UNIVERSITY OF CALIFORNIA
RIVERSIDE

Efficient Polarization Solvers for Classical Molecular Dynamics Simulations

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Chemistry

by

Dominique V. Nocito

June 2019

Dissertation Committee:

Professor Gregory J. O. Beran, Chairperson
Professor Chia-en Angelina Chang
Professor Francisco Zaera

The Dissertation of Dominique V. Nocito is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

None of this would be possible without Greg. Greg provided invaluable guidance and encouragement that shaped me as a researcher. I am also grateful to all my lab mates, Dr. Yuanhang Huang, Dr. Yonaton Heit, Dr. Joshua Hartman, Jessica McKinley, Watit Sontising, Chandler Greenwell, Pablo Unzueta, Cameron Cook, and Elizabeth McMahon for their insightful discussions. I would like to thank my fiancée, Jessica McKinely, for her support and encouragement. I found my daily walks with Jessica through the botanical gardens to be the times when I was most relaxed and most creative.

To my mother Rhonda Nocito who fostered curiosity in countless trips to the science center, zoo, and city museum.

ABSTRACT OF THE DISSERTATION

Efficient Polarization Solvers for Classical Molecular Dynamics Simulations

by

Dominique V. Nocito

Doctor of Philosophy, Graduate Program in Chemistry
University of California, Riverside, June 2019
Professor Gregory J. O. Beran, Chairperson

The primary focus of this dissertation is the acceleration of the evaluation of the self-consistent polarization energy. Two new variants of Jacobi iterations are proposed here that exploit domain decomposition to accelerate the convergence of the induced dipoles. The first, divide-and-conquer JI (DC-JI), is a block Jacobi algorithm which solves the polarization equations within non-overlapping sub-clusters of atoms directly via Cholesky decomposition, and iterates to capture interactions between sub-clusters. The second, fuzzy DC-JI, achieves further acceleration by employing overlapping blocks. These algorithms employ knowledge of the 3-D spatial interactions to group important elements in the 2-D polarization matrix. These methods can be coupled with direct inversion in the iterative subspace (DIIS) extrapolation to accelerate their convergence.

The DC-JI solver is adapted for periodic boundary conditions with particle-mesh Ewald treatment of long-range interactions and implemented in a massively parallel fashion within the Tinker-HP software package. Compared to widely used preconditioned conjugate gradient (PCG) or conventional Jacobi iterations (JI/DIIS) algorithms, DC-JI/DIIS solves

the polarization equations ∼20–30% faster in protein systems ranging from ∼10,000–175,000 atoms run on hundreds of processor cores. Not only is DC-JI/DIIS faster than PCG, but it also gives more energetically robust solutions for a given convergence threshold.

We further demonstrate how one can improve the stability of a polarizable force field molecular dynamics simulation or accelerate the evaluation of self-consistent polarization via a simple extension of the predictor in the Always Stable Predictor-Corrector (ASPC) method. Specifically, increasing the number of prior steps used in the predictor from six to sixteen reduces the energy drift by an order of magnitude. Alternatively, for a given level of energy drift, the induced dipoles can be obtained ∼20% faster due to the reduced number of self-consistent field iterations required to maintain energetic stability.

Finally, we have developed an averaged condensed phase environment (ACPE) model that address the high computational cost associated with modeling configurational average properties with quantum mechanics/molecular mechanics (QM/MM) simulations. In the domain of embedding techniques ACPE lies in between explicit QM/MM evaluation of sampled configurations and continuum models. The ACPE model constructs an effective polarizable environment directly from explicitly sampled molecular dynamics configurations. ACPE can reduce the need for hundreds of QM/MM calculations to a few representative QM/MM calculations.

Contents

List of Figures **xi**

List of Tables **xiv**

1 Introduction **1**
1.1 Multipolar Interactions . . . 2
1.2 Polarizable Force Fields . . . 3
1.2.1 AMOEBA Force Field . . . 4
1.3 Self-Consistent Field Polarization Solvers . . . 5
1.4 SCF Initialization . . . 11
1.5 Embedding Models . . . 13

2 Divide-and-Conquer Jacobi Iterations I **14**
2.1 Introduction . . . 14
2.2 Theory . . . 18
2.2.1 Background . . . 18
2.2.2 Divide-and-conquer Jacobi Iteration Approach . . . 21
2.2.3 Fuzzy DC-JI . . . 24
2.2.4 Software Implementation . . . 26
2.3 Computational Methods . . . 35
2.4 Results and Discussion . . . 36
2.4.1 Behavior of the algorithms . . . 36
2.4.2 Performance in molecular dynamics . . . 45
2.5 Conclusions . . . 49

3 Divide-and-Conquer Jacobi Iterations II **50**
3.1 Introduction . . . 50
3.2 Theory . . . 53
3.2.1 Background . . . 53
3.2.2 Polarization Solvers . . . 55
3.2.3 Divide-and-conquer Jacobi Iteration method . . . 57
3.3 Software Implementation . . . 61

3.4 Computational Methods . . . 66
3.5 Results And Discussion . . . 68
3.5.1 Energy Convergence and Drift . . . 68
3.5.2 Computational Efficiency . . . 71
3.6 Conclusions . . . 75

4 The Always Stable Predictor Corrector **77**
4.1 Introduction . . . 77
4.2 Theory . . . 79
4.3 Computational Methods . . . 83
4.4 Results and Discussion . . . 83
4.5 Conclusions . . . 88

5 Average Condensed Phase Environment **90**
5.1 Introduction . . . 90
5.2 Theory . . . 94
5.2.1 Determination of coarse graining sites . . . 95
5.2.2 Force Field Parameter Interpolation . . . 100
5.2.3 Force Field Parameter Translation . . . 102
5.2.4 Clustering of Coarse-grain sites . . . 105
5.2.5 Molecular excitation energies with polarizable embedding . . . 108
5.3 Computational Methods . . . 109
5.4 Results and Discussion . . . 112
5.4.1 Determination of the ACPE model parameters . . . 112
5.4.2 Solvent structure in the ACPE model . . . 116
5.4.3 Excitation Energies in Aqueous Environment . . . 117
5.4.4 Solute at the benzene-water interface . . . 123
5.5 Conclusions . . . 125

6 Conclusions **128**

Bibliography **131**

A Average Condensed Phase Environment **139**
A.1 Multipolar and Polarizability Rotations . . . 139
A.2 Multipole Translations . . . 160
A.2.1 Rank 0: Charge translation . . . 162
A.2.2 Rank 1: Dipole translation . . . 162
A.2.3 Rank 2: Quadrupole translation . . . 164
A.2.4 Full List of Multipolar Translation Expressions up to Rank 4 . . . 167
A.3 Polarizability Translations . . . 171
A.4 Radial Distribution Functions for Pyrimidine in Water . . . 176
A.5 Convergence of the Excitation Energies with Configuration Sampling . . . 177
A.6 Sensitivity of ACPE Excitation Energies to the K-means Initial Guess . . . 178

B Always Stable Predictor Corrector **179**
B.1 Transferability Across Different Chemical Systems 179
B.2 Predictor Coefficients 180
B.2.1 Recursive Form for the Predictor Coefficients 181
B.2.2 The Predictor Coefficients for N=2–25-Step Predictors 181

List of Figures

2.1 Scheme showing how the DC-JI algorithm breaks up the full system of polarization equations for six polarizable sites into three smaller, coupled sets of sub-system equations. 22
2.2 K-means partitioning of a $(H_2O)_{365}$ droplet into ten sub-clusters. 23
2.3 Comparison of the $\tilde{\mathbf{Z}}$ matrix with (a) random (b) K-means sorting, and (c) Fuzzy K-means sorting of the polarizable sites for a $(H_2O)_{85}$ cluster (255 polarizable sites, 765 induced dipole vector components). The 10 colored sub-clusters in the molecular figure are indicated by the colored boxes in (b) and (c). The red elements in (c) correspond to terms captured in that particular fuzzy block. Matrix elements are plotted in log scale, with coloring for elements with magnitude 0.005 Å^{-3} or larger. 37
2.4 Average number of iterations required to converge the polarization to 10^{-6} Debeye RMS change of the dipoles in ubiquitin/water over 250 MD steps with various block sizes and polarization algorithms. Error bars/shaded regions indicate the range of iterations required in 90% of the time steps. PCG and JI/DIIS do not employ blocks, so their convergence rates are independent of block size. 39
2.5 Timing breakdown in the ubiquitin/water system for a single self-consistent polarization energy calculation using (a) DC-JI/DIIS (200 blocks) (b) Fuzzy DC-JI/DIIS (200 blocks) (c) JI/DIIS or (d) PCG, with the external interaction tensor elements $\mathbf{T}_{IJ}$ either evaluated on-the-fly at each iteration (lower memory requirement algorithm) or pre-computed once and stored in memory throughout. 43
2.6 Comparison of polarization CPU timings for (a)/(c) ubiquitin surrounded by 2835 waters (9,737 atoms) or (b)/(d) a cluster of 6,400 water molecules (19,200 atoms). Solid lines correspond to the "on-the-fly" algorithm variants, while dotted lines result from "pre-computing" and storing the $\mathbf{T}$ matrices. The labels indicate whether the induced dipoles were initialized from the previous iteration or using Kolafa's always stable predictor-corrector algorithm. Percentages indicate the time savings over JI/DIIS. 46
2.7 MD energy conservation in ubiquitin (no solvent) with four different polarization algorithms. Time steps of 0.25 fs were used. 48

3.1 Representation of the polarization equations in DC-JI with PME for a system of six atoms. DC-JI reorganizes the matrix polarization equations on the left to obtain the set of coupled equations on the right that are solved iteratively for each block. 60
3.2 Cartoon of the partitioning scheme employed in the parallel DC-JI algorithm. First, Tinker-HP partitions the entire system into a set of domains, with one domain per processor (large boxes at left). For DC-JI, each domain is further partitioned into a sub-clusters according to the coloring in the enlarged domain at right. In practice, the sub-clusters average 60 atoms each. 62
3.3 A 20 ps NVE simulation using a 1 fs time step and three different polarization convergence criteria for (a) Ubiquitin, (b) DHFR, and (c) COX-2. Note that PCG with a threshold of 10^{-4} D diverges out of frame within the first picosecond for all simulations. 68
3.4 (a) Absolute error in the converged polarization energy relative to the DC-JI/DIIS results with a 10^{-10} Debye convergence threshold, and (b) the number of iterations necessary to achieve that convergence (without using dipoles from previous time steps). For a given convergence threshold, DC-JI/DIIS achieves more converged energy than PCG in the same number of iterations or fewer. Values were averaged over 100 configuration snapshots. 69
3.5 Timing breakdowns for the COX-2 system (174,219 atoms) averaged over 100 steps. A block size of $\sim$60 atoms was used for DC-JI/DIIS. Timings were performed using 120 cores on SDSC Comet. DC-JI/DIIS and PCG converged in 4.0 iterations on average, while JI/DIIS required an average 5.3 iterations. 71
3.6 Performance of the DC-JI/DIIS, PCG, and JI/DIIS polarization solvers (left) and the corresponding total simulation time (right) for (a) Ubiquitin (b) DHFR, and (c) COX-2. The reciprocal of the polarization time is plotted for ease of comparison with the total simulation time. 74

4.1 The predictor coefficients vs coefficient history index. The 6-step predictor (default) is compared to the higher N-step predictors. 81
4.2 Comparison of the energy conservation for NVE simulations on $(H_2O)_{500}$ with different N-step predictors and three SCF iterations at each time step. 84
4.3 Energy drift rate (kcal mol^{-1} ns^{-1}) per atom for NVE simulations on $(H_2O)_{500}$ for the 6-step to 16-step predictor with differing numbers of SCF iterations at each time step. A tightly converged reference simulation with no predictor exhibits energy drift less than 10^{-5} kcal mol^{-1} ns^{-1} per atom. 85
4.4 Comparison of different higher N-step predictors with the DC-JI/DIIS and PCG polarization solvers. 87
4.5 Distributions of errors in the induced dipoles from the initial guess ($\boldsymbol{\mu}_0$) and first four SCF iterations relative to tightly converged dipoles using either the 6-step or 16-step predictor. 88

5.1 Flow-chart outlining the steps involved in the ACPE model. 96

5.2 Box-plot distributions of absolute percent errors in the (a) multipole moments and (b) polarizabilities between directly computed and the interpolated AIFF parameters for 400 water geometries. The boxes indicate the median error (center line) and the central 50% of the data, while the whiskers indicate the largest errors. 101
5.3 (a) Single configuration snapshot of water solvent. (b) ACPE environment from 400 sampled configurations. (c) A single ACPE molecular cluster near an acrolein solute. 104
5.4 Distribution of polarization cutoff distances for the aqueous environment around acrolein. 107
5.5 Hydrogen bonding distribution for molecular dynamics configurations and K-means++ generated grid points around pyrimidine. 116
5.6 Histograms of the first and second singlet excitation energies of acrolein compared with the experimental, configurational average, and ACPE values. Box heights indicate the number of configurations with this excitation energy. . 118
5.7 Histograms of the lowest singlet excitation energies of (a) acetone and (b) pyrimidine compared with the experimental, configurational average, and ACPE values. For pyrimidine, two different reported experimental excitation energies are shown. 121
5.8 Proportion of water and benzene molecules as a function of the z coordinate, averaged over 400 MD configurational snapshots, for phenol at the benzene/water interface. 123
5.9 Histogram of excitation energies for phenol lying at the benzene/water interface. 124

A.1 RDF of the pyrimidine N to H of water as computed with from the explicit MD simulation configurations or from the coarse-grained representation generated via K-Means. 176
A.2 Convergence of the configurational average for the vertical excitation energies with increasing number of configurations (a) acrolein, (b) acetone, (c) pyrimidine, (d) phenol. 177

B.1 Comparison of the energy drift (kcal mol^{-1} ns^{-1} per atom) with different N-steps predictors for a water box and ubiquitin. Each data point was obtained from a 1 ns NVE simulation using three iterations of the DC-JI/DIIS polarization solver. 180

List of Tables

2.1 Effect of the fuzzy membership function on the average number of atoms per block, average iterations to converge the polarization, and CPU time required per iteration to evaluate the induced dipoles in ubiquitin/water with DC-JI/DIIS or Fuzzy DC-JI/DIIS. 41

5.1 Predicted B3LYP/aug-cc-pVDZ ACPE excitation energies for acrolein with different scale factors for the number of heavy and light atom CG sites. Each ACPE calculation was repeated four times with different random initialization. Values report the average excitation energies and standard deviations. For reference, averaging over the 400 configurations explicitly predicts excitation energies of 3.84 eV ($n \rightarrow \pi^*$) and 5.86 eV ($\pi \rightarrow \pi^*$). 113

5.2 Timings for the construction of the ACPE from 400 acrolein in water configurations. CG refers to the time to generate the CG sites via K-means++. Config. Pol. indicates the time to compute the polarization in the individual configurations, and ACPE Pol. Cutoffs the time to identify appropriate polarization cutoffs automatically. 115

5.3 Comparison of CAM-B3LYP/aug-cc-pVTZ excitation energies E and solvatochromic shifts ΔE for three solutes in aqueous solution. The ACPE excitation energies are the average of three calculations. The individual values can be found in the SI. 120

5.4 Comparison of CAM-B3LYP/aug-cc-pVTZ excitation energies E and solvatochromic shifts ΔE for phenol at a benzene-water interface. 125

A.1 Variation in the ACPE excitation energies with different random initial guesses for the K-means algorithm. 178

Chapter 1

Introduction

Polarizable force fields mimic the way the quantum mechanical charge distribution responds to its environment. This dynamic behavior is crucial to modeling many properties correctly. For instance, the dipole moment of water can decrease ∼20% as it moves from bulk water to a non-polar protein pocket.[1] Inclusion of polarization can be necessary to capture the subtle balance of intra-protein and protein-environment interactions correctly[2]. This flexible description leads to polarizable force fields parameters having better transferability between chemical systems relative to fixed charge force fields.

The relatively high computational cost of polarizable force fields has prevented their widespread adoption for classical molecular dynamics simulations. In practice the polarization evaluation is too costly to be solved directly using standard linear algebra techniques and instead is calculated with an iterative self-consistent field (SCF) method.[3, 4] In this dissertation we focus on SCF methods that accelerate the polarization evaluation without loss of accuracy. Chapter 2 introduces a SCF method, divide-and-conquer Jacobi

iterations, which we show is superior to existing SCF methods. Chapter 3 extends the work from chapter 2 to large chemical systems via particle-mesh Ewald treatment of long-range interactions and a massively parallel implementation in Tinker-HP. In Chapter 4 we investigate improved energetic stability by use of a predictor for the SCF method with an "extended" history. Finally, Chapter 5 explores an efficient embedding algorithm for quantum mechanics/molecular mechanics calculations that reproduces the time-averaged behavior of the chemical system.

1.1 Multipolar Interactions

Intermolecular interactions such as electrostatics, induction, and dispersion stem from Coulombic interactions. In order to model these phenomena we must have a tractable means of describing the electronic charge distributions of our chemical system. One way to do this is to use a multipole expansion on the site of our molecules. While the sites can exist anywhere, normally atom-centered sites and occasionally bond centered sites are used. The first few terms of the multipolar expansion are listed with the symbols used to denote them: charges (q), dipoles (μ), quadrupoles (Θ), octupoles (Ω), and hexadecapoles (Φ). The rank at which the multipole moment expansion is truncated and where the sites are defined comes down to the user's desired balance of computational cost and needed spatial resolution of the electronic charge distribution. The multipole moments provide an excellent means to describe interactions of electronic interactions at mid to long range, however at short range when the electronic charge distributions have significant overlap the model begins to breakdown. As a point-localized representation, the multiple moments neglect

the diffuse nature of the electronic charge distribution, so at short range contributions from penetration energy are neglected. Advanced models have begun to address the breakdown of the multipolar expansion due to charge penetration.[5]

1.2 Polarizable Force Fields

Classical force fields (FF) refer to a parametric function relating the atomic coordinates to the potential energy of the chemical system. The parameters of the force field are fitted to high level computed data or experiment. Generally the total energy is composed of separable energy terms. These energy terms can be broken into two groups: bonded and non-bonded interactions. Much of the diversity of FFs encountered in the literature generally stems from the description of these non-bonded terms. A broad group of force fields called fixed-charge force fields generally only include van der Waals and electrostatic interactions. Polarizable force fields effectively incorporate the dynamic response of the electronic charge distributions into molecular dynamics simulations, but they do so at a significant increase in computational cost. One of the most robust models of polarization is the induced dipole model. However, the induced dipole model is also one of the more expensive force field descriptions of polarization, accounting for $\sim 50\%$ of the force field's computational cost. In this work all simulations will be performed with the AMOEBA (atomic multipole optimized energetics for bimolecular simulation) force field[6].

1.2.1 AMOEBA Force Field

Parameters for the AMOEBA force field have been worked out for a wide range of biomolecular species[7, 8, 9]. The functional form of AMOEBA is presented in eq. 1.1.

$$\begin{aligned}\mathbf{U}^{\mathbf{AMOEBA}} = \mathbf{U}^{\mathbf{bond}} + \mathbf{U}^{\mathbf{angle}} + \mathbf{U}^{\mathbf{bond-angle}} + \mathbf{U}^{\mathbf{out-of-plane}} \\ + \mathbf{U}^{\mathbf{torsional}} + \mathbf{U}^{\mathbf{vdW}} + \mathbf{U}^{\mathbf{electrostatic}} + \mathbf{U}^{\mathbf{polarization}}\end{aligned} \tag{1.1}$$

The first five terms correspond to the bonding contributions. The first four terms bond stretching, angle bending, bond-angle cross term and out-of-plane bending take a similar form to the terms found in the in the MM3 force field.[10] Thought it is typically classified as a bonding contribution, the torsion contribution in reality has some non-negligible dependence on non-bonded interactions. For this reason the torsional term is parameterized after the non-bonded terms in order to lead to a better physical balance. A buffered 14-7 vdW form is used providing a "softer" repulsive region. This buffered 14-7 vdW term is some times called the Halgren potential has been shown to better reproduce rare-gas potentials over a range of interatomic distances, relative to the Lennard-Jones form[11]. AMOEBA models the electronic charge distributions as an atomic centered multipole expansion up to the quadrupole moment. AMOEBA models the mutual polarization using the induced dipole model. Polarization is the response of the electronic charge distribution to an external electric field. In the induced dipole model each site is assigned a polarizability which models the flexibility of the electronic distribution around that site. The induced dipole on a site is equal to the polarizability of that sites times the external electric field on the site.

$$\boldsymbol{\mu}_i = \alpha_i \mathbf{V}_i + \sum_i \mathbf{T}_{ij} \boldsymbol{\mu}_j \tag{1.2}$$

Where $\boldsymbol{\mu}_i$ is the induced dipole on site i, α_i is the polarizability on site i, $\mathbf{V}_i$ is the electric field from the permanent multipole moments and the last term is the electric field from all other induced dipoles. $\mathbf{T}_{ij}$ is the interaction matrix which captures the distance and orientational dependence of the dipole-dipole interaction between atomic sites a and b. The mutual polarization must be solved self-consistently. Common polarization solvers will be discussed in the next section.

1.3 Self-Consistent Field Polarization Solvers

The induced dipoles are the solution to system of linear equations listed below,

$$\mathbf{Z}\boldsymbol{\mu} = \mathbf{V} \tag{1.3}$$

where $\mathbf{Z}$ is the symmetric response matrix with blocks for each atom a, b, c, etc:

$$\mathbf{Z} = \begin{bmatrix} (\boldsymbol{\alpha}_a)^{-1} & -\mathbf{T}_{ij} & -\mathbf{T}_{ik} & \dots \\ -\mathbf{T}_{ji} & (\boldsymbol{\alpha}_b)^{-1} & -\mathbf{T}_{jk} & \dots \\ -\mathbf{T}_{ki} & -\mathbf{T}_{kj} & (\boldsymbol{\alpha}_c)^{-1} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{1.4}$$

Here, $\boldsymbol{\alpha}_a$ is a 3×3 diagonal matrix with the inverse of the isotropic polarizability along the diagonal, and $\mathbf{T}_{ij}$ is the interaction matrix described above. In practice the system

of linear equations is too large to be solved directly using techniques such as Cholesky factorization. Instead, self-consistent field (SCF) solvers are used to solve for the induced dipoles iteratively.

Jacobi iterations (JI) offers a simple iterative procedure for solving for the induced dipoles. It corresponds to the intuitive picture of mutual polarization where site A polarizes site B, site B polarizes A, and the process iterates until self-consistency is achieved for the induced dipoles. In JI, one partitions $\mathbf{Z}$ into its diagonal $\mathbf{D}$ and off-diagonal elements $\mathbf{Y}$, $\mathbf{Z} = \mathbf{D} + \mathbf{Y}$. After minor rearrangement one finds,

$$\mathbf{D}\boldsymbol{\mu} = \mathbf{V} - \mathbf{Y}\boldsymbol{\mu} \tag{1.5}$$

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

$$\boldsymbol{\mu} = \mathbf{D}^{-1}\left(\mathbf{V} - \mathbf{Y}\boldsymbol{\mu}\right) = \boldsymbol{\alpha}\left(\mathbf{V} + \mathbf{T}\boldsymbol{\mu}\right) \tag{1.6}$$

However, since the right-hand side depends on $\boldsymbol{\mu}$, this equation must be solved iteratively.

The convergence behavior of JI can be determined using eigenvalue analysis. We can expand the iterative induced dipoles as the true induced dipole plus some error vector for that iteration.

$$\boldsymbol{\mu} + \mathbf{e^{i+1}} = \mathbf{D^{-1}V} - \mathbf{D^{-1}Y}\boldsymbol{\mu} - \mathbf{D^{-1}Ye_i} \tag{1.7}$$

Where $\mathbf{e_i}$ is the error vector at iteration i and $\boldsymbol{\mu}$ is the true induced dipole vector. The first two term on the right hand side of eq. 1.7 equate to $\boldsymbol{\mu}$ since in the limit of convergence

eq. 1.6 is equal to the true induced dipoles. The error for each successive iteration is then given by.

$$\mathbf{e_{i+1}} = -\mathbf{D^{-1}Ye_i} \tag{1.8}$$

Where $\boldsymbol{\mu}$ is the true induced dipole and $\mathbf{e_i}$ is the error vector for the induced dipole at iteration i. We can expand $\mathbf{e_i}$ in the basis of the eigenvectors of $-\mathbf{D}^{-1}\mathbf{Y}$ to get.

$$\sum_{j}^{3n} \lambda_j c_j \mathbf{v}_j = -\mathbf{D}^{-1}\mathbf{Y} \sum_{j}^{3n} c_j \mathbf{v}_j \tag{1.9}$$

Where $\mathbf{v}_j$ is the jth eigenvector, c_j is the linear combination coefficient for the jth eigenvector and λ_j is the corresponding eigenvalue. We see that if λ_j is less than unity then that component of the error vector decreases with each additional iteration, but if it is close to unity the error component is reduced slowly and if it is greater than unity that component of the error vector diverges. JI is not guaranteed to converge, and in practice, if it does converge, it usually does so slowly. We can address this shortcoming using a method called successive over relaxation (SOR). In SOR we update the induced dipoles each iteration as the previous induced dipole plus some scaled change in the induced dipole from this iteration.

$$\boldsymbol{\mu}^{n+1} = \boldsymbol{\mu}^n + \omega(\overline{\boldsymbol{\mu}}^{n+1} - \boldsymbol{\mu}^n) \tag{1.10}$$

Where $\overline{\boldsymbol{\mu}}^{n+1}$ is the induced dipole from the desired iterative method, in this context it would be the JI method. If ω is unity we have recovered the JI method. If ω is less than

unity we are damping the change in the induced dipole, and if ω is greater than unity we are accelerating the change in the induced dipole. Since JI has troubles converging for the systems of interest, a ω less than unity is used. A typical optimal value for ω in the context of the polarization problem is ~ 0.78. This dampening factor mutes the the components of the induced dipole error vector that would be diverging, but unfortunately this also mutes the desirable converging components. For this reason SOR is quickly abandoned for better performing methods.

One of these methods is Jacobi Iterations coupled with the Direct Inversion of the Iterative Subspace (DIIS) method. That is, after establishing a short history of the induced dipoles during the first few iterations, the induced dipoles are extrapolated via DIIS after each Jacobi iteration. DIIS extrapolates the induced dipoles $\boldsymbol{\mu}^{extrap}$ as a linear combination of $\boldsymbol{\mu}^{(j)}$ from previous iterations j.

$$\boldsymbol{\mu}^{extrap} = \sum_{j}^{n} c_j \boldsymbol{\mu}_j \tag{1.11}$$

The extrapolation coefficients c_j are obtained by solving,

$$\begin{bmatrix} B_{11} & B_{12} & \cdots & B_{1n} & -1 \\ B_{21} & B_{22} & \cdots & B_{2n} & -1 \\ \vdots & \vdots & \ddots & \vdots & -1 \\ B_{n1} & B_{n2} & \cdots & B_{nn} & -1 \\ -1 & -1 & -1 & -1 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -1 \end{bmatrix} \tag{1.12}$$

where B_{ij} is the inner product between two residual vectors that represent the change in our

vector of interest between iterations, $B_{ij} = \left(\Delta\boldsymbol{\mu}^{(i)}\right)^T \Delta\boldsymbol{\mu}^{(j)}$ where $\Delta\boldsymbol{\mu}^{(j)} = \boldsymbol{\mu}^{(j)} - \boldsymbol{\mu}^{(j-1)}$.

A brief overview of the Divide-and-Conquer JI (DC-JI) method is presented here a more in depth discussion can be found in chapters 3 and 4. DC-JI is a block Jacobi-like method where mutual polarization in spatial proximal clusters of atoms is solved directly and the contributions between clusters is captured iteratively in a JI-like manner. Another possible naming scheme for these methods might be Cluster-and-Conquer JI to stress that the grouping of the clusters is based on some relevant physical property. Regardless, we will use the the original name of DC-JI. DC-JI partitions $\mathbf{Z}$ into a block diagonal matrix $\mathbf{D}$ and a matrix of the remaining off-diagonal elements $\mathbf{Y}$, $\mathbf{Z} = \mathbf{D} + \mathbf{Y}$. The partitioning of the matrix does not affect the simplification, so we see DC-JI has an identical form to JI.

$$\boldsymbol{\mu} = \mathbf{D}^{-1}\left(\mathbf{V} - \mathbf{Y}\boldsymbol{\mu}\right) = \boldsymbol{\alpha}\left(\mathbf{V} + \mathbf{T}\boldsymbol{\mu}\right) \tag{1.13}$$

However, since $\mathbf{D}$ is no longer a diagonal matrix, we avoid the higher cost of inversion by solving for the induced dipoles of the blocks using Cholesky decomposition. We can apply the same eigenvalue analysis that we applied to JI to look at the convergence properties of DC-JI for a small cluster of 80 water molecules. For this system the spectral radius of $-\mathbf{D}^{-1}\mathbf{Y}$ for JI is 1.0298 and for DC-JI it is 0.5655. We see that for this system, JI alone will diverge very slowly, however DC-JI will converge reasonably fast.

Alternatively, the conjugate gradient (CG) method solves for the polarization by minimizing E_{pol} in Eq 1.14.

$$E_{pol} = \frac{1}{2}\boldsymbol{\mu}^T\mathbf{Z}\boldsymbol{\mu} - \boldsymbol{\mu}^T\mathbf{V} \tag{1.14}$$

Conjugate gradients is referred to as a Krylov subspace method, referring to the growing subspace $\mathbf{D_j}$.

$$\mathbf{D_j} = \mathbf{span}\{\mathbf{r_0}, \mathbf{Zr_0}, \mathbf{Z^2 r_0}, \mathbf{Z^{j-1} r_0}\} \tag{1.15}$$

Where $\mathbf{D_j}$ is the subspace defined at iteration j, $\mathbf{r}_0$ is the residual and with each additional iteration the matrix $\mathbf{Z}$ is applies to the residual growing the subspace by the residual $\mathbf{r}_{j+1}$ which is orthogonal to $\mathbf{D_j}$. The $\boldsymbol{\mu}_j$ can be viewed as the projection of $\boldsymbol{\mu}$ onto the subspace $\mathbf{D_j}$. CG does not search along $\mathbf{r}_j$; instead the search direction $\mathbf{d}_j$ is constructed by the conjugation of the residuals. This has the benefit that the residual is orthogonal to the previous search directions, which means we will not have to store all previous search directions.

The $\boldsymbol{\mu}_{j+1}$ are given as linear combination of the past search directions.

$$\boldsymbol{\mu}_{j+1} = \boldsymbol{\mu}_j + \alpha_j \mathbf{d}_j \tag{1.16}$$

Where $\boldsymbol{\mu}_j$ is the dipoles at iteration j, $\mathbf{d}_j$ is the search direction at iteration j, and α_j is the step size given by

$$\alpha_j = \frac{\mathbf{r}_j^T \mathbf{r}_j}{\mathbf{d}_j^T \mathbf{Z} \mathbf{d}_j} \tag{1.17}$$

The $\mathbf{r}_{j+1}$ are given by,

$$\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{Z} \mathbf{d}_j \tag{1.18}$$

The search direction $\mathbf{d}_{j+1}$ is given by,

$$\mathbf{d}_{j+1} = \mathbf{r}_{j+1} + \beta_j \mathbf{d}_j \tag{1.19}$$

Where $\mathbf{d}_j$ is the search direction at iteration j, $\mathbf{r}_{j+1}$ is the residual at iteration j+1, and β_j is the Gram-Schmidt constant given by

$$\beta_j = \frac{\mathbf{r}_{j+1}^T \mathbf{r}_{j+1}}{\mathbf{d}_j^T \mathbf{Z} \mathbf{d}_j} \tag{1.20}$$

Poor convergence behavior is observed when the condition number of $\mathbf{Z}$ is large, but this can be improved by applying a preconditioner (PCG) to solve a modified system of equations:

$$\mathbf{P}^{-1}\mathbf{Z}\boldsymbol{\mu} = \mathbf{P}^{-1}\mathbf{V} \tag{1.21}$$

where $\mathbf{P}^{-1}$ is some easily computed matrix that approximates $\mathbf{Z}^{-1}$. Preconditioning reduces the range of the eigenvalues for the polarization matrix $\mathbf{Z}$, speeding convergence. Sophisticated preconditioners exist that evaluate only the short range interactions, but for massively parallel implementations the diagonal preconditioner offers a satisfactory improvement in convergence without the need for additional communication between processes.

1.4 SCF Initialization

The number of iterations an SCF method takes to reach convergence is dependent on two factors: the starting point for the method and the convergence properties of the solver that is used. There are several commonly used starting points. These include the "direct guess" where the induced dipoles are set as the atomic polarizabilities times the permanent

electric field. This guess arises from initially assuming the induced dipoles on all other atoms are zero. Alternatively, one can use the "previous guess", where the induced dipoles from the last MD step are used. Because MD time steps are typically relatively short (a few fs or less), the induced dipoles from the previous time step provide a reasonable starting point for the next iteration. Finally, one can employ the "predictor guess" which in this context is the induced dipoles from Kolafa's Always Stable Predictor-Corrector (ASPC) algorithm[12, 13]. The predictor guess differs from the previous guess in that it relies on a longer history of induced dipoles from earlier MD steps to predict a guess for the dipoles at the next step. It has been shown that the predictor guess provides the optimal reduction in the number of SCF iterations needed for a stable MD simulation[4]. Other alternative methods such as inertial extended Lagrangian approach have also been proposed.[14]

The ASPC uses a history-based predictor for the induced dipoles,

$$\boldsymbol{\mu}^p(t+1) = \sum_{j=0}^{k+1} B_{j+1}\boldsymbol{\mu}(t-jh) \tag{1.22}$$

where $\boldsymbol{\mu}^p(t+1)$ is the predicted dipole, B_{j+1} are the scaling coefficients and $\boldsymbol{\mu}(t-jh)$ are the induced dipoles from previous time steps. The time step size is h and $k+2$ is the total number of values stored in history. The B_{j+1} scaling coefficients are derived such that the contributions that lead to time irreversibility error are chosen to be zero. It is known that the use of previous information from a simulation destroys the time reversibility of the method.[15] A compromise between an acceptable degree of time reversibility and improvement in the SCF starting point must be made when using the predictor guess.

1.5 Embedding Models

Descriptions of extended chemical systems can be computationally expensive. One way to address this expense is with the use of hybrid quantum mechanics/molecular mechanics models. In these approaches the area of interest is described with expensive QM method while the extended system is modeled with the MM method. QM/MM is a broad term used to describe a family of methods, but the type of method we will be focusing on is the polarizable embedding models. In polarizable embedding models the mutual polarization is accounted for between the QM and MM system. This embedding method can be slightly more expensive than the pure QM evaluation, but the physics that is captured allows for a better description of chemical properties. For instance the evaluation of excitation energies of solvated molecules is heavily dependent on the interactions between solvent and solute. When the presence of the solvent shifts the excitation energy of our molecule relative to the gas phase this is called solvatochromic shift. In order to properly model the solvatochromic shifts of our system we must capture the electronic interaction between the solute and solvent to the best of our ability. We can define the QM region of our model as the solute alone or the solute with a few of its nearest neighbor solvent molecules. With more solute molecules included in the QM region the better we expect or results to be, however this also increases the computational cost. The remaining solvent system can be modeled at MM level where the sites of the solvent are assigned a multipole expansion to model the electronic distribution around the solvent molecules and a polarizability on the sites that allows the MM solvents to respond to the electronic interactions with the QM region.

Chapter 2

Divide-and-Conquer Jacobi Iterations I

2.1 Introduction

Much research effort is currently being devoted toward the developments of physically motivated force fields which are more transferable between a variety of systems and environments. Polarization typically plays an important role in allowing a given set of force field parameters to describe widely different electrostatic environments. In protein folding, for example, hydrophobic amino acids transition from a polar aqueous environment to the non-polar protein interior, and polarizable force fields can mimic how the quantum mechanical electron densities vary across these different environments.

On the other hand, evaluating the many-body polarization energy can increase the overall computational cost by an order of magnitude compared to simpler pairwise-additive

potentials. Considerable effort has been expended in recent years to make molecular dynamics simulations with polarizable force fields more computationally practical,[16] including accelerating the evaluation of the polarization energy, improving parallel efficiency for evaluating the polarization energy on multiple processors (including both conventional processors and graphical processing units),[4, 17, 18, 19] and developing strategies for taking longer time steps/accelerated molecular dynamics.[20, 21].

Evaluating the self-consistent polarization energy for a force field like AMOEBA (atomic multipole optimized energetics for biomolecular applications)[8, 6, 7, 9] requires solving a set of linear equations of dimension three times the number of polarizable sites to obtain the induced dipoles. Although these equations can be solved via direct (non-iterative) linear algebra techniques in small systems, iterative techniques are required in larger systems. Traditionally, techniques like Jacobi iterations (JI), accelerated with over-relaxation (JOR)[7] or direct inversion of the iterative subspace (DIIS)[22, 23, 24, 4] are used. More recently, the preconditioned conjugate gradient (PCG) algorithm has also been employed.[3, 4, 17]

While these iterative methods vary in their rate of convergence and overall computational cost, they solve the polarization equations to within a user-specified tolerance. Several other techniques introduce approximations to the polarization equations in order to accelerate the calculation of the polarization energy. Examples include perturbation theory approaches that correct for couplings between induced dipoles,[25, 26] extended Lagrangian dynamics for the induced dipole vectors,[14] truncated many-body expansions for the polarization energy (3-AMOEBA),[27], united-atom models that reduce the number

of polarizable sites (uAMOEBA),[28] and re-parameterized versions of the force field that compensate for omitting self-consistency (iAMOEBA).[29] Very recently, a systematically improvable hierarchy of truncated PCG models has been proposed that gives analytic gradients and allows a user-defined balance between cost and accuracy.[30]

Physically, the strongest polarization effects occur locally between adjacent molecules, but they are influenced by longer-range and many-body effects. We propose two different divide-and-conquer (DC) models for solving the polarization equations that exploit this feature. The first model, DC-JI, is equivalent to the non-overlapping domain decomposition technique known as Block Jacobi iteration.[31] DC-JI partitions the polarization problem into local clusters of polarizable sites (blocks). The self-consistent polarization equations are solved directly (non-iteratively) within each block, while the couplings between blocks are incorporated iteratively.

While the DC-JI approach solves the polarization problem within each block efficiently, the iterative inclusion of the interactions between blocks in Block Jacobi is slow.[31] The second approach proposed here, Fuzzy DC-JI, employs overlapping domain decomposition. The "fuzziness" stems from the fact that a given polarizable site can simultaneously be part of many different blocks. The same site will have different induced dipoles in each block in which it is a member. Borrowing the terminology of fuzzy logic, these multiple fuzzy values of the induced dipoles must then be "defuzzified" into a single set of "crisp" dipoles. Here, the crisp dipoles are computed as distance-weighted averages of the induced dipoles from each different block. In principle, both DC-JI and fuzzy DC-JI converge to the numerically exact solution of the polarization equations. In practice, they are converged to

within a user-defined tolerance.

Domain decomposition is well established in solving systems of linear equations, of course. The primary challenge lies in identifying effective domains. The most important couplings between atoms depend on the three-dimensional (3-D) spatial arrangement of those atoms, but the polarization equations project the 3-D physical interactions onto a 2-D matrix. The challenge is magnified when overlapping domains are used—how should one choose additional off-diagonal elements to incorporate into the overlapping blocks? What weights should be used to perform the defuzzification?

A key feature in the current work is the use of the K-means clustering algorithm to identify natural sub-clusters of atoms in the system automatically. The K-means sorting concentrates the largest matrix elements along the block diagonal. In the fuzzy algorithm variant, K-means identifies which atoms lie on near the edges of the sub-clusters and should therefore be distributed across multiple sub-clusters. It also provides a mechanism for defining the necessary defuzzification weights.

In the end, both the DC-JI/DIIS and fuzzy DC-JI/DIIS algorithms require fewer iterations to converge and are substantially faster than JI/DIIS. Moreover, fuzzy DC-JI/DIIS converges just as rapidly as PCG, with appreciably lower computation cost per iteration. In the serial implementation of the algorithms developed here, the AMOEBA polarization time is up 2–3 faster than JI/DIIS and PCG (using the Tinker 7.1[32] implementation of the latter). Finally, while this paper focuses on AMOEBA, the algorithms extend readily to other polarizable force fields involving discrete polarizable sites and comparable polarization equations.

2.2 Theory

2.2.1 Background

The AMOEBA model assigns a set of multipoles (charges, dipoles, and quadrupoles) and a scalar polarizability to each atom. Dipoles are induced on each site due to the mutual interactions of permanent and induced multipoles between sites. The polarization energy is computed from these induced dipoles according to,

$$E_{pol} = \frac{1}{2}\boldsymbol{\mu}^T \mathbf{T}\mathbf{m} = \frac{1}{2}\boldsymbol{\mu}^T \mathbf{V} \tag{2.1}$$

where $\boldsymbol{\mu}$ is a vector of induced dipoles, $\mathbf{m}$ is a vector of the permanent multipoles, $\mathbf{T}$ is the interaction tensor, and $\mathbf{V} = \mathbf{T}\mathbf{m}$ is the permanent field. The induced dipoles $\boldsymbol{\mu}$ are obtained by solving the linear equations,

$$\tilde{\mathbf{Z}}\boldsymbol{\mu} = \mathbf{V} \tag{2.2}$$

where $\tilde{\mathbf{Z}}$ is a symmetric matrix with blocks for each atom a, b, c, etc:

$$\tilde{\mathbf{Z}} = \begin{bmatrix} (\boldsymbol{\alpha}_a)^{-1} & -\mathbf{T}_{ab} & -\mathbf{T}_{ac} & \dots \\ -\mathbf{T}_{ba} & (\boldsymbol{\alpha}_b)^{-1} & -\mathbf{T}_{bc} & \dots \\ -\mathbf{T}_{ca} & -\mathbf{T}_{cb} & (\boldsymbol{\alpha}_c)^{-1} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{2.3}$$

Here, $\boldsymbol{\alpha}_a$ is a 3×3 diagonal matrix with the inverse of the isotropic polarizability along the diagonal, and $\mathbf{T}_{ab}$ is a symmetric 3×3 matrix with six unique elements corresponding to the various components of the dipole-dipole interaction between atomic sites a and b. For small systems, this set of linear equations can be solved directly using techniques such as Cholesky factorization. For larger systems, however, iterative solution becomes necessary.

In the Jacobi iterations (JI) approach, one partitions $\tilde{\mathbf{Z}}$ into its diagonal $\mathbf{D}$ and off-diagonal elements $\mathbf{Y}$, $\tilde{\mathbf{Z}} = \mathbf{D} + \mathbf{Y}$. After minor rearrangement one finds,

$$\mathbf{D}\boldsymbol{\mu} = \mathbf{V} - \mathbf{Y}\boldsymbol{\mu} \tag{2.4}$$

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

$$\boldsymbol{\mu} = \mathbf{D}^{-1}\left(\mathbf{V} - \mathbf{Y}\boldsymbol{\mu}\right) = \boldsymbol{\alpha}\left(\mathbf{V} + \mathbf{T}\boldsymbol{\mu}\right) \tag{2.5}$$

However, since the right-hand side depends on $\boldsymbol{\mu}$, this equation must be solved iteratively, starting from an initial guess $\boldsymbol{\mu} = \boldsymbol{\alpha}\mathbf{V}$.

The notoriously slow convergence of JI can be improved using Jacobi over-relaxation or DIIS extrapolation.[4] DIIS extrapolation in particular can reduce the number of iterations required to achieve convergence several-fold and even sometimes converges in cases where JI alone does not.

Another widely used approach to solving the polarization equations relies on pre-

conditioned conjugate gradients. The quadratic form,

$$f(\boldsymbol{\mu}) = \frac{1}{2}\boldsymbol{\mu}^T\tilde{\mathbf{Z}}\boldsymbol{\mu} - \mathbf{V}^T\boldsymbol{\mu} \tag{2.6}$$

has a minimum when $\tilde{\mathbf{Z}}\boldsymbol{\mu} = \mathbf{V}$. Conjugate gradients can be viewed as a minimization algorithm that takes a series of steps in conjugate directions (determined from the residuals from previous iterations k, $\boldsymbol{\delta}^{(k)} = \tilde{\mathbf{Z}}\boldsymbol{\mu}^{(k)} - \mathbf{V}$) to minimize $f(\boldsymbol{\mu})$. Each CG step provides monotonic improvement toward the exact solution. The rate of convergence depends significantly on the condition number of $\tilde{\mathbf{Z}}$, and preconditioning can be very important. Preconditioning involves solving the modified linear equations,

$$\mathbf{P}^{-1}\tilde{\mathbf{Z}}\boldsymbol{\mu} = \mathbf{P}^{-1}\mathbf{V} \tag{2.7}$$

where $\mathbf{P}^{-1}$ is some easily computed matrix that approximates $\tilde{\mathbf{Z}}^{-1}$. Preconditioning reduces the range of the eigenvalues for the polarization matrix $\tilde{\mathbf{Z}}$, speeding convergence. We use the PCG implementation found in Tinker 7.1[32]. This preconditioner approximates $\tilde{\mathbf{Z}}$ from the first terms in its power series expansion,[3, 30]

$$\tilde{\mathbf{Z}}^{-1} = \boldsymbol{\alpha}(\mathbf{I} - \boldsymbol{\alpha}\mathbf{T})^{-1} \approx \boldsymbol{\alpha}(\mathbf{I} + \boldsymbol{\alpha}\mathbf{T}) = \mathbf{P}^{-1} \tag{2.8}$$

where $\mathbf{I}$ is the identity matrix. For computational efficiency, this preconditioner is evaluated only for short-range interactions (e.g. to within 3 Å). While the cost per PCG iteration is higher than that of JI, it converges more robustly and in fewer iterations. Note too that a hierarchy of approximate and efficient PCG methods have also been proposed recently.[30]

2.2.2 Divide-and-conquer Jacobi Iteration Approach

Here, we propose a divide-and-conquer Jacobi Iteration (DC-JI) scheme. Physically, the strongest mutual polarization effects will occur among sub-clusters of spatially proximal atoms/polarizable sites. DC-JI solves the mutual polarization problem within each cluster "exactly" (subject to the limits of numerical precision) using non-iterative Cholesky factorization. The effects of mutual polarization between the different sub-clusters are then incorporated iteratively. In the terminology of numerical linear algebra, this divide-and-conquer approach amounts to a block Jacobi algorithm or a non-overlapping domain decomposition. DIIS extrapolation is used to accelerate the convergence of the block Jacobi iterations, just as in conventional JI.

Like the JI approach, DC-JI partitions $\tilde{\mathbf{Z}} = \mathbf{Z} + \mathbf{Y}$, but in this case $\mathbf{Z}$ is block diagonal matrix, rather than a diagonal one. Blocks of $\mathbf{Z}$ include both the inverse of the polarizability tensors $\boldsymbol{\alpha}$ and the negative of the dipole-dipole interaction tensors $\mathbf{T}$ between the atoms in the block. Exploiting the block structure of $\mathbf{Z}$, one can break Eq 3.7 into a separate matrix equation for each sub-cluster of polarizable sites,

$$\mathbf{Z}_{II}\boldsymbol{\mu}_I = \mathbf{V}_{0,I} - \mathbf{Y}_{IJ}\boldsymbol{\mu}_J = \mathbf{V}_{0,I} + \sum_J \mathbf{T}_{IJ}\boldsymbol{\mu}_J \tag{2.9}$$

where I and J here refer to the different blocks of atoms, rather than individual matrix elements. In other words, the induced dipoles on atoms in block I depend on the total field, which includes contributions from both the static field for block I and interactions with the induced moments from atoms in other blocks J (induced field). This is shown schematically in Figure 2.1. With appropriately small block sizes, one can solve each block equation for

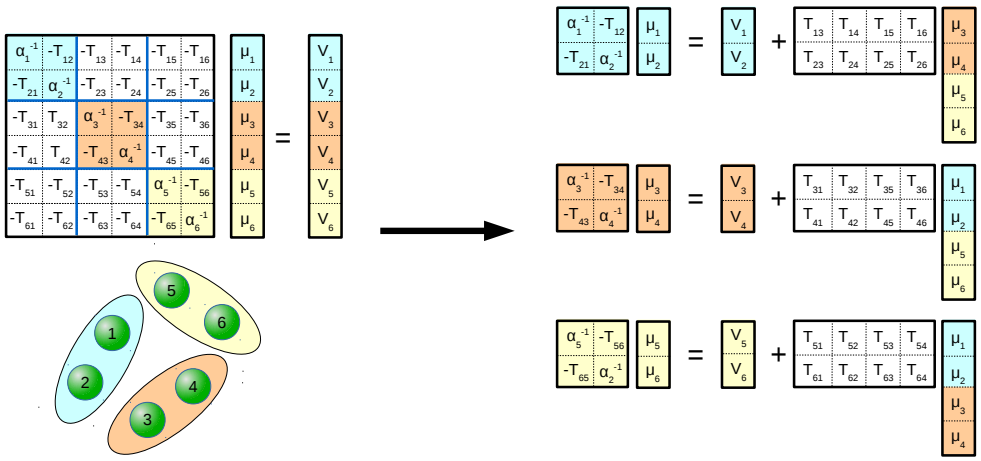


Figure 2.1: Scheme showing how the DC-JI algorithm breaks up the full system of polarization equations for six polarizable sites into three smaller, coupled sets of sub-system equations.

$\boldsymbol{\mu}_I$ directly. However, because the induced dipoles in block I depend on those in all other blocks J, the full set of equations must be solved iteratively.

Defining appropriate blocks is important to the efficiency of DC-JI. First, dividing the system into fewer, larger blocks will reduce the number of iterations required to achieve self-consistency. On the other hand, the cost of solving the polarization equations directly for each block increases with larger block sizes. Optimal efficiency will be a balance between the cost of each iteration and the total number of iterations required. Second, convergence will be most rapid if the largest coupling elements $\mathbf{T}$ are included in the diagonal blocks $\mathbf{Z}$, which requires ordering of the matrix elements to ensure that spatially proximal polarizable sites occur near one another in the rows/columns of $\tilde{\mathbf{Z}}$.

Accordingly, before solving the polarization equations, we perform K-means clustering to identify natural, three-dimensional groups of polarizable sites. The K-means

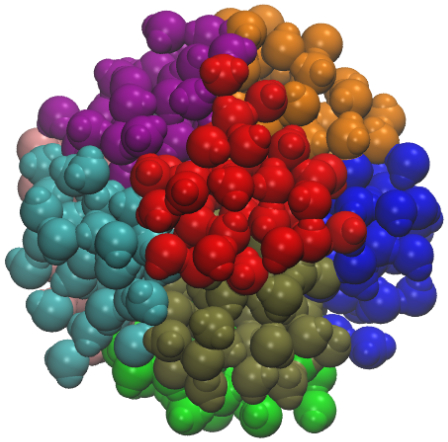


Figure 2.2: K-means partitioning of a $(H_2O)_{365}$ droplet into ten sub-clusters.

clusters are defined using a distance-based criterion, which leads to roughly spherical sub-clusters. All polarizable sites are clustered independently, with no effort made to maintain entire molecules/fragments within the same cluster. Splitting a molecule up over multiple sub-clusters does not affect the resulting energy, since the AMOEBA force field includes scaling parameters that control which intramolecular sites polarize one another that are handled when evaluating the total field.[33] Figure 2.2 shows K-means clustering of a $(H_2O)_{365}$ droplet into ten sub-clusters. Note that other researchers have also recognized the importance of spatially-driven domain decomposition in the polarization problem.[17] That work provides few details about the domain decomposition, but it does not appear to use K-means clustering nor the direct solution of the polarization within each block as proposed here.

The specific clustering generated by the K-means procedure does depend on the random initial guess. As will be demonstrated in Section 5.4, the convergence rate of DC-JI is variable and does depend on the random seed. However, switching to either Fuzzy DC-JI

(see Section 2.2.3) or employing DIIS extrapolation eliminates virtually all variability in the convergence rate (i.e. to within no more than ± 1 iteration). The chosen number of blocks K determines the average number of polarizable sites per block, but individual block sizes are not constrained during the K-means clustering. Inhomogeneous block sizes pose no problem for a serial implementation, though it might be beneficial to homogenize the block sizes to achieve better load balancing in a parallel implementation.

2.2.3 Fuzzy DC-JI

Block Jacobi iterations provide a substantial improvement in convergence compared to conventional JI, but it still can be slow to incorporate the off-diagonal couplings between blocks. If chosen appropriately, the use of overlapping domains can significantly accelerate the convergence without substantially increasing the cost per iteration. The fuzzy DC-JI algorithm has the same basic structure as the DC-JI one, except that it allows a given polarizable site to exist in multiple blocks. The "fuzzier" the partitioning, the more different blocks a given polarizable site will belong to. Because the induced dipole on a site with fuzzy blocks incorporates information from multiple distinct blocks, each iteration better reflects the overall many-body polarization in the system, reducing the total number of iterations required to achieve self-consistency. On the other hand, spreading sites over multiple blocks effectively increases the size of each block. If the blocks are too fuzzy, the growth in block sizes will increase the computational costs of solving each block equation faster than the computational savings gained from reducing the number of iterations.

To assign the polarizable sites to appropriate blocks, K-means clustering is once again used to locate the centroid of each block. However, instead of simply assigning each

polarizable site to its nearest centroid, fuzzy DC-JI computes the membership weight $w(R)$ at every nearby centroid. The site is then assigned to all blocks on whose centroids it has sufficiently large membership, and excluded from the remaining blocks. In Section 5.4, we consider several potential membership functions of the form $w(R) = R^{-n}$, for n = 1–3. The smaller n is, the more slowly the weights decay and the fuzzier the block boundaries obtained will be.

Once the weight $w_{i\in I}$ has been computed for polarizable site i in every potential block I, the weights are normalized to sum to 1. The polarizable site then becomes a member of every block for which its normalized weight is 0.1 or larger. This arbitrarily chosen threshold implies that a given site will have membership in at most ten blocks, but in practice the number of membership weights greater than 0.1 is typically much smaller. Once membership has been determined, the sum of the weights are renormalized to 1 among only those blocks in which the site is actually a member.

The fuzzy DC-JI polarization equations are solved for each block in the same manner as the DC-JI ones. However, the multiple distinct fuzzy dipoles for a given site arising from each block in which it is a member must be defuzzified into a single crisp value that incorporates contributions from all of the fuzzy values. Within a given block, one generally expects that induced dipoles on sites closer to the interior of the cluster will be more accurate than those on the outer boundaries, and those interior dipoles should factor more heavily in the final crisp dipole. The inverse dependence of the membership weights on the distance between the polarizable site and the cluster centroid captures this. Accordingly, the crisp induced dipole $\boldsymbol{\mu}_i^C$ on site i is computed as the weighted average

of the fuzzy induced dipoles $\boldsymbol{\mu}_i^F$ from every block I containing that polarizable site, with weights determined from the final renormalized set of membership weights w_i,

$$\boldsymbol{\mu}_i^C = \sum_I^{blocks} w_{i\in I} \boldsymbol{\mu}_{i\in I}^F \tag{2.10}$$

The resulting crisp induced dipoles can then be extrapolated with DIIS (optional) and used as inputs for the next iteration. Fuzzy DC-JI is similar to the Additive Schwarz method of overlapping domain decomposition.[31, 34] However, fuzzy DC-JI is distinguished by (1) the physically motivated use of K-means clustering to define the spatially proximal overlapping blocks and (2) the use of non-uniform, distance-based weights in the defuzzification step.

2.2.4 Software Implementation

The implementation of DC-JI/DIIS is summarized in Algorithm 1. Key steps involve (1) defining the blocks of polarizable sites, (2) initialization, and (3) evaluation of the induced dipoles and (4) DIIS extrapolation. Modifications required for the fuzzy DC-JI algorithm will be discussed in (5), followed by (6) computational analysis of the two algorithms.

K-Means clustering

To partition the system into K clusters that define the diagonal blocks of $\mathbf{Z}$, K centroids are chosen randomly according to the K-means++ initialization procedure.[35] The initial centroids are placed on random atoms (polarizable sites), with bias toward spatially well-distributed centroids. More specifically, after each centroid is picked, the

Algorithm 1 DC-JI Implementation

```
Define blocks of polarizable sites                    ▷ K-means clustering
V = Tm                                                ▷ Compute permanent potential
Build Z_II blocks                                     ▷ Mixture of (α)^-1 and T elements
Z_II = L_I L_I^T via Cholesky Decomposition           ▷ LAPACK dpotrf
μ = αV                                                ▷ Initial guess
while Induced dipoles are not converged do            ▷ Begin iterations
    loop over blocks I
        Build T_IJ                                    ▷ On-the-fly algorithm only
        V_I = V_0,I + Σ_J T_IJ μ_J                    ▷ Compute total potential
        Solve L_I L_I^T μ_I = V_I for μ_I             ▷ LAPACK dpotrs
    end loop
    Extrapolate μ via DIIS                            ▷ Starting in the 2nd iteration
    E_pol = (1/2) μ^T V                               ▷ Compute polarization energy
end while
```

distance $d(A)$ between each atom A its nearest centroid is computed. Subsequently, the probability of choosing an atom for the next centroid is computed as $P(A) = d(A)/\sum_A d(A)$. This initialization of widely-spread random sites typically leads to faster convergence of the K-means clustering and more optimal solutions.[35]

After this initialization procedure, every polarizable site is associated with its closest centroid. The Cartesian coordinates of each centroid are updated as the mean Cartesian position over all polarizable sites associated with that centroid. Polarizable sites are once again assigned to their nearest centroids, and the process is repeated until the centroid positions no longer change.

Computational savings in the K-means clustering procedure are achieved through the use of a neighbors list, which is similar to the Verlet neighbors list sometimes used in MD calculations to list pairs of particles whose interaction will be calculated.[36] In essence, for each polarizable site, the neighbors list tracks centroids which lie within 15 Å of that site. When deciding which centroid to assign a given polarizable site, comparisons are only

made against centroids in that site's neighbors list. Because the centroids move during the K-means procedure, the neighbors list is re-determined whenever the accumulated average change in position of the centroids reaches one half the radius of the sphere (e.g 7.5 Å). Use of a neighbors list substantially reduces the computational cost of K-means in large systems.[37] In practice, the cost of the K-means clustering is trivial compared to solving the polarization energy, as discussed in Section 2.4.1.

Initialization

After determining the blocks of polarizable sites, one first computes the permanent potential arising from the permanent multipoles,

$$\mathbf{V} = \mathbf{T}\mathbf{m} \tag{2.11}$$

where $\mathbf{m}$ is a vector of all permanent multipoles and $\mathbf{T}$ here is an interaction tensor with elements up to rank 2 (charges, dipoles, and quadrupoles). Second, one constructs the appropriate diagonal blocks $\mathbf{Z}_{II}$ of the $\tilde{\mathbf{Z}}$ matrix. This requires the inverse polarizabilities (which are simply a scalar for each atom in the isotropic AMOEBA model) and interaction tensor elements $\mathbf{T}$. The $\mathbf{T}$ elements in $\mathbf{Z}_{II}$ are a subset of those required in building $\mathbf{V}$, and the loops to build $\mathbf{V}$ and $\mathbf{Z}_{II}$ can be intertwined. Because AMOEBA includes only up to induced dipoles, each site-site interaction tensor in $\mathbf{Z}_{II}$ is a symmetric 3×3 matrix with six unique elements given by,

$$T_{\alpha\beta}^{AB} = \lambda_5 \frac{3R_\alpha^{AB} R_\beta^{AB}}{(R^{AB})^5} - \lambda_3 \frac{\delta_{\alpha\beta}}{(R^{AB})^3} \tag{2.12}$$

where R_α and R_β correspond to Cartesian components of the vector between the two sites, and $\delta_{\alpha\beta}$ is the Kronecker delta involving those two directions. The Thole damping functions λ_3 and λ_5 are defined elsewhere.[8]

Third, Cholesky decomposition $\mathbf{Z}_{II} = \mathbf{L}_I\mathbf{L}_I^T$ is performed for each block I. Because $\mathbf{Z}_{II}$ does not change during the later iterative portions of the DC-JI algorithm, we store the Cholesky decompositions $\mathbf{L}_I$ instead of $\mathbf{Z}_{II}$. The $\mathbf{L}_I$ matrices will be used to solve the linear polarization equations for the induced dipoles $\boldsymbol{\mu}_I$ in each iteration.

Finally, before beginning the iterations, we compute an initial guess for the induced dipoles as $\boldsymbol{\mu} = \boldsymbol{\alpha}\mathbf{V}$. This corresponds to the first iteration of a conventional JI model with $\boldsymbol{\mu} = 0$ on the right-hand side of Eq 3.8. Evaluation of the initial guess is inexpensive, since $\mathbf{V}$ is already available, and it typically reduces the number of iterations to converge the calculation by one. Of course, subsequent time steps in an MD trajectory can initialize the induced dipoles based on those from previous time step(s), further accelerating the convergence of the polarization energy by a few iterations.[38, 13]

Evaluation of the induced dipoles

Once the initialization steps have been performed, the iterative solution of the polarization equations (Eq 3.10) can begin. For each iteration, this process involves building the interaction tensors between blocks $\mathbf{T}_{IJ}$, contracting them with the induced dipoles for other blocks J, and adding them to the static field to obtain the total potential $\mathbf{V}_I$. If sufficient memory is available, the dipole-dipole blocks of $\mathbf{T}_{IJ}$ built to evaluate the permanent potential $\mathbf{V}$ during the initialization phase can be stored in sparse form for each atom pair for subsequent use during the iterations (referred to as the "pre-computed $\mathbf{T}$" algorithm).

Otherwise, they are computed anew each iteration ("on-the-fly **T**" algorithm). Second, one solves Eq 3.10 to obtain the new induced dipoles $\boldsymbol{\mu}_I$ using the Cholesky decomposed form of $\mathbf{Z}_{II}$ (the $\mathbf{L}_I$ computed during the initialization) via the standard `dpotrs` LAPACK routine.

DIIS extrapolation

DIIS extrapolation is used to accelerate the convergence of the DC-JI iterations starting in the second iteration. The extrapolated induced dipoles are written as a linear combination of the sets of induced dipole vectors $\boldsymbol{\mu}^{(j)}$ from recent iterations j,

$$\boldsymbol{\mu}^{extrap} = \sum_{j=1}^{n} c_j \boldsymbol{\mu}^{(j)} \tag{2.13}$$

The extrapolation coefficients c_j are obtained by solving,

$$\begin{bmatrix} B_{11} & B_{12} & \cdots & B_{1n} & -1 \\ B_{21} & B_{22} & \cdots & B_{2n} & -1 \\ \vdots & \vdots & \ddots & \vdots & -1 \\ B_{n1} & B_{n2} & \cdots & B_{nn} & -1 \\ -1 & -1 & -1 & -1 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -1 \end{bmatrix} \tag{2.14}$$

where B_{jk} is the inner product between two residual vectors that represent the change in the induced dipoles between iterations, $B_{jk} = \left(\Delta\boldsymbol{\mu}^{(j)}\right)^T \Delta\boldsymbol{\mu}^{(k)}$ and $\Delta\boldsymbol{\mu}^{(j)} = \boldsymbol{\mu}^{(j)} - \boldsymbol{\mu}^{(j-1)}$. The induced dipole history list is stored for up to twenty iterations. In practice, all the algorithms explored here converge in fewer than twenty iterations when DIIS extrapolation is employed.

Modifications for Fuzzy DC-JI

The fuzzy DC-JI algorithm is summarized in Algorithm 2. The algorithmic structure is largely the same as for DC-JI. The main differences occur in the K-means clustering and defining of the blocks. Once the locations $\mathbf{R}_I$ of the K-means centroids have been found, one must determine the membership list for each block. To do so, one first computes the weights for each polarizable site i in each block I whose centroid lies within 10 Å. The individual weights are given by $w_{i\in I} = 1/(\mathbf{r}_i - \mathbf{R}_I)^2$, were $\mathbf{r}_i$ is the position of the polarizable site. This weight function appears to provide a good balance between the degree of fuzziness and the iterative convergence rate (see Section 2.4.1). Once all weights have been computed for a given site i, the weights are normalized such that $\sum_I w_{i\in I} = 1$. The site becomes a member of every block for which the normalized weight is at least 0.1. The final membership weights are then renormalized among the subset of blocks in which that polarizable site is a member.

The efficient calculation of the total potential is mildly complicated by the overlapping nature of the blocks. For efficiency of the software loop structures, we opted for an implementation which computes the interactions with all blocks J, and then subtracts double-counted contributions from sites i which are already included in fuzzy block I. To facilitate this, the $\mathbf{Z}_{II}$ blocks containing the necessary interaction tensor $\mathbf{T}$ elements are retained during the initialization (in addition to the Cholesky decomposed forms $\mathbf{L}_I$).

Crisp induced dipoles $\boldsymbol{\mu}^C$ are used when computing the total potential during the iterations. After each iteration, the crisp dipoles are obtained as the weighted average of the fuzzy dipoles. If DIIS extrapolation is employed, it is applied to the crisp dipoles after

defuzzification.

Algorithm 2 Fuzzy DC-JI Implementation

Compute block centroids $\mathbf{R}_I$ for each block I ▷ K-means clustering
Compute membership weights $w_{i\in I} = 1/(\mathbf{r}_i - \mathbf{R}_I)^2$ for each site i/block I.
Assign block membership, renormalize $w_{i\in I}$ for each site i. ▷ Blocks and weights now known.
$\mathbf{V} = \mathbf{Tm}$ ▷ Compute permanent potential
Build $\mathbf{Z}_{II}$ blocks ▷ Mixture of $(\boldsymbol{\alpha})^{-1}$ and $\mathbf{T}$ elements
$\mathbf{Z}_{II} = \mathbf{L}_I\mathbf{L}_I^T$ via Cholesky Decomposition ▷ LAPACK `dpotrf`
$\boldsymbol{\mu}^C = \boldsymbol{\alpha}\mathbf{V}$ ▷ Initial guess
while Induced dipoles are not converged **do** ▷ Begin iterations
  loop over blocks I
    Build $\mathbf{T}_{IJ}$ ▷ On-the-fly algorithm only
    $\mathbf{V}_I = \mathbf{V}_{0,I} + \sum_J \mathbf{T}_{IJ}\boldsymbol{\mu}_J^C - \sum_{i\in I} \mathbf{T}_{ii}\boldsymbol{\mu}_i^C$ ▷ Compute total potential
    Solve $\mathbf{L}_I\mathbf{L}_I^T\boldsymbol{\mu}_I^F = \mathbf{V}_I$ for $\boldsymbol{\mu}_I^F$ ▷ LAPACK `dpotrs`
  end loop
  $\boldsymbol{\mu}_i^C = \sum_I w_{i\in I}\boldsymbol{\mu}_{i\in I}^F$ ▷ Defuzzification of dipoles
  Extrapolate $\boldsymbol{\mu}$ via DIIS ▷ Starting in the 2nd iteration
  $E_{pol} = \frac{1}{2}\boldsymbol{\mu}^T\mathbf{V}$ ▷ Compute polarization energy
end while

Computational analysis

The computational effort in the DC-JI algorithm is dominated by a handful of steps. Let M and N be the number of polarizable sites in a block and the entire system, respectively, with $M \ll N$. For simplicity of discussion, assume that all K blocks have identical size (and therefore $N = KM$). In the initialization, the most expensive step is the initial construction of the static field $\mathbf{V}$, which requires $\mathcal{O}(N^2)$ effort. Evaluating all $\mathbf{Z}_{II}$ blocks requires only $\mathcal{O}(KM^2) = \mathcal{O}(NM)$ effort, and all of the off-diagonal elements in $\mathbf{Z}_{II}$ are already computed when evaluating the interaction tensor $\mathbf{T}$ needed to evaluate $\mathbf{V}$. Cholesky factorization for each block of $\mathbf{Z}_{II}$ scales as $\mathcal{O}(M^3)$, or a total $\mathcal{O}(NM^2)$ for all K blocks.

Once the iterations begin, the computational effort for solving each of the K blocks is dominated by the construction of $\mathbf{T}_{IJ}$ and its contraction with $\boldsymbol{\mu}_J$ (when building $\mathbf{V}_I$), which scales $\mathcal{O}(NM)$ per block (or $\mathcal{O}\big(N^2\big)$ total). Solving the linear system of equations for each block from the Cholesky factorized $\mathbf{Z}_{II}$ matrices to obtain the induced dipoles requires only $\mathcal{O}\big(M^2\big)$ effort ($\mathcal{O}(NM)$ for all blocks), which is trivial as long as $M \ll N$. The effort associated with the DIIS extrapolation based on only a handful of previous iterations is also negligible. In summary, DC-JI combines a non-iterative cubic-scaling Cholesky step whose cost is kept reasonable through prudent choice of the block size, and iterative quadratic scaling steps that depend on the overall system size.

The same general scaling arguments apply for the fuzzy DC-JI algorithm. The individual block sizes are larger in the fuzzy version, but the sizes are kept reasonable through the choice of weight function. Computational costs associated with the additional setup, book-keeping, and defuzzification steps in fuzzy DC-JI are all small compared to the main algorithmic steps.

As noted above, the $\mathbf{T}_{IJ}$ interaction tensors computed during the initialization steps can be stored in memory throughout if sufficient memory is available ("pre-computed $\mathbf{T}$" algorithm). While the $\mathbf{T}$ blocks can be stored in moderately sparse form due to the symmetry within the blocks for each atom pair $\mathbf{T}_{ab}$, the total memory storage still scales with the square of the number of polarizable sites (the same as the overall $\mathbf{Z}$ matrix, but with a smaller prefactor). In a serial implementation of the algorithm, this memory storage requirement will become prohibitive in large systems.

Although the current study focuses on the serial software implementation, it is

worthwhile to consider a potential parallel implementation. K-means can readily be distributed over many processors, with each process handling a unique subset of the centroids. Only communication of the K centroid locations is required after each iteration. Once the blocks have been defined, each block I is assigned to a worker. All initialization and iterative steps associated with solving for the induced dipoles on that block can be carried out separately and independently. Only after each iteration must the new induced dipoles be broadcast to the other workers. In the fuzzy DC-JI algorithm, one must broadcast the fuzzy dipoles, compute the crisp dipoles, and then broadcast the crisp dipoles to other nodes.

For DC-JI, the DIIS extrapolation could either be performed separately within each block before communication of the updated induced dipoles, or after the dipoles have been harvested by a central worker but before they are broadcast out to the remaining nodes. For fuzzy DC-JI, DIIS probably cannot be employed until after defuzzification. Load-balancing of the workers might require some finesse, particularly if the K-means block sizes are inhomogeneous. However, in the non-overlapping DC-JI case, subsequent iterations of the self-consistent polarization by a given worker can continue semi-asynchronously, even if updated dipoles from some other blocks have not yet been received.

This algorithm would be well-suited for a hybrid OpenMP/MPI implementation, where OpenMP is used to parallelize the matrix and other operations for a given block I, and MPI is used to distribute the different blocks over many nodes. Such an approach would also make it easier to exploit the computational savings associated with the pre-computed $\mathbf{T}$ algorithm. In that case, each node would only need to store the subset of $\mathbf{T}_{IJ}$ involving the block(s) I being solved on that particular node. This reduces the local node memory

requirement for pre-computed $\mathbf{T}$ from $\mathcal{O}(N^2)$ to $\mathcal{O}(MN)$. As long as enough nodes are available to ensure that M is sufficiently small, the pre-computed $\mathbf{T}$ algorithm should be feasible.

Periodic boundary conditions are not considered in any detail here. However, both the DC-JI/DIIS and Fuzzy DC-JI/DIIS algorithms are amenable to the inclusion of periodic boundary conditions using Ewald techniques or a minimum-image convention. The key requirement for the efficient use of the divide-and-conquer models will be that block sizes are smaller than the real-space interaction cutoffs, such that the interaction tensor elements in each $\mathbf{Z}$ block will have the same basic form as described in Eq 2.3. Longer-range interactions stemming from periodic images, reciprocal space, or an external electric field will couple in through the external potential terms on the right-hand side of Eq 3.10.

2.3 Computational Methods

The JI, DC-JI, and fuzzy DC-JI algorithms described here were implemented in a local copy of Tinker 7.1.[32] We also compare against the existing Tinker 7.1 implementation of PCG. Efficiency comparisons obviously depend both on the basic algorithm and the specific implementations. Common implementation strategies were used in constructing key intermediates such as the field $\mathbf{V}$ and interaction tensor $\mathbf{T}$ throughout to facilitate even-handed comparisons. The performance of both PCG and the DC-JI models could probably benefit from further code refinements.

Constant-energy molecular dynamics simulations were run with time steps of 1.0 femtoseconds. Smaller time step sizes of 0.25 femtoseconds were used in the energy con-

servation analysis. Most calculations were performed on a ubiquitin protein surrounded by 2,835 water molecules (9,737 atoms total). A few calculations employ a large octahedron containing 6,400 water molecules (19,200 atoms total). Both geometries were taken from the example files distributed with the Tinker software package. Induced dipoles were converged to a root-mean-square (RMS) threshold of 10^{-6} Debeye unless otherwise stated. In MD simulations, the induced dipoles at each subsequent time step were initialized with the converged dipoles from the previous step unless otherwise specified. All timings reported are CPU timings on a single core of a 2.4 GHz Xeon E5-2630 v3 chip with 64 gigabytes of RAM.

2.4 Results and Discussion

2.4.1 Behavior of the algorithms

The spatial clustering of the polarizable sites plays an important role in the efficiency of the DC-JI algorithms. Figure 2.3 compares the distributions of significant matrix elements in $\tilde{\mathbf{Z}}$ from (a) a random sorting of the polarizable sites versus (b) the sorting achieved by K-means in a cluster of 85 water molecules. While the ordering of sites stemming from the atom ordering in a user-specified structure coordinate file may exhibit more spatial clustering than the random $\tilde{\mathbf{Z}}$ matrix shown here (depending on how the user arranges the input file), it can still be non-optimal due to both the inherent challenges in mapping a 3-D system onto a two-dimensional 2-D matrix and the changes in atomic positions during an MD simulation. The K-means sort clearly concentrates large-magnitude

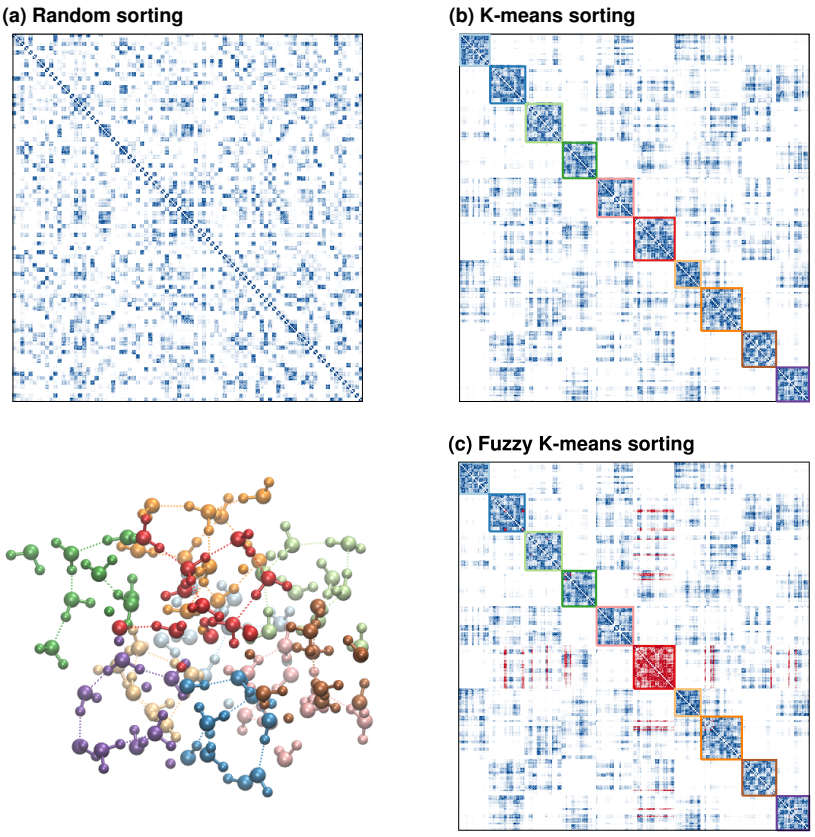


Figure 2.3: Comparison of the $\tilde{\mathbf{Z}}$ matrix with (a) random (b) K-means sorting, and (c) Fuzzy K-means sorting of the polarizable sites for a $(H_2O)_{85}$ cluster (255 polarizable sites, 765 induced dipole vector components). The 10 colored sub-clusters in the molecular figure are indicated by the colored boxes in (b) and (c). The red elements in (c) correspond to terms captured in that particular fuzzy block. Matrix elements are plotted in log scale, with coloring for elements with magnitude 0.005 Å^{-3} or larger.

matrix elements along the diagonal blocks.

Because it allows for the most important mutual polarization effects to be solved directly in DC-JI, this clustering reduces the number of iterations required to converge the self-consistent polarization. Conventional JI requires 41 iterations to converge in this system. In a randomly sorted matrix, employing block Jacobi with the same block sizes but no K-means sorting provides no clear convergence advantage over conventional JI. In

contrast, with the K-means sorting shown in Figure 2.3b, only 15 iterations are required to converge the induced dipoles with DC-JI.

Despite the substantial reduction in the number of iterations in the K-means sorted blocked algorithm, many significant off-diagonal block contributions remain, corresponding to many-body couplings among atoms in the various sub-clusters. The overlapping block treatment in the fuzzy DC-JI algorithm helps capture some of these effects. In Figure 2.3 for example, the atoms in the red cluster directly border atoms in six of the nine other sub-clusters. Polarization on the red atoms will be particularly affected by adjacent atoms in these neighboring clusters. Only the light blue, salmon, and gold-colored sub-clusters are not directly adjacent to the red sub-cluster atoms. Figure 2.3c plots the same K-means sorting as in Figure 2.3b, but with all matrix elements included in the fuzzy version of the red sub-cluster highlighted in red. Close inspection reveals that the fuzzy red block includes couplings with all six adjacent sub-clusters, omitting couplings only from the three more distant sub-clusters mentioned earlier. In other words, the fuzzy K-means procedure identifies off-diagonal coupling elements in the 2-D matrix that are important in the 3-D system. The number of iterations required to converge the polarization drops from 15 with DC-JI (non-overlapping) blocks to 8 in the fuzzy DC-JI (overlapping) case. Further acceleration is possible with DIIS, as will be discussed below.

The next question is how large of blocks one should use. The use of fewer, larger sub-clusters of atoms/polarizable sites should translate to the inclusion of more significant off-diagonal coupling elements from $\tilde{\mathbf{Z}}$ into the direct solution portion of DC-JI, leaving weaker couplings in the off-diagonal blocks to be accounted for iteratively. In the limit of

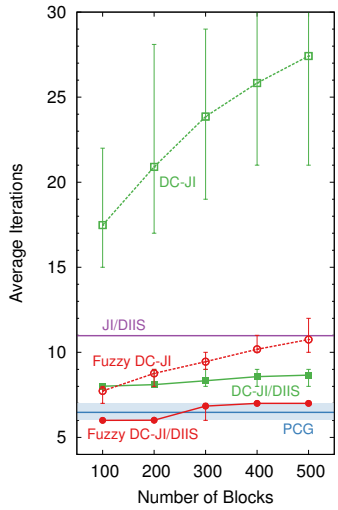


Figure 2.4: Average number of iterations required to converge the polarization to 10^{-6} Debeye RMS change of the dipoles in ubiquitin/water over 250 MD steps with various block sizes and polarization algorithms. Error bars/shaded regions indicate the range of iterations required in 90% of the time steps. PCG and JI/DIIS do not employ blocks, so their convergence rates are independent of block size.

all polarizable sites being placed in a single block, only one iteration is required. On the other hand, the cost of solving for the induced dipoles via Cholesky decomposition increases with block size.

Figure 2.4 plots the mean number of iterations required to converge the polarization as a function of block size in the 9,737-atom ubiquitin/water system. These data reflect an average over 250 1 fs MD steps. Error bars and shaded regions in the figure indicate the range of iterations required in 90% of the MD time steps. For the DC-JI and fuzzy DC-JI models, the simulations involved five sets of 50 steps, with different initial random seeds for the K-means. Since PCG and JI/DIIS do not involve a random seed, the results were averaged over a single longer trajectory. In all cases, converged dipoles from the previous step were used as the initial guess for the next step.

The convergence rate of DC-JI is clearly sensitive to the block size. At 500 blocks (mean block size 19.5 atoms, which corresponds to a matrix block dimension of ∼58 since each atom has three dipole components), converging the polarization requires 27.4 iterations on average. The convergence rate is also highly variable, ranging between 21–38 iterations in 90% of the MD steps. The average number of iterations and range of iterations are consistent across each of the five runs started with different initial random seeds for the K-means clustering, suggesting that the randomness in the clustering is not a significant issue. Using 100 blocks of mean size 97.4 atoms accelerates DC-JI convergence dramatically, down to 17.5 iterations (90% range of 15-22 iterations). So while DC-JI represents an improvement over conventional JI (which frequently fails to converge in this system), the convergence behavior can still be slow if smaller blocks are used. Incorporation of the off-diagonal couplings remains slow in DC-JI.

Fuzzy DC-JI handles those off-diagonal couplings better, significantly reducing both the number of iterations required and the sensitivity to block size. With 500 blocks, fuzzy DC-JI decreases the iterations from 27.4 to only 10.8. With 100 blocks, it requires only 7.7 iterations instead of 17.5. The convergence rate becomes much more consistent with the fuzzy blocks as well, with variations of no more than one iteration. The K-means random seed has negligible effect on the convergence of fuzzy DC-JI.

Further acceleration can be achieved using DIIS extrapolation, especially for the more poorly converging methods. Whereas conventional JI frequently fails to converge in this system, JI/DIIS consistently requires 11 iterations. DC-JI/DIIS converges in 8.0–8.7 iterations (90% range 8–9 iterations), depending on block size. For systems with 100–200

Table 2.1: Effect of the fuzzy membership function on the average number of atoms per block, average iterations to converge the polarization, and CPU time required per iteration to evaluate the induced dipoles in ubiquitin/water with DC-JI/DIIS or Fuzzy DC-JI/DIIS.

Algorithm	Weights	# of Blocks	Block Size	Iterations	Time (s)
DC-JI	n/a	100	97.4 ± 13.9	8.0	14.1
DC-JI	n/a	200	48.7 ± 10.0	8.1	14.1
Fuzzy DC-JI	R^{-3} weight	200	77.5 ± 14.6	7.0	13.0
Fuzzy DC-JI	R^{-2} weight	200	94.9 ± 18.3	6.0	11.9
Fuzzy DC-JI	R^{-1} weight	200	143.9 ± 30.5	6.0	13.4

blocks, fuzzy DC-JI/DIIS consistently converges in 6 iterations, which is almost half that of JI/DIIS and is slightly better than the 6.5 iterations of PCG. With 400–500 blocks, the fuzzy DC-JI/DIIS convergence occurs in only 7.0 iterations. Notice that DIIS substantially reduces the sensitivity of the DC-JI and fuzzy DC-JI convergence to block size, which allows one to focus on which block sizes are most computationally efficient.

To understand better how the fuzzy algorithm behaves, Table 2.1 compares the block sizes, convergence rates, and time per polarization calculation for several different models. For the non-overlapping DC-JI model with 200 blocks, the average block contains 48.7 atoms. For the fuzzy models, the degree of fuzziness depends on the choice of the weighting function, $w(R) = R^{-n}$. Smaller exponents n translate to fuzzier blocks with more overlaps. Progressing through R^{-3}, R^{-2}, and R^{-1}, the mean block size for the same 200 blocks increases by roughly 60%, 100%, and 200%, respectively, compared to the non-overlapping case. The variability in the block size also increases, as indicated by the standard deviations in Table 2.1.

As the average block size increases, the average number of iterations required for convergence decreases. Notably, however, the convergence improvements from the fuzzy

membership go well-beyond the simple block-size increases. Switching from 200 to 100 blocks in the non-overlapping DC-JI model doubles the number of atoms per block, but it only improves the convergence rate by 0.1 iterations on average (hence the essentially identical average DC-JI timings per MD step for both 100 and 200 blocks). For comparison, the fuzzy DC-JI/DIIS model with 200 blocks and R^{-2} weighting has approximately the same average block size as DC-JI/DIIS with 100 blocks, but it converges two iterations (25%) faster. This result implies that the convergence enhancements from the fuzzy DC-JI model stem mostly from the improved treatment of polarization at sites on the edges of the block sub-clusters, rather than from simple increases in average block size.

The decision to adopt the R^{-2} weighting function was made empirically. R^{-3} weighting has physical appeal since it mimics the distance dependence of the dipole-dipole interaction. However, R^{-3} decays relatively quickly, and this lower degree of fuzziness does not accelerate the convergence as much as the more slowly decaying weighting functions (Table 2.1). On the other hand, R^{-1} weighting makes the blocks too large. Despite generating blocks that are 50% larger than those from the R^{-2} weighting, the R^{-1} weighting still requires 6 iterations to converge the polarization. At the same time, the larger block size with R^{-1} weighting starts to increase the costs associated with the Cholesky decomposition and solving for the induced dipoles (see discussion of detailed timing breakdowns below). The R^{-2} weighting balances the number of iterations required for convergence and the overall computational cost.

Before moving on, recall that fuzzy DC-JI employs distance-based membership weights to defuzzify the induced dipoles into crisp values. For ubiquitin/water with 200

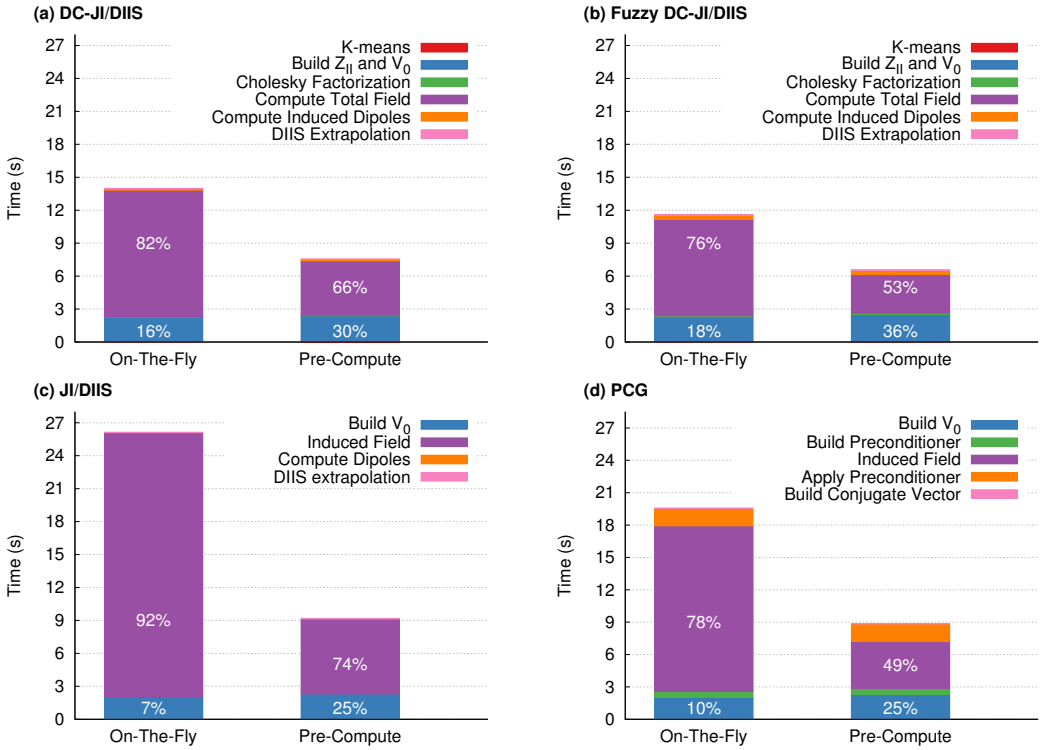


Figure 2.5: Timing breakdown in the ubiquitin/water system for a single self-consistent polarization energy calculation using (a) DC-JI/DIIS (200 blocks) (b) Fuzzy DC-JI/DIIS (200 blocks) (c) JI/DIIS or (d) PCG, with the external interaction tensor elements $\mathbf{T}_{IJ}$ either evaluated on-the-fly at each iteration (lower memory requirement algorithm) or pre-computed once and stored in memory throughout.

blocks and R^{-2} weighting, fuzzy DC-JI/DIIS requires an average of 6.0 iterations to converge (Table 2.1). Alternatively, one might just employ a conventional unweighted average of the fuzzy dipoles as in Additive Schwarz domain decomposition, but this increases the average number of iterations to 6.5. Using the weighted average incurs no appreciable additional overhead, so the average savings of 0.5 iterations translates to a computational savings of 7% compared to Additive Schwarz in this system.

Further insight is gained by analyzing the timing breakdown of the various algorithms. Figure 2.5 plots the timing details for a single self-consistent polarization calculation

on the ubiquitin/water system with DC-JI/DIIS, Fuzzy DC-JI/DIIS, JI/DIIS, and PCG. 200 blocks were used for both DC-JI models. The calculations converged in 8 (DC-JI), 6 (Fuzzy DC-JI) 11 (JI/DIIS), and 6 (PCG) iterations, respectively. For the DC-JI models, the computational effort is dominated by the time required to compute the total potential (right-hand side of Eq 3.10), particularly building and contracting $\sum_J \mathbf{T}_{IJ}\boldsymbol{\mu}_J$. In the "on-the-fly" algorithm variant where $\mathbf{T}_{IJ}$ is rebuilt every iteration, this step consumes 76-82% of the total computational time for the two methods. The second largest portion of the effort (16-18%) goes to the initialization steps, particularly evaluating the static potential $\mathbf{V}$. The remaining few percent of the time is spent doing the initial K-means clustering, performing the Cholesky factorization of the diagonal $\mathbf{Z}_{II}$ blocks, and solving for the induced dipoles.

The larger block size and extra book-keeping of fuzzy DC-JI does increase the cost of some steps by up to a couple tenths of a second compared to DC-JI (most notably the Cholesky factorization and computing of the induced dipoles). Nevertheless, those differences are dwarfed by the savings from the reducing the number of iterations required to reach convergence by two. Instead of re-computing the $\mathbf{T}_{IJ}$ matrices "on-the-fly" during each iteration, one might store them in memory when they are computed during the initialization phase. As noted previously, storing these large matrices quickly becomes memory-prohibitive in a serial implementation, but it may be feasible in a parallel implementation when sufficient processors are available.

It is interesting to compare these timings against ones for JI/DIIS (Figure 2.5c) and PCG (Figure 2.5d). JI/DIIS is much more expensive than either divide-and-conquer algorithm largely due to the larger number of iterations and the need to evaluate the induced

field in each one. Pre-computing and storing the interaction tensor helps reduce costs substantially, from roughly 26 seconds to 9 seconds here. Nevertheless, JI/DIIS still requires a few seconds more than the divide-and-conquer models. PCG requires fewer iterations than JI/DIIS to converge, but each iteration is more expensive. The cost of applying the preconditioner is also notable. Again, pre-computing the interaction tensor elements helps, but PCG is still slower than the divide-and-conquer approaches.

To summarize, both DC-JI and fuzzy DC-JI converge better than conventional JI. When combined with DIIS extrapolation, fuzzy DC-JI converges much faster than JI/DIIS and on par with or better than PCG. The good performance of the fuzzy algorithm stems from its effective averaging of the fuzzy dipoles over multiple clusters, thereby better mimicking polarization in full system. As long as DIIS is employed, the divide-and-conquer algorithms perform well over a range of block sizes. Because specific block size is not critical, the number of blocks can be chosen to optimize computational efficiency.

2.4.2 Performance in molecular dynamics

Next, consider how these various polarization solvers perform in molecular dynamics simulations. The polarization time depends in part on how the induced dipoles are initialized in each MD step. Two different algorithms are considered here: either using the converged dipoles from the previous step, or using Kolafa's always stable predictor-corrector algorithm.[13] Figure 2.6 shows polarization timings with each algorithm over 1 ps of MD simulation time in the ubiquitin/water system and a truncated octahedron of 6,400 water molecules. 200 blocks were used for both systems, which corresponds to an average

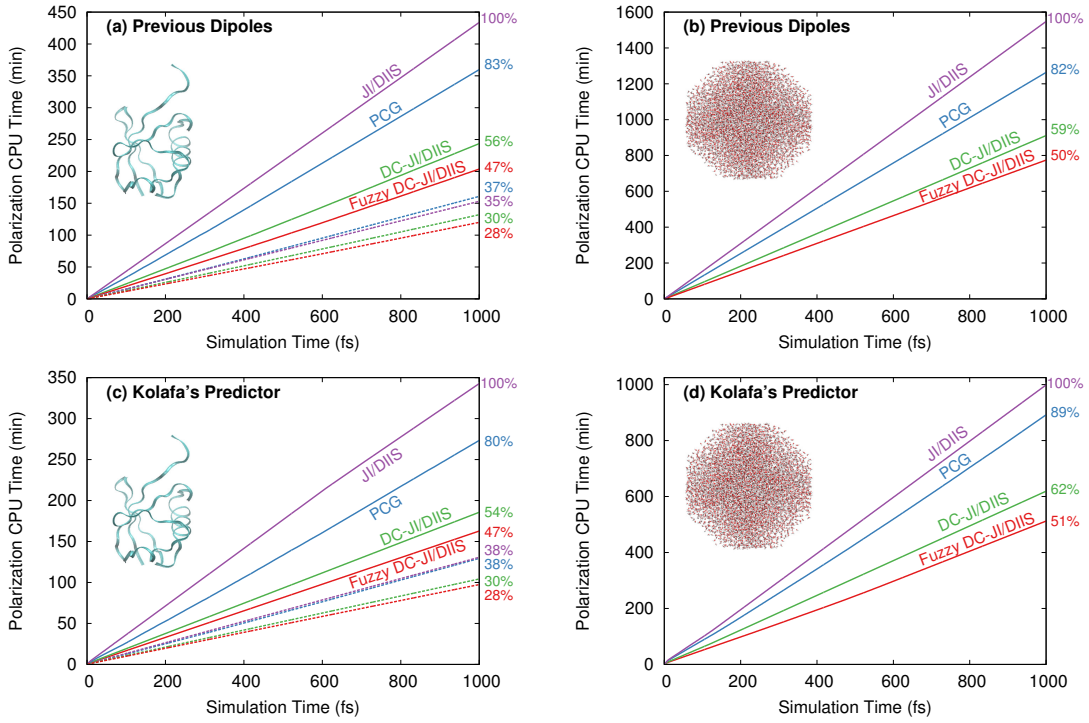


Figure 2.6: Comparison of polarization CPU timings for (a)/(c) ubiquitin surrounded by 2835 waters (9,737 atoms) or (b)/(d) a cluster of 6,400 water molecules (19,200 atoms). Solid lines correspond to the "on-the-fly" algorithm variants, while dotted lines result from "pre-computing" and storing the $\mathbf{T}$ matrices. The labels indicate whether the induced dipoles were initialized from the previous iteration or using Kolafa's always stable predictor-corrector algorithm. Percentages indicate the time savings over JI/DIIS.

of ∼50 and ∼100 atoms per block in the two respective systems. As seen in Figure 2.6, the predictor-corrector algorithm somewhat reduces the number of iterations required to converge the polarization and reduces the overall computational time, but it does not substantially alter the relative performance of the different algorithms. JI/DIIS performs the slowest in these simulations. PCG evaluates the polarization energy 11-20% faster. However, the DC-JI/DIIS and fuzzy DC-JI/DIIS algorithms are roughly a factor of two faster than JI/DIIS.

Because much of the computational cost is dominated by evaluation of the inter-

action tensors $\mathbf{T}$, even larger savings can be achieved if the elements of $\mathbf{T}$ are pre-computed and stored throughout, as shown for ubiquitin in Figure 2.6a,c. In the pre-computed variants of the algorithm, PCG and JI/DIIS require similar amounts of computational time. JI/DIIS disproportionately benefits from pre-computing $\mathbf{T}$ since a greater percentage of the time is spent evaluating the induced potential (Figure 2.5).

Nevertheless, the pre-computed versions of DC-JI/DIIS and Fuzzy DC-JI/DIIS still out-perform all other algorithms here with either dipole initialization scheme. The pre-computed variant of fuzzy DC-JI/DIIS requires only 28% the time of conventional JI/DIIS, and one-third the cost of conventional PCG. Storing $\mathbf{T}$ for large systems requires prohibitive amounts of computer memory (RAM) in a serial implementation. Indeed, we could not store $\mathbf{T}$ for the $(H_2O)_{6400}$ on a machine with 64 GB of RAM. However, as discussed in Section 2.2.4, a distributed-memory parallel implementation would only require storage of a subset of the $\mathbf{T}_{IJ}$ blocks on each node. With sufficient nodes available, the pre-computed $\mathbf{T}$ algorithm could become feasible, and the blocked nature of the divide-and-conquer algorithms makes them well-suited to parallel implementation.

Finally, we investigate the energy conservation behavior of the various solvers. Constant energy MD simulations were performed on ubiquitin (without aqueous solvent for the sake of efficiency) with 0.25 fs time steps for each of the four solvers (JI/DIIS, PCG, DC-JI/DIIS, and fuzzy DC-JI/DIIS) and various convergence thresholds for the induced dipoles. Converged dipoles from each iteration were used as initial guess in the subsequent iteration. As shown in Figure 2.7, the DC-JI and fuzzy DC-JI models behave similarly to JI/DIIS. Converging the induced dipoles to only RMS 10^{-4} Debeye leads to appreciable

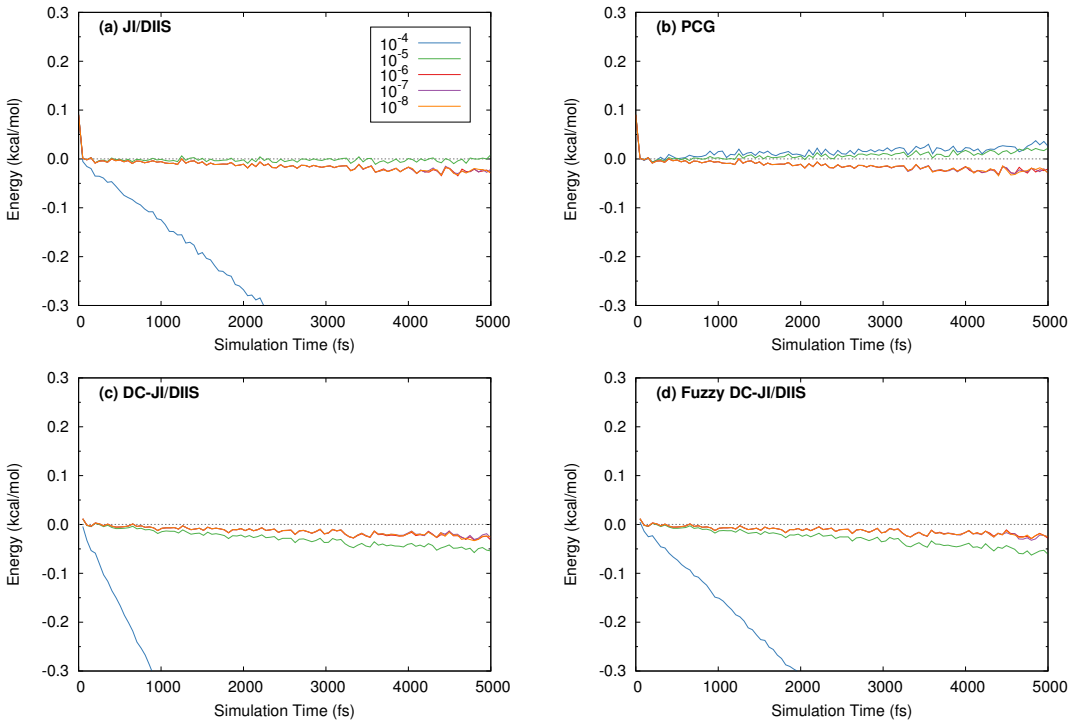


Figure 2.7: MD energy conservation in ubiquitin (no solvent) with four different polarization algorithms. Time steps of 0.25 fs were used.

energy drift almost immediately. PCG does not exhibit the same large drift for the loose 10^{-4} Debeye convergence in this system. For this particular trajectory and convergence criterion, PCG fortuitously converges to a residual that is on average around 50% smaller than that of the other three methods, and this appears sufficient to eliminate much of the energy drift. Indeed, a convergence criterion of 10^{-5} Debeye already behaves much better for all methods. Note that PCG does sometimes exhibit substantial energy drift with a 10^{-4} Debeye convergence criterion.[4] When convergence criteria of 10^{-6} Debeye or tighter are used, the drift becomes no more than a few hundredths of a kcal/mol for all methods in the simulation time frame shown here. In other words, as long as reasonably tight convergence criteria are used, the proposed DC-JI solvers conserve energy well.

2.5 Conclusions

In summary, two new, computationally efficient algorithms for evaluating the self-consistent polarization equations in polarizable force fields have been proposed. Based on non-overlapping and overlapping domain decomposition, these DC-JI and fuzzy DC-JI algorithms can provide savings of ∼2–3 over a conventional JI/DIIS or PCG implementation. They combine direct, non-iterative solution of the polarization equations within sub-clusters of atoms and iterative treatment of the polarization couplings between atoms. K-means clustering automatically identifies spatially localized sub-clusters that ensure rapid convergence of the iterations. The particularly good performance of the fuzzy DC-JI algorithm stems largely from its improved treatment of polarization sites near the sub-cluster edges. An $1/R^2$ weight function was empirically chosen for fuzzy DC-JI to control the degree of fuzziness and provide defuzzification weights for obtaining the final crisp induced dipoles.

Future work should focus on adapting these algorithms to periodic systems and to an efficient parallel implementation. In particular, a massively parallel implementation might be able to achieve significant computational savings by employing the pre-compute $\mathbf{T}$ matrix variant of the algorithms in large systems. The algorithm performs well over a range of block sizes. In a parallel implementation, this flexibility could prove useful by allowing one to adapt the number of blocks to achieve uniform distribution of the work over the available processors. Research in these directions is on-going.

Chapter 3

Divide-and-Conquer Jacobi Iterations II

3.1 Introduction

Quantum mechanical charge distributions respond to their environment, which significantly impacts system behaviors. The dipole moment of water can decrease ∼20% as it moves from bulk water to a non-polar protein pocket, for instance.[1] Capturing how the charge distribution responds to its environment is also critically important for maintaining the subtle balance of intra-protein and protein-environment interactions correctly[2] or for reproducing the proper dynamics in ionic liquids.[39] Reproducing these effects in classical force field simulations can be difficult—fixed-charge models are inherently incapable of describing dynamic changes in the charge distribution.

Polarizable force fields, on the other hand, can mimic these behaviors.[40, 41, 42]

Unfortunately, the improved physical description provided by polarizable force fields comes at significantly increased computational cost. In the AMOEBA (atomic multipole optimized energetics for biomolecular applications) force field,[8, 7] for example, where polarization is represented via an induced dipole model with Thole damping, obtaining the induced dipoles requires solving a set of linear equations of dimension three times the number of polarizable sites N. These equations can be solved "directly" using a finite number of operations, but such methods are only tractable for small systems due to their $O(N^3)$ complexity. Instead, iterative methods are traditionally used to solve for the induced dipoles. Two commonly used iterative solvers are Jacobi iterations (JI), accelerated with either over-relaxation (JOR)[7] or direct inversion of the iterative subspace (DIIS)[22, 23], and the preconditioned conjugate gradient (PCG) algorithm.[3] While these iterative methods vary in their rate of convergence and overall computational cost, they can in principle solve the polarization equations to arbitrary precision. In practice, the solution of the polarization equations is converged to within a computationally tractable user-specified tolerance.

To minimize the computational cost, approximations can be made when solving the polarization equations. Truncated versions of Jacobi iterations[25, 26] and preconditioned conjugate gradients.[30, 43] have been introduced. These methods typically perform the algorithm for a fixed number of iterations, which leads to several benefits. First, the iterative methods can be "unrolled" to a form for which analytical gradients of the polarization with respect to atomic position can be derived. This potentially minimizes the energy drift associated with iteratively solving the systems of equations to a finite convergence tolerance. Second, the user has a "knob" to control the computational cost of the algo-

rithm by choosing when to truncate. In this sense these methods might be characterized as "direct" in that they can be solved in a finite number of operations. However, the resulting dipole vector differs from the exact one, meaning that the potential energy surface on which the dynamics occur differs from the true one. Similarly, extended Lagrangian approaches eliminate the need to iterate the polarization equations at each time step by integrating the dynamics along an approximate shadow potential.[14, 44, 45] Other strategies express the polarization interactions via a truncated many-body expansion, such that mutual polarization is only considered up to 3-body interactions (3-AMOEBA),[27] or neglect mutual polarization completely and reparameterize the force field to compensate (iAMOEBA)[29].

In addition to algorithmic advances, improved hardware has opened the door to applying polarizable force fields for biological systems on chemically relevant timescales that would otherwise have been unattainable. The Tinker software package, which includes the polarizable AMOEBA force field, has expanded to a family of three codes that take advantage of hardware advances: canonical Tinker v8.1,[46] the Tinker-OpenMM[47] package which leverages graphical processing units for fast mixed-precision dynamics, and the Tinker-HP package[48] which enables massively parallel MPI applications on high performance computing systems.

Recently, we introduced a new, formally exact iterative polarization solver, the divide-and-conquer Jacobi iterations (DC-JI) algorithm.[49] DC-JI uses physically-motivated partitioning of the polarization problem to accelerate evaluation of the self-consistent induced dipoles. It partitions the set of polarizable sites into spatially local sub-clusters, which will generally include the strongest polarization interactions. The mutual polariza-

tion within each sub-cluster is solved directly within the field generated by more distant polarizable sites outside the sub-cluster. Mutual polarization between sub-clusters is captured iteratively. Convergence of the DC-JI solver can be accelerated further with DIIS and/or the use of fuzzy clustering.[49] The pseudo-direct nature of DC-JI/DIIS leads to a convergence of the polarization equations in a number of iterations comparable to PCG, but with a lower overall computational cost.

The initial implementation of DC-JI/DIIS in Tinker 7.1 performed ∼30–40% faster than PCG for non-periodic systems on a single processor.[49] In this paper, we adapt DC-JI/DIIS to periodic systems via the particle-mesh Ewald algorithm and introduce a massively parallel implementation within the Tinker-HP software package. Good parallel performance is demonstrated for systems containing hundreds of thousands of atoms and many hundreds of processor cores. Crucially, results presented here show that DC-JI/DIIS is not only faster than the conventional iterative polarization solvers like PCG, but that the solution it obtains at any given convergence threshold is simultaneously more energetically robust.

3.2 Theory

3.2.1 Background

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

is computed by minimizing the functional,

$$E_{pol} = \frac{1}{2}\boldsymbol{\mu}^T \tilde{\mathbf{Z}} \boldsymbol{\mu} - \boldsymbol{\mu}^T \mathbf{V}_0 \tag{3.1}$$

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

$$\tilde{\mathbf{Z}} = \begin{bmatrix} (\boldsymbol{\alpha}_A)^{-1} & -\mathbf{T}^{AB} & -\mathbf{T}^{AC} & \dots \\ -\mathbf{T}^{BA} & (\boldsymbol{\alpha}_B)^{-1} & -\mathbf{T}^{BC} & \dots \\ -\mathbf{T}^{CA} & -\mathbf{T}^{CB} & (\boldsymbol{\alpha}_C)^{-1} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{3.2}$$

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

$$T^{AB}_{\alpha\beta} = \lambda_5 \frac{3R^{AB}_\alpha R^{AB}_\beta}{(R^{AB})^5} - \lambda_3 \frac{\delta_{\alpha\beta}}{(R^{AB})^3} \tag{3.3}$$

where R_α and R_β correspond to Cartesian components of the vector between the two sites, capturing the anisotropic contributions to the interactions, and $\delta_{\alpha\beta}$ is the Kronecker delta which switches on the isotropic term. The Thole damping functions λ_3 and λ_5 prevent polarization catastrophe and are defined elsewhere.[8]

The vector of induced dipoles that minimizes the polarization energy is obtained

as the solution to the system of linear equations,

$$\tilde{\mathbf{Z}}\boldsymbol{\mu} = \mathbf{V}_0 \tag{3.4}$$

Upon substitution of Eq 3.4 into Eq 3.1, the minimized polarization energy is given by,

$$E_{pol} = \frac{1}{2}\boldsymbol{\mu}^T\mathbf{V}_0 - \boldsymbol{\mu}^T\mathbf{V}_0 = -\frac{1}{2}\boldsymbol{\mu}^T\mathbf{V}_0 \tag{3.5}$$

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

$$\frac{dE_{pol}}{dr_k^a} = \frac{\partial E_{pol}}{\partial r_k^a} + \frac{\partial E_{pol}}{\partial \boldsymbol{\mu}}\frac{\partial \boldsymbol{\mu}}{\partial r_k^a} \tag{3.6}$$

If the dipole vector minimizes the polarization energy, then $\frac{\partial E_{pol}}{\partial \boldsymbol{\mu}}$ is at a stationary point and the second term of Eq 3.6 vanishes. Iterative equation solvers converge towards the dipole vector that minimizes the energy, but they typically stop at a user-defined convergence threshold for computational expediency. Care must be taken to balance between using a looser-tolerance for computational efficiency and a tighter one for a more numerically exact solution. Loosely converged polarization equations will exhibit a non-zero second term in Eq 3.6 which can introduce energy drift during a molecular dynamics simulation.

3.2.2 Polarization Solvers

Jacobi iterations (JI) offers a simple iterative procedure for finding the dipole vector that minimizes the polarization energy functional. It corresponds to the intuitive

picture in which site A polarizes site B, site B polarizes A, and the process iterates until self-consistency is achieved for the induced dipoles. Formally, the response matrix $\tilde{\mathbf{Z}}$ can be partitioned into its diagonal elements $\mathbf{D}$ and off-diagonal elements $\mathbf{Y}$, $\tilde{\mathbf{Z}} = \mathbf{D} + \mathbf{Y}$. Substitution of the partitioned $\tilde{\mathbf{Z}}$ into Eq 3.4 and rearrangement yields,

$$\mathbf{D}\boldsymbol{\mu} = \mathbf{V}_0 - \mathbf{Y}\boldsymbol{\mu} \tag{3.7}$$

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

$$\boldsymbol{\mu} = \mathbf{D}^{-1}\left(\mathbf{V}_0 - \mathbf{Y}\boldsymbol{\mu}\right) = \boldsymbol{\alpha}\left(\mathbf{V}_0 + \mathbf{T}\boldsymbol{\mu}\right) \tag{3.8}$$

However, since the right-hand side depends on $\boldsymbol{\mu}$, Eq 3.8 must be solved iteratively. JI converges poorly if the spectral radius $p(\mathbf{D}^{-1}(-\mathbf{Y}))$ is near 1 and diverges if $p(\mathbf{D}^{-1}(-\mathbf{Y}))$ is greater than 1. The convergence behavior can be improved by coupling JI with DIIS. That is, after establishing a short history of the induced dipoles during the first few iterations, the induced dipoles are extrapolated via DIIS after each Jacobi iteration. DIIS has a long history in computational chemistry.[24]

Alternatively, the conjugate gradient (CG) method can solve for the polarization by minimizing E_{pol} in Eq 3.1. Poor convergence behavior is observed when the condition number of $\tilde{\mathbf{Z}}$ is large, but this can be improved by applying a preconditioner (PCG) to solve a modified system of equations:

$$P^{-1}\tilde{\mathbf{Z}}\boldsymbol{\mu} = P^{-1}\mathbf{V}_0 \tag{3.9}$$

where P^{-1} is some easily computed matrix that approximates $\tilde{\mathbf{Z}}^{-1}$. The diagonal precon-

ditioner used in Tinker-HP is easily parallelized and offers a satisfactory reduction in the condition number for high performance parallel implementations. The robustness of PCG depends on $\tilde{\mathbf{Z}}$ being symmetric and positive definite, and it can be sensitive to numeric precision.[48]

3.2.3 Divide-and-conquer Jacobi Iteration method

DC-JI amounts to a block JI method[31] with physically motivated blocking.[49] Because nearby polarizable sites are expected to polarize each other most strongly, DC-JI partitions the system into spatially localized sub-clusters of sites.[49] The polarization equations within each sub-cluster are solved directly via Cholesky decomposition. Polarization effects between the sub-clusters are captured iteratively through the contributions of the field in a JI-like fashion. Formally, DC-JI partitions $\tilde{\mathbf{Z}} = \mathbf{Z} + \mathbf{Y}$, but in this case $\mathbf{Z}$ is block diagonal matrix, rather than a diagonal one. Each block of $\mathbf{Z}$ corresponds to a sub-cluster of polarizable sites. These $\mathbf{Z}$ blocks include both the inverse of the polarizability tensors $\boldsymbol{\alpha}$ along the diagonal and dipole-dipole interaction tensors $\mathbf{T}$ between the atoms within the sub-cluster.

Exploiting the block structure of $\mathbf{Z}$, one can break Eq 3.7 into a separate matrix equation for each sub-cluster of polarizable sites,[49]

$$\mathbf{Z}_{II}\boldsymbol{\mu}_I = \mathbf{V}_{0,I} - \mathbf{Y}_{IJ}\boldsymbol{\mu}_J = \mathbf{V}_{0,I} + \sum_J \mathbf{T}_{IJ}\boldsymbol{\mu}_J \tag{3.10}$$

where I and J here refer to the different blocks of atoms. In other words, the induced dipoles on atoms in block I depend on the total field, which includes contributions from

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

Particle-Mesh Ewald

The original implementation of DC-JI was limited to non-periodic systems/direct space interactions.[49] For large systems, the evaluation of the electric field via multiplication of the matrix of the interaction tensors $\mathbf{T}$ and the vector of the multipole moments becomes cost prohibitive. Here, DC-JI is extended to large/periodic systems by combining it with the particle-mesh Ewald (PME) algorithm. PME partitions the total electric field $\mathbf{V}$ into short-range and long-range contributions,

$$\mathbf{V} = \mathbf{V}' + \tilde{\mathbf{V}} \tag{3.11}$$

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

The direct space contribution is computed by direct particle-particle interactions, contracting the interaction tensors $\mathbf{T}$ with the multipole moments. The reciprocal space contribution is defined such that it excludes unphysical self-interaction terms and short-range interactions. A smoothing function is applied to avoid discontinuities in the forces.

Further details on PME and computing $\tilde{\mathbf{V}}$ can be found elsewhere,[42, 3, 17] The specific details for how the reciprocal space contribution $\tilde{\mathbf{V}}$ is evaluated are not necessary for understanding the PME version of DC-JI here.

Within DC-JI, all interactions must either be captured directly on the left-hand side of Eq 3.10 or iteratively through the field on the right-hand side. Specific interaction tensor $\mathbf{T}$ elements in $\mathbf{Z}_{II}$ can be zeroed out on the left-hand side, and the corresponding contribution is then included in the field on the right-hand side. Optimal convergence with DC-JI will be achieved when the strongest interactions are captured directly within $\mathbf{Z}_{II}$ on the left-hand side. In the case of PME, the interactions come in two forms: the direct space interactions for which the interaction tensor is available, and reciprocal space ones for which the field is known but the interaction tensor is not readily available. Accordingly, the direct-space-only analog of the $\mathbf{Z}_{II}$ matrix, $\mathbf{Z}'_{II}$, is modified to include only the direct-space $\mathbf{T}'$ interaction tensor elements corresponding to the (typically strong) short-range interactions which are treated in direct space, while the longer-range direct-space interactions between blocks and all reciprocal space interactions are treated through the field on the right-hand side. The resulting DC-JI equations with PME are given by,

$$\mathbf{Z}'_{II}\boldsymbol{\mu}_I = \mathbf{V}'_{0,I} + \tilde{\mathbf{V}}_{0,I} + \tilde{\mathbf{W}}_I + \sum_J \mathbf{T}'_{IJ}\boldsymbol{\mu}_J \tag{3.12}$$

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

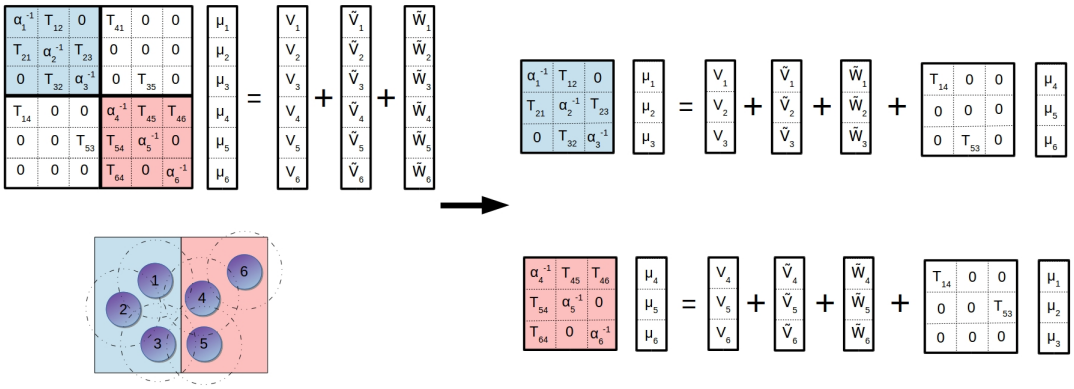


Figure 3.1: Representation of the polarization equations in DC-JI with PME for a system of six atoms. DC-JI reorganizes the matrix polarization equations on the left to obtain the set of coupled equations on the right that are solved iteratively for each block.

blocks J.

Figure 3.1 depicts these matrix equations graphically for a toy system of six atoms clustered into blue and pink groups. The dotted circles illustrate the radius of the direct space interactions around each atom. For simplicity of illustration, the interactions in this toy system are handled entirely in either direct or reciprocal space, neglecting the smoothing used to transition between the two. The response matrix $\mathbf{Z}'$ is illustrated in the upper left of Figure 3.1, with diagonal blocks corresponding to the two sub-clusters highlighted in blue and pink. Interactions that lie outside the direct space radius are zeroed out in $\mathbf{Z}'$. On the right hand side of the equation, one finds the contributions from the permanent direct space field $\mathbf{V}'$, the permanent reciprocal space field $\tilde{\mathbf{V}}$, and the induced reciprocal space field $\tilde{\mathbf{W}}$ (whose contributions include the terms zeroed out in $\mathbf{Z}'$). DC-JI then rearranges this matrix equation for the entire system into the set of smaller coupled matrix equations shown on the right half of Figure 3.1. Those coupled equations corresponding to Eq 3.12

are solved iteratively.

3.3 Software Implementation

Tinker-HP offers a platform for massively parallel simulations using polarizable force fields.[48] Double precision operations are used throughout. Distributed memory setup also allows for modeling large systems, with the largest simulation to-date including over 23 million atoms.[48] This is accomplished by using a 3D spatial decomposition of the chemical system which can then be distributed across thousands of processors. Tinker-HP contains two parallelization strategies for the polarization solvers that differ in how/when the bottleneck evaluation of the electric field is handled. The "sequential" scheme evaluates the reciprocal and direct space contributions sequentially, distributing the work for each over all available processor cores. Because the reciprocal space contributions do not parallelize as efficiently as the direct space ones, the "load-balancing" scheme partitions the processor cores into two groups: a larger fraction of cores which focus on the direct space, and a smaller fraction that perform the reciprocal space evaluations. The load-balancing scheme is potentially faster if the processors are appropriately partitioned such that the two components finish at the same time. In practice, however, the sequential approach proved slightly faster in our testing with DC-JI/DIIS and PCG (at least for the numbers of processor cores used here), so the remainder of this work discusses that implementation.

Because the serial DC-JI/DIIS algorithm without PME has been described in detail,[49] this section focuses on features specific to the new parallel PME implementation. While that earlier work examined both overlapping ("fuzzy") and non-overlapping sub-

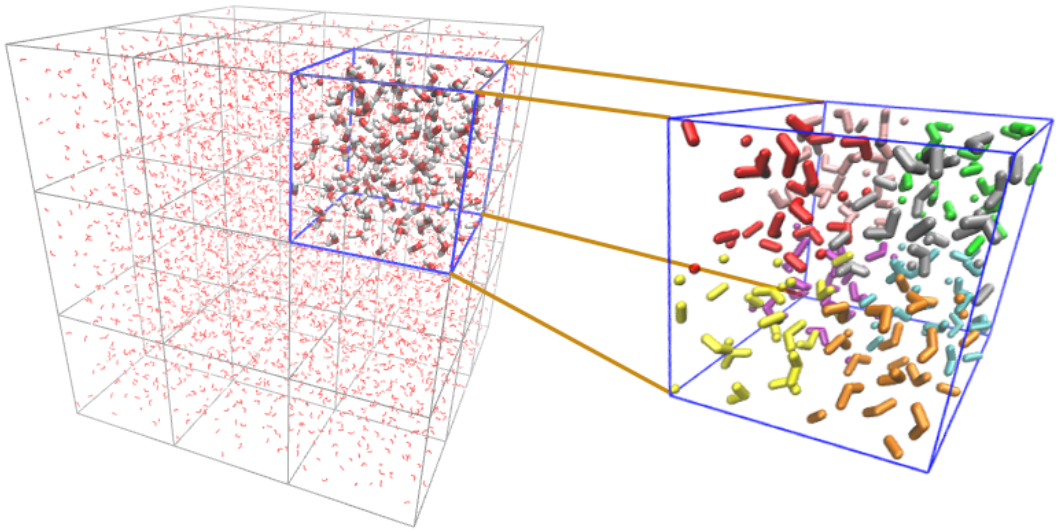


Figure 3.2: Cartoon of the partitioning scheme employed in the parallel DC-JI algorithm. First, Tinker-HP partitions the entire system into a set of domains, with one domain per processor (large boxes at left). For DC-JI, each domain is further partitioned into a sub-clusters according to the coloring in the enlarged domain at right. In practice, the sub-clusters average 60 atoms each.

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

The first step is to identify the sub-clusters of atoms that define the DC-JI block-

ing of $\mathbf{Z}'$. Previously, this was performed via K-means clustering of the entire system into a user-defined number of blocks.[49] For large systems, global K-means clustering could become expensive. Tinker-HP already implements a three-dimensional spatial domain decomposition,[48] dividing the system into one domain per processor (Figure 3.2), and each processor evaluates the forces and coordinate updates for the atoms in its domain. Calculations of interactions between two atoms in different domains are handled using the midpoint method, which assigns the interaction to the domain in which the midpoint of the two sites lies. These domains are defined before the software enters the polarization routines.

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

With the system partitioned into smaller domains, K-means clustering could be

performed efficiently within each domain. However, since the atoms have already been sorted into spatially local domains, we adopt a simpler, non-iterative approach for forming sub-clusters that divides the domain into the appropriate number of sub-systems, striving for similar volumes and roughly equal dimensions. The domain is partitioned by dividing it into evenly spaced portions along each coordinate axis such that the appropriate total number of blocks is obtained. Assuming uniform density of the atoms, this procedure will allocate similar numbers of atoms to each block. Dividing atoms from a single molecule over multiple sub-clusters does not cause any problems.[49] This scheme performs the clustering slightly faster than K-means (since it does not require the iterative atom-atom distance calculations needed in K-means), and the convergence of the DC-JI polarization equations is comparably good.

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

Once the DC-JI blocks are defined, remaining initialization steps involve constructing the permanent field (both direct and reciprocal space contributions), building the $\mathbf{Z}'_{II}$ blocks (and exploiting their symmetric nature), evaluating and storing the Cholesky decomposition of those blocks $\mathbf{Z}'_{II} = \mathbf{L}_I\mathbf{L}_I^T$ (for facile subsequent solution of the linear equations), and obtaining the initial guess induced dipole moments. The reciprocal space field is con-

structed in parallel as described elsewhere,[48] and the results are communicated between processors. In the first molecular dynamics time step, the guess dipoles are obtained as the polarizabilities contracted with the total permanent field, $\boldsymbol{\mu} = \boldsymbol{\alpha}\mathbf{V}_0$. In later time steps, guess dipoles are obtained from the previous steps (e.g. using the previous converged dipoles or Kolafa's always stable predictor-corrector algorithm[13]). Either way, the initial dipoles are communicated between processors.

Next, the iterative portion begins. The reciprocal contribution to the induced field $\tilde{\mathbf{W}}_I$ is constructed and communicated. The direct space contributions to the field for each block I from atoms in other blocks J are evaluated using a neighbors list. Once the total field (right-hand side of Eq 3.12) has been built, the induced dipoles in each block are solved via Cholesky decomposition (using the already factorized version of $\mathbf{Z}'_{II}$). In the first iteration, these dipoles are communicated and the algorithm proceeds to the next iteration. Starting at the end of the second iteration, DIIS extrapolation is performed before dipole communication. Each processor evaluates its contribution to the inner products of the residual vectors needed for DIIS[49] and communicates those. Then the processor extrapolates its induced dipoles and communicates them. The iterations continue until the root-mean-square change in the induced dipole moments is smaller than the user-selected convergence threshold. To summarize, each iteration requires communication of the reciprocal field contributions, the DIIS matrix updates, and the resulting induced dipole moments. All other work is done locally.

3.4 Computational Methods

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

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

All molecular dynamics simulations were performed in the microcanonical (NVE) ensemble with the reversible reference system propagator algorithm (RESPA) multi-time step integrator[20] using a 1 fs time step for non-bonded forces and 0.5 fs time step for the bonded forces. A range of thresholds for converging the polarization solvers were examined, and the 10^{-5} threshold provides satisfactory stability. Kolafa's always stable predictor-

corrector was used to obtain initial induced dipoles at the beginning of each time step.[13] All DC-JI/DIIS runs requested an average block size of 60 atoms. A 7 Å Ewald cutoff for PME and 9 Å cutoff for the van der Waals interactions were used.

Performance testing is carried out on three different protein systems in explicit aqueous solution: ubiquitin, dihydrofolate reductase (DHFR), and cyclooxygenase-2 (COX-2). These systems are representative of typical biological problems for which one might use polarizable force fields and span a wide range of system sizes. The smallest system, ubiquitin, has been extensively studied due to its prevalence in eukaryotic organisms. Here, it consists of the 1,227-atom protein surrounded by 2,835 waters (9,732 atoms total).[4] The ubiquitin system has unit cell dimensions of 54.99×41.91×41.91 Å and employed a 72×54×54 PME grid. The DHFR system was taken from the joint Amber/CHARM benchmark.[50] DHFR is necessary in the path to form purines and pyrimidines, which act as the building blocks for DNA and RNA. The system consists of 2,489 protein atoms in a box of 7,023 waters (23,558 atoms total). The DHFR system occupies a cubic box of dimension 62.23 Å, and a 64×64×64 PME grid was used. The COX-2 system was taken from the Tinker benchmark suite.[51] COX-2 is an enzyme that is a part of the rate limiting step in the formation of prostanoids. The system consist of the 17,742 protein atoms surrounded by 52,159 waters (174,219 atoms total). The COX-2 system has cubic box edge length of 120 Å, and a 128×128×128 PME grid was used. All three systems have been studied previously using Tinker-HP.[48]

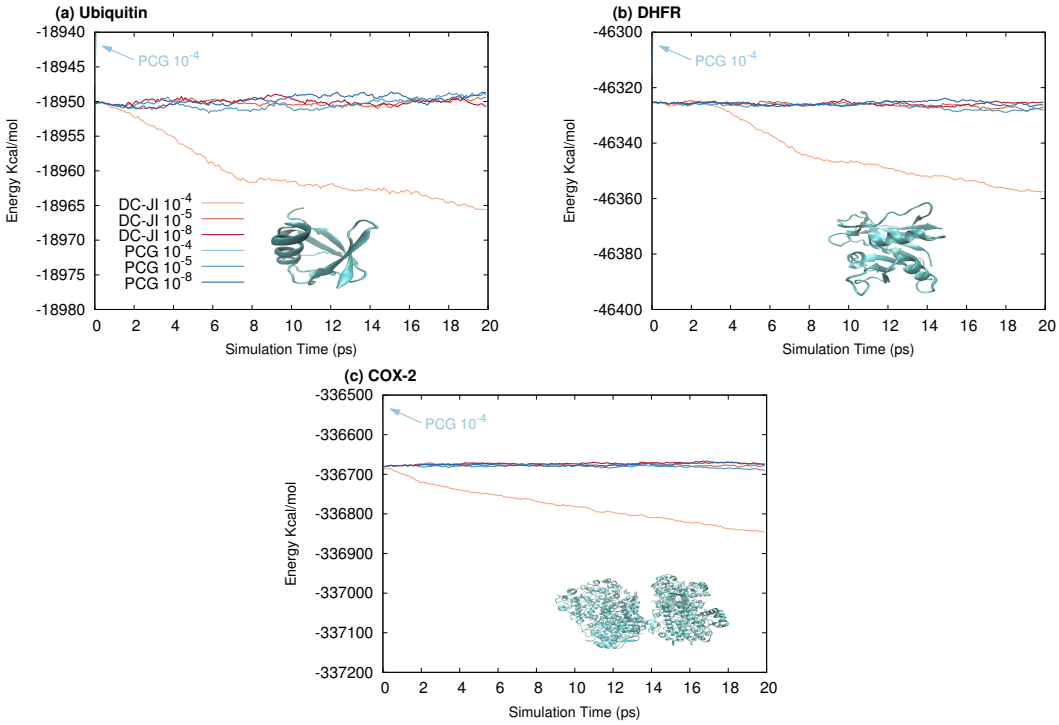


Figure 3.3: A 20 ps NVE simulation using a 1 fs time step and three different polarization convergence criteria for (a) Ubiquitin, (b) DHFR, and (c) COX-2. Note that PCG with a threshold of 10^{-4} D diverges out of frame within the first picosecond for all simulations.

3.5 Results And Discussion

3.5.1 Energy Convergence and Drift

Converging an iterative solver to a finite tolerance leaves residual error in the resulting induced dipoles. If the residual error is too large, it can cause energy drift as discussed in Section 3.2.1. Accordingly, we first investigate the energy conservation of DC-JI/DIIS and PCG with various convergence thresholds. The comparison focuses on PCG (as implemented in Tinker-HP) because of its widespread use and good numerical and computational performance.[3, 4, 48] Figure 3.4 plots the energies for 20 ps NVE simulations for all three protein systems with a 1 fs time step and three different polarization convergence

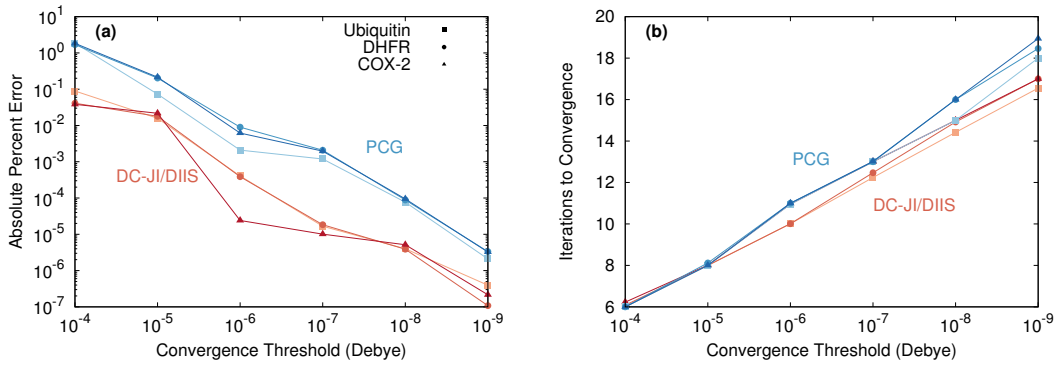


Figure 3.4: (a) Absolute error in the converged polarization energy relative to the DC-JI/DIIS results with a 10^{-10} Debye convergence threshold, and (b) the number of iterations necessary to achieve that convergence (without using dipoles from previous time steps). For a given convergence threshold, DC-JI/DIIS achieves more converged energy than PCG in the same number of iterations or fewer. Values were averaged over 100 configuration snapshots.

thresholds. The loose 10^{-4} Debye (D) convergence threshold leads to divergent behavior for both PCG and DC-JI/DIIS. However, whereas the PCG energy diverges out of the plot frame within a fraction of a picosecond, the DC-JI/DIIS energy drifts more slowly over the 20 ps simulation. This suggests that DC-JI/DIIS is effectively converging the equations more robustly. Tightening the convergence threshold by one order of magnitude to 10^{-5} D is sufficient to stabilize both the PCG and DC-JI/DIIS trajectories, with both algorithms exhibiting similar energy conservation. For both solvers, the energy conservation with a 10^{-5} D convergence criteria is similar to that from the much tighter 10^{-8} D criterion, with the the looser 10^{-5} D criterion requiring fewer iterations (typically 4) and less overall computational effort to converge.

To obtain further insight into the differences between DC-JI/DIIS and PCG, 100 configuration snapshots were extracted from a 10 ps NVE simulation of each protein at 100 fs intervals. Single-point energy evaluations were performed on each snapshot using

either PCG or DC-JI/DIIS and various convergence thresholds. Initial guess dipoles were obtained directly from the permanent field, without using any information from previous time steps. Figure 3.4a plots the absolute percent error in the polarization energy relative to the DC-JI/DIIS energy obtained with a 10^{-10} Debye convergence criterion, averaged over the 100 snapshots. The use of percent error normalizes across the different energy scales of the differently sized systems. For a given convergence threshold, DC-JI/DIIS consistently exhibits an energy error that is 1–2 orders of magnitude smaller than for PCG. Moreover, Figure 3.4b shows that DC-JI/DIIS achieves this better energy convergence in the same number of iterations as PCG or fewer.

The good numerical behavior of DIIS-JI/DIIS can be understood in terms of two factors. First, the direct solution of each block provides fully self-consistent polarization within the blocks at every iteration. Second, the DC-JI/DIIS convergence criterion definition is based on the root-mean-square change in the dipole vector, while the PCG implementation in Tinker-HP weights those dipole residuals by the preconditioner. For a given convergence threshold, the PCG criterion leads to slightly less-converged dipoles in practice. Altering the PCG implementation to use the same convergence criterion definition as DC-JI/DIIS causes PCG to require one additional iteration to converge. Doing so does improve the PCG energetics, but the percent error in energy remains $\sim$3–10 times larger than for DC-JI/DIIS, depending on the convergence threshold. It is possible that preconditioners more sophisticated than the diagonal one used in PCG here could alter this analysis.[3]

3.5.2 Computational Efficiency

Next, we examine the overall parallel performance of DC-JI/DIIS, JI/DIIS, and PCG. Figure 3.5 examines the total timings for one self-consistent polarization time step for each algorithm on 120 processor cores in the large COX-2 system. The results were averaged over 100 time steps taken sampled from a trajectory in which the initial dipoles were obtained via Kolafa's predictor-corrector guess. The vast majority of the computational effort is associated with construction of the permanent field (during initialization) and induced fields (at each iteration). The computational time required to evaluate the permanent electric field $\mathbf{V}$ and $\tilde{\mathbf{V}}$ is practically identical for PCG and JI/DIIS. Doing the same for DC-JI/DIIS takes slightly longer, since this step also includes building the $\mathbf{Z}'$ matrices for later use. All three algorithms use identical routines to build the induced reciprocal field $\tilde{\mathbf{W}}$ during the iterations, so the time required for that step varies only on the number of iterations required to converge the induced dipoles. Both DC-JI/DIIS and PCG required 4.0 iterations on average and therefore required virtually identical amounts of time to build $\tilde{\mathbf{W}}$, while JI/DIIS required more time due to its average of 5.3 iterations. Building the induced

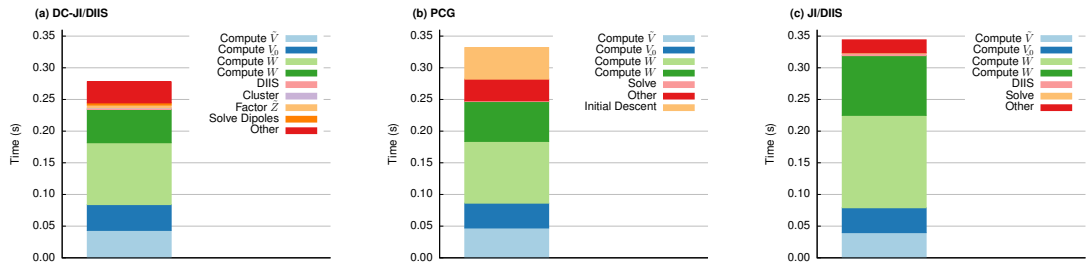


Figure 3.5: Timing breakdowns for the COX-2 system (174,219 atoms) averaged over 100 steps. A block size of ∼60 atoms was used for DC-JI/DIIS. Timings were performed using 120 cores on SDSC Comet. DC-JI/DIIS and PCG converged in 4.0 iterations on average, while JI/DIIS required an average 5.3 iterations.

direct space field $\mathbf{W}$ is fastest for DC-JI/DIIS, since a portion of the interactions are already computed and stored during the initialization as part of $\mathbf{Z}'$, whereas PCG recalculates those at each iteration. JI/DIIS requires even longer, since it both recomputes all the interaction contributions at each iteration and requires additional iterations to converge.

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

Finally, Figure 3.6 examines parallel timings for the polarization solver alone and for the total simulation (i.e. including evaluation of all bonded and non-bonded contributions) for the three proteins with DC-JI/DIIS, PCG, and JI/DIIS. For the DHFR and ubiquitin systems, timings are reported up to 480 cores. On 480 cores, ubiquitin and DHFR have an average of 20 and 49 atoms per domain, respectively. This means that each domain consists of just one small DC-JI block. Increasing the number of cores further would

decrease the DC-JI block size and lead to diminishing returns as the as the block size decreases. To utilize more cores, one would probably want to switch to the load-balancing parallelization strategy. Allocating some processors to work exclusively on the reciprocal space contributions translates to fewer, larger direct-space domains. For the much larger COX-2 system we test up to 720 cores, which corresponds to 242 atoms per domain, or about six DC-JI blocks per domain.

Figure 3.6 clearly demonstrates that solving the for the induced dipoles with DC-JI/DIIS is faster than using either PCG and JI/DIIS for all numbers of cores, and the computational savings increase with the number of cores employed. The analysis above indicates that DC-JI/DIIS is effectively one iteration faster than PCG in timings, or 20% faster at a 10^{-5} D convergence threshold. Indeed, for COX-2, DC/JI proves on average ∼20% faster than PCG, and ∼27% faster than JI/DIIS. Similar average speed-ups of 17–18% over PCG and 29-31% over JI/DIIS are observed for the two smaller proteins. Larger 30–40% acceleration was seen in our earlier work on non-periodic systems.[49] The DC-JI algorithm accelerates only the treatment of direct space interactions. In the non-periodic case, all interactions are treated in direct space. For periodic systems modeled with PME, the DC-JI algorithm only impacts the non-reciprocal space portions of the calculation, thereby reducing the possible savings. Differences in the PCG preconditioner between Tinker-HP and Tinker 8.1 might also play a role.

Unsurprisingly, the best parallel performance is observed for the largest systems. However, the fractional speed-up provided by DC-JI/DIIS over the other two polarization solvers is fairly consistent across different numbers of processor cores, typically increasing by

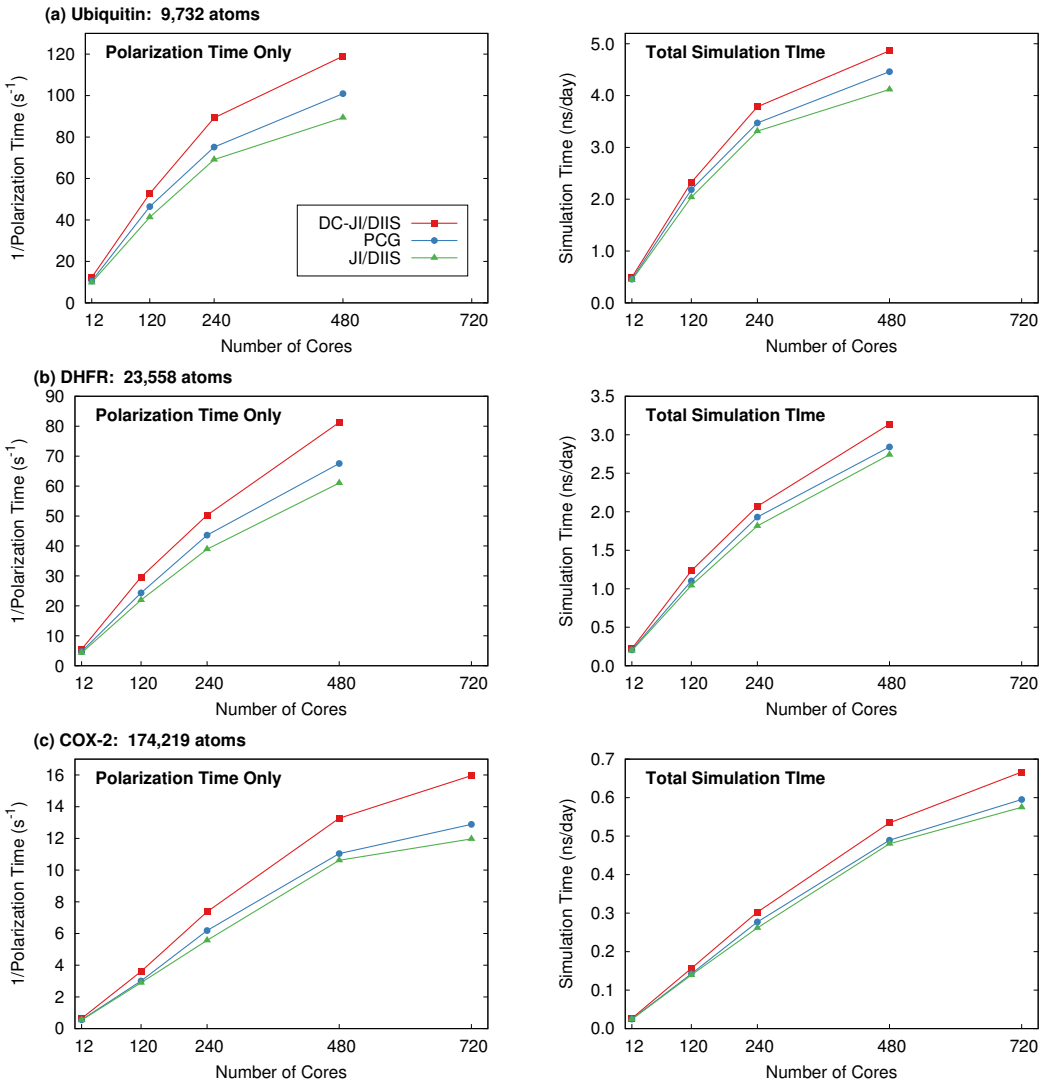


Figure 3.6: Performance of the DC-JI/DIIS, PCG, and JI/DIIS polarization solvers (left) and the corresponding total simulation time (right) for (a) Ubiquitin (b) DHFR, and (c) COX-2. The reciprocal of the polarization time is plotted for ease of comparison with the total simulation time.

a few percentage points as the number of cores is increased. For sufficiently large numbers of cores, however, the parallel efficiency of all three algorithms decreases, due in large part to the communication bottleneck associated with the reciprocal space terms. For COX-2, building the reciprocal space contributions to the permanent and induced fields in DC-

JI/DIIS consumes around 30% of the computational time on 12 cores, but this increases to over 60% of the computational time on 720 cores. Those computational costs are essentially the same across the three solvers, aside from any differences in the number of iterations required to reach convergence. Thanks to the minimal communication required, the direct-space portions of DC-JI/DIIS parallelize very efficiently. Perhaps the biggest limitation comes when the number of cores is sufficiently large that the number of atoms per domain falls below the optimal ∼40–80 atom block size. For very small blocks (∼10–20 atoms or fewer), fewer polarization interactions are solved for directly, and the number of iterations required to converge the dipoles begins to increase slowly as DC-JI/DIIS asymptotes toward the JI/DIIS model.[49]

Finally, switching focus from the polarization solvers to the total simulation time in Figure 3.6, we observe similar overall parallel performance and relative efficiencies of the algorithms, except that the total computational savings from DC-JI/DIIS are smaller. The non-polarization portions account for about half the total simulation time, and therefore the ∼20–30% acceleration in the polarization solver translates to a ∼10–15% increase in the number of nanoseconds of simulation time per day.

3.6 Conclusions

In summary, the DC-JI/DIIS polarization solver has been implemented for periodic systems via particle-mesh Ewald in the massively parallel Tinker-HP software package. DC-JI/DIIS solves the AMOEBA polarization equations up to 20% faster than PCG and 30% faster than JI/DIIS. Factoring in all other steps in the force field evaluation, DC-JI/DIIS

reduces the overall simulation time by ∼10–15%. Such performance is observed on chemical systems with tens or hundreds of thousands of atoms running on hundreds of processor cores. Furthermore, at a given convergence threshold, the polarization energies obtained from DC-JI/DIIS are closer to the exact solution than those from PCG. This translates to better numerical stability, as evidenced by the decreased energy drift in simulations with loosely-converged induced dipoles. Overall, given the combination of superior numerical behavior and decreased computational effort for DC-JI/DIIS over PCG and JI/DIIS, the new DC-JI solver can be recommended as an excellent alternative to traditional self-consistent polarization solvers.

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

Chapter 4

The Always Stable Predictor Corrector

4.1 Introduction

Consideration of force field polarization is necessary to capture the transport properties of ionic liquids[52] and to adequately describe protein structure.[53, 54] However, the inclusion of polarization significantly increases the computational cost of classical molecular dynamics (MD) simulations. For instance, solving the large system of linear equations to obtain the induced dipoles in the AMOEBA force field[8, 7, 9] accounts for about 50% of the computational cost of an MD simulation. In practice, this system of equations is too large to be solved exactly, and instead the solution is solved iteratively via a self consistent field (SCF) method.[3, 4, 17]

In these SCF methods, successive iterations generally converge the induced dipoles

toward their exact, mutually polarized values. The convergence thresholds for these SCF solvers must be chosen with care: When evaluating the polarization contributions to the nuclear forces, it is assumed that the iteratively determined induced dipoles have converged completely to the exact induced dipoles. Loose convergence of the induced dipoles can introduce instabilities in the simulation, such as problematic long-term energy conservation or deviations in physical properties.[13] On the other hand, converging the induced dipoles more tightly via additional SCF iterations can increase the computational costs appreciably. Strategies based on perturbation theory,[25, 26] truncated conjugate gradients,[30, 43] and extended-Lagrangian models[14, 44] have been developed to circumvent the computational costs of converging the induced dipoles tightly during the polarization procedure.

Alternatively, use of a history-based predictor to construct a good initial guess for the SCF solver can significantly reduce the iterations and computational cost required to reach convergence. A predictor can provide an efficient means of calculating the induced dipoles without introducing additional approximations. However, the use of induced dipoles from previous time steps destroys the time reversibility of the method.[15] A useful predictor should therefore exhibit an acceptable degree of time reversibility while substantially improving the starting point of the SCF method. In this letter, we focus on the predictor from the Always Stable Predictor-Corrector (ASPC) method.[12] We demonstrate how incorporating a longer history in this approach addresses stability concerns and/or reduces the computational cost of computing the induced dipoles by $\sim$20%.

4.2 Theory

The ASPC uses a history-based predictor for the induced dipoles,

$$\boldsymbol{\mu}^p(t+1) = \sum_{j=0}^{k+1} B_{j+1}\boldsymbol{\mu}(t-jh) \tag{4.1}$$

where $\boldsymbol{\mu}^p(t+1)$ is the predicted dipole, B_{j+1} are the scaling coefficients and $\boldsymbol{\mu}(t-jh)$ are the induced dipoles from previous time steps. The time step size is h and $k+2$ is the total number of values stored in history. The B_{j+1} scaling coefficients are derived such that the contributions that lead to time irreversibility error are chosen to be zero. In the original ASPC approach, the predicted induced dipoles are subsequently corrected by performing a single iteration of the SCF solver and then damping the resulting dipole update. The specific value of the damping coefficient is determined empirically, and its optimal value can potentially vary between systems and/or over the course of a simulation.

The Tinker software packages [55, 48, 47] (and possibly others) avoid this empirical damping parameter part of the corrector. Instead they employ the so-called "predicted iteration" method,[13] in which the predictor generates the initial guess for the induced dipoles, after which the SCF iterations are allowed to proceed until some user-defined convergence value is reached or a desired number of iterations has been performed. This predicted iteration variant of the ASPC is more accurate and obviates the need to determine the optimal damping parameter.[13]

Previously, the predictor coefficients were worked out and tested up to the 6-step predictor ($k = 4$), but the 4-step predictor was suggested as a compromise between accuracy

and memory storage.[13] The additional SCF iterations performed in the predicted iteration method mitigate accumulation of error that might arise from the use of only a single SCF iteration in the ASPC. These additional SCF iterations potentially change the calculus regarding the optimal number of prior steps to include in the predictor, since a longer history might lead to a better guess for the dipoles and therefore require fewer iterations to converge at the next time step. Whereas the current implementation of this method in the Tinker packages employs a 6-step predictor, in this letter we test up to the 16-step predictor in the Tinker-HP v1.1 package.[48] The necessary coefficients for these higher N-step predictors can be derived from the recursive expressions presented previously.[13] Using these expressions, we have derived the coefficients for up to a 25-step predictor, and these are included in the Supplementary Information.

Augmenting an existing implementation of the ASPC predictor to use higher N-step predictors is straightforward and adds little computational overhead. For a given system size, the memory requirements increase linearly with the number of induced dipole vectors stored in history. In a parallel implementation such as the one in Tinker-HP,[48] these historical induced dipoles can be distributed across nodes, since each processor only needs knowledge of the dipole elements handled by that processor. Regardless, the total memory requirements are insignificant even with global storage: the induced dipole history for a 100,000 atom system for the 16-step predictor requires only 38.4 MB of memory in double precision. Evaluating the predictor requires just scalar multiplication, so the computational cost is negligible relative to the cost of an SCF iteration, and it scales linearly with the number of prior steps included.

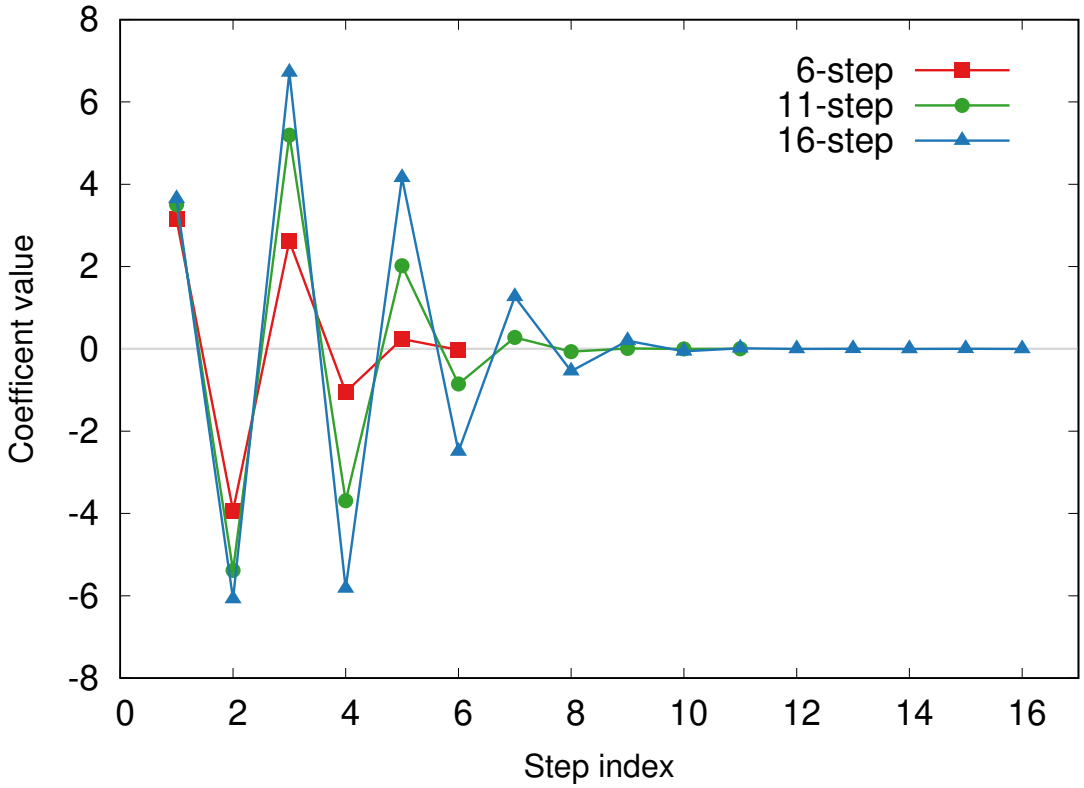


Figure 4.1: The predictor coefficients vs coefficient history index. The 6-step predictor (default) is compared to the higher N-step predictors.

In the ASPC predictor, the predicted induced dipole depends most strongly on the recent induced dipoles in history. In Figure 4.1 we see this trend holds for across a range of N-step predictors. In the 16-step predictor, for instance, the eight most recent history points account for 99.1% of the predicted induced dipole magnitude, while the oldest 8 history points contribute the remaining 0.9%. That means for a 1 fs time step, the 16-step predictor is dominated by contributions from the last 8 fs of simulation, which is shorter than the time period during which any substantial structural or conformational changes to the chemical system might occur.

The current work explores up to 16-step predictors, for which the B_{j+1} coefficients

span seven orders of magnitude. The coefficients for the 25-step predictor span twelve orders of magnitude. The decision to stop at the 16-step predictor here is somewhat arbitrary. The 16-step predictor provides significant computational benefits (as shown below) while avoiding the need to handle many tiny contributions that would arise from employing a longer history. To reduce round-off error, the predictor contributions are accumulated in quadruple precision before being reduced to double precision in the final predicted dipoles.

We test the different N-step predictors here with the two SCF polarization solvers: the widely used preconditioned conjugate gradients (PCG) solver[3] and our recently developed divide-and-conquer Jacobi iterations accelerated with direct inversion in the iterative subspace (DC-JI/DIIS) solver. We have previously demonstrated the superior speed and stability of DC-JI/DIIS relative to PCG.[49, 56] DC-JI/DIIS is used here unless otherwise specifically noted. Typically one iterates the SCF equations until a user-chosen convergence criterion is met. However, given the small numbers of iterations typically required to meet commonly used convergence criteria, even a change of one iteration arising from slightly different initial guesses can substantially alter how tightly converged the induced dipoles are. That in turn would obscure the stability behavior resulting from the predictor. To ensure an even-handed comparison of stability across the different solvers and predictors, all results here employ a fixed number of SCF iterations in the polarization solver. Testing indicates that the stability improvements reported here for the longer-history predictors also occur with more traditional threshold-based convergence criteria.

4.3 Computational Methods

Stability of the predicted iteration approach is assessed here in terms of energy conservation in an NVE ensemble. The method also performs well for NVT ensembles, though using a thermostat obscures differences between the different predictors. Testing was done on a 500-molecule water box[57] and on the ubiquitin system.[4] The 9,737-atom ubiquitin system consists of the 1,227-atom protein surrounded by 2,835 waters. All simulations were performed with the reversible reference system propagator algorithm (RESPA) multi-step integrator[20] using a 1 fs time step for non-bonded forces and 0.5 fs time step for the bonded forces. A direct space cutoff of 7 Å was employed for the particle-mesh Ewald treatment of long-range interactions. Energy drift was typically measured via linear regression of the energies over 1 ns of simulation time. For more tightly converged cases with less energy drift, 5–10 ns of simulation were used. Empirical testing indicates that these simulation lengths are sufficient to provide converged regression slopes (energy drift).

4.4 Results and Discussion

Figure 4.2 plots the energy conservation from NVE simulations on $(H_2O)_{500}$ with different N-step predictors. The tightly converged reference simulation (i.e. 20 DC-JI/DIIS iterations, starting from initial guess dipoles equal to the polarizability times the permanent electric field) converges the dipoles to a root-mean-square change of $\sim 10^{-13}$ Debye, and it exhibits negligible drift ($<10^{-5}$ kcal mol^{-1} ns^{-1} atom^{-1}). In contrast, the N=6 predictor with three SCF iterations per time step drifts by -0.042 kcal mol^{-1} ns^{-1} atom^{-1}. Increasing

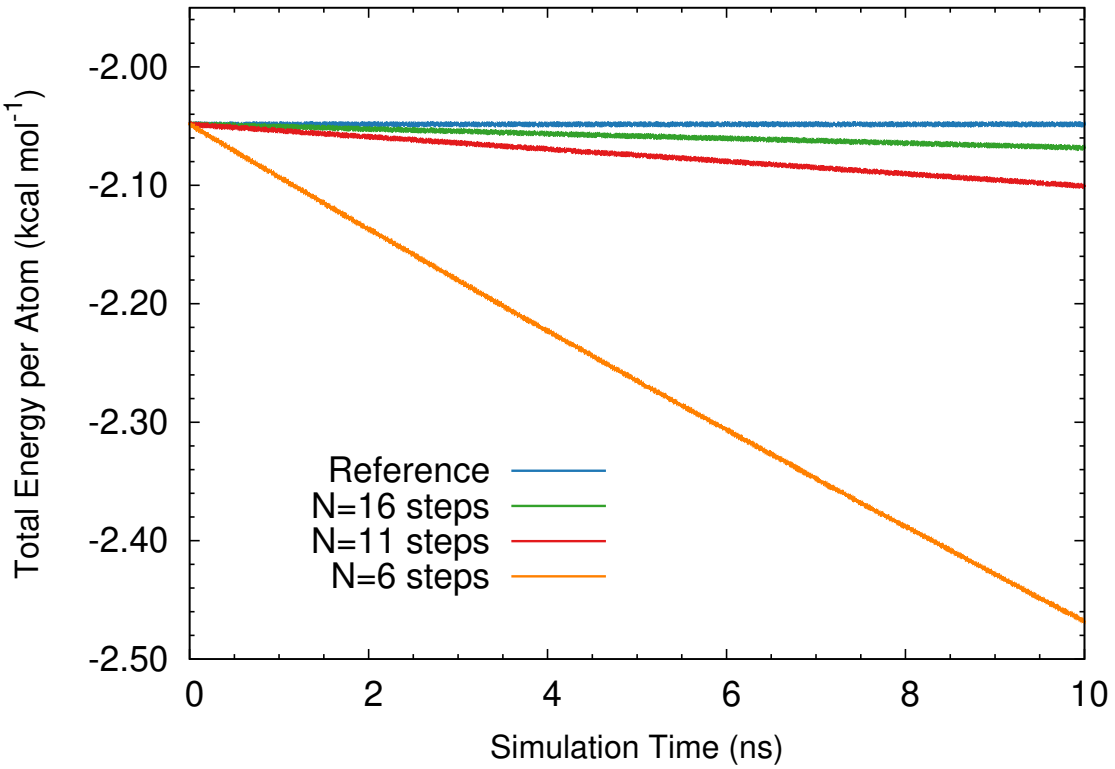


Figure 4.2: Comparison of the energy conservation for NVE simulations on $(H_2O)_{500}$ with different N-step predictors and three SCF iterations at each time step.

the length of the history employed in the predictor reduces the drift considerably. Despite taking only three SCF iterations per time step, the N=16 predictor case drifts by only -0.002 kcal mol^{-1} ns^{-1} atom^{-1} over the 10 ns trajectory.

For a broader perspective, Figure 4.3 plots the drift per nanosecond in the $(H_2O)_{500}$ box as a function of the number of steps included in the predictor and the number of SCF iterations. Each data point in this plot corresponds to a drift rate extracted from linear regression of an NVE simulation under those conditions. Independent of the number of SCF iterations, increasing the history from the 6-step to the 16-step predictor decreases the energy drift rate by an order of magnitude. Moreover, the use of the 16-step predictor

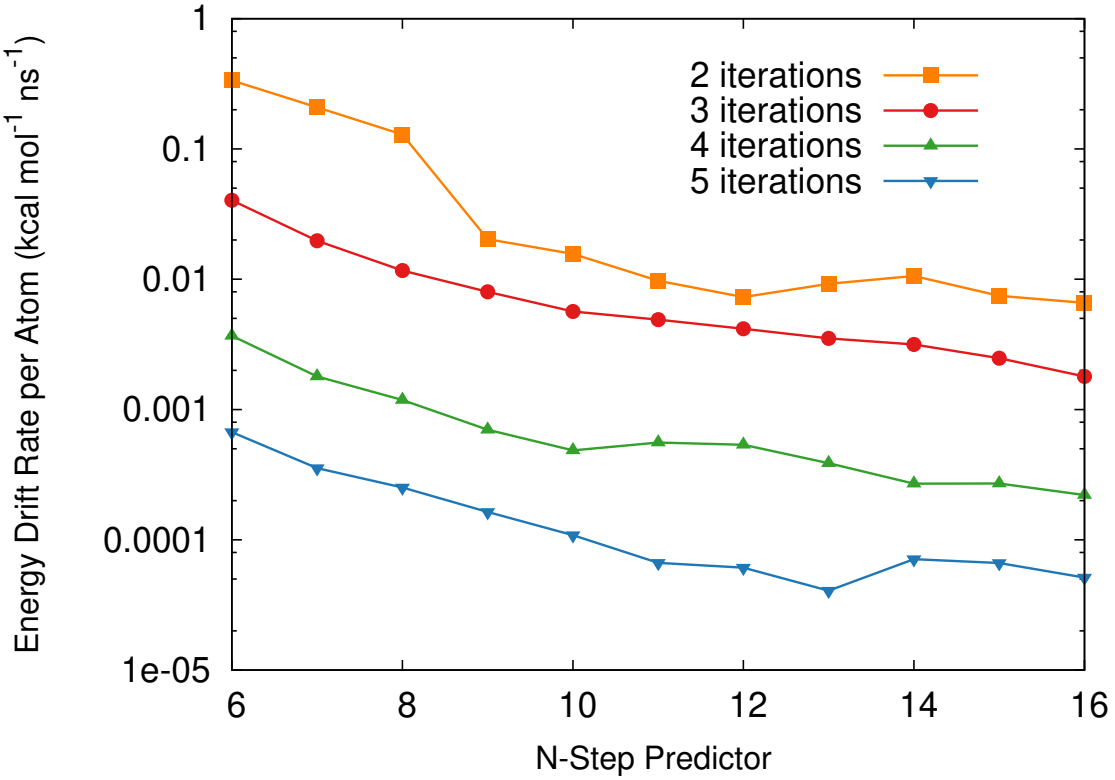


Figure 4.3: Energy drift rate (kcal mol^{-1} ns^{-1}) per atom for NVE simulations on $(H_2O)_{500}$ for the 6-step to 16-step predictor with differing numbers of SCF iterations at each time step. A tightly converged reference simulation with no predictor exhibits energy drift less than 10^{-5} kcal mol^{-1} ns^{-1} per atom.

consistently improves the energy stability by an amount comparable to what one would obtain by performing an additional SCF iteration with the 6-step predictor. From that perspective, the better predictor can be used to accelerate the evaluation of the induced dipoles without increasing energy drift. For example, DC-JI/DIIS generally requires four SCF iterations to converge to a 10^{-5} Debye threshold.[56] With the 16-step predictor, comparable energy conservation can be obtained at the cost of only three SCF iterations, or a computational savings of ∼20%. The performance here is not unique to water, either. Similar energy drift behavior is observed for the ubiquitin system as well (see Supporting

Information).

It is interesting to compare the present approach with other recently developed strategies for accelerating polarizable force field simulations. For example, in the 16-step predicted iteration method with three iterations, the energy drift rate is only 0.002 kcal mol^{-1} ns^{-1} $atom^{-1}$. For comparison, a thermostatted extended-Lagrangian approach employing the same number of SCF iterations exhibited a somewhat larger energy drift of $\sim$0.009 kcal mol^{-1} ns^{-1} $atom^{-1}$ for a similar water box.[14] The approximate OPT3 perturbative polarization solver[26] also effectively utilizes three SCF iterations, but it requires several empirically fitted parameters to achieve good accuracy. Furthermore, the large N-step predictor approach here is probably as fast or faster than the truncated conjugate gradient approximate solvers (at least for 1 fs time steps), since those effectively employ 2-3 PCG iterations and have more expensive analytic gradients.[43] A direct performance comparison among these different approaches over a range of simulation scenarios would be a valuable future work.

To assess the role of the polarization solver, Figure 4.4 compares the behavior of the higher N-step predictors for the DC-JI/DIIS and PCG solvers. Both SCF methods benefit from employing the higher N-step predictors. However, DC-JI/DIIS exhibits less drift relative to PCG for all simulations. Surprisingly, in the case of the PCG solver using two iterations, using higher N-step predictors does not decrease the energy drift. Perhaps two iterations of the PCG solver is insufficient to nullify the accumulation of error in the higher N-step predictors. Regardless, this odd behavior is not observed for the more robust DC-JI/DIIS solver.

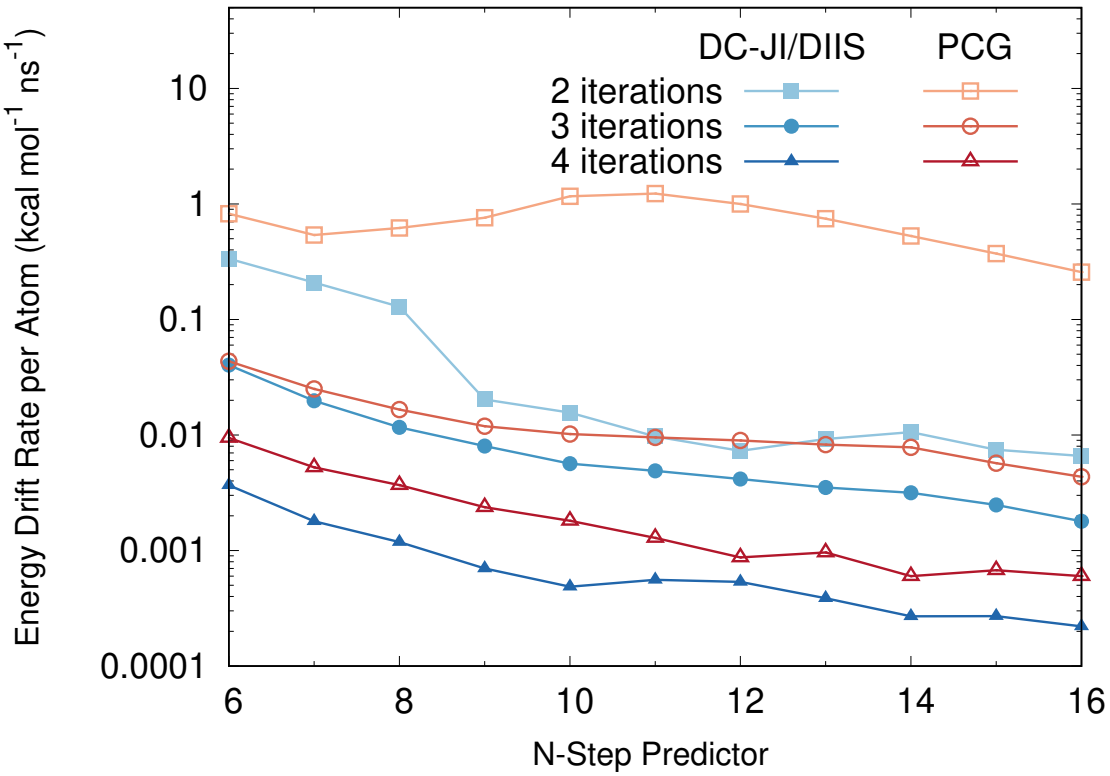


Figure 4.4: Comparison of different higher N-step predictors with the DC-JI/DIIS and PCG polarization solvers.

Finally, to understand why the longer N-step predictors perform better than shorter-history ones, Figure 4.5 shows the distribution of dipole errors in the initial guess (μ_0) and after each successive iteration relative to a tightly converged (20 SCF iterations) reference set. With both 6-step and 16-step predictors, the initial guess dipoles have errors around 10^{-4} D, but the root-mean-square (rms) errors for the 16-step case are about $\sim$20% smaller. The errors in the induced dipoles decrease several fold with each SCF iteration, but the dipoles from the 16-step predictor consistently maintain $\sim$20% higher accuracy. These small accuracy improvements in the induced dipoles are sufficient to increase the stability of the simulations signifcantly.

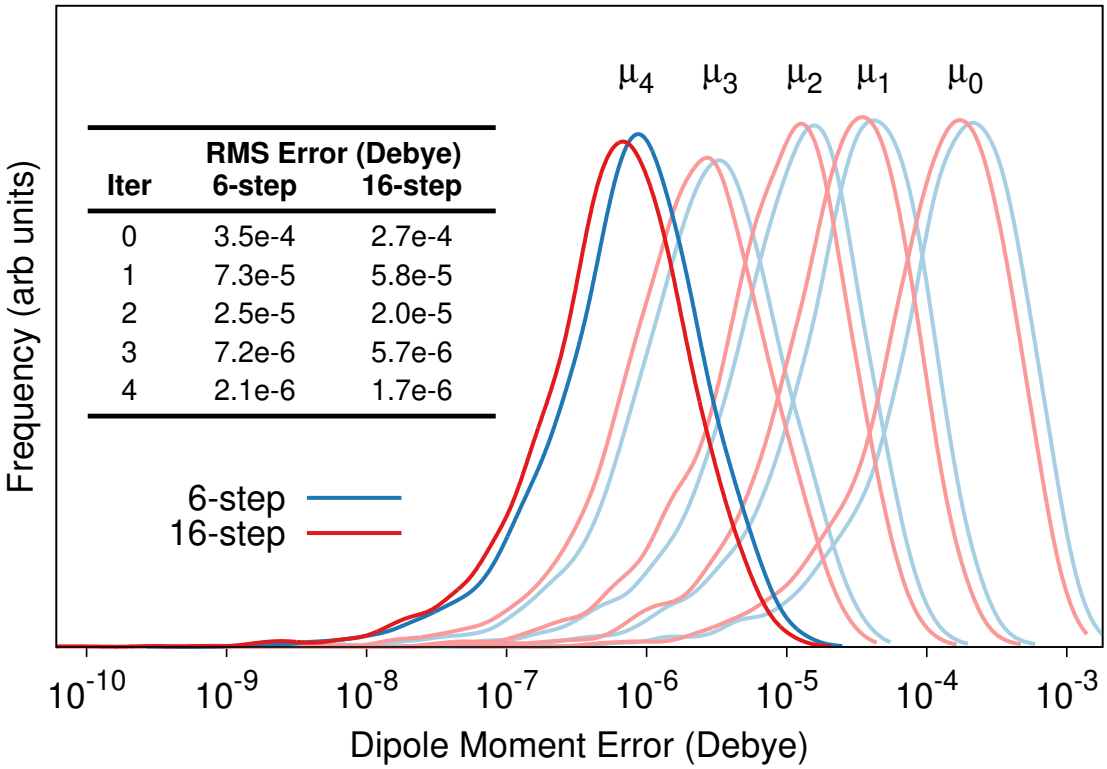


Figure 4.5: Distributions of errors in the induced dipoles from the initial guess ($\boldsymbol{\mu}_0$) and first four SCF iterations relative to tightly converged dipoles using either the 6-step or 16-step predictor.

4.5 Conclusions

In conclusion, we have demonstrated that use of a longer history in the "predicted iteration" variant of the ASPC provides substantial computational benefits in the context of the AMOEBA force field. Increasing the history from 6 to 16 steps requires only minor software modifications and adds little computational overhead, yet it reduces the rate of energy drift by an order of magnitude. Alternatively, one can employ this strategy to reduce the number of SCF iterations and accelerate the calculation of force field polarization by ∼20%. The ability to achieve acceptable energy conservation with only three SCF

iterations makes the combination of the 16-step predictor and DC-JI/DIIS polarization solver competitive with other approximate and extended-Lagrangian schemes for handling the induced dipoles. The extended predictor should prove useful for other polarizable force fields in addition to AMOEBA, and perhaps it could also be adapted for *ab initio* molecular dynamics simulations that employ the ASPC.[58]

Chapter 5

Average Condensed Phase Environment

5.1 Introduction

Combined quantum mechanical/molecular mechanics (QM/MM) calculations provide an effective route toward modeling complex systems in the condensed phase with much less computational effort than fully QM simulations. Nevertheless, modeling condensed phase systems, such as a molecule in solution, remains challenging due to the need to perform the QM/MM calculations on large numbers of sampled configurations. QM/MM configuration sampling can be performed directly with QM/MM dynamics, or indirectly by first sampling with MM and subsequently performing QM/MM calculations on configurations extracted from the MM ensemble. Even in the latter approach, repeated QM calculations are needed for each change in the MM environment. Strategies that reduce

the computational effort associated with evaluating the response of the QM region to the environment are therefore important.

One might circumvent the need to perform a new QM calculation for each new configuration by either approximating or pre-computing the response of the QM solute to the solvent using point-charge or more elaborate representations of the solute.[59, 60, 61, 62, 63, 64, 65, 66, 67] More recently, Sodt et al proposed a family of multiple environment, single system (MESS) models which use an efficient correction to update the QM energy and orbitals/density in response to a change in the MM environment.[68] For example, the Hessian (H) based MESS-H variant estimates the QM/MM energy in a new solvent configuration from the energy of a previous configuration based on a single Newton-Raphson step update of the Kohn-Sham orbitals. The approximate orbital Hessian used in the Newton-Raphson step only needs to be computed once, and it can be re-used for each new configuration of the environment.

Polarizable continuum models (PCMs) lie at the opposite extreme. Rather than explicitly sampling configurations of the environment, PCMs represent the environment as a bulk dielectric medium.[69, 70, 71] Polarizable continuum models often do an excellent job of capturing bulk solution behaviors, but they perform more poorly when specific, local solute-solvent interactions are important. For example, the Cope elimination reaction rate accelerates a million-fold upon switching from a protic to aprotic solvents.[72, 73, 74] Hydrogen bonding between the solute and protic solvent molecules preferentially stabilizes the reactant, effectively increasing the activation barrier and slowing the reaction rate. A PCM will not capture this effect without inclusion of explicit solvent molecules.[75] Contin-

uum models also have difficulty describing inhomogeneous environments for which a bulk dielectric is ill-defined. The effective dielectric constant of a protein has been frequently debated, for instance.[76, 77, 78, 79]

An interesting family of methods lies in between explicit QM/MM evaluation of sampled configurations and polarizable continuum models. In these methods, one embeds the QM calculation in an averaged or effective representation of the environment. As in a PCM approach, replacing hundreds or more QM/MM calculations with a single calculation in an averaged environment reaps massive computational savings. At the same time, constructing the averaged environment from explicit configurations can retain essential features that might otherwise be lost in a bulk continuum approximation. These methods do assume that the response of a system to its averaged environment is consistent with taking the average over many individual responses of the system to different instantaneous environments. Though there may be situations where this approximation does not behave well, it often appears to be a useful approximation.

One such model, the averaged solvent electrostatic potential (ASEP) model developed by Aguilar and co-workers, embeds a solute monomer in a field of point charges fitted to reproduce the average electrostatic potential felt on the solute due to the environment.[80, 81, 82, 83] Another model, the three-dimensional reference site interaction model (3D-RISM) approach,[84, 85] allows the computationally efficient evaluation of solvent density distributions and thermodynamic parameters without requiring explicit solvent simulations, and RISM approaches can be combined with QM simulations to study solvation effects.[86, 87, 85]

Previously, we presented a mean-field model that employs a mathematically rigorous coarse graining of the environment.[88] This coarse-grained (CG) model constructed a radial grid of CG points about the solute, and then averaged the effective force field parameters at these grid points over space and time. The coarse graining relied on formally exact spherical harmonic translation formula to translate the MM parameters from their explicit atomic sites to these CG grid points. These translations are analogous to the ones used in the fast multipole method,[89] for example, except in this case they were applied to multipoles (electrostatics), polarizabilities (induction), and frequency-dependent polarizabilities (van der Waals dispersion). The resulting translated MM parameters at each CG grid point are summed and then averaged over the ensemble of configurations. The use of a grid of effective polarizable multipoles to represent the solvent is also akin to the Langevin dipole solvation model.[90, 90, 91, 92, 93] In the Langevin dipole model, the magnitude and orientation of the dipole at each grid point is optimized simultaneously with the wave function of the solute. Whereas the Langevin dipoles model has primarily been parameterized for aqueous solution, our CG approach can be applied to an arbitrary molecular environment more readily.

Here, we extend that earlier CG model in two key ways. First, we improve the manner in which the coarse-grained points are chosen. The previous grid approach proved too sensitive to the specific grid-point locations. Instead of enforcing a regular grid, the current work employs clustering algorithms to place CG site locations "naturally" based on the explicit locations of atoms/molecules in the sampled configurations. Second, we improved the physical behavior of the coarse-grained polarization model. In particular, to

retain distinctions in the polarizabilities of different atoms or molecules, the coarse-graining is now performed separately over each unique atom type. Furthermore, the atomistic model employed here would typically include only intermolecular polarization, since intramolecular polarization is already accounted for in the multipolar expansion. However, the earlier CG approach lost the distinction between inter- and intramolecular polarization. To regain some of that distinction in the coarse-grained representation, nearby CG sites are now clustered and polarization occurs only between clusters, rather than within them. These clusters loosely correspond to the dynamic region inhabited by a given solvent molecule during the simulation.

We examine the performance of the refined averaged condensed phase environment (ACPE) model by computing excitation energies of small organic molecules in solution. We demonstrate that the ACPE model maintains important features of the underlying solvent structure and that it can be used to describe inhomogeneous features in complex environments that would be difficult to describe with a conventional, homogeneous polarizable continuum model. At the same time, the predicted ACPE excitation energies in the averaged environment agree very well with those from a more traditional QM/MM average over many configurational snapshots. Importantly, the predicted excitation energies prove fairly robust to variations in the specific CG sites generated by the ACPE model.

5.2 Theory

As illustrated by the flow chart in Figure 5.1, the ACPE procedure consists of six main steps:

1. Sample the configurational space of the system.

2. Superimpose the atomic coordinates of sampled configurations.

3. Generate a set of CG sites via the K-means++ algorithm

4. Obtain MM parameters for each molecule via interpolation.

5. Translate like atoms' MM parameters to the nearest CG site.

6. Group CG sites into CG clusters via a second round of K-means++.

Step 1 uses standard molecular dynamics, Monte Carlo, or related techniques to sample the configurations of the environment. In all cases here, the molecule of interest (the "solute") is frozen at a fixed geometry during the configurational sampling to allow straightforward superposition of the sampled environment ("solvent") configurations in Step 2. This approximation has been used in the MESS[68] and ASEP models[94] as well. In principle, one could repeat steps 1–6 for various solute geometries if solute dynamics are also important.[94, 95] Step 2 involves merging the atomic coordinate files for the molecules in the environment over all sampled configurations into a single list. Step 4 assumes that the force field parameters can vary as a function of the specific molecular geometries in the environment. Here, we vary the water force field parameters with intramolecular geometry. If the MM parameters are constant, Step 4 can be skipped. Steps 3–6 are described in more detail below.

5.2.1 Determination of coarse graining sites

Step 3 in the ACPE procedure automatically determines the locations of the CG sites in the environment region based on the atomic coordinates in sampled configurations

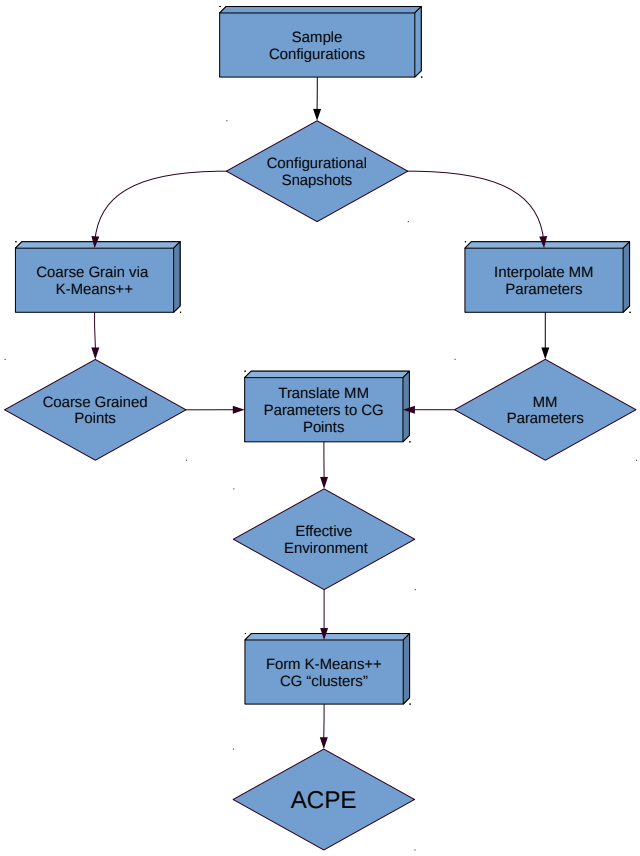


Figure 5.1: Flow-chart outlining the steps involved in the ACPE model.

via K-means clustering. K-means is widely used in machine learning, pattern recognition and data-mining.[96] In general, K-means clusters a data set of n points into k groupings. Here we adapt K-means to automatically cluster n molecules into k coarse-grained sites.

Distinct sets of CG sites are determined here for each symmetrically unique atom type present in the environment. For an aqueous environment, for instance, K-means coarse-graining is applied separately for the solvent oxygen and hydrogen atoms. This preserves the physical behaviors of different atom types and retains aspects of molecular structure within

the CG model. The number of CG sites k is chosen as the average number of atoms per configuration of the type being coarse-grained times a user-selected scaling factor. Based on empirical testing, scaling factors of 1–10 for light atoms and 5–40 for heavier atoms work well in the examples considered here, as will be discussed further in Section 5.4.1. Further study is needed to develop a more universal algorithm for choosing the scaling factor. Although this "coarse-graining" utilizes many more sites than are present in a single configuration of the initial system, it contains orders of magnitude fewer sites than the total number of sites found across the hundreds (or more) configurations being averaged over. In other words, this approach corresponds to coarse-graining over space and time (configurational snapshots) simultaneously, rather than simply spatial coarse-graining.

The K-means algorithm seeks to identify the set of CG sites which minimizes the sum of the distances between the atomic sites and their nearest CG site. Specifically, it assigns each atom in the atomistic picture to a CG site and seeks to minimize the objective function:

$$\underset{S}{argmin} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 \tag{5.1}$$

where S is the set of all atoms, x is the position of a single atom in S, S_i is the set of atoms associated with the i-th CG site, and μ_i is the centroid (position) of the i-th CG site. Finding the globally optimal solution to the problem of clustering n points into k groups scales as $O(n^{(d+2)k+1})$, where d is the dimensionality of each point (three for Cartesian coordinates).[97] The K-means algorithm iteratively searches for the solution to this problem in $O(nkdi)$ effort, where i is the number of iterations needed to converge. Inclusion of a neighbors list (described below) can be used to effectively eliminate k from

the scaling. Because the K-means algorithm used here relies on Euclidean distance between the atoms and CG sites, the individual atoms clustered into a single CG site tend to be roughly spherical.

In this paper we employ a variant of the K-means algorithm known as K-means++, which differs from K-means only in the initialization step. In traditional K-means, a poor initialization of the CG sites can lead to poor clustering. K-means++ addresses this problem by defining an initialization procedure that is biased toward evenly-distributed CG sites.[98] The K-means++ initialization typically leads to faster convergence of the algorithm and more optimal solutions.[98] The K-means++ initialization step is performed as follows:

1. Select one atom uniformly at random to be a CG site.

2. Compute $d(A)$, the square of the distance between each atom and its nearest centroid.

3. Weight the probability that each remaining atom A will be chosen as the next CG site by $\frac{d(A)}{\sum_{A=1}^{n} d(A)}$.

4. Choose the next CG site at random and repeat steps 2–4 until k CG sites have been initialized.

Step 3 increases the probability that the initial guess CG sites will be well-separated throughout the environment. Note that while this K-means++ guess initializes CG sites on individual atoms, the final CG sites obtained upon converging the K-means clustering algorithm are not constrained and can lie anywhere in space.

A neighbors list was implemented to reduce the number of distance calculations required during the K-means clustering process. The neighbors list defines a sphere of

inclusion around every atom, and only CG sites that lie within this sphere are considered in the clustering of that atom. This is similar to a Verlet neighbors list[99] used in molecular mechanics calculations to keep a list of all neighboring particles for which interactions will be calculated. Assuming uniformly distributed CG sites, the neighbors list reduces the number of distance calculations per atom from k to $\rho\pi r^3$, where r is the radius of the neighbors list and ρ is the density of centroids. For the examples considered in this paper, using the neighbors list accelerates the K-means algorithm by an order of magnitude, but this improvement is ultimately dependent on k and the size of the system. With the neighbors list included, the K-means algorithm runs as follows:

1. Determine neighbors list of CG sites lying within 3 Å of each atom.
2. Compute distance squared $\|x - \mu_i\|^2$ between each atom x and the CG sites μ_i associated with x in the neighbors list.
3. Assign each atom to its nearest CG site.
4. Compute new CG site positions as the mean of the positions of atoms assigned to it.
5. Calculate the sum of the absolute change in positions of the CG sites from their previous position.
6. Recompute the neighbors list if the sum from step 5 has reached a defined threshold.
7. Repeat steps 2-6 until the sum from step 5 equals zero.

For step 6, the neighbors list is updated if the average change in position of the CG sites exceeds half the radius of the sphere in Step 1 (i.e. 1.5 Å). This K-means clustering is applied to the complete set of superimposed atomic configurations.

5.2.2 Force Field Parameter Interpolation

Once the set of k CG sites has been determined via K-means++, parameters are needed to describe the interactions between the system and its coarse-grained environment. That requires obtaining force field parameters for the species in the original atomistic representation of the environment (e.g. for each solvent molecule in each sampled configuration) and then mapping those parameters onto the coarse-grained representation (Section 5.2.3).

The success of any embedding treatment is dependent on the manner in which the environment is modeled. The molecules in the environment here are modeled using a polarizable *ab initio* Force Field (AIFF) which has been demonstrated to perform well for describing long-range and many-body interactions.[100, 101, 102, 103] The AIFF is parameterized in terms of atom-centered distributed multipoles (electrostatics),[104, 105, 106] distributed polarizabilities (polarization),[107, 108, 109] and distributed frequency-dependent polarizabilities (van der Waals dispersion).[110] In the examples studied here, we perform electrostatic embedding only, so the dispersion contributions are ignored. Dispersion contributions were included in our earlier coarse-graining work, though.[88]

The AIFF parameters are typically calculated on the fly from density functional theory (DFT) for each molecule in its current geometry. Using CamCASP,[111] calculating these parameters for a water molecule typically takes a few minutes. Performing such calculations over hundreds of solvent molecules in hundreds of configurations quickly becomes computationally demanding. Instead, we pre-computed the water parameters at 20 water bond angles and 20 bond lengths for each O-H bond (i.e. 8,000 geometries total). The force field parameters (distributed multipoles and polarizabilities) can then be interpolated from

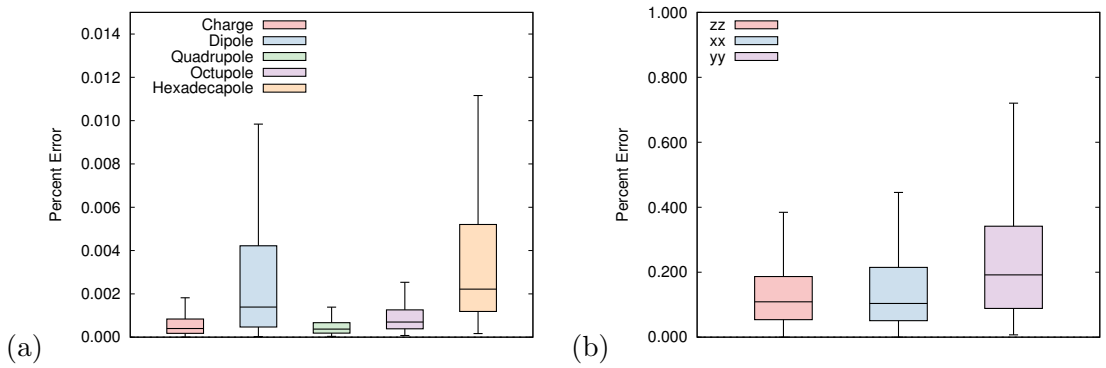


Figure 5.2: Box-plot distributions of absolute percent errors in the (a) multipole moments and (b) polarizabilities between directly computed and the interpolated AIFF parameters for 400 water geometries. The boxes indicate the median error (center line) and the central 50% of the data, while the whiskers indicate the largest errors.

this grid of configurations using a "natural" cubic spline.

The spherical tensor multipole and polarizability force field parameter data on the interpolation grid is stored in a local coordinate frame. Mapping the interpolated force field parameters onto each individual molecule in the environment requires rotating the local-frame parameters into the global coordinate system. Rotations of the multipole moments and polarizabilities are performed using explicit expressions.[112] The rotation matrix elements are expressed as polynomials of degree $\leq l$, where l is the rank of moment being rotated, in terms of the elements of the 3×3 rotation matrix. Rotation matrices for up to $l = 4$ (hexadecapole) are tabulated in the Supporting Information.

Overall, interpolation of the force field parameters provides excellent accuracy at a tiny fraction of the computational cost of computing the parameters directly. Figure 5.2 presents the errors in the multipole moments and polarizabilities arising from the interpolation for 400 solvent water conformations taken from an MD simulation. Errors in the multipoles are typically well below 0.01%, while errors in the polarizabilities are no more

than a few tenths of a percent. At the same time, this interpolation procedure reduces the computational cost of obtaining AIFF parameters dramatically. In a test job containing 1000 solvent configurations of 1600 water molecules (i.e. 1.6 million waters total), interpolation lowers the cost of generating the force field parameters from 9.1 years (at ∼3 min each) to only 9.7 hours of CPU time. Each interpolation is independent of the others, so it can be performed in highly parallel fashion if desired.

5.2.3 Force Field Parameter Translation

Once the set of CG sites have been determined via the K-means++ algorithm (Section 5.2.1) and force field parameters have been obtained for each molecule in the original explicit representation of the environment (Section 5.2.2), we then translate the force field parameters for each atom to its associated CG site. Parameters are summed at each CG site and divided by N, the number of configurations, to obtain average values. As described previously,[88] the translation exploits the fact that a multipole moment $Q_{l'k'}$ at a given point in space O can be exactly represented as a linear combination of multipoles Q_{lk} at another point C. [113] The functional form for the translation of the moments is.

$$Q_{lk}^{C} = \sum_{l'=0}^{l} \sum_{k'=-l'}^{l'} \left[\binom{l+k}{l'+k'} \binom{l-k}{l'-k'} \right]^{\frac{1}{2}} Q_{l'k'}^{O} R_{l-l',k-k'}(-c) \tag{5.2}$$

where the Q_{lk}^{C} are the multipole moments at the final position, the terms in curved brackets are binomial coefficients, $Q_{l'k'}^{O}$ are the multipole moments at the initial location and $R_{l-l',k-k'}(-c)$ is a regular spherical harmonic. If k is not equal to zero the resulting multipole moment will be complex. Real multipole moments can be constructed according

to,

$$R_{lm} = \frac{(R_{lmc} + iR_{lms})}{2b_m} \tag{5.3}$$

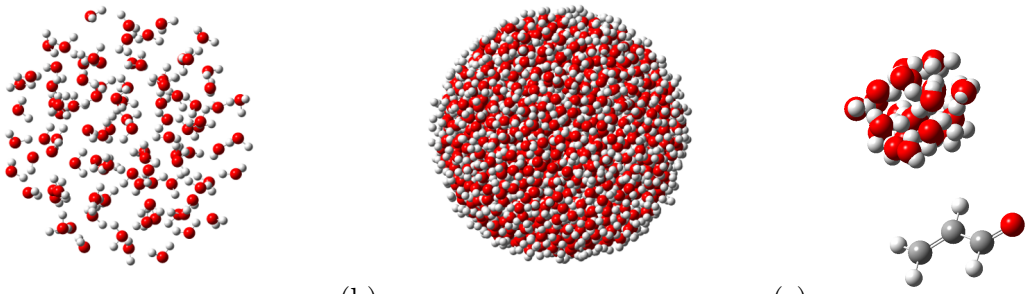
where b_m is a piece-wise defined coefficient, R_{lmc} and R_{lms} are the regular spherical harmonics. Additional details for deriving the translation expressions and a complete set of translations for up to hexadecapole moments are listed in the Supporting Information.

In this approach, multipolar translations are expressed as polynomials of degree l in terms of the elements of the translation vector, with coefficients of the moments of rank $\leq l$. A charge distribution described by a finite number of moments at a point would require moments of rank up to infinity to completely describe it at another point. For computational expediency, we truncate the multipole expansion at hexadecapoles ($l = 4$). In principle, errors introduced by translating the multipole moments to the CG sites could be systematically reduced by including higher-order moments.

The translation of the polarizabilities can be determined by applying Eq 5.2 to the multipolar operators that occur in the formula for the polarizability.[88] For example, for the dipole-dipole polarizability tensor elements α_{tu} are given by,

$$\alpha_{tu} = \sum_{n}^{'} \frac{\langle 0|\hat{\mu}_t|n\rangle \langle n|\hat{\mu}_u|0\rangle + \langle 0|\hat{\mu}_u|n\rangle \langle n|\hat{\mu}_t|0\rangle}{W_n - W_0} \tag{5.4}$$

where $\hat{\mu}_u$ and $\hat{\mu}_t$ are different components of the dipole moment operator, $|0\rangle$ and $|n\rangle$ are the ground and excited states of the system with energies W_n. For example, the t-th component of the dipole moment operator translated from initial point O to some new point C is given



(a) (b) (c)

Figure 5.3: (a) Single configuration snapshot of water solvent. (b) ACPE environment from 400 sampled configurations. (c) A single ACPE molecular cluster near an acrolein solute.

by,

$$\hat{\mu}_t^C = \hat{\mu}_t^O + qC \tag{5.5}$$

where $\hat{\mu}_t^C$ is the t-th element of the translated dipole operator, $\hat{\mu}_u^O$ is the original t-th element of the dipole operator, q is the charge of this site, and C is the t-th element of the translation vector. Substituting the operator form for this expression in for the dipole operators in Eq 5.4, one finds that because qC term is constant and the eigenstates are orthogonal, matrix elements involving the charge q are zero by orthogonality. See the Supporting Information of Ref [88] for details. In other words, the translated dipole-dipole polarizability is invariant to translation. Note that polarizability tensor elements involving higher-rank contributions are not invariant; however, only dipole-dipole polarizabilities are used in the embedding model here. Nevertheless, polarizability translation expressions up to rank 2 (quadrupole-quadrupole) are provided as Supporting Information.

5.2.4 Clustering of Coarse-grain sites

The K-means++ coarse graining in Section 5.2.1 produces a dense grid of points that can accurately reproduce the electrostatic interactions between the solute and environment. Figures 5.3a and 5.3b compare an individual water solvent configuration and the cluster of oxygen and hydrogen coarse-graining points representing an ACPE constructed by averaging over 400 solvent configurations. However, the coarse graining eliminates the definitions of individual molecules in the environment, which blurs the distinction between intra- and intermolecular polarization. Intramolecular polarization is already implicitly included in the AIFF monomer distributed multipoles, while intermolecular polarization needs to be modeled explicitly.

To recapture some of the distinction between intra- and intermolecular and enable intermolecular polarization in the coarse-grained representation, a second round of K-means++ is employed to group CG sites into CG clusters. A given CG cluster is composed only of CG sites derived from atoms from a given type of molecule. For example, in the mixed water/benzene environment described in Section 5.4.4, a given CG cluster would involve only oxygen and hydrogen sites derived from water molecules or carbon and hydrogen sites derived from benzene molecules. Loosely speaking, a given CG cluster corresponds roughly to the dynamic domain sampled by a single molecule (though it may be derived from contributions from various molecules). The ACPE model treats polarization only between CG clusters. Polarization within a CG cluster is forbidden.

Figure 5.3c shows a single ACPE water CG cluster interacting with an acrolein solute molecule. Note that while the CG sites in the CG clusters may seemingly resemble

water molecules, the actual bond distances and angles between oxygen and hydrogen sites correlate only loosely with real water molecules. It is also worth emphasizing that these water-like distributions of hydrogen and oxygen CG sites arise "naturally" from the K-means++ algorithm. The model was not steered to produce water-like CG sites.

Two parameters are used to define the CG clusters and their interactions. The first parameter is the number of CG clusters, K. We choose K to equal the average number of the solvent molecules from which the cluster was derived. For instance, if a solute is surrounded by 256 water molecules in each configuration being averaged over, there will be 256 CG clusters in the final ACPE.

The second parameter is the minimal distance between points in different clusters for which polarization is allowed. The goal is to allow maximal polarization while avoiding the polarization catastrophe. To determine this ACPE calculates self-consistent atom-centered induced dipoles due to many-body polarization. ACPE then calculates the average induced dipole for each atom type. For atoms with induced dipoles less than the system average, the cutoff is decreased by 0.1 Å. For atoms with induced dipoles greater than 0.03 a.u., is increased by 0.1 Å. This process of calculating the polarization and examining the induced dipoles is repeated until the many-body induction energy is between 95–105% of the configurational average of the original atomistic model or until 10 iterations have been reached. These calculations are performed purely at the MM level, so they can be done inexpensively.

This procedure was derived empirically, but it ensures that the coarse-grained polarization model faithfully reproduces the original atomistic polarization while avoiding

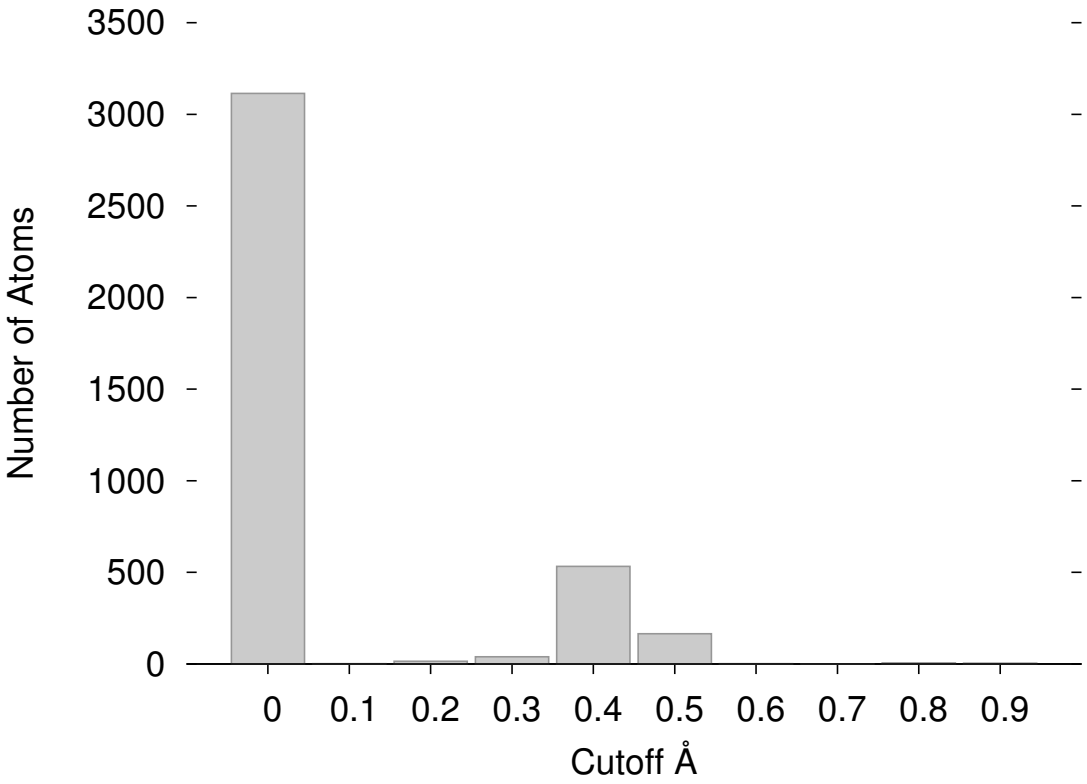


Figure 5.4: Distribution of polarization cutoff distances for the aqueous environment around acrolein.

the polarization catastrophe from close-lying clusters. As shown in Figure 5.4, the majority of CG sites are located on the interior of CG clusters and are sufficiently far from atoms in other CG clusters that they have a polarization cutoff of 0.0 Å(i.e. full polarization). Non-zero polarization cutoffs are primarily needed for atoms on the edges of adjacent CG clusters. In the current implementation, the polarization cutoff is a hard step function—polarization is either allowed or not. One could of course use a smooth damping function to interpolate between complete polarization and no polarization, but that is not investigated here.

In principle, the ideas here can be applied to any fixed-charge, multipolar, or

polarizable force field which is derived from a multipolar expansion. However, because translation of the force field parameters introduces higher-order multipolar components, it complicates the force field model. Translating atomic parameters from a simple fixed-charge force model like all-atom OPLS, for instance, leads to the introduction of dipoles and higher-order terms into the electrostatic model. Similarly, translating polarizable force field parameters from the Amoeba model introduces higher-rank multipoles and polarizabilities. Of course, one truncates the multipolar expansions to ignore some of the new, higher-rank contributions, albeit with some loss in accuracy. In the case considered here, we maintain the original force field expansions with maximal rank 4 (hexadecapole) multipoles and rank 1 (dipole-dipole polarizabilities) in the coarse-grained model. This captures the leading contributions arising from the translated dipoles, quadrupoles, and octupoles, but it neglects higher-rank contributions from translating hexadecapoles. Similarly, we neglect any changes in the polarizability beyond dipole-dipole that arise from translation. In other words, the ideas here are best-suited for force fields that already include multipoles beyond point charges.

5.2.5 Molecular excitation energies with polarizable embedding

In the end, the ACPE procedure described above produces a set of multipole moments and polarizabilities which can then be used to construct an embedding environment for a variety of quantum mechanical calculations. For the purposes of this paper, the ACPE model is used to construct a polarizable solvent environment for computing solute excitation energies with time-dependent density functional theory (TDDFT).

All embedding calculations use a self-consistent polarizable embedding (PE) scheme,[114]

which is implemented in Dalton 2013.[115] This model allows polarizable embedding with point multipoles and polarizabilities. The electrostatic potential is modeled with multipole moment up to rank 4 (hexadecapole), which is sufficient to model the permanent charge distribution. Polarization is treated using anisotropic dipole-dipole polarizabilities.

5.3 Computational Methods

Configuration sampling: Configuration sampling was performed via molecular dynamics (MD) simulations using Tinker 7.1[116] and the OPLS-AA force field.[117] In the first three test systems (s-trans acrolein, acetone, and pyrimidine) in aqueous solvent, MD simulations were performed under periodic boundary conditions in a cell containing a single solute molecule and 1600 water molecules. 500 ps of NPT dynamics at 298.15 K and 1.0 atm were carried out to equilibrate the system, followed by a 1.0 ns NVT production run. The solute molecule was held fixed during the MD simulations at a geometry optimized using the CAM-B3LYP functional, aug-cc-pVTZ basis, and implicit water solvation (using the integral equation formalism polarizable continuum model[118] in Gaussian 09[119]. 1.0 fs time steps were used throughout. 400 configurations were sampled at intervals of 1.0 ps over the last 0.4 ns of the production run.

For the benzene/water interface example, a rectangular box consisting of a single, frozen phenol molecule at the interface of 1600 waters and 138 benzene molecules stacked along the z-coordinate was generated (see Figure 5.8). Periodic boundary conditions were employed. The system was allowed to equilibrate for 100 ps under NPT conditions. The phenol was then frozen, and 400 ps of additional NPT equilibration were carried out to

obtain a box with lengths 36.02 Å, 36.02 Å, and 55.31 Å along x, y, and z, respectively. Subsequently, 1.0 ns of NVT MD production run was carried out. Again, 400 configurations were sampled at 1.0 ps intervals over the last 0.4 ns of the NVT simulation.

Finally, a spherical solvation shell consisting of all solvent molecules lying within 9 Å of any atom in the solute molecule was extracted from each MD configuration. These large clusters were then used to construct the ACPE model. These finite clusters may not fully capture bulk solvation effects, but they provide a useful test for the ACPE coarse-graining procedure. One could employ larger clusters or in some cases bulk continuum models for longer-range effects, though we do not do so here.

ACPE construction: The ACPE for a given system was constructed from the 400 configurations sampled from the MD. Unless otherwise mentioned, scaling factors of 5 (H atoms) and 30 (heavy atoms) were used to define the number of CG sites k used in the initial K-means++ coarse-graining for each system. This means, for example, that in a system with an average of 100 water molecules (100 oxygen and 200 hydrogen atoms) per configuration, there would be $5 \times 200 = 1000$ H sites and $30 \times 100 = 3000$ oxygen coarse-graining sites. These scaling factors were chosen empirically based on a survey of scaling parameters for acrolein—see Section 5.4.1 for details.

The AIFF distributed multipoles and polarizabilities for water and benzene were computed with CamCASP version 5.6[111] using asymptotically corrected PBE0 and the Sadlej basis. An ionization potential of 0.4638 au was used for the PBE0 asymptotic correction of water. For water, the force field parameters at each geometry were interpolated as described in Section 5.2.2. For benzene, force field parameters computed at the equi-

librium geometry were used throughout. Multipole moments up to hexadecapoles (rank 4) on heavy atoms and dipoles (rank 1) on hydrogen were used, along with polarizabilities up to dipole-dipole (rank 1) The translated multipoles and polarizabilities were truncated at ranks 4 and 1, respectively.

Excitation energy calculation: TDDFT excitation energies were computed using the polarizable embedding (PE) module in Dalton 2013[115] using the CAM-B3LYP density functional[120] and the aug-cc-pVTZ basis.[121] For the CAM-B3LYP functional, the parameters α modify the fraction of HF exchange and β modifies the fraction of the DFT exchange for short- and long-range interactions. Here, α and β values of 0.19 and 0.46 were used, respectively.[122] The switching factor between HF and DFT exchange μ is equal to 0.33, as proposed in the original work.[120] For the purposes of exploring the effect of the ACPE parameters in Section 5.4.1, calculations were performed with the less-expensive B3LYP/aug-cc-pVDZ model.

The PE was modeled with multipole moments up to hexadecapole (rank 4) and anisotropic dipole-dipole polarizabilities (rank 1). Polarization was treated self-consistently among the ACPE CG clusters and the QM region. Polarization within a CG cluster is omitted, and short-range polarization cutoffs between CG sites in different clusters were implemented as described in Section 5.2.4.

ACPE Validation: For comparison purposes, distributions of excitation energies were also computed using the polarizable embedding model for each of the 400 individual explicit configurations for each system. A "configurational average" excitation energy is obtained by computing the mean excitation energy for each state over the 400 configurations.

In some examples, integral equation formalism PCM calculations using default parameters for an aqueous environment were also performed with Dalton.

5.4 Results and Discussion

To investigate the performance of the ACPE, we first examine the structure of the averaged environment generated by the model. Next, we compute low-lying vertical singlet excitation energies for several small molecules in aqueous solution. Finally, to demonstrate application of the ACPE model to a more complicated, inhomogeneous environment, we study the excitations of a phenol molecule residing at the interface between benzene and water solvents. The sharp differences in solvent polarity and the spatial phase separation exhibited by the two solvents would make this type of system much harder to describe with conventional implicit solvent models.

5.4.1 Determination of the ACPE model parameters

The ACPE coarse-graining procedure contains a number of potential parameters that might affect the model results. First, the initial CG sites in the K-means++ algorithm are determined randomly, which raises the question of the reproducibility of the results for different random seeds. Second, one must choose the density of CG sites, which is defined as some scaling factor times the average number of atoms of a given type in the MD configuration snapshots. Third, short-range damping is used to avoid the polarization catastrophe in the embedding procedure as described in Section 5.2.4.

To explore how the first two parameters affect the excitation energies in the ACPE

Table 5.1: Predicted B3LYP/aug-cc-pVDZ ACPE excitation energies for acrolein with different scale factors for the number of heavy and light atom CG sites. Each ACPE calculation was repeated four times with different random initialization. Values report the average excitation energies and standard deviations. For reference, averaging over the 400 configurations explicitly predicts excitation energies of 3.84 eV ($n \rightarrow \pi^*$) and 5.86 eV ($\pi \rightarrow \pi^*$).

		$(n \rightarrow \pi*)$					$(\pi \rightarrow \pi*)$			
		Light					Light			
		5	SD	10	SD		5	SD	10	SD
Heavy	5	3.79	0.01	3.79	0.0	5	5.95	0.00	5.96	0.01
	10	3.82	0.02	3.8	0.03	10	5.94	0.00	5.94	0.03
	15	3.82	0.01	3.75	0.02	15	5.92	0.017	5.98	0.01
	20	3.88	0.02	3.85	0.03	20	5.90	0.013	5.92	0.01
	25	3.83	0.03	3.91	0.04	25	5.94	0.01	5.90	0.03
	30	3.85	0.02	3.91	0.06	30	5.91	0.02	5.89	0.03
	35	3.80	0.03	3.90	0.08	35	5.92	0.01	5.87	0.04
	40	3.82	0.05	3.88	0.04	40	5.94	0.02	5.91	0.02

environment, we consider the lowest two excited states of acrolein in water. As a reference, we first computed the excitation energy with B3LYP/aug-cc-pVDZ via polarizable embedding for each of 400 solvent configurations. Each configuration consists of a single acrolein surrounded by an average of 97 water molecules. Averaging over the 400 configurations, we obtain average excitation energies of 3.84 eV ($n \rightarrow \pi^*$) and 5.87 eV ($\pi \rightarrow \pi^*$).

Next, we performed ACPE calculations with four different random seeds (i.e. distinct K-means++ initializations) and varying scale factors for the number of heavy (oxygen) and light (hydrogen) atom CG sites (Table 5.1). For any given set of light/heavy atom scale factors, the standard deviation in the predicted excitation energies due to different random seeds in the K-means++ initialization is less than 0.1 eV.

Similarly, the excitation energies are relatively insensitive to the heavy and light atom scale factors. Using larger scale factors (more CG sites) reduces the typical distance between the original atom and its assigned CG site. Translating the force field parameters

(multipoles and polarizabilities) shorter distances reduces the magnitude of the higher-rank components introduced upon translation. The embedding model here includes up to hexadecapoles for the permanent multipole moments. Thus, the higher-order components introduced by translation are described fairly well. However, the PE model only supports dipole-dipole polarizabilities, so important higher-rank contributions to polarization introduced by longer translation distances will be omitted.

Higher-rank distributed multipoles can be significant in magnitude on heavy atoms,[123] which in turn means it may be beneficial to translate their parameters less distance (to minimize the introduction of components with rank > 4 that are not included in our model). For hydrogen, the force field representation before translation includes only up to dipoles (rank 1),[100, 101] so the higher-order contributions introduced by translation are captured more completely by the final rank 4 representation of the embedding environment. Despite these considerations, there does not appear to be any clear preference for certain combinations of scale factors in practice, as seen in Table 5.1. Scale factors of 30 for heavy atoms and 5 for light atoms seem to behave well and are used for all calculations described below.

With these parameters, the mean distance between the original atoms and their corresponding CG sites in the four ACPE runs described above is 0.42 ± 0.14 Å for oxygen and 0.69 ± 0.22 Å for hydrogen, respectively. Those individual CG sites are grouped into CG clusters (Section 5.2.4). If both the explicit solvent molecules and the CG sites were uniformly distributed, each heavy or light atom CG cluster would contain 30 (oxygen) or 2x5 (hydrogen) CG sites (i.e. the number of sites would match the scale factors). In the four ACPE runs described above, the clusters averaged 30.0 ± 3.9 CG sites for oxygen and

Table 5.2: Timings for the construction of the ACPE from 400 acrolein in water configurations. CG refers to the time to generate the CG sites via K-means++. Config. Pol. indicates the time to compute the polarization in the individual configurations, and ACPE Pol. Cutoffs the time to identify appropriate polarization cutoffs automatically.

	Time (s)	% of Total Time
CG Hydrogen	405	1.8%
CG Oxygen	303	1.3%
CG Clusters	41	0.16%
Translations	3	0.011%
Config. Pol.	597	2.3%
ACPE Pol. Cutoffs	5649	21.6%
ACPE Total	6998	26.7%

10.0 ± 1.6 for hydrogen.

Our implementation of the ACPE algorithm has not been fully optimized for computational efficiency. Nevertheless, timings of the current implementation demonstrate that the construction of the ACPE requires only a fraction of the subsequent excitation energy calculation. Table 5.2 breaks the timings down into individual components of the ACPE algorithm. The large number of polarizable sites employed in the polarizable embedding with ACPE also modestly increases the time for the TDDFT calculation (by ∼3,000 seconds). Overall, generating the ACPE and computing the ten lowest excited states of acrolein in the water via the ACPE model requires 26,141 seconds, compared to 15,994 seconds for embedding with multipoles and polarizabilities from a single configuration (i.e. without any ACPE model). In other words, employing the ACPE model allows one to mimic the effect of hundreds of solvent configurations at a cost only 60% higher than that of a TDDFT calculation on a single configuration snapshot. Table 5.2 also suggests that further work should be done to simplify the handling of polarization damping to reduce the computational time further.

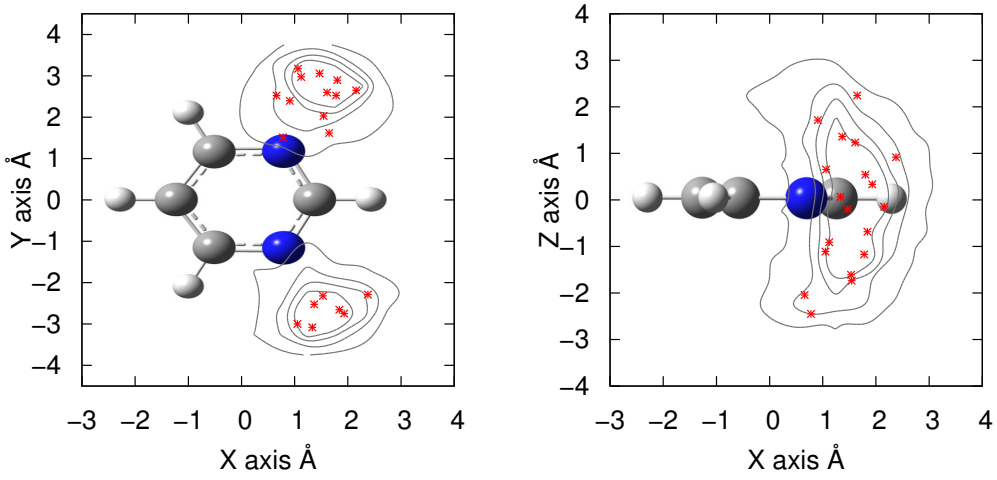


Figure 5.5: Hydrogen bonding distribution for molecular dynamics configurations and K-means++ generated grid points around pyrimidine.

5.4.2 Solvent structure in the ACPE model

One of the primary motivations underlying the ACPE model is to retain important local structural features of the environment that would not be found in a more traditional implicit solvent model. Continuum solvent models typically have difficulty describing local solute-solvent hydrogen bonding interactions, for instance, which sometimes necessitates the inclusion of explicit solvent molecules. Because ACPE derives its representation of the environment from explicit solvent configurations, it naturally retains some of these localized interactions.

Consider the hydrogen bonding interactions between pyrimidine and solvent water. The contours in Figure 5.5 plot the distribution of hydrogen atoms near the two hydrogen-bond accepting nitrogen atoms in pyrimidine over the 400 configurations, projected onto the

molecular xy or xz planes. The red symbols represent hydrogen atom CG sites identified by the K-means++ algorithm. The K-means++ algorithm naturally places CG sites in regions with the highest hydrogen atom density. It captures the variability in hydrogen bond lengths and angles observed across the MD configuration snapshots.

Another perspective on solute-solvent interactions can be gleaned from the radial distribution functions (RDFs), which are plotted in Figure S1 in the Supporting Information. One can compute the average number of hydrogen bonds by counting the number of atoms within the first solvent shell of the N$\cdots$H-O RDF and dividing by the number of snapshots considered. From the MD simulations, the first solvent shell ends at an N$\cdots$H distance of 2.7 Å, and integrating the RDF indicates that a pyrimidine nitrogen averages 1.84 hydrogen bonds. For the ACPE model, we obtain the number of hydrogen bonds by counting the number of CG sites within in the same N$\cdots$H distance and dividing by the scaling factor (5 here). Doing so, one finds an average number of 1.90 hydrogen bonds in ACPE, in very good agreement with the explicit MD result.

5.4.3 Excitation Energies in Aqueous Environment

Next, we examine the performance of ACPE for reproducing small-molecule vertical excitation energies in solution. We compare the ACPE excitation energies against values obtained from a traditional configurational average approach, a polarizable continuum model, and experiment. One should bear in mind that discrepancies between the predicted and experimental results can arise for reasons including limitations of the TDDFT functional, basis set, and embedding model, the quality of the ensemble generated from OPLS,

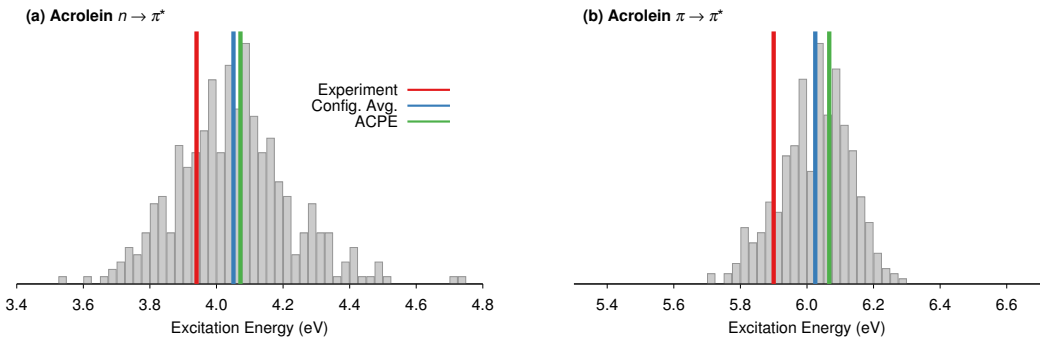


Figure 5.6: Histograms of the first and second singlet excitation energies of acrolein compared with the experimental, configurational average, and ACPE values. Box heights indicate the number of configurations with this excitation energy.

and the finite cluster truncation of the bulk solvent model in addition to the ACPE approximations. Accordingly, the comparison between the configurational average and the ACPE model results provides more direct insight into the behavior of the ACPE approximations.

Figure 5.6 plots a histogram of excitation energies from each of the 400 individual polarizable embedding calculations, where the height of each box corresponds to the number of configurations exhibiting excitation energies within the particular energy interval. Across the 400 sampled configurations, the $n \rightarrow \pi^*$ excitation energies occur between 3.55 and 4.74 eV, with an average value of 4.05 eV. This average excitation energy is in good agreement with the value of 4.11 eV from an earlier work using polarizable embedding and the M2P2 force field.[124] The second excitation in acrolein, $\pi \rightarrow \pi^*$, occurs between 5.71 and 6.29 eV across the 400 configurations, with an average of 6.03 eV. The configurational averages for these two excitation energies also lie within 0.1–0.2 eV of experiment (Table A.1), which is well within the accuracy expected for TDDFT with CAM-B3LYP.[125] Plots of the excitation energies as a function of the number of solvent configurations averaged over suggest that both of these configurational averages are converged to within a few hundredths of an

eV with respect to the number of configurations sampled (see Figure S2 in the Supporting Information).

By definition, the ACPE model does not capture the full distribution of excitation energies observed over the 400 configurations. However, it would ideally mimic the configurational average. Indeed, for both the $n \rightarrow \pi^*$ and $\pi \rightarrow \pi^*$ transitions in acrolein, the ACPE reproduces the configurational average to within 0.03–0.05 eV (Figure 5.6 and Table A.1), despite performing only a single QM calculation instead of 400. Though the specific CG sites identified by the K-means clustering algorithm will vary with the initial guess, the resulting excitation energies from three different initial guesses varied by only $\pm$0.01–0.02 eV both states (see Table S1 in Supporting Information).

Reliable prediction of solvatochromic shifts is often important when modeling electronic excitations in solution. Upon switching from the gas-phase to an aqueous environment, the ACPE model predicts solvatochromic shifts of +0.23 eV for the $n \rightarrow \pi^*$ transition and -0.40 for the $\pi \rightarrow \pi^*$ one (Table A.1). These shifts agree very well with the configurational average shifts of +0.21 eV and -0.44 eV, respectively. They are also in fairly good agreement with the experimental shifts of +0.25 eV and -0.52 eV.[122]

The good agreement in the solvatochromic shifts for the two lowest excited states is notable because the electronic character of these transitions differs notably. The acrolein $\pi \rightarrow \pi^*$ excitation shows a sizable red shift of -0.30 eV in iso-octane, while the $n \rightarrow \pi^*$ transition shows much smaller solvatochromic shift in non-polar solvents.[122] Aidas et al suggest that the $\pi \rightarrow \pi^*$ shift depends on both electrostatics and intermolecular polarization effects, while the $n \rightarrow \pi^*$ solvent shift is dominated by electrostatic interactions.[122] The

Table 5.3: Comparison of CAM-B3LYP/aug-cc-pVTZ excitation energies E and solvatochromic shifts ΔE for three solutes in aqueous solution. The ACPE excitation energies are the average of three calculations. The individual values can be found in the SI.

	Acrolein		Acetone		Pyrimidine	
	E $(n \rightarrow \pi*)$	ΔE	E $(n \rightarrow \pi*)$	ΔE	E $(n \rightarrow \pi*)$	ΔE
Gas	3.84	–	4.49	–	4.55	–
PCM	3.97	0.13	4.59	0.11	4.71	0.16
Config. Avg.	4.05	0.21	4.65	0.17	5.01	0.46
ACPE	4.07	0.23	4.64	0.16	5.01	0.46
Experiment	3.94[a]	0.25[a]	4.68[b]	0.22[b]	4.57[c], 4.84[d]	0.35[c],0.62[e]
	E $(\pi \rightarrow \pi*)$	ΔE	–	–	–	–
Gas	6.46	–	–	–	–	–
PCM	5.76	-0.70	–	–	–	–
Config. Avg.	6.03	-0.44	–	–	–	–
ACPE	6.07	-0.40	–	–	–	–
Experiment	5.90[a]	−0.52[a]	–	–	–	–

[a] Ref [122] [b] Ref [126] [c] Ref [127] [d] Ref [128] [e] Inferred using the gas-phase excitation energy from Ref [127] and the solution-phase excitation energy from Ref [128].

ACPE describes both solvatochromic shifts well, despite the differences in the excitation characters and their responses to the solvent environment.

Next, we consider the lowest singlet excitation energies in acetone and pyrimidine. For acetone, the lowest excitation corresponds to the forbidden $n \rightarrow \pi^*$ transition. The energy of this excitation ranges from 4.33 to 5.29 eV over the sampled MD configurations (Figure 5.7a). Averaging over all 400 configurations produces a configurational average excitation energy of 4.65 eV, which is in excellent agreement with both the earlier M3P2 force field prediction of 4.75 eV[124] and the experimental value of 4.68 eV. A single ACPE calculation reproduces the configurational average for the first excitation to within 0.01 eV. The solvatochromic shift of 0.16–0.17 eV predicted by both the ACPE model and the configurational average are also in very good agreement with the experimental shift of 0.22

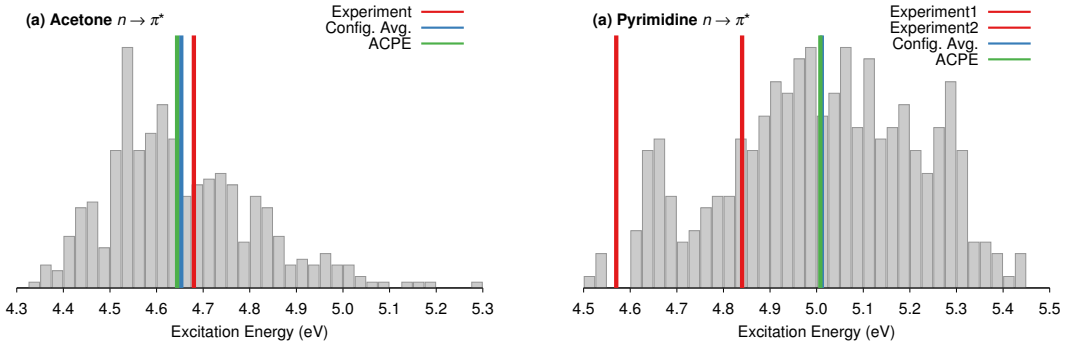


Figure 5.7: Histograms of the lowest singlet excitation energies of (a) acetone and (b) pyrimidine compared with the experimental, configurational average, and ACPE values. For pyrimidine, two different reported experimental excitation energies are shown.

eV (Table A.1).[126]

Figure 5.7b plots the analogous results for the $n \rightarrow \pi^*$ transition in pyrimidine. The excitation energies range 4.52–5.44 eV over the 400 MD configurations, with a configurational average of 5.01 eV. Once again, the ACPE calculation reproduces the configurational average excitation energy to within 0.01 eV. Experimentally, the $n \rightarrow \pi^*$ excitation is very broad, making it difficult to assign a precise excitation energy. Values ranging from 4.57 eV[127] to 4.84 eV[128] are reported in the literature. Our predictions agree with the latter value to within 0.20 eV. The predicted ACPE and configurational average solvatochromic shifts of 0.46–0.46 eV are also in similarly good agreement with the corresponding experimental value of 0.62 eV (see Table A.1). Like for acrolein, the variation in the acetone and pyrimidine ACPE excitation energies with the initial random K-means clustering guess is only a few hundredths of an eV (Table S1).

Finally, it is interesting to compare the ACPE results against those obtained with an implicit PCM water model. As shown in Table A.1, the excitation energies in the

PCM are consistently lower than the configurational average ones for the cases examined here. The PCM excitation energy errors with respect to the experiment are similar to or slightly smaller than those from the configurational averages or the ACPE model. However, given the few tenths of an eV errors typically expected for valence excitation energies with TDDFT,[125] none of the approaches is clearly superior in terms of the excitation energies. On the other hand, the solvatochromic shifts computed with the configurational averages and/or ACPE model are consistently better than those from the PCM model. This is most notable for pyrimidine, for which it has been argued that obtaining reliable solvatochromic shifts requires the inclusion of several explicit waters.[129] The pyrimidine PCM model shift of 0.16 eV (without any explicit solvent molecules) is reasonably close to the 0.35 eV shift from Ref [127], but it is much further away from the value of 0.62 eV value inferred from Ref [128]. The ACPE and configurational average shifts of 0.46 eV lie in between the two experimental values.

Overall, for these simple examples of computing small-molecule excitation energies in aqueous solution, the ACPE model performs very well. A single QM excitation energy calculation embedded in the ACPE reproduces the excitation energies and solvatochromic shifts obtained from a much more expensive configurational average to within a few hundredths of an eV. Of course, PCM and other simple models often can describe these sorts of homogeneous bulk environments well. In the next section, however, we consider a spatially inhomogeneous model that would be much harder to describe with standard implicit models.

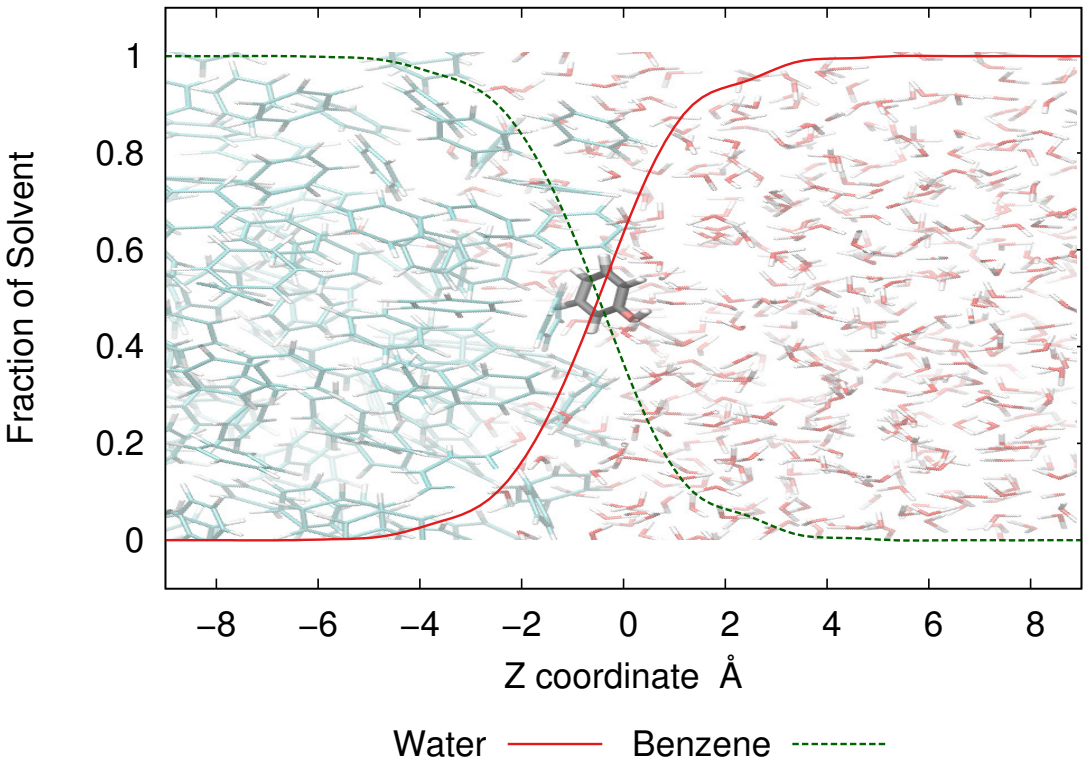


Figure 5.8: Proportion of water and benzene molecules as a function of the z coordinate, averaged over 400 MD configurational snapshots, for phenol at the benzene/water interface.

5.4.4 Solute at the benzene-water interface

To test the ability of the ACPE model to treat an inhomogeneous environment, we construct a model system consisting of a phenol solute molecule at the interface of liquid benzene and water. This system was chosen because (1) the two solvents exhibit very different polarities and would create an interface with an asymmetric electrostatic environment and (2) the inherent rigidity of the benzene simplifies the treatment of its force field parameters. Figure 5.8 shows a sample MD configuration of this system and plots the proportion of water and benzene molecules (averaged over all 400 configuration snapshots) as a function of the z-coordinate in the box. The left side of the box is dominated by

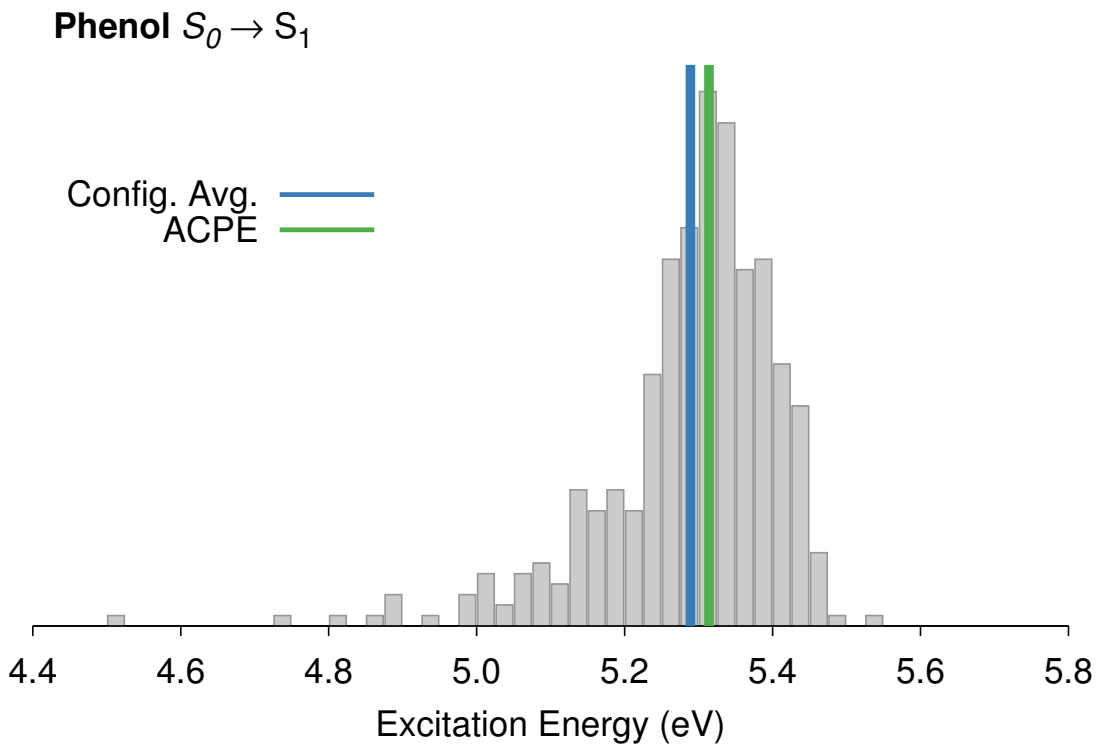


Figure 5.9: Histogram of excitation energies for phenol lying at the benzene/water interface.

benzene molecules, while the right side consists mostly of water ones. The phenol molecule resides right at the interface, with the hydroxyl group oriented toward the water region.

Figure 5.9 plots the distribution of excitation energies observed for phenol across the 400 MD configurations. Despite the strong asymmetry of the environment, the ACPE reproduces the configurational average to within less than 0.02 eV (Table 5.4). Further insight is obtained by investigating the effects of each solvent layer on the phenol first excitation energy separately. Table 5.4 compares the configurational average and ACPE phenol excitation energies for phenol with only the aqueous solvent molecules present, only the benzene solvent molecules, and in the presence of both solvents. The water-only and

Table 5.4: Comparison of CAM-B3LYP/aug-cc-pVTZ excitation energies E and solvatochromic shifts ΔE for phenol at a benzene-water interface.

	Water		Benzene		Interface	
	E ($S_0 \rightarrow S_1$)	ΔE	E ($S_0 \rightarrow S_1$)	ΔE	E ($S_0 \rightarrow S_1$)	ΔE
Gas	5.16	–	–	–	–	–
Config. Avg.	5.33	0.17	5.05	-0.11	5.29	0.13
ACPE	5.33	0.17	5.08	-0.08	5.31	0.16

benzene-only cases use the same solvent configurations as the system as a whole, just with the other solvent molecules deleted. The two solvents induce opposing solvatochromic shifts on the $S_0 \rightarrow S_1$ excitation. The pure benzene layer red shifts the excitation energy by -0.11 eV, while the pure water layer causes a 0.17 eV blue shift. When both sets of solvent molecules are present, however, the excitation energy undergoes a 0.13 eV blue shift. In other words, the effect of the solvent interface is more than a simple average of the two parts.

Overall, the water/benzene interface provides a nice example of the robustness of the ACPE model. Despite the heterogeneous and dynamic nature of the environment surrounding the solute, the ACPE model captures the average environment in a single calculation.

5.5 Conclusions

In conclusion, we have presented an automated procedure for constructing a configuration-averaged condensed-phase environment model around a region of interest based on K-means++ clustering and force field parameter translation procedures. The model has a few adjustable parameters (the number of coarse-graining sites, the initial random seed,

and the polarization cutoffs), but fortunately the results seem relatively insensitive over a range of reasonable choices for these parameter values.

The chief advantages of this approach are that the resulting coarse-grained embedding model (1) retains specific structural features of the underlying atomistic model and (2) it reproduces the conventional configurational average approach with very high accuracy at orders of magnitude lower computational cost. We demonstrated, for example, that it reproduces key locations of hydrogen bonding partners and accurately describes the behavior of a phenol molecule located at the interface of benzene and water solvents. More generally, the ACPE model may prove useful in situations where an inhomogeneous environment precludes the use of more traditional continuum environment models.

Once a set of configurations has been obtained via some sampling procedure, constructing the ACPE model requires minimal computational effort—typically only a fraction of the time required to perform a single embedded excitation energy calculation here. In other words, the computational savings factor for the ACPE model compared to a conventional QM/MM configurational average is approaches the number of configurations sampled. At the same time, the ACPE excitation energies reported here all reproduce the configurational average values to within less than 0.1 eV, which is well within the sorts of errors one expects from TDDFT valence excitation energies.

The ACPE model does currently have limitations and opportunities for future work. Most pressingly, all results here utilized a fixed QM region (the frozen solute) in a dynamic environment. In practice, one should also consider the dynamics of the QM region. One possible path forward would be to sample configurations of the QM region, freeze them,

and then perform additional sampling of the environment to generate an ACPE for each sampled QM configuration, though other alternatives may also exist.

Additionally, the model is predicated on the notion that the configuration averaging can be performed before the property calculation (e.g. excitation energies), instead of afterwards, as is more traditional. This clearly works well in the examples tested here, and it will likely work well in cases where the observable properties of interest occur on time scales which are long relative to the configuration averaging. Experimentally observed nuclear magnetic resonance chemical shifts, for instance, typically represent a time average over nuclear motions. Predictions of other observables which occur on much shorter time scales may be less amenable to such *a priori* configurational averaging.

Finally, the examples here involved rather simple model systems. It will be interesting to extend these ideas to more general systems and a broader range of polarizable force fields. Generalization to more classes of systems might also provide additional insight into how to choose appropriate values for the handful of user-defined parameters in the ACPE model (number of coarse-graining sites, polarization cutoffs, etc.).

Chapter 6

Conclusions

In summary, two new computationally efficient algorithms for evaluating the self-consistent polarization equations in polarizable force fields have been proposed. Based on non-overlapping and overlapping domain decomposition, these divide-and-conquer Jacobi iterations (DC-JI) and fuzzy DC-JI algorithms can provide substantial savings over a conventional preconditioned conjugate gradients (PCG) implementation. They achieve this by solving the mutual polarization within clusters of atoms directly while mutual polarization between cluster is captured iteratively. K-means clustering is used to identify near optimal clusters that ensure rapid convergence of the iterations. We have also demonstrated that DC-JI can be coupled with direct inversion of the iterative subspace (DIIS) to further accelerate the convergence.

We implemented the non-overlapping DC-JI/DIIS polarization solver for periodic systems via particle-mesh Ewald in the massively parallel Tinker-HP software package. This massively parallel implementation of DC-JI/DIIS solves the polarization equations up

to 20% faster than PCG and 30% faster than JI/DIIS. DC-JI/DIIS also obtains induced dipoles closer to the exact solution than those from PCG. As of this writing, DC-JI/DIIS is the default polarization solver in Tinker-HP.

The use of a longer history in the "predicted iteration" variant of the always stable predictor-corrector (ASPC) method provides substantial computational benefits for SCF polarization solvers. The energy drift is reduced by an order of magnitude going from the standard 6-step predictor to our recommended 16-step predictor. The impact of this starting point for the SCF solvers is so large that with the 16-step predictor one less iteration is needed to achieve comparable stability to the previously used 6-step predictor. This one less iteration can lead to acceleration of the polarization evaluation by $\sim$20%.

The combination of all the techniques outlined in this work can lead to substantial acceleration of polarizable force fields. Prior to the techniques introduced here a typical polarization calculation might use PCG with 4 iterations per polarization evaluation and the 6 step predictor, replacing the solver with DC-JI/DIIS using 3 iterations per polarization evaluation and the 16 step predictor will accelerate the polarization evaluation by $\sim$ 50% and also achieve less energy drift than PCG. Given that solving the polarization equations consumes about half the total time, this amounts to a $\sim$ 25% speedup overall and higher-quality numerics.

Finally, we presented the average condensed phase environment (ACPE) for QM/MM embedding. ACPE retains atomistic details of the environment leading to a better description of the environment relative to the polarizable continuum model. ACPE faithfully models the time averaged behavior of an extended environment. We demonstrated that

APCE reproduces the configurational average values of vertical excitation energies for several solutes in a water solution to within less than 0.1 eV. In addition, we demonstrated ACPE's ability to describe complex environments like interfaces.

Bibliography

[1] A. Morozenko, I. V. Leontyev, and A. A. Stuchebrukhov, Journal of Chemical Theory and Computation **10**, 4618 (2014).

[2] C. Ji and Y. Mei, Acc. Chem. Res. **47**, 2795 (2014), pMID: 24883956.

[3] W. Wang and R. D. Skeel, J. Chem. Phys. **123**, 164107 (2005).

[4] F. Lipparini, L. Lagardère, B. Stamm, E. Cancès, M. Schnieders, P. Ren, Y. Maday, and J.-P. Piquemal, J. Chem. Theory Comput. **10**, 1638 (2014).

[5] J. A. Rackers, Q. Wang, C. Liu, J.-P. Piquemal, P. Ren, and J. W. Ponder, Phys. Chem. Chem. Phys. **19**, 276 (2017).

[6] J. W. Ponder, C. Wu, P. Ren, V. S. Pande, J. D. Chodera, M. J. Schnieders, I. Haque, D. L. Mobley, D. S. Lambrecht, R. A. DiStasio, M. Head-Gordon, G. N. I. Clark, M. E. Johnson, and T. Head-Gordon, The Journal of Physical Chemistry B **114**, 2549 (2010), pMID: 20136072.

[7] P. Ren, C. Wu, and J. W. Ponder, J. Chem. Theory Comput. **7**, 3143 (2011).

[8] P. Ren and J. W. Ponder, J. Phys. Chem. B **107**, 5933 (2003).

[9] Y. Shi, Z. Xia, J. Zhang, R. Best, C. Wu, J. W. Ponder, and P. Ren, J. Chem. Theory Comput. **9**, 4046 (2013).

[10] N. L. Allinger, Y. H. Yuh, and J. H. Lii, Journal of the American Chemical Society **111**, 8551 (1989).

[11] T. A. Halgren, Journal of the American Chemical Society **114**, 7827 (1992).

[12] J. Kolafa, Mol. Sim. **18**, 193 (1996).

[13] J. Kolafa, J. Comp. Chem. **25**, 335 (2003).

[14] A. Albaugh, O. Demerdash, and T. Head-Gordon, J. Chem. Phys. **143**, 174104 (2015).

[15] D. Frenkel and B. Smit, *Understanding Molecular Simulation: From Algorithms to Applications* (Academic Press, San Diego, 2002).

[16] A. Albaugh, H. A. Boateng, R. T. Bradshaw, O. N. Demerdash, J. Dziedzic, Y. Mao, D. T. Margul, J. Swails, Q. Zeng, D. A. Case, P. Eastman, L.-P. Wang, J. W. Essex, M. Head-Gordon, V. S. Pande, J. W. Ponder, Y. Shao, C.-K. Skylaris, I. T. Todorov, M. E. Tuckerman, and T. Head-Gordon, The Journal of Physical Chemistry B **120**, 9811 (2016), pMID: 27513316.

[17] L. Lagardère, F. Lipparini, E. Polack, B. Stamm, E. Cancès, M. Schnieders, P. Ren, Y. Maday, and J.-P. Piquemal, J. Chem. Theory Comput. **11**, 2589 (2015).

[18] J. A. Anderson, C. D. Lorenz, and A. Travesset, Journal of Computational Physics **227**, 5342 (2008).

[19] P. Eastman, M. S. Friedrichs, J. D. Chodera, R. J. Radmer, C. M. Bruns, J. P. Ku, K. A. Beauchamp, T. J. Lane, L.-P. Wang, D. Shukla, T. Tye, M. Houston, T. Stich, C. Klein, M. R. Shirts, and V. S. Pande, Journal of Chemical Theory and Computation **9**, 461 (2013), pMID: 23316124.

[20] M. Tuckerman, B. J. Berne, and G. J. Martyna, J. Chem. Phys. **97**, 1990 (1992).

[21] S. Lindert, D. Bucher, P. Eastman, V. Pande, and J. A. McCammon, Journal of Chemical Theory and Computation **9**, 4684 (2013), pMID: 24634618.

[22] P. Pulay, Chem. Phys. Lett. **73**, 393 (1980).

[23] P. Pulay, J. Comput. Chem. **3**, 556 (1982).

[24] T. Rohwedder and R. Schneider, J. Math. Chem. **49**, 1889 (2011).

[25] A. C. Simmonett, F. C. Pickard IV, Y. Shao, T. E. Cheatham III, and B. R. Brooks, J. Chem. Phys. **143**, 074115 (2015).

[26] A. C. Simmonett, F. C. Pickard, J. W. Ponder, and B. R. Brooks, J. Chem. Phys. **145**, 164101 (2016).

[27] O. Demerdash and T. Head-Gordon, J. Chem. Theory Comput. **12**, 3884 (2016), pMID: 27405002.

[28] R. Qi, L.-P. Wang, Q. Wang, V. S. Pande, and P. Ren, J. Chem. Phys. **143**, 014504 (2015).

[29] L.-P. Wang, T. Head-Gordon, J. W. Ponder, P. Ren, J. D. Chodera, P. K. Eastman, T. J. Martinez, and V. S. Pande, J. Phys. Chem. B **117**, 9956 (2013), pMID: 23750713.

[30] F. Aviat, A. Levitt, B. Stamm, Y. Maday, P. Ren, J. W. Ponder, L. Lagardère, and J.-P. Piquemal, J. Chem. Theory Comput. **13**, 180 (2017).

[31] J. W. Demmel, *Applied Numerical Linear Algebra, 2nd ed* (SIAM, ADDRESS, 1997).

[32] J. W. Ponder, TINKER v7.1, 2015, `http://dasher.wustl.edu/tinker/`, accessed September 21, 2016.

[33] P. Ren and J. W. Ponder, Journal of Computational Chemistry **23**, 1497 (2002).

[34] X. Cai, in *Advanced Topics in Computational Partial Differential Equations: Numerical Methods and Diffpack Programming*, edited by H. P. Langtangen and A. Tveito (Springer Berlin Heidelberg, Berlin, Heidelberg, 2003), pp. 57–95.

[35] in *SODA '07: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007).

[36] M. P. Allen and D. J. Tildesley, *Computer Simulations of Liquids* (Oxford Science Publications, Oxford Science Publications, Great Clarendon Street, Oxford OX2 6DP, 1987), pp. 147–149.

[37] D. Nocito and G. J. O. Beran, Journal of Chemical Theory and Computation **13**, 1117 (2017), pMID: 28170251.

[38] P. Ahlström, A. Wallqvist, S. EngstrÃ¶m, and B. Jönsson, Molecular Physics **68**, 563 (1989).

[39] T. Yan, C. J. Burnham, M. G. Del Popolo, and G. A. Voth, J. Phys. Chem. B **108**, 11877 (2004).

[40] A. Warshel, M. Kato, and A. V. Pisliakov, Journal of Chemical Theory and Computation **3**, 2034 (2007).

[41] P. Cieplak, F. Dupradeau, Y. Duan, and J. Wang, J. Phys. Condens. Mat. **21**, 333102 (2009).

[42] G. A. Cisneros, M. Karttunen, P. Ren, and C. Sagui, Chem. Rev. **114**, 779 (2014), pMID: 23981057.

[43] F. Aviat, L. Lagardère, and J.-P. Piquemal, J. Chem. Phys. **147**, 161724 (2017).

[44] A. Albaugh, A. M. N. Niklasson, and T. Head-Gordon, J. Phys. Chem. Lett. **8**, 1714 (2017).

[45] A. Albaugh, T. Head-Gordon, and A. M. N. Niklasson, J. Chem. Theory Comput. **14**, 499 (2018).

[46] J. W. Ponder, TINKER v8.1, 2018, `http://dasher.wustl.edu/tinker/`, accessed March 14, 2018.

[47] M. Harger, D. Li, Z. Wang, K. Dalby, L. Lagardère, J.-P. Piquemal, J. Ponder, and P. Ren, J. Comp. Chem. **38**, 2047 (2017).

[48] L. Lagardère, L.-H. Jolly, F. Lipparini, F. Aviat, B. Stamm, Z. F. Jing, M. Harger, H. Torabifard, G. A. Cisneros, M. J. Schnieders, N. Gresh, Y. Maday, P. Y. Ren, J. W. Ponder, and J.-P. Piquemal, Chem. Sci. **9**, 956 (2018).

[49] D. Nocito and G. J. O. Beran, J. Chem. Phys. **146**, 114103 (2017).

[50] AMBER/CHARMM DHFR Benchmark, 2018, `http://ambermd.org/amber8.bench2.html`, accessed April 2, 2018.

[51] Tinker Benchmark Suite, 2018, `https://dasher.wustl.edu/tinker/distribution/bench/`, accessed April 2, 2018.

[52] O. Borodin, J. Phys. Chem. B **113**, 11463 (2009).

[53] J. Huang and A. D. MacKerell, Curr. Opin. Struct. Bio. **48**, 40 (2018).

[54] D. S. Davidson, A. M. Brown, and J. A. Lemkul, J. Mol. Biol. **430**, 3819 (2018).

[55] J. A. Rackers, Z. Wang, C. Lu, M. L. Laury, L. Lagardére, M. J. Schnieders, J.-P. Piquemal, P. Ren, and J. W. Ponder, J. Chem. Theory Comput. **14**, 5273 (2018).

[56] D. Nocito and G. J. O. Beran, J. Chem. Theory Comput. **14**, 3633 (2018).

[57] Tinker example directory https://dasher.wustl.edu/tinker/distribution/example/, accessed: 2018-11-10.

[58] B. Kirchner, P. J. di Dio, and J. Hutter, in *Multiscale Molecular Methods in Applied Chemistry*, edited by B. Kirchner and J. Vrabec (Springer, Berlin, Heidelberg, 2012), pp. 109–153.

[59] J. Chandrasekhar, S. F. Smith, and W. L. Jorgensen, J. Am. Chem. Soc. **107**, 154 (1985).

[60] R. V. Stanton, M. Perakyla, D. Bakowies, and P. A. Kollman, J. Am. Chem. Soc. **120**, 3448 (1998).

[61] Y. Zhang, H. Liu, and W. Yang, J. Chem. Phys. **112**, 3483 (2000).

[62] N. Ferre and J. Angyan, Chem. Phys. Lett. **356**, 331 (2002).

[63] T. H. Rod and U. Ryde, J. Chem. Theory Comput. **1**, 1240 (2005).

[64] J. Kastner, H. M. Senn, S. Thiel, N. Otte, and W. Thiel, J. Chem. Theory Comput. **2**, 452 (2006).

[65] P. Pulay and T. Janowski, Int. J. Quant. Chem. **109**, 2113 (2009).

[66] T. Janowski, K. Wolinski, and P. Pulay, Chem. Phys. Lett. **530**, 1 (2012).

[67] H. Nakano and T. Yamamoto, J. Chem. Theory Comput. **9**, 188 (2013).

[68] A. J. Sodt, Y. Mei, G. König, P. Tao, R. P. Steele, B. R. Brooks, and Y. Shao, Journal of Physical Chemistry A **119**, 1511 (2015).

[69] C. J. Cramer and D. G. Truhlar, Chem. Rev. **99**, 2161 (1999).

[70] J. Tomasi, B. Mennucci, and R. Cammi, Chem. Rev. **105**, 2999 (2005).

[71] B. Mennucci, WIREs Comput. Mol. Sci. **2**, 386 (2012).

[72] D. J. Cram and J. E. McCarty, J. Am. Chem. Soc. **76**, 5740 (1954).

[73] D. J. Cram and M. R. V. Sahyun, J. Am. Chem. Soc. **84**, 1734 (1962).

[74] M. R. V. Sahyun and D. J. Cram, J. Am. Chem. Soc. **85**, 1263 (1963).

[75] O. Acevedo and W. L. Jorgensen, J. Am. Chem. Soc. **128**, 6141 (2006).

[76] C. N. Schutz and A. Warshel, Proteins **44**, 400 (2001).

[77] L. Li, C. Li, Z. Zhang, and E. Alexov, J. Chem. Theory Comput. **9**, 2126 (2013).

[78] P. Kukic, D. Farrell, L. P. McIntosh, B. García-Moreno E, K. S. Jensen, Z. Toleikis, K. Teilum, and J. E. Nielsen, J. Am. Chem. Soc. **135**, 16968 (2013).

[79] L. An, Y. Wang, N. Zhang, S. Yan, A. Bax, and L. Yao, J. Am. Chem. Soc. **136**, 12816 (2014).

[80] M. L. Sanchez, M. A. Aguilar, and F. J. Olivares del Valle, J. Comput. Chem. **18**, 313 (1997).

[81] M. Sanchez Mendoza, M. Aguilar, and F. Olivares del Valle, J. Mol. Struct. THEOCHEM **426**, 181 (1998).

[82] M. L. Sanchez, M. E. Martin, M. A. Aguilar, and F. J. Olivares del Valle, J. Comp. Chem. **21**, 705 (2000).

[83] K. Coutinho, H. Georg, T. Fonseca, V. Ludwig, and S. Canuto, Chem. Phys. Lett. **437**, 148 (2007).

[84] D. Beglov and B. Roux, J. Phys. Chem. B **101**, 7821 (1997).

[85] E. L. Ratkova, D. S. Palmer, and M. V. Fedorov, Chem. Rev. **115**, 6312 (2015).

[86] A. Kovalenko and F. Hirata, J. Chem. Phys. **110**, 10095 (1999).

[87] S. Gusarov, T. Ziegler, and A. Kovalenko, J. Phys. Chem. A **110**, 6083 (2006).

[88] K. L. Theel, S. Wen, and G. J. O. Beran, J. Chem. Phys. **139**, 081103 (2013).

[89] L. Greengard and V. Rokhlin, J. Comp. Phys. **73**, 325 (1987).

[90] A. Warshel, J. Phys. Chem. **83**, 1640 (1979).

[91] A. Warshel and S. T. Russell, Q. Rev. Biophys **17**, 283 (1984).

[92] Journal of Molecular Structure: THEOCHEM **366**, 1 (1996).

[93] J. Florián and A. Warshel, J. Phys. Chem. B **101**, 5583 (1997).

[94] I. Galvan, M. Sanchez, M. Martin, F. Olivares del Valle, and M. Aguilar, Comput. Phys. Comm. **155**, 244 (2003).

[95] I. F. Galván, M. E. Martín, and M. a. Aguilar, J. Comp. Chem. **25**, 1227 (2004).

[96] Berkhin, P. Survey of Clustering Data Mining Techniques. Technical Report, Accrue Software, San Jose (2002). http://www.cc.gatech.edu/ isbell/reading/papers/berkhin02survey.pdf.

[97] H. I. Mary Inaba, Naoki Katoh, Proceedings of the Tenth Annual Symposium on Computational Geometry 332 (1994).

[98] D. Arthur and S. Vassilvitskii, Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics Philadelphia 1027 (2007).

[99] D. J. T. M. P. Allen, *Computer Simulations of Liquids* (Oxford Science Publications, Great Clarendon Street, Oxford OX2 6DP, 1987).

[100] A. Sebetci and G. J. O. Beran, J. Chem. Theory Comput. **6**, 155 (2010).

[101] S. Wen and G. J. O. Beran, J. Chem. Theory Comput. **7**, 3733 (2011).

[102] D. P. O. Neill, N. L. Allan, and F. R. Manby, in *Accurate Quantum Chemistry in the Condensed Phase*, edited by F. Manby (CRC Press, Boca Raton, FL, 2010), pp. 163–193.

[103] A. J. Stone and A. J. Misquitta, Int. Rev. Phys. Chem. **26**, 193 (2007).

[104] A. J. Stone, Chem. Phys. Lett. **83**, 233 (1981).

[105] A. J. Stone and M. Alderton, Mol. Phys. **56**, 1047 (1985).

[106] A. J. Stone, J. Chem. Theory Comput. **1**, 1128 (2005).

[107] C. R. LeSueur and A. J. Stone, Mol. Phys. **78**, 1267 (1993).

[108] A. J. Misquitta and A. J. Stone, J. Chem. Theory Comput. **4**, 7 (2008).

[109] A. J. Misquitta, A. J. Stone, and S. L. Price, J. Chem. Theory Comput. **4**, 19 (2008).

[110] G. J. Williams and A. J. Stone, J. Chem. Phys. **119**, 104101 (2003).

[111] A. J. Misquitta and A. J. Stone, CamCASP v5.6 (2011), `http://www-stone.ch.cam.ac.uk/programs.html`. Accessed Februrary 23, 2011.

[112] K. R. Joseph Ivanic, J. Chem. Phys. **15**, 6342 (1996).

[113] A. J. Stone, *The Theory of Intermolecular Forces* (Clarendon Press, Oxford, 2002).

[114] J. M. Olsen, K. Aidas, and J. Kongsted, J. Chem. Theory Comput. **6**, 3721–3734 (2010).

[115] K. Aidas *et al.*, WIREs: Comput. Mol. Sci. **4**, 269 (2014).

[116] J. W. Ponder, TINKER v7.1, 2015, `http://dasher.wustl.edu/tinker/`, accessed August 31, 201.

[117] W. L. Jorgensen, D. S. Maxwell, and J. Tirado-Rives, J. Am. Chem. Soc. **118**, 11225 (1996).

[118] S. Miertuš, E. Scrocco, and J. Tomasi, Chem. Phys **55**, 117 (1981).

[119] M. J. Frisch *et al.*, , gaussian Inc. Wallingford CT 2009.

[120] N. C. H. Takeshi Yanai, David P Tew, Chem. Phys. Lett. **393**, 51 (2004).

[121] T. H. Dunning, J. Chem. Phys. **90**, 1007 (1989).

[122] K. Aidas, A. Møgelhøj, E. J. K. Nilsson, M. S. Johnson, K. V. Mikkelsen, O. Christiansen, p. Söderhjelm, and J. Kongsted, J. Chem. Phys. **128**, 194503 (2008).

[123] R. D. Amos, N. C. Handy, P. J. Knowles, J. E. Rice, and A. J. Stone, J. Phys. Chem. **89**, 2186 (1985).

[124] T. Schwabe, J. M. H. Olsen, K. Sneskov, J. Kongsted, and O. Christiansen, J. Chem. Theory Comput. **7**, 2209 (2011).

[125] S. S. Leang, F. Zahariev, and M. S. Gordon, J. Chem. Phys. **136**, 104101 (2012).

[126] I. Renge, J. Phys. Chem. A **113**, 10678 (2009).

[127] A. V. Marenich, C. J. Cramer, and D. G. Truhlar, J. Chem. Soc. 1240 (1959).

[128] H. C. Börresen, Acta Chem. Scand. **17**, 921 (1963).

[129] V. Manzoni, M. L. Lyra, R. M. Gester, K. Coutinho, and S. Canuto, Phys. Chem. Chem. Phys. **12**, 14023 (2010).

[130] B. Donald and D. A. Case., Ann. Rev. Phys. Chem. **51**, 129 (2001).

[131] H. Sato, Phys. Chem. Chem. Phys. **15**, 7450 (2013).

[132] A. V. Marenich, C. J. Cramer, and D. G. Truhlar, J. Phys. Chem. B **119**, 958 (2015), pMID: 25159827.

[133] F. R. Manby, *Accurate Condensed-Phase Quantum Chemistry* (CRC Press, 6000 Broken Sound Parkway NW, Suite 300, Boca Raton, FL 33487-2742, 2011).

[134] J. Applequist, J. R. Carl, and K.-K. Fung, J. Am. Chem. Soc. **94**, 2952 (1972).

[135] A. Warshel and M. Levitt, J. Mol. Biol. **103**, 227 (1976).

[136] J. Gao, J. Comput. Chem. **18**, 1061 (1997).

[137] J. Shewchuk, Carnegie-Mellon University. Department of Computer Science (1994).

[138] J. Kolafa, J. Chem. Phys. **122**, 164105 (2005).

[139] N. Wilkins-Diehr and T. D. Crawford, Comput. Sci. Eng. **20**, 26 (2018).

[140] A. Krylov, T. L. Windus, T. Barnes, E. Marin-Rimoldi, J. A. Nash, B. Pritchard, D. G. A. Smith, D. Altarawy, P. Saxe, C. Clementi, T. D. Crawford, R. J. Harrison, S. Jha, V. S. Pande, and T. Head-Gordon, J. Chem. Phys. **149**, 180901 (2018).

Appendix A

Average Condensed Phase Environment

A.1 Multipolar and Polarizability Rotations

The averaged condensed-phase environment model explored in this work interpolates the water multipoles and polarizability parameters in a local coordinate system and then rotates them into the global coordinate frame. If we expand the local coordinates in the global coordinate the rotation matrix that takes us from the local to the global system has column vectors of the local coordinate axis. For water, we defined the Y-axis to be along the first O–H bond in global coordinates and the second O–H bond to be in the third quadrant of the XY plane of the local coordinates. A more general coordinate system might use the eigenvectors of the inertia tensor to define the local coordinate axes.

The necessary rotation matrices can be derived according to Refs [112]. The

elements of the rotation matrix D have been worked out here for rotations from the local to global coordinate system, and expressions up to rank 4 (hexadecapole) are provided. All expressions here are based on a spherical tensor formulation. The following ordering of the elements in the multipole vectors, polarizability matrices, and rotation matrices was used throughout:

- Charge: Q_{00}
- Dipole: Q_{10}, Q_{11c}, Q_{11s} (a.k.a. z, x, y)
- Quadrupole: $Q_{20}, Q_{21c}, Q_{21s}, Q_{22c}, Q_{22s}$
- Octupole: $Q_{30}, Q_{31c}, Q_{31s}, Q_{32c}, Q_{32s}, Q_{33c}, Q_{33s}$
- Hexadecapole: $Q_{40}, Q_{41c}, Q_{41s}, Q_{42c}, Q_{42s}, Q_{43c}, Q_{43s}, Q_{44c}, Q_{44s}$

The overall rotation matrix $\mathbf{D}$ for terms up to rank 4 can be written as a block-diagonal matrix, with block $\mathbf{D_l}$ corresponding to terms of rank l.

$$\mathbf{D} = \begin{pmatrix} \mathbf{D_0} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{D_1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{D_2} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{D_3} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{D_4} \end{pmatrix} \tag{A.1}$$

Multipole rotation: Rotations of the multipole moments Q from local coordinate system o to global system c involves a matrix-vector multiplication with rotation matrix $\mathbf{D}$,

$$\mathbf{Q}^g = \mathbf{D}\mathbf{Q}^l \tag{A.2}$$

In practice, this multiply can be carried out separately for each rank due to the block-diagonal structure of $\mathbf{D}$.

Polarizability rotation:

Rotations of the polarizabilities behave similarly. In this case, dipole-dipole and quadrupole-quadrupole blocks of the polarizabilities tensor can be rotated separately with the appropriate sub-block of the rotation matrix.

$$\boldsymbol{\alpha}_{\boldsymbol{l},\boldsymbol{l}}{}^g = \mathbf{D}_\mathbf{l}\boldsymbol{\alpha}_{\boldsymbol{l},\boldsymbol{l}}{}^l\mathbf{D}_\mathbf{l}{}^T \tag{A.3}$$

Rotations of the dipole-quadrupole and quadrupole-dipole block of the polarizabilities tensor are performed according to:

$$\boldsymbol{\alpha}_{\mathbf{1},\mathbf{2}}{}^g = \mathbf{D}_\mathbf{1}\boldsymbol{\alpha}_{\mathbf{1},\mathbf{2}}{}^l\mathbf{D}_\mathbf{2}{}^T \tag{A.4}$$

Since the polarizability tensor is symmetric, the remaining elements can be obtained as:

$$\boldsymbol{\alpha}_{\mathbf{2},\mathbf{1}}{}^g = (\boldsymbol{\alpha}_{\mathbf{1},\mathbf{2}}{}^l)^T \tag{A.5}$$

Rotation matrix elements: Individual elements of the rotation matrix can be computed

as follows. Elements from off-diagonal blocks not listed here are zero.

$$D(0,0) = 1 \tag{A.6}$$

$$D(1,1) = zz \tag{A.7}$$

$$D(1,2) = zx \tag{A.8}$$

$$D(1,3) = zy \tag{A.9}$$

$$D(2,1) = xz \tag{A.10}$$

$$D(2,2) = xx \tag{A.11}$$

$$D(2,3) = xy \tag{A.12}$$

$$D(3,1) = yz \tag{A.13}$$

$$D(3,2) = yx \tag{A.14}$$

$$D(3,3) = yy \tag{A.15}$$

$$D(4,4) = (3 \cdot zz^2 - 1)/2 \tag{A.16}$$

$$D(4,5) = \sqrt{3} \cdot zx \cdot zz \tag{A.17}$$

$$D(4,6) = \sqrt{3} \cdot zy \cdot zz \tag{A.18}$$

$$D(4,7) = (\sqrt{3} \cdot (-2 \cdot zy^2 - zz^2 + 1))/2 \tag{A.19}$$

$$D(4,8) = \sqrt{3} \cdot zx \cdot zy \tag{A.20}$$

$$D(5,4) = \sqrt{3} \cdot xz \cdot zz \tag{A.21}$$

$$D(5,5) = 2 \cdot xx \cdot zz - yy \tag{A.22}$$

$$D(5,6) = yx + 2 \cdot xy \cdot zz \tag{A.23}$$

$$D(5,7) = -2 \cdot xy \cdot zy - xz \cdot zz \tag{A.24}$$

$$D(5,8) = xx \cdot zy + zx \cdot xy \tag{A.25}$$

$$D(6,4) = \sqrt{3} \cdot yz \cdot zz \tag{A.26}$$

$$D(6,5) = 2 \cdot yx \cdot zz + xy \tag{A.27}$$

$$D(6,6) = -xx + 2 \cdot yy \cdot zz \tag{A.28}$$

$$D(6,7) = -2 \cdot yy \cdot zy - yz \cdot zz \tag{A.29}$$

$$D(6,8) = yx \cdot zy + zx \cdot yy \tag{A.30}$$

$$D(7,4) = (\sqrt{3} \cdot (-2 \cdot yz^2 - zz^2 + 1))/2 \tag{A.31}$$

$$D(7,5) = -2 \cdot yx \cdot yz - zx \cdot zz \tag{A.32}$$

$$D(7,6) = -2 \cdot yy \cdot yz - zy \cdot zz \tag{A.33}$$

$$D(7,7) = (4 \cdot yy^2 + 2 \cdot zy^2 + 2 \cdot yz^2 + zz^2 - 3)/2 \tag{A.34}$$

$$D(7,8) = -2 \cdot yx \cdot yy - zx \cdot zy \tag{A.35}$$

$$D(8,4) = \sqrt{3} \cdot xz \cdot yz \tag{A.36}$$

$$D(8,5) = xx \cdot yz + yx \cdot xz \tag{A.37}$$

$$D(8,6) = xy \cdot yz + yy \cdot xz \tag{A.38}$$

$$D(8,7) = -2 \cdot xy \cdot yy - xz \cdot yz \tag{A.39}$$

$$D(8,8) = xx \cdot yy + yx \cdot xy \tag{A.40}$$

$$D(9,9) = (-8 \cdot xx \cdot yy + 8 \cdot yx \cdot xy + 5 \cdot zz^3 + 5 \cdot zz)/2 \tag{A.41}$$

$$D(9,10) = (\sqrt{6} \cdot zx \cdot (5 \cdot zz^2 - 1))/4 \tag{A.42}$$

$$D(9,11) = (\sqrt{6} \cdot zy \cdot (5 \cdot zz^2 - 1))/4 \tag{A.43}$$

$$D(9,12) = (\sqrt{15} \cdot zz \cdot (-2 \cdot zy^2 - zz^2 + 1))/2 \tag{A.44}$$

$$D(9,13) = \sqrt{15} \cdot zx \cdot zy \cdot zz \tag{A.45}$$

$$D(9,14) = (\sqrt{10} \cdot zx \cdot (-4 \cdot zy^2 - zz^2 + 1))/4 \tag{A.46}$$

$$D(9,15) = (\sqrt{10} \cdot zy \cdot (-4 \cdot zy^2 - 3 \cdot zz^2 + 3))/4 \tag{A.47}$$

$$D(10,9) = (\sqrt{3} \cdot xz \cdot (5 \cdot zz^2 - 1))/(2 \cdot \sqrt{2}) \tag{A.48}$$

$$D(10,10) = (-10 \cdot xx \cdot yy^2 + 15 \cdot xx \cdot zz^2 - xx + 10 \cdot yx \cdot xy \cdot yy)/4 \tag{A.49}$$

$$D(10,11) = (10 \cdot xy \cdot yz^2 + 15 \cdot xy \cdot zz^2 - 11 \cdot xy - 10 \cdot yy \cdot xz \cdot yz)/4 \quad \text{(A.50)}$$

$$D(10,12) = (\sqrt{10} \cdot (4 \cdot xy \cdot yy \cdot yz - 4 \cdot yy^2 \cdot xz - 6 \cdot zy^2 \cdot xz - 3 \cdot xz \cdot zz^2 + 5 \cdot xz))/4 \quad \text{(A.51)}$$

$$D(10,13) = \sqrt{10} \cdot (-xx \cdot yy \cdot yz - yx \cdot xy \cdot yz + 2 \cdot yx \cdot yy \cdot xz + 3 \cdot zx \cdot zy \cdot xz)/2 \quad \text{(A.52)}$$

$$D(10,14) = (\sqrt{15} \cdot (-2 \cdot xx \cdot yy^2 - 4 \cdot xx \cdot zy^2 - xx \cdot zz^2 + 3 \cdot xx + 2 \cdot yx \cdot xy \cdot yy))/4 \quad \text{(A.53)}$$

$$D(10,15) = (\sqrt{15} \cdot (-4 \cdot xy \cdot zy^2 - 2 \cdot xy \cdot yz^2 - 3 \cdot xy \cdot zz^2 + 3 \cdot xy + 2 \cdot yy \cdot xz \cdot yz))/4 \quad \text{(A.54)}$$

$$D(11,9) = (\sqrt{3} \cdot yz \cdot (5 \cdot zz^2 - 1))/(2 \cdot \sqrt{2}) \quad \text{(A.55)}$$

$$D(11,10) = (10 \cdot yx \cdot zy^2 + 15 \cdot yx \cdot zz^2 - 11 \cdot yx - 10 \cdot zx \cdot yy \cdot zy)/4 \quad \text{(A.56)}$$

$$D(11,11) = (5 \cdot yy \cdot zz^2 - yy + 10 \cdot zy \cdot yz \cdot zz)/4 \quad \text{(A.57)}$$

$$D(11,12) = (\sqrt{10} \cdot (-4 \cdot yy \cdot zy \cdot zz - 2 \cdot zy^2 \cdot yz - 3 \cdot yz \cdot zz^2 + yz))/4 \quad \text{(A.58)}$$

$$D(11,13) = (\sqrt{10} \cdot (yx \cdot zy \cdot zz + zx \cdot yy \cdot zz + zx \cdot zy \cdot yz))/2 \quad \text{(A.59)}$$

$$D(11,14) = (\sqrt{15} \cdot (-2 \cdot yx \cdot zy^2 - yx \cdot zz^2 + yx - 2 \cdot zx \cdot yy \cdot zy))/4 \quad \text{(A.60)}$$

$$D(11,15) = (\sqrt{15} \cdot (-4 \cdot yy \cdot zy^2 - yy \cdot zz^2 + yy - 2 \cdot zy \cdot yz \cdot zz))/4 \tag{A.61}$$

$$D(12,9) = \sqrt{15} \cdot zz \cdot (-2 \cdot yz^2 - zz^2 + 1)/2 \tag{A.62}$$

$$D(12,10) = (\sqrt{10} \cdot (4 \cdot yx \cdot yy \cdot zy - 4 \cdot zx \cdot yy^2 - 6 \cdot zx \cdot yz^2 - 3 \cdot zx \cdot zz^2 + 5 \cdot zx))/4 \tag{A.63}$$

$$D(12,11) = (\sqrt{10} \cdot (-4 \cdot yy \cdot yz \cdot zz - 2 \cdot zy \cdot yz^2 - 3 \cdot zy \cdot zz^2 + zy))/4 \tag{A.64}$$

$$D(12,12) = (-4 \cdot xx \cdot yy - 4 \cdot yx \cdot xy + 12 \cdot yy^2 \cdot zz + 6 \cdot zy^2 \cdot zz + 6 \cdot yz^2 \cdot zz + 3 \cdot zz^3 - 9 \cdot zz)/2 \tag{A.65}$$

$$D(12,13) = -6 \cdot yx \cdot yy \cdot zz - 3 \cdot zx \cdot zy \cdot zz - 4 \cdot xy \cdot yy - 2 \cdot xz \cdot yz \tag{A.66}$$

$$D(12,14) = (\sqrt{6} \cdot (4 \cdot yx \cdot yy \cdot zy + 4 \cdot zx \cdot yy^2 + 4 \cdot zx \cdot zy^2 + 2 \cdot zx \cdot yz^2 + zx \cdot zz^2 - 3 \cdot zx))/4 \tag{A.67}$$

$$D(12,15) = (\sqrt{6} \cdot (8 \cdot yy^2 \cdot zy + 4 \cdot yy \cdot yz \cdot zz + 4 \cdot zy^3 + 2 \cdot zy \cdot yz^2 + 3 \cdot zy \cdot zz^2 - 5 \cdot zy))/4 \tag{A.68}$$

$$D(13,9) = \sqrt{15} \cdot xz \cdot yz \cdot zz \tag{A.69}$$

$$D(13,10) = (\sqrt{10} \cdot (-xx \cdot yy \cdot zy - yx \cdot xy \cdot zy + 2 \cdot zx \cdot xy \cdot yy + 3 \cdot zx \cdot xz \cdot yz))/2 \tag{A.70}$$

$$D(13,11) = (\sqrt{10} \cdot (xy \cdot yz \cdot zz + yy \cdot xz \cdot zz + zy \cdot xz \cdot yz))/2 \tag{A.71}$$

$$D(13,12) = -4 \cdot yx \cdot yy - 2 \cdot zx \cdot zy - 6 \cdot xy \cdot yy \cdot zz - 3 \cdot xz \cdot yz \cdot zz \tag{A.72}$$

$$D(13,13) = 3 \cdot xx \cdot yy \cdot zz + 3 \cdot yx \cdot xy \cdot zz - 4 \cdot yy^2 - 2 \cdot zy^2 - 2 \cdot yz^2 - zz^2 + 3 \tag{A.73}$$

$$D(13,14) = (\sqrt{6} \cdot (-xx \cdot yy \cdot zy - yx \cdot xy \cdot zy - 2 \cdot zx \cdot xy \cdot yy - zx \cdot xz \cdot yz))/2 \tag{A.74}$$

$$D(13,15) = (\sqrt{6} \cdot (-4 \cdot xy \cdot yy \cdot zy - xy \cdot yz \cdot zz - yy \cdot xz \cdot zz - zy \cdot xz \cdot yz))/2 \tag{A.75}$$

$$D(14,9) = (\sqrt{5} \cdot xz \cdot (-4 \cdot yz^2 - zz^2 + 1))/(2 \cdot \sqrt{2}) \tag{A.76}$$

$$D(14,10) = (\sqrt{15} \cdot (-2 \cdot xx \cdot yy^2 - 4 \cdot xx \cdot yz^2 - xx \cdot zz^2 + 3 \cdot xx + 2 \cdot yx \cdot xy \cdot yy))/4 \tag{A.77}$$

$$D(14,11) = (\sqrt{15} \cdot (-2 \cdot xy \cdot yz^2 - xy \cdot zz^2 + xy - 2 \cdot yy \cdot xz \cdot yz))/4 \tag{A.78}$$

$$D(14,12) = (\sqrt{6} \cdot (4 \cdot xy \cdot yy \cdot yz + 4 \cdot yy^2 \cdot xz + 2 \cdot zy^2 \cdot xz + 4 \cdot xz \cdot yz^2 + xz \cdot zz^2 - 3 \cdot xz))/4 \tag{A.79}$$

$$D(14,13) = (\sqrt{6} \cdot (-xx \cdot yy \cdot yz - yx \cdot xy \cdot yz - 2 \cdot yx \cdot yy \cdot xz - zx \cdot zy \cdot xz))/2 \tag{A.80}$$

$$D(14,14) = (10 \cdot xx \cdot yy^2 + 4 \cdot xx \cdot zy^2 + 4 \cdot xx \cdot yz^2 + xx \cdot zz^2 - 7 \cdot xx + 6 \cdot yx \cdot xy \cdot yy)/4 \tag{A.81}$$

$$D(14,15) = (16 \cdot xy \cdot yy^2 + 4 \cdot xy \cdot zy^2 + 6 \cdot xy \cdot yz^2 + 3 \cdot xy \cdot zz^2 - 7 \cdot xy + 6 \cdot yy \cdot xz \cdot yz)/4 \quad \text{(A.82)}$$

$$D(15,9) = (\sqrt{5} \cdot yz \cdot (-4 \cdot yz^2 - 3 \cdot zz^2 + 3))/(2 \cdot \sqrt{2}) \quad \text{(A.83)}$$

$$D(15,10) = (\sqrt{15} \cdot (-2 \cdot yx \cdot zy^2 - 4 \cdot yx \cdot yz^2 - 3 \cdot yx \cdot zz^2 + 3 \cdot yx + 2 \cdot zx \cdot yy \cdot zy))/4 \quad \text{(A.84)}$$

$$D(15,11) = (\sqrt{15} \cdot (-4 \cdot yy \cdot yz^2 - yy \cdot zz^2 + yy - 2 \cdot zy \cdot yz \cdot zz))/4 \quad \text{(A.85)}$$

$$D(15,12) = (\sqrt{6} \cdot (8 \cdot yy^2 \cdot yz + 4 \cdot yy \cdot zy \cdot zz + 2 \cdot zy^2 \cdot yz + 4 \cdot yz^3 + 3 \cdot yz \cdot zz^2 - 5 \cdot yz))/4 \quad \text{(A.86)}$$

$$D(15,13) = (\sqrt{6} \cdot (-4 \cdot yx \cdot yy \cdot yz - yx \cdot zy \cdot zz - zx \cdot yy \cdot zz - zx \cdot zy \cdot yz))/2 \quad \text{(A.87)}$$

$$D(15,14) = (16 \cdot yx \cdot yy^2 + 6 \cdot yx \cdot zy^2 + 4 \cdot yx \cdot yz^2 + 3 \cdot yx \cdot zz^2 - 7 \cdot yx + 6 \cdot zx \cdot yy \cdot zy)/4 \quad \text{(A.88)}$$

$$D(15,15) = (16 \cdot yy^3 + 12 \cdot yy \cdot zy^2 + 12 \cdot yy \cdot yz^2 + 3 \cdot yy \cdot zz^2 - 15 \cdot yy + 6 \cdot zy \cdot yz \cdot zz)/4 \quad \text{(A.89)}$$

$$D(16,16) = (-68 \cdot yy^2 \cdot zz^2 + 68 \cdot yy^2 + 136 \cdot yy \cdot zy \cdot yz \cdot zz - 68 \cdot zy^2 \cdot yz^2 + 68 \cdot zy^2 + 68 \cdot yz^2 + 35 \cdot zz^4 + 38 \cdot zz^2 - 65)/8 \quad \text{(A.90)}$$

$$D(16,17) = (\sqrt{10} \cdot (10 \cdot yx \cdot yy \cdot zy \cdot zz - 10 \cdot yx \cdot zy^2 \cdot yz + 10 \cdot yx \cdot yz - 10 \cdot zx \cdot yy^2 \cdot zz + 10 \cdot zx \cdot yy \cdot zy \cdot yz + 7 \cdot zx \cdot zz^3 + 7 \cdot zx \cdot zz))/4 \quad \text{(A.91)}$$

$$D(16,18) = (\sqrt{10} \cdot zy \cdot zz \cdot (7 \cdot zz^2 - 3))/4 \tag{A.92}$$

$$D(16,19) = (\sqrt{5} \cdot (-14 \cdot zy^2 \cdot zz^2 + 2 \cdot zy^2 - 7 \cdot zz^4 + 8 \cdot zz^2 - 1))/4 \tag{A.93}$$

$$D(16,20) = (\sqrt{5} \cdot zx \cdot zy \cdot (7 \cdot zz^2 - 1))/2 \tag{A.94}$$

$$\begin{aligned} D(16,21) = (\sqrt{70} \cdot (2 \cdot yx \cdot yy \cdot zy \cdot zz - 2 \cdot yx \cdot zy^2 \cdot yz + 2 \cdot yx \cdot yz - 2 \cdot zx \cdot yy^2 \cdot zz \\ + 2 \cdot zx \cdot yy \cdot zy \cdot yz - 4 \cdot zx \cdot zy^2 \cdot zz - zx \cdot zz^3 + 3 \cdot zx \cdot zz))/4 \end{aligned} \tag{A.95}$$

$$D(16,22) = (\sqrt{70} \cdot zy \cdot zz \cdot (-4 \cdot zy^2 - 3 \cdot zz^2 + 3))/4 \tag{A.96}$$

$$\begin{aligned} D(16,23) = (\sqrt{35} \cdot (4 \cdot yy^2 \cdot zz^2 - 4 \cdot yy^2 - 8 \cdot yy \cdot zy \cdot yz \cdot zz + 8 \cdot zy^4 + 4 \cdot zy^2 \cdot yz^2 \\ + 8 \cdot zy^2 \cdot zz^2 - 12 \cdot zy^2 - 4 \cdot yz^2 + zz^4 - 6 \cdot zz^2 + 5))/8 \end{aligned} \tag{A.97}$$

$$D(16,24) = (\sqrt{35} \cdot zx \cdot zy \cdot (-2 \cdot zy^2 - zz^2 + 1))/2 \tag{A.98}$$

$$\begin{aligned} D(17,16) = (\sqrt{5} \cdot (10 \cdot xy \cdot yy \cdot yz \cdot zz - 10 \cdot xy \cdot zy \cdot yz^2 + 10 \cdot xy \cdot zy - 10 \cdot yy^2 \cdot xz \\ \cdot zz + 10 \cdot yy \cdot zy \cdot xz \cdot yz + 7 \cdot xz \cdot zz^3 + 7 \cdot xz \cdot zz))/(2 \cdot \sqrt{2}) \end{aligned} \tag{A.99}$$

$$\begin{aligned} D(17,17) = (-24 \cdot xx \cdot yy^2 \cdot zz + 28 \cdot xx \cdot zz^3 + 28 \cdot xx \cdot zz + 24 \cdot yx \cdot xy \cdot yy \cdot zz \\ - 31 \cdot yy \cdot zz^2 + 3 \cdot yy + 34 \cdot zy \cdot yz \cdot zz)/4 \end{aligned} \tag{A.100}$$

$$D(17,18) = (-34 \cdot yx \cdot zy^2 - 3 \cdot yx \cdot zz^2 + 31 \cdot yx + 34 \cdot zx \cdot yy \cdot zy + 24 \cdot xy \cdot yz^2 \cdot zz + 28 \cdot xy \cdot zz^3 + 4 \cdot xy \cdot zz - 24 \cdot yy \cdot xz \cdot yz \cdot zz)/4 \quad \text{(A.101)}$$

$$D(17,19) = (\sqrt{2} \cdot (4 \cdot xy \cdot yy \cdot yz \cdot zz + 4 \cdot xy \cdot zy \cdot yz^2 + xy \cdot zy \cdot zz^2 - 3 \cdot xy \cdot zy - 4 \cdot yy^2 \cdot xz \cdot zz - 4 \cdot yy \cdot zy \cdot xz \cdot yz - 15 \cdot zy^2 \cdot xz \cdot zz - 7 \cdot xz \cdot zz^3 + 8 \cdot xz \cdot zz))/2 \quad \text{(A.102)}$$

$$D(17,20) = (\sqrt{2} \cdot (-22 \cdot xx \cdot yy^2 \cdot zy - 30 \cdot xx \cdot zy^3 - xx \cdot zy \cdot zz^2 + 29 \cdot xx \cdot zy + 22 \cdot yx \cdot xy \cdot yy \cdot zy + 30 \cdot zx \cdot xy \cdot zy^2 + 22 \cdot zx \cdot xy \cdot yz^2 + 29 \cdot zx \cdot xy \cdot zz^2 - 23 \cdot zx \cdot xy - 22 \cdot zx \cdot yy \cdot xz \cdot yz))/4 \quad \text{(A.103)}$$

$$D(17,21) = (\sqrt{7} \cdot (-8 \cdot xx \cdot yy^2 \cdot zz - 16 \cdot xx \cdot zy^2 \cdot zz - 4 \cdot xx \cdot zz^3 + 12 \cdot xx \cdot zz + 8 \cdot yx \cdot xy \cdot yy \cdot zz + 4 \cdot yy \cdot zy^2 + yy \cdot zz^2 - yy + 2 \cdot zy \cdot yz \cdot zz))/4 \quad \text{(A.104)}$$

$$D(17,22) = (\sqrt{7} \cdot (-2 \cdot yx \cdot zy^2 - yx \cdot zz^2 + yx - 2 \cdot zx \cdot yy \cdot zy - 16 \cdot xy \cdot zy^2 \cdot zz - 8 \cdot xy \cdot yz^2 \cdot zz - 12 \cdot xy \cdot zz^3 + 12 \cdot xy \cdot zz + 8 \cdot yy \cdot xz \cdot yz \cdot zz))/4 \quad \text{(A.105)}$$

$$D(17,23) = (\sqrt{14} \cdot (-2 \cdot xy \cdot yy \cdot yz \cdot zz + 8 \cdot xy \cdot zy^3 + 2 \cdot xy \cdot zy \cdot yz^2 + 4 \cdot xy \cdot zy \cdot zz^2 - 6 \cdot xy \cdot zy + 2 \cdot yy^2 \cdot xz \cdot zz - 2 \cdot yy \cdot zy \cdot xz \cdot yz + 4 \cdot zy^2 \cdot xz \cdot zz + xz \cdot zz^3 - 3 \cdot xz \cdot zz))/4 \quad \text{(A.106)}$$

$$D(17,24) = (\sqrt{14} \cdot (-2 \cdot xx \cdot yy^2 \cdot zy - 4 \cdot xx \cdot zy^3 - xx \cdot zy \cdot zz^2 + 3 \cdot xx \cdot zy + 2 \cdot yx \cdot xy \cdot yy \cdot zy - 4 \cdot zx \cdot xy \cdot zy^2 - 2 \cdot zx \cdot xy \cdot yz^2 - 3 \cdot zx \cdot xy \cdot zz^2 + 3 \cdot zx \cdot xy + 2 \cdot zx \cdot yy \cdot xz \cdot yz))/4 \quad \text{(A.107)}$$

$$D(18,16) = (\sqrt{5} \cdot yz \cdot zz \cdot (7 \cdot zz^2 - 3))/(2 \cdot \sqrt{2}) \tag{A.108}$$

$$D(18,17) = (24 \cdot yx \cdot zy^2 \cdot zz + 28 \cdot yx \cdot zz^3 + 4 \cdot yx \cdot zz - 24 \cdot zx \cdot yy \cdot zy \cdot zz - 34 \cdot xy \cdot yz^2 - 3 \cdot xy \cdot zz^2 + 31 \cdot xy + 34 \cdot yy \cdot xz \cdot yz)/4 \tag{A.109}$$

$$D(18,18) = (-34 \cdot xx \cdot yy^2 + 3 \cdot xx \cdot zz^2 + 3 \cdot xx + 34 \cdot yx \cdot xy \cdot yy + 4 \cdot yy \cdot zz^3 + 28 \cdot yy \cdot zz + 24 \cdot zy \cdot yz \cdot zz^2)/4 \tag{A.110}$$

$$D(18,19) = (\sqrt{2} \cdot (-7 \cdot yy \cdot zy \cdot zz^2 + yy \cdot zy - 7 \cdot zy^2 \cdot yz \cdot zz - 7 \cdot yz \cdot zz^3 + 4 \cdot yz \cdot zz))/2 \tag{A.111}$$

$$D(18,20) = (\sqrt{2} \cdot (-8 \cdot yx \cdot zy^3 - yx \cdot zy \cdot zz^2 + 7 \cdot yx \cdot zy + 8 \cdot zx \cdot yy \cdot zy^2 + 7 \cdot zx \cdot yy \cdot zz^2 - zx \cdot yy + 22 \cdot zx \cdot zy \cdot yz \cdot zz))/4 \tag{A.112}$$

$$D(18,21) = (\sqrt{7} \cdot (-8 \cdot yx \cdot zy^2 \cdot zz - 4 \cdot yx \cdot zz^3 + 4 \cdot yx \cdot zz - 8 \cdot zx \cdot yy \cdot zy \cdot zz - 4 \cdot xy \cdot zy^2 - 2 \cdot xy \cdot yz^2 - 3 \cdot xy \cdot zz^2 + 3 \cdot xy + 2 \cdot yy \cdot xz \cdot yz))/4 \tag{A.113}$$

$$D(18,22) = (\sqrt{7} \cdot (2 \cdot xx \cdot yy^2 + 4 \cdot xx \cdot zy^2 + xx \cdot zz^2 - 3 \cdot xx - 2 \cdot yx \cdot xy \cdot yy - 16 \cdot yy \cdot zy^2 \cdot zz - 4 \cdot yy \cdot zz^3 + 4 \cdot yy \cdot zz - 8 \cdot zy \cdot yz \cdot zz^2))/4 \tag{A.114}$$

$$D(18,23) = (\sqrt{14} \cdot (8 \cdot yy \cdot zy^3 + 4 \cdot yy \cdot zy \cdot zz^2 - 4 \cdot yy \cdot zy + 4 \cdot zy^2 \cdot yz \cdot zz + yz \cdot zz^3 - yz \cdot zz))/4 \tag{A.115}$$

$$D(18,24) = (\sqrt{14} \cdot (-2 \cdot yx \cdot zy^3 - yx \cdot zy \cdot zz^2 + yx \cdot zy - 6 \cdot zx \cdot yy \cdot zy^2 - zx \cdot yy \cdot zz^2 + zx \cdot yy - 2 \cdot zx \cdot zy \cdot yz \cdot zz))/4 \quad \text{(A.116)}$$

$$D(19,16) = \sqrt{5} \cdot (-14 \cdot yz^2 \cdot zz^2 + 2 \cdot yz^2 - 7 \cdot zz^4 + 8 \cdot zz^2 - 1)/4 \quad \text{(A.117)}$$

$$D(19,17) = (\sqrt{2} \cdot (4 \cdot yx \cdot yy \cdot zy \cdot zz + 4 \cdot yx \cdot zy^2 \cdot yz + yx \cdot yz \cdot zz^2 - 3 \cdot yx \cdot yz - 4 \cdot zx \cdot yy^2 \cdot zz - 4 \cdot zx \cdot yy \cdot zy \cdot yz - 15 \cdot zx \cdot yz^2 \cdot zz - 7 \cdot zx \cdot zz^3 + 8 \cdot zx \cdot zz))/2 \quad \text{(A.118)}$$

$$D(19,18) = (\sqrt{2} \cdot (-7 \cdot yy \cdot yz \cdot zz^2 + yy \cdot yz - 7 \cdot zy \cdot yz^2 \cdot zz - 7 \cdot zy \cdot zz^3 + 4 \cdot zy \cdot zz))/2 \quad \text{(A.119)}$$

$$D(19,19) = (14 \cdot yy^2 \cdot zz^2 - 10 \cdot yy^2 + 14 \cdot zy^2 \cdot yz^2 + 14 \cdot zy^2 \cdot zz^2 - 12 \cdot zy^2 + 14 \cdot yz^2 \cdot zz^2 - 12 \cdot yz^2 + 7 \cdot zz^4 - 20 \cdot zz^2 + 11)/2 \quad \text{(A.120)}$$

$$D(19,20) = -7 \cdot yx \cdot yy \cdot zz^2 + 5 \cdot yx \cdot yy - 7 \cdot zx \cdot zy \cdot yz^2 - 7 \cdot zx \cdot zy \cdot zz^2 + 6 \cdot zx \cdot zy \quad \text{(A.121)}$$

$$D(19,21) = (\sqrt{14} \cdot (4 \cdot yx \cdot zy^2 \cdot yz + yx \cdot yz \cdot zz^2 - 3 \cdot yx \cdot yz + 4 \cdot zx \cdot yy^2 \cdot zz + 4 \cdot zx \cdot zy^2 \cdot zz + zx \cdot yz^2 \cdot zz + zx \cdot zz^3 - 4 \cdot zx \cdot zz))/2 \quad \text{(A.122)}$$

$$D(19,22) = (\sqrt{14} \cdot (4 \cdot yy^2 \cdot zy \cdot zz + 4 \cdot yy \cdot zy^2 \cdot yz + 3 \cdot yy \cdot yz \cdot zz^2 - yy \cdot yz + 4 \cdot zy^3 \cdot zz + 3 \cdot zy \cdot yz^2 \cdot zz + 3 \cdot zy \cdot zz^3 - 4 \cdot zy \cdot zz))/2 \quad \text{(A.123)}$$

$$D(19,23) = (\sqrt{7} \cdot (-16 \cdot yy^2 \cdot zy^2 - 8 \cdot yy^2 \cdot zz^2 + 8 \cdot yy^2 - 8 \cdot zy^4 - 8 \cdot zy^2 \cdot yz^2 - 8 \cdot zy^2 \cdot zz^2 + 16 \cdot zy^2 - 2 \cdot yz^2 \cdot zz^2 + 6 \cdot yz^2 - zz^4 + 8 \cdot zz^2 - 7))/4 \tag{A.124}$$

$$D(19,24) = \sqrt{7} \cdot (2 \cdot yx \cdot yy \cdot zy^2 + yx \cdot yy \cdot zz^2 - yx \cdot yy + 2 \cdot zx \cdot yy^2 \cdot zy + 2 \cdot zx \cdot zy^3 + zx \cdot zy \cdot yz^2 + zx \cdot zy \cdot zz^2 - 2 \cdot zx \cdot zy) \tag{A.125}$$

$$D(20,16) = \sqrt{5} \cdot xz \cdot yz \cdot (7 \cdot zz^2 - 1)/2 \tag{A.126}$$

$$D(20,17) = (\sqrt{2} \cdot (-22 \cdot xx \cdot yy^2 \cdot yz - 30 \cdot xx \cdot yz^3 - xx \cdot yz \cdot zz^2 + 29 \cdot xx \cdot yz + 22 \cdot yx \cdot xy \cdot yy \cdot yz + 22 \cdot yx \cdot zy^2 \cdot xz + 30 \cdot yx \cdot xz \cdot yz^2 + 29 \cdot yx \cdot xz \cdot zz^2 - 23 \cdot yx \cdot xz - 22 \cdot zx \cdot yy \cdot zy \cdot xz))/4 \tag{A.127}$$

$$D(20,18) = (\sqrt{2} \cdot (14 \cdot xy \cdot yz^3 + 21 \cdot xy \cdot yz \cdot zz^2 - 15 \cdot xy \cdot yz - 14 \cdot yy \cdot xz \cdot yz^2 + 7 \cdot yy \cdot xz \cdot zz^2 - yy \cdot xz))/4 \tag{A.128}$$

$$D(20,19) = -7 \cdot xy \cdot yy \cdot zz^2 + 5 \cdot xy \cdot yy - 7 \cdot zy^2 \cdot xz \cdot yz - 7 \cdot xz \cdot yz \cdot zz^2 + 6 \cdot xz \cdot yz \tag{A.129}$$

$$D(20,20) = (-28 \cdot xx \cdot yy^3 - 14 \cdot xx \cdot yy \cdot zy^2 - 14 \cdot xx \cdot yy \cdot yz^2 + 7 \cdot xx \cdot yy \cdot zz^2 + 23 \cdot xx \cdot yy + 28 \cdot yx \cdot xy \cdot yy^2 + 14 \cdot yx \cdot xy \cdot zy^2 + 14 \cdot yx \cdot xy \cdot yz^2 + 21 \cdot yx \cdot xy \cdot zz^2 - 19 \cdot yx \cdot xy)/2 \tag{A.130}$$

$$D(20,21) = (\sqrt{14} \cdot (2 \cdot xx \cdot yy^2 \cdot yz - 4 \cdot xx \cdot zy^2 \cdot yz + 2 \cdot xx \cdot yz^3 - xx \cdot yz \cdot zz^2 + xx \cdot yz + 6 \cdot yx \cdot xy \cdot yy \cdot yz - 8 \cdot yx \cdot yy^2 \cdot xz - 6 \cdot yx \cdot zy^2 \cdot xz - 2 \cdot yx \cdot xz \cdot yz^2 - 3 \cdot yx \cdot xz \cdot zz^2 + 5 \cdot yx \cdot xz - 6 \cdot zx \cdot yy \cdot zy \cdot xz))/4 \tag{A.131}$$

$$D(20,22) = (\sqrt{14} \cdot (8 \cdot xy \cdot yy^2 \cdot yz - 4 \cdot xy \cdot zy^2 \cdot yz - 6 \cdot xy \cdot yz^3 - 9 \cdot xy \cdot yz \cdot zz^2 + 7 \cdot xy \cdot yz - 8 \cdot yy^3 \cdot xz - 12 \cdot yy \cdot zy^2 \cdot xz + 6 \cdot yy \cdot xz \cdot yz^2 - 3 \cdot yy \cdot xz \cdot zz^2 + 9 \cdot yy \cdot xz))/4 \tag{A.132}$$

$$D(20,23) = (\sqrt{7} \cdot (8 \cdot xy \cdot yy \cdot zy^2 + 4 \cdot xy \cdot yy \cdot zz^2 - 4 \cdot xy \cdot yy + 4 \cdot zy^2 \cdot xz \cdot yz + xz \cdot yz \cdot zz^2 - 3 \cdot xz \cdot yz))/2 \tag{A.133}$$

$$D(20,24) = (\sqrt{7} \cdot (-4 \cdot xx \cdot yy \cdot zy^2 + 2 \cdot xx \cdot yy \cdot yz^2 - xx \cdot yy \cdot zz^2 + xx \cdot yy - 4 \cdot yx \cdot xy \cdot zy^2 - 2 \cdot yx \cdot xy \cdot yz^2 - 3 \cdot yx \cdot xy \cdot zz^2 + 3 \cdot yx \cdot xy))/2 \tag{A.134}$$

$$D(21,16) = (\sqrt{35} \cdot (2 \cdot xy \cdot yy \cdot yz \cdot zz - 2 \cdot xy \cdot zy \cdot yz^2 + 2 \cdot xy \cdot zy - 2 \cdot yy^2 \cdot xz \cdot zz + 2 \cdot yy \cdot zy \cdot xz \cdot yz - 4 \cdot xz \cdot yz^2 \cdot zz - xz \cdot zz^3 + 3 \cdot xz \cdot zz))/(2 \cdot \sqrt{2}) \tag{A.135}$$

$$D(21,17) = (\sqrt{7} \cdot (-8 \cdot xx \cdot yy^2 \cdot zz - 16 \cdot xx \cdot yz^2 \cdot zz - 4 \cdot xx \cdot zz^3 + 12 \cdot xx \cdot zz + 8 \cdot yx \cdot xy \cdot yy \cdot zz + 4 \cdot yy \cdot yz^2 + yy \cdot zz^2 - yy + 2 \cdot zy \cdot yz \cdot zz))/4 \tag{A.136}$$

$$D(21,18) = (\sqrt{7} \cdot (-2 \cdot yx \cdot zy^2 - 4 \cdot yx \cdot yz^2 - 3 \cdot yx \cdot zz^2 + 3 \cdot yx + 2 \cdot zx \cdot yy \cdot zy - 8 \cdot xy \cdot yz^2 \cdot zz - 4 \cdot xy \cdot zz^3 + 4 \cdot xy \cdot zz - 8 \cdot yy \cdot xz \cdot yz \cdot zz))/4 \tag{A.137}$$

$$D(21,19) = (\sqrt{14} \cdot (4 \cdot xy \cdot zy \cdot yz^2 + xy \cdot zy \cdot zz^2 - 3 \cdot xy \cdot zy + 4 \cdot yy^2 \cdot xz \cdot zz + zy^2 \cdot xz \cdot zz + 4 \cdot xz \cdot yz^2 \cdot zz + xz \cdot zz^3 - 4 \cdot xz \cdot zz))/2 \tag{A.138}$$

$$D(21,20) = (\sqrt{14} \cdot (2 \cdot xx \cdot yy^2 \cdot zy + 2 \cdot xx \cdot zy^3 - 4 \cdot xx \cdot zy \cdot yz^2 - xx \cdot zy \cdot zz^2 + xx \cdot zy + 6 \cdot yx \cdot xy \cdot yy \cdot zy - 8 \cdot zx \cdot xy \cdot yy^2 - 2 \cdot zx \cdot xy \cdot zy^2 - 6 \cdot zx \cdot xy \cdot yz^2 - 3 \cdot zx \cdot xy \cdot zz^2 + 5 \cdot zx \cdot xy - 6 \cdot zx \cdot yy \cdot xz \cdot yz))/4 \tag{A.139}$$

$$D(21,21) = (40 \cdot xx \cdot yy^2 \cdot zz + 16 \cdot xx \cdot zy^2 \cdot zz + 16 \cdot xx \cdot yz^2 \cdot zz + 4 \cdot xx \cdot zz^3 - 28 \cdot xx \cdot zz + 24 \cdot yx \cdot xy \cdot yy \cdot zz - 48 \cdot yy^3 - 36 \cdot yy \cdot zy^2 - 36 \cdot yy \cdot yz^2 - 9 \cdot yy \cdot zz^2 + 45 \cdot yy - 18 \cdot zy \cdot yz \cdot zz)/4 \tag{A.140}$$

$$D(21,22) = (48 \cdot yx \cdot yy^2 + 18 \cdot yx \cdot zy^2 + 12 \cdot yx \cdot yz^2 + 9 \cdot yx \cdot zz^2 - 21 \cdot yx + 18 \cdot zx \cdot yy \cdot zy + 64 \cdot xy \cdot yy^2 \cdot zz + 16 \cdot xy \cdot zy^2 \cdot zz + 24 \cdot xy \cdot yz^2 \cdot zz + 12 \cdot xy \cdot zz^3 - 28 \cdot xy \cdot zz + 24 \cdot yy \cdot xz \cdot yz \cdot zz)/4 \tag{A.141}$$

$$D(21,23) = (\sqrt{2} \cdot (-32 \cdot xy \cdot yy^2 \cdot zy - 6 \cdot xy \cdot yy \cdot yz \cdot zz - 8 \cdot xy \cdot zy^3 - 10 \cdot xy \cdot zy \cdot yz^2 - 4 \cdot xy \cdot zy \cdot zz^2 + 14 \cdot xy \cdot zy - 10 \cdot yy^2 \cdot xz \cdot zz - 6 \cdot yy \cdot zy \cdot xz \cdot yz - 4 \cdot zy^2 \cdot xz \cdot zz - 4 \cdot xz \cdot yz^2 \cdot zz - xz \cdot zz^3 + 7 \cdot xz \cdot zz))/4 \tag{A.142}$$

$$D(21,24) = (\sqrt{2} \cdot (10 \cdot xx \cdot yy^2 \cdot zy + 4 \cdot xx \cdot zy^3 + 4 \cdot xx \cdot zy \cdot yz^2 + xx \cdot zy \cdot zz^2 - 7 \cdot xx \cdot zy + 6 \cdot yx \cdot xy \cdot yy \cdot zy + 16 \cdot zx \cdot xy \cdot yy^2 + 4 \cdot zx \cdot xy \cdot zy^2 + 6 \cdot zx \cdot xy \cdot yz^2 + 3 \cdot zx \cdot xy \cdot zz^2 - 7 \cdot zx \cdot xy + 6 \cdot zx \cdot yy \cdot xz \cdot yz))/4 \tag{A.143}$$

$$D(22,16) = (\sqrt{35} \cdot yz \cdot zz \cdot (-4 \cdot yz^2 - 3 \cdot zz^2 + 3))/(2 \cdot \sqrt{2}) \tag{A.144}$$

$$D(22,17) = (\sqrt{7} \cdot (-8 \cdot yx \cdot zy^2 \cdot zz - 16 \cdot yx \cdot yz^2 \cdot zz - 12 \cdot yx \cdot zz^3 + 12 \cdot yx \cdot zz + 8 \cdot zx \cdot yy \cdot zy \cdot zz - 2 \cdot xy \cdot yz^2 - xy \cdot zz^2 + xy - 2 \cdot yy \cdot xz \cdot yz))/4 \tag{A.145}$$

$$D(22,18) = (\sqrt{7} \cdot (2 \cdot xx \cdot yy^2 + 4 \cdot xx \cdot yz^2 + xx \cdot zz^2 - 3 \cdot xx - 2 \cdot yx \cdot xy \cdot yy - 16 \cdot yy \cdot yz^2 \cdot zz - 4 \cdot yy \cdot zz^3 + 4 \cdot yy \cdot zz - 8 \cdot zy \cdot yz \cdot zz^2))/4 \tag{A.146}$$

$$D(22,19) = (\sqrt{14} \cdot (4 \cdot yy^2 \cdot yz \cdot zz + 4 \cdot yy \cdot zy \cdot yz^2 + 3 \cdot yy \cdot zy \cdot zz^2 - yy \cdot zy + 3 \cdot zy^2 \cdot yz \cdot zz + 4 \cdot yz^3 \cdot zz + 3 \cdot yz \cdot zz^3 - 4 \cdot yz \cdot zz))/2 \tag{A.147}$$

$$D(22,20) = (\sqrt{14} \cdot (8 \cdot yx \cdot yy^2 \cdot zy - 4 \cdot yx \cdot zy \cdot yz^2 - 3 \cdot yx \cdot zy \cdot zz^2 + yx \cdot zy - 8 \cdot zx \cdot yy^3 - 12 \cdot zx \cdot yy \cdot yz^2 - 3 \cdot zx \cdot yy \cdot zz^2 + 9 \cdot zx \cdot yy - 6 \cdot zx \cdot zy \cdot yz \cdot zz))/4 \tag{A.148}$$

$$D(22,21) = (64 \cdot yx \cdot yy^2 \cdot zz + 24 \cdot yx \cdot zy^2 \cdot zz + 16 \cdot yx \cdot yz^2 \cdot zz + 12 \cdot yx \cdot zz^3 - 28 \cdot yx \cdot zz + 24 \cdot zx \cdot yy \cdot zy \cdot zz + 48 \cdot xy \cdot yy^2 + 12 \cdot xy \cdot zy^2 + 18 \cdot xy \cdot yz^2 + 9 \cdot xy \cdot zz^2 - 21 \cdot xy + 18 \cdot yy \cdot xz \cdot yz)/4 \tag{A.149}$$

$$D(22,22) = (-30 \cdot xx \cdot yy^2 - 12 \cdot xx \cdot zy^2 - 12 \cdot xx \cdot yz^2 - 3 \cdot xx \cdot zz^2 + 21 \cdot xx - 18 \cdot yx \cdot xy \cdot yy + 64 \cdot yy^3 \cdot zz + 48 \cdot yy \cdot zy^2 \cdot zz + 48 \cdot yy \cdot yz^2 \cdot zz + 12 \cdot yy \cdot zz^3 - 60 \cdot yy \cdot zz + 24 \cdot zy \cdot yz \cdot zz^2)/4 \tag{A.150}$$

$$D(22,23) = (\sqrt{2} \cdot (-32 \cdot yy^3 \cdot zy - 16 \cdot yy^2 \cdot yz \cdot zz - 24 \cdot yy \cdot zy^3 - 16 \cdot yy \cdot zy \cdot yz^2 - 12 \cdot yy \cdot zy \cdot zz^2 + 28 \cdot yy \cdot zy - 12 \cdot zy^2 \cdot yz \cdot zz - 4 \cdot yz^3 \cdot zz - 3 \cdot yz \cdot zz^3 + 7 \cdot yz \cdot zz))/4 \tag{A.151}$$

$$D(22,24) = (\sqrt{2} \cdot (16 \cdot yx \cdot yy^2 \cdot zy + 6 \cdot yx \cdot zy^3 + 4 \cdot yx \cdot zy \cdot yz^2 + 3 \cdot yx \cdot zy \cdot zz^2 - 7 \cdot yx \cdot zy + 16 \cdot zx \cdot yy^3 + 18 \cdot zx \cdot yy \cdot zy^2 + 12 \cdot zx \cdot yy \cdot yz^2 + 3 \cdot zx \cdot yy \cdot zz^2 - 15 \cdot zx \cdot yy + 6 \cdot zx \cdot zy \cdot yz \cdot zz))/4 \tag{A.152}$$

$$D(23,16) = \sqrt{35} \cdot (4 \cdot yy^2 \cdot zz^2 - 4 \cdot yy^2 - 8 \cdot yy \cdot zy \cdot yz \cdot zz + 4 \cdot zy^2 \cdot yz^2 - 4 \cdot zy^2 + 8 \cdot yz^4 + 8 \cdot yz^2 \cdot zz^2 - 12 \cdot yz^2 + zz^4 - 6 \cdot zz^2 + 5)/8 \tag{A.153}$$

$$D(23,17) = (\sqrt{14} \cdot (-2 \cdot yx \cdot yy \cdot zy \cdot zz + 2 \cdot yx \cdot zy^2 \cdot yz + 8 \cdot yx \cdot yz^3 + 4 \cdot yx \cdot yz \cdot zz^2 - 6 \cdot yx \cdot yz + 2 \cdot zx \cdot yy^2 \cdot zz - 2 \cdot zx \cdot yy \cdot zy \cdot yz + 4 \cdot zx \cdot yz^2 \cdot zz + zx \cdot zz^3 - 3 \cdot zx \cdot zz))/4 \tag{A.154}$$

$$D(23,18) = (\sqrt{14} \cdot (8 \cdot yy \cdot yz^3 + 4 \cdot yy \cdot yz \cdot zz^2 - 4 \cdot yy \cdot yz + 4 \cdot zy \cdot yz^2 \cdot zz + zy \cdot zz^3 - zy \cdot zz))/4 \tag{A.155}$$

$$D(23,19) = (\sqrt{7} \cdot (-16 \cdot yy^2 \cdot yz^2 - 8 \cdot yy^2 \cdot zz^2 + 8 \cdot yy^2 - 8 \cdot zy^2 \cdot yz^2 - 2 \cdot zy^2 \cdot zz^2 + 6 \cdot zy^2 - 8 \cdot yz^4 - 8 \cdot yz^2 \cdot zz^2 + 16 \cdot yz^2 - zz^4 + 8 \cdot zz^2 - 7))/4 \tag{A.156}$$

$$D(23,20) = (\sqrt{7} \cdot (8 \cdot yx \cdot yy \cdot yz^2 + 4 \cdot yx \cdot yy \cdot zz^2 - 4 \cdot yx \cdot yy + 4 \cdot zx \cdot zy \cdot yz^2 + zx \cdot zy \cdot zz^2 - 3 \cdot zx \cdot zy))/2 \tag{A.157}$$

$$D(23,21) = (\sqrt{2} \cdot (-32 \cdot yx \cdot yy^2 \cdot yz - 6 \cdot yx \cdot yy \cdot zy \cdot zz - 10 \cdot yx \cdot zy^2 \cdot yz - 8 \cdot yx \cdot yz^3 - 4 \cdot yx \cdot yz \cdot zz^2 + 14 \cdot yx \cdot yz - 10 \cdot zx \cdot yy^2 \cdot zz - 6 \cdot zx \cdot yy \cdot zy \cdot yz - 4 \cdot zx \cdot zy^2 \cdot zz - 4 \cdot zx \cdot yz^2 \cdot zz - zx \cdot zz^3 + 7 \cdot zx \cdot zz))/4 \tag{A.158}$$

$$D(23,22) = (\sqrt{2} \cdot (-32 \cdot yy^3 \cdot yz - 16 \cdot yy^2 \cdot zy \cdot zz - 16 \cdot yy \cdot zy^2 \cdot yz - 24 \cdot yy \cdot yz^3 - 12 \cdot yy \cdot yz \cdot zz^2 + 28 \cdot yy \cdot yz - 4 \cdot zy^3 \cdot zz - 12 \cdot zy \cdot yz^2 \cdot zz - 3 \cdot zy \cdot zz^3 + 7 \cdot zy \cdot zz))/4 \tag{A.159}$$

$$D(23,23) = (64 \cdot yy^4 + 64 \cdot yy^2 \cdot zy^2 + 64 \cdot yy^2 \cdot yz^2 + 20 \cdot yy^2 \cdot zz^2 - 84 \cdot yy^2 + 24 \cdot yy \cdot zy \cdot yz \cdot zz + 8 \cdot zy^4 + 20 \cdot zy^2 \cdot yz^2 + 8 \cdot zy^2 \cdot zz^2 - 28 \cdot zy^2 + 8 \cdot yz^4 + 8 \cdot yz^2 \cdot zz^2 - 28 \cdot yz^2 + zz^4 - 14 \cdot zz^2 + 21)/8 \tag{A.160}$$

$$D(23,24) = (-16 \cdot yx \cdot yy^3 - 8 \cdot yx \cdot yy \cdot zy^2 - 8 \cdot yx \cdot yy \cdot yz^2 - 4 \cdot yx \cdot yy \cdot zz^2 + 12 \cdot yx \cdot yy - 8 \cdot zx \cdot yy^2 \cdot zy - 2 \cdot zx \cdot zy^3 - 4 \cdot zx \cdot zy \cdot yz^2 - zx \cdot zy \cdot zz^2 + 5 \cdot zx \cdot zy)/2 \tag{A.161}$$

$$D(24,16) = \sqrt{35} \cdot xz \cdot yz \cdot (-2 \cdot yz^2 - zz^2 + 1)/2 \tag{A.162}$$

$$D(24,17) = (\sqrt{14} \cdot (-2 \cdot xx \cdot yy^2 \cdot yz - 4 \cdot xx \cdot yz^3 - xx \cdot yz \cdot zz^2 + 3 \cdot xx \cdot yz + 2 \cdot yx \cdot xy \cdot yy \cdot yz - 2 \cdot yx \cdot zy^2 \cdot xz - 4 \cdot yx \cdot xz \cdot yz^2 - 3 \cdot yx \cdot xz \cdot zz^2 + 3 \cdot yx \cdot xz + 2 \cdot zx \cdot yy \cdot zy \cdot xz))/4 \tag{A.163}$$

$$D(24,18) = (\sqrt{14} \cdot (-4 \cdot xy \cdot yz^3 - 3 \cdot xy \cdot yz \cdot zz^2 + 3 \cdot xy \cdot yz - 4 \cdot yy \cdot xz \cdot yz^2 - yy \cdot xz \cdot zz^2 + yy \cdot xz))/4 \tag{A.164}$$

$$D(24,19) = \sqrt{7} \cdot (2 \cdot xy \cdot yy \cdot yz^2 + xy \cdot yy \cdot zz^2 - xy \cdot yy + 2 \cdot yy^2 \cdot xz \cdot yz + zy^2 \cdot xz \cdot yz + 2 \cdot xz \cdot yz^3 + xz \cdot yz \cdot zz^2 - 2 \cdot xz \cdot yz) \tag{A.165}$$

$$D(24,20) = (\sqrt{7} \cdot (2 \cdot xx \cdot yy \cdot zy^2 - 4 \cdot xx \cdot yy \cdot yz^2 - xx \cdot yy \cdot zz^2 + xx \cdot yy - 2 \cdot yx \cdot xy \cdot zy^2 - 4 \cdot yx \cdot xy \cdot yz^2 - 3 \cdot yx \cdot xy \cdot zz^2 + 3 \cdot yx \cdot xy))/2 \tag{A.166}$$

$$D(24,21) = (\sqrt{2}\cdot(10\cdot xx\cdot yy^2\cdot yz + 4\cdot xx\cdot zy^2\cdot yz + 4\cdot xx\cdot yz^3 + xx\cdot yz\cdot zz^2 - 7\cdot xx\cdot yz + 6\cdot yx\cdot xy\cdot yy\cdot yz + 16\cdot yx\cdot yy^2\cdot xz + 6\cdot yx\cdot zy^2\cdot xz + 4\cdot yx\cdot xz\cdot yz^2 + 3\cdot yx\cdot xz\cdot zz^2 - 7\cdot yx\cdot xz + 6\cdot zx\cdot yy\cdot zy\cdot xz))/4 \tag{A.167}$$

$$D(24,22) = (\sqrt{2}\cdot(16\cdot xy\cdot yy^2\cdot yz + 4\cdot xy\cdot zy^2\cdot yz + 12\cdot xy\cdot yz^3 + 9\cdot xy\cdot yz\cdot zz^2 - 13\cdot xy\cdot yz + 16\cdot yy^3\cdot xz + 12\cdot yy\cdot zy^2\cdot xz + 12\cdot yy\cdot xz\cdot yz^2 + 3\cdot yy\cdot xz\cdot zz^2 - 15\cdot yy\cdot xz))/4 \tag{A.168}$$

$$D(24,23) = (-16\cdot xy\cdot yy^3 - 8\cdot xy\cdot yy\cdot zy^2 - 8\cdot xy\cdot yy\cdot yz^2 - 4\cdot xy\cdot yy\cdot zz^2 + 12\cdot xy\cdot yy - 8\cdot yy^2\cdot xz\cdot yz - 4\cdot zy^2\cdot xz\cdot yz - 2\cdot xz\cdot yz^3 - xz\cdot yz\cdot zz^2 + 5\cdot xz\cdot yz)/2 \tag{A.169}$$

$$D(24,24) = (8\cdot xx\cdot yy^3 + 4\cdot xx\cdot yy\cdot zy^2 + 4\cdot xx\cdot yy\cdot yz^2 + xx\cdot yy\cdot zz^2 - 7\cdot xx\cdot yy + 8\cdot yx\cdot xy\cdot yy^2 + 4\cdot yx\cdot xy\cdot zy^2 + 4\cdot yx\cdot xy\cdot yz^2 + 3\cdot yx\cdot xy\cdot zz^2 - 5\cdot yx\cdot xy)/2\cdot yz - 4\cdot zy^2\cdot xz\cdot yz - 2\cdot xz\cdot yz^3 - xz\cdot yz\cdot zz^2 + 5\cdot xz\cdot yz)/2 \tag{A.170}$$

A.2 Multipole Translations

Using the following formula

$$Q^C_{lk} = \sum_{l'=0}^{l}\sum_{k'=-l'}^{l'}\left[\binom{l+k}{l'+k'}\binom{l-k}{l'-k'}\right]^{\frac{1}{2}} Q^O_{l'k'} R_{l-l',k-k'}(-\mathbf{c}) \tag{A.171}$$

where the Q^C_{lk} are the multipole moments at the final position, the terms in curved brackets are binomial coefficients, $Q^O_{l'k'}$ are the multipole moments at the initial location and

$R_{l-l',k-k'}(-c)$ is a regular spherical harmonic. The spherical harmonics are a function of $-\mathbf{c}$ or $\{-x,-y,-z\}$. Let

$$W_{lk,l'k'} = \left[\binom{l+k}{l'+k'}\binom{l-k}{l'-k'}\right]^{\frac{1}{2}} R_{l-l',k-k'}(-\mathbf{c}) \tag{A.172}$$

such that

$$Q^C_{lk} = \sum_{l'=0}^{l}\sum_{k'=-l'}^{l'} Q^O_{l'k'} W_{lk,l'k'}(-\mathbf{c}). \tag{A.173}$$

If k is non-zero, the resulting spherical harmonic (R) and the corresponding multipole moment (Q) will be complex. In that case, Q or R can be converted into real functions using linear combinations of regular harmonics. The expression here is expressed in terms of R, but one can obtain expressions for the multipoles by replacing R with Q.

$$R_{lm} = \frac{R_{lmc} + iR_{lms}}{2b_m} \tag{A.174}$$

with the following conditions

$$\text{if } m > 0 \quad b_m = (-1)^m\sqrt{\frac{1}{2}} \tag{A.175}$$

$$\text{if } m < 0 \quad b_m = \sqrt{\frac{1}{2}} \tag{A.176}$$

The above definitions ensure that the cosine component satisfies $R_{lmc} = R_{l|m|c}$ and the sine component satisfies $R_{lms} = -R_{l|m|s}$.[113] By definition spherical harmonics with l values smaller than the absolute value of the m term are equal to zero.

The next several sections derive sample multipole translation expressions. Section A.2.4 lists a full set of multipole translation expressions up to rank 4.

A.2.1 Rank 0: Charge translation

For Rank 0, there is only one term:

$$Q_{00}^{C} = Q_{00}^{O} W_{00,00}(-\mathbf{c}) \tag{A.177}$$

$l = 0$; $k = 0$; $l' = 0$; $k' = 0$;

$$Q_{00}^{O} W_{00,10}(-\mathbf{c}) = Q_{00}^{O} \left[\binom{0}{0} \binom{0}{0} \right]^{\frac{1}{2}} R_{00}(-\mathbf{c}) \tag{A.178}$$

Since $R_{00}(-\mathbf{c}) = 1$,

$$Q_{00}^{C} = Q_{00}^{O} \tag{A.179}$$

A.2.2 Rank 1: Dipole translation

For Rank 1, we will examine Q_{10}^{C}.

$$Q_{10}^{C} = Q_{00}^{O} W_{00,10}(-\mathbf{c}) + Q_{10}^{O} W_{10,10}(-\mathbf{c}) + Q_{11}^{O} W_{11,10}(-\mathbf{c}) + Q_{1,-1}^{O} W_{1,-1,10}(-\mathbf{c}) \tag{A.180}$$

Consider each term in the the expression above:

- First term: $l = 1$; $k = 0$; $l' = 0$; $k' = 0$;

$$Q^O_{00} W_{00,10}(-\mathbf{c}) = Q^O_{00} \left[\binom{1}{0} \binom{1}{0} \right]^{\frac{1}{2}} R_{10}(-\mathbf{c}) \tag{A.181}$$

$$= -z Q^O_{00} \tag{A.182}$$

since $R_{10}(-\mathbf{c}) = -z$,.

- Second term: $l = 1$; $k = 0$; $l' = 1$; $k' = 0$;

$$Q^O_{10} W_{10,10}(-\mathbf{c}) = Q^O_{10} \left[\binom{1}{1} \binom{1}{1} \right]^{\frac{1}{2}} R_{00}(-\mathbf{c}) \tag{A.183}$$

$$= Q^O_{10} \tag{A.184}$$

where the final equality follows from the fact that $R_{00}(-\mathbf{c}) = 1$.

- Third term: $l = 1$; $k = 0$; $l' = 1$; $k' = 1$;

$$Q^O_{11} W_{11,10}(-\mathbf{c}) = Q^O_{11} \left[\binom{1}{2} \binom{1}{0} \right]^{\frac{1}{2}} R_{0,1}(-\mathbf{c}) = 0 \tag{A.185}$$

Since $|m| > l$, the spherical harmonic is equal to zero, making the whole term equal to zero.

- Fourth term: $l = 1$; $k = 0$; $l' = 1$; $k' = -1$;

$$Q^O_{1,-1} W_{1,-1,10}(-\mathbf{c}) = Q^O_{1,-1} \left[\binom{1}{0} \binom{1}{2} \right]^{\frac{1}{2}} R_{0,-1}(-\mathbf{c}) = 0 \tag{A.186}$$

Again, the spherical harmonic is equal to zero, and so the whole term goes to zero.

- Summing all the terms:

$$Q_{10}^{C} = Q_{10}^{O} - zQ_{00}^{O} \tag{A.187}$$

A.2.3 Rank 2: Quadrupole translation

We will also examine a Rank 2 case. In this example, complex spherical harmonics arise.

$$Q_{20}^{C} = Q_{00}^{O}W_{00,20} + Q_{10}^{O}W_{10,20} + Q_{11}^{O}W_{11,20} + Q_{1,-1}^{O}W_{1,-1,20} + Q_{20}^{O}W_{20,20} + Q_{21}^{O}W_{21,20} + Q_{2,-1}^{O}W_{2,-1,20} + Q_{22}^{O}W_{22,20} + Q_{2,-2}^{O}W_{2,-2,20} \tag{A.188}$$

Again, we evaluate this expression term by term:

- First term: $l = 2$; $k = 0$; $l' = 0$; $k' = 0$;

$$Q_{00}^{O}W_{00,20}(-\mathbf{c}) = Q_{00}^{O}\left[\binom{2}{0}\binom{2}{0}\right]^{\frac{1}{2}} R_{20}(-\mathbf{c}) = \frac{1}{2}(2z^2 - x^2 - y^2)Q_{00}^{O} \tag{A.189}$$

  where $R_{20}(-\mathbf{c}) = \frac{1}{2}(3z^2 - r^2) = \frac{1}{2}(2z^2 - x^2 - y^2)$ was used to obtain the final equality.

- Second Term: $l = 2$; $k = 0$; $l' = 1$; $k' = 0$;

$$Q_{10}^{O}W_{10,20} = Q_{10}^{O}\left[\binom{2}{1}\binom{2}{1}\right]^{\frac{1}{2}} R_{10}(-\mathbf{c}) = -2zQ_{10}^{O} \tag{A.190}$$

  since $R_{10}(-\mathbf{c}) = -z$.

- Third term: $l = 2$; $k = 0$; $l' = 1$; $k' = 1$;

$$Q_{11}^{O} W_{11,20}(-\mathbf{c}) = Q_{11}^{O} \left[\binom{2}{2} \binom{2}{0} \right]^{\frac{1}{2}} R_{1,-1}(-\mathbf{c}) \tag{A.191}$$

The spherical harmonic is complex, so we apply the rules for changing complex spherical harmonics to linear combinations of regular spherical harmonics:

$$R_{1,-1} = \frac{R_{11c} - iR_{11s}}{2\sqrt{\frac{1}{2}}} = \sqrt{\frac{1}{2}} \left(R_{11c} - iR_{11s} \right) \tag{A.192}$$

Using the same rules for the complex multipole:

$$Q_{11} = \frac{Q_{11c} + iQ_{11s}}{2(-1)^1\sqrt{\frac{1}{2}}} = -\sqrt{\frac{1}{2}}(Q_{11c} + iQ_{11s}) \tag{A.193}$$

Combining these two gives:

$$Q_{11}^{O} R_{1,-1}(-\mathbf{c}) = -\sqrt{\frac{1}{2}}(Q_{11c} + iQ_{11s})\sqrt{\frac{1}{2}} \left(R_{11c}(-\mathbf{c}) - iR_{11s}(-\mathbf{c}) \right) \tag{A.194}$$

Simplifying the expression gives,

$$Q_{11}^{O} R_{1,-1}(-\mathbf{c}) = -\frac{1}{2}(Q_{11c}R_{11c}(-\mathbf{c}) + iQ_{11s}R_{11c}(-\mathbf{c}) - iQ_{11c}R_{11s}(-\mathbf{c}) + Q_{11s}R_{11s}(-\mathbf{c})) \tag{A.195}$$

While this expression still includes complex terms, the imaginary parts will cancel with other terms in the overall expression, resulting in a final expression for the translation that is real.

- Fourth term: $l = 2$; $k = 0$; $l' = 1$; $k' = -1$;

$$Q^O_{1,-1}W_{1,-1,20}(-\mathbf{c}) = Q^O_{1,-1}\left[\binom{2}{0}\binom{2}{2}\right]^{\frac{1}{2}} R_{1,1}(-\mathbf{c}) = Q^O_{1,-1}R_{1,1}(-\mathbf{c}) \tag{A.196}$$

Once again, the multipole and spherical harmonic are complex.

$$R_{11} = \frac{R_{11c} + iR_{11s}}{2(-1)^1\sqrt{\frac{1}{2}}} = -\sqrt{\frac{1}{2}}(R_{11c} + iR_{11s}) \tag{A.197}$$

$$Q_{1,-1} = \frac{Q_{11c} - iQ_{11s}}{2\sqrt{\frac{1}{2}}} = \sqrt{\frac{1}{2}}(Q_{11c} - iQ_{11s}) \tag{A.198}$$

So the entire term is:

$$Q^O_{1,-1}R_{1,1}(-\mathbf{c}) = -\frac{1}{2}(Q_{11c}R_{11c}(-\mathbf{c}) + iQ_{11c}R_{11s}(-\mathbf{c}) - iQ_{11s}R_{11c}(-\mathbf{c}) + Q_{11s}R_{11s}(-\mathbf{c})) \tag{A.199}$$

- Fifth term: $l = 2$; $k = 0$; $l' = 2$; $k' = 0$;

$$Q^O_{20}W_{020,20} = Q^O_{20}\left[\binom{2}{2}\binom{2}{2}\right]^{\frac{1}{2}} R_{20}(-\mathbf{c}) = Q^O_{20}R_{00}(-\mathbf{c}) = Q^O_{20} \tag{A.200}$$

since $R_{00} = 1$. One can show that terms six through nine have R_{lm} with $|m| > l$ and are therefore equal to zero.

- Combining the first, second, third, fourth and fifth terms,

$$\begin{aligned} Q_{20}^C =& \frac{1}{2}(2z^2 - x^2 - y^2)Q_{00}^O - 2zQ_{10}^O - \frac{1}{2}(Q_{11c}R_{11c}(-\mathbf{c}) \\ &+ iQ_{11s}R_{11c}(-\mathbf{c}) - iQ_{11c}R_{11s}(-\mathbf{c}) + Q_{11s}R_{11s}(-\mathbf{c})) \\ &- \frac{1}{2}(Q_{11c}R_{11c}(-\mathbf{c}) + iQ_{11c}R_{11s}(-\mathbf{c}) - iQ_{11s}R_{11c}(-\mathbf{c}) + Q_{11s}R_{11s}(-\mathbf{c})) + Q_{20}^O \end{aligned} \tag{A.201}$$

$$Q_{20}^C = \frac{1}{2}(2z^2 - x^2 - y^2)Q_{00}^O - 2zQ_{10}^O - Q_{11c}^O R_{11c}(-\mathbf{c}) - Q_{11s}^O R_{11s}(-\mathbf{c}) + Q_{20}^O \tag{A.202}$$

Finally, since $R_{11c}(-\mathbf{c}) = -x$ and $R_{11s}(-\mathbf{c}) = -y$,

$$Q_{20}^C = \frac{1}{2}(2z^2 - x^2 - y^2)Q_{00}^O - 2zQ_{10}^O + xQ_{11c}^O + yQ_{11s}^O + Q_{20}^O \tag{A.203}$$

A.2.4 Full List of Multipolar Translation Expressions up to Rank 4

Expressions for translating multipoles from some origin O to a new position C have been tabulated below. The vector $\mathbf{c} = (x, y, z)$ defines the translation from O to C.

Rank 0:

$$Q_{00}^C = Q_{00}^O \tag{A.204}$$

Rank 1:

$$Q_{10}^C = Q_{10}^O - x \cdot Q_{00}^O \tag{A.205}$$

$$Q_{11c}^C = Q_{11c}^O - y \cdot Q_{00}^O \tag{A.206}$$

$$Q_{11s}^C = Q_{11s}^O - z \cdot Q_{00}^O \tag{A.207}$$

Rank 2:

$$Q_{20}^{C} = Q_{20}^{O} + 0.5 \cdot (2 \cdot x \cdot Q_{10}^{O} + 2 \cdot y \cdot Q_{11c}^{O} - 4 \cdot z \cdot Q_{11s}^{O} + 2 \cdot z^2 \cdot Q_{00}^{O} - x^2 \cdot Q_{00}^{O} - y^2 \cdot Q_{00}^{O}) \quad \text{(A.208)}$$

$$Q_{21c}^{C} = Q_{21c}^{O} - \sqrt{3} \cdot z \cdot Q_{10}^{O} - \sqrt{3} \cdot x \cdot Q_{11s}^{O} + \sqrt{3} \cdot x \cdot z \cdot Q_{00}^{O} \quad \text{(A.209)}$$

$$Q_{21s}^{C} = Q_{21s}^{O} - \sqrt{3} \cdot z \cdot Q_{11c}^{O} - \sqrt{3} \cdot y \cdot Q_{11s}^{O} + \sqrt{3} \cdot y \cdot z \cdot Q_{00}^{O} \quad \text{(A.210)}$$

$$Q_{22c}^{C} = Q_{22c}^{O} + 0.5 \cdot (2 \cdot \sqrt{3} \cdot y \cdot Q_{11c}^{O} - 2 \cdot \sqrt{3} \cdot x \cdot Q_{10}^{O} + \sqrt{3} \cdot x^2 \cdot Q_{00}^{O} - \sqrt{3} \cdot y^2 \cdot Q_{00}^{O}) \quad \text{(A.211)}$$

$$Q_{22s}^{C} = Q_{22s}^{O} - \sqrt{3} \cdot x \cdot Q_{11c}^{O} - \sqrt{3} \cdot y \cdot Q_{10}^{O} + \sqrt{3} \cdot x \cdot y \cdot Q_{00}^{O} \quad \text{(A.212)}$$

Rank 3:

$$Q_{30}^{C} = Q_{30}^{O} + 0.5 \cdot (3 \cdot x^2 \cdot z \cdot Q_{00}^{O} + 3 \cdot y^2 \cdot z \cdot Q_{00}^{O} - 2 \cdot z^3 \cdot Q_{00}^{O} - 3 \cdot x^2 \cdot Q_{11s}^{O} - 3 \cdot y^2 \cdot Q_{11s}^{O} + 6 \cdot z^2 \cdot Q_{11s}^{O} - 6 \cdot z \cdot Q_{20}^{O} - 6 \cdot x \cdot z \cdot Q_{10}^{O} - 6 \cdot y \cdot z \cdot Q_{11c}^{O} + 2 \cdot \sqrt{3} \cdot x \cdot Q_{21c}^{O} + 2 \cdot \sqrt{3} \cdot y \cdot Q_{21s}^{O}) \quad \text{(A.213)}$$

$$Q_{31c}^{C} = Q_{31c}^{O} + 0.25 \cdot (\sqrt{6} \cdot x^3 \cdot Q_{00}^{O} + \sqrt{6} \cdot x \cdot y^2 \cdot Q_{00}^{O} - 4 \cdot \sqrt{6} \cdot x \cdot z^2 \cdot Q_{00}^{O} + 8 \cdot \sqrt{6} \cdot x \cdot z \cdot Q_{11s}^{O} - 4 \cdot \sqrt{6} \cdot x \cdot Q_{20}^{O} - 3 \cdot \sqrt{6} \cdot x^2 \cdot Q_{10}^{O} - \sqrt{6} \cdot y^2 \cdot Q_{10}^{O} + 4 \cdot \sqrt{6} \cdot z^2 \cdot Q_{10}^{O} - 2 \cdot \sqrt{6} \cdot x \cdot y \cdot Q_{11c}^{O} - 8 \cdot \sqrt{2} \cdot z \cdot Q_{21c}^{O} + 2 \cdot \sqrt{2} \cdot x \cdot Q_{22c}^{O} + 2 \cdot \sqrt{2} \cdot y \cdot Q_{22s}^{O}) \quad \text{(A.214)}$$

$$Q_{31s}^{C} = Q_{31s}^{O} + 0.25 \cdot (\sqrt{6} \cdot x^2 \cdot y \cdot Q_{00}^{O} + \sqrt{6} \cdot y^3 \cdot Q_{00}^{O} - 4 \cdot \sqrt{6} \cdot y \cdot z^2 \cdot Q_{00}^{O} + 8 \cdot \sqrt{6} \cdot y \cdot z \cdot Q_{11s}^{O} - 4 \cdot \sqrt{6} \cdot y \cdot Q_{20}^{O} - 2 \cdot \sqrt{6} \cdot x \cdot y \cdot Q_{10}^{O} - \sqrt{6} \cdot x^2 \cdot Q_{11c}^{O} - 3 \cdot \sqrt{6} \cdot y^2 \cdot Q_{11c}^{O} + 4 \cdot \sqrt{6} \cdot z^2 \cdot Q_{11c}^{O} - 8 \cdot \sqrt{2} \cdot z \cdot Q_{21s}^{O} - 2 \cdot \sqrt{2} \cdot y \cdot Q_{22c}^{O} + 2 \cdot \sqrt{2} \cdot x \cdot Q_{22s}^{O}) \quad \text{(A.215)}$$

$$Q_{32c}^{C} = Q_{32c}^{O} + 0.5 \cdot (-\sqrt{15} \cdot x^2 \cdot z \cdot Q_{00}^{O} + \sqrt{15} \cdot y^2 \cdot z \cdot Q_{00}^{O} + \sqrt{15} \cdot x^2 \cdot Q_{11s}^{O} - \sqrt{15} \cdot y^2 \cdot Q_{11s}^{O} + 2 \cdot \sqrt{15} \cdot x \cdot z \cdot Q_{10}^{O} - 2 \cdot \sqrt{15} \cdot y \cdot z \cdot Q_{11c}^{O} - 2 \cdot \sqrt{5} \cdot x \cdot Q_{21c}^{O} + 2 \cdot \sqrt{5} \cdot y \cdot Q_{21s}^{O} - 2 \cdot \sqrt{5} \cdot z \cdot Q_{22c}^{O}) \tag{A.216}$$

$$Q_{32s}^{C} = Q_{32s}^{O} - \sqrt{15} \cdot x \cdot y \cdot z \cdot Q_{00}^{O} + \sqrt{15} \cdot x \cdot y \cdot Q_{11s}^{O} + \sqrt{15} \cdot y \cdot z \cdot Q_{10}^{O} + \sqrt{15} \cdot x \cdot z \cdot Q_{11c}^{O} - \sqrt{5} \cdot y \cdot Q_{21c}^{O} - \sqrt{5} \cdot x \cdot Q_{21s}^{O} - \sqrt{5} \cdot z \cdot Q_{22s}^{O} \tag{A.217}$$

$$Q_{33c}^{C} = Q_{33c}^{O} + 0.25 \cdot (-\sqrt{10} \cdot x^3 \cdot Q_{00}^{O} + 3 \cdot \sqrt{10} \cdot x \cdot y^2 \cdot Q_{00}^{O} + 3 \cdot \sqrt{10} \cdot x^2 \cdot Q_{10}^{O} - 3 \cdot \sqrt{10} \cdot y^2 \cdot Q_{10}^{O} - 6 \cdot \sqrt{10} \cdot x \cdot y \cdot Q_{11c}^{O} - 2 \cdot \sqrt{30} \cdot x \cdot Q_{22c}^{O} + 2 \cdot \sqrt{30} \cdot y \cdot Q_{22s}^{O}) \tag{A.218}$$

$$Q_{33s}^{C} = Q_{33s}^{O} + 0.25 \cdot (-3 \cdot \sqrt{10} \cdot x^2 \cdot y \cdot Q_{00}^{O} + \sqrt{10} \cdot y^3 \cdot Q_{00}^{O} + 6 \cdot \sqrt{10} \cdot x \cdot y \cdot Q_{10}^{O} + 3 \cdot \sqrt{10} \cdot x^2 \cdot Q_{11c}^{O} - 3 \cdot \sqrt{10} \cdot y^2 \cdot Q_{11c}^{O} - 2 \cdot \sqrt{30} \cdot y \cdot Q_{22c}^{O} - 2 \cdot \sqrt{30} \cdot x \cdot Q_{22s}^{O}) \tag{A.219}$$

Rank 4:

$$Q_{40}^{C} = Q_{40}^{O} + 0.125 \cdot (3 \cdot x^4 \cdot Q_{00}^{O} + 6 \cdot x^2 \cdot y^2 \cdot Q_{00}^{O} + 3 \cdot y^4 \cdot Q_{00}^{O} - 24 \cdot x^2 \cdot z^2 \cdot Q_{00}^{O} - 24 \cdot y^2 \cdot z^2 \cdot Q_{00}^{O} + 8 \cdot z^4 \cdot Q_{00}^{O} + 48 \cdot x^2 \cdot z \cdot Q_{11s}^{O} + 48 \cdot y^2 \cdot z \cdot Q_{11s}^{O} - 32 \cdot z^3 \cdot Q_{11s}^{O} - 24 \cdot x^2 \cdot Q_{20}^{O} - 24 \cdot y^2 \cdot Q_{20}^{O} + 48 \cdot z^2 \cdot Q_{20}^{O} - 32 \cdot z \cdot Q_{30}^{O} - 12 \cdot x^3 \cdot Q_{10}^{O} - 12 \cdot x \cdot y^2 \cdot Q_{10}^{O} + 48 \cdot x \cdot z^2 \cdot Q_{10}^{O} - 12 \cdot x^2 \cdot y \cdot Q_{11c}^{O} - 12 \cdot y^3 \cdot Q_{11c}^{O} + 48 \cdot y \cdot z^2 \cdot Q_{11c}^{O} - 32 \cdot \sqrt{3} \cdot x \cdot z \cdot Q_{21c}^{O} - 32 \cdot \sqrt{3} \cdot y \cdot z \cdot Q_{21s}^{O} + 4 \cdot \sqrt{3} \cdot x^2 \cdot Q_{22c}^{O} - 4 \cdot \sqrt{3} \cdot y^2 \cdot Q_{22c}^{O} + 8 \cdot \sqrt{3} \cdot x \cdot y \cdot Q_{22s}^{O} + 8 \cdot \sqrt{6} \cdot x \cdot Q_{31c}^{O} + 8 \cdot \sqrt{6} \cdot y \cdot Q_{31s}^{O}) \tag{A.220}$$

$$Q_{41c}^{C} = Q_{41c}^{O} + 0.25 \cdot (-3 \cdot \sqrt{10} \cdot x^3 \cdot z \cdot Q_{00}^{O} - 3 \cdot \sqrt{10} \cdot x \cdot y^2 \cdot z \cdot Q_{00}^{O} + 4 \cdot \sqrt{10} \cdot x \cdot z^3 \cdot Q_{00}^{O} + 3 \cdot \sqrt{10} \cdot x^3 \cdot Q_{11s}^{O} + 3 \cdot \sqrt{10} \cdot x \cdot y^2 \cdot Q_{11s}^{O} - 12 \cdot \sqrt{10} \cdot x \cdot z^2 \cdot Q_{11s}^{O} + 12 \cdot \sqrt{10} \cdot x \cdot z \cdot Q_{20}^{O} - 4 \cdot \sqrt{10} \cdot x \cdot Q_{30}^{O} + 9 \cdot \sqrt{10} \cdot x^2 \cdot z \cdot Q_{10}^{O} + 3 \cdot \sqrt{10} \cdot y^2 \cdot z \cdot Q_{10}^{O} - 4 \cdot \sqrt{10} \cdot z^3 \cdot Q_{10}^{O} + 6 \cdot \sqrt{10} \cdot x \cdot y \cdot z \cdot Q_{11c}^{O} - 3 \cdot \sqrt{30} \cdot x^2 \cdot Q_{21c}^{O} - \sqrt{30} \cdot y^2 \cdot Q_{21c}^{O} + 4 \cdot \sqrt{30} \cdot z^2 \cdot Q_{21c}^{O} - 2 \cdot \sqrt{30} \cdot x \cdot y \cdot Q_{21s}^{O} - 2 \cdot \sqrt{30} \cdot x \cdot z \cdot Q_{22c}^{O} - 2 \cdot \sqrt{30} \cdot y \cdot z \cdot Q_{22s}^{O} - 4 \cdot \sqrt{15} \cdot z \cdot Q_{31c}^{O} + 2 \cdot \sqrt{6} \cdot x \cdot Q_{32c}^{O} + 2 \cdot \sqrt{6} \cdot y \cdot Q_{32s}^{O}) \tag{A.221}$$

$$Q^C_{41s} = Q^O_{41s} + 0.25 \cdot (-3 \cdot \sqrt{10} \cdot x^2 \cdot y \cdot z \cdot Q^O_{00} - 3 \cdot \sqrt{10} \cdot y^3 \cdot z \cdot Q^O_{00} + 4 \cdot \sqrt{10} \cdot y \cdot z^3 \cdot Q^O_{00} + 3 \cdot \sqrt{10} \cdot x^2 \cdot y \cdot Q^O_{11s} + 3 \cdot \sqrt{10} \cdot y^3 \cdot Q^O_{11s} - 12 \cdot \sqrt{10} \cdot y \cdot z^2 \cdot Q^O_{11s} + 12 \cdot \sqrt{10} \cdot y \cdot z \cdot Q^O_{20} - 4 \cdot \sqrt{10} \cdot y \cdot Q^O_{30} + 6 \cdot \sqrt{10} \cdot x \cdot y \cdot z \cdot Q^O_{10} + 3 \cdot \sqrt{10} \cdot x^2 \cdot z \cdot Q^O_{11c} + 9 \cdot \sqrt{10} \cdot y^2 \cdot z \cdot Q^O_{11c} - 4 \cdot \sqrt{10} \cdot z^3 \cdot Q^O_{11c} - 2 \cdot \sqrt{30} \cdot x \cdot y \cdot Q^O_{21c} - \sqrt{30} \cdot x^2 \cdot Q^O_{21s} - 3 \cdot \sqrt{30} \cdot y^2 \cdot Q^O_{21s} + 4 \cdot \sqrt{30} \cdot z^2 \cdot Q^O_{21s} + 2 \cdot \sqrt{30} \cdot y \cdot z \cdot Q^O_{22c} - 2 \cdot \sqrt{30} \cdot x \cdot z \cdot Q^O_{22s} - 4 \cdot \sqrt{15} \cdot z \cdot Q^O_{31s} - 2 \cdot \sqrt{6} \cdot y \cdot Q^O_{32c} + 2 \cdot \sqrt{6} \cdot x \cdot Q^O_{32s})$$
(A.222)

$$Q^C_{42c} = Q^O_{42c} + 0.25 \cdot (-\sqrt{5} \cdot x^4 \cdot Q^O_{00} + \sqrt{5} \cdot y^4 \cdot Q^O_{00} + 6 \cdot \sqrt{5} \cdot x^2 \cdot z^2 \cdot Q^O_{00} - 6 \cdot \sqrt{5} \cdot y^2 \cdot z^2 \cdot Q^O_{00} - 12 \cdot \sqrt{5} \cdot x^2 \cdot z \cdot Q^O_{11s} + 12 \cdot \sqrt{5} \cdot y^2 \cdot z \cdot Q^O_{11s} + 6 \cdot \sqrt{5} \cdot x^2 \cdot Q^O_{20} - 6 \cdot \sqrt{5} \cdot y^2 \cdot Q^O_{20} + 4 \cdot \sqrt{5} \cdot x^3 \cdot Q^O_{10} - 12 \cdot \sqrt{5} \cdot x \cdot z^2 \cdot Q^O_{10} - 4 \cdot \sqrt{5} \cdot y^3 \cdot Q^O_{11c} + 12 \cdot \sqrt{5} \cdot y \cdot z^2 \cdot Q^O_{11c} + 8 \cdot \sqrt{15} \cdot x \cdot z \cdot Q^O_{21c} - 8 \cdot \sqrt{15} \cdot y \cdot z \cdot Q^O_{21s} - 2 \cdot \sqrt{15} \cdot x^2 \cdot Q^O_{22c} - 2 \cdot \sqrt{15} \cdot y^2 \cdot Q^O_{22c} + 4 \cdot \sqrt{15} \cdot z^2 \cdot Q^O_{22c} - 2 \cdot \sqrt{30} \cdot x \cdot Q^O_{31c} + 2 \cdot \sqrt{30} \cdot y \cdot Q^O_{31s} - 8 \cdot \sqrt{3} \cdot z \cdot Q^O_{32c} + 2 \cdot \sqrt{2} \cdot x \cdot Q^O_{33c} + 2 \cdot \sqrt{2} \cdot y \cdot Q^O_{33s})$$
(A.223)

$$Q^C_{42s} = Q^O_{42s} + 0.5 \cdot (-\sqrt{5} \cdot x^3 \cdot y \cdot Q^O_{00} - \sqrt{5} \cdot x \cdot y^3 \cdot Q^O_{00} + 6 \cdot \sqrt{5} \cdot x \cdot y \cdot z^2 \cdot Q^O_{00} - 12 \cdot \sqrt{5} \cdot x \cdot y \cdot z \cdot Q^O_{11s} + 6 \cdot \sqrt{5} \cdot x \cdot y \cdot Q^O_{20} + 3 \cdot \sqrt{5} \cdot x^2 \cdot y \cdot Q^O_{10} + \sqrt{5} \cdot y^3 \cdot Q^O_{10} - 6 \cdot \sqrt{5} \cdot y \cdot z^2 \cdot Q^O_{10} + \sqrt{5} \cdot x^3 \cdot Q^O_{11c} + 3 \cdot \sqrt{5} \cdot x \cdot y^2 \cdot Q^O_{11c} - 6 \cdot \sqrt{5} \cdot x \cdot z^2 \cdot Q^O_{11c} + 4 \cdot \sqrt{15} \cdot y \cdot z \cdot Q^O_{21c} + 4 \cdot \sqrt{15} \cdot x \cdot z \cdot Q^O_{21s} - \sqrt{15} \cdot x^2 \cdot Q^O_{22s} - \sqrt{15} \cdot y^2 \cdot Q^O_{22s} + 2 \cdot \sqrt{15} \cdot z^2 \cdot Q^O_{22s} - \sqrt{30} \cdot y \cdot Q^O_{31c} - \sqrt{30} \cdot x \cdot Q^O_{31s} - 4 \cdot \sqrt{3} \cdot z \cdot Q^O_{32s} - \sqrt{2} \cdot y \cdot Q^O_{33c} + \sqrt{2} \cdot x \cdot Q^O_{33s})$$
(A.224)

$$Q^C_{43c} = Q^O_{43c} + 0.25 \cdot (\sqrt{70} \cdot x^3 \cdot z \cdot Q^O_{00} - 3 \cdot \sqrt{70} \cdot x \cdot y^2 \cdot z \cdot Q^O_{00} - \sqrt{70} \cdot x^3 \cdot Q^O_{11s} + 3 \cdot \sqrt{70} \cdot x \cdot y^2 \cdot Q^O_{11s} - 3 \cdot \sqrt{70} \cdot x^2 \cdot z \cdot Q^O_{10} + 3 \cdot \sqrt{70} \cdot y^2 \cdot z \cdot Q^O_{10} + 6 \cdot \sqrt{70} \cdot x \cdot y \cdot z \cdot Q^O_{11c} + \sqrt{210} \cdot x^2 \cdot Q^O_{21c} - \sqrt{210} \cdot y^2 \cdot Q^O_{21c} - 2 \cdot \sqrt{210} \cdot x \cdot y \cdot Q^O_{21s} + 2 \cdot \sqrt{210} \cdot x \cdot z \cdot Q^O_{22c} - 2 \cdot \sqrt{210} \cdot y \cdot z \cdot Q^O_{22s} - 2 \cdot \sqrt{42} \cdot x \cdot Q^O_{32c} + 2 \cdot \sqrt{42} \cdot y \cdot Q^O_{32s} - 4 \cdot rt7 \cdot z \cdot Q^O_{33c})$$
(A.225)

$$Q^C_{43s} = Q^O_{43s} + 0.25 \cdot (3 \cdot \sqrt{70} \cdot x^2 \cdot y \cdot z \cdot Q^O_{00} - \sqrt{70} \cdot y^3 \cdot z \cdot Q^O_{00} - 3 \cdot \sqrt{70} \cdot x^2 \cdot y \cdot Q^O_{11s} + \sqrt{70} \cdot y^3 \cdot Q^O_{11s} - 6 \cdot \sqrt{70} \cdot x \cdot y \cdot z \cdot Q^O_{10} - 3 \cdot \sqrt{70} \cdot x^2 \cdot z \cdot Q^O_{11c} + 3 \cdot \sqrt{70} \cdot y^2 \cdot z \cdot Q^O_{11c} + 2 \cdot \sqrt{210} \cdot x \cdot y \cdot Q^O_{21c} + \sqrt{210} \cdot x^2 \cdot Q^O_{21s} - \sqrt{210} \cdot y^2 \cdot Q^O_{21s} + 2 \cdot \sqrt{210} \cdot y \cdot z \cdot Q^O_{22c} + 2 \cdot \sqrt{210} \cdot x \cdot z \cdot Q^O_{22s} - 2 \cdot \sqrt{42} \cdot y \cdot Q^O_{32c} - 2 \cdot \sqrt{42} \cdot x \cdot Q^O_{32s} - 4 \cdot rt7 \cdot z \cdot Q^O_{33s})$$
(A.226)

$$Q^C_{44c} = Q^O_{44c} + 0.125 \cdot (\sqrt{35} \cdot x^4 \cdot Q^O_{00} - 6 \cdot \sqrt{35} \cdot x^2 \cdot y^2 \cdot Q^O_{00} + \sqrt{35} \cdot y^4 \cdot Q^O_{00} - 4 \cdot \sqrt{35} \cdot x^3 \cdot Q^O_{10} + 12 \cdot \sqrt{35} \cdot x \cdot y^2 \cdot Q^O_{10} + 12 \cdot \sqrt{35} \cdot x^2 \cdot y \cdot Q^O_{11c} - 4 \cdot \sqrt{35} \cdot y^3 \cdot Q^O_{11c} + 4 \cdot \sqrt{105} \cdot x^2 \cdot Q^O_{22c} - 4 \cdot \sqrt{105} \cdot y^2 \cdot Q^O_{22c} - 8 \cdot \sqrt{105} \cdot x \cdot y \cdot Q^O_{22s} - 8 \cdot \sqrt{14} \cdot x \cdot Q^O_{33c} + 8 \cdot \sqrt{14} \cdot y \cdot Q^O_{33s}) \tag{A.227}$$

$$Q^C_{44s} = Q^O_{44s} + 0.5 \cdot (\sqrt{35} \cdot x^3 \cdot y \cdot Q^O_{00} - \sqrt{35} \cdot x \cdot y^3 \cdot Q^O_{00} - 3 \cdot \sqrt{35} \cdot x^2 \cdot y \cdot Q^O_{10} + \sqrt{35} \cdot y^3 \cdot Q^O_{10} - \sqrt{35} \cdot x^3 \cdot Q^O_{11c} + 3 \cdot \sqrt{35} \cdot x \cdot y^2 \cdot Q^O_{11c} + 2 \cdot \sqrt{105} \cdot x \cdot y \cdot Q^O_{22c} + \sqrt{105} \cdot x^2 \cdot Q^O_{22s} - \sqrt{105} \cdot y^2 \cdot Q^O_{22s} - 2 \cdot \sqrt{14} \cdot y \cdot Q^O_{33c} - 2 \cdot \sqrt{14} \cdot x \cdot Q^O_{33s}) \tag{A.228}$$

A.3 Polarizability Translations

One can employ similar techniques to derive translation expressions for polarizabilities. Sample derivations are provided in the Supporting Information of Ref [88]. Here, a complete set of polarizability translations up to rank 2 (quadrupole-quadrupole) are provided.

In the following expression, $\alpha_{lm,l'm'}$ refers to an original, un-translated polarizability, and $\alpha'_{lm,l'm'}$ is the polarizability following translation to the new coordinates. Since the polarizability tensor is symmetric $\alpha'_{lm,l'm'} = \alpha'_{l'm',lm}$, only symmetrically unique elements are given below.

As noted in the main paper, the charge polarizability is zero, and the dipole-dipole polarizabilities do not change upon translation. See Ref [113] for more details.

Dipole-Quadrupole:

$$\alpha'_{11c,20} = \alpha_{11c,20} + \alpha_{11c,11c}x + \alpha_{11c,11s}y - 2\alpha_{11c,10}z \tag{A.229}$$

$$\alpha'_{11c,21c} = \alpha_{11c,21c} - \sqrt{3}\alpha_{11c,10}x - \sqrt{3}\alpha_{11c,11c}z \tag{A.230}$$

$$\alpha'_{11c,21s} = \alpha_{11c,21s} - \sqrt{3}\alpha_{11c,10}y - \sqrt{3}\alpha_{11c,11s}z \tag{A.231}$$

$$\alpha'_{11c,22c} = \alpha_{11c,22c} - \sqrt{3}\alpha_{11c,11c}x + \sqrt{3}\alpha_{11c,11s}y \tag{A.232}$$

$$\alpha'_{11c,22s} = \alpha_{11c,22s} - \sqrt{3}\alpha_{11c,11s}x - \sqrt{3}\alpha_{11c,11c}y \tag{A.233}$$

$$\alpha'_{11s,20} = \alpha_{11s,20} + \alpha_{11c,11s}x + \alpha_{11s,11s}y - 2\alpha_{11s,10}z \tag{A.234}$$

$$\alpha'_{11s,21c} = \alpha_{11s,21c} - \sqrt{3}\alpha_{11s,10}x - \sqrt{3}\alpha_{11c,11s}z \tag{A.235}$$

$$\alpha'_{11s,21s} = \alpha_{11s,21s} - \sqrt{3}\alpha_{11s,10}y - \sqrt{3}\alpha_{11s,11s}z \tag{A.236}$$

$$\alpha'_{11s,22c} = \alpha_{11s,22c} - \sqrt{3}\alpha_{11c,11s}x + \sqrt{3}\alpha_{11s,11s}y \tag{A.237}$$

$$\alpha'_{11s,22s} = \alpha_{11s,22s} - \sqrt{3}\alpha_{11s,11s}x - \sqrt{3}\alpha_{11c,11s}y \tag{A.238}$$

$$\alpha'_{10,20} = \alpha_{10,20} + \alpha_{11c,10}x + \alpha_{11s,10}y - 2\alpha_{10,10}z \tag{A.239}$$

$$\alpha'_{10,21c} = \alpha_{10,21c} - \sqrt{3}\alpha_{10,10}x - \sqrt{3}\alpha_{11c,10}z \tag{A.240}$$

$$\alpha'_{10,21s} = \alpha_{10,21s} - \sqrt{3}\alpha_{10,10}y - \sqrt{3}\alpha_{11s,10}z \tag{A.241}$$

$$\alpha'_{10,22c} = \alpha_{10,22c} - \sqrt{3}\alpha_{11c,10}x + \sqrt{3}\alpha_{11s,10}y \tag{A.242}$$

$$\alpha_{10,22s} = \alpha_{10,22s} - \sqrt{3}\alpha_{11s,10}x - \sqrt{3}\alpha_{11c,10}y \tag{A.243}$$

Quadrupole-quadrupole:

$$\alpha_{20,20} = \alpha_{20,20} + 2\alpha_{11c,20}x + \alpha_{11c,11c}xx + 2\alpha_{11s,20}y + 2\alpha_{11c,11s}xy + \alpha_{11s,11s}yy - 4\alpha_{10,20}z - 4\alpha_{11c,10}xz - 4\alpha_{11s,10}yz + 4\alpha_{10,10}zz \quad \text{(A.244)}$$

$$\alpha_{20,21c} = \alpha_{20,21c} - \sqrt{3}\alpha_{10,20}x + \alpha_{11c,21c}x - \sqrt{3}\alpha_{11c,10}xx + \alpha_{11s,21c}y - \sqrt{3}\alpha_{11s,10}xy - \sqrt{3}\alpha_{11c,20}z - 2\alpha_{10,21c}z - \sqrt{3}\alpha_{11c,11c}xz + 2\sqrt{3}\alpha_{10,10}xz - \sqrt{3}\alpha_{11c,11s}yz + 2\sqrt{3}\alpha_{11c,10}zz \quad \text{(A.245)}$$

$$\alpha_{20,21s} = \alpha_{20,21s} + \alpha_{11c,21s}x - \sqrt{3}\alpha_{10,20}y + \alpha_{11s,21s}y - \sqrt{3}\alpha_{11c,10}xy - \sqrt{3}\alpha_{11s,10}yy - \sqrt{3}\alpha_{11s,20}z - 2\alpha_{10,21s}z - \sqrt{3}\alpha_{11c,11s}xz - \sqrt{3}\alpha_{11s,11s}yz + 2\sqrt{3}\alpha_{10,10}yz + 2\sqrt{3}\alpha_{11s,10}zz \quad \text{(A.246)}$$

$$\alpha_{20,22c} = \alpha_{20,22c} - \sqrt{3}\alpha_{11c,20}x + \alpha_{11c,22c}x - \sqrt{3}\alpha_{11c,11c}xx + \sqrt{3}\alpha_{11s,20}y + \alpha_{11s,22c}y + \sqrt{3}\alpha_{11s,11s}yy - 2\alpha_{10,22c}z + 2\sqrt{3}\alpha_{11c,10}xz - 2\sqrt{3}\alpha_{11s,10}yz \quad \text{(A.247)}$$

$$\alpha_{20,22s} = \alpha_{20,22s} - \sqrt{3}\alpha_{11s,20}x + \alpha_{11c,22s}x - \sqrt{3}\alpha_{11c,11s}xx - \sqrt{3}\alpha_{11c,20}y + \alpha_{11s,22s}y$$
$$- \sqrt{3}\alpha_{11c,11c}xy - \sqrt{3}\alpha_{11s,11s}xy - \sqrt{3}\alpha_{11c,11s}yy - 2\alpha_{10,22s}z + 2\sqrt{3}\alpha_{11s,10}xz$$
$$+ 2\sqrt{3}\alpha_{11c,10}yz \quad \text{(A.248)}$$

$$\alpha_{21c,21c} = \alpha_{21c,21c} - 2\sqrt{3}\alpha_{11c,21c}z - 2\sqrt{3}\alpha_{10,21c}x + 3\alpha_{10,10}xx + 3\alpha_{11c,11c}zz + 6\alpha_{11c,10}xz$$
$$\text{(A.249)}$$

$$\alpha_{21c,21s} = \alpha_{21c,21s} - \sqrt{3}\alpha_{11s,21c}z - \sqrt{3}\alpha_{10,21c}y - \sqrt{3}\alpha_{11c,21s}z + 3\alpha_{11c,11s}zz + 3\alpha_{11c,10}zy$$
$$- \sqrt{3}\alpha_{10,21s}x + 3\alpha_{11s,10}xz + 3\alpha_{10,10}xy \quad \text{(A.250)}$$

$$\alpha_{21c,22c} = \alpha_{21c,22c} + \sqrt{3}\alpha_{11s,21c}y - \sqrt{3}\alpha_{11c,21c}x - \sqrt{3}\alpha_{11c,22c}z - 3\alpha_{11c,11s}yz + 3\alpha_{11c,11c}xz$$
$$- \sqrt{3}\alpha_{10,22c}x - 3\alpha_{11s,10}xy + 3\alpha_{11c,10}xx \quad \text{(A.251)}$$

$$\alpha_{21c,22s} = \alpha_{21c,22s} - \sqrt{3}\alpha_{11s,21c}x - \sqrt{3}\alpha_{11c,21c}y - \sqrt{3}\alpha_{11c,22s}z + 3\alpha_{11c,11s}xz + 3\alpha_{11c,11c}yz$$
$$- \sqrt{3}\alpha_{10,22s}x + 3\alpha_{11s,10}xx + 3\alpha_{11c,10}xy \quad \text{(A.252)}$$

$$\alpha_{21s,21s} = \alpha_{21s,21s} - 2\sqrt{3}\alpha_{10,21s}y + 3\alpha_{10,10}yy - 2\sqrt{3}\alpha_{11s,21s}z + 6\alpha_{11s,10}yz + 3\alpha_{11s,11s}zz \tag{A.253}$$

$$\alpha_{21s,22c} = \alpha_{21s,22c} - \sqrt{3}\alpha_{11c,21s}x + \sqrt{3}\alpha_{11s,21s}y - \sqrt{3}\alpha_{10,22c}y + 3\alpha_{11c,10}xy - 3\alpha_{11s,10}yy - \sqrt{3}\alpha_{11s,22c}z + 3\alpha_{11c,11s}xz - 3\alpha_{11s,11s}yz \tag{A.254}$$

$$\alpha_{21s,22s} = \alpha_{21s,22s} - \sqrt{3}\alpha_{11s,21s}x - \sqrt{3}\alpha_{11c,21s}y - \sqrt{3}\alpha_{10,22s}y + 3\alpha_{11s,10}xy + 3\alpha_{11c,10}yy - \sqrt{3}\alpha_{11s,22s}z + 3\alpha_{11s,11s}xz + 3\alpha_{11c,11s}yz \tag{A.255}$$

$$\alpha_{22c,22c} = \alpha_{22c,22c} - 2\sqrt{3}\alpha_{11c,22c}x + 3\alpha_{11c,11c}xx + 2\sqrt{3}\alpha_{11s,22c}y - 6\alpha_{11c,11s}xy + 3\alpha_{11s,11s}yy \tag{A.256}$$

$$\alpha_{22c,22s} = \alpha_{22c,22s} - \sqrt{3}\alpha_{11s,22c}x - \sqrt{3}\alpha_{11c,22s}x + 3\alpha_{11c,11s}xx - \sqrt{3}\alpha_{11c,22c}y + \sqrt{3}\alpha_{11s,22s}y + 3\alpha_{11c,11c}xy - 3\alpha_{11s,11s}xy - 3\alpha_{11c,11s}yy \tag{A.257}$$

$$\alpha_{22s,22s} = \alpha_{22s,22s} - 2\sqrt{3}\alpha_{11s,22s}x + 3\alpha_{11s,11s}xx - 2\sqrt{3}\alpha_{11c,22s}y + 6\alpha_{11c,11s}xy + 3\alpha_{11c,11c}yy \tag{A.258}$$

A.4 Radial Distribution Functions for Pyrimidine in Water

The distribution of the CG sites as determined by K-means reproduces the general structure of the explicit solvent. Figure A.1 compares the radial distribution function (RDF) for pyrimidine in water using both the explicit and coarse-grained models. The roughness of the ACPE RDF stems from the smaller number of data points in the ACPE representation.

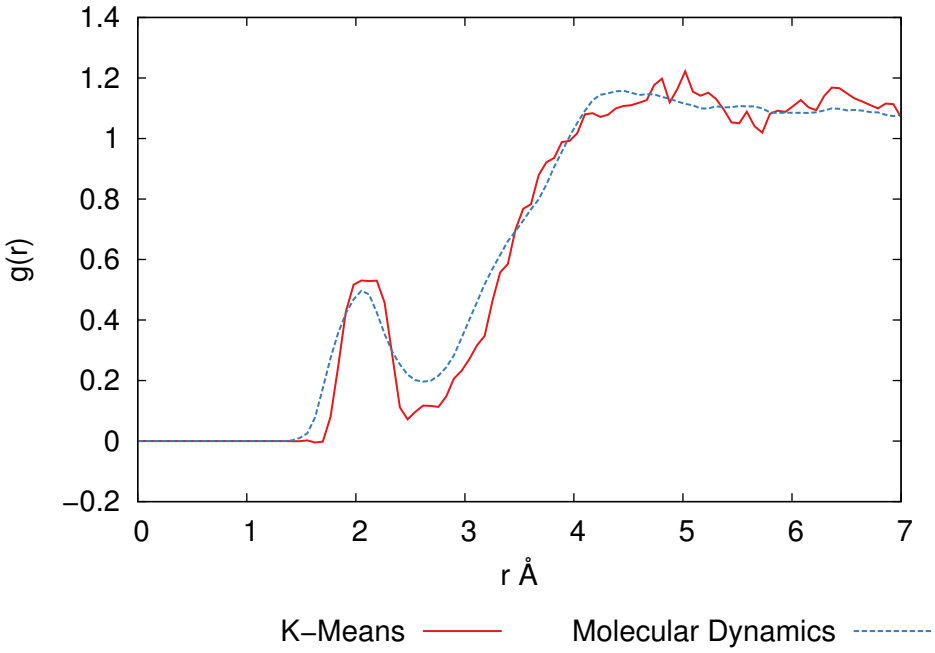


Figure A.1: RDF of the pyrimidine N to H of water as computed with from the explicit MD simulation configurations or from the coarse-grained representation generated via K-Means.

A.5 Convergence of the Excitation Energies with Configuration Sampling

Figure A.2 demonstrates that the excitation energies are well converged with respect to the number of configurations included in the configurational average. By 400 configurations, the variations are a few hundredths of an eV or less.

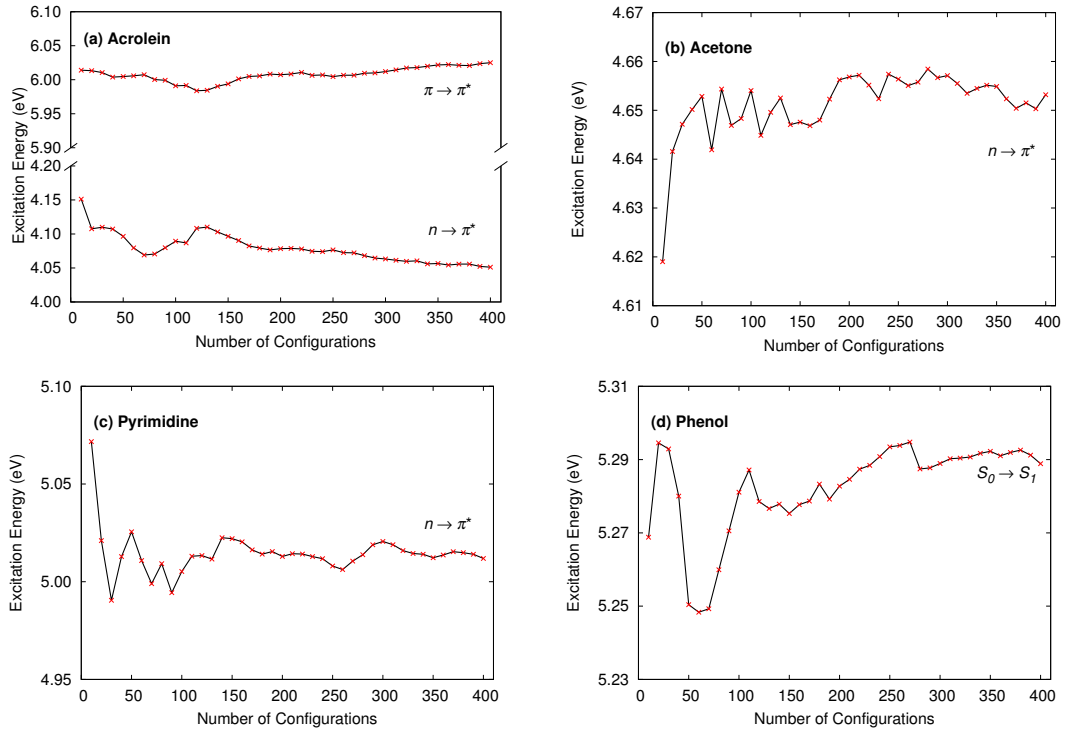


Figure A.2: Convergence of the configurational average for the vertical excitation energies with increasing number of configurations (a) acrolein, (b) acetone, (c) pyrimidine, (d) phenol.

A.6 Sensitivity of ACPE Excitation Energies to the K-means Initial Guess

The K-means clustering algorithm used to determine coarse-grained sites in the ACPE model begins with a random initial guess. Three trials with different random seeds were performed for each of the aqueous solution examples. Table A.1 shows that the resulting excitation energies vary only be a few hundredths of an eV or less with different initial guesses. The ACPE excitation energies reported in Table 3 of the main paper are averages of these values.

Table A.1: Variation in the ACPE excitation energies with different random initial guesses for the K-means algorithm.

Trial	Acrolein	Acetone	Pyrimidine
	$E\ (n \rightarrow \pi*)$	$E\ (n \rightarrow \pi*)$	$E\ (n \rightarrow \pi*)$
1	4.09	4.64	5.01
2	4.07	4.62	4.97
3	4.05	4.66	5.05
	$E\ (\pi \rightarrow \pi*)$	–	–
1	6.06	–	–
2	6.07	–	–
3	6.06	–	–

Appendix B

Always Stable Predictor Corrector

B.1 Transferability Across Different Chemical Systems

In addition to validation between self consistent field (SCF) solvers we have also tested the behaviour of the N-step predictor in two different chemical systems. In Figure B.1 we demonstrate the energy drift behavior for three SCF iterations is practically identical between a system of 500 water molecules and a system of a ubiquitin protein solvated by 2835 water molecules. Future testing should focus on the behavior of more chemical systems.

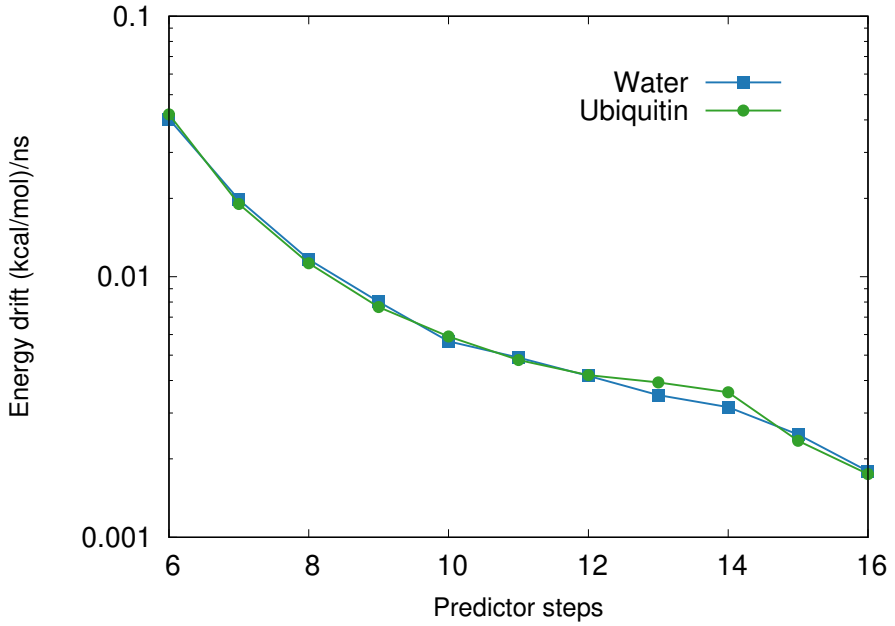


Figure B.1: Comparison of the energy drift (kcal mol^{-1} ns^{-1} per atom) with different N-steps predictors for a water box and ubiquitin. Each data point was obtained from a 1 ns NVE simulation using three iterations of the DC-JI/DIIS polarization solver.

B.2 Predictor Coefficients

The Always Stable Predictor Corrector (ASPC) uses a history-based predictor for the induced dipoles,

$$\boldsymbol{\mu}^p(t+1) = \sum_{j=0}^{k+1} B_{j+1}\boldsymbol{\mu}(t-jh) \tag{B.1}$$

where $\boldsymbol{\mu}^p(t+1)$ is the predicted dipole, B_{j+1} are the scaling coefficients and $\boldsymbol{\mu}(t-jh)$ are the induced dipoles from previous time steps. The time step size is h and $k+2$ is the total number of values stored in history. The coefficients of the ASPC predictor are derived such that the predictor error contributions that are time irreversible are zeroed out.[13] A recursive description of the coefficients that maintains this property is described below, followed by the coefficients worked out to the 25-step predictor.

B.2.1 Recursive Form for the Predictor Coefficients

These following recursive expressions for B_i are taken directly from Ref [13]:

$B_1 = 1(4k+6)\frac{1}{(k+3)}$

$B_2 = -2(4k+6)\frac{(k+1)}{(k+3)(k+4)}$

$B_3 = 3(4k+6)\frac{(k+1)(k+0)}{(k+3)(k+4)(k+5)}$

$B_4 = -4(4k+6)\frac{(k+1)(k+0)(k-1)}{(k+3)(k+4)(k+5)(k+6)}$

...

$B_i = 0$ for i $> k+2$

B.2.2 The Predictor Coefficients for N=2–25-Step Predictors

The present study considered up to 16-step predictors, but one conceivably could wish to explore even longer-history predictors. The expressions have been derived here for all N-step predictors ranging from N=2–25.

2-step predictor ($k = 0$)

$B_1 = 2$

$B_2 = -1$

3-step predictor ($k = 1$)

$B_1 = 5/2$

$B_2 = -2$

$B_3 = 1/2$

4-step predictor ($k = 2$)

$B_1 = 14/5$

$B_2 = -14/5$

$B_3 = 6/5$

$B_4 = -1/5$

5-step predictor ($k = 3$)

$B_1 = 3$

$B_2 = -24/7$

$B_3 = 27/14$

$B_4 = -4/7$

$B_5 = 1/14$

6-step predictor ($k = 4$)

$B_1 = 22/7$

$B_2 = -55/14$

$B_3 = 55/21$

$B_4 = -22/21$

$B_5 = 5/21$

$B_6 = -1/42$

7-step predictor ($k = 5$)

$B_1 = 13/4$

$B_2 = -13/3$

$B_3 = 13/4$

$B_4 = -52/33$

$B_5 = 65/132$

$B_6 = -1/11$

$B_7 = 1/132$

8-step predictor ($k = 6$)

$B_1 = 10/3$

$B_2 = -14/3$

$B_3 = 42/11$

$B_4 = -70/33$

$B_5 = 350/429$

$B_6 = -30/143$

$B_7 = 14/429$

$B_8 = -1/429$

9-step predictor ($k = 7$)

$B_1 = 17/5$

$B_2 = -272/55$

$B_3 = 238/55$

$B_4 = -1904/715$

$B_5 = 170/143$

$B_6 = -272/715$

$B_7 = 119/1430$

$B_8 = -8/715$

$B_9 = 1/1430$

10-step predictor ($k = 8$)

$B_1 = 38/11$

$B_2 = -57/11$

$B_3 = 684/143$

$B_4 = -456/143$

$B_5 = 228/143$

$B_6 = -171/286$

$B_7 = 399/2431$

$B_8 = -76/2431$

$B_9 = 9/2431$

$B_{10} = -1/4862$

11-step predictor ($k = 9$)

$B_1 = 7/2$

$B_2 = -70/13$

$B_3 = 135/26$

$B_4 = -48/13$

$B_5 = 105/52$

$B_6 = -189/221$

$B_7 = 245/884$

$B_8 = -280/4199$

$B_9 = 68/6043$

$B_{10} = -5/4199$

$B_{11} = 1/16796$

12-step predictor ($k = 10$)

$B_1 = 46/13$

$B_2 = -506/91$

$B_3 = 506/91$

$B_4 = -759/182$

$B_5 = 3795/1547$

$B_6 = -253/221$

$B_7 = 1771/4199$

$B_8 = -506/4199$

$B_9 = 208/8055$

$B_{10} = -115/29393$

$B_{11} = 11/29393$

$B_{12} = -1/58786$

13-step predictor $(k = 11)$

$B_1 = 25/7$

$B_2 = -40/7$

$B_3 = 165/28$

$B_4 = -550/119$

$B_5 = 1375/476$

$B_6 = -559/383$

$B_7 = 385/646$

$B_8 = -440/2261$

$B_9 = 225/4522$

$B_{10} = -167/17369$

$B_{11} = 27/20423$

$B_{12} = -6/52003$

$B_{13} = 1/208012$

14-step predictor $(k = 12)$

$B_1 = 18/5$

$B_2 = -117/20$

$B_3 = 1053/170$

$B_4 = -429/85$

$B_5 = 2145/646$

$B_6 = -875/488$

$B_7 = 1287/1615$

$B_8 = -468/1615$

$B_9 = 236/2775$

$B_{10} = -59/2997$

$B_{11} = 47/13565$

$B_{12} = -10/22929$

$B_{13} = 13/371450$

$B_{14} = -1/742900$

15-step predictor ($k = 13$)

$B_1 = 29/8$

$B_2 = -203/34$

$B_3 = 2639/408$

$B_4 = -2511/461$

$B_5 = 2655/709$

$B_6 = -1821/851$

$B_7 = 2639/2584$

$B_8 = -557/1372$

$B_9 = 363/2725$

$B_{10} = -126/3547$

$B_{11} = 222/29543$

$B_{12} = -17/13998$

$B_{13} = 1/7094$

$B_{14} = -4/382063$

$B_{15} = 1/2674440$

16-step predictor ($k = 14$)

$B_1 = 62/17$

$B_2 = -310/51$

$B_3 = 2170/323$

$B_4 = -2329/400$

$B_5 = 1701/409$

$B_6 = -806/323$

$B_7 = 1024/809$

$B_8 = -479/883$

$B_9 = 257/1316$

$B_{10} = -434/7429$

$B_{11} = 191/13375$

$B_{12} = -62/22287$

$B_{13} = 3/7217$

$B_{14} = -3/67015$

$B_{15} = 2/646323$

$B_{16} = -1/9694845$

17-step predictor $(k = 15)$

$B_1 = 11/3$

$B_2 = -352/57$

$B_3 = 132/19$

$B_4 = -352/57$

$B_5 = 260/57$

$B_6 = -1248/437$

$B_7 = 2002/1311$

$B_8 = -659/944$

$B_9 = 594/2185$

$B_{10} = -352/3933$

$B_{11} = 73/2966$

$B_{12} = -70/12601$

$B_{13} = 572/570285$

$B_{14} = -7/50224$

$B_{15} = 3/214289$

$B_{16} = -1/1104927$

$B_{17} = 1/35357670$

18-step predictor ($k = 16$)

$B_1 = 70/19$

$B_2 = -119/19$

$B_3 = 136/19$

$B_4 = -1360/209$

$B_5 = 1114/225$

$B_6 = -1342/417$

$B_7 = 2779/1542$

$B_8 = -1904/2185$

$B_9 = 476/1311$

$B_{10} = -170/1311$

$B_{11} = 29/737$

$B_{12} = -206/20567$

$B_{13} = 44/20951$

$B_{14} = -55/155636$

$B_{15} = 25/544726$

$B_{16} = -2/463017$

$B_{17} = 1/3813082$

$B_{18} = -1/129644790$

19-step predictor ($k = 17$)

$B_1 = 37/10$

$B_2 = -222/35$

$B_3 = 1316/179$

$B_4 = -3055/448$

$B_5 = 2131/400$

$B_6 = -1833/512$

$B_7 = 1276/611$

$B_8 = -611/576$

$B_9 = 421/898$

$B_{10} = -2516/14007$

$B_{11} = 131/2210$

$B_{12} = -172/10307$

$B_{13} = 53/13402$

$B_{14} = -44/56823$

$B_{15} = 7/57374$

$B_{16} = -6/403411$

$B_{17} = 1/759362$

$B_{18} = -1/13267742$

$B_{19} = 1/477638700$

20-step predictor ($k = 18$)

$B_1 = 26/7$

$B_2 = -494/77$

$B_3 = 2764/367$

$B_4 = -4346/611$

$B_5 = 3124/549$

$B_6 = -1627/413$

$B_7 = 1611/676$

$B_8 = -1611/1274$

$B_9 = 1069/1816$

$B_{10} = -271/1130$

$B_{11} = 229/2691$

$B_{12} = -717/27461$

$B_{13} = 223/32521$

$B_{14} = -31/20390$

$B_{15} = 27/96688$

$B_{16} = -1/24172$

$B_{17} = 2/420877$

$B_{18} = -1/2517469$

$B_{19} = 1/46506926$

$B_{20} = -1/1767263190$

21-step predictor ($k = 19$)

$B_1 = 41/11$

$B_2 = -1640/253$

$B_3 = 3895/506$

$B_4 = -3451/467$

$B_5 = 761/126$

$B_6 = -5287/1231$

$B_7 = 1573/586$

$B_8 = -1013/684$

$B_9 = 844/1169$

$B_{10} = -345/1111$

$B_{11} = 941/8014$

$B_{12} = -223/5745$

$B_{13} = 37/3324$

$B_{14} = -30/10949$

$B_{15} = 31/54307$

$B_{16} = -11/111406$

$B_{17} = 3/217331$

$B_{18} = -1/667085$

$B_{19} = 1/8426342$

$B_{20} = -1/164103010$

$B_{21} = 1/6564120420$

22-step predictor ($k = 20$)

$B_1 = 86/23$

$B_2 = -301/46$

$B_3 = 903/115$

$B_4 = -2739/358$

$B_5 = 1154/181$

$B_6 = -13889/2990$

$B_7 = 1791/599$

$B_8 = -938/549$

$B_9 = 1579/1819$

$B_{10} = -355/906$

$B_{11} = 71/453$

$B_{12} = -142/2567$

$B_{13} = 146/8527$

$B_{14} = -71/15402$

$B_{15} = 17/15919$

$B_{16} = -3/14297$

$B_{17} = 3/87464$

$B_{18} = -1/220280$

$B_{19} = 1/2139034$

$B_{20} = -1/28449148$

$B_{21} = 1/582530167$

$B_{22} = -1/24466267020$

23-step predictor ($k = 21$)

$B_1 = 15/4$

$B_2 = -33/5$

$B_3 = 2079/260$

$B_4 = -308/39$

$B_5 = 1045/156$

$B_6 = -1881/377$

$B_7 = 1359/412$

$B_8 = -1397/718$

$B_9 = 2482/2419$

$B_{10} = -1021/2111$

$B_{11} = 226/1111$

$B_{12} = -342/4495$

$B_{13} = 180/7147$

$B_{14} = -295/40243$

$B_{15} = 12/6451$

$B_{16} = -14/34397$

$B_{17} = 7/92496$

$B_{18} = -2/170555$

$B_{19} = 1/678629$

$B_{20} = -1/6930497$

$B_{21} = 1/96806946$

$B_{22} = -1/2079149174$

$B_{23} = 1/91482563640$

24-step predictor ($k = 22$)

$B_1 = 94/25$

$B_2 = -2162/325$

$B_3 = 2179/268$

$B_4 = -2179/268$

$B_5 = 1535/219$

$B_6 = -5002/939$

$B_7 = 1263/350$

$B_8 = -1882/859$

$B_9 = 386/323$

$B_{10} = -809/1381$

$B_{11} = 457/1773$

$B_{12} = -211/2078$

$B_{13} = 435/12193$

$B_{14} = -128/11509$

$B_{15} = 101/33056$

$B_{16} = -23/31365$

$B_{17} = 6/39467$

$B_{18} = -3/111823$

$B_{19} = 3/759220$

$B_{20} = -1/2115693$

$B_{21} = 1/22668139$

$B_{22} = -1/331779123$

$B_{23} = 1/7457817688$

$B_{24} = -1/343059613650$

25-step predictor ($k = 23$)

$B_1 = 49/13$

$B_2 = -784/117$

$B_3 = 322/39$

$B_4 = -5729/686$

$B_5 = 1878/257$

$B_6 = -809/143$

$B_7 = 3237/826$

$B_8 = -1627/666$

$B_9 = 1829/1331$

$B_{10} = -1523/2182$

$B_{11} = 564/1763$

$B_{12} = -508/3847$

$B_{13} = 277/5660$

$B_{14} = -155/9558$

$B_{15} = 7/1465$

$B_{16} = -43/34591$

$B_{17} = 17/60065$

$B_{18} = -14/251105$

$B_{19} = 4/427229$

$B_{20} = -1/761002$

$B_{21} = 1/6667825$

$B_{22} = -1/74785723$

$B_{23} = 1/1144546715$

$B_{24} = -1/26873003069$

$B_{25} = 1/1289904147324$