n}|^5}+d_z^2{\sum_{|\vec{n}|<R}}'\frac{n_z^2}{|\vec{n}|^5})\right]\\
&= \frac{1}{2}\lim_{R\to\infty}\left[|\vec{d}|^2{\sum_{|\vec{n}|<R}}'\frac{1}{|\vec{n}|^3}-|\vec{d}|^2{\sum_{|\vec{n}|<R}}'\frac{1}{|\vec{n}|^3}\right]\\
&= \lim_{R\to\infty}0=0\,,
\end{aligned} \tag{A.14}$$

where $\sum'$ means the summation excludes the $\vec{n}=0$ term. Eq. (2.2) should give the same result:

$$\frac{1}{2}\boldsymbol{d}^{\mathsf{T}}\boldsymbol{G}_2\boldsymbol{d}+\frac{2\pi}{3V}\boldsymbol{d}^{\mathsf{T}}\boldsymbol{d} \;=\; 0\,. \tag{A.15}$$

So the diagonal elements of $\boldsymbol{G}_2$, which should be all equal, must satisfy (A.13).

A.4 A velocity rescaling pitfall in constraint dynamics

For comparison purpose, we implement the Berendsen's rescaling method [13] since it is used in [140]. We are aware that the method can lead to problems such as flying ice cube [64, 32].

The implementation of the rescaling method is tricky when holonomic constraints (e.g., constant bond lengths) are present. SHAKE [118] or RATTLE [4] is usually used to enforce the constraint. So if the velocity-Verlet method is the underlying integration algorithm, we have the following pseudocode

```
for i = 1,2, ...
    half kick
    drift
    SHAKE (modify position and velocity)
    half kick
    (RATTLE, modify velocity)
    compute and output energy
end
```

After the drift step, the positions do not satisfy the holonomic constraints, so SHAKE changes the position and modifies velocity so that it is the centered difference of positions at two consecutive timesteps. At the end of each MD step, the optional RATTLE further modifies the velocities to make them satisfy their implicit constraints.

Unphysical velocities that do not satisfy the constraints appear temporarily right after each half-kick step. In particular, the velocities of light atoms, such as hydrogens, are unphysically large due to their small masses. It is wrong to rescale the velocity right after a kick step. Doing this effectively drives the system to a lower temperature. The computed physical quantities may not show observable errors if they are not very sensitive to temperature. So sometimes, many of them seem right except one or two, making the user confused. This problem can cause troubles in implementing the Nosé-Hoover [95, 68] or the Nosé-Poincaré [18] method too. The correct way is always rescaling physical velocities, e.g., after the RATTLE step. If the leapfrog formulation is used and we do not compute the velocities at integer timesteps, the rescaling step should be done after the SHAKE step, as shown in reference [13].

References

[1] P. Ahlström, A. Wallqvist, S. Engström, and B. Jönsson. A molecular dynamics study of polarizable water. *Mol. Phys.*, 68:563–581, 1989.

[2] J. Alejandre and D. J. Tildesley. Molecular dynamics simulation of the orthobaris densities and surface tension of water. *J. Chem. Phys.*, 102:4574–4583, 1995.

[3] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Clarendon Press, Oxford, New York, 1987. Reprinted in paperback in 1989 with corrections.

[4] H. C. Andersen. Rattle: A 'velocity' version of the shake algorithm for molecular dynamics calculations. *J. Comput. Phys.*, 52:24–34, 1983.

[5] J. Applequist. An atom dipole interaction model for molecular optical properties. *Molecular Optical Properties*, 10:79–85, 2977.

[6] J. Applequist, J. R. Carl, and K. K. Fung. An atom dipole interaction model for molecular polarizability: Application to polyatomic molecules and determination of atom polarizabilities. *J. Am. Chem. Soc.*, 94:2952–2960, 1972.

[7] V. I. Arnol'd. *Mathematical Methods of Classical Mechanics*. Springer-Verlag, New York, second edition, 1989.

[8] J. S. Bader, C. M. Cortis, and B. J. Berne. Solvation and reorganization energies in polarizable molecular and continuum solvents. *J. Chem. Phys.*, 106:2372–2387, 1997.

[9] J. L. Banks, G. A. Kaminski, R. Zhou, D. T. Mainz, B. B. Berne, and R. A. Friesner. Parameterizing a polarizable force field from ab initio data. I. the fluctuating point charge model. *J. Chem. Phys.*, 110:741–754, 1999.

[10] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994. On–line at ftp.netlib.org/templates/templates.ps.

[11] J. Baucom, T. Transue, M. Fuentes-Cabrera, J. M. Krahn, T. A. Darden, and C. Sagui. Molecular dynamics simulations of the $d(\text{ccaacgttgg})_2$ decamer in crystal environment: Comparison of atomic point-charge, extra-point, and polarizable force fields. *J. Chem. Phys.*, 121:6998–7008, 2004.

[12] M. D. Beachy, D. Chasman, R. B. Murphy, T. A. Halgren, and R. A. Friesner. Accurate ab initio quantum chemical determination of the relative energetics of peptide conformations and assessment of empirical force fields. *J. Am. Chem. Soc.*, 119:5908–5920, 1997.

[13] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak. Molecular dynamics with coupling to an external bath. *J. Chem. Phys.*, 81(8):3684–3690, 1984.

[14] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, and J. Hermans. Interaction models for water in relation to protein hydration. In B. Pullman, editor, *Intermolecular Forces*, pages 331–342. D. Reidel Publishing Company, 1981.

[15] H. J. C. Berendsen, D. van der Spoel, and R. van Drunen. GROMACS: A message-passing parallel molecular dynamics implementation. *Comput. Phys. Comm.*, 91:43–56, 1995.

[16] M. Bergdorf, C. Peter, and P. H. Hünenberger. Influence of cut-off truncation and artificial periodicity of electrostatic interactions in molecular simulations of solvated ions: A continuum electrostatics study. *J. Chem. Phys.*, 119:9129–9144, 2003.

[17] D. N. Bernado, Y. Ding, K. Krogh-Jespersen, and R. M. Levy. An anisotropic polarizable water model: Incorporation of all–atom polarizabilities into molecular dynamics force fields. *J. Phys. Chem.*, 98:4180–4187, 1994.

[18] S. D. Bond, B. J. Leimkuhler, and B. B. Laird. The Nosé-Poincaré method for constant temperature molecular dynamics. *J. Comput. Phys*, 151(1):114–134, May 1, 1999.

[19] F. A. Bornemann and C. Schütte. A mathematical investigation of the Car-Parrinello method. *Numerische Mathematik*, 78:259–376, 1998.

[20] A. Brandt and A. A. Lubrecht. Multilevel matrix multiplication and fast solution of integral equations. *J. Comput. Phys.*, 90:348–370, 1990.

[21] J. Brodhold, M. Sampoli, and R. Vallauri. Parameterizing a polarizable intermolecular potential for water. *Mol. Phys.*, 86:149–158, 1995.

[22] B. Bursulaya, J. Jeon, D. Zichi, and H. Kim. Generalized molecular mechanics including quantum electronic structure variation of polar solvents. II. a molecular dynamics simulation study of water. *J. Chem. Phys.*, 108:3286–3295, 1998.

[23] B. Bursulaya and H. Kim. Generalized molecular mechanics including quantum electronic structure variation of polar solvents. I. theoretical formulation via a truncated adiabatic basis set description. *J. Chem. Phys.*, 108:3277–3285, 1998.

[24] J. Caldwell, L. X. Dang, and P. A. Kollman. Implementation of nonadditive intermolecular potentials by use of molecular dynamics: development of a water–water potential and water–ion cluster interactions. *J. Am. Chem. Soc.*, 112:9144–9147, 1990.

[25] J. W. Caldwell and P. A. Kollman. Cation-π interactions: nonadditive effects are critical in their accurate representation. *J. Am. Chem. Soc.*, 117:4177–4178, 1995.

[26] J. W. Caldwell and P. A. Kollman. Structure and properties of neat liquids using nonadditive molecular dynamics: water, methanol and n-methylacetamide. *J. Phys. Chem.*, 99:6208–6219, 1995.

[27] R. Car and M. Parrinello. Unified approach for molecular dynamics and density-functional theory. *Phys. Rev. Lett.*, 55:2471–2474, 1985.

[28] M. Carrillo-Tripp, H. Saint-Martin, and I. Ortega-Blake. A comparative study of the hydration of Na^+ and K^+ with refined polarizable model potentials. *J. Chem. Phys.*, 118:7062–7073, 2003.

[29] R. Chelli and P. Procacci. A transferable polarizable electrostatic force field for molecular mechanics based on the chemical potential equalization principle. *J. Chem. Phys.*, 117:9175–9189, 2002.

[30] B. Chen and J. I. Siepmann. Monte Carlo algorithms for simulating systems with adiabatic separation of electronic and nuclear degrees of freedom. *Theoret. Chem. Acc.*, 103:87–104, 1999.

[31] A. A. Chialvo and P. T. Cummings. Engineering a simple polarizable model for the molecular simulation of water applicable over wide ranges of state conditions. *J. Chem. Phys.*, 105:8274–8281, 1996.

[32] S. Chiu, M. Clark, S. Subramaniam, and E. Jakobsson. Collective motion artifacts arising in long-duration molecular dynamics simulations. *J. Comput. Chem.*, 21(2):121–131, Jan. 30, 2000.

[33] W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gould, J. K. M. Merz, D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. W. Caldwell, and P. A. Kollman. A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. *J. Am. Chem. Soc.*, 117:5179–5197, 1995.

[34] R. E. Crandall. *Topics in Advanced Scientific Computation*. Springer, 1996.

[35] G. Dahlquist and Å. Björck. *Numerical Methods*. Prentice Hall, Englewood Cliffs, New Jersey, 1974.

[36] L. X. Dang. The nonadditive intermolecular potential for water revised. *J. Chem. Phys.*, 97:2659–2660, 1992.

[37] L. X. Dang. Computational study of ion binding of the liquid interface of water. *J. Phys. Chem.*, 106:10388–10394, 2002.

[38] L. X. Dang and T. Chang. Molecular dynamics study of water clusters, liquid and liquid–vapor interface of water with many-body potentials. *J. Chem. Phys.*, 106:8149–8159, 1997.

[39] T. A. Darden, D. M. York, and L. G. Pedersen. Particle mesh Ewald: An N·log(N) method for Ewald sums in large systems. *J. Chem. Phys.*, 98(10):10089, November 15 1993.

[40] S. W. de Leeuw, J. W. Perram, and E. R. Smith. Simulation of electrostatic systems in periodic boundary conditions. I. Lattice sums and dielectric constants. *Proc. R. Soc. Lond. A*, 373:27–56, 1980.

[41] P. Deuflhard and A. Hohmann. *Numerical Analysis in Modern Scientific Computing: An Introduction*. Springer, 2003.

[42] Y. Ding, D. N. Bernardo, K. Krogh-Jespersen, and R. M. Levy. Solvation free energies of small amides and amines from molecular dynamics/free energy perturbation simulations using pairwise additive and many–body polarizable potential. *J. Phys. Chem. B*, 99:11575–11583, 1995.

[43] R. D. Engle, R. D. Skeel, and M. Drees. Monitoring energy drift with shadow Hamiltonians. *J. Comput. Phys.*, 2005. In press.

[44] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. Pederson. A smooth particle mesh Ewald method. *J. Chem. Phys.*, 103:8577–8593, 1995.

[45] P. Ewald. Die βerechnung optischer und elektrostatischer δitterpotentiale. *Ann. Phys.*, 64:253–287, 1921.

[46] S. E. Feller, R. W. Pastor, A. Rojnuckarin, S. Bogusz, and B. R. Brooks. Effect of electrostatic force truncation on interfacial and transport properties of water. *J. Phys. Chem.*, 100:17011–17020, 1996.

[47] G. G. Ferenczy and C. A. Reyholds. Modeling polarization through induced atomic charges. *J. Phys. Chem.*, 105:11470–11479, 2001.

[48] D. Fincham. Optimization of the Ewald sum for large systems. *Molecular Simulation*, 13:1–9, 1994.

[49] D. Frenkel and B. Smit. *Understanding Molecular Simulation: From Algorithms to Applications*. Academic Press, second edition, 2002.

[50] J. Gao. Toward a molecular orbital derived empirical potential for liquid simulations. *J. Phys. Chem. B*, 101:657–663, 1997.

[51] J. Gao. A molecular-orbital derived polarization potential for liquid water. *J. Chem. Phys.*, 109:2346–2354, 1998.

[52] A. Glattli, X. Daura, and W. F. van Gunsteren. Derivation of an improved simple point charge model for liquid water: SPC/A and SPC/L. *J. Chem. Phys.*, 116:9811–9828, 2002.

[53] J. M. Googfellow. Cooperative effects in water-biomolecule crystal systems. *Proc. Natl. Acad. Sci. USA*, 79:4977–4979, 1982.

[54] L. Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems.* MIT Press, Cambridge, 1988.

[55] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73:325–348, 1987.

[56] N. Gresh, S. A. Kafafi, J. Truchon, and D. R. Salahub. Intramolecular interaction energies in model alanine and glycine tetrapeptides. evaluation of anisotropy, polarization and correlation effects. a parallel *ab initio* HF/MP2, DFT, and polarizable molecular mechanics study. *J. Comput. Chem.*, 25:823–834, 2003.

[57] H. Grubmüller, H. Heller, A. Windemuth, and K. Shulten. Generalized Verlet algorithm for efficient molecular dynamics simulations with long–range interactions. *Molecular Simulation*, 6:121, 1991.

[58] B. Guillot and Y. Guissani. How to build a better pair potential for water. *J. Chem. Phys.*, 114:6720–6733, 2001.

[59] H. Guo, N. Gresh, B. P. Roques, and D. R. Salahub. Many-body effects in systems of peptide hydrogen-bonded networks and their contributions to ligand bind: A comparison of the performance of DFT and polarizable molecular mechanics. *J. Phys. Chem. B*, 104:9746–9754, 2000.

[60] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration*, volume 31 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2002.

[61] T. A. Halgren. MMFF VII: characterization of MMFF94, MMFF94s, and other widely available force fields for conformational energies and for intermolecular-interaction energies and geometries. *J. Comput. Chem.*, 19:730–748, 1999.

[62] T. A. Halgren and W. Damm. Polarizable force fields. *Curr. Opinion Struct. Biol.*, 11(2):236–242, Apr. 2001.

[63] E. Harder, B. Kim, R. A. Friesner, and B. J. Berne. Efficient simulation method for polarizable protein force fields: application to the simulation of BPTI in liquid water. *Journal of Chemical Theory and Computation*, 1:169–180, 2005.

[64] S. C. Harvey, R. K. Z. Tan, and T. E. Cheatham III. The flying ice cube—velocity rescaling in molecular dynamics leads to violation of energy equipartition. *J. Comput. Chem.*, 19(7):726–740, May 1998.

[65] M. T. Heath. *Scientific Computing An Introductory Survey*. McGraw Hill, second edition, 2001.

[66] J. M. Hermida-Ramon, S. Brdarski, G. Karlström, and U. Berg. Inter- and intramolecular potential for the n-formylglycinamide-water system. A comparison between theoretical modeling and empirical force fields. *J. Comput. Chem.*, 24:161–176, 2002.

[67] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles*. McGraw-Hill, New York, 1981.

[68] W. G. Hoover. Canonical dynamics: Equilibrium phase-space distributions. *Phys. Rev. A*, 31:1695–1697, 1985.

[69] T. Ikeda, M. Hirata, and T. Kimura. Ab initio molecular dynamics study of polarization effects on ionic hydration in aqueous $AlCl_3$ solution. *J. Chem. Phys.*, 119:12386–12392, 2003.

[70] J. D. Jackson. *Classical Electrodynamics*. John Wiley & Sons, Inc., 2nd edition, 1975.

[71] J. Jeon, A. E. Lefohn, and G. A. Voth. An improved Polarflex water model. *J. Chem. Phys.*, 118:7504–7518, 2003.

[72] W. Jorgensen, D. Maxwell, and J. Tirado-Rives. Development of testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids. *J. Am. Chem. Soc.*, 118:11225–11236, 1996.

[73] P. Jungwirth, J. E. Curtis, and D. J. Tobias. Polarizability and aqueous solvation of the sulfate dianion. *Chem. Phys. Lett.*, 367:704–710, 2003.

[74] L. Kalé, R. Skeel, R. Brunner, M. Bhandarkar, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, and K. Schulten. NAMD2: Greater scalability for parallel molecular dynamics. *J. Comput. Phys*, 151(1):283–312, May 1, 1999.

[75] G. A. Kaminski, R. A. Friesner, and R. Zhou. A computationally inexpensive modification of the point dipole electrostatic polarization model for molecular simulations. *J. Comput. Chem.*, 24:267–276, 2003.

[76] G. A. Kaminski, H. A. Stern, B. J. Berne, and R. A. Friesner. Development of an accurate and robust polarizable molecular mechanics force field from ab initio quantum chemistry. *J. Phys. Chem. A*, 108:621–627, 2004.

[77] G. A. Kaminski, H. A. Stern, B. J. Berne, R. A. Friesner, Y. X. Cao, R. B. Murphy, R. Zhou, and T. A. Halgren. Development of a polarizable force field for proteins via ab initio quantum chemistry: First generation model and gas phase tests. *J. Comput. Chem.*, 23:1515–1531, 2002.

[78] M. A. Kastenholz and P. H. Hünenberger. Influence of artificial periodicity and ionic strength in molecular dynamics simulations of charged biomolecules employing lattice-sum methods. *J. Phys. Chem. B*, 108:774–788, 2004.

[79] J. Kolafa. Time–reversible always stable predictor–corrector method for molecular dynamics of polarizable molecules. *J. Comput. Chem.*, 25:335–342, 2003.

[80] M. Lísal, J. Kolafa, and I. Nezbeda. An examination of the five-site potential (TIP5P) for water. *J. Chem. Phys.*, 117:8892–8897, 2002.

[81] G. Lamoureux, , and B. Roux. Modeling induced polarization with classical Drude oscillators: Theory and molecular dynamics simulation algorithm. *J. Chem. Phys.*, 119:3025–3039, 2003.

[82] G. Lamoureux, A. D. MacKerell Jr., and B. Roux. A simple polarizable model of water based on classical Drude oscillators. *J. Chem. Phys.*, 119:5185–5197, 2003.

[83] P. Linz. *Theoretical Numerical Analysis: An Introduction to Advanced Techniques*. Dover, 2001.

[84] Y. Liu, K. Kim, B. J. Berne, R. A. Friesner, and S. W. Rick. Constructing *ab initio* force fields for molecular dynamics simulations. *J. Chem. Phys.*, 108:124739–4755, 1998.

[85] R. J. Loncharich and B. R. Brooks. The effects of truncating long-range forces on protein dynamics. *Proteins: Stru. Func. and Genetics*, 6:32–45, 1989.

[86] A. D. MacKerell Jr. Empirical force fields for biological macromolecules: Overview and issues. *J. Comput. Chem.*, 25:1584–1604, 2004.

[87] A. D. MacKerell Jr., D. Bashford, M. Bellott, R. L. Dunbrack Jr., J. Evanseck, M. J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph, L. Kuchnir, K. Kuczera, F. T. K. Lau, C. Mattos, S. Michnick, T. Ngo, D. T. Nguyen, B. Prodhom, I. W. E. Reiher, B. Roux, M. Schlenkrich, J. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiorkiewicz-Kuczera, D. Yin, and M. Karplus. All-hydrogen empirical potential for molecular modeling and dynamics studies of proteins using the CHARMM22 force field. *J. Phys. Chem. B*, 102:3586–3616, 1998.

[88] M. G. Martin, B. Chen, and J. I. Siemann. A novel Monte Carlo algorithm for polarizable force fields: application to a fluctuating charge model for water. *J. Chem. Phys.*, 108:3383–3385, 1998.

[89] G. Mathias, B. Egwolf, M. Nonella, and P. Tavan. A fast multipole method combined with a reaction field for long-range electrostatics in molecular dynamics simulations: the effect of truncation on the properties of water. *J. Chem. Phys.*, 118:10847–10860, 2003.

[90] M. Medeiros and M. E. Costas. Gibbs ensemble Monte Carlo simulation of the properties of water with a fluctuating charges model. *J. Chem. Phys.*, 107:2012–2019, 1997.

[91] S. Miyamoto and P. A. Kollman. SETTLE: An analytical version of the SHAKE and RATTLE algorithm for rigid water molecules. *J. Comput. Chem.*, 13(8):952–962, 1992.

[92] T. Morishita and S. Nosé. Momentum conservation law in the Car-Parrinello method. *Phys. Rev. B*, 59:15126–15132, 1999.

[93] A. Nakano. Parallel multilevel preconditioned conjugate-gradient approach to variable-charge molecular dynamics. *Computer Physics Comm.*, 104:59–69, 1997.

[94] M. Neumann. The dielectric constant of water. computer simulations with the MCY potential. *J. Chem. Phys.*, 82:5663–5672, 1985.

[95] S. Nosé. A unified formulation of the constant temperature molecular dynamics methods. *J. Chem. Phys.*, 81:511, 1984.

[96] T. N. Nymand and P. Linse. Ewald summation and reaction field methods for potentials with atomic charges, dipoles and polarizabilities. *J. Chem. Phys.*, 112:6152–6160, 2000.

[97] T. N. Nymand and P. Linse. Molecular dynamics simulations of polarizable water at different boundary conditions. *J. Chem. Phys.*, 112:6386–6395, 2000.

[98] G. Pastore, E. Smargiassi, and F. Buda. Theory of *ab initio* molecular-dynamics calculations. *Phys. Rev. A*, 44:6334–6347, 1991.

[99] S. Patel and C. L. Brooks III. CHARMM fluctuating charge force field for proteins: I Parameterization and application to bulk organic liquid simulations. *J. Comput. Chem.*, 25:1–15, 2003.

[100] S. Patel, A. D. MacKerell Jr., and C. L. Brooks III. CHARMM fluctuating charge force field for proteins: II Protein/solvent properties from molecular dynamics simulations using a nonadditive electrostatic model. *J. Comput. Chem.*, 25:1504–1514, 2004.

[101] M. C. Payne, M. P. Teter, D. C. Allan, T. A. Arias, and J. D. Joannopoulos. Iterative minimization techniques for *ab initio* total-energy calculations: molecular dynamics and conjugate gradients. *Rev. Mod. Phys.*, 64:1045–1097, 1992.

[102] M. Předota, P. T. Cummings, and A. A. Chialvo. Pair approximation for polarization interaction and adiabatic nuclear and electronic sampling method for fluids and dipole polarizability. *Mol. Phys.*, 100:2703–2717, 2002.

[103] D. A. Pearlman, D. A. Case, J. W. Caldwell, W. S. Ross, T. Cheatham, S. Debolt, D. Ferguson, G. Seibel, and P. Kollman. AMBER, a package of computer-programs for applying molecular mechanics, normal-mode analysis, molecular-dynamics and free-energy calculations to simulate the structural and energetic properties of molecules. *Comput. Phys. Commun.*, 91:1–41, 1995.

[104] A. S. Petrov, G. R. Pack, and G. Lamm. Calculations of magnesium-nucleic acid site binding in solution. *J. Phys. Chem. B*, 108:6072–6081, 2004.

[105] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, R. Skeel, and L. Kalé. Scalable molecular dynamics with NAMD. *J. Comput. Chem.*, 2005. submitted.

[106] J. Piquemal, B. William-Hubbard, N. Fey, R. J. Deeth, N. Gresh, and C. Grissner-Prettre. Inclusion of the ligand field contribution in a polarizable molecular mechanics: SIBFA-LF. *J. Comput. Chem.*, 24:1963–1970, 2003.

[107] E. L. Pollock and J. Glosli. Comments on P^3M, FMM, and the Ewald method for large periodic Coulombic systems. *Computer Phys. Commun.*, 95(2 and 3):93–110, June 6, 1996.

[108] J. W. Ponder and D. A. Case. Force fields for protein simulations. In F. M. Richards, D. S. Eisendberg, and J. Kuriyan, editors, *Advances in Protein Chemistry*, volume 66 Protein Simulations, pages 27–85. Elsevier Academic Press, 2003.

[109] P. Pulay. Direct inversion in the iterative subspace (DISS) optimization of open-shell, excited-state, and small multiconfiguration SCF wave functions. *Chem. Phys. Lett.*, 73:393–398, 1980.

[110] S. Reich. Backward error analysis for numerical integrators. *SIAM J. Numer. Anal.*, 36(5):1549–1570, 1999.

[111] D. K. Remler and P. A. Madden. Molecular dynamics without effective potentials via the Car-Parrinello approach. *Mol. Phys.*, 70:921–966, 1990.

[112] P. Ren and J. W. Ponder. Polarizable atomic multipole water model for molecular mechanics simulation. *J. Phys. Chem. B*, 107(24):5933–5947, 2003.

[113] S. W. Rick. A reoptimization of the five-site water potential (TIP5P) for use with Ewald sums. *J. Chem. Phys.*, 120:6085–6093, 2004.

[114] S. W. Rick and S. J. Stuart. Potentials and algorithms for incorporation polarizability in computer simulations. In K. B. Lipkowitz and D. B. Boyd, editors, *Reviews in Computational Chemistry*, volume 18, pages 89–146. Wiley-VCH, Berlin, 2002.

[115] J. E. Roberts and J. Schnitker. Boundary conditions in simulations of aqueous ionic solutions: A systematic study. *J. Phys. Chem.*, 99(4):1322–1331, 1995.

[116] J. A. C. Rullmann and R. T. van Duijnen. A polarizable water model for calculation of hydration energies. *Molecular Physics*, 63:451–475, 1988.

[117] G. Ruocco and M. Sampoli. Computer simulation of polarizable fluids: a consistent and fast way for dealing with polarizability and hyperpoliarzability. *Mol. Phys.*, 82:875–886, 1994.

[118] J. Ryckaert, G. Ciccotti, and H. Berendsen. Numerical integration of the cartesian equation of motion of a system with constraints: molecular dynamics of n-alkanes. *J. Comput. Phys.*, 23:327–341, 1977.

[119] Y. Saad. *Iterative methods for sparse linear systems*. PWS publishing company, 1st edition, 1996. Available on–line at http://www-users.cs.umn.edu/∼saad/books.html.

[120] 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. *J. Chem. Phys.*, 120:73–87, 2004.

[121] H. Saint-Martin, J. Hernández-Cobos, M. I. Bernal-Uruchurtu, I. Ortega-Blake, and H. J. C. Berendsen. A mobile charge densities in harmonic oscillators (MCDHO) molecular model for numerical simulations: The water–water interaction. *J. Chem. Phys.*, 113:10899–10912, 2000.

[122] H. Saint-Martin, B. Hess, and H. J. C. Berendsen. An application of flexible constraints in Monte Carlo simulations of the isobaric-isothermal ensemble of liquid water and ice Ih with the polarizable and flexible mobile charge densities in harmonic oscillators model. *J. Chem. Phys.*, 120:11133–11143, 2004.

[123] P. Salvador, J. E. Curtis, D. J. Tobias, and P. Jungwirth. Polarizability of the nitrate anion and its solvation at the air/water interface. *Phys. Chem. Chem. Phys.*, 5:3752–3757, 2003.

[124] W. Scott, P. Hünenberger, I. Tironi, A. Marks, T. Huber, P. Krüger, and W. van Gunsteren. The GROMOS biomolecular simulation program package. *J. Phys. Chem. A*, 103:3596–2607, 1999.

[125] K. Shimizu, H. Chaimovich, J. P. S. Farah, L. G. Dias, and D. L. Bostick. Calculation of the dipole moment for polypeptides using the generalized Born-electronegativity equalization method: results in vacuum and continuum-dielectric solvent. *J. Phys. Chem. B*, 108:4171–4177, 2004.

[126] R. D. Skeel, I. Tezcan, and D. J. Hardy. Multiple grid methods for classical molecular dynamics. *J. Comput. Chem.*, 23(6):673–684, Apr. 30, 2002.

[127] D. E. Smith and L. X. Dang. Computer simulations of NaCl association in polarizable water. *J. Chem. Phys.*, 100:3757, 1994.

[128] W. Smith. Point multipoles in the Ewald summation (revisited). *CCP5 Quarterly*, 26:43, 1987. a revised version is available from the author.

[129] D. Spangberg and K. Hermansson. Many-body potentials for aqueous Li^+, Na^+, Mg^{2+}, and Al^{3+}: comparison of effective three-body potentials and polarizable models. *J. Chem. Phys.*, 120:4829–4843, 2004.

[130] M. Sprik and M. L. Klein. A polarizable model for water using distributed charge sites. *J. Chem. Phys.*, 89:7556–7560, 1988.

[131] H. A. Stern, G. A. Kaminski, J. L. Banks, R. Zhou, B. J. Berne, and R. A. Friesner. Fluctuating charge, polarizable dipole, and combined models: Parameterization from ab initio quantum chemistry. *J. Phys. Chem. B*, 103:4730–4737, 1999.

[132] H. A. Stern, F. Rittner, B. J. Berne, and R. A. Friesner. Combined fluctuating charge and polarizable dipole models: Application to a five–site water potential function. *J. Chem. Phys.*, 115:2237–2251, 2001.

[133] Stillinger. Dynamics and ensemble averages for the polarization models of molecular interaction. *J. Chem. Phys.*, 71:1647–1651, 1979.

[134] A. J. Stone. *The Theory of Intermolecular Forces*. Oxford University Press, 1996.

[135] I. M. Svishchev, P. G. Kusalik, J. Wang, and R. J. Boyd. Polarizable point-charge model for water: Results under normal and extreme conditions. *Mol. Sim.*, 29:63–69, 2003.

[136] G. Tabacchi, C. J. Mundy, J. Hutter, and M. Parrinello. Classical polarizable force fields parametrized from ab initio calculations. *J. Chem. Phys.*, 117(4):1416–1433, July 2002.

[137] P. Tangney and S. Scandolo. A many-body interatomic potential for ionic systems: Application to MgO. *J. Chem. Phys.*, 119:9673–9685, 2003.

[138] B. T. Thole. Molecular polarizabilities calculated with a modified dipole interaction. *Chem. Phys.*, 59:341–350, 1981.

[139] G. Tiraboschi, B. Roques, and N. Gresh. Joint quantum chemical and polarizable molecular mechanics investigation of formate complexes with penta- and hexahydrated Zn^{2+}: Comparison between energetics of model bidentate, monodentate, and through-water Zn^{2+} binding modes and evaluation of nonadditivity effects. *J. Comput. Chem.*, 20:1379–1390, 1999.

[140] A. Toukmaji, C. Sagui, J. Board, and T. Darden. Efficient particle-mesh Ewald based approach to fixed and induced dipolar interactions. *J. Chem. Phys.*, 113(24):10913–10927, Dec. 22, 2000.

[141] L. N. Trefethen and D. Bau III. *Numerical Linear Algebra*. SIAM, 1997.

[142] M. Tuckerman, B. J. Berne, and G. J. Martyna. Reversible multiple time scale molecular dynamics. *J. Chem. Phys*, 97(3):1990–2001, 1992.

[143] P. J. van Maaren and D. van der Spoel. Molecular dynamics simulations of water with novel shell-model potentials. *J. Phys. Chem. B*, 105:2618–2626, 2001.

[144] W. Weber, P. H. Hünenberger, and J. A. McCammon. Molecular dynamics simulations of a polyalanine octapeptide under Ewald boundary conditions: Influence on artificial periodicity of peptide conformation. *J. Phys. Chem. B*, 104:3668–3675, 2000.

[145] S. J. Wierzchowski and D. A. Kofke. Fugacity coefficients of saturated water from molecular simulation. *J. Phys. Chem. B*, 107:12808–12813, 2003.

[146] H. Xu, H. A. Stern, and B. J. Berne. Can water polarizability be ignored in hydrogen bond kinetics? *J. Phys. Chem. B*, 106:2054–2060, 2002.

[147] S. Yoo, Y. A. Lei, and X. C. Zeng. Effect of polarizability of halide anions on the ionic solvation in water clusters. *J. Chem. Phys.*, 119:6083–6091, 2003.

[148] H. Yu, T. Hansson, and W. F. van Gunsteren. Development of a simple, self-consistent polarizable model for liquid water. *J. Chem. Phys.*, 118:221–234, 2003.

[149] H. Yu and W. F. van Gunsteren. Charge-on-spring polarizable water model revisited: from water clusters to liquid water to ice. *J. Chem. Phys.*, 121:9549–9564, 2004.

Author's Biography

Wei Wang was born on September 12, 1969 in Beijing, China. He attended Tsinghua University in 1988 and Peking University in 1993, majoring in physics. He joined the Department of Computer Science at the University of Illinois at Urbana-Champaign in 2000.