Speeding up electrostatic computations for molecular dynamics

Ramamoorthi Anandakrishnan

Dissertation submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science and Applications

Alexey V. Onufriev, Chair
David R. Bevan
Yang Cao
Adrian Sandu
Jianhua Xing

October 18, 2011
Blacksburg, Virginia

Keywords: Molecular Dynamics, Biomolecular Electrostatics, Statistical Mechanics
Copyright 2011, Ramu Anandakrishnan
Molecular dynamics (MD) simulations are routinely used to study the structure and function of biological molecules. However, the accuracy and duration of these simulations are constrained by their computational costs, thus limiting the ability to accurately simulate systems of realistic sizes over biologically relevant time periods. The two most computationally demanding steps in these simulations are (1) determining the charge state of ionizable sites in biomolecules, which is a key input to the simulation, and (2) calculating long range electrostatic interactions during the simulation. Presented here are two novel methods, the direct interaction approximation (DIA) and the hierarchical charge partitioning (HCP) approximation, for speeding up each of these two computations.

The average charge state of ionizable sites in biomolecules can be calculated as the statistical average over all possible \(2^N\) microstates for a molecule, where \(N\) is the number of ionizable sites. In general, this computation scales exponentially as \(O(N^2 2^N)\). The DIA is an \(O(N^2)\) approximation for calculating the average charge state of ionizable sites. For each site, the DIA treats direct interactions (interactions involving the site of interest) exactly, while using an average value for indirect interactions (interactions not involving the site of interest). The DIA was tested on two problems. The computation of thermal average properties for the 2-D Ising model of ferromagnetism, and the average charge state of ionizable residues in biomolecules. Compared to the commonly used non-deterministic Monte Carlo method, for the same computational cost, the deterministic DIA was found to be at least as accurate, as measured by RMS error relative to the exact computation. Thus, the DIA may be a practical alternative to the Monte Carlo method for some problems.

In atomistic MD simulations, the computation of long range electrostatic interactions scale as \(O(n^2)\), where \(n\) is the number of atoms. For most biologically relevant timescales the simulations involve \(10^{12-16}\) simulation steps. Thus, the computational cost of long range interactions becomes the limiting factor in the size and duration of MD simulations. The HCP is an \(O(n \log n)\) approximation for computing long range electrostatic interactions. The approximation is based on multiple levels of natural partitioning of biomolecular structures into a hierarchical set of components. For components that are far from the point of interest, the charge distribution for each component is approximated by a much smaller number of charges. For nearby components, the HCP uses the full set of atomic charges. For large structures the HCP can be several orders of magnitude faster than the exact pairwise \(O(n^2)\) all-atom computation. For a representative set of structures, the accuracy of the HCP is comparable to the industry standard explicit solvent particle mesh Ewald (PME), and in general, more accurate than the spherical cutoff method. And, unlike the PME, the DIA can be easily extended to implicit solvent GB models. 50 ns implicit solvent simulations for a representative set of four biomolecules suggests that the HCP could be a practical alternative for implicit solvent simulations, and preferable to the cutoff based method. The HCP is available for general use in the open source MD software, NAB within AmberTools.
This work was supported primarily by NIH grant GM076121, and partially by NSF I/UCRC grant IIP-0804155 with support from AMD.
Dedication

To my wife Pam, for her love and unconditional support.
Acknowledgments

I would like to thank first and foremost Dr. Alexey Onufriev for his guidance, support, and advice. I would also like to thank Dr. Liqing Zhang for her help on the “functional bias” project, and Dr. Wu Feng for his advice on the GPU projects.

In addition, many of my fellow students and colleagues have contributed directly and indirectly to the success of my research. In particular I would like to thank Boris Aguilar, Casey Baker, Puranjoy Battacharjee, Mayank Daga, Andrew Fenley, John Gordon, John Myers, Abhishek Mukhopadyay, Corban Rivera, Tom Scogland, Igor Tolokh, Jory Ruscio, Grigori Sigalov, Venkat Srinivasan, and Andrew Warren.
# Contents

Organization of the thesis xiv

## 1 Extended abstract 1

1.1 Speeding up the computation of average charge state 2

1.1.1 Error bound for simple clustering algorithms 2

1.1.2 Direct interaction approximation (DIA) 3

1.1.3 DIA using a binary split-merge algorithm 3

1.2 Speeding up the computation of long range electrostatic interactions 4

1.2.1 Hierarchical charge partitioning (HCP) 4

1.2.2 HCP for the implicit solvent generalized Born model 5

1.2.3 HCP implementation on GPUs for computing electrostatic surface potential 5

1.2.4 HCP implementation on GPUs for molecular dynamics 6

## 2 Speeding up the computation of average charge state 7

2.1 Introduction 7

2.2 Error bound for simple clustering algorithms 10

2.2.1 Introduction 10

2.2.2 Methods 10

2.2.2.1 Basic clustering algorithms 10

2.2.2.2 Theoretical error bound 15

2.2.3 Test setup 17

2.2.3.1 Description of model systems 18

2.2.3.2 Baseline for calculating error 18
# List of Figures

2.1 Illustrative five particle system ......................................................... 12  
2.2 Global vs. local clustering ................................................................. 14  
2.3 Example of system with $Err \approx Err_{max}$ ........................................ 17  
2.4 RMS error for inverse distance-squared interactions .......................... 20  
2.5 Error bound for inverse distance-squared interactions ........................ 21  
2.6 RMS error for inverse distance interactions ....................................... 22  
2.7 Error bound for inverse distance interactions ..................................... 23  
2.8 RMS error as a function of the error bound ....................................... 24  
2.9 Protein structure for 1VII ................................................................. 25  
2.10 RMS error for sample structures ..................................................... 27  
2.11 RMS error as a function of error bound for sample structures .............. 28  
2.12 Comparison of interaction potentials ................................................. 29  
2.13 Accuracy of the DIA ........................................................................ 35  
2.14 Partition function for the 5x5 Ising model ......................................... 43  
2.15 Probability of magnetization for the 5x5 Ising model ........................... 44  
2.16 Per particle thermal average properties of the 5x5 Ising model ............ 45  
2.17 Average charge state for 60 ionizable sites in 3 biomolecules .............. 46  

3.1 Natural partitioning of a chromatin fibre ............................................. 53  
3.2 Illustration of the HCP algorithm ....................................................... 55  
3.3 Accuracy and speed-up comparison ..................................................... 61  
3.4 Accuracy as a function of size/charge ................................................. 62  
3.5 Trade-offs between accuracy and speed .............................................. 63  
3.6 Geometric setup used for estimating discontinuity ............................. 64
| 3.7 | Effect of the smoothing function | 66 |
| 3.8 | Violation of Newton’s third law | 67 |
| 3.9 | 10 ns simulation of 1VII | 70 |
| 3.10 | The HCP threshold distance | 75 |
| 3.11 | Accuracy of the HCP-GB and cutoff-GB methods | 85 |
| 3.12 | Speedup for the HCP-GB and cutoff-GB methods | 86 |
| 3.13 | HCP-GB: Tradeoff between accuracy and speed | 86 |
| 3.14 | Accuracy of effective Born radii approximations | 87 |
| 3.15 | RMS deviations for 50 ns MD simulations | 89 |
| 3.16 | RMS fluctuation in residue positions for a 50 ns simulation of 2TRX | 90 |
| 3.17 | Distribution of $\chi_1$ angles | 92 |
| 3.18 | DNA flexibility | 92 |
| 3.19 | Simulation of nucleosome tail collapse | 93 |
| 3.20 | HCP-GB simulation of a 348 000 atom chromatin fibre | 94 |
| 3.21 | Constant energy MD simulation | 97 |
| 3.22 | Comparison of methods for computing component effective Born radii | 99 |
| 3.23 | ALPB model parameters | 106 |
| 3.24 | Acceleration of the electrostatic potential computations | 110 |
| 3.25 | Mapping of HCP onto the GPU | 117 |
| 3.26 | GPU-NAB Speedup | 122 |
| 3.27 | GPU-NAB Scalability | 125 |
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Numerical errors due to catastrophic cancellation</td>
<td>36</td>
</tr>
<tr>
<td>3.1</td>
<td>List of representative structures used for testing</td>
<td>81</td>
</tr>
<tr>
<td>3.2</td>
<td>Effect of Born radii approximaton on relative RMS error</td>
<td>88</td>
</tr>
<tr>
<td>3.3</td>
<td>RMS deviations for 50 ns MD simulations</td>
<td>88</td>
</tr>
<tr>
<td>3.4</td>
<td>Detailed characteristics of simulation dynamics</td>
<td>91</td>
</tr>
<tr>
<td>3.5</td>
<td>Center-of-mass motion due to violation of Newton’s third law for the HCP-GB</td>
<td>95</td>
</tr>
<tr>
<td>3.6</td>
<td>Effect of smoothing function on energy conservation</td>
<td>98</td>
</tr>
<tr>
<td>3.7</td>
<td>Summary of HCP GPU test results</td>
<td>109</td>
</tr>
<tr>
<td>3.8</td>
<td>Overview of GPU Architectures</td>
<td>119</td>
</tr>
<tr>
<td>3.9</td>
<td>Characteristics of Structures.</td>
<td>120</td>
</tr>
<tr>
<td>3.10</td>
<td>Number of Divergent Branches and Global Memory Transactions</td>
<td>123</td>
</tr>
<tr>
<td>3.11</td>
<td>Memory Usage</td>
<td>124</td>
</tr>
</tbody>
</table>
Organization of the thesis

The two main goals of the research described here, were to (1) speed up the computation of the average charge state of ionizable sites in biomolecules, and (2) speed up the computation of long range electrostatic interactions for molecular dynamics. Achieving each of these goals involved several independent research projects. This thesis is organized accordingly, by research project within each of the two broader research topics. Chapter 1 provides a more detailed abstract for each of the research projects within each of the two main research areas. Chapters 2 and 3 present the details for each of the research projects. Within these chapters, each research project is discussed in a separate section, each of which in turn consist of a separate subsection for introduction, methods, test setup, results and discussion, and conclusions. Chapter 4 provides overall conclusions from the entire research work.
Chapter 1

Extended abstract

Atomistic molecular dynamics (MD) simulations can be used to study biomolecules where experimental investigation is expensive or infeasible. The two main stages in the MD simulation pipeline are structure preparation and running the simulation. The accuracy and duration of MD simulations are limited due to computational bottlenecks in each of these stages. The computational bottleneck in the structure preparation stage is the computation of the average charge state of ionizable sites in biomolecules, which scales exponentially as $O(N^2 2^N)$ in the number of ionizable sites. The computational bottleneck in the simulation stage is the computation of long range electrostatic interactions which scales as $O(n^2)$ in the number of atoms in the system. Due to the large number of simulation steps required for most biologically relevant timescales, $10^{12–16}$ steps for microsecond to second timescales – the relatively small $O(n^2)$ cost becomes computationally prohibitive. Thus limiting useful MD simulations to the study of very small systems or activities that occur at very short timescales.

The direct interaction approximation is an $O(N^2)$ approximation for speeding up the computation of average charge state of ionizable sites in biomolecules. The hierarchical charge partitioning approximation is an $O(n \log n)$ approximation for speeding up the computation of long range electrostatic interactions in molecular dynamics. Fully developing, testing and implementing each of these methods involved a number of independent research projects. Following is a more detailed abstract for each of these research projects.
1.1 Speeding up the computation of average charge state

A number of approximation methods exist for speeding up the computation of average charge state. One class of such methods are the clustering algorithms. A rigorous error analysis of two simple clustering algorithms (Sec. 1.1.1), indicated that direct interactions (interactions involving the site of interest) were more important to the accuracy of the computation than indirect interactions (interactions not involving the site of interest). This suggested a novel approximation method, the direct interaction approximation (DIA) (Sec. 1.1.2). The initial implementation of the DIA using Newton’s identity, a generating function for computing combinatorial sums of products, resulted in large numerical errors due to catastrophic cancellations. An alternate binary split-merge algorithm was developed to eliminate catastrophic cancellations (Sec. 1.1.3).

1.1.1 Error bound for simple clustering algorithms

The most basic clustering algorithms first subdivide the system into a set of smaller subsets (clusters). Then, interactions between particles within each cluster are treated exactly, while all interactions between different clusters are ignored. These smaller clusters have far fewer microstates, making the summation over these microstates tractable. These algorithms have been previously used for biomolecular computations, but remain relatively unexplored in the context of biomolecular electrostatics. Presented in Sec. 2.2, is a theoretical analysis of the error and computational complexity for the two most basic clustering algorithms – local clustering and global clustering – that were previously applied in the context of biomolecular electrostatics. Local clustering uses only direct interactions for building clusters, whereas global clustering uses all interactions. We derive a tight, computationally inexpensive, error bound for the equilibrium state of a particle computed via these clustering algorithms. For some practical applications, it is the root mean square error, which can be significantly lower than the error bound, that may be more important. We show that there is a strong empirical relationship between error bound and root mean square error, suggesting that the error bound could be used as a computationally inexpensive metric for predicting the accuracy of clustering algorithms for practical applications. An example of error analysis for such an application – computation of average charge state of ionizable amino acids in proteins – is given, demonstrating that the clustering algorithm can be accurate enough for practical purposes. The empirical analysis also showed that local clustering is in general more accurate than global clustering.

This work has been published in the Journal of Computational Biology,\(^1\) and can be accessed from: http://www.liebertonline.com/doi/abs/10.1089/cmb.2007.0144.
1.1.2 Direct interaction approximation (DIA)

The analysis of the simple clustering algorithms, described above, suggested that direct interactions may be more important for the accuracy of statistical sums (partition function) calculated by the clustering algorithms. Thus, to approximate statistical sums, the DIA computes the contribution of direct interactions \textit{exactly}, while using an \textit{average} value for indirect interaction (Sec. 2.3). Even with this approximation the computation of the partition function involves the sum over $2^N$ microstates, where $N$ is the number of sites. With this approximation, the exact contribution of direct interactions to the partition function can be represented as a combinatorial sum of products known as elementary symmetric functions. Although the number of terms in the elementary symmetric function grows exponentially with $N$, it can be computed in $O(N^2)$ operations using the recursive generating function, Newton’s identity. The computation of Newton’s identity can however result in large numerical errors due to catastrophic cancellations. An alternative formulation of Newton’s identity was developed to reduce these cancellation errors. However, when applied to the computation of ionization states in proteins, the errors were still too large for all but very small systems ($<20$ sites).

1.1.3 DIA using a binary split-merge algorithm

An alternate algorithm was developed to eliminate the catastrophic cancellations inherent in Newton’s identity (Sec. 2.4). The algorithm is similar to the classic merge sort algorithm. The sites are partitioned into a hierarchical binary tree with the root node containing all the sites, and each of the leaf nodes containing a single site. The algorithm then starts with the leaf nodes and proceeds up the binary tree by merging the results of computations between the two branches at each node. Similar to Newton’s identity, the binary split-merge algorithm also scales as $O(N^2)$. To test its accuracy, the DIA was applied to two problems. The computation of statistical (thermal) average magnetization, internal energy, and heat capacity for the 2-D Ising model of ferromagnetism, and for the computation of average charge states of ionizable sites in biomolecules. The accuracy, as measured by RMS error relative to the exact computation, for the DIA was compared to that of the basic Monte Carlo method. For the same computational cost, the deterministic DIA was on average more accurate than the non-deterministic Monte Carlo method for the computation of average magnetization and heat capacity, while being comparable in accuracy for the computation of internal energy and average charge state. These results suggest that the DIA may be a practical alternative to the Monte Carlo method for some problems.
1.2 Speeding up the computation of long range electrostatic interactions

Long range electrostatic interactions drop off slowly, as $1/r$, where $r$ is the distance between charges. Thus, completely ignoring distant interactions, as the spherical cutoff method (a simple approximation used for MD) does, can result in large errors. The more accurate methods use approximations for these long range interactions. However, the most commonly used approximation method for biomolecular MD – the particle mesh Ewald (PME) – cannot be easily extended to implicit solvent models. We propose an alternate approximation, hierarchical charge partitioning (HCP), which is described below. Its development consisted of the following four projects. The HCP was first developed and tested for the simple distant-dependent-dielectric implicit solvent model (Sec. 1.2.1). It was then extended to the industry standard generalized Born implicit solvent model (Sec. 1.2.2). To achieve further speedup, the HCP was implemented on graphical processing units (GPUs), first for the computation of surface potential (Sec. 1.2.3), and then for molecular dynamics (Sec. 1.2.4).

1.2.1 Hierarchical charge partitioning (HCP)

The HCP approximation is based on multiple levels of natural partitioning of biomolecular structures into a hierarchical set of its constituent structural components (Sec. 3.2). The charge distribution in each component is systematically approximated by a small number of point charges, which, for the highest level component, are much fewer than the number of atoms in the component. For short distances from the point of interest, the HCP uses the full set of atomic charges available. For long distances from the point of interest, the approximate charge distributions with smaller sets of charges are used instead. For a structure consisting of $n$ charges, the computational cost of computing the pairwise interactions via the HCP scales as $O(n \log n)$, under assumptions about the structural organization of biomolecular structures generally consistent with reality. A proof-of-concept implementation of the HCP shows that for large structures it can lead to speed-up factors of up to several orders of magnitude relative to the exact pairwise $O(n^2)$ all-atom computation used as a reference. For structures with more than 2-3 thousand atoms the relative accuracy of the HCP (relative root-mean-square force error per atom), approaches the accuracy of the particle mesh Ewald (PME) method with parameter settings typical for biomolecular simulations. When averaged over a set of 600 representative biomolecular structures, the relative accuracies of the two methods are roughly equal. The HCP is however more accurate than the spherical cutoff method. The HCP has been implemented in the freely available NAB molecular dynamics package in AmberTools. A 10 ns simulation of a small protein, using the distance-dependant-dielectric implicit-solvent model, indicates that the HCP simulation is stable. A critical benefit of the HCP approximation is that it is algorithmically very simple, and unlike the PME, the HCP is straightforward to use with implicit solvent models.

This work has been published in the *Journal of Computational Chemistry* and can be accessed from: [http://onlinelibrary.wiley.com/doi/10.1002/jcc.21357/abstract](http://onlinelibrary.wiley.com/doi/10.1002/jcc.21357/abstract).
1.2.2 HCP for the implicit solvent generalized Born model

Molecular dynamics (MD) simulations based on the generalized Born (GB) model of implicit solvation offer a number of important advantages over the traditional explicit solvent based simulations. Yet, in MD simulations, the GB model has not been able to reach its full potential partly due to its computational cost, which scales as $\sim n^2$, where $n$ is the number of solute atoms. Presented in Sec. 3.3 is an $\sim n \log n$ approximation for the generalized Born (GB) implicit solvent model. The approximation is based on the hierarchical charge partitioning (HCP) method (Anandakrishnan and Onufriev, J. Comput. Chem. 2010, 31, 691-706) previously developed and tested for electrostatic computations in gas-phase and distant dependent dielectric models. To apply the HCP concept to the GB model, we define the equivalent of the effective Born radius for components. The component effective Born radius is then used in GB computations for points that are distant from the component. This HCP approximation for GB (HCP-GB) is implemented in the open source MD software, NAB in AmberTools, and tested on a set of representative biomolecular structures ranging in size from 632 atoms to $\sim$ 3 million atoms. For this set of test structures the HCP-GB method is 1.1 - 390 times faster than the GB computation without additional approximations (the reference GB computation), depending on the size of the structure. Similar to the spherical cutoff method with GB (cutoff-GB) which also scales as $\sim n \log n$, the HCP-GB is relatively simple. However, for the structures considered here, we show that the HCP-GB method is more accurate than the cutoff-GB method as measured by relative RMS error in electrostatic force relative to the reference (no cutoff) GB computation. MD simulations of four biomolecular structures on 50 ns time-scales show that the backbone RMS deviation for the HCP-GB method is in reasonable agreement with the reference GB simulation. A critical difference between the cutoff-GB and HCP-GB methods is that the cutoff-GB method completely ignores interactions due to atoms beyond the cutoff distance, whereas the HCP-GB method uses an approximation for interactions due to distant atoms. Our testing suggests that completely ignoring distant interactions, as the cutoff-GB does, can lead to qualitatively incorrect results. In general, we found that the HCP-GB method reproduces key characteristics of dynamics, such as residue fluctuation, $\chi_1/\chi_2$ flips, and DNA flexibility, more accurately than the cutoff-GB method. As a practical demonstration, the HCP-GB simulation of a 348 000 atom chromatin fibre was used to refine the starting structure. Our findings suggest that the HCP-GB method is preferable to the cutoff-GB method for molecular dynamics based on pairwise implicit solvent GB models.

This work has been published in the Journal of Chemical Theory and Computation$^3$ and can be accessed from: http://pubs.acs.org/action/showCitFormats?doi=10.1021\%2Fct100390b.

1.2.3 HCP implementation on GPUs for computing electrostatic surface potential

Tools that compute and visualize biomolecular electrostatic surface potential have been used extensively for studying biomolecular function. However, determining the surface potential for large biomolecules on a typical desktop computer can take days or longer using currently
available tools and methods. Two commonly used techniques to speed up these types of electrostatic computations are, approximations based on multi-scale coarse-graining, and parallelization across multiple processors. The study in Sec. 3.4 demonstrates that for the computation of electrostatic surface potential, these two techniques can be combined to deliver significantly greater speed-up than either one separately, something that is in general not always possible. Specifically, the electrostatic potential computation, using an analytical linearized Poisson Boltzmann (ALPB) method, is approximated using the hierarchical charge partitioning (HCP) multiscale method, and parallelized on an ATI Radeon 4870 graphical processing unit (GPU). The implementation delivers a combined 934-fold speed-up for a 476,040 atom viral capsid, compared to an equivalent non-parallel implementation on an Intel E6550 CPU without the approximation. This speed-up is significantly greater than the 42-fold speed-up for the HCP approximation alone or the 182-fold speed-up for the GPU alone.

This work has been published in the *Journal of Molecular Graphics and Modeling* and can be accessed from: [http://www.sciencedirect.com/science/article/pii/S1093326310000537](http://www.sciencedirect.com/science/article/pii/S1093326310000537).

### 1.2.4 HCP implementation on GPUs for molecular dynamics

Efficient parallelization on the GPU requires highly synchronized computation across processors. On the other hand, certain approximation algorithms, such as multiscale methods, can result in highly asynchronous processing requirements due to the divergent branching inherent in such algorithms. One may expect that implementing such asynchronous algorithms on the GPU would result in an overall loss of performance, and that the total speedup obtained would be less than the product of speedups for the two techniques separately, i.e. less than multiplicative speedup. To test this expectation, we implemented the hierarchical charge partitioning (HCP) multiscale method on the NVIDIA GPU platform. The code was implemented using NAB, the molecular dynamics module in the open source AmberTools, and tested using the distance-depend-dielectric implicit-solvent model. The study in Sec. 3.5 demonstrates that, for the multiscale approximation and simulation model considered here, the degradation in performance due to asynchronous processing is mostly offset by a reduction in other asynchronous operations, and a reduction in slow global memory accesses. As a result we are able to realize near multiplicative speedups. For a 475,000-atom virus capsid we were able to achieve a 11,071-fold speedup, only slightly less than the possible 11,706-fold speedup – 48.0-fold from the parallelization on the GPU and 243.9-fold from the multiscale approximation. Speedup is computed relative to the molecular dynamics code, without the multiscale approximation, running on a single CPU. The overall speedup depends on structure size, with smaller structures having lower speedups. An additional benefit of the HCP implementation on the GPU is the reduced memory requirement, which allows the processing of much larger structures than would otherwise be impossible on the limited memory GPU platform.
Chapter 2

Speeding up the computation of average charge state

2.1 Introduction

In statistical mechanics, the probability of a canonical ensemble of particles\(^1\) being in microstate \(X = \{x_1, x_2, ..., x_n\}\) where particle \(k\) is in state \(x_k\), is given by the Boltzmann average,

\[
P_X = \frac{e^{-E_X/k_BT}}{Z} = \frac{e^{-E_X/k_BT}}{\sum_X e^{-E_X/k_BT}}
\]

where \(k_B\) is the Boltzmann constant, \(T\) is the temperature, \(E_X\) is the energy of microstate \(X\), and the sum in the denominator, commonly referred to as the partition function \(Z\), is the summation over all accessible microstates \(X\).\(^5\) This statistical formulation is used in a diverse class of problems from quantum mechanics to astrophysics.\(^6\)

For applications where energy is a function of non-uniform interactions between particles, such as the 3D Ising model for magnetic phase transition, the calculation of the statistical sum in equation (2.1) has been shown to be computationally intractable, i.e. NP complete.\(^7-10\)

Other examples of systems involving non-uniform interactions include neural networks,\(^11\) plasma effects of the solar interior,\(^6\) forces that drive protein folding,\(^12\) and the computation of average ionization in biomolecules.\(^13\)

The Boltzmann average, equation (2.1), can be used to find the average value of macroscopic properties of the system. For example, average energy \(\langle E \rangle = \sum X E_X P_X\), and entropy \(S = k_B \sum X P_X \ln(P_X)\). The accuracy of an algorithm that approximates the sums involved in the calculation of these properties will depend on the specific mathematical form of the property being calculated. We have chosen to focus our analysis on the calculation of ‘average state

\(^1\)A canonical ensemble of particles is a thermodynamically large system in constant thermal contact with the environment with volume and number of particles being fixed.
of a particle’ due to its practical application in a number of fields, including biomolecular electrostatics.

The average state of particle \( k \), \( \langle x_k \rangle \), can be computed as the weighted sum of the Boltzmann average, equation (2.1), over all microstates of the system, \( X = \{ x_1, x_2, ..., x_n \} \),

\[
\langle x_k \rangle = \sum_X x_k P_X = \frac{\sum_X x_k e^{-E_X/k_BT}}{\sum_X e^{-E_X/k_BT}}
\]  

(2.2)

For a system of \( n \) particles that interact with each other via a pairwise potential \( w_{ij} \), the energy of microstate \( X \) is given by \( E_X = k_BT \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} x_i x_j \), where \( w_{ij} \) is the physical interaction potential between particles \( i \) and \( j \) divided by \( 2k_BT \) (to make \( w_{ij} \) dimensionless and to avoid double counting), \( w_{ii} \) is the physical intrinsic energy of particle \( i \), and \( x_i, x_j \) are the states of particles \( i \) and \( j \) in the microstate represented by \( X \). The meaning of particle, state, and interaction, varies by the type of problem\(^2\).

Representing the interaction potential, \( w_{ij} \), as a matrix \( W \), where \( w_{ij} \) are elements of the matrix, the average state of a particle, equation (2.2), can be expressed as

\[
\langle x_k \rangle = \frac{\sum_X x_k e^{-XW^T X}}{\sum_X e^{-XW^T X}}
\]  

(2.3)

For the purpose of this analysis we assume that there are only two possible states for each particle. This is true for many problems. For example, in the electrostatics of biomolecules, each so called ionizable amino acid can have absolute charge of 0 (neutral) or 1 (charged), and in the 3D Ising model, each particle can have a spin of \( \pm \frac{1}{2} \). Without loss of generality we also assume that the values of the two possible states are 0 and 1, because any two state problem can be mapped to an equivalent problem with states \( \{0,1\}\)^3.

The computation of the statistical sums in equation (2.3) grows exponentially with system size \( n \), requiring \( O(n^22^n) \) operations: \( O(n^2) \) operations to compute the energy for each of

---

\(^2\)For example, in the problem of assigning states of ionization to ionizable amino acids in proteins, considered in detail later in section 2.2.4.3, particles represent individual ionizable amino acids, states the absolute charge of individual amino acids, and interactions the electrostatic interactions between these amino acids. Whereas, in the 3D Ising model particles may represent impurities localized at the vertices of a 3 dimensional lattice representing a magnetic alloy, states the magnetic spins of the impurities, and interactions the magnetic coupling constants between impurities.

\(^3\)For any two state problem with two possible states \( x_i = \{a, b\} \), state vector \( X \), and interaction matrix \( W \), the energy of the system \( E_X = X^T W X \). This is equivalent to the two state problem with \( \hat{x}_i = \{0, 1\} \) and \( E_X = \hat{X}^T \hat{W} \hat{X} + C \), where \( \hat{x}_i = (x_i - a)/(b - a) \), \( \hat{w}_{ij} = w_{ij}(b - a)^2 \forall i \neq j \), \( \hat{w}_{ii} = (b - a)^2 w_{ii} + a(b - a) \sum_j w_{ij} \).
the $2^n$ possible microstates. This makes the computation intractable for many practical biomolecular problems which often involve system sizes of $n > 50$.\textsuperscript{14}

To reduce the computational complexity involved, a number of approximation algorithms have been employed: the Monte Carlo method,\textsuperscript{15–17} mean field approximation,\textsuperscript{18} genetic algorithm,\textsuperscript{19} dead end elimination,\textsuperscript{20} reduced site approximation,\textsuperscript{14} and clustering of strongly interacting particles.\textsuperscript{18,21–27} Clustering algorithms have been shown empirically to provide efficient approximations to statistical sums in large systems where other accepted methods, such as Monte Carlo, may prove too slow for practical use, especially in bioinformatics applications where high throughput is critical.\textsuperscript{23} However, despite demonstrated promise, use of these algorithms in biomolecular applications, and potentially in other problems of similar complexity, is currently impeded by what appears to be a lack of rigorous analysis of the accuracy of the approach (and to a lesser extent, its computational complexity).

The original goal of this work was to conduct a rigorous error analysis of two basic clustering algorithms (Sec. 2.2). One of the results of the error analysis indicated that, for the types of problems studied, direct interactions may be more important than indirect interactions, to the accuracy of approximations of the statistical sums in Eq. 2.3. The direct interaction approximation was developed later, based on this insight (Sec. 2.3 and 2.4).

\[ C = a^2 \sum_i \sum_j w_{ij} \]  

The constant term $C$ cancels out in the computation of average state of a particle, therefore, any two-state average state problem can be solved by an equivalent problem with state values of 0 and 1.
2.2 Error bound for simple clustering algorithms

2.2.1 Introduction

While a number of different clustering methods exist,\textsuperscript{28} for the purpose of this analysis, we focus on the most basic algorithms which have been applied to the problem of computing statistical sums in biomolecular electrostatics. To the best of our knowledge the two such algorithms are the so-called \textit{global clustering} and \textit{local clustering} algorithms.\textsuperscript{23} While other, more sophisticated versions of these approximations have been explored,\textsuperscript{22,24} we believe that the minimal versions are best suited for our goal of understanding the fundamental limitations of the algorithms.

The main objective of this study is to conduct a theoretical analysis of the error inherent in the global and local clustering algorithms, when applied to the computation of average state of a particle.

The rest of this section is organized as follows. First, we describe the two algorithms considered here, and examine their computational complexity. Next, we derive a tight, computationally inexpensive, theoretical error bound. Using a large number of model systems, in which particle-particle interactions mimic interactions found in a broad range of realistic systems, we analyze the distribution of errors introduced by clustering. The statistical analysis shows that, although individual errors may be as large as the error bound, the root mean square error is often significantly lower. The analysis also shows that, for the types of problems studied here, the error bound itself may be used as an empirical metric for estimating the root mean square error. Finally, we use a practical problem in biomolecular electrostatics – determining the charge state of ionizable amino acids in proteins – to show how the concepts developed above may be applied.

2.2.2 Methods

2.2.2.1 Basic clustering algorithms

For a system of non-interacting particles, many properties for a given particle can be trivially computed by considering the particle separately and ignoring all other particles. Similarly, if particles in a subset (cluster) only interact with each other, and not with any of the particles outside the cluster, properties of the particles in the cluster do not depend on the particles outside the cluster. The following subsection shows how this intuitive expectation can be translated into a useful algorithm for the computation of the average state of a particle.

The idea behind clustering is to identify clusters of particles that most closely approximate such subsets of non-interacting particles. Considering only the particles and interactions within the cluster, and ignoring all other interactions, may then be sufficient to determine some equilibrium properties with reasonable accuracy.\textsuperscript{29} The two clustering algorithms considered in this paper, are described in the second subsection below. Computing the statistical sums using these smaller clusters, instead of the entire set of particles, then becomes
computationally feasible, as shown by the analysis of computational complexity in the last subsection below.

### 2.2.2.1.1 Theoretical basis for approximating statistical sums by clustering

Consider a limiting case where an \( n \times n \) interaction matrix \( W \) can be decomposed into two disconnected submatrices \( W_c \) and \( W_d \). An example of such a block diagonal matrix is shown in figure 2.1(a), where there is no interactions between the clusters, i.e. \( w_{ij} = 0 \ \forall \ i \in \text{cluster } C, \ j \in \text{cluster } D \). Since the two submatrices are disconnected, the energy of microstate \( X \) can be represented as a sum of the energies of the two submatrices,

\[
E_X = X_c W_c X_c^T + X_d W_d X_d^T
\]

where the subscripts \( c \) and \( d \) represent the two disconnected clusters or submatrices of lower dimensions \( c \) and \( d \), with \( c + d = n \). Furthermore, each of the possible \( 2^n \) microstates for \( X \) is a combinatorial product of each of the \( 2^c \) possible microstates for cluster \( C \) with each of the \( 2^d \) possible microstates for cluster \( D \). Therefore, the exponential terms in the statistical sum can be expressed as the product of sums over all possible microstates for each of the clusters

\[
\sum_X e^{-X W X^T} = \sum_{X_c} e^{-X_c W_c X_c^T} \sum_{X_d} e^{-X_d W_d X_d^T}
\]

The average state of particle \( k \in C \) can then be expressed as

\[
\langle x_k \rangle = \frac{\sum_{X_c} x_k e^{-X_c W_c X_c^T} \sum_{X_d} e^{-X_d W_d X_d^T}}{\sum_{X_c} e^{-X_c W_c X_c^T} \sum_{X_d} e^{-X_d W_d X_d^T}}
\]

The above equation shows that when interactions between particles belonging to separate clusters is strictly zero, the average state of particle \( k \) in cluster \( C \) is independent of the interactions within the other cluster, \( D \). Therefore, the average state of particle \( k \) can be computed exactly using summation over the reduced set of microstates of cluster \( C \) only.

The global clustering algorithm, described in detail below, attempts to approximate such a grouping of particles in a realistic case of non-zero cross-cluster interactions. It proceeds by constructing cluster in such a way as to limit all cross-cluster interactions to a threshold value \( h \), i.e. \( w_{ij} \leq h \ \forall \ i \in \text{cluster } C, \ j \notin \text{cluster } C \). Subsequently neglecting these cross-cluster interactions constitutes the approximation made by the algorithm. Local clustering, also described in detail below, uses a slightly simpler algorithm that limits the strength of ‘direct’ interactions with the particle of interest \( k \) to a threshold value \( h \), i.e. \( w_{kj} \leq h \ \forall \ j \notin \text{cluster } C \). Direct interactions are the terms highlighted in figure 2.1(b).

The accuracy of the clustering algorithms depend on how closely the clusters constructed by the algorithms approximate the disconnected submatrices described above. Consider for example, an extreme case where the interaction energies are all strong, and evenly distributed.
Figure 2.1: Illustrative five particle system. Particles \{a, b, c, d, e\} are separated into two clusters. Cluster C with particles \{a, b, c\} and cluster D with particles \{d, e\}. (a) Block diagonal matrix can be used to approximate a realistic interaction matrix if cross-cluster interactions are much smaller than intra-cluster interactions. Once this approximation is made, the average state of each particle within a cluster then depends only on the interactions between particles within the cluster. (b) Direct interactions for particle \(a\) are identified by the shaded area. Local clustering uses the relative strength of direct interactions to identify clusters of strongly interacting particles.

Any partitioning of such a system into clusters will result in high cross cluster interactions. Ignoring these strong interactions, as the clustering algorithms do, will result in large errors.

2.2.2.1.2 Definition of the basic clustering algorithms: global and local clustering

Two basic clustering algorithms were recently considered by Myers et. al., and termed global and local clustering. As illustrated in figure 2.2, both algorithms first identify a subset of strongly interacting particles which includes the particle of interest, i.e. the particle for which the average state value is being calculated. The average state of a particle is then calculated using only particles in the cluster, ignoring all other particles and interactions outside the cluster. The difference between the global and local algorithms is in the criteria used to construct the clusters.

Global clustering defines the clusters such that all pairwise interactions between particles from two different clusters is less than a threshold value, determined by the desired cluster size. The following steps identify the global clusters:

1. Initially place each of the particles in a separate cluster of its own (cluster size = 1)
2. Sort the interaction energies between particles, \(w_{ij}\), in descending order
3. Repeat the following steps for each \(w_{ij}\), starting from the largest to the smallest:
(a) If particles $i$ and $j$, for a given $w_{ij}$, are in two different clusters, combine particles from both clusters into a single cluster

(b) Stop when the largest cluster is greater than or equal to the predetermined maximum cluster size $c$

4. If the largest cluster size is greater than the maximum cluster size, $c$, then revert to the set of clusters from the previous iteration of the step above\(^4\).

This process results in clusters where all cross-cluster interactions are less than the last value of $w_{ij}$ examined before the maximum cluster size is reached. This is the threshold value, $h$, $w_{ij} < h$, mentioned above.

**Local clustering** defines the clusters such that all pairwise interactions between the particle of interest and all other particles in the cluster (direct interactions) are greater than a threshold value. The following steps identify a local cluster:

1. Place particle of interest $k$ into its local cluster (cluster size = 1)
2. Sort the direct interaction energies, $w_{kj}$, between particle $k$ and all the other particles, $j$, in descending order
3. Repeat the following step for each $w_{kj}$, starting from largest to smallest:
   (a) Add particle $j$ to the local cluster
   (b) Stop when the cluster size is equal to the predetermined maximum cluster size $c$.

The above procedure is repeated for every particle of interest, which typically includes all particles.

As illustrated in figure 2.2 the two algorithms will not necessarily produce the same set of clusters.

**2.2.2.1.3 Computational complexity** Without approximations the exact computation of average state of a particle scales as $O(n^2 2^n)$, equation (2.3), whereas the clustering methods are considerably faster, and scale as $O(n)$ for a fixed cluster size $c < n$, as shown by the following analysis.

Computation of the statistical sums using clustering algorithms consists of two steps. The first step is to identify clusters with a maximum cluster size of $c$, as described above. The second is to compute the statistical sum over all microstates in each of the clusters. Consider the first step. For global clustering, the computational bottleneck is identifying the clusters, which involves sorting the $n^2$ interaction terms. With an efficient sorting algorithm, such as heapsort,\(^30\) this step requires $O(n^2 \log(n^2))$ operations. For local clustering, the longest

\(^4\)Global clusters are built by combining smaller clusters, therefore the cluster size will not necessarily grow in increments of 1.
Figure 2.2: Global vs. local clustering. A set of five particles, \( \{a, b, c, d, e\} \) are shown with strong interactions between them shown as bold lines, weaker interactions shown as plain lines, and very weak interactions shown as dotted lines. The clusters resulting from the two clustering algorithms, using a maximum cluster size of three, are shown here. **Left:** Global clustering separates the particles into distinct subsets consisting of \( \{a, b, c\} \) and \( \{d, e\} \) since the interactions \( a-b, b-c, \) and \( d-e \) are the strongest. **Right:** For computing the average state of particle \( a \), local clustering uses the subset consisting of \( \{a, b, e\} \) since \( b \) and \( e \) have the strongest interaction with \( a \). The local cluster for \( b \) on the other hand would consist of \( \{a, b, c\} \) and overlaps the cluster for \( a \).

process in the first step involves sorting the \( n \) direct interaction terms, once for each of the \( n \) particle. This requires \( O(n(n \log(n))) \) operations. Now consider the second step. From equation (2.8), computation of the statistical sum for a cluster consisting of \( c \) particles, involves \( O(c^2) \) operations to calculate the energy, \( XWXT \), for each of \( 2^c \) possible microstates. Therefore, the second step, computation of the average state for each of the \( n \) particles after the clusters have been defined, requires \( O(nc^22^c) \) operations.

Assuming a maximum cluster size \( c = 20 \) and system size \( n < 10^5 \), the second step takes much longer than the first \( (n \log(n) < 10^6 << c^22^c \approx 10^8) \), therefore the cost of the first step can be ignored. Considering just the second step with a fixed cluster size \( c \), computation time for the clustering algorithms scale as \( O(n) \).

Comparison to other algorithms for computing statistical averages, is outside the scope of this paper. We note however that the Monte Carlo algorithm, used as a reference in the following error analysis, scales as \( O(n^2) \).\(^{31}\) In a recent study of a biomolecular electrostatics application, clustering was found to be 1 to 2 order of magnitude faster than the Monte Carlo algorithm.\(^{23}\)
2.2.2.2 Theoretical error bound

Usefulness of the clustering algorithms for any application depends on the magnitude of the associated errors. The theoretical error bound derived here places an upper limit on this error. This error bound is also a tight bound, that is, some errors can be as large as the bound.

To define a theoretical error bound for the clustering algorithms introduced above, equation (2.4) for the average state of a particle, is first restated in terms of $\delta$, where $\delta$ is the ratio of the statistical sum when particle $k$ is in state 1 ($x_k = 1$) and when particle $k$ is in state 0 ($x_k = 0$).

\[
\langle x_k \rangle = \frac{\sum_{X:x_k=1} x_k e^{-XWX^T}}{\sum_{X:x_k=0} e^{-XWX^T} + \sum_{X:x_k=1} e^{-XWX^T}} \quad (2.9)
\]

\[
= \frac{\delta}{(1 + \delta)} \quad (2.10)
\]

where, $\sum_{X:x_k=1}$ and $\sum_{X:x_k=0}$ denote the summation over all microstates where the value of $x_k = 1$ and $x_k = 0$, and

\[
\delta = \frac{\sum_{X:x_k=1} e^{-XWX^T}}{\sum_{X:x_k=0} e^{-XWX^T}}
\]

\[
= \frac{\sum_{X:x_k=0} e^{-\sum_i \sum_j w_{ik}x_i + w_{kj}x_j + w_{ij}x_i x_j}}{\sum_{X:x_k=0} e^{-\sum_i \sum_j w_{ik}x_i x_j}} \quad (2.12)
\]

**Theorem 1.** For both local and global clustering algorithms, an error bound on the average state of each particle is

\[
Err_{\text{max}} = \max(|\frac{\delta_{\text{min}}}{1 + \delta_{\text{min}}} - \frac{\delta_{\text{max}}}{1 + \delta_{\text{max}}}|, |\frac{\delta_{\text{max}}}{1 + \delta_{\text{max}}} - \frac{\delta_{\text{min}}}{1 + \delta_{\text{min}}}|) \quad (2.13)
\]

where

\[
\delta_{\text{min}} = e^{-w_{kk}}e^{-\sum_{j:w_{kj}>0} w_{kj}} \quad (2.14)
\]

\[
\delta_{\text{max}} = e^{-w_{kk}}e^{-\sum_{j:w_{kj}<0} w_{kj}} \quad (2.15)
\]

\[
\delta_{\text{cmin}} = e^{-w_{kk}}e^{-\sum_{j:w_{kj}>0, j \in c} w_{kj}} \quad (2.16)
\]

\[
\delta_{\text{cmax}} = e^{-w_{kk}}e^{-\sum_{j:w_{kj}<0, j \in c} w_{kj}} \quad (2.17)
\]

with $j : w_{kj} > 0$ denoting all positive (repulsive) interactions between particles $k$ and $j$, and $j : w_{kj} < 0$ denoting all negative (attractive) interactions between particles $k$ and $j$. 
Proof. The proof consists of three steps. First, we derive \( \delta^{\text{max}} \) and \( \delta^{\text{min}} \), the upper and lower bounds on \( \delta \), which is defined by equation (2.12), and \( \delta^{\text{max}}_c \) and \( \delta^{\text{min}}_c \), the upper and lower bounds on \( \delta_c \), which is the value of \( \delta \) calculated for the particles in the cluster, of size \( c \), only. Next, we derive upper and lower bounds on the values of \( \langle x_k \rangle \), the exact value of the average state of a particle, and \( \langle x_k \rangle_c \), the average state of the particle calculated using clustering, in terms of the bounds on \( \delta \) and \( \delta_c \). The bounds on the average state of a particle are then used to derive the error bound in equation (2.13).

To derive bounds on \( \delta \), note that for any microstate \( X \), the sum \( \sum_j w_{kj} x_j \) is bounded by the sum of all positive interactions and by the sum of all negative interaction, i.e. \( \sum_j w_{kj} \geq \sum_{j : w_{kj} > 0} w_{kj} \). Substituting this relationship into equation (2.12) we have

\[
\delta \leq \sum_{X : x_k = 0} e^{-w_{kk}} e^{-\sum_{j : w_{kj} < 0} w_{kj}} e^{-\sum_j w_{ij} x_i x_j} \sum_{X : x_k = 0} e^{-\sum_j w_{ij} x_i x_j} \leq e^{-w_{kk}} e^{-\sum_{j : w_{kj} < 0} w_{kj}} = \delta^{\text{max}}
\]

Similarly, one can derive the expressions for \( \delta^{\text{min}} \), \( \delta^{\text{max}}_c \), and \( \delta^{\text{min}}_c \), shown in the theorem statement.

Substituting these bounds into equation (2.10), we derive the bounds for the average state of a particle.

\[
\langle x_k \rangle^{\text{max}} = \frac{\delta^{\text{max}}}{1 + \delta^{\text{max}}} \quad (2.20)
\]
\[
\langle x_k \rangle^{\text{min}} = \frac{\delta^{\text{min}}}{1 + \delta^{\text{min}}} \quad (2.21)
\]
\[
\langle x_k \rangle^{\text{max}}_c = \frac{\delta^{\text{max}}_c}{1 + \delta^{\text{max}}_c} \quad (2.22)
\]
\[
\langle x_k \rangle^{\text{min}}_c = \frac{\delta^{\text{min}}_c}{1 + \delta^{\text{min}}_c} \quad (2.23)
\]

Therefore, the maximum absolute error is

\[
Err^{\text{max}} = \max(\left| \langle x_k \rangle^{\text{min}}_c - \langle x_k \rangle^{\text{max}}_c \right|, \left| \langle x_k \rangle^{\text{max}} - \langle x_k \rangle^{\text{min}} \right|) \quad (2.24)
\]
\[
= \max(\left| \frac{\delta^{\text{min}}}{1 + \delta^{\text{min}}_c} - \frac{\delta^{\text{max}}}{1 + \delta^{\text{max}}_c} \right|, \left| \frac{\delta^{\text{max}}_c}{1 + \delta^{\text{max}}_c} - \frac{\delta^{\text{min}}_c}{1 + \delta^{\text{min}}_c} \right|) \quad (2.25)
\]

Corollary 1.1. The above error bound is tight in the following sense: \( \forall \epsilon > 0, \exists \) a matrix \( W \) such that \( |Err - Err^{\text{max}}| < \epsilon \), where \( Err \) is the error in the calculated value for the average state of a particle, and \( Err^{\text{max}} \) is the error bound as defined by equation (2.13).
The above statement is easily proven by a trivial case, the block diagonal matrix shown in figure 2.1(a), where the interactions between the particles in cluster $C$, \{a, b, c\}, and the particles not in the cluster, \{d, e\}, is 0, i.e. $w_{kj} = 0 \forall k \in C, j \notin C$. For such a matrix $|Err - Err_{max}| = 0 < \epsilon$.

To illustrate the corollary statement using a non-trivial case, consider the example shown in figure 2.3. The figure shows a four particle system \{a, b, c, d\}, with a cluster of size three. Using the interaction energies shown in the figure, the error in the average state of particle $a$, $|Err| = |\langle x_k \rangle_c - \langle x_k \rangle| = |0.0069 - 0.0479| = 0.0410$. The error bound from (2.13), $Err_{max} = 0.0445$, which is only 9% greater than the actual error.

![Figure 2.3: Example of system with $Err \approx Err_{max}$](image)

Figure 2.3: Example of system with $Err \approx Err_{max}$. The system consists of four particles \{a, b, c, d\}, with the diagonal elements $w_{ii}$ for each particle shown next to the particle, and non-zero interactions, $w_{ij}$, between each pair of particles shown next to the link between the particles. For cluster size of 3, the resulting cluster (global and local) for particle $a$ consists of \{a, b, c\}. The average state calculated for particle $a$ using clustering is 0.0069 whereas the exactly calculated value is 0.0479 with an error $|Err| = 0.0410$. From equation (2.13), the error bound, $Err_{max} = 0.0445$, which is only 9% larger than the actual error.

Computation of the error bound in equation (2.13) is also computationally inexpensive, requiring $O(n)$ operations. Therefore it can be easily performed in advance to determine if the results from the clustering algorithms will be accurate enough for the practical application at hand.

### 2.2.3 Test setup

The error bound defined in the previous section by equation (2.13) gives a tight bound, which, in realistic systems may be much larger than the root mean square (RMS) error. And for some practical applications, the RMS error may also be much smaller than the error due to other sources, such as inaccuracies in interaction potentials, $w_{ij}$. For such applications the clustering algorithms may be accurate enough, even if the exact error bound is large. To identify system characteristics that may determine RMS error, errors are analyzed using a large number of model systems.
For the purpose of statistical analysis, the model systems can be classified into different categories based on the type of interaction between particles. Although these interactions can take many forms, two common categories are inverse distance-squared \( w_{ij} \propto \frac{1}{r^2} \), and inverse distance \( w_{ij} \propto \frac{1}{r} \) interactions, where \( r \) is the distance between particles \( i \) and \( j \). Examples of the first category include short range electrostatic interactions in biomolecular systems. Electromagnetic and gravitational interactions in vacuum, and long range electrostatic interactions in biomolecules, are examples of the second category of systems. A large number of model systems were generated for each of these two categories, as described below.

### 2.2.3.1 Description of model systems

Interaction matrices \( W \) are generated by first randomly placing \( n \) particles in a \( 10 \times 10 \times 10 \) cube, i.e. by randomly generating a set of \((x \ y \ z)\) coordinates for each particle. The interaction between these particles is then computed by the function \( w_{ij} = \pm \frac{1}{r_{ij}^\alpha} \ \forall \ i \neq j \) where \( r_{ij} \) is the distance between the particles, and \( \alpha \) is set to 1 and 2, to represent the two different classes of problems. The diagonal terms in \( W \), which represent intrinsic energy of each particle, are also randomly generated as a uniform random distribution in a range \([-b, +b]\). A value of \( b = 5 \) is used to produce model systems with values for average state of a particle, \( \langle x_k \rangle \), that span the entire range from 0 to 1, reasonably uniformly\(^5\).

For the statistical analysis, 1,576 model systems were generated, as described above. The matrix sizes range from \( 5 \times 5 \) to \( 500 \times 500 \), representing model systems of size \( n = 5 \) to 500, and included short range \( (w_{ij} = \pm \frac{1}{r^2}) \) and long range \( (w_{ij} = \pm \frac{1}{r}) \) interactions.

### 2.2.3.2 Baseline for calculating error

Due to long run times, it is not practical to compute the exact value of the statistical sums, equation (2.3), for this many systems, for anything larger than a \( 25 \times 25 \) matrix. Therefore, for larger systems, the root mean square difference between the clustering algorithm, and a reference algorithms, is used as an estimate of root mean square (RMS ) error. RMS error is calculated as \( \sqrt{\frac{\sum_{i=1}^{N} (\langle x_k \rangle_i^c - \langle x_k \rangle_i^{ref})^2}{N}} \), where \( N \) is the number of samples, and \( \langle x_k \rangle_i^c \) and \( \langle x_k \rangle_i^{ref} \) are the \( i^{th} \) sample of the average state of a particle computed by the clustering and the reference algorithms. We use the Monte Carlo method, as implemented by\(^31\) as the reference algorithm. The Monte Carlo method is well established for calculating the average state of a particle\(^22\) and provides a reasonable estimate of RMS error, because error for the stochastic Monte Carlo algorithm is small, and uncorrelated to error in the clustering algorithm, as shown by figure 2.4(a) below, where a comparison with the exact computation is presented.

\(^5\)For values of \( b > 5 \), the values for average state are highly skewed towards 0 or 1 with very few values between 0.1 and 0.9. For values of \( b < 5 \), the values for average state are highly skewed towards the middle with very few extreme values < 0.1 and > 0.9.
2.2.4 Results and discussion

The following analysis examines RMS error due to the clustering algorithms for the model systems, as a function of system size, cluster size, and error bound. The empirical relationship found between error bound and RMS error, suggests that the error bound may serve as an empirical metric for predicting the RMS error for the clustering algorithm.

2.2.4.1 Accuracy of the basic clustering algorithms

The analysis of the clustering algorithms performed using the model systems with inverse distance-squared interactions, \( w_{ij} = \pm 1/r^2 \), are summarized in Figures 2.4 and 2.5. The analysis shows that (a) in general local clustering is slightly more accurate than global clustering, (b) RMS error increases as the ratio of system size to maximum cluster size increases, and (c) statistically speaking, RMS errors are small, but at the same time, a small fraction of particles can exhibit large errors, approaching the bound.

Figures 2.6 and 2.7 summarize the analysis of the clustering algorithms performed using the model systems with inverse distance interactions, i.e. \( w_{ij} = \pm 1/r \). The analysis shows that (a) local clustering is only very slightly more accurate than global clustering, (b) RMS error increases as the ratio of system size to maximum cluster size increases, and (c) compared to the inverse distance-squared model, errors are larger and a larger number of particles exhibit errors approaching the bound.

The plot of RMS error as a function of the error bound in figure 2.8, shows that RMS error is considerably smaller than the error bound for the types of systems considered here. For many practical applications, the RMS error may be more important than the strict error bound, for example, the results of the algorithm may be acceptable if the additional RMS error it introduces is much less than the statistical error inherent in the experimental data. Figure 2.8 also shows an empirical relationship between error bound and RMS error. The RMS error is less than 0.05 when the error bound is less than 0.8 for local clustering, and less than 0.4 for global clustering; the RMS error increases sharply when the error bound is greater than 0.8 for local clustering and greater than 0.4 for global clustering. This relationship between error bound and RMS error is true for model systems with both short range (inverse distance-squared) and long range (inverse distance) interactions. This suggests the possibility that, for the types of systems considered here, it may be possible to use the error bound as a practical mechanism for predicting the RMS error.

Comparing RMS errors for the two types of model systems in figure 2.8, we also see that, on average clustering is more accurate for systems with short range (inverse distance-squared) interactions compared to systems with long range (inverse distance) interactions. We speculate that clustering is even more accurate for shorter range interactions, i.e. \( w_{ij} = 1/r^\alpha \), \( \alpha > 2 \).
Figure 2.4: RMS error for inverse distance-squared interactions. RMS error as a function of system size using model systems varying in size from (a) $n = 5$ to 25 with cluster size $c = 5$, and (b) $n = 20$ to 500 with cluster size $c = 20$. (c) RMS error as a function of maximum cluster size using model systems of size $n = 25$ and maximum cluster size $c = 5$ to 25. (d) Distribution of error for model systems with size $n = 20$ to 500 with cluster size $c = 20$, showing standard deviation of 0.0684 for local clustering and 0.1192 for global clustering. Values are grouped into 100 bins of size 0.02 for calculating frequency distribution. For (a) and (c) RMS error is the RMS difference between, the average state of a particle computed by the clustering algorithm, and by exact calculation. For (b) RMS error is the RMS difference between, the average state of a particle computed by the clustering algorithm, and by an established Monte Carlo algorithm. Connecting lines in (a), (c), and (d) shown to guide the eye.

### 2.2.4.2 Choice of cluster size

The preceding analysis shows that RMS error increases as a ratio of system size to cluster size, indicating that one should maximize cluster size within available computational capacity. However, for practical purposes the range of values for cluster size is limited to $c < 40$. For example, a system of size $n = 100$ and cluster size $c = 30$ takes approximately 40 CPU hours on a typical gigaflop ($10^9$ floating point operations per second) desktop computer. Even
Figure 2.5: Error bound for inverse distance-squared interactions. Relative location of individual absolute error within the error bound for (a) local and (b) global clustering, as a function of system size. Each point represents the value of $|\text{Err}/\text{Err}^{\text{max}}|$ for each particle in a model system with system sizes $n = 20$ to 500 and maximum cluster size $c = 20$. Although a large percentage of errors are much smaller than the error bound there are cases where error approaches the bound.

with a teraflop ($10^{12}$ floating point operations per second) modern supercomputer cluster, assuming perfect scalability, the cluster size can only be increased from 30 to 40 without increasing the CPU time. This is because the computation time scales as $O(c^{3.2}c^2)$, as shown in section 2.2.2.1.3, which means that increasing $c$ from 30 to 40 increases computation time by more than $10^3$.

The reduction in computation cost from reducing cluster size below 20 is negligible due to the fixed set up cost. Within this range of $c = 20$ to 40, and system sizes much larger than $c$, the improvement in accuracy from increasing cluster size is relatively small compared to the increase in computational cost. For example, consider the standard deviation in error for model systems with inverse distance squared interactions, shown in figure 2.4(d). Increasing cluster size from 20 to 23 reduces the standard deviation from 0.0684 to 0.0668 for local clustering, and from 0.1192 to 0.1173 for global clustering, however, for an order of magnitude increase in computational cost. Given the narrow range of practical values for maximum cluster size, and the small incremental reduction in standard deviation, we believe the results of our analysis using a maximum cluster size of $c = 20$, is qualitatively representative of any practically reasonable calculation of statistical sums based on the clustering algorithms and the physical systems discussed above.

2.2.4.3 A practical application: Computing average charge state of ionizable groups in proteins

Electrostatic properties play a key role in the structure, function, and properties of proteins and other biomolecules. For example, the distribution of charges on a protein are a
Figure 2.6: RMS error for inverse distance interactions. RMS error as a function of system size using model systems varying in size from (a) $n = 5$ to 25 with maximum cluster size $c = 5$, and (b) $n = 20$ to 500 with maximum cluster size $c = 20$. (c) RMS error as a function of maximum cluster size using model systems of size $n = 25$ and maximum cluster size $c = 5$ to 25. (d) Distribution of error for model systems with size $n = 20$ to 500 with cluster size $c = 20$, showing standard deviation of 0.2519 for local clustering and 0.2818 for global clustering. Values are grouped into 100 bins of size 0.02 for calculating frequency distribution. For (a) and (c) RMS error is the RMS difference between, the average state of a particle calculated by the clustering algorithm, and by exact calculation. For (b) RMS error is the RMS difference between, the average state of a particle computed by the clustering algorithm, and by an established Monte Carlo algorithm. Connecting lines in (a), (c), and (d) shown to guide the eye.

key determinant of its binding affinity for specific molecules, and it is by binding to these specific molecules that different proteins perform their different functions.\textsuperscript{33} Knowledge of this charge distribution is therefore essential for modeling these molecular processes,\textsuperscript{34} for example, for drug design, or for understanding how biological organisms function or malfunction. However, experimentally determined charge distributions in proteins are generally not available due to the limitations of current experimental techniques.\textsuperscript{35} Therefore, in practice the charge composition of proteins, represented by the average charge of ionizable groups, is
Figure 2.7: Error bound for inverse distance interactions. Relative location of individual absolute error within the error bound for (a) local and (b) global clustering, as a function of system size. Each point represents the value of $|\frac{\text{Err}}{\text{Err}_{\text{max}}}|$ for each particle for model systems with system size $n = 20$ to 500 and maximum cluster size $c = 20$. Although a large percentage of errors are much smaller than the error bound there are cases where error approaches the bound.

calculated using theoretical methods.$^{13,31,36–41}$

In the following subsections, we first describe the application and show that this problem of finding the average charge of ionizable groups, is the same as the problem of finding the average state of a particle. We then analyze the error bound and RMS error for twelve sample proteins, showing results similar to the model systems with short range (inverse distance-squared) interactions, described earlier. The last subsection below, compares the characteristics of interactions in real protein systems to those of the model systems with inverse distance-squared interactions, to show some of the similarities between the two.

To be consistent with the analysis in section 2.2.4.2, we use a maximum cluster size of $c = 20$.

**2.2.4.3.1 Average charge state of ionizable groups in proteins** Proteins are made up of amino acids (groups) some of which may be charged (ionized) or uncharged depending on whether or not it has a bound proton (protonation state). An example is shown in figure 2.9,$^{42}$ the protein is made up of 36 amino acids, 12 of which are ionizable. The ionizable groups include cations, e.g. lysine, arginine, and anions, e.g. aspartic acid, glutamic acid, that are able to take/release a proton depending on environmental conditions, such as pH (acidity of the environment). Cations are uncharged in their unprotonated state and carry a positive charge when protonated. Anions carry a negative charge in their unprotonated state and are uncharged when protonated. The protonation state of individual ionizable groups, and consequently the electrostatics of the molecule, depend intricately on interactions between all ionizable groups, as well as interactions with the surrounding environment. The energy of the system, which determines the protonation state of individual ionizable groups, is also non-trivially dependent on the energy of interactions between ionizable groups, as


Figure 2.8: RMS error as a function of the error bound. For model systems of size $n = 20$ to 500 with cluster size $c = 20$. Both types of model systems show a sharp increase in RMS error for (a) error bound greater than 0.8 for local clustering and (b) error bound greater than 0.4 for global clustering. Error values are grouped by error bound into 20 bins of size 0.05. RMS error points are plotted at the center of the bin intervals. Connecting lines shown to guide the eye.

Let us define the protonation microstate as $Y = \{y_1, y_2, \ldots, y_n\}$, where $y_i$ is the state of protonation of group $i$, with $y_i = 1$ being the protonated state and $y_i = 0$ the unprotonated state. Then, the energy of microstate $Y$ is given by

$$E_Y = \sum_{i=1}^{n} \left[ y_i k_B T \ln 10 (pH - pK_{intr}^{i}) + \frac{1}{2} \sum_{j=1}^{n} v_{ij} (q_i + y_i)(q_j + y_j) \right]$$

(2.26)

where $k_B$ is the Boltzmann constant, $T$ the temperature, $q_i = \{0, -1\}$ the charge of group $i$ in an unprotonated state, $v_{ij}$ the electrostatic potential between groups, and $pK_{intr}^{i}$ the so called intrinsic $pK_a$ values of each group. $pK_a$ is the $pH$ value at which the group has probability 1/2 of being protonated, and $pK_{intr}^{i}$ is the $pK_a$ of the group when all other groups are held at their formally neutral state.

Finding the average protonation of group $k$ is equivalent to finding the average state of the particle $k$ in equation (2.4), which is shown by mapping equation (2.26) to

$$E_Y = E_X = XWX^T + C$$

(2.27)
Protein backbone and 3 ionizable groups

Figure 2.9: Protein structure for 1VII. Only the backbone structure and 3 of its 12 ionizable groups are shown, along with the interaction potential between these 3 groups. Image created using VMD (Visual Molecular Dynamics) software.43

where

\[ x_i = |y_i + q_i| \quad (0 = \text{uncharged}, 1 = \text{charged}) \quad (2.28) \]

\[
\begin{align*}
\omega_{ij} &= \begin{cases} 
\frac{v_{ij}}{2k_B T} & i \neq j, q_i = q_j \\
-\frac{v_{ij}}{2k_B T} & i \neq j, q_i \neq q_j \\
\ln10(pH - pK_{int}^i) & i = j, q_i = 0 \\
-\ln10(pH - pK_{int}^i) & i = j, q_i = -1 
\end{cases} \\
\end{align*}
\]

\[ C = \sum_i -q_i v_{ij} \quad (2.30) \]

The exponent of the constant, \( C \), cancels out in the computation of average charge of a group \( \langle x_k \rangle \), which can then be stated as

\[ \langle x_k \rangle = \frac{\sum_X x_k e^{-XWx^T}}{\sum_X e^{-XWx^T}} \quad (2.31) \]

which is the same as equation (2.4) for the average state of a particle. Therefore, all of the analysis we have presented above can be directly applied to the problem of finding average charge of ionizable groups in proteins.

2.2.4.3.2 Required accuracy for the calculated value of average charge  
When studying the electrostatic properties of molecules, biochemists typically look at the \( pK_a \)
values of ionizable groups, defined above. The relationship between average protonation and $pK_a$ is given by the Henderson-Hasselbalch equation,\textsuperscript{45}

$$\langle y_k \rangle = 10^{(pK_a - pH)/(1 + 10^{(pK_a - pH)})}$$  \hspace{1cm} (2.32)

When $pK_a = pH$ in the above equation, $\langle y_k \rangle = 0.5$.

On average, the accuracy of calculated $pK_a$ values has been shown to be about $\pm 1$ $pK_a$ unit, due to approximations in the underlying physical model used to compute interactions $w_{ij}$.\textsuperscript{46} Therefore, the additional error introduced by clustering may be acceptable as long as it is much smaller. If we conservatively limit RMS error due to clustering to $\pm 0.05$ $pK_a$ units, it would correspond to an accuracy of $\pm 0.03$ for average protonation $(0.5 - (10^{\pm 0.05}/(1 + 10^{\pm 0.05})))$, which also corresponds to an accuracy of $\pm 0.03$ for average charge, from the mapping in equation (2.28). Therefore, as long as the RMS error introduced by clustering is less than 0.03, it may be acceptable in practice.

\subsection*{2.2.4.3.3 Results of application to real proteins}

The two basic clustering algorithms were tested on twelve sample proteins ranging in size from 20 to 275 ionizable groups (system size). The protein databank identification\textsuperscript{47} for the twelve proteins, with the number of ionizable groups shown in parenthesis, are: 4PTI (20), 2LZT (32), 1BVG (46), 1A23 (59), 1MYF (61), 1CBX (80), 1DS1(111), 1FDL (130), 1E56 (163), 2HHD (174), 2OLB (174), and 1KX3 (275).

Figures 2.10 and 2.11 summarize the results. Similar to the model systems studied earlier, the analysis shows that (a) local clustering is more accurate than global clustering, (b) RMS error increases with system size, and (c) statistically speaking, errors are small, but at the same time, a small fraction of particles can exhibit large errors, approaching the bound.

For the acceptable error of $\pm 0.03$ identified above, we see from figure 2.10 that, for this small sample set of proteins, local clustering may be acceptable for proteins with less than 100 ionizable groups, and global clustering may be acceptable for proteins with less than 50 ionizable groups. This is similar to the results seen in figure 2.4(b) for the model systems with inverse distance squared interactions.

The plot of RMS error as a function of error bound, figure 2.11, shows that RMS error increases sharply when error bound is greater than 0.4 for global clustering and greater than 0.8 for local clustering. This is also similar to the trends seen for the model systems in figure 2.8.

\subsection*{2.2.4.4 Relationship of model systems to real proteins}

The model systems used in the preceding analysis were designed to be representative of realistic problems. Figure 2.12 illustrates the similarity between twelve randomly generated model systems with inverse distance-squared interactions, and the interaction energies for the twelve sample proteins. The interaction potentials for these proteins are computed as described in Myers et. al.\textsuperscript{23} Comparison of model systems and the sample protein interaction matrices, figure 2.12, show similar distributions of interactions between particles.
### 2.2.5 Conclusions

In statistical mechanics, many equilibrium properties of a system can be calculated using the Boltzmann average. The computation of the average state of a particle, for a system of \( n \) particles, involves calculating statistical sums over \( 2^n \) microstates, which is computationally intractable, in general. In the context of biomolecular electrostatics, one of the relatively new methods used to approximate the statistical sums, are the clustering algorithms. The most basic algorithm first partitions the system into smaller subsets of particles (clusters). The statistical sums are then calculated using just the particles and interactions within these smaller clusters, ignoring all cross-cluster interactions. These smaller clusters have far fewer microstates, making the summation over these microstates tractable.

This study analyzes the computational characteristics of the most basic clustering algorithms as applied to physically realistic systems.

The analysis of computational complexity shows that computation time scales as \( O(n) \), for a fixed maximum cluster size \( c \), and increases exponentially with the cluster size. As a result, the maximum cluster size itself is limited to \( c \approx 40 \) for a modern computer. Most of our analysis uses a cluster size of \( c = 20 \), although larger values have also been explored. For system sizes much larger than \( c \), we find that increasing the cluster size above 20 results in only a very modest increase in accuracy, but at high computational costs.

We derive a tight error bound for average state of a particle, as calculated by the clustering approximations. Statistical characteristics of errors due to the clustering approximations were then analyzed on over 1,000 model systems of sizes ranging from 5 to 500. The model systems were designed to represent two of the most common classes of physical interactions, inverse distance-squared (short range) interactions and inverse distance (long range) interactions. The three main findings from the statistical analysis are as follows.

![Figure 2.10: RMS error for sample structures. RMS error as a function of system size. Clustering algorithms use maximum cluster size \( c = 20 \). RMS error is the RMS difference between, average charge computed by the clustering algorithm, and by an established Monte Carlo algorithm. Connecting lines shown to guide the eye.](image-url)
Figure 2.11: RMS error as a function of error bound for sample structures. Figure shows a sharp increase in RMS error for (a) error bound greater than 0.8 for local clustering, and (b) error bound greater than 0.4 for global clustering. Error values are grouped by error bound into 20 bins of size 0.05. RMS error points are plotted at the center of the bin intervals. Connecting lines shown to guide the eye.

One, although the actual error can approach the error bound, in some cases the root mean square (RMS) error can be much smaller. For many applications the RMS error may be more important than the strict error bound. For example, the problem may already have other inherent sources of statistical error, so one only needs to make sure that the error introduced by clustering is much less on average.

Two, there exists a strong empirical relationship between error bound and RMS error. This relationship was found to be true for the model systems considered here. Since the cost of computing the error bound is negligible compared to the cost of running the clustering algorithm, such a metric could be used to determine if the RMS error is acceptable, before incurring the cost of running the algorithm.

Three, clustering is more accurate for systems with short range interactions, in our case those modeled as inverse distance-squared interactions, compared to long range interactions, modeled as inverse distance interactions. In general, we expect the clustering algorithms to be more accurate for shorter range interactions, and less accurate for longer range interactions.

To demonstrate the practical usefulness of the analysis, we explored the errors due to clustering approximation in the context of computing charge states of ionizable groups in proteins—a computational problem very important for biomolecular applications. For this purpose, a set of twelve sample proteins, spanning the range of sizes found in nature, were selected. All of the general conclusions we have made for randomly generated model systems held true. For proteins with less than 100 ionizable groups (system size), application of local clustering for calculating average charge resulted in RMS error of less than 0.03, representing 3% of the allowed range. This level of accuracy for computing average charge may be tolerable in practice, where the error is dominated by other sources, namely the inaccuracies inherent in the computation of charge-charge interactions. Given sufficient level of accuracy, the clustering
Figure 2.12: Comparison of interaction potentials. Twelve randomly generated model systems with inverse distance-squared interactions, were compared to the interaction potential for twelve sample proteins. (a) Distribution of pairwise interaction potential, the non-diagonal terms in the interaction potential matrix, for model systems. Values are grouped into bins of size 0.02 for the purpose of computing distribution. (b) Distribution of pairwise interaction potential for sample proteins, showing a similar distribution. (c) Distribution of the intrinsic energies, the diagonal terms in the interaction potential matrix, for the model systems. Values are grouped into bins of size 10 for the purpose of computing distribution. (d) Distribution of intrinsic energy for the sample proteins.

algorithms are well suited for applications where computation speed is an important consideration, such as the interactive web based system, H++, http://biophysics.cs.vt.edu/H++, for the computation of electrostatic properties of biomolecules.48

In this work, we have focused on a set of a few, most basic computational properties of the clustering algorithms applied to the approximation of statistical sums in physically realistic systems. The expansion of this study to explore finer details of this promising approximation should be considered in the future.

The empirical relationships observed during this analysis are based on the statistical analysis of model systems that represent two of the most common types of interactions. Although
these relationships also hold true for the practical application described above, a theoretical formulation of RMS error would be desirable to understand the limitations of these empirical relationships.

Over the course of the analysis some possible improvements to the clustering algorithms were also identified. To the best of our knowledge, only variations of the basic local and global clustering algorithms have been applied to the computation of these statistical sums in the context of biomolecules. Many, more sophisticated clustering algorithms are available, such as those identified by Jain et. al.,\textsuperscript{28} may produce more accurate results in this context. The basic clustering algorithms as implemented by Myers et. al.\textsuperscript{23} also do not consider intrinsic energies, the diagonal terms in the interaction matrix, in the process of constructing the clusters. Including these terms, on the same footing with non-diagonal terms, may also improve the accuracy of the resulting cluster based approximations.
### 2.3 Direct interaction approximation (DIA)

#### 2.3.1 Introduction

In statistical mechanics, the partition function is defined as

\[
Z = \sum_X e^{-E_X/k_B T} \tag{2.33}
\]

where \(k_B\) is the Boltzmann constant, \(T\) is the temperature, \(E_X\) is the energy of microstate \(X\), and the sum is over all accessible microstates \(X\). The partition function can be used to determine macroscopic properties of systems in equilibrium. For example, internal energy \(U = k_B T^2 \partial \ln Z / \partial T\), and entropy \(S = \partial (k_B T \ln Z) / \partial T\).

For a system with \(N\) particles, where each particle can be in one of two possible states, the number of microstates, \(X\), in Eq. 2.33 grows exponentially as \(O(2^N)\). And, the exact computation of \(Z\) using Eq. 2.33 becomes intractable for \(N \gtrsim 50\).

Presented here is an \(O(N^2)\) approximation – direct interaction approximation (DIA) – for computing the partition function. For a given particle, the DIA approximates the partition function such that the contribution to the partition function due to direct interactions (pairwise interaction involving the given particle) are calculated exactly, while using an average value for the contribution due to indirect interactions (pairwise interaction not involving the given particle). Despite the above approximation, the exact computation of the contribution to the partition function due to direct interaction still involves the sum over \(2^N\) microstates. However, the sum can be computed in \(O(N^2)\) operations for each particle by using a generating function – Newton’s identity.

The DIA was tested using a set of randomly generated systems where interaction potential is proportional to the inverse of the distance between particles, as described earlier in Section 2.2.3.1. For small structure \((N < 10)\), the DIA was more accurate than the commonly used Metropolis Monte Carlo method. However, for larger structures computations using Newton’s identity suffer from large numerical errors due to catastrophic cancellations (loss of precision when computations involve small differences between large numbers). An alternative formulation of the DIA was developed to reduce the numerical error. The alternate formulation does reduce the error, but does not eliminate it.

The rest of this section is organized as follows. First, the Methods section describes the formulation of the DIA and the use of Newton’s identity for exactly computing the contribution of direct interactions to the partition function. Then, we describe the test setup used to evaluate the accuracy of the approximation. Next, the results of the testing are discussed. Finally the findings are summarized in the Conclusions section.

#### 2.3.2 Methods

For a selected site \(k\) direct interactions are pairwise interactions \(w_{ij}\) that involve the selected site, i.e. \(i\) or \(j = k\). Indirect interactions are pairwise interactions \(w_{ij}\) that do not involve
the selected site, i.e. \( i \) and \( j \neq k \). The DIA also assumes that each particle can be in one of two states 0 or 1. As shown earlier in Sec. 2.1, any two state system with states \{a, b\} can be mapped to an equivalent system with states \{0, 1\}.

The derivation of the DIA consists of two steps. First, the partition function \( Z \) in Eq. 2.33 is reformulated to separate out microstates with the same number of indirect interactions, which is approximated by an average value. Then the contribution of direct interaction to the partition function is represented as a combinatorial sum of products known as elementary symmetric functions. Elementary symmetric functions can be computed exactly using a recursive generating function - Newton’s identity.

### 2.3.2.1 Separating out terms in the partition function

The sum over all microstates \( X \) in the partition function \( Z \) (Eq. 2.33) can be subdivided into the sum over two subsets of microstates. One subset in which the state of the selected particle \( k \), \( x_k = 0 \), and another subset in which \( x_k = 1 \), i.e.

\[
Z = \sum_{X : x_k = 0} e^{-E_X/k_B T} + \sum_{X : x_k = 1} e^{-E_X/k_B T} \tag{2.34}
\]

\[
= Z_0 + Z_1 \tag{2.35}
\]

where \( Z_0 \) and \( Z_1 \) are the contributions to the partition function due to microstates with \( x_k = 0 \) and \( x_k = 1 \) respectively.

Each of these subsets of microstates can be further subdivided into \( N - 1 \) subsets each, such that each subset has the same number of particles in state 1, i.e.

\[
Z_0 = \sum_{m=0}^{N-1} Z_0(m) \tag{2.36}
\]

\[
Z_1 = \sum_{m=0}^{N-1} Z_1(m) \tag{2.37}
\]

where \( Z_0(m) \) and \( Z_1(m) \) represent the contribution of microstates with \( m \) particles, other than \( k \), in state 1, and with the particle \( k \) being in state \( x_k = 0 \) and \( x_k = 1 \) respectively.

The contribution of each of these subsets of microstates can now be represented as the product of contributions due to direct interactions and indirect interaction, i.e.

\[
E_X = E_X^{\text{dir}} + E_X^{\text{indir}} \tag{2.38}
\]

\[
Z_X = e^{-E_X/k_B T} \tag{2.39}
\]

\[
= e^{-E_X^{\text{dir}}/k_B T} e^{-E_X^{\text{indir}}/k_B T} \tag{2.40}
\]

\[
= Z_X^{\text{dir}} Z_X^{\text{indir}} \tag{2.41}
\]

\[
Z_0(m) = Z_0^{\text{dir}}(m) Z_0^{\text{indir}}(m) \tag{2.42}
\]

\[
Z_1(m) = Z_1^{\text{dir}}(m) Z_1^{\text{indir}}(m) \tag{2.43}
\]
where the superscript “dir” and “indir” represent the contributions due to direct and indirect interactions, respectively.

Now the contribution due to indirect interactions can be approximated as follows

\[ E_{X}^{\text{indir}} = \sum_{i \neq k} \sum_{j > i, j \neq k} w_{ij} x_{i} x_{j} \] (2.44)

\[ \hat{\omega} = \sum_{i \neq k} \sum_{j > i, j \neq k} w_{ij} / ((N - 1)(N - 2)/2) \] (2.45)

\[ Z_{0}^{\text{indir}}(m) \approx Z_{1}^{\text{indir}}(m) \approx \exp(-\hat{\omega}m(m - 1)/2k_{B}T) \] (2.46)

where \( \hat{\omega} \) is the average indirect interaction potential. The factor \((N - 1)(N - 2)/2\) in Eq. 2.45 is the total number of indirect interactions, and the factor \(m(m - 1)/2\) in Eq. 2.46 is the number of indirect interactions when \( m \) particles other than \( k \) are in state 1.

### 2.3.2.2 Contribution due to direct interactions

The contribution due to direct interactions \( Z_{0}^{\text{dir}}(m) \) and \( Z_{1}^{\text{dir}}(m) \) can be represented by the combinatorial sum of products known as elementary symmetric functions \( f_{0}(m) \) and \( f_{1}(m) \), as follows

\[ E_{X}^{\text{dir}} = \sum_{i \neq k} w_{kk} x_{k} + w_{ik} x_{i} x_{k} + w_{ii} x_{i} \]

\[ Z_{0}^{\text{dir}}(0) = e^{0} = f_{0}(0) \]

\[ Z_{0}^{\text{dir}}(1) = \sum_{i \neq k} e^{-w_{1,i}/k_{B}T} = f_{0}(1) \]

\[ Z_{0}^{\text{dir}}(2) = \sum_{i \neq k} \sum_{i2 > i1, i2 \neq k} e^{-(w_{1,i1} + w_{2,i2})/k_{B}T} \]

\[ = \sum_{i \neq k} \sum_{i2 > i1, i2 \neq k} e^{-w_{1,i1}/k_{B}T} e^{-w_{2,i2}/k_{B}T} = f_{0}(2) \]

\[ Z_{0}^{\text{dir}}(3) = \sum_{i \neq k} \sum_{i2 > i1, i2 \neq k} \sum_{i3 > i2, i3 \neq k} e^{-w_{1,i1}/k_{B}T} e^{-w_{2,i2}/k_{B}T} e^{-w_{3,i3}/k_{B}T} = f_{0}(3) \]
\begin{align*}
Z_{0}^{\text{dir}}(m) &= \sum_{i\neq k} \sum_{im>(m-1): im\neq k} e^{-w_{1,1}/k_{B}T} \cdots e^{-w_{m,m}/k_{B}T} = f_{0}(m) \\
Z_{1}^{\text{dir}}(m) &= \sum_{i\neq k} \sum_{im>(m-1): im\neq k} e^{-(w_{1,1}+w_{1,k})/k_{B}T} \cdots e^{-(w_{m,m}+w_{m,k})/k_{B}T} e^{-w_{k,k}/k_{B}T} = f_{1}(m) 
\end{align*}

(2.47)

The number of terms in the elementary symmetric functions \( f_{0} \) and \( f_{1} \) in Eq. 2.47 and 2.48, grow exponentially with \( N \). However, these elementary symmetric functions can be computed recursively in \( O(N^2) \) operations using the Newton’s identity\(^{49,50} \) generating function as follows.

\begin{align*}
a_{0}(i) &= \exp(-w_{ii}/k_{B}T) \\
g_{0}(j) &= \sum_{i} a_{0}(i)^{j} \\
f_{0}(m) &= \sum_{k=1}^{m} (-1)^{m-k+1} f_{0}(m-k) g_{0}(k)/m \\
a_{1}(i) &= \exp(-w_{ii}+w_{ik}/k_{B}T) \\
g_{1}(j) &= \sum_{i} a_{1}(i)^{j} \\
f_{1}(m) &= \sum_{k=1}^{m} (-1)^{m-k+1} f_{1}(m-k) g_{1}(k)/m
\end{align*}

(2.49)

(2.50)

(2.51)

(2.52)

(2.53)

(2.54)

\( a_{0}(i) \) and \( a_{1}(i) \) in Eq. 2.49 and 2.52 are the exponential terms from Eq. 2.47 and 2.48. \( g_{0}(j) \) and \( g_{1}(j) \) in Eq. 2.50 and 2.53 are known as Newton’s functions in the Newton’s identity formulation.

To see more clearly how the Newton’s identity works, it can be represented as the solution to a linear system of equations:

\[
\begin{bmatrix}
1 & 0 & 0 & \cdots & 0 \\
-g(1) & 1 & 0 & \cdots & 0 \\
+g(2) & -g(1) & 2 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots \\
(-1)^{m}g(m) & \cdots & -g(1) & m & 0 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
(-1)^{N-1}g(N-1) & \cdots & N-1 & 0 & \cdots & 0 & \cdots & 0 \\
\end{bmatrix}
\begin{bmatrix}
f(0) \\
f(1) \\
f(2) \\
\vdots \\
f(m) \\
\vdots \\
f(N-1) \\
\end{bmatrix} =
\begin{bmatrix}
1 \\
0 \\
\vdots \\
\vdots \\
0 \\
\vdots \\
0 \\
\end{bmatrix}
\]

This lower triangular system can be easily solved by forward substitution in \( O(N^2) \) operations.
2.3.3 Test setup

For the purpose of this analysis, we used the same randomly generated systems with inverse distance interactions, described earlier in Section 2.2.3.1.

2.3.4 Results and discussion

2.3.4.1 Accuracy of the DIA

Fig. 2.13 shows that, for very small systems ($N < 20$), the DIA is on average significantly more accurate than the Metropolis Monte Carlo method, as measured by the RMS error in average occupancy relative to the exact computation. But for larger systems the error increases dramatically. The dramatic increase in error is primarily due to numerical errors in the computation of Newton’s identity, as described below.

2.3.4.2 Catastrophic cancellation

When the elementary symmetric functions, $f(m)$, were calculated using Newton’s identity, the results were extremely inaccurate even for relatively small systems. For example, consider the system shown in table 2.1, where $N = 14$. The table shows the values of $f_0(m)$ calculated (a) using infinite precision arithmetic with Matlab’s symbolic math toolbox and (b) using double precision arithmetic. For this example $f_0(1)$ through $f(8)$ are accurate to
Table 2.1: Numerical errors due to catastrophic cancellation. Elementary symmetric function, \( f_0(m) \), calculated using different methods. \( a(m) \) are the exponents of direct interactions (Eq. 2.49). \( g_0(m) \) are the Newton functions in equation (2.50). (a) “Exact” values are computed using infinite precision. (b) “Original” values are computed with rounding using the formulation in equation (2.51). (c) “Inverse” values are computed with rounding using the formulation in equation (2.60). (d) “Separate sums” values are computed by separately summing up positive and negative intermediate values to reduce the number of subtractions. Highly inaccurate values are shown in italics.

<table>
<thead>
<tr>
<th>( m )</th>
<th>( a_0(m) )</th>
<th>( g_0(m) )</th>
<th>( f_0(m) ) calculated using</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>(a) Exact</td>
</tr>
<tr>
<td>1</td>
<td>1.1000e-1</td>
<td>1.8628e+02</td>
<td>1.8628e+2</td>
</tr>
<tr>
<td>2</td>
<td>8.2940e-3</td>
<td>7.8767e+03</td>
<td>1.3413e+4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(b) Original</td>
</tr>
<tr>
<td>1</td>
<td>1.1000e-1</td>
<td>1.8628e+02</td>
<td>1.8628e+2</td>
</tr>
<tr>
<td>2</td>
<td>8.2940e-3</td>
<td>7.8767e+03</td>
<td>1.3413e+4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(c) Inverse</td>
</tr>
<tr>
<td>1</td>
<td>1.1000e-1</td>
<td>1.8628e+02</td>
<td>1.8628e+2</td>
</tr>
<tr>
<td>2</td>
<td>8.2940e-3</td>
<td>7.8767e+03</td>
<td>1.3413e+4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(d) Separate sums</td>
</tr>
<tr>
<td>1</td>
<td>1.1000e-1</td>
<td>1.8628e+02</td>
<td>1.8628e+2</td>
</tr>
<tr>
<td>2</td>
<td>8.2940e-3</td>
<td>7.8767e+03</td>
<td>1.3413e+4</td>
</tr>
<tr>
<td>3</td>
<td>5.8407e-2</td>
<td>3.7392e+05</td>
<td>4.6840e+5</td>
</tr>
<tr>
<td>4</td>
<td>6.5258e-1</td>
<td>1.9238e+07</td>
<td>8.0065e+6</td>
</tr>
<tr>
<td>5</td>
<td>6.1026e+1</td>
<td>1.0480e+09</td>
<td>5.6328e+7</td>
</tr>
<tr>
<td>6</td>
<td>3.2772e+1</td>
<td>5.9375e+10</td>
<td>6.5948e+7</td>
</tr>
<tr>
<td>7</td>
<td>1.5554e-1</td>
<td>3.4527e+12</td>
<td>2.9686e+7</td>
</tr>
<tr>
<td>8</td>
<td>3.2128e+1</td>
<td>2.0421e+14</td>
<td>6.6700e+6</td>
</tr>
<tr>
<td>9</td>
<td>1.7440e-1</td>
<td>1.2212e+16</td>
<td>8.1245e+5</td>
</tr>
<tr>
<td>10</td>
<td>4.1831e+1</td>
<td>7.3540e+17</td>
<td>5.3340e+4</td>
</tr>
<tr>
<td>11</td>
<td>1.4467e-2</td>
<td>4.4489e+19</td>
<td>1.7303e+3</td>
</tr>
<tr>
<td>12</td>
<td>1.5218e-1</td>
<td>2.6994e+21</td>
<td>2.2692e+1</td>
</tr>
<tr>
<td>13</td>
<td>1.7201e+1</td>
<td>1.6410e+23</td>
<td>9.6018e-2</td>
</tr>
</tbody>
</table>

The displayed precision, while \( f(11) \) through \( f(13) \) are highly inaccurate.

These errors are due to the limited precision of computer arithmetic. When values of numbers are rounded to the available precision, the resulting small inaccuracies can get magnified over a large number of operations. In particular computing the difference between two large but very close numbers can result in a very large magnification of inaccuracies in the input numbers.\(^{51,52}\) It is evident from the linear system represented in equation (2.55) that solving the system involves a large number of subtractions any one of which can lead to catastrophic cancellation errors.

2.3.4.3 Reducing cancellation errors

Most of the literature on reducing cancellation errors discuss reformulating the problem to avoid calculating small differences between large numbers.\(^{51,52}\) None of these specific techniques are readily applicable to the problem at hand. However two other techniques were tested using the example system in table 2.1. (1) Reduce the number of subtractions (2) Inverse Newton’s identity formulation

Since the cancellation errors are due to subtractions, intuitively one would expect that
reducing the number of subtractions would reduce the likelihood of cancellation errors. This can be accomplished by separately accumulating all positive and negative quantities in the summation for Newton’s identity in equation (2.51). Then the value of \( x_k \) can be calculated with only one subtraction instead of up to \( k \) subtractions. This technique was tested using the example in table 2.1. Although the results improved, they were still highly inaccurate as shown also in column (d) of table 2.1. The reason there is still a large cancellation error is that the individual terms in equation (2.51) are much larger, \( O(10^{23}) \), compared to the net value of \( f_0(13) \) which is \( O(10^{-1}) \). Therefore the summation requires calculating a small difference between large terms, regardless of the order in which the summation is done.

The second technique investigated involves an inverse formulation of Newton’s identity as follows:

\[
q(i) = \sum_{j=1}^{N} \frac{1}{a(j)^i} \quad (2.55)
\]

\[
f(N) = a(1)a(2) \ldots a(N) = C \quad (2.56)
\]

\[
f(N - 1) = f(N)q(1) \quad (2.57)
\]

\[
f(N - 2) = (f(N - 1)q(1) - f(N)q(2))/2 \quad (2.58)
\]

\[\vdots\]

\[
f(m) = \sum_{k=1}^{N-m} (-1)^{m-k+1} f(m + k)q(i)/(N - m) \quad (2.60)
\]

This formulation is an upper triangular matrix that can be solved by backwards substitution.

\[
\begin{bmatrix}
N - 1 & -q(1) & \cdots & (-1)^{N-1}q(N - 1) \\
0 & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & \vdots \\
2 & -q(1) & +q(2) & \vdots \\
0 & 1 & -q(1) & \vdots \\
0 & 0 & 1 & \vdots \\
\end{bmatrix}
\begin{bmatrix}
f(1) \\
f(2) \\
f(m) \\
f(N - 1) \\
f(N)
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix}
\]

The values calculated using the inverse Newton’s identity formulation are shown in column (c) of table 2.1. The values for \( f(4) \) through \( f(13) \) are now accurate to the precision displayed, but the values for \( f(1) \) and \( f(3) \) are now extremely inaccurate.

Combining results from the original Newton’s identity formulation with results from the inverse Newton’s identity formulation will give accurate results for all \( f(m) \) for this specific example. Does this approach then resolve the problem? The combined approach was tested using two more systems with \( N = 20 \) and \( N = 30 \), and did significantly reduce the impact of catastrophic cancellations (results not shown). However, for larger systems the impact of catastrophic cancellations are still large. Therefore, this Newton’s identity based algorithm is not likely to be useful for most practical problems.
2.3.5 Conclusion

In statistical mechanics, the partition function $Z$ can be used to compute equilibrium thermodynamic properties of a system. However, the computation of the partition function scales exponentially in the number of particles $N$, and is therefore intractable for systems with $N \gtrsim 50$. Presented above is an $O(N^2)$ method, direct interaction approximation (DIA), for approximating the partition function.

For a given particle, the DIA computes the contribution to the partition function due to direct interactions (interactions involving the given particle) exactly, while using an average value for indirect interactions. Even with this approximation, the exact computation of the contribution due to direct interaction involves the calculation of the so called elementary symmetric functions, whose terms grow exponentially with $N$. However, the Newton’s identity generating function can be used to recursively calculate these elementary symmetric functions in $O(N^2)$ operations.

The DIA was tested on a set of randomly generated sample systems. The accuracy (as measured by RMS error relative to the exact calculation) of the DIA was compared to that of the basic Monte Carlo (MC) method. For small systems ($N < 20$) the DIA was on average more accurate than the MC method. However, for larger systems, the error grows dramatically. The primary reason for these large errors was numerical errors due to catastrophic cancellations (computations involving small differences between very large numbers).

Two techniques for reducing the cancellation error were examined. The first reduces the number of subtractions which are the cause of the error. However, this technique failed to significantly improve the results. The computations involved in Newton’s identity reduce very large values to much smaller values through subtraction. Therefore ultimately there will be a subtraction between very large numbers with a very small difference, resulting in a loss of precision, i.e. cancellation error.

The second technique involved reformulating Newton’s identity, the inverse Newton’s identity formulation. This formulation significantly reduced the errors in the computations that had large error in the original method. However, the inverse method had large errors where the original method had small errors. Thus one can combine the two methods to reduce the overall error. The combined approach did reduce the impact of catastrophic cancellation on smaller systems ($n < 30$), but the impact was still large for larger systems. Thus, the Newton’s identity algorithm described above may not be useful for most practical problems.
2.4 DIA using a binary split-merge algorithm

2.4.1 Introduction

The partition function $Z$ (Eq. 2.33) is used in statistical mechanics to compute macroscopic thermodynamic properties of a system. The computation of the partition function however grows exponentially with the size $N$ of the system. In the previous section we developed the direct interaction approximation (DIA) for approximating the partition function in $O(N^2)$ operations. For a given particle, the DIA approximates the contribution due to direct interactions (interactions involving the given site) exactly, while using an average value for indirect interactions. The direct interactions in the DIA are represented as a combinatorial sum of products known as elementary symmetric functions (Eq. 2.47). These elementary symmetric functions can be calculated in $O(N^2)$ operations using the Newton’s identity generating function (Eq. 2.51). However, for the larger systems tested ($N > 20$) the Newton’s identity suffers from large numerical errors due to catastrophic cancellations (computations involving small differences between very large number).

Presented here is an alternate algorithm developed to eliminate the catastrophic cancellations inherent in Newton’s identity. The algorithm is similar to the classic merge sort algorithm. The particles in the system are partitioned into a hierarchical binary tree with the root node containing all the sites, and each of the leaf nodes containing a single site. The algorithm then starts with the leaf nodes and proceeds up the binary tree by merging the results of computations between the two branches at each node. A more detailed description and proof of the algorithm is included in the Methods section below. The algorithm is tested on two types of problems which are described in the Test setup section. Accuracy of the DIA using the binary split merge algorithm is compared to the basic Monte Carlo method, in the Results and Discussion section. These results suggest that for some problems, the deterministic DIA may be preferable to the non-deterministic Monte Carlo method. These findings are summarized in the Conclusions.

2.4.2 Methods

The binary split-merge algorithm presented here is an alternative to the Newton’s identity (Eq. 2.51) for computing the elementary symmetric function (Eq. 2.47). The binary split-merge algorithm is based on the following theorem.

**Theorem 2.** If a system is partitioned into two subsystems $A$ and $B$, then the elementary symmetric function $f(m)$ for the combined system can be calculated from the elementary symmetric functions $f^A(i)$ and $f^B(j)$, for $A$ and $B$ respectively, as follows.

$$f(m) = \sum_{k=0}^{m} f^A(k) f^B(m-k)$$  \hspace{1cm} (2.62)
Proof. Expanding the right hand side of Eq. 2.62

\[ f^A(i) = \sum_{k_1 < k_2 < \ldots < k_i} a^A(k_1)a^A(k_2)\ldots a^A(k_i) \]  

\[ f^B(j) = \sum_{l_1 < l_2 < \ldots < l_j} a^B(l_1)a^B(l_2)\ldots a^B(l_j) \]  

\[ f^A(i)f^B(j) = \left[ \sum_{k_1 < \ldots < k_i} a^A(k_1)\ldots a^A(k_i) \right] \left[ \sum_{l_1 < \ldots < l_j} a^B(l_1)\ldots a^B(l_j) \right] \]  

\[ \sum_{i+j=m} f^A(i)f^B(j) = \sum_{i+j=m} \left[ \sum_{k_1 < \ldots < k_i} a^A(k_1)\ldots a^A(k_i) \right] \left[ \sum_{l_1 < \ldots < l_j} a^B(l_1)\ldots a^B(l_j) \right] \]  

where \( a^A(k_1)\ldots a^A(k_i) \) and \( a^B(l_1)\ldots a^B(l_j) \) are the contributions of direct interactions to the partition function when particles \( k_1\ldots k_i \) and \( l_1\ldots l_j \) are in state 1, from Eq. 2.49 and 2.52. The above sum represents all combinations with a total of \( m \) particles in state 1 for the combined system. Therefore

\[ \sum_{i+j=m} f^A(i)f^B(j) = f(m) \]  

2.4.2.1 Binary split merge algorithm

The binary split-merge algorithm consists of two stages. First, in the split stage, the particles are recursively split into a binary tree where each node with \( n \) particles has two branches with \( n/2 \) and \( n-n/2 \) particles in each. Then, in the merge stage, the elementary symmetric function from the two branches are recursively combined using Eq. 2.62. The elementary symmetric function values, for the leaf nodes with only one particle \( i \), are \( f(0) = 1 \) and \( f(1) = a(i) \).

2.4.3 Test setup

The accuracy of the DIA was tested on two problems. (1) The computation of thermal average magnetization, internal energy and heat capacity, for the 2-D Ising model of ferromagnetism with no external field, as described in Sec. 2.4.3.1 below. (2) The computation of average charge state of ionizable amino acids in biomolecular structure, as described in Sec. 2.4.3.2 below. The accuracy (as measured by RMS error relative to the exact computation) of the DIA was compared to that of the basic Monte Carlo method as described in Sec. 2.4.3.3 below.

2.4.3.1 2-D Ising model

The 2-dimensional Ising model is one of the most thoroughly investigated system in statistical mechanics. Despite its simplicity it exhibits many of the thermodynamic properties, such as
spontaneous magnetization, of real, and much more complex systems. Thus one could gain insights into these more complex systems through the study of the simpler 2D Ising model. Moreover, there exists an exact solution for the 2D ising model, with no external field. The exact solution of Onsager as generalized by Kaufman and implemented by Beale is used as the baseline for assessing the accuracy of the DIA.

The Ising model used for testing the DIA is an isotropic 2-D model with nearest neighbor interaction potential $J$, on an $n \times n$ lattice, with periodic boundary conditions and no external field. Each vertex of the lattice can have a spin of $+1$ or $-1$. The energy $E_X$ of this system in state $X$ is given by

$$E_X = -J \sum_{nn} s_is_j$$

where the sum is over pairs of nearest neighbors only and the $s_i$ and $s_j$ are the spins of vertex points $i$ and $j$ respectively.

The thermal average magnetization $\langle M \rangle$, internal energy $U$, and heat capacity $C_v$ are defined as follows:

$$\langle M \rangle = \frac{\sum_X M_X e^{-E_X/k_B T}}{Z}$$

$$M_X = \sum_i s_i$$

$$U = \langle E \rangle = \frac{\sum_X E_X e^{-E_X/k_B T}}{Z}$$

$$C_v = \langle E^2 \rangle - \langle E \rangle^2$$

$$\langle E^2 \rangle = \frac{\sum_X E_X^2 e^{-E_X/k_B T}}{Z}$$

where $E_X$ is the energy of state $X$, $k_B$ is the Boltzmann constant, $T$ is the temperature, and $M_X$ is the total magnetization (spin) of state $X$.

The partition function and the above thermodynamic properties can be computed exactly for the isotropic 2-D Ising model with periodic boundary condition and no external field. These exact computations form the baseline for assessing the accuracy of the DIA.

The spin states $s_i = \{+1, -1\}$ in the 2-D Ising model can be mapped to $x_i = \{0, 1\}$ as follows:

$$x_i = (s_i + 1)/2$$

$$E_X = -4J \sum_{nn} x_ix_j + 8J \sum_{i=1}^N x_i - 2NJ$$

For the DIA, the thermal average magnetization $\langle M \rangle$, internal energy $U$, and heat capacity $C_v$ are computed using the following alternate formulations:

$$\langle M \rangle \approx 2 \sum_m m(Z_0(m) + Z_1(m - 1)) - 1$$

$$U = \langle E \rangle = k_BT^2 \partial \ln Z / \partial T$$

$$C_v = \partial \langle E \rangle / \partial T$$
where $Z_0(m)$ and $Z_1(m)$ are as defined earlier by Eq. 2.42 and 2.43.

### 2.4.3.2 Average charge state of ionizable amino acids in biomolecules

Many biomolecules are made up of amino acids that can be charged or uncharged, depending on the ionization state of the amino acid. The average charge state of these amino acids can be computed as the thermal average over all possible combinations of charge states (Eq. 2.1). The DIA can be used to compute the statistical sums in Eq. 2.1. Three representative biomolecular structures, villin headpiece protein (protein databank (pdb) id 1VII), immunoglobulin binding domain (pdb id 1BDD), and thioredoxin (pdb id 2TRX), were used to test the DIA. These three structures contain 12, 20 and 32 ionizable amino acids respectively.

The mapping of the charge states of biomolecules from \{0, +1\} for cations and \{-1, 0\} for anions, to \{0, 1\} (uncharged or charged) for all ions was described earlier in Sec. 2.2.4.3.1. The structures tested here are small enough that the average charge state $\langle x_k \rangle$ for a given site $k$ can be calculated exactly as follows, and used as the baseline for assessing the accuracy of the DIA.

\[
\langle x_k \rangle = \sum_X x_k e^{-E_X/k_BT} / Z
\]  

(2.79)

For the DIA, the average charge state $\langle x_k \rangle$ for a given site $k$ is approximated as follows:

\[
\langle x_k \rangle \approx \frac{Z_1}{Z_0 + Z_1}
\]  

(2.80)

where $Z_0$ and $Z_1$ are as defined earlier by Eq. 2.35.

### 2.4.3.3 Monte Carlo method

For comparison we also calculate thermal average magnetization, internal energy and heat capacity for the 2D Ising model and average charge state for biomolecules, using the basic Metropolis Monte Carlo (MC) method. The number of Monte Carlo steps is selected such that CPU time\(^6\) for the MC method is at least as long as that of the DIA. 1,000 Monte Carlo steps were used for all testing, except for computing the average charge state for thioredoxin. In the case of thioredoxin 2,000 Monte Carlo steps were used to ensure that the computation time for the MC was at least as long as for the DIA. Before calculating the above thermodynamic properties, the system was thermalized using an equal number of Monte Carlo steps.

Additional refinements to the basic MC method, such as clustering, are beyond the scope of this preliminary proof-of-concept paper, and are not considered here.

---

\(^6\)All testing was performed on a workstation with a dual core Intel pentium 4, 3.2 GHz processor
2.4.4 Results and discussion

The DIA was tested on two problems: the computation of thermodynamic properties of the 2D Ising model, and the computation of average charge states of ionizable amino acids in biomolecules, which are described in the previous section. Accuracy (as measured by the RMS error relative to the exact computation) of the DIA was compared to the basic Monte Carlo method.

On average, for the computation of thermal average magnetization and heat capacity for the 2D Ising model, the DIA was found to be more accurate than the Monte Carlo method. For the computation of internal energy for the 2D Ising model and the computation of average charge state in biomolecules, the accuracy of the DIA was found to be comparable to that of the Monte Carlo method.

In the following analysis, temperature $T$ is divided by $J/k_B$, internal energy is divided by $J$, and heat capacity $C_v$ is divided by $k_B$, to make these quantities dimensionless.

2.4.4.1 2-D Ising model

Figure 2.14 shows the partition function for a 5x5 2-D Ising model of ferromagnetism, calculated using the exact computation, and the DIA. For the range of temperatures shown, $k_B T / J = 1.0$ to 4.0, the relative RMS error for the DIA is 25% compared to the exact computation. The error is higher for higher temperatures, and lower below the critical temperature, $T_c$, where $k_B T_c / J \approx 2.2$. 

Figure 2.14: Partition function for the 5x5 Ising model. Partition function calculated using the exact computation and the DIA, for a range of temperatures.
Figure 2.15: Probability of magnetization for the 5x5 Ising model. Probability of magnetization calculated using the DIA, Monte Carlo method, and the exact computation at a low \((k_B T/J = 2.0)\) and a high temperature \((k_B T/J = 4.0)\). The number of Monte Carlo steps (1000) used is such that the computational cost of the DIA and MC method are approximately the same. Connecting lines are shown to guide the eye.

Figure 2.15 shows the distribution of the probability of magnetization, for the 2-D Ising model of ferromagnetism at two temperatures \(k_B T/J = 2.0\) and 4.0. For the same computational cost, the DIA is more accurate than the MC method, on average in both cases. RMS error relative to the exact computation at the low temperature is 0.0171 and 0.0391 for the DIA and MC methods respectively. RMS error at the high temperature is 0.0105 and 0.0127 for the DIA and MC methods respectively.

For the same computational cost, thermal average magnetization calculated using the DIA is on average significantly more accurate than using the MC method (Fig. 2.16(a)). The RMS error relative to the exact computation, is 0.07 for the DIA compared to 0.46 for the MC. Internal energy calculated using the DIA is comparable to MC method (Fig. 2.16(b)). The RMS error relative to the exact computation, is 0.09 for the DIA compared to 0.08 for the MC. Heat capacity calculated using the DIA is comparable to MC method (Fig. 2.16(c)). The RMS error relative to the exact computation, is 0.10 for the DIA compared to 0.13 for the MC.

### 2.4.4.2 Average charge state of biomolecules

Figure 2.17 shows average charge state for each of the ionizable sites computed by the DIA and MC methods for the same computational cost, compared to the exact computation. The RMS error in average occupancy across the 60 ionizable sites in 3 biomolecules, calculated using the DIA is 0.08, which is comparable to the MC with RMS error of 0.07.
Figure 2.16: Per particle thermal average properties of the 5x5 Ising model. Thermal average magnetization, internal energy, and heat capacity, calculated using the exact computation, the DIA, and the MC method, for a range of temperatures for the 5x5 2-D Ising model of ferromagnetism. The number of Monte Carlo steps (1000) used is such that the computational cost of the DIA and MC method are approximately the same. Connecting lines are shown to guide the eye.
2.4.5 Conclusions

In statistical mechanics, the partition function $Z$ can be used to compute equilibrium thermodynamic properties of a system. However, the computation of the partition function scales exponentially in the number of particles $N$, and is therefore intractable for systems with $N \gtrsim 50$. An $O(N^2)$ method, direct interaction approximation (DIA), was developed for approximating the partition function. For a given particle, the DIA computes the contribution to the partition function due to direct interactions (interactions involving the given particle) exactly, while using an average value for indirect interactions. Even with this approximation, the exact computation of the contribution due to direct interaction involves the calculation of the so called elementary symmetric functions, whose terms grow exponentially with $N$. The Newton’s identity generating function can be used to recursively calculate these elementary symmetric functions in $O(N^2)$ operations. However, the Newton’s identity can result in large numerical errors due to catastrophic cancellations.

A binary split-merge algorithm was developed above to compute the elementary symmetric function without numerical errors due to catastrophic cancellations. This algorithm first partitions the system into a hierarchical binary tree where the root node contains the entire system and each leaf node contains a single particle. Starting with the leaf node, the algorithm then merges the elementary symmetric function for the two child nodes to compute the elementary symmetric function for the parent node.

The direct interaction approximation (DIA) using the binary split-merge algorithm was tested on two problems. One, the computation of thermodynamic properties of the 2D...
Ising model with no external field. And two, the computation of average charge state of ionizable sites in biomolecules. Accuracy (as measured by RMS error relative to the exact computation) of the DIA was compared to that of the commonly used Monte Carlo method. For the computation of thermal average magnetization and heat capacity of the 2D Ising model, the DIA was found to be more accurate the Monte Carlo method, for the same computational cost. For the computation of internal energy of the 2D Ising model, and the computation of average charge state of ionizable sites in biomolecules, the accuracy of the DIA was comparable to the Monte Carlo method. These results suggest that with additional validation and analysis, the deterministic DIA could be a practical alternative to the non-deterministic Monte Carlo method for some problems.
Chapter 3

Speeding up the computation of long range electrostatic interactions

3.1 Introduction

Today molecular dynamics simulations are routinely used to study the structure and function of biological molecules.\textsuperscript{57–60} Traditional all-atom molecular dynamics simulations model the system as a collection of atoms with additive pairwise interactions. Despite several critical approximations employed by this methodology, an all-atom simulation of a typical system with tens of thousands of atoms is limited to tens of microseconds or less, even with the computational capabilities of today’s supercomputers.\textsuperscript{61–64} Various additional approximations, for example implicit solvent models, can further speed up these simulations, but still not enough to dramatically extend the simulation times, especially for large systems.\textsuperscript{65} These simulation times are still much shorter than the time scale for many important biomolecular processes, such as ligand binding, folding of most proteins, and enzyme turnover, which have time scales in the range of tens of microseconds to seconds and longer.\textsuperscript{59,66–68} The limiting factor in most all-atom simulations is the computation of long range electrostatic interactions.\textsuperscript{69,70} For a set of $N$ atomic charges $q_1 \ldots q_N$, the long range electrostatic potential, energy, and force can be calculated as

\begin{align}
\phi_j &= \sum_{i=1}^{N} \frac{q_i}{\epsilon_{ij} r_{ij}} \\
E &= \sum_{i=1}^{N} \sum_{j>i}^{N} \frac{q_i q_j}{\epsilon_{ij} r_{ij}} \\
\vec{F}_j &= -q_j \nabla \phi_j
\end{align}

where $\phi_j$ is the electrostatic potential at point $j$, $E$ the electrostatic energy of the system, $\vec{F}_j$ the electrostatic force on atom $j$, $r_{ij}$ the distances from atom $i$ to point $j$, and $\epsilon_{ij}$ the effective dielectric function of the medium between atom $i$ and point $j$. Depending on the specific solvent model\textsuperscript{71} used, the function $\epsilon_{ij}$ may be as simple as constant $\epsilon_{ij} = 1$ as in
the explicit solvent model, or may depend on the geometry of the system in a more com-
plex manner, as in many analytical implicit solvent models such as the distance dependent
dielectric\textsuperscript{72} or the generalized Born models\textsuperscript{73–97} However, regardless of the specific math-
ematical form, the computation of electrostatic energy or forces on all $N$ atoms performed
via a pairwise formalism such as equation (3.2) scales as $O(N^2)$, unless additional approxi-
mations are made. The cost of such exact computation, incurred for each time step of the
simulation, can severely limit the duration of the simulation, or the size of the system be-
ing simulated. Several approximations have been developed specifically to reduce the costs
associated with computation of long-range pairwise interactions. For biomolecular simula-
tions, the most commonly used approximations are the spherical cutoff and the particle mesh
Ewald (PME) methods. Another approximation, the fast multipole method, has also been
used for biomolecular simulations. The simplest of these, the spherical cutoff method ignores
electrostatic interactions between a given atom and all the atoms outside a predefined cutoff
distance from the atom in question, while treating the interactions between atoms within
the cutoff distance exactly\textsuperscript{98, 99} The “industry standard” PME method uses a mathemati-
cal transformation to represent long range electrostatic interactions which decay slowly with
distance, as a sum of two rapidly converging series, one in real space and the other in Fourier
space. The mathematical transformation assumes a periodic boundary condition, where a
central cell containing the molecules of interest is surrounded by an infinite array of images
of the central cell\textsuperscript{100–103} The fast multipole method partitions the system into a hierarchi-
cal set of cubic lattices. The size of the lattices used in the multipole expansion depends on the
distance from the point in question, with a larger lattice size being used for more distant
lattices\textsuperscript{104–106} The increased speed of simulations based on the approximation methods described above
comes at a price of reduced accuracy. For example, the simple and robust spherical cutoff
method can produce many artifacts, such as spurious forces or artificial structures around
the cutoff distance\textsuperscript{107–109} The fast multipole method is more accurate than the spherical
cutoff method. It is available, for example, in the CHARMM software package\textsuperscript{110} and has
been applied to biomolecular simulations\textsuperscript{111, 112} However, so far it has not been as widely
adopted for biomolecular simulations, probably due to its algorithmic complexity and insta-
bilities caused by discontinuities inherent in the method\textsuperscript{113} The Particle Mesh Ewald (PME)
method, which is much more accurate than the spherical cutoff method, is also significantly
more complex and can produce artifacts too, such as biasing the simulation toward the most
compact conformations\textsuperscript{114, 115} Perhaps the most important limitation of the PME method
is that it is not clear if it can be extended to practical implicit solvent models where the
effective dielectric constant, $\epsilon_{ij}$, between atoms can be a complex (non-pairwise) function of
the position of all the atoms in the system\textsuperscript{116} For example, in the generalized Born model
interactions between any two atoms depend, through the effective Born radii, on the position
of all other atoms in the molecule. A clean decomposition of the atom-atom interaction into
the sums in direct and reciprocal space, required for the PME formulation may be impossible
for such potential functions. The PME method has therefore not been implemented to speed
up practical implicit solvent models, such as the generalized Born model that offers a number
of potential advantages including lower computational cost and faster conformational search.
Despite these limitations, through years of refinement the PME method has become the $de$
A very different class of approximation methods for speeding up molecular dynamics simulations are the coarse grain models. These models replace collections of atoms with a single particle. Examples of these methods include united atom, united residue, segment chain, and lattice representation models. These models can simulate very large systems over millisecond and longer timescales, however at the cost of needing approximate coarse-grained force fields and non-trivial mechanics for moving complex shapes.

Our goal is to develop and implement a systematic approximation for pairwise long-range interactions that (i) has the algorithmic simplicity and robustness of the spherical cutoff method, (ii) has accuracy approaching that of the PME method, (iii) can be used with implicit solvent models, and (iv) uses the high quality, extensively tested, all-atom force fields. Described below, is the development and implementation of such an approximation.
3.2 Hierarchical charge partitioning (HCP)

3.2.1 Introduction

The hierarchical charge partitioning (HCP) approximation presented here exploits the natural partitioning of biomolecules into its constituent components to speed up the computation of electrostatic interactions with limited and controllable impact on accuracy. Biomolecules can be systematically partitioned into multiple molecular complexes, which consist of multiple polymer chains or subunits, which in turn are made up of multiple amino acid or nucleotide groups. These components form a hierarchical set with, for example, complexes consisting of multiple subunits, subunits consisting of multiple groups, and groups consisting of multiple atoms. Atoms represent the lowest “mandatory” level in the hierarchy while the highest level depends on the problem. We assume that this hierarchy is approximately preserved during dynamics, eliminating the need to recompute the hierarchical partitioning at every step of the simulation. This is one of the key advantages of the HCP, compared to some of the more traditional approaches based on space partitioning, such as the fast multipole method.

The distribution of charges for each component used in the computation varies depending on distance from the point in question: the farther away a component, the fewer charges are used to represent the component. This study systematically explores the accuracy and speed of this simple idea in the context of molecular dynamics.

The remainder of this section is organized as follows. First we provide a detailed description of the HCP. To assess its accuracy and speed, the approximation is then applied to the computation of electrostatic interactions for a representative sample set of 600 realistic biomolecular structures. Next, we examine other practical factors that may effect the accuracy of the HCP in the context of molecular dynamics simulations. As a proof-of-concept the HCP is then implemented with some basic molecular dynamics capability and tested in a 10 ns implicit solvent simulation of a protein. In conclusion, we discuss our findings along with possible refinements.

3.2.2 Methods

3.2.2.1 Hierarchical charge partitioning

The concept of the HCP, as described here, is formalized for the specific case of speeding up electrostatic computations in classical molecular dynamics simulations using non-polarizable force fields. The atoms are treated as single point charges located at the atomic center-of-mass. Possible extensions to other types of simulations, such as QM/MM and polarizable force fields, are briefly discussed in the “Conclusions”.

There are three types of biomolecules that control the functioning of biological organisms: DNA, RNA and proteins. These biomolecules are typically organized into complexes of multiple molecular subunits. The subunits are protein, DNA or RNA polymer chains, consisting of nucleotide or amino acid groups (nucleotides in DNA and RNA subunits, and amino
acids in proteins subunits). This hierarchical grouping of atoms into groups, groups into subunits, and subunits into complexes, offers a natural hierarchical partitioning of biomolecular structures that is exploited by the method presented here. An example of such a partitioning is illustrated in figure 3.1. The figure shows a chromatin fibre made up of 100 nucleosome complexes, constructed as described by Wong et al. Each complex consists of 13 subunits. The subunits are DNA and protein polymer chains each consisting of 49 to 142 groups. And each group consists of 7 to 32 atoms. In total this chromatin fibre is made up of approximately 3 million atoms.

The HCP exploits this natural partitioning of biomolecular structures into hierarchical levels of structural components to approximate electrostatic potential, energy, and force. The atomic charges for each of these components, other than at the atomic level (level 0), is approximated by a much smaller number of point charges than the original all-atom representation. For a given point in space, the electrostatic effect of distant components is calculated using this smaller set of point charges, while the full set of atomic charges are used for nearby components. The HCP algorithm determines the level of approximation to be used in the computation, starting from the top level down to the lowest level. If a component is beyond the pre-specified threshold distance, the approximate distribution of charges for the component are used in the computation of electrostatic interactions and no further computations are performed involving its constituent lower level components. The threshold distance also ensures that electrostatic interactions between nearby atoms are treated exactly even if they belong to different components. Figure 3.2 illustrates the HCP algorithm for a biomolecular structure consisting of two levels of hierarchical organization. Although the example for two levels of hierarchical organization, the algorithm can be extend to additional levels, e.g. to three levels in the case of the chromatin fibre shown in figure 3.1. The general HCP algorithm is described below. The computation of magnitude and location of the smaller set of point charges, that approximate the charge distribution of a component in step 2 of the HCP algorithm is described in the following subsection. To determine the level of approximation to be used in the computations, we define a threshold distance $h_k$ for each structural level $k$: $h_1$ for groups (level 1), $h_2$ for subunits (level 2), $h_3$ for complexes (level 3), etc. Then, the HCP algorithm proceeds as follows:

1. Partition the structure into a hierarchical set of components. The partitioning may depend on the type of problem. For example, while the chromatin fibre shown in figure 3.1 is naturally partitioned into three levels, smaller structures may only consist of one or two levels, and larger structures may be partitioned into more than three levels. The partitioning may also depend on the expected dynamics. For example, where the dynamics may result in the separation of subunits that initially form a complex, one may not want to treat the complex as a single next level component.

2. Compute the approximate distribution of point charges for each component, as described in the subsection below, other than at the atomic level (level 0) which is assumed to be given.

---

\[1\] In the structure shown here the 49 nucleotide segments of DNA linking nucleosome complexes are treated as separate chains (subunits).
Figure 3.1: Natural partitioning of a chromatin fibre. (a) The fibre is made up of 100 nucleosome complexes. The individual nucleotide groups in the fibre are represented as red beads and amino acid groups as grey beads. (b) Each complex (level 3) is made up of 13 subunits with the segments of DNA linking nucleosome complexes being treated as separate subunits. A complex is shown here with each subunit represented in a different color. (c) Each subunit (level 2) is made up of 49-142 groups. The linker histone subunit is shown here with the groups colored by the type of amino acid. (d) Each group (level 1) is made up of 7-32 atoms (level 0). A histidine amino acid group is shown here with atoms represented as small spheres and covalent bonds between the atoms represented as links. The atoms are colored by the type of atom. The total fibre consists of approximately 3 million atoms. The images were rendered using VMD.\textsuperscript{43} For clarity only 10 of the 13 subunits are shown in (a) and (b).
3. Initially set the current component, $C$, to be the entire system. The current component $C$ is updated in step 5.3 as the algorithm proceeds from the top (entire system) down to the atomic level.

4. Set $k$ to be the level of the next lower level components within the current component $C$, and $h_k$ the corresponding threshold distance. For example, for the chromatin fibre shown in figure 3.1, if the current component, $C$, is the entire system (the chromatin fibre), then the next lower level components are the nucleosome complexes (level 3), and the corresponding threshold distance is $h_3$. Similarly if the current component is a nucleosome complex (level 3), then the next lower level components are DNA and protein subunits (level 2) and the corresponding threshold distance is $h_2$. For the atomic level (level 0) set the threshold distance $h_0$ to 0.

5. For each next lower level component, $S$, within the current component $C$:

5.1 Calculate the distance from the point in question to the next lower level component $S$. The distance can be suitably defined, for example as the center-of-mass or center-of-charge.

5.2 If the distance to the next lower level component $S$ is greater than the threshold distance $h_k$, the approximate distribution of charges for $S$ are used for the computation of electrostatic interactions in equations (3.1)-(3.3). In the case where $S$ is a lowest level (level 0) component the given atomic charge is used in the computation.

5.3 Otherwise set the current component $C = S$ and return to step 4.

The top down approach described by the algorithm above can result in significant computational savings. For example, for a typical amino acid group consisting of 16 atoms, if the group is beyond the threshold distance for groups, $h_1$, the 16 all-atom computations are reduced to a much smaller number of computations, for example one or two, corresponding to the reduced number of charges that approximate the charge distribution at level 1. A detailed analysis of computational complexity in the general case is beyond the scope of this proof-of-concept study. However, one can get a general idea of the scaling of computational cost by considering the following hypothetical example. Consider a hypothetical structure consisting of $N$ atomic charges distributed within the entire structure in such a way that for each atomic charge there are exactly $k$ atomic charges (level 0) within the level 1 threshold distance, $k$ groups (level 1) between the level 1 and 2 threshold distances, $k$ subunits (level 2) between the level 2 and 3 threshold distances, and so forth. Now assume that each component is represented by a single charge. For such a hypothetical structure consisting of $L$ levels, for each atomic charge the HCP calculates $k$ pairwise interactions per level, giving a total of $NLk$ interactions to compute. To express computational cost in terms $N$ we use the relationship between $N$ and $L$. Note that the structure is a hierarchical tree where each node represents a component with $k + 1$ nodes below it. The lowest level zero is $L$ levels from the top and thus has exactly $(k + 1)^L = N$ charges. Therefore $L = \log_k N$ which gives a $O(Nk \log_k N) \sim O(N \log N)$ scaling for the HCP, just like the PME method. The above estimate assumes that the computational cost of calculating the approximate distribution
Figure 3.2: Illustration of the HCP algorithm. Biomolecular structures are naturally organized into multiple hierarchical levels of components – complexes, subunits, groups and atoms – as represented by the tree structure shown here. Approximations are used for computations involving distant components, while exact atomic computations are used for atoms within nearby groups. The HCP algorithm proceeds from the top level down to the lowest level to determine the level of approximation to use. The level of approximation used is determined by the distance of a component from the point of interest compared to the threshold distance for the level of the component – $h_1, h_2, h_3$ for level 1, 2 and 3 respectively.
of charges in step 2, scales equally or better than $O(N \log N)$ which is the case as will be shown in the subsection below. It is interesting to note that similar hierarchical structures occur in nature at vastly different scales, such as in astrophysics where similar hierarchical partitioning methods have been applied to solve an $O(N^2)$ gravitational interaction problem with an $O(N \log N)$ approximation.

3.2.2.2 Approximating the charge distribution of components

There are many different ways to approximate a larger set of atomic charges with a smaller set. For example, one could identify charges that most closely approximate the electrostatic potential distribution at the surface of a group, subunit or complex, similar to the methods used by the restrained electrostatic potential (RESP) fit\textsuperscript{126} or by the discrete surface charge optimization (DiSCO) approach.\textsuperscript{127} For this proof-of-concept study we have chosen a simpler approach, and defer the optimization of this method to future studies. The approach presented here uses a generalization of the “center-of-charge” to compute locations of the smaller set of point charges. The center-of-charge for a set of point charges is defined in a manner similar to the more familiar center-of-mass. To approximate a structural component by more than one charge we group the charges into subsets, and then compute the “center-of-charge” for each subset. Specifically, where we seek to approximate a structural component by $m$ point charges, the $n$ original atomic charges of the structural component are grouped into $m$ subsets. The location of the $m$ charges is then represented by the center-of-charge for each of these $m$ subsets, and their magnitude is given by the sum of charges in the corresponding subset. The grouping of the original atomic charges of a structural component into $m$ subsets is based on the magnitude of the charges. For example, for a 2-charge approximation ($m = 2$), the charges are separated into two subsets, one for all positive charges and another for all negative charges. More generally, the charges are separated into subsets such that subset $j$ contains all charges $q_i$ such that $Q_j^{\text{min}} \leq q_i < Q_j^{\text{max}}$, $i = 1 \ldots n$, $j = 1 \ldots m$, where $Q_j^{\text{min}}$ and $Q_j^{\text{max}}$ are predefined ranges for each subset and $n$ is the number of the original atomic charges in the component. For example, the ranges for a 2-charge approximation may be defined as, $Q_1^{\text{min}} = -\infty \leq q_i < 0 = Q_1^{\text{max}}$, and $Q_2^{\text{min}} = 0 \leq q_i < +\infty = Q_2^{\text{max}}$. The location of the approximate point charge, $Q_j$, is then calculated as the center-of-charge for the atomic charges in subset $j$. For a component with $n$ charges, $q_1 \ldots q_n$ located at $r_1 \ldots r_n$, the $m$ approximate point charges $Q_1 \ldots Q_m$ located at $R_1 \ldots R_m$ are calculated as follows:

$$Q_j = \sum_{i: Q_j^{\text{min}} \leq q_i < Q_j^{\text{max}}} q_i$$

$$R_j = \begin{cases} \sum_{i: Q_j^{\text{min}} \leq q_i < Q_j^{\text{max}}} \frac{q_i r_i}{Q_j} & Q_j \neq 0 \\ 0 & Q_j = 0 \end{cases}$$

where $\sum_{i: Q_j^{\text{min}} \leq q_i < Q_j^{\text{max}}}$ represents the sum over the subset of atomic charges within a component with charge values $q_i$ in a predefined range $Q_j^{\text{min}}$ through $Q_j^{\text{max}}$. The computational cost of calculating the magnitude and location of these approximate point charges scales as $O(N)$. 

56
To define the range of charge values, $Q_j^{\text{min}}$ and $Q_j^{\text{max}}$, the entire range of typically encountered charge values is divided into $m$ ranges of equal sizes. For biomolecules with atomic charges typically in the range of -1 to +1 atomic units (au), we have the following ranges for $q_i$ in the summation in equation (3.4). For a 1-charge approximation $Q_1^{\text{min}} = -\infty \leq q_i < +\infty = Q_1^{\text{max}}$ (all charges), for a 2-charge approximation $Q_1^{\text{min}} = -\infty \leq q_i < 0 = Q_1^{\text{max}}$ (negative charges), and $Q_2^{\text{min}} = 0 \leq q_i < +\infty = Q_2^{\text{max}}$ (positive charges), for a 3-charge approximation $Q_1^{\text{min}} = -\infty \leq q_i < -1/3 = Q_1^{\text{max}}, Q_2^{\text{min}} = -1/3 \leq q_i < +1/3 = Q_2^{\text{max}},$ and $Q_3^{\text{min}} = +1/3 \leq q_i < +\infty = Q_3^{\text{max}}$, and so on. For example the amino acid leucine with 19 atoms and a zero net charge, can be approximated by a 1-charge approximation of 0, or a 2-charge approximation consisting of charges -2.0059 and +2.0059 placed at each respective center-of-charge of all negative and positive charges, as calculated by equation (3.5).

The analysis in the following sections is limited to 1- and 2-charge approximations. Although using a larger number of charges to represent each component can improve accuracy, it also increases computational cost. Preliminary studies (results not shown) indicate that for the proof-of-concept method given by equations (3.4) and (3.5), the incremental improvement in accuracy is small compared to the additional computational cost. A detailed analysis of the costs and benefits of using 3 or more charges is beyond the scope of this proof-of-concept study.

As described in the HCP algorithm, the approximate distribution of charge for a component, used in the computation of electrostatic interactions, equations (3.1) - (3.3), depends on the distance from the component to the point of interest. In the current implementation of the HCP, the distance to a component is calculated as the distance to its geometric center which is defined as:

$$\text{Geometric center} = \sum_{i=1}^{n} \frac{r_i}{n} \quad (3.6)$$

One can avoid calculating the geometric center by using the distance to one of the approximate point charges instead of the distance to the geometric center. Also, one can avoid recomputing the approximate distribution of charges at every single step of a molecular dynamics simulation by placing the approximate point charges at the location of the closest atomic charge. Assuming that the relative location of charges within a component do not change significantly during some limited number of steps of a molecular dynamics simulation, the approximate distribution of charges may not need to be recalculated at every single step. The analysis of these alternative approaches is deferred to a future study.

### 3.2.3 Test setup

we compute the electrostatic force and energy for a sample set of 600 real biomolecular structures ranging in size from 297 to 2,899,800 atoms and ranging in charge from -20,800 to +236 au. The biomolecular structures consist of 595 subunits ranging in size from 297 to 8,542 atoms, 4 complexes ranging in size from 25,086 to 490,040 atoms, and the chromatin fibre consisting of 2,899,800 atoms. The atomic coordinates for these biomolecular structures are obtained from the protein databank (PDB).128 The atomic charges for the biomolecular
structures are assigned using the H++ system (www.cs.vt.edu/biophysics/H++) which uses the standardized continuum solvent methodology to compute protonation states of ionizable groups. For the 595 subunits in the test set, the HCP uses only the group (level 1) approximation because each of these structures consists of a single subunit. Similarly, for the 4 complexes in the test set, only the group and subunit (level 1 and 2) approximations are used. For the chromatin fibre, figure 3.1, all three levels of the approximation, – group, subunit and complex – are used.

The accuracy of the approximation is measured by the relative force error and relative energy error. Relative force error is defined as the root mean square (RMS) error in force divided by the mean force, \( \frac{\sqrt{\langle \| \vec{F}_e - \vec{F}_a \| \rangle^2 / N}}{|F_m|} \) where \( \vec{F}_e \) is the force from the exact all-atom calculation, \( \vec{F}_a \) the force from an approximate calculation, \( N \) the total number of atoms, and \( F_m \) the mean value for \( \| \vec{F}_e \| \). Relative energy error is defined as the RMS error in energy divided by the mean energy, \( \frac{\sqrt{\langle (E_e - E_a)^2 / N_s \rangle / |E_m|}}{|E_m|} \) where \( E_e \) is the energy from the exact all-atom calculation, \( E_a \) the energy from the approximate calculation, \( N_s \) the number of structures, and \( E_m \) the mean value for \( E_e \). The exact all-atom calculation of electrostatic energy and force uses the Coulomb equations (3.2) and (3.3). Unless otherwise specified, we use a fixed value of 1 for the dielectric constant. As we will show in the next section, including the solvent effect via an implicit solvent model, may not significantly change the relative force error for the HCP. The speed-up factor is calculated as \( t_e / t_a \), where \( t_e \) is the CPU time for the exact all-atom calculation and \( t_a \) the CPU time for the approximation.

For comparison we also calculate the error in electrostatic force and energy for the spherical cutoff and the particle mesh Ewald (PME) methods. While comparison between the HCP and the spherical cutoff method is straightforward, this is not the case for the PME which assumes periodic boundary conditions. Within the readily available implementations of the PME, such as the Amber software package which is used here for the error analysis, it is not straightforward to separate the error resulting from the artificial periodic boundary and all other errors inherent to the PME. Since most biomolecular systems are non-periodic we use the following strategy to compare the HCP and the PME. We use PME parameter settings that are suggested for typical biomolecular simulations. In particular we assume that the buffer between the biomolecule and the edge of the periodic cell is 10 Å. The dielectric constant is set to 1 for both the HCP and PME calculations. Other parameters are set to their defaults for the Amber version 9 software package. Error for the PME approximation is calculated as the difference between Amber PME energy/force and the corresponding Amber value calculated without the PME (spherical cutoff distance greater than the system size). Due to the limitations of our system configuration the computation of error for the PME approximation is limited to less than 50,000 atoms.

\[ ^{3}\text{Eliminating the error due to artificial periodicity by using a very large box size is not a realistic strategy for anything but the smallest systems.} \]
3.2.4 Results and discussion

3.2.4.1 Accuracy and speed of the HCP

To assess the accuracy and speed of the HCP, Figure 3.3 shows the accuracy and speed-up comparison for our sample set of 600 biomolecular structures in vacuum. The HCP calculations use threshold distances of $h_1 = 10\, \text{Å}$, $h_2 = 70\, \text{Å}$, and $h_3 = 125\, \text{Å}$, for groups (level 1), subunits (level 2), and complexes (level 3) respectively. The spherical cutoff and PME methods use a cutoff distance of 10 Å. For this sample set of biomolecular structures, the HCP results in a relative force error of 0.024 for the 1-charge HCP and 0.011 for the 2-charge HCP, which is on average significantly more accurate than the spherical cutoff method that has a relative force error of 0.089. When averaged over the entire test set, the 1-charge approximation approaches the accuracy of the more sophisticated PME method that has a relative force error of 0.021, and the 2-charge approximation is on average slightly more accurate than the PME method. However, unlike the relative accuracy of the HCP, the relative accuracy of the PME method is not uniform across the range of molecular sizes, Figure 3.3. For structures in our test set that are smaller than a few thousand atoms, we find that the PME accuracy is very high compared to its accuracy for larger structures, and it is also significantly higher than that of the HCP. As the structure size increases, the force error of PME becomes comparable to that of the HCP, and for even larger structures the PME eventually becomes less accurate. We stress that in these estimates we have used the same parameter settings for the PME for all the structures, including a constant (10 Å) “solvent buffer” size, that is the smallest distance from the biomolecular structure to the cell boundary. For larger biomolecular structures, the structure in the central cell is therefore closer, relative to the structure size, to its surrounding periodic images, resulting in larger errors due to the artificial periodicity. Although the 10 Å (direct sum) cutoff distance is typically plentiful for high accuracy of the PME, larger values of the spherical cutoff are often used. Therefore we also compare the spherical cutoff method with cutoff distances of 15, 20 and 25 Å to the HCP with level 1 threshold distances of 15, 20 and 25 Å. The 1-charge HCP with corresponding relative force errors of 0.01, 0.005, and 0.003 is significantly more accurate than the spherical cutoff method with relative force errors of 0.05, 0.03 and 0.02.

The electrostatic energy calculated by the HCP results in a relative energy error of 0.001 for the 1- and 2-charge HCP. This is on average much more accurate than both the PME and spherical cutoff methods that have relative energy errors of 0.097 and 0.111 respectively. Electrostatic energy calculated by the particle mesh Ewald method is on average less accurate than the HCP due to the contributions of artificial periodic images of the central cell. However, for many applications, such as molecular dynamics, the value of energy in a given state is less important than the difference in energy between states, and so the force error is the more relevant metric.

Figure 3.3(c) shows that the HCP can be multiple orders of magnitude faster than the corresponding exact all-atom computation; the amount of speed-up depends on the size of the structure. For the chromatin fibre with 2,899,800 atoms the 1-charge HCP is 906 times faster than the exact all-atom calculation, while for a protein of 287 atoms (protein G) the 1-charge
HCP is 2 times faster. These results are consistent with the $O(N \log N)$ scaling of the HCP discussed in section 3.2.2.1. We do not attempt a direct comparison of computational times between the unoptimized, proof-of-concept implementation of the HCP with that of optimized implementations of the PME and the spherical cutoff methods, because these timings are highly implementation dependent. However, for a molecular dynamics implementation, described below, we show that the HCP can be faster than the spherical cutoff method.

The above results are based on the simple proof-of-concept approach described in the previous section. Specifically, we use the center-of-charge method for calculating the approximate distribution of charges and a fixed threshold distance for each level. We speculate that using a more sophisticated approach, such as calculating the approximate distribution of charges by optimizing fit to surface potential using methods such as restrained electrostatic potential (RESP) fits, or varying the threshold distance based on component size, should result in greater accuracy.

### 3.2.4.2 Dependence on molecular size and charge

Although not apparent from the log-log scale plot in figure 3.3, there is a weak correlation between the accuracy of the HCP, as assessed by relative force error, and the size of the molecular structure. The correlation coefficient is 0.49 for a 1-charge approximation and 0.47 for a 2-charge approximation. The correlation is more evident in the linear scale plot of relative force error as a function of size in figure 3.4(a). For smaller structures a larger percentage of atoms are treated exactly therefore one would expect the error to be lower. A similar comparison for the particle mesh Ewald (PME) method shows a correlation coefficient of 0.49. For the test set used here, the spherical cutoff method has a slightly weaker correlation with size, with a correlation coefficient of 0.36. Relative force error as a function of molecular charge, figure 3.4(b), shows that accuracy of the spherical cutoff and PME methods are weakly correlated to molecular charge with correlation coefficients of 0.57 and 0.59 respectively. However, the accuracy of the HCP is not correlated to molecular charge. The correlation coefficient for a 1-charge approximation is 0.11, and 0.19 for a 2-charge approximation. A detailed theoretical analysis of these trends is beyond the scope of this proof-of-concept study.

The above correlation analysis represents the group (level 1) approximation for the 595 subunits, out of the 600 biomolecular structures in our sample set. Correlation analysis for the subunit and complex (level 2 and 3) approximation would not be statistically meaningful due to the small sample size of only four complexes and one multi-complex structure. The preceding analysis uses a group (level 1) threshold distance of $h_1 = 10 \, \text{Å}$ for the HCP, and a 10 Å cutoff distance for the PME and spherical cutoff methods.

### 3.2.4.3 Choice of parameter values: accuracy/speed trade-off

Accuracy and speed of the HCP is controlled by two sets of parameters: the number of charges in the approximate distribution of charges per component, $m$, and threshold dis-
Figure 3.3: Accuracy and speed-up comparison. Accuracy and speed-up for a sample set of 600 biomolecular structures in vacuum. (a) Relative force error shows that for electrostatic force the hierarchical charge partitioning (HCP) is more accurate than the spherical cutoff method, and on average approaches the accuracy of the particle mesh Ewald (PME) method for all but very small structures. (b) Relative error in energy shows that for electrostatic energy the HCP is on average more accurate than both the spherical cutoff and PME methods. (c) Speedup factor compared to the exact all-atom calculation shows that the HCP can be multiple orders of magnitude faster than the exact $O(N^2)$ computation for large structures. The above computations use the following parameters. Spherical cutoff distance = 10 Å, the HCP threshold distances of $h_1 = 10$ Å, $h_2 = 70$ Å, and $h_3 = 125$ Å, and PME cutoff distance = 10 Å. RMS error and speed-up factor for individual structures are grouped into bins, based on structure size in increments of 1000 atoms, and plotted at the midpoint of each bin.
Figure 3.4: Accuracy as a function of size/charge. Accuracy, as measured by relative force error, is plotted as a function of (a) molecular size as measured by the number of atoms, and (b) absolute molecular charge. The HCP uses a 1-charge approximation with group threshold $h_1 = 10 \text{ Å}$. The PME and spherical cutoff methods use a 10 Å cutoff distance.

The trade-off between accuracy and speed for our sample set is shown in figure 3.5. The 2-charge HCP is more accurate on average, but slower, than the 1-charge approximation. And a larger threshold distance results in a more accurate approximation, on average. It is interesting to note that interactions between atoms at very large distances can affect the accuracy of the computations. For example, changing the subunit threshold distance, $h_2$, from 50 Å to 75 Å significantly improves accuracy.

Figure 3.5 also suggests how one might select parameter values for the HCP. The trade-off shown in these figures is the average value for our sample set described in the previous section. For any given structure the trade-off may be different, therefore a similar analysis for the specific structure may be performed to identify the best set of parameters. Alternatively, one can consider a more flexible mechanism where threshold distance is a function of the size and charge of the component. We do not consider these more elaborate schemes in this proof-of-concept work.

3.2.4.4 Removing discontinuity at threshold boundaries

At threshold boundaries between levels of the HCP, there is a switch in the level of approximation used by the HCP for the calculation of energy and forces. The use of two different levels of approximation across the threshold boundary introduces discontinuities in the calculated values for energy and forces at the threshold boundary. Due to these discontinuities
Figure 3.5: Trade-offs between accuracy and speed. Accuracy is measured as relative force error. (a) Average trade-off for our sample set of 595 real biomolecular subunits. Group (level 1) threshold distance for each data point shown next to the data point. (b) Average trade-off for our sample set of 4 complexes, using group (level 1) threshold distance $h_1 = 10$ Å. Subunit (level 2) threshold distance for each data point shown next to the data point. (c) Average trade-off for the chromatin fibre, using $h_1 = 10$ Å and $h_2 = 70$ Å group (level 1) and subunit (level 2) threshold distances. Complex (level 3) threshold distance for each data point shown next to the data point. Connecting lines shown to guide the eye.
energy may not be conserved during the course of molecular dynamics simulation. The discontinuity in calculated energy and forces can be eliminated using a smoothing function. To demonstrate the effect of such a smoothing function on the HCP, we calculate electrostatic potential along a line passing through the hollow core of the chromatin fibre shown in figure 3.6. To highlight the effect of the discontinuity, the line through the central core is offset from the center to pass as close as 3 Å – much less than the HCP threshold distances – from some of the atoms surrounding the core.

The smoothing function used here makes the electrostatic potential and its derivative continuous by gradually switching from one level of approximation to another over a small switching distance beyond the threshold distance. The function is adapted from a similar function used for the spherical cutoff approximation as described by Loncharich and Brooks. For an HCP threshold distance $h_k$ and a smoothing distance $s_k$, where the subscript $k$ refers to the HCP approximation level, the potential, $\phi$ at distance $r$ within the smoothing region, $h_k < r < h_k + s_k$, is calculated as:

$$\phi(r) = S(r)[\phi(h_k + s_k) - \phi(h_k)] + \phi(h_k) \quad h_k < r < h_k + s_k \quad (3.7)$$

where the smoothing function $S(r)$ is defined as

$$S(r) = \frac{(h_k^2 - r^2)^2(h_k^2 + 2r^2 - 3(h_k + s_k)^2)}{(h_k^2 - (h_k + s_k)^2)^3} \quad h_k < r < h_k + s_k \quad (3.8)$$

The smoothing region in the above equations starts at the HCP threshold distance, $h_k$, and extends out to $h_k + s_k$. 

Figure 3.6: Geometric setup used for estimating discontinuity. Left: Chromatin fibre showing the line along which electrostatic potential is calculated. Right: Cross-section of chromatin fibre looking down the fibre axis. The line passes through the central core as close as 3 Å from some of the atoms surrounding the core. Image rendered using VMD.

108

132
Figure 3.7 shows the effect of applying the smoothing function to the HCP. The figure shows the electrostatic potential, \( \phi \), and the \( z \)-component of force, \( f_z \), acting on a unit charge. The force, \( f_z \), for the HCP with smoothing is numerically calculated as \( -\partial \phi / \partial z \), along the line passing through the central core of the chromatin fibre, whereas \( f_z \) for the HCP without smoothing, the exact all-atom calculation and the spherical cutoff methods are calculated by the Coulomb equation (3.3). For the chromatin fibre, electrostatic potential and force calculated by the HCP, with and without smoothing, is significantly more accurate than the calculations based on the spherical cutoff method. The HCP calculations use threshold distances of \( h_1 = 10 \, \text{Å} \), \( h_2 = 70 \, \text{Å} \), and \( h_3 = 125 \, \text{Å} \), for groups (level 1), subunits (level 2) and complexes (level 3). The smoothing function uses a smoothing region, \( s_k \), that is 30% of the threshold distance, \( h_k \).

3.2.4.5 Effect of the violation of Newton’s Third Law

Adaptive resolution methods, such as the spherical cutoff method with a switching function and multi-scale models with a mixing function, are known to result in a net non-zero force (and torque) on a closed system, in violation of Newton’s third law. This is also true for the HCP. This residual force is due to the mixing of two different levels of approximation. See figure 3.8 for an illustration. During molecular dynamics simulations these residual forces introduce an artificial motion of the center-of-mass of the system and an overall rotation of the structure.

To estimate the effect of the residual force due to the HCP, we calculate the linear and rotational displacement and kinetic energies for our sample set of 600 biomolecular structures described above. For these calculations we treat the molecule as a rigid body. The rotational velocity is approximated assuming the principle axis of rotation passes through the center-of-mass. For our sample set of biomolecular structures in vacuum the average net residual force is \( 1.4 \times 10^{-8} \, \text{N} \), and the average net residual torque about the center-of-mass is \( 2.3 \times 10^{-17} \, \text{N} \). After 10 steps of a typical molecular dynamics simulation (10 fs), the average linear center-of-mass displacement due to the net residual force, would be \( 1.4 \times 10^{-5} \, \text{Å} \), with an average linear kinetic energy of 0.0358 kT, and the average rotation about the center-of-mass due to the net residual torque would be \( 8.5 \times 10^{-7} \, \text{rad} \), with an average rotational kinetic energy of 0.0002 kT. These displacements are very small, so one can expect that the standard techniques used to eliminate center-of-mass motion and rotation in molecular dynamics simulations will work for the HCP. As we shall see in the next section, these spurious motions might also be “eliminated” by the use of stochastic collisions employed in constant temperature simulations. The chromatin fibre in figure 3.1 represents the worst case with a net force of \( 8.2 \times 10^{-6} \, \text{N} \), and net torque about the center-of-mass of \( 1.4 \times 10^{-14} \, \text{N} \). After 10 steps of a typical molecular dynamics simulation (10 fs), the linear displacement for the chromatin fibre would be \( 1.1 \times 10^{-4} \, \text{Å} \), linear kinetic energy 21.3555 kT, rotation about the center-of-mass 6.6 \times 10^{-6} \, \text{rad} \), and rotational kinetic energy 0.0595 kT. We speculate that the large number of highly charged components in the chromatin fibre give rise to a larger net residual force on the molecule compared to the other structures in the sample set.
Figure 3.7: Effect of the smoothing function. Figures show electrostatic potential and force calculated using the spherical cutoff method, the exact all-atom computation, and the HCP with and without smoothing. (a) Electrostatic potential, $\phi$, along the $z$-axis passing through the central core. (b) Expanded view of potential for a 100 Å region. (c) $z$-component of electrostatic force, $f_z$, acting on a unit charge along the $z$-axis. (d) Expanded view of the force for a 100 Å region. For the HCP with smoothing, force is numerically calculated as $f_z = -\partial \phi / \partial z$. The spherical cutoff method uses a 10 Å cutoff distance. The HCP uses $h_1 = 10$, $h_2 = 70$ and $h_3 = 125$ Å for level 1, 2 and 3 threshold distances. The smoothing region for each threshold is 30% of the corresponding threshold distance.
A simple 3–atom system

\[ q_1 = e \]

\[ f_{21} + f_{31} \]

\[ F_{\text{HCP}} = C(q_2 + q_3) = 0 \]

\[ q_2 = -e \]

\[ f_{12} + f_{32} \]

\[ F_{\text{HCP}} = f_{12} + f_{13} \neq 0 \]

\[ q_3 = e \]

\[ f_{13} + f_{23} \]

Figure 3.8: Violation of Newton’s third law. Illustrative 3-atom system with charges \( q_1, q_2, \) and \( q_3 \). Atoms 2 and 3 have equal and opposite charges, \( q_2 = -q_3 = e \). Calculated forces between atoms \( i \) and \( j \) are shown as \( f_{ij} \). For the exact all-atom calculation, the total force acting on the center-of-mass, \( \sum_{i,j} f_{ij} = 0 \) since \( f_{ij} = f_{ji} \) per Newton’s third law. For the HCP, let atoms 2 and 3 belong to the same group while atom 1 does not belong to the group. The approximate point charge for the group represented by \( a \), is \( q_a = q_2 + q_3 = 0 \). For such a system the force acting on atom 1 due to the group is zero, while, in general, the net force on atoms 2 and 3, \( f_{12} + f_{13} \), is not zero. Therefore the HCP can result in a non-zero net force on a closed system, in violation of Newton’s third law.

3.2.4.6 Treatment of solvent

The HCP exploits the natural partitioning of biomolecular structures into a hierarchical set of components: complexes, subunits, groups and atoms, to reduce the computational cost of molecular dynamics simulations. This concept could in principle be extended to treat solvent molecules as subunits. However, since the most important solvent is composed of 3-atom water molecules, the computational cost reductions obtained through the HCP would be small, while introducing significant additional errors. For example a water molecule represented by three point charges, e.g. the TIP3P water model, could be treated as a single subunit and approximated by one or two point charges. The 1-charge HCP for water molecule would result in an approximate point charge of zero, effectively ignoring water molecules beyond the threshold distance. This is similar to the spherical cutoff method which has been shown to produce severe artifacts. The 2-charge HCP, on the other hand, would not result in any tangible computational cost savings while still most likely producing serious artifacts.

We speculate that the HCP may still be of benefit in explicit water simulations of very large solutes, such as viral particles, surrounded by a relatively thin layer of solvent. In this case, one can treat the solvent atoms exactly (explicit solvent model) and still benefit from the HCP being applied to the solute. Another type of explicit solvent simulations where the HCP may be of benefit is where artificial periodicity must be avoided. However, we expect the HCP to make the most impact in computations that utilize pairwise implicit solvent models.
The implicit solvent model treats the solvent as a continuum and calculates the electrostatic effect of the solvent on individual solute atoms using various approximations, including those based on analytical expressions. Although not as accurate as the explicit solvent model, there are a number of potential benefits from using the implicit solvent model.\textsuperscript{65,74,80,81,90,135–137} One of the key benefits is the computational cost reduction from treating only the solute atoms explicitly. However, for the fast analytical implicit solvent models based on pairwise summation schemes popular in molecular dynamics simulations, such as the generalized Born (GB) model, the computation of the electrostatic effect of the solvent on solute atoms scales poorly as $O(N^2)$, where $N$ is the number of solute atoms. Therefore the HCP, with its $O(N\log N)$ computational complexity, is poised to make a large impact in these types of simulations. The key to implementing the HCP for implicit solvent models, is handling of the dielectric function $\epsilon_{ij}$ in equations (3.1)-(3.3), where $j$ can represent an HCP component instead of an atom. For “simple” implicit solvent models, such as the distance dependent dielectric model, $\epsilon_{ij}$ is simply a function of distance, from atom $i$ to atom $j$ only. To implement HCP for such a model, one can simply replace the distance to individual atomic charges within a component by the distance to the approximate charges for a component. This is how it is done in the proof-of-concept simulation described in section 3.2.4.7 below. For more complex implicit solvent models, such as the GB model, $\epsilon_{ij}$ is a non-trivial function of the entire structure, related to the so-called effective Born radii of both atoms.\textsuperscript{79} The Born radius for an atom is generally estimated as a function of it’s relative location within the molecule.\textsuperscript{65,74} To implement HCP for the GB model, one can define the appropriately averaged effective Born radius for each hierarchical component. The residue-level effective radii defined by Archontis and Simonson\textsuperscript{87} give an example of one such averaging scheme that might also be useful here. We stress that any specific definition will have to be thoroughly tested in the specific context of the HCP. Then, similar to using approximate charge distribution for components, the HCP can use the “component Born radii” for GB computations involving interactions with distant components. Similarly, we envision an $O(N\log N)$ hierarchical partitioning method for the calculation of individual atomic effective Born radii. At present an immediate speed-up of the computation of individual effective Born radii can be achieved by limiting the maximum distance between atom pairs used in the computation of effective Born radii, the RGBMAX option already implemented in Amber.\textsuperscript{131} A detailed analysis of implicit solvent implementations of HCP is deferred to a future study.

An additional comment is warranted on the effect of solvent on the accuracy of the HCP. Since effective pairwise interactions in the presence of solute/solvent boundary fall off faster than $1/r$, one expects\textsuperscript{1} that the HCP will perform at least as well for solvated molecules. To be specific, we have estimated the influence of solvation on HCP accuracy by using the analytical linearized Poisson-Boltzmann (ALPB) approach,\textsuperscript{138} adapted to calculate the electrostatic potential near the molecular surface, and available in the GEM software.\textsuperscript{139,140} For this proof-of-concept analysis we use only one level of approximation for HCP, which is applicable to 595 biomolecular subunits out of the full sample set of 600 biomolecular structures. The error in surface potential is calculated with solvent dielectric $\epsilon = 80$ (water), and solute dielectric $\epsilon = 1$. The relative RMS error is found to be similar to the vacuum case analyzed above, which suggests that the findings from the preceding assessment of accuracy, in vacuum, should be similar to what one might expect to find in implicit solvent.
In the following section we perform the most relevant test of the HCP: test it directly in the context of a molecular dynamics simulation based on an implicit solvent model.

### 3.2.4.7 An MD implementation of the HCP

As a final proof-of-concept – to demonstrate stability of the HCP algorithm – it is tested in the context of a realistic molecular dynamics simulation. For this implementation the nucleic acid builder (NAB) MD component of the Amber tools software package is modified to incorporate the HCP. In the context of MD simulations, the NAB is a minimal version of the more comprehensive Amber molecular modeling software package. For example, NAB includes the spherical cutoff method but not the PME method.

A 10 ns simulation of a 36-residue protein using the HCP for computing long-range electrostatic interactions, is compared to two reference simulations: one using the exact all-atom (infinite cutoff) computation, and the other using the spherical cutoff method. The simulations use \( h_1 = 10 \, \text{Å} \) group level (level 1) threshold distance for the HCP and a 10 Å cutoff distance for the spherical cutoff method. The structure was protonated as described in Section 3.2.4.1. Similar to the spherical cutoff method, the 6-12 van der Waals interactions for the HCP are computed using only the atoms of components within the threshold distance, that is those that are treated exactly. The simulations use the Amber \( ff99SB \) force field. The sigmoidal distant dependent dielectric implicit solvent model is used along with temperature control using Langevin dynamics with a 50 ps\(^{-1} \) collision frequency. The integration time step is 1 fs. Default values are used for all other parameters. The conjugate gradient method with a restraint weight of 5.0 is used to first minimize the crystal structure. The minimized structures is then heated to 300 K over 10 ps with a restraint weight of 0.1. The resulting structure is then equilibrated at 300 K for 10 ps with a restraint weight of 0.01. The smoothing function was not implemented.

Figure 3.9 shows the energy, temperature and root mean square deviation (RMSD) of backbone atoms from the initial structure, as a function of simulation time. The results show that the HCP is in reasonable agreement with the exact all-atom computation for this simulation. The results also show that for this single-CPU simulation, the 1-charge and 2-charge HCP are 40% and 14% faster respectively than the spherical cutoff method, and that the 3\( \times \) speedup compared to the exact all-atom simulation is consistent with the speedup for the single point computation described earlier. Curiously, the stochastic collisions with the thermal bath were enough to “eliminate” the drift resulting from spurious center-of-mass motion due to the effects discussed in the earlier section. We stress that our goal here was only to demonstrate stability of the basic algorithm.

### 3.2.5 Conclusions

Molecular simulations are an important tool for biomolecular modeling. However, the computational cost of calculating long range electrostatic interactions limits the accessible time scales: a number of approximation methods have been developed to speed up these compu-
Figure 3.9: A 10 ns simulation of 1VII. Simulation using the HCP for long-range electrostatic computations, is compared to simulations using the exact all-atom computation and the spherical cutoff method. (a) RMS deviation of backbone atoms from the initial structure, (b) total energy, (c) temperature, and (d) potential energy, are shown as a function of simulation time. Data is plotted every 0.5 ns. Connecting lines are shown to guide the eye.
tations. The most commonly used of these are the spherical cutoff and the particle mesh Ewald (PME) methods. The PME method, which scales as $O(N \log N)$ with the number of particles, $N$, is the de facto “industry standard” for explicit solvent calculations due to its accuracy that far exceeds that of the spherical cut-off method. Although widely used, the PME has several drawbacks, including inapplicability to many practical implicit solvent models such as the generalized Born approximation. We present here an $O(N \log N)$ method, the hierarchical charge partitioning (HCP) approximation, which retains the algorithmic simplicity of the basic spherical cutoff method, can be multiple orders of magnitude faster than an exact all-atom computation, and approaches the accuracy of the much more sophisticated PME method. The HCP exploits the natural partitioning of biomolecular structures into a hierarchical set of structural components, such as molecular complexes, DNA or RNA subunits, and amino and nucleic acid groups. By systematically approximating the charge distribution within larger but more distant components by fewer point charges than in the original distribution, the HCP needs to compute fewer pairwise interactions than the much larger number needed by the corresponding exact all-atom computation which scales poorly as $O(N^2)$. To understand the suitability of this simple idea for molecular dynamics, we have systematically explored the accuracy and speed of the HCP. We tested the approximation on a sample set of 600 representative biomolecular structures ranging in size from approximately 300 to 3 million atoms and ranging in charge from -20,800 to +236 a.u. For a mini-protein of ~ 300 atoms, a proof-of-concept implementation of the HCP is at least a factor of 2 faster than the corresponding exact all-atom computation, and can be multiple orders of magnitude faster for larger structures, with the speed-up approaching three orders of magnitude for the chromatin fibre consisting of approximately 3 million atoms. To assess the accuracy of the HCP, we compared it with two other popular methods for speeding-up computations of long range interactions in molecular simulations: the spherical cut-off and the particle mesh Ewald (PME) methods. To make such comparisons between very different methods meaningful, we have assumed parameter settings typical of biomolecular simulations. Our primary metric is the root-mean-square error of the per atom force, estimated relative to the corresponding exact computation, and averaged over groups of molecules of similar size. For all the size groups, the HCP is considerably more accurate than the spherical cutoff method. Compared to the the particle mesh Ewald (PME) method, the HCP is as accurate or nearly as accurate for structures larger than about 2-3 thousand atoms, and can be even more accurate than the PME for very large structures ($N > 10^4$ atoms). Perhaps the most critical advantage of the HCP relative to the PME method is that it can be straightforwardly applied to speed-up analytical pairwise implicit solvent models (e.g. the GB, the distance-dependent dielectric, etc.) to further enhance their advantages for biomolecular simulations.

Use of the natural hierarchical partitioning of biomolecules, which is largely preserved during dynamics, offers several important advantages. The natural partitioning provides the basis for an algorithmically simple yet accurate approach for approximating the charge distribution of biomolecular structures. This hierarchical nature of the partitioning is exploited by the top down HCP algorithm to significantly reduce computational costs. The HCP offers significant additional savings for proteins: 80% of the amino acids in proteins typically have a zero net charge. Thus, within the simplest “1-charge” HCP that approximates the actual charge distribution of a structural component by a single charge equal to the total charge of
the component, there is no computational cost for approximating the distant electrostatic contributions of the net neutral groups. Finally, sets of atoms that make up each component do not change during dynamics thus eliminating one possible source of discontinuity.

The HCP is a new method, although it may appear to have similarities to some other approximations also aimed at increasing the speed of computing the long-range interactions. For example, similar to the fast multipole method, the HCP replaces an exact charge distribution with an approximate one to estimate the contribution due to a distant group of charges. However, unlike the fast multipole method based on an artificially imposed regular mesh, the HCP uses the natural hierarchical organization of biomolecular structures for grouping charges. This results in a much simpler algorithm. In addition, the grouping of charges do not need to be recomputed during dynamics since within the HCP there is no movement of atoms between components, unlike the movement of atoms across artificial spatial boundaries in the fast multipole method, which is a potential source of discontinuities. Also, we speculate that the HCP is more accurate in the near-field, where the multipole expansion is not intended to work. Consequently, the same level of accuracy can presumably be achieved at a smaller computational cost. Implementation of the HCP in existing MD codes is also more straightforward since the mathematical form of the existing energy functions need not be changed. For electrostatic computations, the HCP approximates biomolecular components (e.g. groups, subunits and complexes), with a small number of point charges. This may appear similar to some coarse grain approximation methods where groups (e.g. united residue method) and subunits (e.g. segment chain method) are replaced by a single particle. However, unlike these methods that calculate forces on coarse-grained components treated as single particles, the HCP still calculates the force on each individual atom. Thus, another critical advantage for the HCP is that it does not require additional approximations to map all-atom force fields onto a less accurate coarse grained description. The HCP uses the same force field as the underlying the exact all-atom computation.

The speed and accuracy of the HCP suggests that it may be a practical option for molecular dynamics simulations. As a proof-of-concept, the HCP has been implemented in the NAB molecular dynamics software and tested in a 10 ns long implicit solvent simulation of a 36-residue protein. For this simulation, the HCP produced a stable MD trajectory in reasonable agreement with the reference simulation without the approximation. The simulation also shows that the HCP can be faster than the spherical cutoff method. This indicates that the HCP may be an attractive alternative to the spherical cutoff and PME methods for molecular dynamics simulations, particularly those based on implicit solvent models. However, more extensive testing is needed before the method can be routinely applied in practice. There are also a number of refinements to the HCP that were identified during this study that should be considered in a subsequent study. These refinements include calculating the approximate point charges by fitting to potential distribution in the solvent space, defining threshold distances as a function of component size and charge, and refining the algorithm for placing the approximate point charges of components.

The HCP approximation presented here is designed to speed up electrostatic computations in classical molecular dynamics simulations that use non-polarizable force fields. With some refinements we expect that the HCP can be extended to other types of calculations such as the
hybrid quantum mechanical/molecular mechanical (QM/MM) simulations and simulations using polarizable force fields. The general idea is to treat interactions that matter the most, exactly, while using the HCP where the sensitivity to inaccuracies in the estimates of the interactions is lower. For example, for QM/MM models the HCP can be applied outside of the QM region.
3.3 HCP for implicit solvent MD

3.3.1 Introduction

Realistic simulations require that the biomolecular structure be immersed in a solvent, typically water with ions. Implicit solvent models, such as the generalized Born (GB) approximation, analytically represent the solvent as a continuum. An important benefit of implicit solvent simulations is that conformational space is sampled faster due to the reduction of solvent viscosity. Other benefits include instantaneous dielectric response from the solvent due to changes in solute charge state, and the elimination of “noise” in the energy landscape due to small variations in solvent structure. Consequently, implicit solvent models are often used for applications where it is important to explore a large number of conformational states, such as for protein folding, replica exchange, and docking simulations. However, the functional form for the most widely used practical implicit solvent model for MD, the GB model, scales as $\sim n^2$, where $n$ represents the number of solute atoms only. ($n < N$ where $N$ refers to the total number of atoms, including solute and solvent atoms, used for explicit solvent computations, while $n$ refers to the number of solute atoms only, used in the implicit solvent computations). One approach for reducing computational cost is to apply the spherical cutoff concept to the GB implicit solvent model, i.e., ignore interactions and computations involving atoms beyond a cutoff distance. We refer to this approach as the cutoff-GB method. Such an approach can reduce computational cost to $\sim n \log n$. However, the cutoff-GB may suffer from the same shortcomings as the spherical cutoff method, such as spurious forces and artificial structures around the cutoff distance. Although there are studies based on the successful use of the cutoff-GB method, we are not aware of a large scale systematic study that examines the effect of the cutoff on the accuracy of the GB model. To the best of our knowledge, the GB model has not been used with the PME or the fast multipole methods, most likely because the functional form of the GB model does not easily lend itself to the Ewald transformation used by the PME method or the multipole expansion used by the fast multipole method.

We present here an $\sim n \log n$ GB approximation that retains the simplicity of the cutoff-GB approximation, while in most cases being more accurate for the set of test structures considered here. Moreover, our testing demonstrates that the method presented here more accurately reproduces important characteristics of dynamics compared to the cutoff-GB method. Our approach is based on the hierarchical charge partitioning (HCP) approximation developed by us previously. To approximate long-range electrostatic interactions, the HCP uses the natural organization of biomolecules into multiple hierarchical levels of components, as illustrated in Fig. 3.1 – atoms (level 0), nucleic and amino acid groups (level 1), protein, DNA and RNA subunits (level 2), complexes of multiple subunits (level 3), and higher level structures such as fibres and membranes. The charge distribution for components above the atomic level are approximated by a much smaller number of charges. For components that are distant from the point of interest, these approximate charges are used in the computation of electrostatic interaction, while the atomic charges (level 0) are used for nearby components (Fig. 3.10). The greater the distance from the point of interest, the larger (higher level)
is the component used in the approximation of electrostatic interactions. In our previous study, we have shown that this approximation scales as \( \sim n \log n \) for biomolecular structures. The HCP concept is used here to reduce the computational cost of each of the three \( \sim n^2 \) computations in the GB model – the computation of electrostatic vacuum energy, solvation energy, and the so-called effective Born radii – to \( \sim n \log n \).

Figure 3.10: The HCP threshold distance. For the first level of approximation shown here, groups within the threshold distance from the point of interest are treated exactly using atomic charges (level 0), while groups beyond the threshold distance are approximated by a small number of charges (level 1). The distance to a group is computed from the point of interest to the geometric center of the group.

The remainder of this section is organized as follows. In the Methods section, we briefly review the GB implicit solvent model, and describe how the HCP concept is applied to the implicit solvent GB model (HCP-GB). The HCP-GB method was tested using a set of representative biomolecular structures ranging in size from 632 atoms to 3 016 000 atoms with absolute total charge ranging from 1 to 21 424 \( e \). The accuracy, speedup, dynamics and conservation of momentum and energy are discussed in the Results section. In the Conclusion section we summarize our finding and discuss the applicability of HCP-GB to practical MD problems.

### 3.3.2 Methods

A short description of the GB implicit solvent model is included below, followed by a detailed description of how the HCP approximation is used to reduce the computational cost of the GB model from \( \sim n^2 \) to \( \sim n \log n \). Also included in this section is a description of the structures and protocols used for testing the HCP-GB method.
3.3.2.1 Generalized Born (GB) implicit solvent model

The electrostatic energy of a system $E^{\text{elec}}$ in the presence of a solvent can be approximated by the GB implicit solvent model\(^{151}\) as:

\[
E^{\text{elec}} = E^{\text{vac}} + E^{\text{solv}}
\]

\[
E^{\text{vac}} = \sum_i^n \sum_{j>i}^n \frac{q_i q_j}{r_{ij}}
\]

\[
E^{\text{solv}} \approx -\frac{1}{2} \left( 1 - \frac{1}{\epsilon_w} \right) \sum_i^n \sum_{j}^n \frac{q_i q_j}{[r_{ij}^2 + B_i B_j e^{-r_{ij}^2/(4B_i B_j)}]^{1/2}}
\]

where $E^{\text{vac}}$ and $E^{\text{solv}}$ are the electrostatic vacuum and solvation energy, $\epsilon_w$ is the dielectric constant of the solvent, $q_i$ and $q_j$ are the charges of atoms $i$ and $j$, $r_{ij}$ is the distance between the atoms, and $B_i$ and $B_j$ are their effective Born radii. For the purpose of this work, we consider the following Coulomb field approximation for the effective Born radius, $B_i$, as implemented in NAB (or Amber):\(^{131}\)

\[
\frac{1}{B_i} \approx \frac{1}{R_i} - \frac{1}{4\pi} \int_{|r_{ik}|>R_i}^{\text{solute}} \frac{1}{|r_{ik}|^4} dV
\]

where $R_i$ is the intrinsic radius of charge $i$, $r_{ik}$ is the distance from $i$ to any point $k$ in the solute volume, and $\int_{|r_{ik}|>R_i}^{\text{solute}} dV$ is the volume integral over the volume occupied by the solute (the cavity formed in the solvent by the solute) excluding the volume of the atom $i$ itself.

3.3.2.2 HCP-GB - an $n \log n$ GB approximation

Note that the computation of electrostatic vacuum energy $E^{\text{vac}}$ in Eq. 3.10 and solvation energy $E^{\text{solv}}$ in Eq. 3.11 both scale as $\sim n^2$. In MD software that implement the GB implicit solvent model, such as Amber,\(^{131}\) analytical pairwise approximations for computing the effective Born radii $B_i$ in Eq. 3.12 also scale as $\sim n^2$, unless further approximations are made. The HCP concept is used to reduce the computational cost for each of these computations to $\sim n \log n$, as described below.

3.3.2.2.1 $n \log n$ approximation for electrostatic vacuum energy The previous HCP study\(^2\) describes in detail the $\sim n \log n$ approximation for computing electrostatic vacuum energy $E^{\text{vac}}$. The key concepts from the study are summarized here. Biomolecular structures are naturally organized into multiple hierarchical levels as illustrated in Fig. 3.1 for a chromatin fibre. Atoms are at the lowest level (level 0); groups of atoms form amino and nucleic acids (level 1); protein, DNA and RNA chains made up of these groups form subunits (level 2); multiple subunits form complexes (level 3); and multiple complexes join together to form larger structures such as fibres and membranes. The HCP approximates atomic charges within each of the components above level 0, by a much smaller number of charges (1 or 2). For the 1-charge approximation for a component, the approximate charge
is placed at the “center of charge” for the component with a charge value equal to the net charge of the component. For the 2-charge approximation the two approximate charges are placed at the “center of charge” of the positive and negative charges with charge values equal to the total positive and negative charges, respectively. The center of charge is calculated in a manner similar to center of mass when the total charge is nonzero.\(^2\) When the total charge is zero, the component does not contribute to the approximate computation and is ignored.

The HCP then uses these approximate charges for computing electrostatic interactions beyond predefined threshold distances (Fig. 3.10). For example, consider a structure consisting of four levels, 0 - 3, see Fig. 3.2. A separate threshold distance, \(h_1\), \(h_2\), and \(h_3\), is defined for level 1, 2 and 3, respectively. For complexes (level 3) farther than \(h_3\) from the point of interest, the approximate charges for the complex are used in the computation. Otherwise, for subunits (level 2) within the complex that are farther than \(h_2\), the approximate charges for the subunit are used in the computation. Otherwise, for groups (level 1) within the subunit that are farther than \(h_1\), the approximate charges for the group are used in the computation. Finally, individual atomic charges are used in the computations for charges within the level 1 threshold distance \(h_1\). This top-down algorithm results in \(\sim n \log n\) scaling based on assumptions generally consistent with realistic biomolecular systems. Consider a hypothetical structure consisting of \(n\) atoms such that, for any given atom, there are \(k\) atoms (level 0) within the level 1 threshold distance \(h_1\), \(k\) groups (level 1) between \(h_1\) and the level 2 threshold distance \(h_2\), \(k\) subunits (level 2) between \(h_2\) and the level 3 threshold distance \(h_3\), and so on. Such a structure can be represented as a hierarchical tree with each internal node representing a component with \(k\) nodes immediately below each internal node, and with a total of \(n\) leaf nodes representing the atoms. The computational cost of the HCP algorithm for such a structure scales as \(\sim n \log n\). For a more detailed description, refer to the previous HCP study.\(^2\) This previous study also showed that for the computation of electrostatic vacuum energy, \(E^{\text{vac}}\), in Eq. 3.10, the relatively simple HCP approximation can be comparable in accuracy to the more complex particle mesh Ewald (PME) method and more accurate than the simple spherical cutoff method.

**3.3.2.2.2 \(n \log n\) approximation for solvation energy** To reduce the computational cost of solvation energy, \(E^{\text{solv}}\), in Eq. 3.11 from \(\sim n^2\) to \(\sim n \log n\), we first define a component effective Born radius, \(B_c\), for components above the atomic level. The component effective Born radii, \(B_c\), are then used to approximate the contribution of distant components to the solvation energy of atom \(i\) instead of the effective Born radii, \(B_j\), of the individual atoms within these distant components.

To derive a simple functional form for the component effective Born radius, \(B_c\), we consider the limit of \(r_{ij}, r_{ic} \to 0\), where \(r_{ij}\) is the distance from atom \(i\) to atom \(j \in c\), and \(r_{ic}\) is the distance from atom \(i\) to component \(c\). Let \(E^{\text{solv}}_{ic}\) represent the contribution of component \(c\)
to the solvation energy of atom $i$.

$$E_{ic}^{solv} = -\frac{1}{2} \left( 1 - \frac{1}{\epsilon_w} \right) \sum_{j \in c} \frac{q_i q_j}{[B_i B_j]^{1/2}} e^{-r_{ij}/4B_i B_j}$$

(3.13)

$$\approx -\frac{1}{2} \left( 1 - \frac{1}{\epsilon_w} \right) \sum_{j \in c} \frac{q_i q_j}{[B_i B_j]^{1/2}} \quad (r_{ij} \to 0)$$

(3.14)

Also, $E_{ic}^{solv} = -\frac{1}{2} \left( 1 - \frac{1}{\epsilon_w} \right) \sum_{j \in c} \frac{q_i q_j}{[B_i B_j]^{1/2}}$ 

(3.15)

$$\approx -\frac{1}{2} \left( 1 - \frac{1}{\epsilon_w} \right) \frac{q_i q_c}{[B_i B_c]^{1/2}} \quad (r_{ic} \to 0)$$

(3.16)

$$\Rightarrow \frac{1}{B_c} \approx \left[ \frac{1}{q_c} \sum_{j \in c} \frac{q_j}{B_j^{1/2}} \right]^2$$

(3.17)

For the 1-charge HCP approximation, $q_c$ is the net charge of the component, and the sum in the above equations is over all the atoms in the component. For the 2-charge approximation, two separate component effective Born radii are computed, one for each of the two approximate charges, i.e., one for the positively charged atoms and another for the negatively charged atoms. In this case, $q_c$ represents the total positive or negative charge and the sum is over the positively or negatively charged atoms, respectively.

In this work, we also considered two alternatives to Eq. 3.17 for component effective Born radii. One approximation, by Archontis and Simonson, developed in the context of a coarse grain model, defines the equivalent of the component effective Born radius as the harmonic average of its constituent atomic Born radii weighted by the square of atomic charges. The resulting expression is similar to Eq. 3.17 except that the constituent atomic Born radii are weighted by atomic charges. Another approach is to use the analytical approximation for effective Born radii defined by Eq. 3.18 described below, with $i$ representing a component $c$ instead of an atom, and $j \neq i$ replaced by $j \notin c$. We examined these alternatives (results included in Sec. 3.3.4.10) and found that on average the approach described above by Eq. 3.17 is more accurate, although in some specific instances one of the other alternatives can be more accurate. We have therefore chosen to base all further analysis on Eq. 3.17, but note that future work may lead to better approximations.

3.3.2.2.3 $n \log n$ approximation for effective Born radii $B_i$ To compute the integral in Eq. 3.12, the Coulomb field approximation for effective Born radii $B_i$, we will consider here one commonly used approximation to Eq. 3.12. However, the main idea can be applied to any volume based approximation for computing effective Born radii. In the specific approximation considered here, the integral in Eq. 3.12 is computed over the volume occupied by individual atoms, ignoring overlaps between atoms and spaces between atoms that are inaccessible to the solvent. For the case where the atoms $i$ and $j$ do not overlap, this analytical approximation computes the effective Born radius $B_i$ as

$$\frac{1}{B_i} \approx \frac{1}{R_i} - \sum_{j \neq i} \left[ \frac{R_j}{2(r_{ij}^2 - R_j^2)} - \frac{1}{4r_{ij}} \log \frac{r_{ij} + R_j}{r_{ij} - R_j} \right]$$

(3.18)
The computation of effective Born radii using the above approximation scales as $\sim n^2$. The HCP approximation can be used to reduce the computational cost of the above equation to $\sim n \log n$, as follows. We define a component radius $R_c$ for a component $c$ which can be used to approximate the contribution of distant components to the effective Born radius of an atom, replacing the computations involving the individual atoms within the component. Then, using the HCP approach described in Sec. 3.3.2.2.1 above, the effective Born radius $B_i$ for atom $i$ can be approximated in $\sim n \log n$ computations.

To derive a simple expression for component radius $R_c$, we consider the limit of $r_{ij} \to r_{ic}$, where $r_{ij}$ is the distance from atom $i$ to atom $j \in c$, and $r_{ic}$ is the distance from atom $i$ to component $c$. For distant components $c$, let $B_{ic}$ be the contribution of the atoms $j \in c$, to the effective Born radius of $i$. $B_{ic}$ can be approximated by a truncated Taylor series expansion of Eq. 3.18 as

$$\frac{1}{B_{ic}} = C \sum_{j \in c} \frac{R_j^3}{r_{ij}^4} + \text{higher order terms} \quad (3.19)$$

$$\approx C \sum_{j \in c} \frac{R_j^3}{r_{ic}^4} \quad (r_{ij} \to r_{ic}) \quad (3.20)$$

Also,

$$\frac{1}{B_{ic}} = C \frac{R_c^3}{r_{ic}^4} \quad (3.21)$$

$$\Rightarrow R_c^3 \approx \sum_{j \in c} R_j^3 \quad (3.22)$$

where $C$ is a constant. Interestingly, $R_c$ is the radius of a sphere with the same volume as the sum of volumes of its constituent atoms, which is what one may intuitively expect. On the basis of the simple and intuitive nature of the expression for $R_c$, we conjecture that the form of Eq. 3.22 is independent of the specifics of Eq. 3.18. For a distant component $c$, the component radius $R_c$ is used in Eq. 3.18 in place of $R_j$ and $r_{ic}$ in place of $r_{ij}$ for atoms $j \in c$. Higher level components are used in the computation of effective Born radii $B_i$ for more distant components such that computational cost scales as $\sim n \log n$, as described in Sec. 3.3.2.2.1 above.

**3.3.2.2.4 $n \log n$ approximation for solvation forces** The solvation force on an atom $i$, $F_{i}^{\text{solv}}$, is computed as the derivative of the solvation potential $\phi_i^{\text{solv}}$ using the chain rule,
as follows:

\[
\phi_{i}^{\text{solv}} = \sum_{j \neq i} \phi_{ij}^{\text{solv}}
\]

\[
\phi_{ij}^{\text{solv}} = -\frac{1}{2} \left(1 - \frac{1}{\epsilon_w}\right) \frac{q_i q_j}{[r_{ij}^2 + B_i B_j e^{-r_{ij}^2/4B_i B_j}]^{1/2}}
\]

\[
F_{i}^{\text{solv}} = \sum_{j \neq i} F_{ij}^{\text{solv}}
\]

\[
F_{ij}^{\text{solv}} = -q_i \partial \phi_{ij}^{\text{solv}} / \partial r_{ij}
\]

\[
\frac{\partial B_i}{\partial r_{ij}} = B_i^2 \sum_{j \neq i} \left[ \frac{-R_j r_{ij}}{(r_{ij}^2 - R_j^2)^2} + \frac{1}{4r_{ij}^2} \log \frac{R_j + r_{ij}}{R_j - r_{ij}} + \frac{R_j}{2r_{ij}(R_j^2 - r_{ij}^2)} \right]
\]

where \(\phi_{ij}^{\text{solv}}\) is the solvation potential contribution of atom \(j\) at atom \(i\), \(F_{ij}^{\text{solv}}\) is the corresponding force contribution, \(\epsilon_w\) is the dielectric constant of the solvent, \(r_{ij}\) is the distance between atoms \(i\) and \(j\), \(q_i, q_j\) are the atomic charges, and \(R_i, R_j\) are the intrinsic radii. Here, \(\partial B_i / \partial r_{ij}\) is the derivative of the effective Born radii (Eq. 3.18). For distant components, component intrinsic radii and component effective Born radii are used in the above equations instead of the atomic intrinsic radii and atomic effective Born radii. Using the HCP approach described in Sec. 3.3.2.2.1 above, solvation forces can then be approximated in \(\sim n \log n\) computations.

### 3.3.3 Test setup

To assess performance of the HCP-GB method in the context of molecular dynamics, we implemented the method in NAB, the open source molecular dynamics (MD) software in AmberTools v1.3\textsuperscript{152} The HCP implementation in NAB is scheduled to be released with AmberTools v1.5, for general use. In some sense, NAB is a minimal version of the production Amber MD software\textsuperscript{131} and is particularly well suited for experimentation unlike the highly optimized but also more complex production version. NAB however does use the same force fields and implements the same GB implicit solvent methods and options as the production Amber code. For the purpose of this study, we used the commonly used OBC GB model (IGB=5 in Amber\textsuperscript{153}).

Performance in both accuracy and speed was evaluated relative to the reference GB computation without any additional approximations (reference GB). We also compared the HCP-GB method to the same GB implicit solvent model with a spherical cutoff (cutoff-GB). The cutoff-GB method ignores all interactions beyond a cutoff distance for the computation of electrostatic energy and effective Born radii in Eq. 3.9, 3.10, 3.11, and 3.18. Our previous
study\textsuperscript{2} had compared the electrostatic vacuum energy and forces computed by the HCP method to the particle mesh Ewald (PME) explicit solvent method. However, to the best of our knowledge, the GB implicit solvent model has not been implemented for the PME method in readily available molecular dynamics software. Therefore, a similar comparison for the HCP-GB method was not performed here.

The HCP-GB method was tested on a set of eight representative biomolecular structures ranging in size from 632 atoms to 3 016 000 atoms with absolute total charge ranging from 1 to 21 424 e (Table 3.1). The H++ server (https://biophysics.cs.vt.edu/H++) was used to add missing hydrogens to these structures.\textsuperscript{48}

The HCP threshold distances were chosen such that, for a given atom within a given test structure, the exact atomic computation (level 0) is used for interactions with other atoms within its own and nearest neighboring groups (level 1) as illustrated in Fig. 3.10. To satisfy this condition, threshold distances $h_l$ are calculated as $h_l = R_{l,\text{max}} + 2 \times R_{1,\text{max}}$ where $l$ is the HCP level, $R_{l,\text{max}}$ is the maximum component radius at level $l$, and $R_{1,\text{max}}$ is the maximum group (level 1) radius, for a given structure. The HCP threshold distances thus calculated for each of the test structures are shown in Table 3.1. These are the suggested conservative defaults for these and other similar structures. The HCP-GB level 1 threshold distance for a given structure is also used as the cutoff distance for the cutoff-GB computations. Unless stated otherwise, these threshold and cutoff distances were used for all of the testing described in the Results section.

| Structure                          | PDB ID | Size (atoms) | |Charge| | Cutoff dist (Å) | Threshold dist (Å) |
|-----------------------------------|--------|--------------|-----------------|-------|-----------------|-------------------|
| 10 bp B-DNA fragment              | 2BNA   | 632          | 18              | 21    | 21 n/a          | n/a               |
| Immunoglobulin binding domain     | 1BDD   | 726          | 2               | 15    | 15 n/a          | n/a               |
| Ubiquitin                         | 1UBQ   | 1231         | 1               | 15    | 15 n/a          | n/a               |
| Thioredoxin                       | 2TRX   | 1654         | 5               | 15    | 15 n/a          | n/a               |
| Nucleosome core particle          | 1KX5   | 25101        | 133             | 21    | 21 90           | n/a               |
| Microtubule sheet                 | *      | 158016       | 360             | 15    | 15 48           | n/a               |
| Virus capsid                      | 1A6C   | 475500       | 120             | 15    | 15 66           | n/a               |
| Chromatin fibre                   | **     | 3016000      | 21424           | 21    | 21 90 169       |                   |

Table 3.1: List of representative structures used for testing. Unless stated otherwise, the cutoff and threshold distances listed here were used for all testing. * The microtubule sheet was constructed as described in Wang and Nogales.\textsuperscript{154} ** The chromatin fibre was constructed as described in Wong et. al.\textsuperscript{125}

Four metrics were used to measure the accuracy of the approximate methods: relative error in electrostatic energy (relative energy error) $Err^E$, for vacuum, solvation, and net electrostatic energy, and relative RMS error in electrostatic force (relative force error) $Err^F$, calculated

81
as

\[
Err^E = \frac{|E_{approx} - E_{ref}|}{E_{ref}} \quad (3.29)
\]

\[
Err^F = \frac{Err^{rms}}{F_{avg}} \quad (3.30)
\]

\[
Err^{rms} = \left( \sum_{i=1}^{n} |F_{approx}^i - F_{ref}^i|^2 / n \right)^{1/2} \quad (3.31)
\]

\[
F_{avg} = \frac{\sum_{i=1}^{n} |F_{ref}^i|}{n} \quad (3.32)
\]

where \( E_{approx} \) is the energy calculated using an approximation, \( E_{ref} \) the energy calculated using the reference GB computation without cutoffs or the use of HCP, \( Err^{rms} \) the root-mean-square (RMS) error in force for the atoms in a given structure, \( F_{avg} \) the average force, and \( F_{approx}^i \) and \( F_{ref}^i \) the force on atom \( i \) calculated using the approximate and reference GB computations, respectively.

Speedup was measured as CPU time for the reference (no cutoff) GB computation divided by the CPU time for the approximation tested. All testing was conducted on Virginia Tech’s System X computer cluster (http://www.arc.vt.edu) consisting of 1100 dual core 2.5 GHz PowerPC 970FX processors with 4 GB of RAM, running the Apple Mac OS X 10.3.9, and connected by 10 Gbps InfiniBand switches. Where possible, testing was performed using a single CPU (a single core of the dual core processor) to reduce the potential variability due to interprocessor communication. However, due to the large memory requirements for the neighbor list used by the cutoff-GB method, it was not possible to run the cutoff-GB computation for structures larger than 200,000 atoms using a single CPU in the test environment described above. Therefore 16 CPUs were used for the 475,500 atom virus capsid and 128 CPUs for the 3,016,000 atom chromatin fibre. For comparison on an equal footing, the reference GB computations and the HCP-GB computations were also performed with the same number of CPUs. When multiple CPUs were used the CPU time for the longest running CPU was used to calculate speedup. To limit the run time for the reference GB computation to a few days, speedup was calculated for 1000 iterations of MD for structures with < 10000 atoms, 100 iterations for structures with 10000 – 1000000 atoms and 10 iterations for the structure with > 1000000 atoms. To make the results representative of typical simulations involving much larger numbers of iterations, the CPU time excludes the time for loading the data and initialization prior to starting the simulation. Note that the speedup may vary with the computing system characteristics, such as interprocessor communication network,

\[ ^2 \text{NAB, and the production Amber software, do not explicitly include a “no-cutoff” option, instead the no-cutoff computation is performed by using the cutoff method with the cutoff distance set to a value greater than the structure size (large-cutoff), e.g. 999. This approach requires a large amount of memory for the neighbor list used by the cutoff method, even though the list is unnecessary in this case since it always contains all the atoms. Due to the large memory requirement, structures larger than 200,000 atoms require a larger number of processors (> 128) than was readily available on the system used. Therefore we implemented a no-cutoff option in NAB that does not use a neighbor list. Since a neighbor list does not need to be computed, the no-cutoff option is faster than the large-cutoff approach, and the speedup results reported here are somewhat lower than what would have been obtained using the large-cutoff approach available in NAB.} \]
number of processors used, processor architecture, memory configuration, etc. A detailed analysis of the effect of these characteristics on speedup is beyond the scope of this study, which focuses on the algorithm.

The following parameters and protocol were used for the simulations, unless otherwise stated. The threshold distances used are listed in Table 3.1. 6-12 van der Waals interactions for the HCP-GB were computed using only the atoms that are within the level 1 threshold distance, i.e., atoms that are treated exactly. The simulations used the Amber ff99SB force field. Langevin dynamics with a collision frequency of 50 ps\(^{-1}\) (appropriate for water) was used for temperature control, a surface-area dependent energy of 0.005 kcal/mol/Å\(^2\) was added, and an inverse Debye-Hückel length of 0.125 Å\(^{-1}\) was used to represent a 0.145 M salt concentration. A 1 fs time step was used for the simulation with the nonbonded neighbor list being updated after every step. Note that updating the nonbonded neighbor list less frequently will improve the speedup of the cutoff-GB method; however, the speedup of the HCP-GB method can also be improved similarly by updating component radii and charges less frequently. For simplicity and for comparison on an equal footing, the non-bonded neighbor list and component radii and charges are updated after every step. Default values were used for all other parameters. The simulation protocol consisted of five stages. First, the starting structure was minimized using the conjugate gradient method with a restraint weight of 5.0 kcal/mol/Å\(^2\). Next, the system was heated to 300 K over 10 ps with a restraint weight of 1.0 kcal/mol/Å\(^2\). The system was then equilibrated for 10 ps at 300 K with a restraint weight of 0.1 kcal/mol/Å\(^2\), and then for another 10 ps with a restraint weight of 0.01 kcal/mol/Å\(^2\). Finally, all restraints were removed for the production stage.

3.3.4 Results and discussion

We examined a number of characteristics of the HCP-GB method that are important for molecular dynamics – accuracy, speed, dynamics, and conservation of energy and momentum, which are discussed below.

3.3.4.1 Accuracy

Fig. 3.11 shows the accuracy for the 1- and 2-charge HCP-GB methods compared to the cutoff-GB method. For the test structures considered here, the values of the two components of electrostatic interactions – vacuum and solvation energies – as calculated by the HCP-GB method are significantly more accurate than that of the cutoff-GB method (Fig. 3.11 (a) and (b)). For the net electrostatic energy and force, the relative improvement provided by the HCP-GB is significant for the smaller structures and decreases with structure size (Fig. 3.11(c) and (d)). For the largest structure considered here – the 3 million atom chromatin fibre – the relative energy error for the HCP-GB method is slightly higher than that of the cutoff-GB method. Preliminary analysis suggests that the larger HCP-GB error for the chromatin fibre may be due to the negligible contribution of very distant components to the net energy computation, even though the individual vacuum and solvation components may be large. Small errors in the estimation of these individual components can result in
a large relative error in net electrostatic energy. Thus, ignoring the contribution of these distant components, as the cutoff-GB method does, may actually decrease the error in total energy as defined by the above metrics. However, as our examination of key characteristics of dynamics in Sec. 3.3.4.6 below shows, the single point net force and net energy error metrics presented above are too crude to unambiguously differentiate between the expected performance of the cutoff-GB and HCP-GB methods in the context of molecular dynamics. For example, one can expect the cutoff scheme to neglect a roughly equal number of pairwise interactions of roughly equal magnitude but of opposite sign. The resulting cancellation of error in total electrostatic energy can be deceptive. As we shall see later in Sec. 3.3.4.6, neglect of charge-charge interactions clearly manifests itself by producing artifacts in dynamics.

### 3.3.4.2 Speedup

Fig. 3.12 shows that the speedup for the 1-charge HCP-GB and cutoff-GB methods are comparable,\(^3\) while the 2-charge HCP-GB is slower. Surprisingly, the cutoff-GB method is slower than the 1-charge HCP-GB method for the 3-million atom chromatin fibre. We speculate that this is because the NAB implementation of the cutoff method does not scale well with system size due to the additional memory accesses required for the large neighbor list used by the method.

As noted earlier, unlike the production \texttt{pmemd} module of Amber 8, NAB is not highly optimized. However, on the basis of the run times for a 0.1 ns simulation of the nucleosome core particle (1KX5), compared to an equivalent simulation by Ruscio et. al.,\(^{155}\) we estimate that NAB v1.3 is only about 1.5 times slower than the production \texttt{pmemd} module of Amber 8 on Virginia Tech’s System X computer cluster described above.

### 3.3.4.3 Tradeoff between speed and accuracy

For a given structure, the speed and accuracy of the HCP-GB method depends primarily on two parameters: the number of charges used to approximate the components and the threshold distances. As seen in Fig. 3.11 and 3.12, on the basis of net energy and force metrics, the 2-charge approximation is more accurate but slower than the 1-charge approximation. Fig. 3.13 shows that increasing the threshold distance improves accuracy but reduces speed. However, as our analysis of key characteristics of dynamics (Sec. 3.3.4.6) shows, the single-point error metrics, based on net energy or force, used above may not provide a complete measure of correctness in the context of molecular dynamics. The optimal choice of parameters depends on the structure and problem under consideration. For the purpose of this study, we have chosen conservative threshold distances (Table 3.1) such that for any given atom, atoms within it’s own group and immediately neighboring groups (level-1) are treated exactly, as described in Sec. 3.3.3. It is possible that shorter than default threshold

---

\(^3\)The average speedup for the 7 structures tested here was 82\(\times\) for the cutoff-GB, 85\(\times\) for the 1-charge HCP-GB, and 36\(\times\) for the 2-charge HCP-GB methods.
Figure 3.11: Accuracy of the HCP-GB and cutoff-GB methods. Accuracy is computed as the relative error in (a) vacuum, (b) solvation, and (c) net electrostatic energy, and (d) relative RMS error in electrostatic force, relative to the reference GB computation without cutoffs. Connecting lines are shown to guide the eye.

Distances may be acceptable for specific applications, but we suggest that the decision to use shorter threshold distances be made on a case-by-case basis. For example, the 2-charge HCP-GB simulation of immunoglobulin binding domain (1BDD) remains stable when the level 1 threshold distance is reduced from the recommended 15 Å to 10 Å, but the protein quickly unfolds when the threshold distance is further reduced to 5 Å.
Figure 3.12: Speedup for the HCP-GB and cutoff-GB methods. Speedup is calculated relative to the reference GB computation without cutoffs. Threshold and cutoff distances used for the different structures are listed in Table 3.1. Connecting lines are shown to guide the eye.

Figure 3.13: HCP-GB: Tradeoff between accuracy and speed. (a) Accuracy and (b) speed for the 158 016 atom microtubule structure. Cutoff and level 1 threshold distances are varied from 10 Å to 20 Å. Level 2 threshold distance is 48 Å. Connecting lines are shown to guide the eye.

3.3.4.4 Accuracy of the HCP approximation for effective Born radii

We tested the HCP based approximation for effective Born radii on a typical structure used in this context, thioredoxin (2TRX). Fig. 3.14 shows that for this structure, the HCP-GB
approximation with a threshold distance of 15 Å is slightly more accurate than a cutoff based approximation with a 15 Å cutoff distance. The overall RMS error in effective Born radii relative to the reference GB computation without cutoffs is 0.0058 Å for the 1-charge HCP-GB, 0.0017 Å for the 2-charge HCP-GB, and 0.0557 Å for the cutoff-GB method.

Both the HCP-GB and the cutoff-GB introduce two sources of error into the total electrostatic energy, relative to the no-cutoff reference. One is the approximations to effective Born radii, and the other is the approximations to the electrostatic interactions. The relative impact of these two sources is shown in Table 3.2. Clearly, for the spherical cutoff, the error in effective Born radii is the dominant source of error in the total electrostatic energy. The use of the HCP-GB approximation for effective Born radii can reduce this error by an order of magnitude. Whether these errors in effective Born radii will have a material impact on dynamics depends on, among other factors, the relative magnitude of the errors inherent in the approximation used to compute effective Born radii in the reference model. Nevertheless, the improvement in the accuracy of effective Born radii using the HCP approximation, compared to the cutoff approximation, comes at little or no additional cost and should therefore be used instead of the cutoff approximation.

### 3.3.4.5 Stability in MD simulations

To test the stability of the HCP-GB algorithm, we ran 50 ns MD simulations of the immunoglobulin binding domain (1BDD), ubiquitin (1UBQ), a 10 base-pair fragment of B-DNA (2BNA), and thioredoxin (2TRX). Fig. 3.15 shows the backbone RMS deviation from the crystal structure for the simulations, which are summarized in Table 3.3. These results sug-
Table 3.2: Effect of Born radii approximations on relative RMS error. Relative RMS error in total electrostatic energy for thioredoxin (2TRX) due to different approximations of effective Born radii. RMS error is calculated relative to the reference GB computation. Cutoff and level 1 threshold distances of 15 Å were used for these computations.

<table>
<thead>
<tr>
<th>Effective Born radii approximated using</th>
<th>Energy calculated with cutoff</th>
<th>Energy calculated with no cutoff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cutoff</td>
<td>0.50 %</td>
<td>0.49 %</td>
</tr>
<tr>
<td>1-q HCP</td>
<td>0.05 %</td>
<td>&lt; 0.01 %</td>
</tr>
</tbody>
</table>

suggest that, the trajectory for the cutoff-GB and HCP-GB methods are generally in reasonable agreement with the reference GB simulation. For 1BDD the 1-charge HCP-GB trajectory shows RMS deviations similar to the cutoff-GB trajectory, but substantially larger than the 2-charge HCP-GB or the reference GB trajectories. This example emphasizes how subtle errors in charge-charge interactions can result in qualitatively different conformational dynamics. On a practical level, it suggests that the 1-charge HCP-GB may not be appropriate for the simulation of small flexible structures, such as 1BDD, where small inaccuracies in the potential can lead to large structural deviations over the course of the trajectory. For such structures we recommend the 2-charge HCP-GB.

The above simulations were run with a Langevin collision frequency of 50 ps$^{-1}$ for thermal coupling. We also performed, for the same set of structures, 10 ns simulations with the thermal coupling reduced to 0.01 ps$^{-1}$ (results not shown). As expected, these simulations resulted in an enhanced sampling of conformational space, as was seen by more frequent excursions in RMS space. The weak Langevin coupling simulations were in general agreement with the simulations that used strong Langevin coupling. For example, for 1BDD the 2-charge HCP-GB and the reference GB simulations exhibited similar RMS deviations from the starting structure, while the 1-charge HCP-GB and cutoff-GB resulted in much higher RMS deviations towards the end of the respective trajectories.

Table 3.3: RMS deviation for 50 ns MD simulations. RMS deviation is calculated for backbone heavy atoms. The trajectory is sampled every 10 ps. Averages are for the last 40 ns of the 50 ns simulations. Standard deviation is computed as $\sqrt{\sum_i (RMS_i - \mu)^2 / s}$, where $RMS_i$ is the RMS deviation for the $i^{th}$ sample, $\mu$ is the average RMS deviation, and $s$ is the number of samples.

<table>
<thead>
<tr>
<th>PDB ID</th>
<th>reference GB</th>
<th>cutoff-GB</th>
<th>1-charge HCP-GB</th>
<th>2-charge HCP-GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1BDD</td>
<td>2.64 ± 0.45</td>
<td>3.69 ± 0.88</td>
<td>3.92 ± 1.66</td>
<td>2.72 ± 0.48</td>
</tr>
<tr>
<td>1UBQ</td>
<td>2.29 ± 0.37</td>
<td>2.39 ± 0.29</td>
<td>2.32 ± 0.33</td>
<td>2.22 ± 0.36</td>
</tr>
<tr>
<td>2BNA</td>
<td>2.24 ± 0.30</td>
<td>2.24 ± 0.30</td>
<td>2.23 ± 0.31</td>
<td>2.25 ± 0.31</td>
</tr>
<tr>
<td>2TRX</td>
<td>1.60 ± 0.33</td>
<td>1.41 ± 0.23</td>
<td>1.50 ± 0.23</td>
<td>1.67 ± 0.19</td>
</tr>
</tbody>
</table>
Figure 3.15: RMS deviation for 50 ns MD simulations. RMS deviation is calculated for backbone heavy atoms relative to the starting structure. The trajectory is sampled every 1 ns. Connecting lines are shown to guide the eye.

### 3.3.4.6 Detailed characteristics of simulation dynamics

An important qualitative difference between the HCP-GB method and the cutoff-GB method is that the cutoff-GB method completely ignores the effect of all charges beyond the cutoff distance, while the HCP-GB method approximates the effect of distant charges. We believe that ignoring these distant charges can, under many circumstances, lead to qualitatively different, and incorrect, results. Consider for example the RMS fluctuation in the position of residues – a characteristic of internal dynamics of the structure. To quantify the overall
difference in fluctuation for all residues compared to the reference GB simulation, we compute the RMS difference in RMS fluctuation for the 50 ns simulation of the four structures described in the “Stability” section above. For the structures tested here, the RMS difference in fluctuation (Table 3.4) indicates that on average both the 1-charge and 2-charge HCP-GB simulations are in better agreement with the reference GB simulation than the cutoff-GB method. The differences in RMS fluctuation from the 50 ns simulation of thioredoxin are highlighted in Fig. 3.16.

Similarly, consider the $\chi_1$ angles for the functionally important CYS-32 of thioredoxin and THR-7 of ubiquitin (Fig. 3.17). The $\chi_1$ angle for CYS-32 flips between approximately $-180^\circ$ and $+60^\circ$ during the 2-charge HCP-GB simulations as does the “correct” reference GB simulation. Whereas, for the 1-charge HCP-GB and cutoff-GB methods, the angle stays at approximately $-180^\circ$. And the $\chi_1$ angle for THR-7 stays around approximately $60^\circ$ during the reference GB and the HCP-GB simulations, whereas for the cutoff-GB simulation, the angle flips briefly between approximately $-60^\circ$ and $+60^\circ$. To quantify the overall difference in the distribution of $\chi_1$ and $\chi_2$ angles we computed the RMS difference in the distribution compared to the reference GB simulation. The RMS difference in the distribution of $\chi_1$ and $\chi_2$ angles for 50 ns simulations of the four structures described in the “Stability” section above indicates that on average the 1- and 2-charge HCP-GB simulations are in better agreement with the reference GB simulation than the cutoff-GB simulations (Table 3.4).

To further examine the effect of ignoring distant charges we ran a 10 ns simulation of a 30 base-pair DNA strand with the same setup as described in Sec. 3.3.3 above, but without the salt. Fig. 3.18 shows the distribution of distances between terminal base pairs. The distribution shows that for the cutoff-GB method, with an average end-to-end distance of 89 Å, the structure is more flexible than for the reference GB or the 1 and 2-charge HCP-GB methods, with average end-to-end distances of 103, 105 and 100 Å respectively. This
### Table 3.4: Detailed characteristics of simulation dynamics. Results from 50 ns simulations of immunoglobulin binding domain (1BDD) ubiquitin (1UBQ), B-DNA (2BNA), and thioredoxin (2TRX). RMS difference was calculated relative to the reference GB simulation. The trajectory was sampled every 10 ps. $\chi$ angles do not apply to the DNA strand 2BNA. A bin size of $10^\circ$ was used for calculating the distribution of $\chi$ angles.

<table>
<thead>
<tr>
<th>PDB ID</th>
<th>cutoff-GB</th>
<th>1-charge HCP-GB</th>
<th>2-charge HCP-GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMS difference in RMS residue fluctuations (Å)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1BDD</td>
<td>0.51</td>
<td>0.54</td>
<td>0.37</td>
</tr>
<tr>
<td>1UBQ</td>
<td>0.24</td>
<td>0.20</td>
<td>0.15</td>
</tr>
<tr>
<td>2BNA</td>
<td>0.003</td>
<td>0.01</td>
<td>0.03</td>
</tr>
<tr>
<td>2TRX</td>
<td>0.37</td>
<td>0.32</td>
<td>0.22</td>
</tr>
<tr>
<td>Average</td>
<td>0.28</td>
<td>0.26</td>
<td>0.19</td>
</tr>
<tr>
<td>RMS difference in distribution of $\chi_1$ angles (% occurrence)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1BDD</td>
<td>3.26</td>
<td>2.19</td>
<td>2.49</td>
</tr>
<tr>
<td>1UBQ</td>
<td>3.15</td>
<td>3.26</td>
<td>2.81</td>
</tr>
<tr>
<td>2TRX</td>
<td>3.17</td>
<td>3.48</td>
<td>3.16</td>
</tr>
<tr>
<td>Average</td>
<td>3.19</td>
<td>2.98</td>
<td>2.82</td>
</tr>
<tr>
<td>RMS difference in distribution of $\chi_2$ angles (% occurrence)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1BDD</td>
<td>2.79</td>
<td>2.01</td>
<td>2.21</td>
</tr>
<tr>
<td>1UBQ</td>
<td>2.66</td>
<td>2.63</td>
<td>2.37</td>
</tr>
<tr>
<td>2TRX</td>
<td>2.87</td>
<td>2.91</td>
<td>3.01</td>
</tr>
<tr>
<td>Average</td>
<td>2.77</td>
<td>2.52</td>
<td>2.53</td>
</tr>
</tbody>
</table>

difference in flexibility is most likely due to the fact that the cutoff-GB method completely ignores distant charges which contribute to the bending rigidity of the DNA chain.

We also ran a 0.3 ns simulation of the nucleosome core particle, with the same setup as described in Sec. 3.3.3 above, but without the salt. When using the reference GB and the HCP-GB methods, all of the histone tails collapse onto the DNA chain within 0.1 ns, consistent with experimental observations at low salt concentrations,\(^\text{156}\) whereas when the cutoff-GB method is used, two of the positively charged tails fail to collapse onto the negatively charged DNA, Fig. 3.19. Again, this is most likely because, in the case of the cutoff-GB method, the positive charges at the ends of the histone tail do not “feel” the attraction from the highly charged DNA chain.

The above results suggest that in general the HCP-GB reproduces the dynamics of the reference GB simulation more accurately than the cutoff-GB method.

#### 3.3.4.7 A practical application: chromatin fibre

We expect the HCP-GB to be indispensable in the modeling of large structures where the pairwise GB without further approximation is impractical. One such example is the chromatin fibre where a 348 000 atom (12 nucleosome) structure is needed at a minimum to
Figure 3.17: Distribution of $\chi_1$ angles. Results for the functionally important CYS-32 of thioredoxin (2TRX) and THR-7 of ubiquitin (1UBQ) from 50 ns simulations. The trajectory was sampled every 10 ps. A bin size of 10° was used for calculating the distribution of $\chi_1$ angles. Connecting lines are shown to guide the eye.

Figure 3.18: DNA flexibility. Distribution of distances between terminal base pairs for a 30 bp DNA strand from a 10 ns MD simulation. Trajectories were sampled every 10 ps and a bin size of 2 Å is used for calculating the distribution of distances between terminal base pairs. Connecting lines are shown to guide the eye.

study its functional characteristics. Such a structure can be constructed using the crystal
structure for the nucleosome (1KX5) as a starting point. Multiple copies of the nucleosome can then be combined to construct the chromatin fibre, using a set of coordinate transformations described by Wong et. al.\textsuperscript{125} The coordinate transformations result in a number of severe steric clashes. A 15 ps simulation of the fibre using the 2-charge HCP-GB significantly reduces the steric clashes, as seen by the large reduction in the potential energy (Fig. 3.20). To reduce run time for this simulation, the protocol described in Sec. 3.3.3 was modified to reduce the heating and equilibration stages from 10 ps to 2 ps.
3.3.4.8 Mitigating the effect of violating Newton’s third law

Although the HCP uses the same all-atom force field as the reference GB computation, the HCP is a multiscale model in that different levels of approximations are used for the same set of atoms depending on their distance from the point of interest. The asymmetric interactions due to the multiscale approximations can violate Newton’s third law resulting in a residual force on the system.\textsuperscript{2,132} This residual force can produce an artificial center of mass motion and an overall rotation of the structure. A net residual force within a closed system causes the system as a whole to accelerate, even though there is no external force, resulting in the nonconservation of energy. Table 3.5 shows the net force and torque due to the violation of Newton’s third law for the HCP-GB method, on the set of test structures considered here, along with estimated center of mass displacement, rotation, and kinetic energy after 10 steps of a typical molecular dynamics simulation (10 fs). To estimate the kinetic energies, we treat the structures as rigid bodies and assume that the principle axis of rotation passes through the center of mass.

For the test structures considered here, the 3 million atom chromatin fibre represents the worst case with a linear displacement of $6 \times 10^{-6}$ Å, rotation of $1 \times 10^{-8}$ radians and kinetic energy of 0.09 kT after 10 steps of MD. These spurious motions are small compared to the stochastic collisions used in constant temperature simulations, which are on the order of 1 kT, and may not materially affect the dynamics of the simulation if a strong enough coupling to a thermal bath is used. However, implicit solvent MD simulations often use minimal or no viscosity to increase the sampling of conformation space. These regimes can result in a large center of mass drift, which can be inconvenient when visualizing or analyzing the trajectory. For example, for a 1 ns simulation of thioredoxin using the 2-charge HCP-GB method and
<table>
<thead>
<tr>
<th>Structure</th>
<th>PDB ID</th>
<th>Residual force (kcal/mol/Å)</th>
<th>Residual torque (kcal/mol)</th>
<th>After 10 iterations of dynamics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Displacement (Å)</td>
<td>Rotation (radians)</td>
<td>Kinetic energy (kT)</td>
</tr>
<tr>
<td>1-q HCP-GB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2BNA</td>
<td>0.003</td>
<td>0.38</td>
<td>$1 \times 10^{-8}$</td>
<td>$1 \times 10^{-8}$</td>
</tr>
<tr>
<td>1BDD</td>
<td>0.22</td>
<td>0.73</td>
<td>$9 \times 10^{-7}$</td>
<td>$3 \times 10^{-8}$</td>
</tr>
<tr>
<td>1UBQ</td>
<td>0.24</td>
<td>0.64</td>
<td>$6 \times 10^{-7}$</td>
<td>$1 \times 10^{-8}$</td>
</tr>
<tr>
<td>2TRX</td>
<td>0.97</td>
<td>3.81</td>
<td>$2 \times 10^{-6}$</td>
<td>$4 \times 10^{-8}$</td>
</tr>
<tr>
<td>1KX5</td>
<td>1.36</td>
<td>51.40</td>
<td>$1 \times 10^{-7}$</td>
<td>$3 \times 10^{-9}$</td>
</tr>
<tr>
<td>Microtubule</td>
<td>179.24</td>
<td>2603.63</td>
<td>$3 \times 10^{-6}$</td>
<td>$5 \times 10^{-9}$</td>
</tr>
<tr>
<td>1A6C</td>
<td>$2 \times 10^{-11}$</td>
<td>$6 \times 10^{-10}$</td>
<td>$1 \times 10^{-19}$</td>
<td>$2 \times 10^{-22}$</td>
</tr>
<tr>
<td>Chromatin</td>
<td>6804.00</td>
<td>1198829.76</td>
<td>$6 \times 10^{-6}$</td>
<td>$1 \times 10^{-8}$</td>
</tr>
<tr>
<td>2-q HCP-GB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2BNA</td>
<td>0.015</td>
<td>0.07</td>
<td>$5 \times 10^{-8}$</td>
<td>$2 \times 10^{-8}$</td>
</tr>
<tr>
<td>1BDD</td>
<td>0.18</td>
<td>0.44</td>
<td>$7 \times 10^{-7}$</td>
<td>$2 \times 10^{-8}$</td>
</tr>
<tr>
<td>1UBQ</td>
<td>0.23</td>
<td>0.58</td>
<td>$6 \times 10^{-7}$</td>
<td>$1 \times 10^{-8}$</td>
</tr>
<tr>
<td>2TRX</td>
<td>0.98</td>
<td>4.64</td>
<td>$2 \times 10^{-6}$</td>
<td>$5 \times 10^{-8}$</td>
</tr>
<tr>
<td>1KX5</td>
<td>2.94</td>
<td>57.98</td>
<td>$3 \times 10^{-7}$</td>
<td>$4 \times 10^{-9}$</td>
</tr>
<tr>
<td>Microtubule</td>
<td>59.76</td>
<td>598.27</td>
<td>$1 \times 10^{-6}$</td>
<td>$1 \times 10^{-9}$</td>
</tr>
<tr>
<td>1A6C</td>
<td>$2 \times 10^{-11}$</td>
<td>$4 \times 10^{-10}$</td>
<td>$1 \times 10^{-19}$</td>
<td>$1 \times 10^{-22}$</td>
</tr>
<tr>
<td>Chromatin</td>
<td>3483.09</td>
<td>836570.85</td>
<td>$3 \times 10^{-6}$</td>
<td>$7 \times 10^{-9}$</td>
</tr>
</tbody>
</table>

Table 3.5: Center-of-mass motion due to violation of Newton’s third law for the HCP-GB.

Langevin dynamics with a collision frequency of 50 ps$^{-1}$, the center of mass drift is 2.24 Å, similar to the 1.81 Å drift for the reference GB simulation. Whereas with a collision frequency of 1 ps$^{-1}$, the center of mass drift in the HCP-GB simulation is 30.96 Å, and as much as 183.54 Å with a collision frequency of 0.1 ps$^{-1}$. A commonly used approach for removing center of mass drift and rotation during the course of molecular dynamics is to employ a velocity correction algorithm, e.g., the NSCM option in Amber$^{131}$ which specifies the frequency at which center-of-mass motion is removed. We have implemented the same option in NAB. The velocity correction approach does not however correct the source of the problem – the net residual force. Moreover, a velocity correction not only eliminates the artificial motion caused by the violation of Newton’s third law but also affects the random motion due to Langevin dynamics which may not be desirable in some situations. Therefore, we considered applying a force correction aimed at mitigating the effects of the third law violation. Sec. 3.3.4.11 describes the two force correction approaches we considered – a molecular level and a component level force correction. By neutralizing the net residual forces, the force correction eliminates the systematic drift in the center of mass position caused by the violation of Newton’s third law. For example, for the 1 ns simulation of thioredoxin using the 2-charge HCP-GB method with a Langevin dynamics collision frequency of 0.1 ps$^{-1}$, the molecular level force correction reduces the drift from 183.54 Å to 29.49 Å which is similar to the 29.47 Å drift for the reference GB simulation which, of course, does not violate the third law within numerical precision of the integrator. The force correction however has several
shortcomings compared to the velocity correction often used by existing MD algorithms. It causes an increase in the force error as described in Sec. 3.3.4.11, while velocity correction does not effect the forces. Unlike a velocity correction, the force correction only eliminates drift, not rotation. And, the force correction must be applied at every step of the simulation since it eliminates the center of mass acceleration (change in velocity), whereas a velocity correction can be applied less frequently since it eliminates center of mass velocity itself. For these reasons, a velocity correction, its drawbacks notwithstanding, may be preferable to the force correction for eliminating the drift and rotation caused by the violation of Newton’s third law. Very preliminary testing suggests that the velocity correction may improve the stability of HCP-GB simulations for small structures. For example, the RMS deviation for the 1-charge HCP-GB simulation of 1BDD (Figure 8a), is in closer agreement with the reference GB simulation when the velocity correction is used (results not shown).

3.3.4.9 Mitigating the effect of the discontinuity at threshold boundaries

During the course of molecular dynamics, atoms may cross threshold boundaries that determine the level of approximation used in the computation of potentials and forces. These discontinuous changes in the level of approximation result in changes in potential that are inconsistent with the forces acting on individual atoms; i.e., the force is not equal to the derivative of the potential with respect to distance and can result in the nonconservation of energy and instability in the simulation.\(^\text{157}\) For example, 1000 steps of constant energy simulation for thioredoxin, without Langevin dynamics or surface-area dependent energy (Fig. 3.21), show that energy is not conserved for the cutoff-GB and HCP-GB methods. However, the non-conservation of energy is much larger in the case of the cutoff-GB method. For the cutoff-GB method, the energy contribution of an atom abruptly drops to zero when the atom moves beyond the cutoff boundary, resulting in a larger change in energy compared to the HCP-GB method, where the energy contribution of the atom is replaced by an approximation when the component containing the atom moves beyond the threshold boundary.

The discontinuity at threshold and cutoff boundaries can be eliminated by the use of a smoothing function.\(^\text{17,34}\) The smoothing function eliminates the discontinuity by gradually switching from one level of approximation to another over a short switching distance. Delle Site\(^\text{132}\) has however shown that the smoothing function cannot in general restore the conservation of energy for multiscale methods. To eliminate the discontinuity at threshold boundaries, we adapted the smoothing function described by Loncharich and Brooks.\(^\text{108}\) The smoothing function is used to calculate the force \(f(r)\) at a distance \(r\) from the point of interest inside the switching region \(h < r < h + s\) where \(h\) is the threshold distance and \(s\) is the switching distance, as follows:

\[
f(r) = S(r) f_{h+s} + (1 - S(r)) f_h \quad (h \leq r \leq h + s)
\]

\[
S(r) = \frac{(h^2 - r^2)^2 (h^2 + 2r^2 - 3(h + s)^2)}{(h^2 - (h + s)^2)^3}
\]
where $f_h$ and $f_{h+s}$ are the forces due to a component computed by the HCP-GB at $h$ and $h+s$ respectively.

The nonconservation of energy can be measured as the standard deviation in total energy, and the discontinuity in computed energy as the standard deviation in the change in total energy between consecutive steps in the MD simulation. Table 3.6 summarizes these metrics for the HCP-GB method with and without smoothing, the cutoff-GB method, and the reference GB computation. The table shows that, although the HCP-GB method does not conserve energy, it represents a significant improvement over the cutoff-GB method in that respect. On the other hand, although smoothing does improve energy conservation, it is comparable to extending the level 1 threshold distance to the end of the smoothing region, 18 Å in this case, as measured by the standard deviation in total energy (Table 3.6). The HCP-GB with smoothing does show less discontinuity than the HCP-GB with the extended threshold distance; however, the difference may not be sufficient to justify the higher computational cost of smoothing compared to simply extending the threshold distance. Note that the smoothing function can also improve the accuracy of the spherical cutoff method and has been studied previously.\textsuperscript{17,34,108} The results shown in the preceding subsections do not include the smoothing function for either the cutoff-GB or the HCP-GB methods.

We stress that in the case of multiscale approximations based on pairwise potentials, such as the HCP, exact energy conservation can not be achieved due to the violation of Newton’s third law.
Standard deviation (kcal/mol) & reference GB & cutoff GB & 1-q HCP-GB & 2-q HCP-GB \\
--- & --- & --- & --- & --- \\
Total Energy (no smoothing) & 0.0012 & 2.8213 & 0.2483 & 0.2476 \\
(HCP-GB with smoothing) & 0.0803 & 0.0992 & 0.0942 & 0.1013 \\
(HCP-GB with \(h_1 = 18\) Å) & 0.0228 & 0.0203 & 0.0033 & 0.0025 \\
Δ Total Energy (no smoothing) & 0.0007 & 0.3316 & 0.0062 & 0.0051 \\
(HCP-GB with smoothing) & 0.0033 & 0.0025 & 0.0062 & 0.0051 \\
(HCP-GB with \(h_1 = 18\) Å) & 0.0033 & 0.0025 & 0.0062 & 0.0051 \\

Table 3.6: Effect of smoothing function on energy conservation. Degree of energy conservation is measured as standard deviation in total energy and degree of discontinuity as the standard deviation in change in total energy between consecutive steps of MD simulation. The default level 1 threshold distance \(h_1\) for the HCP-GB method is 15 Å. For comparison we include the reference GB computation, the cutoff-GB method with a 15 Å cutoff distance, and the HCP-GB method with a 18 Å level 1 threshold distance. The small fluctuation in total energy for the reference GB computation is due to the finite integrator time-step.

3.3.4.10 Component effective Born radii

The component effective Born radii are used to approximate the contribution of distant components to the solvation energy, as described in the Methods section. We examined three different alternatives for this approximation: the approximation defined by Eq. 3.17, the approximation defined by Archontis and Simonson\(^8\) and the volume integral approximation based on Eq. 3.18. Archontis and Simonson approximate the Born radius \(B_c\) for a component \(c\) as

\[
\frac{1}{B_c} \approx \frac{1}{\sum_{i \in c} q_i^2} \sum_{i \in c} \frac{q_i^2}{B_i} \quad (3.35)
\]

where \(q_i\) and \(B_i\) are the charges and effective Born radii respectively for the atoms \(i\) belonging to component \(c\). On average we found that the first alternative (Eq. 3.17) was most accurate as shown in Fig. 3.22.

3.3.4.11 Force corrections for neutralizing net residual force

The violation of Newton’s third law by the HCP-GB method, as described in Sec. 3.3.4.8, results in a net residual force. We considered two approaches for neutralizing the net residual force – a molecular level and a component level force correction.

The molecular level approach applies a mass weighted force correction to each atom to neutralize the total residual force on the whole structure. This force correction is computed as

\[
f^{corr}_i = f^{res} m_i / M \quad (3.36)
\]

where \(f^{corr}_i\) is the force correction subtracted from the force on atom \(i\), \(f^{res}\) the total residual force, \(m_i\) the mass of atom \(i\) and \(M\) the total mass of the structure.
Figure 3.22: Comparison of methods for computing component effective Born radii. Figure shows relative error in electrostatic energy for (a) 1-charge HCP-GB, (b) 2-charge HCP-GB, and relative RMS error in electrostatic force for (c) 1-charge HCP-GB, and (d) 2-charge HCP-GB, for four alternative methods. The four alternative methods are the spherical cutoff method, the volume integral (vol int) based on Eq. 3.18, Anandakrishnan-Onufriev (ao) defined by Eq. 3.17, and Archontis-Simonson (as) defined by Eq. 3.35. Cutoff distance $= 15 \text{ Å}$. HCP threshold distance $h_1 = 15 \text{ Å}, h_2 = 80 \text{ Å}, h_3 = 175 \text{ Å}$. Connecting lines are shown to guide the eye.
The component level approach is more complex. It aims to eliminate not only the net residual force, but also the net residual force for each component, where the residual force for a component is the difference between the total force on a component due to all other atoms and the total force on all other atoms due to the component. In other words, it aims to restore Newton’s third law at the component level. For the first level of HCP approximation the force correction \( f_i^{\text{corr}} \) for an atom \( i \) belonging to component \( c \), is calculated as

\[
f_i^{\text{corr}} = \frac{f_i^{\text{diff}}}{n_c} \quad (i \in c)
\]

(3.37)

\[
f_c^{\text{diff}} = \left[ \sum_{j \not\in c} \sum_{i \in c} f_{ji} + \sum_{k \not= c} \sum_{i \in c} f_{ki} \right] + \left[ \sum_{i \in c} \sum_{j \not\in c} f_{ij} + \sum_{j \not\in c} f_{cj} \right]
\]

(3.38)

where \( f_c^{\text{diff}} \) is the difference between the total force on a component due to all other atoms and the total force on all other atoms due to the component. The first term on the right hand side of Eq. 3.38 is the force \( f_{ji} \) on the atoms \( i \) belonging to component \( c \) due to other atoms \( j \) not belonging to \( c \), computed at the atomic level (level 0). The second term is the force \( f_{ki} \) on atoms \( i \) belonging to \( c \) due to other components \( k \) treated at the component level (level 1). The third term is the force \( f_{ij} \) due to \( c \) on atoms \( j \) not belonging to \( c \) where the atoms \( i \) within \( c \) are treated at the atomic level (level 0). And the last term is the force \( f_{cj} \) due to \( c \) on atoms \( j \) not belonging to \( c \) where \( c \) is treated at the component level (level 1).

This force correction is generalized for higher levels of HCP by including terms in Eq. 3.38 for higher level HCP components. Since the individual terms on the right hand side of Eq. 3.38 are already being computed, the incremental cost of both force correction approaches scale as \( \sim n \) which is \( < n \log n \).

Although the above force corrections neutralize the net force due to the violation of Newton’s third law, they also cause an increase in the force error. To see why, consider the case where the force error compared to the reference GB computation is approximately zero. In this case any net force correction will result in an increase in the force error. In general, when the force error is less than half the net force correction, the net force correction will cause an increase in force error. Thus on average where the force error is randomly distributed, the net force correction will result in an increase in force error.

### 3.3.5 Conclusions

Implicit solvent models are routinely used where it is important to sample a large conformation space, such as for protein folding, replica exchange, and docking simulations. However, the implicit solvent model employed most extensively in molecular dynamics – the generalized Born (GB) model – scales poorly as \( \sim n^2 \), where \( n \) is the number of solute atoms, limiting their usefulness for long time-scale simulations or the simulation of large structures. We have presented here an \( \sim n \log n \) implementation of the implicit solvent GB model based on the hierarchical charge partitioning (HCP) approximation previously developed by us. The HCP method uses the natural organization of biomolecular structures to partition the structures into multiple hierarchical levels of components such as atoms, groups (residues), subunits (chains), and complexes. The charge distribution for each of these components
other than the atoms are approximated by a small (one or two) number of charges. For the computation of electrostatic interactions with distant components, the HCP uses the approximate charges while using the atomic charges for nearby components. The greater the distance from the point of interest, the larger (higher level) is the component used in the approximation. We have previously described a top-down algorithm for the HCP that scales as $\sim n \log n$ for biomolecular structures.

This study extends the HCP approximation to the GB model (HCP-GB) such that both the computation of pairwise interactions and the effective Born radii scale as $\sim n \log n$.

The HCP-GB method is implemented in the open source molecular dynamics software, NAB, in AmberTools v1.3. The HCP implementation in NAB is scheduled to be released with AmberTools v1.5, for general use. The accuracy, speed, and stability of the method were then evaluated on a set of representative biomolecular structures ranging in size from 632 to $\sim 3$ million atoms. The performance of the HCP-GB method was compared to the spherical cutoff method with GB (cutoff-GB) where all computations, including the computation of the effective Born radii, ignore all atoms beyond a specified cutoff distance. Our results show that the HCP-GB method is more accurate, as measured by the relative RMS error in electrostatic force, than the cutoff-GB method for the structures tested. Depending on the size of the structure, the HCP-GB method was also 1.1 to 390 times faster than the reference GB computation. An analysis of 50 ns simulations of four structures – B-DNA, immunoglobulin binding domain, ubiquitin and thioredoxin – shows that the results for the HCP-GB simulation are in reasonable agreement with the reference GB simulation without cutoffs. For the very small (726 atom) immunoglobulin binding domain protein (1BDD), the 1-charge HCP-GB method exhibited RMS deviations from the crystal structure similar to the cutoff-GB and larger than the reference-GB and 2-charge HCP-GB simulations. Therefore, we do not recommend the use of 1-charge HCP-GB for the simulation of such small structures. However, very preliminary testing suggests that the velocity correction, described below, may improve the stability of HCP-GB simulations for small structures.

There is also an important qualitative difference between the HCP-GB method and the cutoff-GB method. The cutoff-GB ignores charges beyond the cutoff distance while the HCP-GB method approximates the influence of distance charges. Our testing suggests that this difference can have a significant impact on details of the dynamics. For example, for the 50 ns simulations of four structures, the residue flexibility and $\chi_1$ and $\chi_2$ angles for the cutoff-GB simulations show larger deviations from the reference GB simulation than the HCP-GB simulations. Similarly, a 10 ns simulation of a 30 base-pair DNA strand showed that the flexibility of the molecule, as measured by end-to-end distance, using the HCP-GB method was similar to that of the reference GB simulation, whereas the cutoff-GB method results showed a more flexible molecule. And a series of simulations of the nucleosome core particle showed that with the reference GB and HCP-GB methods all of the positively charged tails of the histone chains collapsed onto the negatively charged DNA, whereas two of the histone tails failed to do so with the cutoff-GB method.

Due to its multiscale nature, the HCP-GB method can violate Newton’s 3rd law resulting in a residual center of mass force and torque. For the structures tested here, the effect of the residual force and torque is much smaller than the “noise” due to stochastic collisions.
used in constant temperature simulations with strong coupling to a thermal bath. However, when a weak coupling is used to increase the sampling of conformational space, the residual force and torque may cause the structure to drift and rotate, making it inconvenient for visualization and analysis. For simulations with weak coupling to a thermal bath, the center of mass motion and rotation can be eliminated by using a velocity correction. The multiscale nature of the HCP-GB can also result in discontinuities at threshold boundaries, which can cause energy not to be conserved. The discontinuity and the resultant nonconservation of energy for the HCP-GB method is however much smaller than that of the cutoff-GB method. Smoothing functions can be used to reduce the discontinuities and the non-conservation of energy. However, we found that increasing the threshold distance may be a more effective way of achieving the same result.

To demonstrate a practical application of the HCP-GB method, we used it to refine a 348 000 atom chromatin fibre. The 15 ns all-atom simulation successfully resolved numerous severe steric clashes, significantly improving the quality of the starting structure.

In conclusion, the $\sim n \log n$ HCP-GB method is always faster than the $\sim n^2$ reference GB computation without additional approximations. Although the speed of the HCP-GB method is comparable to using a spherical cutoff for GB computations, which also scales as $\sim n \log n$, the HCP-GB method on average more closely reproduces key characteristics of the dynamics of the reference GB simulations. Our testing suggests that this may be because the HCP-GB method approximates the influence of distant charges, unlike the cutoff-GB method, which completely ignores them. In general, our findings suggest that compared to the cutoff-GB, the HCP-GB method may always be the preferable approach for speeding up pairwise GB computations for molecular dynamics. Where speed is critical, one can consider using the 1-charge HCP-GB instead of the 2-charge HCP-GB or reducing threshold distances from the recommended conservative threshold distances.

This study was intended to be a proof-of-concept of a novel method, and a number of potential improvements and optimizations remain to be studied. In particular, further optimization of the placement of approximate charges, comparison of alternate approximations for component effective Born radii, choice of parameters, comparison of velocity vs. force correction, and the treatment of very distant components. Most importantly, more extensive testing is required to further define the applicability and limitations of the proposed $n \log n$ GB method.
3.4 Implementation on GPUs for computing electrostatic surface potential

3.4.1 Introduction

Electrostatic interactions are critical for biomolecular function. At the same time, the long-range nature of these interactions often makes their estimation a computational bottleneck in current atomistic molecular modeling. Approaches to speed-up these computations can generally (though not always cleanly) be subdivided into two very broad categories: (1) those that seek to gain speed by making computationally effective approximations to the underlying physical model and (2) those that do not affect the accuracy of the physical model but strive to accelerate the computation at the software or hardware levels. Examples in the first category include the spherical cut-off method, the fast multipole approximation, and the current “industry standard” for explicit solvent simulations — the particle mesh Ewald (PME) method. Among the most prominent practical approaches in the second category is parallel computation on multiple processing cores. Each of these approaches has its advantages and limitations. For example, the PME method can be very accurate but requires an artificial periodicity to be imposed on the molecular system. In addition, the method is currently not suitable for implicit solvent simulations.

Parallel computation on multiple processing cores may lead to spectacular speed-ups for certain types of problems, but not for others — for example, specific PME implementations may not scale all that well on large parallel machines. Furthermore, access to such expensive resources may be limited.

Within either of the above general categories of approaches, estimating the long-range electrostatic interactions is still computationally intensive. Even a single state computation which involves long range electrostatics, may require an extraordinary amount of computational resources for larger structures. For example, a 2006 study of electrostatic properties of viral capsids using the adaptive Poisson-Boltzmann solver, required 1000 processors on the Blue Horizon and Data Star supercomputers for each run. Computational requirements for modern molecular dynamics, which may require millions of such single state estimates, are even more demanding. In the past, in addition to algorithmic advances, scientists could also rely on Moore’s Law to continually accelerate these computations. The rapid hardware advances from Moore’s Law gave software a “free ride” to better performance, but this free ride is now over. With clock speeds stalling out and computational horsepower instead increasing due to the rapid doubling of the number of cores per processor, serial computing is now moribund in many areas of natural science, and the vision for parallel computing, which started over forty years ago, is a revolution that is now upon us.

The traditional approach to parallel computing has made use of large-scale supercomputers, oftentimes referred to as big iron. However, such supercomputers present significant chal-
lenges with respect to ease of access and use, and cost, e.g., the fastest supercomputer in the world in 2009 cost $133M to build. In contrast, the growing proliferation of many-core processors, like the graphics processing unit (GPU) on a video card, promises to deliver supercomputing horsepower to the desktop while simultaneously enhancing ease of access as well as dramatically reducing cost. With the peak floating-point performance of a GPU now at a teraflop ($10^{12}$ floating-point operations per second), the GPU delivers supercomputing in a small and economical package. For example, a high-end server with the latest GPU card costs a mere $1,500, resulting in an astounding performance-price ratio of 667 megaflops per dollar and performance-space ratio of 500 teraflops per square foot. In contrast, the world’s fastest supercomputer, Roadrunner, has a peak of 1,457 teraflops at a cost of $133M for a mere performance-price ratio of 11 megaflops per dollar and performance-space ratio of 243 teraflops per square foot. However, the current programming model for GPUs is only amenable to highly data-parallel applications; efficient GPU mappings for less data-parallel applications are extraordinarily difficult to realize. Unlike supercomputer clusters consisting of general-purpose processors and direct support for interprocessor communication, the GPU has limited interprocessor communication capabilities and limited data cache. Therefore the GPU is most effective when performing stream processing, i.e., performing a similar set of computations against a large set of data. This paper discusses the techniques used to address limitations of the GPU while taking advantage of its multiprocessing capabilities in the context of electrostatic computations.

A number of different biomolecular modeling applications have recently been implemented on GPUs, including the computation of long-range electrostatic potential in the context of molecular dynamics. Our implementation focuses on the computation of electrostatic surface potential, using a realistic solvation model, in order to demonstrate that “multiscale” approximation schemes, such as the hierarchical charge partitioning (HCP) algorithm, can be combined with the GPU to achieve significantly greater speed-up than the HCP approximation or the GPU alone. Specifically, we algorithmically map and transform electrostatic potential calculation along with the HCP approximation onto the GPU.

The remainder of this paper is organized as follows. In the next section, we briefly describe the specific application considered here (i.e., computation of molecular surface potential), the HCP, and the GPU implementation. Then we examine the speed and accuracy of this implementation for a set of representative biomolecular structures. In conclusion, we summarize our finding and discuss the future potential for the approach presented here.

### 3.4.2 Methods

This section describes the specific methods used to compute long-range interactions and an implementation of the entire computational process on the GPU. Specifically, we investigate the extent to which the speed-up resulting from approximating the electrostatic potential via a multiscale approach (HCP, described below) can be combined with the speed-up resulting from mapping and transforming the electrostatic potential calculation introduced earlier (GEM) along with the HCP approximation onto the GPU.
http://people.cs.vt.edu/~onufriev/software. The platform specific computational core for running GEM with one level of HCP on the GPU, as described below, is available from the authors upon request.

3.4.2.1 Computation of Biomolecular Electrostatic Potential

All of the electrostatic calculations presented in this paper were done using the analytic, linearized Poisson-Boltzmann (ALPB) model\(^\text{183-185}\) as implemented in the GEM package. The functional form of the ALPB electrostatic potential \(\phi_i\) varies depending on the region of space where the potential is computed; two of the regions – the interior of the molecule and the solvent space – are depicted in figure 3.23 as the two dielectrics. The specific functional forms of \(\phi_i\) for all the regions are given in Gordon et al.\(^\text{183}\) Equation (3.39) is the simplest solution for the calculation of physically admissible \(\phi_i\) anywhere in the solvent, including the molecular surface, see figure 3.23. The constant \(\alpha = 0.580127\), in equation (3.39) minimizes the error in the solvation energy of a random charge distribution inside a sphere.\(^\text{186}\) Adding the effects of salt in the Debye-Hückel (linear) limit is outlined in Gordon et al.\(^\text{183}\) The potential at any single point in space, for example a vertex point on the molecular surface, is computed as a sum of contributions from individual atomic charges in the solute. Such a calculation can be done without the need to also compute the potential at any other point(s) – a computational freedom that the numerical Poisson-Boltzmann solvers are missing. An extensive analysis of the accuracy of the ALPB model in computing biomolecular electrostatic potential is presented elsewhere.\(^\text{183}\)

\[
\phi_{solvent}^i = \frac{q_i}{\epsilon_{out}} \left(1 + \frac{\alpha}{\epsilon_{out}}\right) \left[\frac{(1 + \alpha)}{d_i} - \frac{\alpha(1 - \frac{\epsilon_{in}}{\epsilon_{out}})}{r}\right]
\]  

(3.39)

To assess the speed-up resulting from the combination of the HCP and the GPU, we selected the GEM software package.\(^\text{183}\) GEM is an open-source implementation of the ALPB model. GEM was selected as a platform for experimentation, in this case, due to the facile nature of the algorithm and the flexibility of the platform for innovation and modification to rapidly generate prototypes.

3.4.2.2 Mapping GEM and HCP onto the GPU

The initial exploration of mapping GEM and HCP to the GPU was conducted using the ATI Stream Software Development Kit (SDK) from AMD. The SDK provides a high-level programming language called ATI Brook+\(^\text{187}\) to ease the development of applications for the GPU, and it includes a compiler and run-time layer that handles the low-level details necessary to run computations on the GPU.

Brook+ is based on the Brook stream computing language from Stanford,\(^\text{188}\) which in turn is an extension of standard ANSI C. Brook+ is specifically optimized to compile and run
Figure 3.23: ALPB model parameters. Geometric parameters in equation (3.39) used to compute the electrostatic potential $\phi_i$ due to a single charge located inside an arbitrary biomolecule (in the absence of mobile ions). Here $d_i$ is the distance from the source charge $q_i$ to the point of observation where $\phi_i$ needs to be computed. The molecular surface (solid line) separates the low dielectric interior ($\epsilon_{in}$ blue region) from the high dielectric solvent space, $\epsilon_{out}$. We project the molecular surface defined by pre-computed vertex points a small distance, $p$, outwards into the solvent space along the surface normals. The projected surface is shown as the dashed line. The so-called effective electrostatic size of the molecule, $A$, characterizes its global shape and is computed analytically. The distance from the point of observation to the “center” is then defined as $r = A + p$.

on ATI stream-capable GPUs through the AMD/ATI Compute Abstraction Layer (CAL). Whereas CAL presents a low-level programming interface, Brook+ provides a high-level programming interface, thus easing the development of GPU applications.

The calculation of electrostatic potential via the analytical formula implemented in GEM, mentioned above, can be decomposed as a data-parallel computation across all points on the surface, or vertices, at which the electrostatic potential is calculated. In addition, the potential at each vertex can be computed as a reduction (or more specifically, a sum) of a set of independent calculations on each atom, thus allowing for multiple dimensions of parallelism.

The basic mapping of GEM to the GPU treated the calculation of potential at each vertex as an individual unit of execution, and thus, computed one vertex per GPU thread. In the initial version, the electrostatic surface potential at all the vertices was computed with a single kernel launch, which is equivalent to an offloaded function call to the GPU device. We found that while this approach worked, it did not scale. That is, as the number of threads increased beyond a certain point (in this case, beyond 25,000-30,000 threads), so did the overhead of scheduling them, which in turn, degraded performance. Specifically, our tests showed that less than 25,000 was a reasonable number for the ATI Radeon 4870 card; anything more than 30,000 was materially slower despite the reduced number of kernel
launches. In light of this, subsequent versions only ran at most 25,000 threads per kernel launch.

While the above mapping was effective, it still retained many features of the original code that do not translate favorably into GPU performance. Below we describe additional modifications to improve the execution of GEM on the GPU.

3.4.2.2.1 Data Structures While Brook+ version 1.4 supports structures, it does not support referencing structure members in an array of structures. As a result, all the arrays of structures in GEM had to be flattened into arrays of primitives in order to get the GEM code to even run on the GPU. As much as this change was necessary to enable GEM to run on the GPU, the change also makes sense to do from a performance standpoint as it aligns the data structures to allow for coalesced memory accesses. We intend to investigate this further when the support for structures in Brook+ improves sufficiently for us to implement the code with the original structure of the data intact.

3.4.2.2.2 Conditional Performance Conditional statements, particularly conditionals that cause divergence (or divergent branches), are a common cause of performance degradation in GPU programs, as has been documented in previous work, including the AMD Stream User Guide.\textsuperscript{189} Non-divergent branching, on the other hand, is generally not mentioned in connection with performance issues. However, they too can incur a high cost; in the case of GEM, 30% performance degradation on the GPU and 15% on the CPU. Thus, minimizing conditionals in high traffic areas of code is long-standing conventional wisdom, which is often forgotten when working with modern CPUs. (Programmers sometimes rely on the branch prediction or speculative execution capabilities of CPUs to deal with potentially excessive use of conditionals.) For GPUs, the impact of branching is more severe since GPUs lack branch prediction and speculative execution, and other technologies that reduce the cost of branching, as on modern CPUs.

In GEM's original computational core, most conditionals were used to keep the code maintainable, e.g., selecting between code paths even though they were predetermined by the parameters to the program. Specifically, there are two input parameters: (1) the region in which the potential is computed and (2) the type of potential to be computed, which determines the execution path through five conditionals. For example, two of the conditionals select the region being computed, such as the interior of the molecule or the solvent space shown in Figure 3.23. To remove these conditionals, we created several different versions of the GEM function and Brook+ kernel, 15 total, each of which now contains only the conditionals that must be evaluated on every input item individually, cutting the total number of branches in most kernels to zero and at the most three.

The specific kernel to execute is now selected by a set of conditionals outside the function/kernel being run and the conditionals are only invoked once. A total of 2-5 conditionals is in stark contrast to what was originally required. The initial version required \# of atoms * \# of vertices * \# conditionals conditional statements, or for the viral capsid, \[476,040 \times 593,615 \times 5 = 1,412,922,423,000\] conditionals! Thus, we reduced almost one and
a half trillion conditionals down to five.

Hereafter, we refer to the version of the code with conditionals moved outside the kernel as the *split* version of the code. The *unsplit* version uses conditionals in the computational loop so as to avoid code replication. While this may be good practice in terms of readability and maintainability of the code, it adversely impacts performance. Compiler techniques or source-to-source translation could be used to make this solution more maintainable, but such techniques are outside the scope of this paper. All CPU and GPU versions used in the results section have the conditionals moved outside the main computational loop, as described above.

### 3.4.2.2.3 Hierarchical Charge Partitioning

To further accelerate the computation and to test the performance of GPU-based computation with a multiscale algorithm, we incorporated the HCP method described in the previous section. The incorporation of the HCP into the GPU-based computation introduced several new issues that had to be addressed. First, the “multiscale” charge partitioning introduced additional branching when the HCP threshold distance was less than the diameter of the molecule. As a consequence, some amount of divergent branching could not be avoided in order to efficiently implementing HCP on the GPU. It also added several fields of data, and thus arguments, that did not exist in the non-HCP versions. Due to a limitation in the number of bytes passable as arguments using the current version of Brook+, we moved the calculation of many of the arguments from the CPU to the GPU in order to save space in the argument list.

### 3.4.3 Test Setup

All results were measured on a system with an Intel Core 2 Duo E6550 processor, which contains 2 computing cores, 2GB of DDR2-800 memory, and an ATI Radeon HD 4870 GPU on a PCI-E X16 interface running 32-bit Ubuntu Linux version 8.10. The GPU experiments were run with a single CPU process along with a kernel offloaded to the HD 4870. The CPU experiments were run in a single process on one core of the E6550 processor. All times reported in this section are averages of five runs. The execution times for the runs proved to be quite consistent, with variances in execution time at less than 1%. Our base CPU version of GEM represents the performance attainable by an experienced programmer in C, short of inserting inline assembly code. SSE acceleration is applied via compiler optimizations in GCC 4.2.4, and using arrays of primitives, as discussed earlier. Both of these CPU optimizations keep the baseline as similar to the GPU implementation as possible. The GCC compiler-applied SSE optimizations improved performance by approximately two-fold. All computations were performed with single-precision floating-point arithmetic, both on the CPU and GPU.

A representative set of six atom-level molecular structures was used for testing. The structures were selected to span a large range of sizes. The six structures, their protein databank (PDB) IDs, and sizes are as follows:
Table 3.7: Summary of HCP GPU test results. Times, speed-ups, root mean square (RMS) error and relative RMS error of the surface potential calculations performed with the 4 GEM versions on the virus capsid structure, are averaged over 5 runs of each version. Speedup and error are calculated relative to the CPU version. Relative RMS error is calculated as RMS error divided by average absolute potential.

| Version          | Average Time (seconds) | Speed-up Over CPU | RMS Error (kcal/mol/|e|) | Relative RMS Error |
|------------------|------------------------|-------------------|---------------------|-------------------|-------------------|
| CPU              | 80690.2                | 1.00              | 0                   | 0                 |
| CPU with HCP     | 1442.2                 | 41.68             | 0.1680              | 0.0153            |
| GPU              | 219.2                  | 182.80            | 0.0001              | 0.00001           |
| GPU with HCP     | 35.0                   | 933.59            | 0.1680              | 0.0153            |

1. H helix of myoglobin, 1MBO, 382 atoms
2. chain A of calcium binding protein S100B, 1UWO, 1,441 atoms,
3. chain A of cytochrome CD1 nitrite reductase, 1QKS, 8,542 atoms,
4. nucleosome core particle, 1KX5, 25,086 atoms,
5. chaperonin GroEL, 2EU1, 109,802 atoms, and
6. tobacco ringspot virus capsid, 1A6C, 476,040 atoms.

The atomic coordinates for these structures were obtained from the protein databank and atomic charges assigned using the H++ system (www.cs.vt.edu/biophysics/H++), which uses the standard continuum solvent methodology to compute protonation states of ionizable groups. The surface vertex points at which electrostatic potential is calculated for these structures are obtained using the program MSMS. MSMS is run with a probe radius of 1.5 Å and a triangulation density of 3.0 vertices per Å² for all structures except the virus capsid, for which, due to limitations of the MSMS software a probe radius of 3.0 Å and a triangulation density of 1.0 vertices per Å² is used.

3.4.4 Results and discussion

Wall times from four different implementations of GEM, i.e., the computation of electrostatic surface potential, were collected: (1) GEM without HCP, running on a single CPU; (2) GEM with HCP, running on a single CPU; (3) GEM without HCP but running on the GPU, and (4) GEM with HCP but running on the GPU. Times were measured using time checking calls within the main function in each implementation, which remained unchanged between versions. Only the computational kernel was measured, excluding the file I/O necessary to read and write the data files.

We have three main versions of the code which merit comparison, see figure 3.24. Each version is compared with the reference non-HCP serial CPU version of the code. The use of the
Figure 3.24: Acceleration of the electrostatic potential computations. The red line with triangle data points shows the speed-up of the potential computation resulting from the use of the multiscale approximation HCP alone. The dark blue line with circle data points corresponds to the same potential computed on the GPU alone, and the light blue line with square data points represents the combined speed-up of both the HCP and the GPU. The shaded region visually emphasizes the range of structure sizes where the combined speed-up is greater than either the HCP or GPU speed-up by itself. All speed-ups are measured relative to the CPU serial version.

HCP approximation alone, the red line with triangle data points, speeds up the calculation from about 2x on the smallest structures to roughly 42x on the viral capsid, relative to the reference. The dark blue line with circle data points represents the Brook+ GPU version with all optimizations except the HCP applied, and is found to yield substantially higher speed-up than the HCP alone\(^5\), especially for larger structures. The light blue line with square data points corresponds to the use of GPU in combination with the HCP (GPU-HCP). For small structures, the stand alone GPU version performs better than the GPU-HCP version. However, beyond structures of the order of \(10^4\) atoms, the GPU-HCP version is the clear winner in performance gains. For example, for the largest structure with 476,040 atoms, the speed-up using the GPU only, without HCP, is 182x, the speed-up without the GPU, using HCP only, is 42x and the speed-up using both the GPU and HCP is 934x as can be seen in Table 3.7. In the same table we present the RMS error as a measure of accuracy achieved by each implementation. It is worth noting that the error introduced by switching from running single precision on the CPU to single precision on the GPU is almost immeasurably small. Since most computers are multi-core these days, including the test machine, it is also

\(^5\)In this work, we consider only the lowest level-1 HCP. Higher levels of “multiscale” are achievable within the HCP, see Anandakrishnan et. al.\(^2\) for details.
worth noting that the algorithm scales well across multiple cores, but since the potential combination of GPU acceleration and HCP is the goal of this paper we chose not to include direct results for multi-core CPUs. It is safe to say however that the speed-up is near but not exceeding direct speed-up, 1x per core, or roughly 2x on the test machine, and scaling to as many cores as there may be in a system.

The combined speed-up using both the GPU and HCP is not fully multiplicative due to the following two additional branches in the HCP algorithm. One, for the 1-charge 1-level HCP approximation implemented here, if the distance to a component is beyond a specified threshold distance the HCP algorithm treats the component as a single point charge, otherwise all atoms within the component are used in the computation. Two, if the net charge of a component is zero, no further computations are performed for that component. However, on the GPU, all threads must execute the same instruction in each cycle. Therefore, even if a component in a given thread is beyond the threshold distance or has a zero net charge, and can complete its computations much faster, the thread can not proceed until other threads have completed.

The accuracy of the computations was also evaluated. For the single point computation considered here the error introduced by the single precision GPU is much smaller than the error due to the HCP approximation alone. For example, for the nucleosome the RMS error due to the use of single precision compared to double precision (both on the CPU version) was 0.002916 kcal/mol/|e| and and the error introduced by the GPU compared to the single precision CPU version was 0.000119 kcal/mol/|e|, whereas the error due to the HCP approximation alone was 0.277960 kcal/mol/|e|. However, this small error could accumulate and grow in applications involving large numbers of cumulative computations, such as molecular dynamics simulations. Further analysis would be required to determine the affect of GPU and HCP errors on such applications.

3.4.5 Conclusions

Electrostatic surface potential can be an important indicator of biomolecular function and activity, thus the ability to compute and visualize surface potential can be a valuable tool for studying biomolecules. However, for large structures, the estimation of the surface potential can be computationally demanding, making such computations inaccessible to desktop PCs and even to large clusters. One can reduce these computational costs by using coarse-grain (multiscale) approximations or by parallelization across multiple processors, however it is not in general obvious that the two techniques can be successfully combined. We demonstrate here that for the computation of electrostatic surface potential combining a multiscale approximation with parallelization on the GPU can deliver significantly greater speed-up than either approach separately. The electrostatic surface potential is computed using the analytical linearized Poisson Boltzmann (ALPB) model, which use a set of simple formulae within the implicit solvent framework, to compute potential. The hierarchical charge partitioning (HCP) method is used to speed-up the calculation by using a multiscale approximation for the potential, and further speed-up is achieved by executing the computation on an ATI Radeon 4870 GPU. We find that the errors introduced by the use of single precision GPU
implementation are negligible for the purpose of computing single-point biomolecular potential.

We also show that, for large biomolecular structures, combining the power of the GPU with the HCP approximation can achieve a combined speed-up of up to 934x over the reference computation based on a single processor without the use of the HCP. This combined speed-up is larger than the individual speed-ups for structures larger than about 10,000 atoms, and for the largest structures tested is many times larger than what could have been achieved by the GPU (182x) or HCP (42x) alone. However, for structures smaller than about 10,000 atoms the combined performance is less than that of the GPU-based computation alone because of unequal processing times across multiple threads such that the speed-up is constrained by the slowest thread. One particular challenge to achieving this speed-up was the presence of divergent branching due to the “multiscale” threshold in HCP, which severely limited the performance gain. This optimization helped improve speed-up for large structures (> 10,000 atoms) but the associated overhead reduced performance for smaller structures.

Although this implementation produced impressive speed-ups, we do not believe it represents the full potential of the GPU-HCP approach. For example, only one level of HCP approximation was implemented here. We speculate that implementing additional levels of HCP, despite the additional branching involved, can conservatively result in an additional order of magnitude speed-up. The computation of electrostatic potential as implemented here, involves three steps - computing HCP group charges, determining charges to use in potential computation and the actual potential computation. Only the last of these three steps is executed on the GPU. Additional speed-up may be possible by also executing the first two steps on the GPU. These additional performance improvements will be explored in a future study.

The combined HCP-GPU implementation presented here raises the possibility of making various biomolecular modeling applications accessible to researchers using desktop computers, and in “real time”. Examples of applications that can benefit immediately from the methods presented in this work include calculations of surface potential around large structures. Exploration of the changes in the computed potential due to changes in the environment and/or structures, such as changes in pH or mutations, can now be computed virtually immediately, thus greatly facilitating research.

In the longer term, we hope that the main conceptual result of this work – that acceleration of the computation of long range interactions based on multi-scale ideas can be combined with GPU-based speed-ups – will impact a broad spectrum of applications where the speed of such computations is critical. For example, virtual screening for drug discovery involves screening hundreds of thousands to millions of potential candidate ligands, using “docking” simulations, to identify likely leads for further analysis. Both the ligand and drug target can have multiple conformations and relative orientations resulting in hundreds to thousands of combined degrees of freedom. It may be possible to apply the HCP-GPU approach to accelerate the virtual screening process. Similarly, the all-atom molecular dynamics simulation of even small to medium sized molecular systems (< 100,000 atoms), for biologically meaningful time scales (microseconds), requires the latest massively parallel supercomputers. It may be possible to make such simulations more accessible using the HCP-GPU approach.
presented here.
3.5 Implementation on GPUs for molecular dynamics

3.5.1 Introduction

Atomistic molecular dynamics simulations are routinely used to understand the structure and function of biological molecules. However, the timescales and structure sizes that can be studied through such simulations are generally far smaller than that required to observe many biological processes, such as protein folding and ligand binding. Two common approaches for extending the timescales and structure sizes accessible to atomistic molecular dynamics simulations are, (i) parallelization across multi- and many-core platforms and, (ii) algorithmic approximations.

Widespread adoption of Graphics Processing Units (GPUs) that are capable of general-purpose computations, in desktops and workstations, has made them attractive as accelerators for high performance parallel programs. The increased popularity has been assisted by (i) sheer computing power, (ii) superior performance/dollar ratio and (iii) a compelling performance/watt ratio. For example, a 8-GPU cluster costing thousands of dollars, can simulate 52ns/day of the JAC Benchmark as compared to 46ns/day on the Kraken supercomputer (Oak Ridge National Lab), which costs millions of dollars. Emergence of GPUs as an attractive high-performance computing platform is also evident from the fact that three out of the top five fastest supercomputers on the Top500 list employ GPUs as accelerators.

A wide range of applications in image and video processing, financial modeling and scientific computing have been shown to benefit from the use of GPUs. GPUs, however, suffer from a critical limitation. For maximum efficiency on the GPU, computational operations across processing cores need to be synchronized. Whereas, an important class of algorithmic approximations, the multiscale approximations, have highly asynchronous computational requirements. These approximations involve numerous divergent branches depending on the relative distances between interacting atoms, thereby, resulting in non-uniform computational requirements across processors. Thus, when multiscale algorithms are implemented on the GPU one may expect to achieve less than multiplicative speedups, i.e. total speedup being less than the product of the speedups for the multiscale algorithm and the GPU separately. To test this expectation, we implemented the hierarchical charge partitioning (HCP) multiscale algorithm on the NVIDIA GPU. The HCP code itself is implemented in NAB, the open source molecular dynamics module, in AmberTools v1.4 and tested using the distance-dependent-dielectric implicit-solvent model.

Contrary to our expectation the implementation resulted in near multiplicative speedups. The loss in performance due to the additional divergent branches in HCP algorithm is mostly offset by a corresponding reduction in the number of other divergent branches that need to be considered. These other divergent branches are bypassed by the HCP algorithm. The HCP algorithm also benefits from a reduction in the number of accesses to slower global memory.

The rest of the paper is arranged as follows. In section 3.5.2, we briefly describe the HCP multiscale algorithm, the distance-dependent-dielectric implicit-solvent model used for sim-
ulation, and the GPU platform. Then we discuss the mapping of the HCP algorithm to the GPU in section 3.5.2.3. In section 3.5.3, we describe the GPU and CPU platforms, and the structures and protocols used for testing. Finally in section 3.5.4, we discuss and analyze the results of our tests. Our findings are summarized in section 3.5.5.

3.5.2 Methods

A key objective of this study is to test the ability of combining multiscale approximations and a GPU to realize multiplicative speedups in molecular dynamics. For the purpose of this study we chose the hierarchical charge partitioning (HCP) multiscale approximation (Sec. 3.2.2.1) and accelerated it using the NVIDIA GPU platform. The software is implemented using NAB, the open source MD module in AmberTools version 1.4. The performance of the implementation was tested using the distance-dependent-dielectric implicit solvent model. The HCP approximation, the distance-dependent-dielectric model, and the NVIDIA GPU platform are briefly described below.

3.5.2.1 Distance-Dependent-Dielectric Implicit Solvent Model

Long range electrostatic interactions, which scale as $O(N^2)$, are the primary computational bottleneck in molecular dynamics simulations. Implicit solvent models, which use an approximation for computing long range interactions, reduce this computational cost considerably by analytically representing solvent atoms as a continuum. Solvent atoms typically represent a majority of the atoms in the system. Among other benefits, implicit solvent models can sample conformation space much faster and instantaneously incorporate the effect of dielectric changes in the solvent due to changes in the solute charge distribution. For this study we used a simple implicit solvent model, the sigmoidal distance-dependent-dielectric model. This model computes the long-range electrostatic potential $\phi$ at a distance of $r$ from a charge $q$ as

$$\phi = \frac{q}{\epsilon(r)r}$$ (3.40)

$$\epsilon(r) = D - \frac{(D - 1)}{2} \left[ (rS)^2 + 2rS + 2 \right] e^{-rS}$$ (3.41)

where $\epsilon(r)$ is the distant-dependent-dielectric function, and $D (= 78)$ and $S (= 0.16)$ are constants.

3.5.2.2 GPU Architecture and Programming Interface

Graphics processing units (GPUs) have traditionally been used to accelerate image rendering. However, evolution of the GPU into a compute-capable parallel processing platform has assisted in its adaptation to speed-up computations in various data parallel applications. We used state-of-art NVIDIA GPUs to speedup the computation of the molecular dynamics
simulation algorithms – the HCP and the distance-dependent-dielectric model – described above. These algorithms are implemented using the Compute Unified Device Architecture or CUDA programming interface.

NVIDIA Tesla GPUs consist of 240-512 execution units, grouped into 16-30 Streaming Multiprocessors (SMs). Each SM can run up to a thousand threads, thereby, enabling massively parallel computation. Multiple threads on a GPU execute the same instruction and hence, is a Single Instruction Multiple Thread (SIMT) architecture. This is what makes GPUs very suitable for applications that exhibit data parallelism, i.e. the operation on one data element is independent of the operations on other data elements. Therefore, it is well suited for molecular dynamics where the force on one atom can be computed independently of all others.

On NVIDIA GPUs, threads are organized into groups of 32, referred to as a warp. When threads within a warp follow different execution paths, such as when encountering a conditional, a divergent branch takes place, thus, affecting performance. Furthermore, GPUs have more transistors devoted to performing computations than for caching and managing control flow. This means that on a GPU, computations are much faster compared to a typical CPU, but memory accesses and divergent branching instructions are slower. Effect of slower memory access is mitigated by initiating thousands of threads, such that when one of the threads is waiting on a memory access, other threads can perform meaningful computations.

Every GPU operates in a memory space known as global memory. Data which needs to be operated on by the GPU, needs to be first transferred to the GPU. This process of transferring data to GPU memory is performed over the PCI-e bus, making it an extremely slow process. Therefore, memory transfers should be kept to a minimum to obtain optimum performance. Also, accessing data from the GPU global memory entails the cost of 400-600 cycles and hence, on-chip memory should be used to reduce global memory traffic. On the GT200 architecture, each SM contains a high-speed, 16KB, scratch-pad memory, known as the shared memory. Shared memory enables extensive re-use of data, thereby, reducing off-chip traffic. Whereas on the latest Fermi architecture, each SM contains 64KB of on-chip memory, which can be either be configured as 16KB of shared memory and 48KB of L1 cache or vice versa. Each SM also consists of a L2 cache of size 128KB. The hierarchy of caches on the Fermi architecture allows for more efficient global memory access patterns.

CUDA provides a C/C++ language extension with application programming interfaces (APIs). A CUDA program is executed by a kernel, which is effectively a function call to the GPU, launched from the CPU. CUDA logically arranges the threads into blocks which are in turn grouped into a grid. Each thread has its own ID which provides for one-one mapping. Each block of threads is executed on a SM and share data using the shared memory present.

3.5.2.3 Mapping HCP onto the GPU

In this section we discuss the implementation details of the HCP algorithm on a GPU. We also describe our approach for mitigating the performance bottlenecks to achieve near multiplicative speedup due to the combination of algorithmic efficiency of HCP and the
Figure 3.25: Mapping of HCP onto the GPU. Every SM of the GPU executes thousands of threads. Each GPU-thread is assigned with the task of computing electrostatic forces at one atom and updating that atom’s coordinates. Data pertaining to molecular complexes, subunits and groups are stored in the slow global memory whereas atomic data is stored in the faster shared memory. After the atomic coordinates have been updated, the new coordinates are transferred back to the CPU memory, as and when required for the trajectory to be stored.

3.5.2.3.1 Implementation

The primary step in parallelization is the identification of the parts of application that can be performed in parallel. As per Amdahl’s Law, the part that amounts to maximum execution time should be parallelized. Informally, one should make the common case faster. In molecular dynamics, calculation of non-bonded interactions is the primary computational bottleneck, due to its $O(N^2)$ scaling or as $O(N \log N)$ when approximation algorithms like HCP are used, where $N$ is the number of charges. Hence, computation of non-bonded forces was our obvious first candidate for the parallel implementation on the GPU.

The prerequisite for any GPU computation is the transfer of all data to the GPU memory. Therefore, the charges and coordinates of group, subunits and complexes, were first transferred to the GPU memory. Atomic coordinates and charges are stored in the per-SM shared memory to take advantage of their significant reuse, thereby improving performance of the memory subsystem. A kernel is then called from the CPU to compute the non-bonded forces in parallel on the GPU. As described in Section 3.5.2.2, it is necessary that thousands of threads are in execution at any given time to achieve better performance on the GPU. In any biomolecule, atoms are the most abundant and hence, we assigned each GPU thread to
compute the forces at one atom. The mapping of the HCP algorithm onto a GPU is depicted
in Figure 3.25.

The disadvantage to this approach is that the scaling of GPU performance hits a plateau
when the number of threads that can be executed simultaneously reaches the upper limit.
This limit is governed by the amount of per-thread register utilization by the implementation.
Increasing number of threads beyond this limit does not result in any improved performance
as the extra threads must to wait until some of the threads finish execution.

3.5.2.3.2 Optimization In section 3.5.2.2, we identified that the very slow data transfer
rates between the CPU and GPU over the PCI-Express bus is a major performance bottleneck
for GPU computation. To add to woes of the programmer, GPU global memory is of a limited
amount and hence, its judicious use is necessary. In this section, we describe how we tackled
each of these issues.

3.5.2.3.2.1 Minimizing CPU-GPU Data Transfers The computation of only non-
bonded forces on the GPU resulted in the large amount of to-fro data transfers between CPU
and GPU. This is because the calculated non-bonded forces are transferred back to the host
main memory in order to be summed up with the bonded forces that were computed on the
CPU. This summation of forces is necessary to compute the atom-coordinates for the next
time step of MD simulation. As evident, this approach involves extremely slow CPU-GPU
memory transfers at every step of the simulation, leading to performance degradation.

In order to reduce the number of CPU-GPU memory transfers, we computed both the
non-bonded as well as bonded interactions on the GPU. This approach mitigated the per
time-step memory transfers since, we could now calculate the new atomic coordinates on the
GPU itself.

3.5.2.3.2.2 Minimizing Memory Footprint on the GPU The above strategy of
minimizing CPU-GPU data transfers reduced latency, but it also led to a large memory
footprint on the GPU. In addition to charges and coordinates of molecular components, all
data required to compute bonded interactions, such as bond lengths and angles, were now
required to be in the GPU memory. This limited the size of the structure that could be
simulated. One of the largest data structures in our code was the atomic pair-list. The
atomic pair-lists contain a list of atoms, for which interactions are computed exactly, along
with the list of components for which the interactions are approximated. With the pair-lists
one can reduce computational cost by updating the list periodically, instead of at every step,
with the assumption that the list is unlikely to change significantly between updates. To
reduce memory utilization we eliminated the pre-calculated pair list, and instead determined
if a pair of coordinates should be included in the computation, at the point of computation.
Although this approach eliminates the option of reducing computational cost by periodic
updates of the pair-list, it significantly extended the structure size that could be simulated
and hence, enabled us to simulate the 475,000 atom virus capsid, which was not otherwise
possible.
3.5.2.3.2.3 Generating Random Numbers on the GPU  The implementation of CUDA that we used, i.e., CUDA 3.2, does not include a random number generator for the GPU. However, most molecular dynamics simulations involve a stochastic component, such as the Langevin dynamics method that is used to model solvent viscosity. We overcame this issue by generating all the required random numbers on the CPU and then copying them onto the GPU memory. This way we could use random numbers on the GPU.

Table 3.8: Overview of GPU Architectures

<table>
<thead>
<tr>
<th>GPU</th>
<th>Tesla C1060</th>
<th>Fermi Tesla C2050</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streaming Processor Cores</td>
<td>240</td>
<td>480</td>
</tr>
<tr>
<td>Streaming Multiprocessors (SMs)</td>
<td>30</td>
<td>16</td>
</tr>
<tr>
<td>Memory Bus type</td>
<td>GDDR3</td>
<td>GDDR5</td>
</tr>
<tr>
<td>Device Memory size</td>
<td>4096 MB</td>
<td>3072 MB</td>
</tr>
<tr>
<td>Shared Memory (per SM)</td>
<td>16 KB</td>
<td>Configurable 48 or 16 KB</td>
</tr>
<tr>
<td>L1 Cache (per SM)</td>
<td>None</td>
<td>Configurable 16 or 48 KB</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>None</td>
<td>768 KB</td>
</tr>
<tr>
<td>Double Precision Floating Point</td>
<td>30 FMA ops/clock</td>
<td>256 FMA ops/clock</td>
</tr>
<tr>
<td>Single Precision Floating Point</td>
<td>240 FMA ops/clock</td>
<td>512 FMA ops/clock</td>
</tr>
<tr>
<td>Special Function Units (per SM)</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Compute Capability</td>
<td>1.3</td>
<td>2.0</td>
</tr>
</tbody>
</table>

3.5.3 Test Setup

The GPU implementation described above, was tested on two NVIDIA GPUs and compared to the baseline CPU code running on the Host Machine. The Host Machine consists of an E5404 Intel Xeon CPU running at 2.00 GHz with 8 GB DDR2 SDRAM. The operating system on the host is a 64 bit version of Ubuntu 9.04 distribution running the 2.6.28-18 generic Linux kernel. Programming and access to the GPU was provided by CUDA 3.2 toolkit and SDK with the NVIDIA driver version 260.19.14. For the sake of accuracy of results, all the processes which required graphical user interface were disabled to limit resource sharing of the GPU.

We ran our tests on a NVIDIA Tesla C1060 graphics card with GT200 GPU and the NVIDIA Fermi Tesla C2050 graphics card. Overview of both these GPUs is presented in Table 3.8.

3.5.3.1 Test Structures

To test the scalability of our application, we used seven different structures ranging in size from 632 to 475,500 atoms. The characteristics of the structures used are presented in Table 3.9. The table also lists the threshold distances used for each level of HCP for each structure. These are the recommended threshold distances as described in Anandakrishnan et. al.3
Table 3.9: Characteristics of Structures.

<table>
<thead>
<tr>
<th>Structure</th>
<th>PDB ID</th>
<th>Atoms</th>
<th>HCP Threshold Distance (Å)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 bp B-DNA fragment</td>
<td>2BNA</td>
<td>632</td>
<td>15 n/a</td>
</tr>
<tr>
<td>immunoglobulin binding domain</td>
<td>1BDD</td>
<td>726</td>
<td>15 n/a</td>
</tr>
<tr>
<td>ubiquitin</td>
<td>1UBQ</td>
<td>1,231</td>
<td>15 n/a</td>
</tr>
<tr>
<td>thioredoxin</td>
<td>2TRX</td>
<td>1,654</td>
<td>15 n/a</td>
</tr>
<tr>
<td>nucleosome core particle</td>
<td>1KX5</td>
<td>25,101</td>
<td>21 90</td>
</tr>
<tr>
<td>microtubule sheet</td>
<td>1A6C</td>
<td>158,016</td>
<td>15 48</td>
</tr>
<tr>
<td>virus capsid</td>
<td></td>
<td>475,500</td>
<td>15 66</td>
</tr>
</tbody>
</table>

3.5.3.2 Molecular Dynamics Protocol

Unless otherwise stated, the following parameters and protocol were used for all simulations. The simulations use the sigmoidal distant-dependent-dielectric implicit-solvent model. The HCP threshold distances used are listed in Table 3.9. 6-12 van der Waals interactions for HCP were computed using only the atoms that are within the level 1 threshold distance, i.e., atoms that are treated exactly. The simulations used the Amber ff99SB force field. Langevin dynamics with a collision frequency of 50 ps$^{-1}$ (appropriate for water) was used for temperature control and the integration time step was 2 fs. Default values were used for all other parameters.

The simulation protocol consisted of five stages. First, the starting structure was minimized using the conjugate gradient method with a restraint weight of 5.0 kcal/mol/Å$^2$. Next, the system was heated to 300 K over 10 ps with a restraint weight of 1.0 kcal/mol/Å$^2$. The system was then equilibrated for 10 ps at 300 K with a restraint weight of 0.1 kcal/mol/Å$^2$, and then for another 10 ps with a restraint weight of 0.01 kcal/mol/Å$^2$. Finally, all restraints were removed for the production stage.

3.5.4 Results and Discussion

In this section, we present an analysis of the following: (i) speedup due to the GPU, the HCP and the combined speedup due to both, (ii) impact of divergent branching on speedup, (iii) limitation on structure size due to limited GPU memory, (iv) scaling with structure size, and (v) the stability of simulations. We show that near multiplicative speedups were achieved despite the introduction of additional divergent branching by the HCP algorithm. The largest structure that can be processed by our implementation is approximately 500,000 atoms, which is significantly larger than other GPU implementations. And, in our simulations, the use of single precision and the HCP approximation do not lead to gross instabilities.

1The microtubule structure was constructed as described in Wang and Nogales.
3.5.4.1 Speedup

In this section, we present the speedups obtained by our GPU implementation. Figures 3.26a and 3.26b depict the speedups obtained on two NVIDIA GPUs, i.e., Tesla C1060 and Fermi Tesla C2050 respectively. The figure presents four speedups: (i) speedup obtained due to GPU alone, (ii) speedup obtained due to HCP alone, (iii) the multiplicative limit which is the product of speedups due to GPU and HCP and lastly, (iv) the actual speedup that was realized due to the combination of HCP and GPU. For all these speedups, the baseline was a single-threaded, CPU, all-atom computation, without any approximation. Speedup due to the GPU was computed by comparing the execution times of the all-atom computation on the CPU with that of the all-atom computation on the GPU, without the use of HCP approximation in both cases. Speedup due to HCP was computed by comparing the execution times of an all-atom simulation with and without the HCP approximation on the CPU. The idealistic goal is to achieve multiplicative speedup due to the combination of GPU and HCP.

Figure 3.26 indicates that we are close to achieving the multiplicative limit in terms of speedup due to the combination of GPU and HCP. The key aspect to note is that the speedup achieved is closer to multiplicative limit for the larger structures. This is due to the fact that the larger structures are able to more efficiently utilize the GPU. For example, for the nucleosome core particle, the achieved speedup is within 3% and 8% of the multiplicative speedup, on C1060 and Fermi C2050 respectively, and for the virus capsid, the largest structure we tested, we are within 2.5% and 5%, on the C1060 and Fermi C2050 respectively. Another observation is that the performance improvement on the Fermi C2050 is better than on the C1060. This is as expected since the Fermi C2050 consists of L1 and L2 caches, which mitigates the impact of the random global memory accesses of the HCP algorithm. For memory accesses to be consecutive, all 32 atoms (atoms in a warp) need to follow similar execution paths, i.e., either they are approximated or not. However, due to the fact that each atom may meet different threshold requirements, it is extremely common to have random memory accesses.

We achieved near multiplicative speedup by the judicious optimization of our GPU application. Ryoo et al. point out that GPUs have a large optimization search space and hence, narrowing it down is imperative. We did so by first checking whether HCP is memory bound or compute bound and as presented in, HCP was found to be memory bound. Since, HCP is memory bound, we focused on those optimizations that would reduce the number of global memory transactions. We made a conscious effort to utilize the shared memory and constant memory available on GPUs, both of which help in the reduction of global memory accesses. We used shared memory to store the coordinates and atomic charges while the constant memory, which acts as a read-only cache, was used to store all the constants that were required to compute the forces and energies. We also kept the number of memory transfers to a minimum since, they use the slow PCIe and hence, lead to performance degradation. Use of atomic operations was at times necessary for us to mitigate race conditions in our application. Since, atomic operations negatively impact application performance, we re-factored our code so that use of atomic operations were not necessary. All these strategies, in combination, helped us achieve near multiplicative speedup.
3.5.4.2 Divergent branching

In our implementation, we assign each GPU thread with the task of computing the force at one atom of the molecule due to all other atoms in the molecule. From figure 3.2, one would intuitively expect that HCP introduces a lot of divergent branches on the GPU, which lead to performance degradation, as mentioned in section 3.5.2.2. However, HCP actually reduces the total number of divergent branches when compared to an all-atom simulation. This is because for an all-atom simulation, a conditional is required to determine the type of interaction (bonded or non-bonded), which needs to be carried out for every atom of the molecule. Our analysis indicates that for some structures, this conditional results in as many divergent branches as all other conditionals combined. The number of divergent branches were determined indirectly using the CUDA Visual Profiler\textsuperscript{207} provided by NVIDIA and are presented in Table 3.10. Looking at the difference in number of divergent branches, it
Table 3.10: Number of Divergent Branches and Global Memory Transactions

<table>
<thead>
<tr>
<th>No. of Atoms</th>
<th>Divergent Branches</th>
<th>Global Memory Transactions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>all-atom</td>
<td>HCP</td>
</tr>
<tr>
<td></td>
<td>all-atom</td>
<td>HCP</td>
</tr>
<tr>
<td>632</td>
<td>92,610</td>
<td>51,022</td>
</tr>
<tr>
<td>726</td>
<td>57,586</td>
<td>42,378</td>
</tr>
<tr>
<td>1,231</td>
<td>59,572</td>
<td>37,226</td>
</tr>
<tr>
<td>1,654</td>
<td>127,260</td>
<td>54,150</td>
</tr>
<tr>
<td>25,101</td>
<td>8,088,980</td>
<td>382,275</td>
</tr>
</tbody>
</table>

becomes apparent that said conditional increases the number of divergent branches by an order of magnitude.

Use of HCP mitigates the effect of this conditional. For distant components, the HCP algorithm does not reach the stage where it is necessary to determine the type of interaction (bonded or non-bonded).

The HCP also reduces the number of global memory transactions by reducing the number of atomic coordinates that need to be fetched. For distant components, coordinates of only the higher level component are required, bypassing all other molecular constituents. Table 3.10 portrays the number of per SM global memory transactions with and without the use of HCP and it can be seen that HCP results in an order of magnitude decrease in the number of global memory transactions. Being a memory-bound algorithm, HCP benefits from the reduction in number of global memory accesses. Therefore, the combined effect of reduction in the number of global memory transactions as well as reduction in the number of divergent branches brings about performance improvement of HCP on the GPU which is as follows:

\[
T_{HCP} = T_{NoHCP} - T_{gMem} - T_{divBranch}
\]

\[
\therefore T_{HCP} < T_{NoHCP}
\]

where \(T_{HCP}\) is the execution time with the HCP approximation on the GPU, \(T_{NoHCP}\) is the execution time without the HCP approximation on the GPU, \(T_{gMem}\) is the time for global memory transactions, \(T_{divBranch}\) is the time for divergent branching.

### 3.5.4.3 Memory Footprint

One of the constraints of GPU programming is dealing with the limited memory space available on the GPU. Due to this limitation, there arises a trade-off between the performance benefits that can be obtained on the GPU and the size of the structure that can be simulated. For example, current CUDA-based implementation of the MD module (pmemd) of AMBER molecular modeling package (AMBER GPU), focuses on maximizing speedup on the GPU, whereas, our implementation tries to maximize speedup while being able to handle large structures. Hence, the largest structure that AMBER GPU can simulate is \(1/20^{th}\) of what we can simulate, albeit the AMBER GPU has a higher speedup than our GPU implementation of NAB running without HCP.
<table>
<thead>
<tr>
<th>No. of Atoms</th>
<th>Amount of Memory Used (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>632</td>
<td>0.43</td>
</tr>
<tr>
<td>726</td>
<td>0.50</td>
</tr>
<tr>
<td>1,231</td>
<td>0.83</td>
</tr>
<tr>
<td>1,654</td>
<td>1.13</td>
</tr>
<tr>
<td>25,101</td>
<td>16.88</td>
</tr>
<tr>
<td>158,016</td>
<td>107.06</td>
</tr>
<tr>
<td>475,500</td>
<td>323.46</td>
</tr>
</tbody>
</table>

AMBER GPU uses *scratch arrays* to maximize speedup whereas we took advantage of the HCP algorithm to produce even higher speedup, and at the same time are able to handle much larger structures, though at the potential price of reduced accuracy. We also reduced memory utilization by using the strategy of not storing any sort of *pair-list* on the GPU and hence, were able to simulate much larger structures. Table 3.11 depicts the amount of GPU memory our implementation requires. To simulate the 25,000-atom nucleosome core particle, our implementation uses only 17 MB of GPU memory. The nucleosome is the largest structure that the AMBER GPU implementation can simulate on the Fermi C2050 GPU, whereas on the NVIDIA GTX295 GPU, with 896 MB of memory, the memory is insufficient for running the nucleosome.

The minimal memory requirement of our GPU implementation lets us run molecular dynamics simulation of nucleosome core particle even on an Apple MacBook Pro laptop, which is not possible using the AMBER GPU program.

### 3.5.4.4 Scaling

In figures 3.27a and 3.27b, we show the scalability of our implementation with respect to structure size (number of atoms). The absolute speedup obtained is higher for the Fermi Tesla C2050 GPU due to (i) the greater number of processing cores and (ii) presence of L1 and L2 caches that improve the performance of the memory subsystem.

From figures 3.27a and 3.27b, it is clear that, as a function of structure size, the speedup due to the GPU grows fast at first, then levels off after the structure size is increased into the range of $\sim 10^4$ atoms. This behavior can be explained as follows. In order to realize the full potential of a GPU, its occupancy needs to be high, i.e., none of the processing cores of the GPU should be idle. Better occupancy leads to better performance. In our case, once the size of the structure reaches a several thousand atoms, occupancy of the GPU reaches its maximum, as described in Section 3.5.2.3.1. Beyond which, increasing the size does not increase the speed up as much, since the additional atoms need to “wait” for the computation on the preceding atoms to finish, only after which they can be operated upon. Performance on Fermi C2050 is better than C1060 as it allows for a greater number of threads to be executed simultaneously, primarily due to the presence of a larger register file. At the same
time, the speedup due to the HCP continues to increase with the increase in number of atoms in the molecule. This is because, the HCP algorithm scales as $O(N \log N)$, where $N$ is the number of atoms. Hence, it is almost entirely due to the benefits of HCP, that both the ‘multiplicative limit’ as well as the ‘achieved’ speedup continue to increase as the system size is increased beyond $\sim 10^4$ atoms.

![Graph](image)

(a) NVIDIA Tesla C1060

![Graph](image)

(b) NVIDIA Fermi Tesla C2050

Figure 3.27: GPU-NAB Scalability. Speedup due to the GPU peaks off after a certain thousand atoms due to the inability to launch more threads. Speedup due to HCP continues to increase with the increase in the number of atoms and hence, scales as $O(n \log n)$. The lines are shown only to visually guide the eye.

### 3.5.4.5 Stability in MD simulations

We have performed a standard all-atom 2 ps long ($10^2$ steps) constant temperature (300K) implicit solvent simulation for each of the structures list in Table . The simulation was extended to 50 ns ($2.5 \times 10^7$ steps) for the four smallest structures, ranging in size from 623 atoms (a 12-base-pair long fragment of B-DNA) to 1654 atoms (protein thioredoxin). No numerical instabilities were noticed. For the four smaller structures, further analysis revealed
that, in all cases, the RMS deviation from the original X-ray structure was stable within several nanosecond. For the three proteins, the RMS deviation was within the range of about 2 Å expected for implicit solvent molecular dynamics. For the B-DNA fragment, the RMS deviation was considerably larger than typically expected, \( \sim 5-6 \) Å instead of \( \sim 2 \) Å. However, the same deviation was observed for a control MD run on a CPU with double precision and without the HCP, showing that the larger than expected RMS deviation for the B-DNA fragment was due to the use of the simplified solvent model (distance-dependent dielectric) rather than inherent instabilities of either the HCP algorithm or its GPU implementation. That, of course, does not mean that the single precision arithmetic used here for the GPU implementation is completely “safe” to use, but that any errors due to the use of single precision on the GPU or the errors due to HCP do not accumulate or/and combine in a way to produce gross instabilities in the MD simulation, at least for the structures tested here. However, there may be more subtle ways in which the single precision arithmetic may skew the results of MD simulation over millions of steps, that may not be evident from the RMS deviation metric. These issues are well beyond the scope of this work that focuses on demonstrating the ability to achieve multiplicative speedup from the combined use of the HCP and GPU. On GPU cards such as C1060 the performance loss of about a factor of ten in speed due to the use of double vs. single precision certainly warrants closer examination of the possibility of the use of single precision arithmetics in practical MD simulations. At the same time, for GPU cards such as C2050 the gain of only about a factor of two in speed due to single precision vs. the tried-and-true double precision may not be worth potential risks or even efforts associated with thorough assessment of those risks.

### 3.5.5 Conclusion

Molecular dynamics simulations are routinely used to analyse the structure of biomolecules, and to study their functional activities such as ligand binding, complex formation and proton transport. However, the timescale of simulations necessary for any meaningful insight into such processes is much greater than that achieved via atomistic molecular dynamics. Therefore, it is imperative to extend these timescales, which can be done by (i) parallelization across multi- and many-core processors, and (ii) use of approximation algorithms. In the present work, we combine the two techniques. Specifically, we parallelize molecular dynamics simulations using graphical processing units (GPUs), and use the hierarchical charge partitioning (HCP) approximation.

Presence of asynchronous computations in approximation algorithms make these algorithms less than ideal candidates for implementation on the GPU platform. Hence, there is an expectation that the combination of these two techniques would not result in multiplicative speedups, i.e., total application speedup being the product of speedup due to each technique. However, our hybrid approach of the combination of HCP and GPUs does result in almost multiplicative speedups. For example, for the 25,101-atom nucleosome and the 475,500-atom virus capsid, the difference between multiplicative speedup and the actual speedup realized is only 8% and 5% respectively. The near-multiplicative speedup achieved, despite the additional asynchronous computations introduced by the HCP due to the additional
divergent branching, is due to two factors. One, the HCP eliminates a number of other divergent branches that would have been executed without the HCP. Two, the HCP reduces the number of slow global memory accesses.

Due to the limited amount of GPU memory available (~4 GB), its efficient use is an additional challenge that has to be dealt with while implementing applications on GPUs. Unlike typical implementations of MD algorithms, we do not use pre-calculated pair-lists for identifying interacting charges. Instead, we determine, at the point of computation, if two charges should be included in the computations. Eliminating the pair-list significantly reduces memory utilization allowing us to simulate much larger structures, such as the 475,500 atom virus capsid.

The results shown here are for the distant-dependent-dielectric implicit solvent model. We plan to extend our implementation to use more widely used implicit solvent models such as the generalized Born model. The software code for our implementation is available upon request.
Chapter 4

Overall conclusions

Atomistic molecular dynamics is routinely used to study the function and activity of biomolecules where experimental investigation is expensive or infeasible. However due to computational costs, such simulations are generally limited to system sizes and durations that are much less than needed for most biologically relevant system sizes and time scales. Two of the most demanding computations in atomistic molecular dynamics simulations are (a) the computation of the charge state of ionizable sites, and (b) the computation of long range electrostatic interactions. The computation of charge state scales as $O(2^N)$ in the number of ionizable sites, making the computation intractable for $N \gtrsim 50$. The computation of long range electrostatic interactions scales as $O(n^2)$ in the number of atoms, but due to the large number of simulation steps ($10^{12-16}$) required for meaningful simulations, the simulation of most realistic systems ($n \gtrsim 10^6$) are impractical.

This work describes two novel methods with the potential for extending the accessible structure size and duration of atomic level molecular dynamics (MD). The direct interaction approximation (DIA) speeds up the computation of the charge state for ionizable sites in biomolecular structures – a key input for practical MD simulations – from $O(2^N)$ to $O(N^2)$ in the number of sites. The hierarchical charge partitioning (HCP) approximation speeds up the the computation of long range electrostatic interactions from $O(n^2)$ to $O(n \log n)$ in the number of atoms.

A number of approximation methods exist for speeding up the computation of average charge state of ionizable sites in biomolecules. One class of such methods, clustering algorithms, have been relatively unexplored in the context of biomolecular modeling, despite promising results. To quantify the potential usefulness of the clustering algorithms for biomolecular modeling, a rigorous error analysis was performed. Two basic clustering algorithms – local and global clustering – were considered for the purpose of this study. A strict error bound was derived. This computationally inexpensive error bound was found to be correlated to root mean square error, and therefore could be used as a metric for determining if the potential error due to the clustering algorithm may be acceptable for a given problem. On average for a test set of twelve representative biomolecular structure, the error in the average charge state calculated by the local clustering algorithm was found to be within the range
of experimental error. Thus the local clustering algorithm may be accurate enough for the computation of average charge state of ionizable sites in biomolecules.

The error analysis of the clustering algorithms also showed that local clustering can be significantly more accurate than global clustering. Since local clustering builds clusters based on the strength of direct interactions (pairwise interactions involving the site of interest), this result suggested that direct interactions may be more important than indirect interactions (pairwise interactions not involving the site of interest) for certain applications. A novel algorithm, direct interaction method (DIA), was developed based on this insight. The DIA computes the contribution of direct interactions \textit{exactly}, while using an average value for indirect interactions. The exact computation of direct interactions however, involves the computation of combinatorial sums of products known as elementary symmetric functions, in which the number of terms grow exponentially with problem size. A recursive generating function, Newton’s identity, can be used to compute the elementary symmetric function in \(O(N^2)\) operations. However, the Newton’s identity results in large numerical errors due to catastrophic cancellations (loss of precision when calculating small differences between very large numbers).

A binary split-merge algorithm was developed to compute the elementary symmetric functions without the subtractions that produce catastrophic cancellations. This algorithm first partitions the system into a hierarchical binary tree where the root node contains the entire system and each leaf node contains a single particle. Starting with the leaf node, the algorithm then merges the elementary symmetric function for the two child nodes to compute the elementary symmetric function for the parent node.

The direct interaction approximation (DIA) using the binary split-merge algorithm was tested on two problems. One, the computation of thermodynamic properties of the 2D Ising model with no external field. And two, the computation of average charge state of ionizable sites in biomolecules. Accuracy (as measured by RMS error relative to the exact computation) of the DIA was compared to that of the commonly used Monte Carlo method. For the same computational cost, the computation of thermal average magnetization and heat capacity of the 2D Ising model by the DIA was found to be more accurate the Monte Carlo method. For the computation of internal energy of the 2D Ising model, and the computation of average charge state of ionizable sites in biomolecules, the accuracy of the DIA was comparable to the Monte Carlo method. These results suggest that with additional validation and analysis, the deterministic DIA could be a practical alternative to the non-deterministic Monte Carlo method for some problems.

The second computational bottleneck in atomistic molecular dynamics, is the computation of long range electrostatic interaction at each step of the simulation. Several approximation exist for speeding the computation of long range electrostatic interactions for biomolecular simulations, of which the two most widely used approximations are the particle mesh Ewald (PME) and the spherical cutoff methods. The simple and robust spherical cutoff method can be highly inaccurate since it completely ignores interactions beyond a cutoff distance. The PME, developed for explicit solvent models (solvent atoms are modeled explicitly), can not be easily extended to the most commonly used implicit solvent model for molecular dynamics - the generalized Born (GB) model. Implicit solvent models treat the effect solvent atom
analytically, which offers a number of benefits including lower computational cost and faster conformational search.

A novel approximation, hierarchical charge partitioning (HCP), was developed, which can be easily extended to implicit solvent models and does not completely ignore any long range electrostatic interaction. Like the PME and the spherical cutoff methods, the HCP scales as $O(n \log n)$ in the number of atoms. The HCP partitions biomolecular structures into multiple levels of hierarchical components. The charge distribution for components that are distant from the point of interest are then approximated by a small number of charges. While the exact charge distribution is used for nearby components.

This HCP approximation was tested on representative structures ranging in size from 600 atoms to 3 million atoms. Accuracy was measured as relative RMS error in force and energy compared to the all-atom computation without any further approximations. For gas phase single-point computations the HCP was found to be on average comparable in accuracy to the more complex industry-standard particle mesh Ewald method, and significantly more accurate than the spherical cutoff method. For the implicit solvent single-point computations, the HCP was found to be more accurate than the spherical cutoff method for the computation of forces, while being comparable in accuracy when it comes to the computation of energy.

More importantly, there are important qualitative differences between the HCP and the spherical cutoff method. The spherical cutoff method ignores charges beyond the cutoff distance while the HCP approximates the influence of distance charges. Our testing using the implicit solvent GB model suggests that this difference can have a significant impact on details of the dynamics. For example, for 50 ns simulations of four small structures, the residue flexibility and $\chi_1$ and $\chi_2$ angles for the spherical cutoff simulations show larger deviations from the reference GB simulation than the HCP simulations. Similarly, a 10 ns simulation of a 30 base-pair DNA strand showed that the flexibility of the molecule, as measured by end-to-end distance, using the HCP method was similar to that of the reference GB simulation, whereas the spherical cutoff method results showed a more flexible molecule. And a series of simulations of the nucleosome core particle showed that with the reference GB and HCP methods all of the positively charged tails of the histone chains collapsed onto the negatively charged DNA, whereas two of the histone tails failed to do so with the spherical cutoff method.

Due to its multiscale nature, the HCP method can violate Newton’s 3rd law resulting in a residual center of mass force and torque. For the structures tested here, the effect of the residual force and torque is much smaller than the “noise” due to stochastic collisions used in constant temperature simulations with strong coupling to a thermal bath. However, when a weak coupling is used to increase the sampling of conformational space, the residual force and torque may cause the structure to drift and rotate, making it inconvenient for visualization and analysis. For simulations with weak coupling to a thermal bath, the center of mass motion and rotation can be eliminated by using a velocity correction. The multiscale nature of the HCP can also result in discontinuities at threshold boundaries, which can cause energy not to be conserved. The discontinuity and the resultant nonconservation of energy for the HCP method is however much smaller than that of the cutoff method. Smoothing functions
can be used to reduce the discontinuities and the non-conservation of energy. However, we found that increasing the threshold distance may be a more effective way of achieving the same result.

An analysis of 50 ns implicit solvent GB simulations of four structures – B-DNA, immunoglobulin binding domain, ubiquitin and thioredoxin – shows that the results for the HCP simulation are in reasonable agreement with the reference GB simulation without cut-offs. To demonstrate a practical application of the HCP method, we used it to refine a 348,000 atom chromatin fibre. The 15 ns all-atom simulation successfully resolved numerous severe steric clashes, significantly improving the quality of the starting structure. In general, our findings suggest that, compared to the spherical cutoff, the HCP method may always be the preferable approach for speeding up pairwise GB computations for molecular dynamics.

The HCP has been implemented in the open source molecular dynamics software, NAB, in AmberTools v1.5, and is available for general use. The implementation in NAB includes the ability to run on multiple processors using the open MPI protocol. However the speedup of the HCP peaks at approximately 64 processors, most likely because the interprocessor communication cost exceeded the speedup due to parallelization. Therefore, we explored the implementation of HCP on the graphical processing unit (GPU) platform with hundreds of processor cores on a single chip, which may not incur the high interprocessor communication cost. Moreover, with the GPU one can take advantage of the inherent parallelism in molecular dynamics code on a workstation, without the need for a supercomputer cluster.

The HCP was first implemented in the GEM open source software for computing electrostatic potential on the surface of biomolecules. The GEM-HCP software was then implemented on an ATI Radeon 4870 GPU. We show that, for large biomolecular structures, combining the power of the GPU with the HCP approximation can achieve a combined speed-up of up to 934x over the reference computation based on a single processor without the use of the HCP. This combined speed-up is larger than the individual speed-ups for structures larger than about 10,000 atoms, and for the largest structures tested is many times larger than what could have been achieved by the GPU (182x) or HCP (42x) alone.

Presence of asynchronous computations in approximation algorithms generally make algorithms such as the HCP, less than ideal candidates for implementation on the GPU platform. Hence, there is an expectation that the combination of these two techniques would not result in multiplicative speedups, i.e., total application speedup being the product of speedup due to each technique. The HCP implementation in the NAB molecular dynamics software was ported to the NVIDIA GPU platform to determine if multiplicative speedups could be achieved. With some key optimizations, we were able to achieve near multiplicative speedups. For example, for the 25,101-atom nucleosome and the 475,500-atom virus capsid, the difference between multiplicative speedup and the actual speedup realized is only 8% and 5% respectively. The near-multiplicative speedup achieved, despite the additional asynchronous computations introduced by the HCP due to the additional divergent branching, is due to two factors. One, the HCP eliminates a number of other divergent branches that would have been executed without the HCP. Two, the HCP reduces the number of slow global memory accesses.
In conclusion, this work has produced two novel methods, direct interaction approximation for speeding up the computation of the charge state of ionizable sites in biomolecules, and hierarchical charge partitioning for the computation of long range interactions in implicit solvent molecular dynamics. These methods have been shown to achieve comparable speedups while being equally or more accurate than the current, most commonly used approximations for these applications. Thus these methods represent a practical alternative to the currently used approximation, namely the Monte Carlo method for calculating charge state, and the cutoff GB method for implicit solvent molecular dynamics.
Bibliography


