FEARCF: Graph-based Software Library for Multidimensional Free Energy Simulations

---

Tomás Bruce-Chwatt

Supervisor: **Prof Kevin J. Naidoo**

University of Cape Town

Abstract

This thesis presents the efforts to adapt the Free Energy from Adaptive Reaction Coordinate Forces (FEARCF) method into the form of a software library, enabling it to be interfaced with other molecular dynamics (MD) software packages. There exist many methods to calculate the free energy of molecular systems, an important quantity when studying chemical reactions and molecular structures. One of these, FEARCF, was developed at the Scientific Computing Research Unit (SCRU). Previously, this method was restricted to usage within the CHARMM MD software package. Taking inspiration from graph theory and object-orientated design, a successful software library implementation will be demonstrated by presenting results from a range of theories including: classical, *ab initio* and QM/MM. Simulations are conducted for a range of systems and reaction coordinates including: water, glucose, and the GlcNAc/OGT enzyme-substrate complex.

Declaration

I declare that this dissertation, titled **FEARCF: Graph-based Software Library for Multidimensional Free Energy Simulations**, is a presentation of my original research work done at the Scientific Computing Research Unit, Department of Chemistry, University of Cape Town, South Africa. No part of this thesis has been submitted elsewhere for any other degree or qualification. Whenever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgment of collaborative research and discussions.

Tomás Bruce-Chwatt

TBC

Acknowledgements

I would first of all like to sincerely thank all my funders for providing me the opportunity to pursue this degree, including the South African Research Chair Initiative (SARChI), the National Research Foundation, and the UCT Post-Grad Funding Office.

I would then like to thank my supervisor, Prof Kevin Naidoo, for his endless patience, drive and willingness to share his knowledge and expertise in the field.

I also wish to thank the many staff and students that I have gotten to know in my time at the Scientific Computing Research Unit (SCRU). Special thanks go to the unit's secretary, Lydia Dreyer, for all the invaluable help she has been during my time at SCRU.

With particular interest to the work in this thesis, I would like to thank Ian Rogers for his initial work on the FEARCF library which formed a valuable starting point for my continued developments.

Thanks as well to the fellow academics in the SCRU unit, Dr. Chris Barnett and Dr. Gerhard Venter, for their contributions and suggestions.

I would particularly like to acknowledge my fellow students Jess Nel, Tharindu Senapathi, Daniel Flowers, Malcolm Hillebrand, and Carla Coetzee, for all their efforts, both professional and personal, to help me finish this degree.

Thanks as well to my roommate, Andi Columbo, for her contributions to the editing of the language and structure of my thesis.

A huge thanks to all my friends and family, too numerous to name, who have helped me along the way.

I would especially like to thank my parents, Wendy and Andrew, for their continued support and encouragement.

Lastly, I would like to thank my faithful dog Ollie, for keeping me sane during the most stressful of times.

Contents

List of Figures

Chapter 1

Introduction

This thesis aims to present the packaging of the Free Energy from Adaptive Reaction Coordinates Forces (FEARCF) method in a graph-based, object orientated software library and its successful implementation in several molecular dynamics software packages. This introduction will provide a brief summary of several of the important concepts just mentioned and then give the structure of the overall thesis.

1.1 Scientific Computing

This section will explore the advent of scientific computing as a new way of conducting science, as well as why it has become a useful tool for studying biological systems at the molecular level.

The advent of computing machines and the acceleration of their development in the second half of 20th century, gave rise to a new approach to conducting science that is distinct from the classical dichotomy of theory versus experiment. This new approach is known as scientific computing. It was born at the interface of modern computing, applied mathematics, statistics, domain science and engineering [1]. Figure 1.1 illustrates how scientific computing utilises tools from all of the aforementioned fields. Computational modelling takes knowledge of the physical world from the many domains of science and engineering and translates them into the language and structure of modern computers. This creates a model of any given system that is reliable, accurate and most importantly, informative beyond the scope of previous traditional models. To do this, numerical methods are drawn from the developments of applied mathematics and used in conjunction with computational resources in order to produce results faster and in greater numbers than any human could produce in a lifetime. The ability to produce such huge amounts of data can further be leveraged by drawing approaches from statistics and domain science in the form of statistical mechanics, which informs how useful information about a system of interest can be extracted from this large volume of

data. This approach has found great success in the fields of physics, biology, chemistry, astronomy, engineering, finance, to name a few.

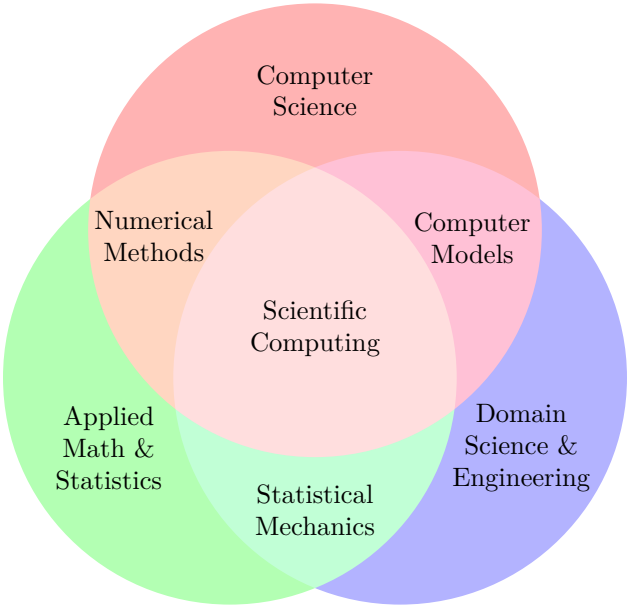


Figure 1.1: Scientific Computing as the intersection of Computer Science, Applied Mathematics & Statistics, and Domain Science & Engineering.

Scientific computing approaches have been used in the fields of chemistry and biology for decades, with applications in bioinformatics, data analysis and systems biology. Arguably the most important application of scientific computing in chemistry is the computational modelling of molecules, with the 1998 [2] and 2013 [3] Nobel Prize in Chemistry being awarded for advancements in this area. At a fundamental level, chemistry is the study of atoms and the bonds that form or break between these atoms, which in turn form molecules and perform chemical reactions. While techniques such as x-ray crystallography and nuclear magnetic resonance (NMR) have been developed in order to study the structure of molecules directly, the direct study of chemical reactions has proven to be far more challenging. Reactions typically involve the exchange (complete or partial) of electrons and occur on small length scales (10^{-10}m) and incredibly short time scales ($10^{-9} - 10^{-12}s$). While reactions between simple molecules can be studied using experimental techniques such as ultrafast laser spectroscopy [4], many reactions that occur between more complex molecules cannot. These include enzymatic reactions, or reactions which take place within complex solvent, molecular, and cellular environments. Even for those systems which have been studied experimentally using techniques like Kinetic Isotope Effect [5], the techniques available fail to provide key information about reaction mechanisms or transition state structures. With no experimental means to study these complex reactions in detail, this has provided an opportunity for the development of computa-

tional approaches to model these systems. It is these methods which have now become standard within the field.

It has now been shown how scientific computing provides a way to fill a gap made by the limitations of current experimental techniques in chemistry and biology. Now greater detail will given about the exact scientific computing methods used to study molecular systems.

1.2 Molecular Simulations

Here we describe the various scientific computing methods used to model molecular systems at varying levels of theory.

The classical approach to modelling molecules is based upon the 'ball and spring' model in which each atom within a molecule is represented as a ball and the bonds between atoms are modelled as springs. The primary bond potential is modelled with a harmonic potential along with torsional terms. Non-bonded atoms still interact therefore these are modelled with an electrostatic potential as well as a van der Waals potential. The exact form of the mentioned potentials depend upon the specific force field but they are typically parametrised using experimental values. The actual movement of atoms is simulated using numerical methods either from the random sampling based Monte-Carlo (MC) approaches [6], or from approaches based upon Newtonian Mechanics such as Euler, leapfrog and Velocity-Verlet integration. These methods are referred to as Molecular Dynamics (MD) [7]. MD allows for the use of parallel calculations, because each atom can be updated independently. This approach is successfully used to model many equilibrium systems. It is not, however, applicable for all systems as extreme bond distortion and, more importantly, bond breaking and formation cannot be modelled using this classical approach.

In order to study the changes in the electronic structure of molecules, classical theory needs to give way to quantum mechanics. Hartree-Fock (HF) theory models molecules using the Born-Oppenheimer approximation where the molecular nuclei are modelled as point charge particles. The molecular electrons in HF are modelled as the product of one-electron wave functions to form an electron 'cloud'. The evolution of this electronic structure is predicted by solving the Schrödinger Equation. Other methods, known as post-Hartree-Fock methods, add effects that Hartree-Fock ignores such as electron correlation. Density functional theory (DFT) re-expresses many of the terms in the Schrödinger Equation as functions of the electron density with the addition of an exchange-correlation energy correction term. Moreover, Møller-Plesset uses perturbation theory to provide corrections to the HF energy. All these methods are considerably more computational expensive than their classical counterparts [8].

While small systems may be more easily studied using *ab initio* methods such as DFT, larger systems with hundreds of atoms become increasingly time-consuming to simulate. For some systems, however, the region of space in which the electronic structure is significantly changing may be much smaller than the system itself. This presents the possibility of dividing the system into two regions: a classical region and a quantum region. Methods that take this approach are known as QM/MM methods. The interaction between the classical and the quantum regions are fairly simple to model. Things become more complicated if there are bonds that cross the boundary between the two regions, or if classical atoms are too close to the quantum region. Methods such as link atoms, frozen orbitals, and non-point charge nuclei have been developed in an attempt to accommodate these complications [9].

Now that the possible ways to model a molecular system have been given, we will now consider how they can be modified in order to calculate the important quantity of free energy.

1.3 Free Energy Methods

The concept of free energy is explained in this section, as well as previous methods used to calculate it. The FEARCF method is then presented as an alternative by detailing its advantages as a free energy method.

The modelling of molecule systems and their dynamics is an incredibly useful tool, with trajectories and snapshots of these trajectories giving insight into the dynamic behaviour of a system at any one point in time. But additional tools are required if greater knowledge of the overall behaviour of the system independent of time is needed. Free energy is an essential quantity in statistical mechanics and is defined as the energy in a system that is capable of doing work. Free energy is therefore the energy within a system that is 'free' to transform without changing the nature of the system itself. Other thermodynamic properties are often calculated using free energy [10]. In the field of chemistry, free energy has most commonly been used when studying reactions where the difference in free energy between the reactants and the products dictates whether the reaction will be spontaneous or not. Other uses for free energy exists such as an indicator for reaction catalyst mechanisms, and predicting most likely molecular structure configurations [11].

Free energy is not easily calculable using traditional molecular simulation methods such as MC or MD because it requires proper calculation of the entropy. This calculation is dependent on having knowledge of the sampling in all regions of phase space. However, some regions of phase space may require too much energy for the system to reach, or the region may be separated from others by sufficiently large energy barriers that the system cannot cross. The design of MC and MD methods favours system configurations of low energy, and any configuration requiring energy greater than the available thermal

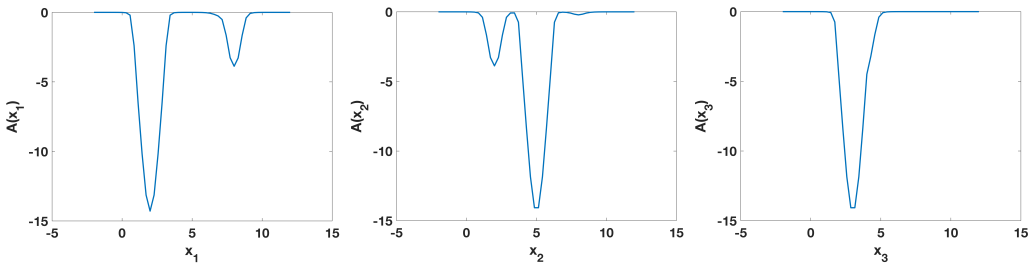
energy are unlikely to be sampled. To solve this problem, several free energy methods have been developed that modify existing MC and MD methods, allowing for the proper calculation of the relative free energy.

The first free energy method to use a biasing force was developed by Torrie and Valleau [12] and is referred to as Umbrella Sampling. In this approach the standard Boltzmann weighting in a MC simulation is replaced with a chosen biased weighting function that favours the system configurations with higher free energy. This approach can also been done in MD with the addition of a chosen biasing potential to the system Hamiltonian . If there are multiple free energy minima, then multiple biasing potentials overlapping each other will need to be used, hence the name 'umbrella' potential. This approach requires a good selection of a biasing potential *a priori*, and is therefore not always easily implementable for all systems. An extension of this approach, referred to as adaptive umbrella sampling, was proposed by Mezei [13], which varies the shape of the umbrella potentials over multiple simulation iterations in a continual effort to sample unsampled regions. Another free energy method is Metadynamics, developed by Laio and Parrinello [14], which uses a sum of gaussian potential terms which have their heights and widths adjusted over several simulation iterations.

FEARCF is a free energy method developed by the Scientific Computing Group and first utilised by Naidoo and Brady [15]. It was first formally described by Barnett and Naidoo [16]. FEARCF falls under the category of flat histogram methods because the ultimate aim is to reach a state in which the sum of the system and biasing potential allows for all configurations to be equally sampled. It does this by generating a biasing force from an initial estimate of the probability distribution of the system. Next, it iteratively updates this potential through various MD simulations in a similar manner to adaptive umbrella sampling. However, unlike adaptive sampling, the biasing potential is not constructed in terms of explicit potential terms but rather as a numerical potential, sourced from the probability distribution. When calculating the biasing force, the gradient of the biasing potential is then found using numerical interpolation techniques. This gradient is then transformed to be in terms of the atoms through the use of the chain rule and products of partial derivatives. This approach has a notable advantage over previous methods since probability densities of parallel simulations can be combined using histogram weighting approaches such as WHAM [17], increasing the overall speed of the free energy calculation. Additionally, the fact that FEARCF does not need Jacobian corrections, which are required when using an analytical potential, is another benefit of using this method. These Jacobian corrections can become particularly complicated when the free energy is defined in terms of multiple dimensions. The rational behind calculating multidimensional free energies is explored below.

1.3.1 Multidimensional Free Energy

Free energy studies for molecular systems are typically described in terms of one or two reaction coordinates. Defining the free energy difference in terms of this small number of variables means that any other potential variables are 'collapsed' or 'folded' into this definition of the free energy. This may have its advantages by simplifying a complex system allowing for easier study, however, the selection of which variables are chosen needs to be carefully considered. A non-optimal selection may lose critical chemical and physical information. It may also lead to potential oversimplification in the analysis and the misidentification of important features. A number of distinct system configurations may be conflated and energy minima may be reduced in size or even hidden completely.



(a) (b) (c)

Figure 1.2: Set of 1D free energy Boltzmann averaged curves of a 3D free energy volume $A(x_1, x_2, x_3)$ a) for x_1 b) for x_2 c) for x_3.

To illustrate the potential danger of reduced dimensionality, Figure 1.2 shows three different ways in which to reduce the dimensionality of a 3D free energy surface. The 3D free energy surface is given in Figure 1.3 and defined by the equation $A(x_1, x_2, x_3) = -2e^{-(x_1-8)^2-(x_2-7)^2-(x_3-3)^2} - 5e^{-(x_1-2)^2-(x_2-8)^2-(x_3-4)^2} - 15e^{-(x_1-5)^2-(x_2-2)^2-(x_3-3)^2}$. This 3D free energy A is an example free energy constructed with three gaussian potentials, each with distinct minima. These three minima have magnitudes of 2, 5, and 15. These are shown in Figure 1.3 as blue circular iso-surfaces. Figure 1.2a shows the free energy purely in terms of the variable x_1, with the dependance on the other variables 'folded' into this representation using a Boltzmann averaging approach. With this dimensionality reduction we can now only identify two minima instead of the three we know should exist. If we look at Figure 1.2b, which is defined in terms of x_2, we can distinguish all three minima however the smallest minima is much shallower than the magnitude of 2 that we expect. This is in contrast to the other minima which are much closer to their true values of 5 and 15. The worst possible case is shown in Figure 1.2c in which all three minima have now been reduced to only one, the one with the largest magnitude, and we have lost all information about the other two minima.

Figure 1.3 also illustrates potential issues with just reducing to two reaction coordinates. While the surfaces for $A(x_1, x_3)$ and $A(x_1, x_2)$ do show all three

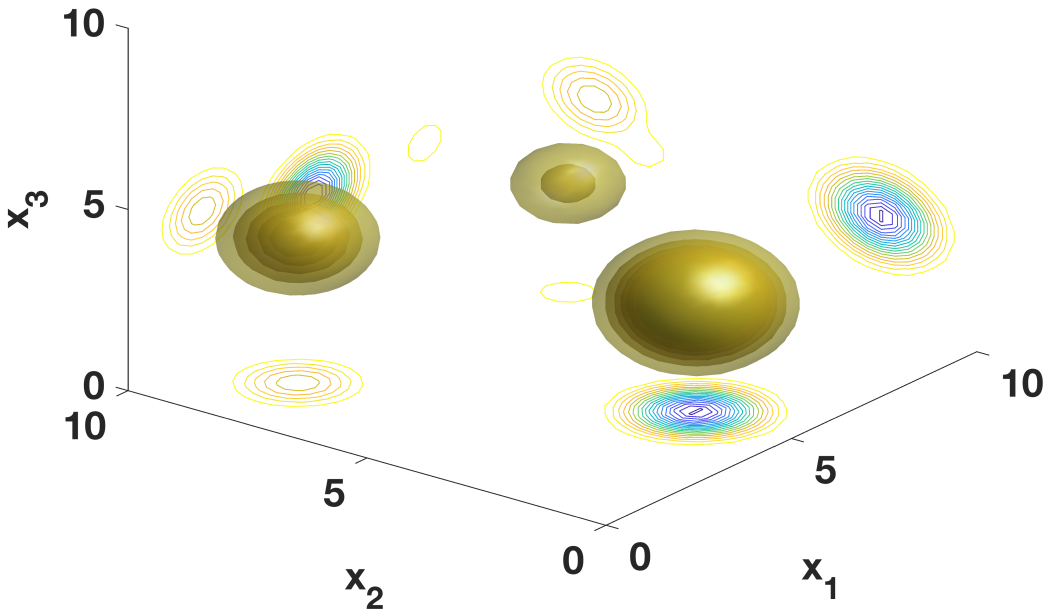


Figure 1.3: Example 3D free energy volume composed of a sum of 3 gaussian potential terms as well as 2D Boltzmann averaged free energy surfaces projected onto their associated axis.

minima, the surface for $A(x_2, x_3)$ loses the information about the lowest minima. So even when reducing to two dimensions, one must be careful about which reaction coordinates are chosen.

The above system is a fictitious one but it demonstrates the problems that can exist when studying real systems. Taking the deceptively complex structure of bulk water for example, it is known that a water molecule can form up to four hydrogen bonds with nearby water molecules. This is the cause of the hexagonal symmetry apparent in the ice crystals that make up a snow flake. The common reduction of the water free energy to a one dimensional function of the inter-water distance, completely loses the orientational information in the hydrogen bonds which make up the structure of bulk water. This has previously been explored using FEARCF by Strümpfer and Naidoo [18], and this will be expanded upon in Chapter 6.

The concept of free energy and why it is important to calculate has been explored, and the advantages of using the FEARCF method for calculating multi-dimensional free energy have been given. The next section will detail a new implementation of FEARCF that will widen its applicability and ease of use.

1.4 Software Libraries

Below is given the need for FEARCF to be packaged in a new manner, and why a software library is the obvious means to do that. The object-orientated

programming paradigm will also be detailed as a means for creating an optimal library structure.

The FEARCF method was originally developed inside a modified version of the classical MD software package CHARMM, and as discussed previously, is therefore limited to studying only a certain number of molecular systems. This excludes any systems involving reactions. In order for the FEARCF method to be usable in conjunction with *ab initio* or QM/MM methods, it would need to be reconstructed into a form that could be more easily integrated into other MD software with the ability to perform quantum mechanical simulations. This effort had been previously begun by Ian Rogers, however, this version lacked the ability to use reaction coordinates other than distances.

Software libraries have become a ubiquitous mode of importing methods into multiple pieces of software since the popularisation of the subroutine approach in FORTRAN II [19]. The ability to construct code in a modular fashion, ensuring consistency and ease of access, has made software libraries the *de facto* means in which to package a given method. A key consideration when designing a software library is which overall programming paradigm will be followed. The main division between paradigms are whether they are imperative or declarative. Imperative paradigms are defined by the explicit description of the series of actions to be performed by the program. Declarative paradigms are defined by the declaration of the overall intent of the program. Since we need to explicitly describe the method that FEARCF uses to calculate the free energy, an imperative paradigm would be preferable to a declarative one.

1.4.1 Object Orientated Programming

The previous implementation of FEARCF in CHARMM was written using a procedural paradigm, which does have its advantages in terms of debugging and modularity. It began to show its limitations, however, when carbohydrate ring puckering was implemented by Chris Barnett and procedures needed to be written for each dimensional case from one to seven dimensions. To avoid this unnecessary repetition in the code and to allow for the easy implementation of new reaction coordinate types, the design of the FEARCF library instead follows an object orientated programming (OOP) paradigm. The exact rational for this decision will be explored in Chapter 5.

OOP is a paradigm based upon the concepts of classes and objects. Classes define the general structure of objects, as well as the methods in which objects of the same class might interact. Objects are specific instances of a class, containing both data and methods for interacting with this data such as accessing, writing or manipulating. Figure 1.4 illustrates this relationship with a class and two instances of objects A and B, and how the class may

contain methods and procedures for object interaction.

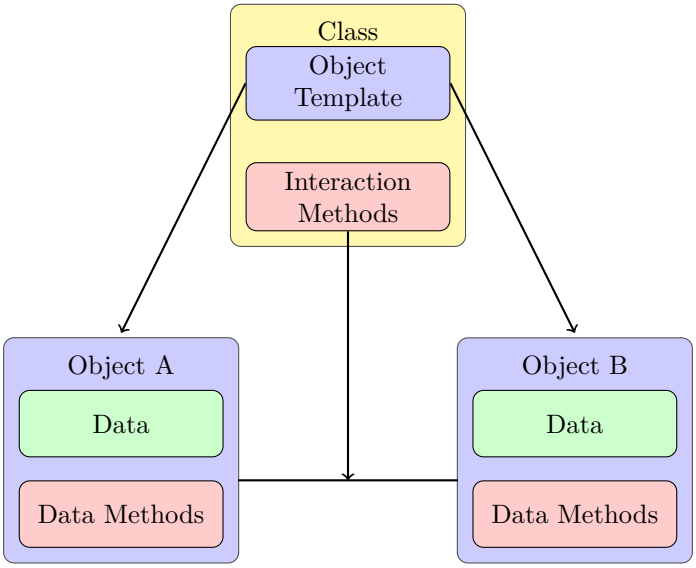


Figure 1.4: Illustration of an example OOP structure with two created objects, labelled A and B, from a single class definition which includes the methods for the objects' interaction.

There are several other key concepts in OOP such as encapsulation, inheritance and polymorphism [19]. These concepts, and why there are useful for a software library, will be explored further in Chapter 4.

The justification for FEARCF to be placed into a software library has now been established, and the nature of a OOP based library has been stated. What follows will be an explanation of how the OOP approach is also a natural fit for the FEARCF method itself.

1.5 Graph Theory

The justification for the OOP design on the FEARCF library will also depend upon the representation and definition of reaction coordinates in the form of graphs from the field of graph theory. This is why graphs are detailed below.

Graph theory is a branch of mathematics that studies the properties of abstract structures known as graphs, which are made up of nodes and the connection between nodes, referred to as edges. The usefulness of graph representation has been demonstrated by the success achieved with using these abstract graphs to model various real world systems. In chemistry, graph theory has been used in many approaches such as predicting structures of complex hydrocarbons, identifying and assigning NMR spectral peaks to molecular structures, and discovering of alternate reaction pathways within

already known reactions [20]. A set of simple graph types and definitions are given in Figure 1.5, showing how simple graphs can also be as directed, coloured or even further categorised as trees depending on the system that is being modelled.

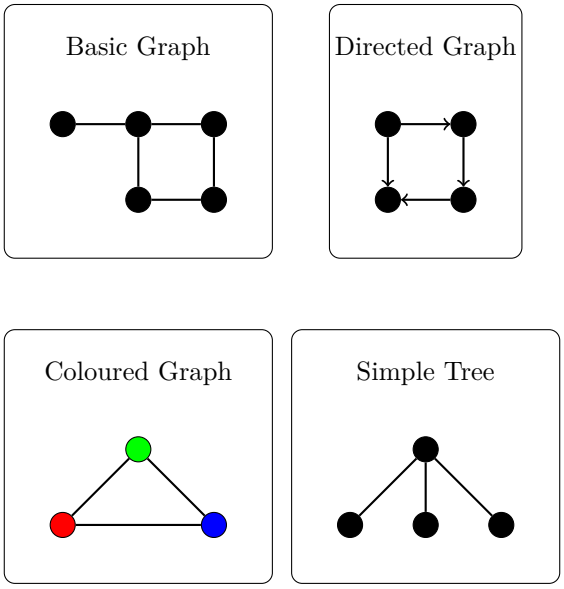


Figure 1.5: Examples of different types of graphs including directed, coloured and tree graphs.

The FEARCF library models reaction coordinates as coloured, directed graphs and the purpose for these graphs will be both for visualisation as well as confirmation that the reaction coordinate can be formed within the OOP design of the FEARCF library. These graphs are then used by FEARCF to calculate a multidimensional free energy surface with minimal effort. The symmetric nature of the graph node/edge structure and the OOP object/class implementation is key to understanding the FEARCF library's ability to easily perform multidimensional free energy simulations for a range of possible system/reaction coordinate combinations.

It has now been established what the central problem is, specifically, the need to implement FEARCF in a form in which it can be accessed by any MD software package. We have also established the plan is to solve this problem by placing FEARCF within a software library, with the library designed around graph theory and OOP. Below is given the structure of this thesis as it presents the work done toward the successful implementation of this plan.

1.6 Summary of Thesis Chapters

The objectives of this thesis is to demonstrate how the graph theory inspired OOP design of the FEARCF software library is capable of performing multi-dimensional free energy simulations for various molecular systems, at several levels of accuracy. To do this, sufficient background will be given on the theory underpinning FEARCF, graph theory and OOP, then the design of the library will be explored in detail, and finally results from various free energy simulations will be presented. These results will demonstrate the range of theories that can now be used in FEARCF, the different types of reaction coordinates that may be defined, the advantages of the graph model within the OOP framework of the library, and the increased clarity when studying molecular systems with multidimensional free energy. The chapters of this thesis will proceed as follows:

Chapter 2

Explanation of why biochemistry is key in our study of the natural world at a micro-scale. A further explanation of enzyme mediated reactions and their importance within biological systems, and how experimental methods have had limited success in probing their underlying mechanisms will be explored. Background of molecular dynamic simulations and the various levels of theory they may be conducted at will be given, with emphasis on how some these approaches give the desired insight for systems such as enzymatic reactions.

Chapter 3

Some background information and definitions in statistical mechanics as well as thermodynamics, and how the Gibbs approach to ensembles enables the extraction of meaningful quantities from sets of molecular dynamic simulations. Additional methods and quantities are also detailed such as Hamiltonian mechanics and time correlation functions. Lastly, several free energy methods and the theory underpinning each of them are given.

Chapter 4

Software libraries are explained and the various ways in which they may be linked to other code packages are explored. OOP is detailed including the key characteristics of the paradigm and how OOP is implemented in the Fortran language. Graph theory is also presented, along with several previous examples of graph theory within the field of chemistry, and how it is used in FEARCF for reaction coordinates.

Chapter 5

The FEARCF method is explored in detail and the partial derivatives for various reaction coordinates types are given. WHAM is also detailed in brief. The design of the FEARCF library is then given with emphasis on the graphing definition of the reaction coordinates and the OOP

implementation. The manner in which the library was compiled with both the CHARMM and NWCHEM packages is also provided.

Chapter 6

Free energy results, as well as time correlations, are given for the water dimer system that was modelled both classically and quantum mechanically. This is to present the advantages to a multidimensional approach to calculating free energy differences, as well as the increased molecular detail provided by *ab initio* methods.

Chapter 7

Free energy results are given for the glucose ring puckering system, as an illustration of the coding simplicity provided by the graph and OOP design of the FEARCF library.

Chapter 8

Free energy and minimum energy path results for the OGT-GlcNAc enzyme complex modelled with a QM/MM approach are presented, as a further example of FEARCF library's modular nature allowing for its utilisation by various MD software packages.

Chapter 2

Biomolecule Simulations

In this chapter the importance of biochemistry, as it relates to the study of biomolecules, will explored by presenting several example molecules and discussing how biochemistry is used to understand their various roles and functions in livings cells. Then the common methods for studying molecular systems using computers are given in anticipation of their use in computing free energy.

2.1 Biochemistry and its importance

Chemistry is defined as the scientific study of matter. Matter is composed of elemental atoms (and their subatomic components) as well as molecules, which are arrangements of atoms. Molecules can range from simple to endlessly complex. Biology is defined as the scientific study of organic life, ranging from viruses and micro-organisms, to complex multi-cellular creatures. The intersection of these two fields is referred to as biochemistry and involves the study of the chemical processes that take place either inside or outside of living organisms. These processes are essential to the organism's structure, continued function, and ultimate generation of progeny.

To illustrate how biochemistry is able to provide insight into biological processes, several molecules and molecular families will be presented along with their roles in biological systems and how biochemistry provides the physical basis for these roles.

2.1.1 Water

One of the key molecules to life on earth is water. Although not technically a biomolecule, water plays an essential role in many biomolecular processes. The most important role it plays is that of a 'universal solvent' able to dissolve a large number of different species of molecules [21] [22]. This is because of water's innate polarity. This polarity is due to the high electronegativity of

oxygen which causes the covalently shared electrons of the bonded hydrogens to be more attracted to the oxygen nucleus, leading to an unequal distribution of charge and the formation of a dipole moment [23]. This allows water to dissolve other polar molecules including charged ions. In living organisms water is therefore responsible for the transport of oxygen used for respiration, nutrients for energy, and the several types of charged ions used in the maintenance of cellular pH [24]. Water's polarity is also what allows for it to form hydrogen bonds with itself, with each water molecule able to form up to four hydrogen bonds with other water molecules. This array of hydrogen bonds is what gives bulk water its structure, as well as many of water's physical properties including its high boiling point and greater density in the liquid state, rather than solid state, which is unlike most other molecules [25].

Another important role water plays is in cellular structure. The presence of water inside cells is what provides cells the hydrostatic pressure to maintain their shape [26]. Without water, cells lose their shape, which in many cases, impedes their function. In addition, water is responsible for the maintenance of the intercellular membrane. All cellular membranes are primarily composed of phospholipids [27]. Phospholipids are a type of molecule whose structure can be separated into two regions: a polar 'head' and a non-polar 'tail'. In the presence of a polar solvent, like water, phospholipids will spontaneous form a lipid bilayer with the non-polar tails facing inwards towards each other and the polar heads facing outward toward the solvent [28]. This membrane allows for the cell to keep its essential structures as well as key molecules within itself.

Water not only affects the shape and structure of the cell, but many molecules within the cell such as DNA and proteins [29]. Proteins are molecules composed of a number of amino acids bonded in a series and they play a large variety of roles within the cell including: the make up of certain structures, cell signalling, transport, DNA replication and reaction catalysts. Although created as a linear molecule, proteins form complex three dimensional structures because of the interactions of its amino acid side chains with each other and with the cellular solvent [30]. Amino acids with non-polar side chains will orientate themselves to reduce their interaction with the polar molecules of water, while the opposite is true for amino acids with polar side chains. Since the structure of proteins is very often a key component in their functions, water is therefore essential for the proper function of these molecules [31].

Many biological reactions also require water as an easily accessible source of oxygen and hydrogen, or as a means to remove oxygens and hydrogens from a molecule in a reaction [32] [33]. These are the functions of acids and bases. Water is able to perform as either an acid or a base due to its neutral pH, and its ability to self-ionise into hydronium and hydroxide ions. As a neutral solvent, water is also able to protect the cell from the effects of other harmful acids and bases [34].

2.1.2 Carbohydrates and simple sugars

Carbohydrates are a class of biological molecule that are defined as being composed primarily of carbon, oxygen and hydrogen atoms. The simplest form of carbohydrates are known as monosaccharides, commonly referred to as sugars. The majority of all other carbohydrates are composed of several units of these monosaccharides. Carbohydrates composed of many units of monosaccharides (called olligo- or polysaccharides) are used as a form of energy storage for many organisms. They are also used as the base unit of large structures such as cellulose in plants [35]. Monosaccharides are a primary fuel source for cellular metabolism as well as an important component in the synthesis of many biomolecules including DNA [36].

Sugars (and their derivatives) have come under increasing study as their roles in living cells are further understood. The collection of all sugars and sugar-derived molecules in a living organism are referred to as the glycome. It is now widely accepted that the glycome exceeds the proteome (the collection of all proteins) and the genome (the collection of all genes) in terms of size and complexity [37] [38]. This is because of the various forms that sugars can take because almost all carbons in a sugar represent chiral centres [39]. Sugars also exist in either linear or cyclic forms, with the cyclic ring being either a 5 membered (furanose) or 6 membered (pyranose) ring. These ring structures are not static, with furanose rings able to 'pucker' into 20 distinct conformations while pyranose rings are able to 'pucker' into 38 distinct conformations [40] [41]. Certain pucker conformations have lower energies than the rest, however, a sugar may have to adopt a high energy conformation in order for a reaction to proceed. Low energy conformations may also be separated by higher energy ones, requiring the sugar to adopt these conformations before proceeding to another lower energy conformation [42]. This is why studying the relative free energies of sugar pucker conformations is important to understanding preferred sugar structures as well carbohydrates that play a role in chemical reactions.

2.1.3 Enzyme mediated reactions

In chemistry, a key component when studying chemical reactions is understanding whether they are spontaneous or not. By spontaneous it is meant that with all the reactants in the presence of each other, the reaction will proceed without further outside intervention. Non-spontaneous reactions, however, will not proceed without outside intervention of some kind. A common way of predicting whether a reaction is spontaneous or not, is comparing the free energies of the reactants vs the products [43]. If the products have a higher free energy than the reactants, energy will most likely have to be added to the system (commonly through heat e.g. a bunsen burner) in order for the reaction to occur. However, even if the products have a lower free energy than the reactants, the reaction may not be spontaneous. This is because there is

often an energy barrier (or activation energy) that the reaction is required to overcome in order for the reaction to proceed. This reaction barrier could be the result of a high energy transition state during the reaction, or simply the electrostatic repulsive forces needing to be overcome [44].

However, adding energy in the form of heat may not always be the most desirable or efficient means to enable a reaction. Biological systems are often highly sensitive to temperature changes that adding heat would result in. Therefore another means of enabling reactions is the use of catalysts. Catalysts are substances that enable a reaction to proceed without actually being consumed in the reaction itself [45]. This is commonly done by lowering the activation energy. An example is the Haber process [46] where hydrogen and nitrogen are transformed into ammonia, a common fertiliser. Normally this reaction is not spontaneous because nitrogen gas is inert, but with the use of iron based catalysts this reaction has become the main way of industrially producing ammonia [47].

In biological systems, a common set of catalysts are a class of proteins called enzymes. Enzymes enable non-spontaneous reactions to occur and also speed up spontaneous ones that proceed too slowly for whatever process they are needed [48]. Almost all biochemical processes are mediated by enzymes, not necessarily because all need to be made spontaneous (although most do) but rather because enzymes also allow for the control of the rate of the reaction. Precise control of reactions is possible because enzymes are often highly specific to their associated reaction(s) [49].

Enzymes are required to bind to reactants (substrates) of their specific reaction, in a part of the enzyme called the active site. It is in the active site that the enzyme performs its catalytic function. Enzymes have various mechanisms of how they catalyse reactions. These include: providing a more favourable chemical environment for the substrates, ionising or bonding with the substrates to provide an alternative reaction pathway, stabilising possible transition states or even destabilising the substrates themselves [50]. Enzymes are not limited to a single mechanisms and may use any number of them in combination to perform their function.

The activity of reactions can in turn be controlled by modifying the enzyme that controls the reaction [51]. This can be done by either down regulating or up regulating expression of the enzyme itself. Enzymes may also have associated promotors or inhibitors which affect their activity. These co-factors may bind to the enzyme and affect its conformational shape, in turn affecting the catalysing mechanisms [52].

Now that we have demonstrated that biochemistry plays a key role in understanding how many molecules perform their functions in biological systems, we will present computational methods that are used to probe these biochemical processes at a molecular level.

2.2 Molecular Dynamics

As discussed briefly in the introduction, MD provide a means to study details in molecular systems that cannot be measured experimentally. In this section three primary approaches to MD are presented, each based upon a different underlying theory.

2.2.1 Classical Dynamics

An important concept to understand about molecular interactions is that they are not static. Almost all biochemical processes occur within solution and therefore all molecules involved are interacting with the solution molecules. These interactions are constant and dynamic in nature. But the interactions between biomolecules themselves are not static either. Biomolecules are constantly changing shape, adjusting orientation, and interacting with new molecules. These changes all occur on the scale of femtoseconds. Therefore, the nature of molecular interactions present an intractable problem when attempting to find any kind of general solution using physics-based analytical models, outside of the most basic and simplified systems. The only solution is to take a numerical approach. The methods that use this approach are referred to as MD methods. This can be done by simply iteratively updating the atomic positions of every atom in a molecular system, by calculating the change in velocity caused by the net forces on the atoms. This method is referred to as the Euler method [53] as is given in Equation (2.1).

$$\begin{aligned} \vec{r}(t+\Delta t) &= \vec{r}(t) + \vec{v}(t)\Delta t \\ \vec{v}(t+\Delta t) &= \vec{v}(t) + \vec{a}(t)\Delta t \\ \vec{a}(t+\Delta t) &= \vec{a}(t) + \vec{F}(t)\Delta t \end{aligned} \tag{2.1}$$

In Equation (2.1), r is the atomic positions, v is the atomic velocities and a is the atomic accelerations. The atomic forces F are derived from some chosen physics model. For example in Equation (2.2) is given the force field used by the classical physics-based CHARMM MD software [54].

$$\begin{aligned} F_{CHARMM} = F_{bonds} + F_{angles} + F_{dihedrals} + F_{impropers} + F_{Urey\_Bradley} \\ + F_{electrostatic} + F_{Lennard-Jones} \end{aligned} \tag{2.2}$$

The first 5 terms on the right hand side of Equation (2.2) have to do with intra-molecular interactions i.e. the interactions between atoms of the same molecule. The last 2 terms have to do with inter-molecular interactions i.e. the interactions between atoms of different molecules. While the above is

referred to as a classical force field, terms such as $F_{Lennard-Jones}$ are actually approximations of primarily quantum phenomena e.g. Pauli repulsion and electron dispersion.

The update of atomic coordinates may proceed via a simple Euler method as given in Equation (2.1), however, most modern MD software packages use alternative, so called 'integrators', such as: the leapfrog integrator, Velocity-Verlet, symplectic integrator, or Beeman's Algorithm [55]. There are several advantages to using these methods instead of Euler's method, such as the fact that many are 2nd order as opposed to Euler which is 1st order. This means they provide a greater level of accuracy. Other advantages include: timer reversibility, allowing for constant temperature and pressure simulations, greater parallelisation, and the use of multiple time step lengths.

The original Verlet algorithm [56] does not in fact include updating the velocity at all since they are technically not needed once the acceleration is known. The update equation for the atomic positions are given in Equation (2.3). It is found by Taylor expansion about $\vec{t}$.

$$r(r + \Delta t) = 2\vec{r}(t) - \vec{r}(t - \Delta t) + \Delta t^2 \vec{a} \tag{2.3}$$

While only having to solve a single equation is convenient, knowing the velocities are desirable since they allow us to calculate the kinetic energy as well as determine (and later control) the temperature of the system. An alternate form of the Verlet algorithm therefore exists called the Leapfrog method [57] which, rather than calculating the positions and velocities at the same time, calculates the velocities at every half Δt step. Then, the algorithm 'leapfrogs' to the next position calculation. This is shown in Equation (2.4)

$$\begin{aligned} \vec{r}(t + \Delta t) &= \vec{r}(t) + \vec{v}(t + \frac{1}{2}\Delta t)\Delta t \\ \vec{v}(t + \frac{1}{2}\Delta t) &= \vec{v}(t - \frac{1}{2}\Delta t) + \vec{a}(t)\Delta t \end{aligned} \tag{2.4}$$

In order to calculate $\vec{v}$ at time t, we will need to do another calculation $\vec{v}(t) = \frac{\vec{v}(t+\frac{1}{2}\Delta t)+\vec{v}(t-\frac{1}{2}\Delta t)}{2}$.

This is still inconvenient therefore another method was developed which calculates both $\vec{r}$ and $\vec{v}$ at the same time, and this is called the Velocity-Verlet algorithm [7]. This algorithm is shown in Equation (2.5).

$$\begin{aligned} \vec{r}(t + \Delta t) &= \vec{r}(t) + \vec{v}(t)\Delta t + \frac{1}{2}\vec{a}(t)\Delta t \\ \vec{v}(t + \Delta t) &= \vec{v}(t) + \frac{1}{2}(\vec{a}(t) + \vec{a}(t + \Delta t))\Delta t \end{aligned} \tag{2.5}$$

Equation (2.5) now gives us position and velocity at the same times which is particularly useful. While Verlet and the subsequent algorithms are more accurate ($\mathcal{O}(\Delta t^2)$) than the simple Euler method ($\mathcal{O}(\Delta t)$), they are still approximations and have some expected error. A more accurate method is one known as Beeman's Algorithm [58] which has accuracy $\mathcal{O}(\Delta t^4)$ with regards to the atomic positions. It is given in Equation (2.6).

$$\begin{aligned}\vec{r}(t+\Delta t) &= \vec{r}(t) + \vec{v}(t)\Delta t + \frac{1}{6}\left(4\vec{a}(t) - \vec{a}(t-\Delta t)\right)\Delta t^2 \\ \vec{v}(t+\Delta t) &= \vec{v}(t) + \frac{1}{6}\left(2\vec{a}(t+\Delta t) + 5\vec{a}(t) - \vec{a}(t-\Delta t)\right)\Delta t\end{aligned} \tag{2.6}$$

This increased accuracy comes at a cost, being more computationally expensive.

Another consideration required when setting up a MD simulation is what the initial velocities should be. Ordinarily the starting atomic coordinates are imported or easily constructed manually, but the same is not true for the starting atomic velocities. What is normally done is that scalar values of the velocities are drawn from a normal distribution focused around the average expected velocity. This average velocity is calculated from the temperate T of the system which is chosen before the simulation is begun. Equation (2.7) shows how the temperature is related to the average kinetic energy.

$$\langle E_{kin} \rangle = \frac{3}{2} N k_B T \tag{2.7}$$

In Equation (2.7), N is the number of atoms in the system and k_B is the Boltzmann constant. The way in which the average velocity is related to the average kinetic energy $\langle v_i \rangle$ is given in Equation (2.8).

$$\langle E_{kin} \rangle = \sum \frac{1}{2} m_i \langle v_i \rangle \tag{2.8}$$

In Equation (2.8), m_i is the mass for each atom i. The system is then typically run for a short period in order to allow for the system to be 'equilibrated'. This is to prevent the system from being biased by the randomly assigned initial velocities.

Methods known as thermostats are then used to keep the temperature of the system the same by periodically updating or correcting the atomic velocities. Even with conservative forces, changes in the average velocity is expected to occur due to numerical error. Methods that allow this control of temperature include: the Nose-Hoover thermostat, velocity rescaling, Gaussian, Langevin and Anderson methods [59].

Velocity rescaling [60] simply scales all the velocities by a factor λ, which is given by $\lambda = (\frac{T_{target}}{T})^{\frac{1}{2}}$ where T_{target} is the desired system temperature while T is the current system temperature. The problem with this method is that it leads to discontinuities in the momentum.

The Gaussian thermostat [61] adds an additional term to Newton's second law in order to keep the temperature constant. This is shown in Equation (2.9) where N is the number of particles and m_i are their associated masses.

$$\begin{aligned} \dot{\vec{p}}_i &= -\frac{\partial U}{\partial \vec{q}_i} - \alpha \vec{p}_i \\ \alpha &= \frac{\sum_i^N \frac{\partial U}{\partial \vec{q}_i} \cdot \frac{\vec{p}_i}{m_i}}{\sum_i^N \frac{\vec{p}_i}{m_i}} \end{aligned} \tag{2.9}$$

The Berendsen thermostat [62] is related to the velocity scaling method but is designed to create an exponential decay toward the target temperature with the scaling factor λ now given by $\lambda = \left(1 + \frac{\Delta t}{\tau}\left(\frac{K_{target}}{K}\right)\right)$ where τ is the time constant which controls the rate of decay.

In the Nosé-Hoover thermostat [63] the system is connected to a heat bath, which is turns changes the Hamiltonian to the form given in Equation (2.10).

$$H = \sum_i^N \frac{\vec{p}_i}{2m_i s^2} + U + \frac{p_s^2}{2M} + g k_B T \ln s \tag{2.10}$$

In Equation (2.10) M is the effective mass of the heat bath, s is the degree of freedom introduced by the heat bath and g is the number of degrees of freedom of the system itself.

The Langevin thermostat [64] is also alters Newton's second law by adding two additional terms, the first representing a drag force and the second being a random force meant to present random kicks given by other particles. This modified version is given in Equation (2.11) where γ_i is a drag coefficient given by $\gamma_i = 6\pi\eta r_i/m_i$ with r_i being the particle radius, m_i the particle mass, and η is the viscosity constant of the system. Also f_i is a random number drawn from a Gaussian distribution with variance $\sigma_i^2 = 2m_i\gamma_i k_B T/\Delta t$ where Δt is the step size.

$$\dot{\vec{p}}_i = -\frac{\partial U}{\partial \vec{q}_i} - \gamma_i \vec{p}_i + f_i \tag{2.11}$$

There are several other considerations when setting up MD simulations or trying to optimise them. These include: short and long term interaction cutoffs

and approximations, periodic boundary conditions or other constraint conditions, parallelisation strategies such as domain distribution, and ensemble selection and enforcement [7].

While the above methods prove themselves to be incredibly useful for almost all MD simulations, the underlying classical theory quickly proves itself to be quite limited in what molecular interactions it can model.

2.2.2 Quantum MD

Atoms (and therefore molecules) are inherently quantum in nature. Classical models may approximate molecules satisfactorily for some cases. In many other cases, however, they particularly fall short when attempting to model the intra-molecular bond formation and breaking that occurs during chemical reactions. Modelling molecules using what is referred to as *ab initio* methods have therefore been done since the late 1920s [65]. Importantly, not the entire molecule needs to be treated quantum mechanically. In chemical reactions only the electrons are involved, therefore, modelling the nuclei of atoms quantum mechanically is unnecessary. The treatment of nuclei as classical point charges and the electrons quantum mechanically is known as the Born-Oppenheimer approximation [66]. To calculate the evolution of the electrons over time, the Schrödinger equation, given in Equation (2.12), needs to be solved.

$$i\hbar\frac{d}{dt}|\Psi(t)\rangle = \hat{H}|\Psi(t)\rangle \tag{2.12}$$

In Equation (2.12), $\hat{H}$ is the Hamiltonian of the system and $\Psi(t)$ is the wave-function. In *ab initio* methods this wave-function represent the electrons.

Solving the Schrödinger equation analytically for any system other than the simple hydrogen atom is not possible, therefore numerical methods need to be used to solve more complex multi-atom systems. The first developed approach is known as the Hartree-Fock [67] [68] method, which uses the approximation of treating each electron as a single particle wave-function. It is then assumed the product of these independent wave-functions is a good approximation to the total electronic wave-function of the molecular system.

This approach is based upon assuming that the system Hamiltonian is 'separable' i.e. can be written as the sum of a hamiltonian term for each electron. The Hamiltonian takes the form $H = \sum_{i=1}^{N} h_i$ where N is the total number of electrons and h_i is given in Equation (2.13). This h_i has a kinetic energy term and a nuclear attraction term for each nuclei, with Z_j being the nuclear charge of the jth atom with a total of M atoms and distance between the electron i and this nucleus being given by r_{ij}.

$$h_i = -\frac{1}{2}\nabla_i^2 - \sum_{j=1}^{M}\frac{Z_j}{r_{ij}} \tag{2.13}$$

As one electron hamiltonians, h_i must satisfy the Schrödinger equation with the appropriate eigenfunctions i.e. $h_i\psi_i = \epsilon_i\psi_i$. Since the system Hamiltonian is the sum of these one electron hamiltonians, the eigenfunction of H can be written as the product of these one electron eigenfunctions i.e. $\Psi_{HP} = \psi_1\psi_2...\psi_N$. This is referred to as the Hartree-product wave function.

The problem with Equation (2.13) is that it does not include electron-electron repulsion. Instead, we can continue with Ψ_{HP} but provide an alternate one electron hamiltonian, which is given in Equation (2.14), where ρ_j is the charge or probability density of the jth electron.

$$h_i = -\frac{1}{2}\nabla_i^2 - \sum_{j=1}^{M}\frac{Z_j}{r_{ij}} + \sum_{j\neq i}^{N}\frac{\rho_j}{r_{ij}}d\vec{r} \tag{2.14}$$

The problem we now face is that ρ_j is given by the one electron wave function i.e. $\rho_j = |\psi_j|^2$. But we need ρ_j in order to use h_i to solve $h_i\psi_i = \epsilon_i\psi_i$ and find ψ_i. We are therefore left with two equations that cannot be solved because they depend on the result of the other. In order to overcome this, Hartree proposed the 'self-consistent field' (SCF) method in which an initial guess of the one electron wave-functions is used to calculate an iteratively better estimate of the wave-functions. When the difference between the previous estimate and the new estimate for the wave function becomes negligibly small (the exact criteria is arbitrary) then the wave functions are said to be converged.

This method was further expanded with the use of Slater determinants [69] [70] to incorporate electron spin, the subsequent reformulation of the Hamiltonian into what is called the Fock Matrix, and then the use of basis set representation to construct expressions for the orbital wave functions. The original basis sets described were of the Slater-Type orbital (STO) [71], however, the far more common type of orbital used in computational chemistry are those of the Gaussian-type orbital (GTO) [72]. The general form of GTOs is given in Equation (2.15) where α is a factor controlling the width of the basis, i, j, k are quantum indices and x, y, z are cartesian coordinates. The reason for the aversion for STOs are that they do not in general have analytic solutions when used with the Fock operator, therefore requiring the use of numerical methods when using STOs. In contrast, GTOs do provide analytical solutions [73].

$$\phi(x, y, z; i, j, k)a = \left(\frac{2\alpha}{\pi}\right)^{3/4}\left[\frac{(8\alpha)^{i+j+k}i!j!k!}{(2i)!(2j)!(2k)!}\right]^{1/2} x^i y^j z^k e^{-\alpha(x^2+y^2+z^2)} \tag{2.15}$$

This is a surprisingly successful approximation but does not explicitly model electron-electron correlation. A set of methods, referred to as post-Hartree-Fock methods, have therefore been developed including: coupled cluster [74], Møller-Plesset (labelled MPN where N is the number of correction terms) which uses perturbation theory to provide corrections to the HF energy [75], and density functional theory (DFT) [76].

The principle behind DFT [77] is the desire to express the components of quantum mechanics, namely the wave function, the Hamiltonian, and the energy, as functions of the electron density ρ. The electron density is itself a function of cartesian coordinates. A function whose argument is in turn a function of another argument is referred to as a 'functional' hence the name of the theory as 'density functional'. The most basic of density functional forms are the classical Coulomb attraction potential of nuclei Z_i and electron density of a molecule given in Equation (2.16), and the self-repulsion potential of the electron density with itself given in Equation (2.17) where both equations are integrated over all space.

$$V_{attraction}(\rho) = \sum_i \int \frac{Z_i}{|\vec{r} - \vec{r}_i|}\rho(\vec{r})d\vec{r} \tag{2.16}$$

$$V_{repulsion}(\rho) = \frac{1}{2}\int\int \frac{\rho(\vec{r}_1)\rho(\vec{r}_2)}{|\vec{r}_1 - \vec{r}_2|}d\vec{r}_1 d\vec{r}_2 \tag{2.17}$$

The potential, however, is just one half of the Hamiltonian, with the kinetic energy K given by the expression in Equation (2.18). This expression is the kinetic energy for a uniform electron gas of constant density first described by Thomas and Fermi [78] [79].

$$K_{EG}(\rho) = \frac{3}{10}(3\pi^2)^{2/3}\int \rho^{5/3}(\vec{e})d\vec{r} \tag{2.18}$$

The above are not useful to us, however, since they rely upon classical approximations. Further developments by Hohenberg and Kohn [80] managed to show that the 'external potential' that electrons interact with in DFT e.g. Equation (2.16), is determined by the ground-state (i.e. lowest energy stationary state) electron density. They also proved that any chosen electron density must give energy greater than, or equal to, the true ground-state density. This means that densities that give lower energies are better approximations to the ground sate energy than densities that give higher energies. It was the work of Kohn and Sham [81], however, that demonstrated an actual approach to obtaining expressions for the energy, wave function, and Hamiltonian in terms of the electron density. In their approach, they began by consider a system of non-interacting electrons. The expression for the energy of this system, E_{NI}, is given by Equation (2.19) where the kinetic energy term K is just the sum of kinetic energy for each electron i.e. $K = \sum_i k_i$.

$$E_{NI}(\rho) = K(\rho) + V_{attraction}(\rho) + V_{repulsion}(\rho) \tag{2.19}$$

Equation (2.19) can then be used to construct a system with the same electron density but where the electrons do interact. The energy of this system is given in Equation (2.20) where the second term on the right hand side of the equation is referred to as the exchange-correlation energy.

$$E(\rho) = E_{NI}(\rho) + E_{XC}(\rho) \tag{2.20}$$

$E_{XC}(\rho)$ contains corrections for the classical self-interaction energy, the difference in kinetic energy between the non-interacting and interacting systems, as well as the electron exchange and correlation. A similar approach to HF is taken where the overall wave function Ψ_{KS} is constructed as a product of individual wave functions ψ_i for each electron. An expression for the individual hamiltonian h_i^{KS} for each ψ_i is given in Equation (2.21).

$$\begin{aligned} h_i^{KS} &= -\frac{1}{2}\nabla_i^2 - \sum_j \frac{Z_j}{|\vec{r}_i - \vec{r}_j|} + \int \frac{\rho(\vec{r'})}{|\vec{r}_i - \vec{r'}|} d\vec{r'} + V_{XC} \\ V_{XC} &= \frac{\partial E_{XC}}{\partial \rho} \end{aligned} \tag{2.21}$$

The challenge is to find an expression for the exchange-correlation energy term $E_{XC}(\rho)$. There are numerous approaches to generating this term including: Local Density Approximation (LDA) [81], Generalised Gradient Approximation (GGA) [82], and Adiabatic Connection Method (ACM) [83]. Some of the most successful XC terms are combinations of the given methods [76].

While the above approaches enable the study of molecule systems at a highly precise level of theory, they all come with a significant computational cost which limits their applications to larger molecular systems.

2.2.3 QM/MM

While *ab initio* methods have been very successful at modelling molecular systems quantum mechanically, they are still computationally expensive and time consuming, therefore, they have been restricted in their application to smaller systems. There are larger systems that could benefit from *ab initio* modelling, particularly in small regions of the system such as in the active site of a large enzyme where chemical reactions take place. With this need in mind, methods have been devised that aim to combine both classical and *ab initio* methods. These are referred to as QM/MM methods [84]. QM/MM methods divide the molecular systems into two regions: the quantum region

and the classical region. A representation of how a QM/MM molecular potential $U_{QM/MM}$ is constructed is given in Equation (2.22).

$$U_{QM/MM} = U_{MM} + U_{QM} + U_{QM/MM} \tag{2.22}$$

U_{MM} is the classical potential, U_{QM} is the *ab initio* potential, and most importantly, $U_{QM/MM}$ is an interaction term between the two regions. This interaction term takes into account the electrostatic interaction between the classical charges in the classical region and the electron density within the quantum region, as well as the dispersion interactions between the regions and the covalent bonds that may cross both regions. The definition of the different regions represent a considerable modelling challenge since atoms moving between regions will complicate the theory required and likely decrease the accuracy of the model. In addition to this, covalent bonds that cross the region boundary create complications if they are cleaved. This is solved by implementing methods such as link atoms, boundary atoms, or fixed orbital schemes [9].

The most simple QM/MM system is one in which the boundary between the QM and MM region does not cut through any bonds. A further simplifying assumption is that neither region experiences polarisation. The $U_{QM/MM}$ term is then composed simply of electrostatic and non-bonded interaction terms, which is given in Equation (2.23) where the i index sums over the MM nuclei and the j index sums over the QM nuclei.

$$U_{QM/MM} = \sum_i \left(\int \frac{Z_i \rho(\vec{r})}{|\vec{r}_i - \vec{r}|} d\vec{r} + \sum_j \frac{Z_i Z_j}{|\vec{r}_i - \vec{r}_j|} \right) + \sum_i \sum_j 4\epsilon_{ij} \left(\frac{\sigma_{ij}^{12}}{r_{ij}^{12}} - \frac{\sigma_{ij}^{6}}{r_{ij}^{6}} \right) \tag{2.23}$$

If bonds do cross the boundary however, they can be partly modelled in the MM force field with standard bond (harmonic potentials), angle, and torsion potentials between the QM nuclei that are involved with the boundary crossing bonds. A greater complication when a bond crosses the region boundary is that in the QM region, the molecular orbitals will be unbalanced due to the missing electrons being in the MM region. A way to address this problem is adding what is called a 'link atom' [85], most commonly a hydrogen atom, which balances the molecular orbital of the bond, but which is not included in the MM calculations. Another technique is to replace the boundary crossing bond with a double occupied orbital which is parameterised and then kept constant throughout the simulation [86]. MM atoms close to the QM region may also create problems and so it also possible to replace these nuclei point charges with a gaussian charge density centred on the MM atom in question [76].

The assumption that neither region polarises the other is not always a good assumption for some systems, therefore, the polarisation of the QM region can

be modelled by including an additional term in the one electron hamiltonian. This additional term is given in Equation (2.24) where i is the index for associated electron and j is an index for all MM atoms with partial charges [87].

$$h_i^{QM/MM} = h_i^{QM} - \sum_j \frac{e^2 Q_j}{4\pi\epsilon_0 |\vec{r}_i - \vec{r}_j|} \tag{2.24}$$

The polarisation of the MM region can also be modelled, but it requires significant changes to the force field and has been shown to have little effect on the overall accuracy of most QM/MM simulations [8].

In this chapter we have discussed biochemistry and its importance for understanding the role of many molecules within living organisms, as well as present various MD methods for modelling these molecules and their interactions at varying levels of theory. In the next chapter we will present the statistical and thermodynamic background for the important quantity that is free energy, and how the above mentioned MD methods can be modified in order to calculate this free energy.

Chapter 3

Statistics and Numerical Methods for MD Simulations

In this chapter the theoretical background for free energy is provided by exploring relevant concepts in thermodynamics and statistical mechanics. Then several numerical methods for calculating the free energy of molecular systems are described, along with a number of other numerical methods that will be referenced later.

3.1 Thermodynamics

The purpose of statistical mechanics is to describe physical systems, which in our particular case are chemical systems. So before we begin our discussion of statistics, we will define a set of quantities and relations that fall under the branch of physics known as thermodynamics, which deals with concepts of energy exchange, heat, and work.

Any system can be said to have some energy E. The first law of thermodynamics [88] tells us how the energy of our system will change as it goes from one state to another. Before we state the first law we need to firstly define some important quantities. The volume V is very simply the amount of physical space the system occupies. Pressure P is defined as the force exerted on an area. To do work W is defined as a force being exerted over a distance. Work turns out to have the same units as energy and therefore represents a form of energy. Work can we performed by a pressure being exerted on the surface of a volume, leading to a change in the volume itself. This can we written simply as $W = \int PdV$ where dV is an infinitesimally small portion of V, which is then integrated over the whole volume. This is just an example of work and there are many other ways for a system to perform work [89].

Heat Q is a slightly more difficult quantity to define since in general it doesn't have an explicit form, but is rather defined as any energy that is exchanged

between two systems that is neither thermodynamic work or the exchange of matter. It is therefore a process quantity rather than a system quantity [90]. With the concept of heat we are able to state the first law of thermodynamics in Equation (3.1). The first law simply states that the change in energy of a system which cannot exchange matter with its surroundings (i.e. all other systems outside itself that it can directly interact with) is the difference between the heat gained from its surroundings and the work done by the system on its surroundings.

$$\Delta E = Q - W \tag{3.1}$$

Equation (3.1) is actually just a statement on the law of conversation of energy i.e. energy gained by a system must be equal to the energy lost to its surroundings and vice versa.

To discuss the second law of thermodynamics [88], we need to introduce two more quantities. The first is the more familiar concept of temperature T. It is a common understanding that T is connected to the average kinetic energy $\tilde{E_k}$ of a system through the relation $\tilde{E_k} = \frac{3}{2}k_bT$, where k_b is a constant known as Boltzmann's constant [43]. This is a definition that actually only holds for relatively simple systems where there is only linear kinetic energy, and no rotational or vibrational kinetic energy.

To obtain a more concrete definition for T, we can jump ahead a little bit and reference some statistics. Consider a system with energy E and an associated number of microstates Ω i.e. the number of possible configurations of the system that all have the same energy E. Using the zeroth law of thermodynamics [88], which states that when two systems are both in thermal equilibrium (i.e. no exchange of heat) with a third system, the three systems can be said to be in thermal equilibrium with each other, the temperature can be defined in a more general way given by Equation (3.2).

$$\frac{1}{k_bT} = \frac{d\ln\Omega}{dE} \tag{3.2}$$

This definition in Equation (3.2) then allows us to introduce our final quantity called entropy S [91]. Rearranging Equation (3.2) we can get a quantity that we call the entropy of a system $S = k_b \ln\Omega$. S clearly scales with Ω and therefore, a system with E_1 that has more microstates than another system with E_2 is said to have greater entropy. Entropy is typically described as a measure of the 'disorder' of a system since greater entropy means more possible microstates that a system can move between without changing its energy. A system with a lower entropy is inversely said to be more 'ordered' because it only has a small number of microstates that the system can be found in, making the system more predictable in its behaviour.

Next, we introduce the concept of free energy. Free energy is the energy that is 'free' for the system to use to perform work i.e. the energy that can be transferred without fundamentally changing the nature of the system. The Helmholtz free energy A [92] is defined as the free energy within a system of constant temperature, maintained by its connection to a heat bath (a system of practically infinite energy.) This heat bath allows the system to keep its temperature constant. The equation for A is given in Equation (3.3).

$$A = U - ST \tag{3.3}$$

An alternative free energy is the Gibbs free energy G [93], which is defined as the free energy of a system at constant pressure and temperature. The equation is given in Equation (3.4).

$$G = U + pV - ST \tag{3.4}$$

The free energy that is useful when studying a system is therefore dependant upon the system in question. The free energy that chemists typically refer to is the Gibbs free energy [94]. In systems of constant temperature, A can actually be used as a thermodynamic potential which defines the thermodynamic state of a system at any given time. As a potential, A reaches a minimum at equilibrium. The differential form of equation (3.3) is given simply as $dA = TdS + Vdp$, which then in turn gives the relations $T = (\frac{\partial A}{\partial S})_p$ and $V = (\frac{\partial A}{\partial p})_T$.

We have now detailed some important thermodynamic variables so we will move onto discussing the statistics of multiple molecular systems.

3.2 Statistical Mechanics

Statistics is a field of work that revolves around the collection, study, and display of data, or 'populations' of quantities. While statistics is viewed by some as a branch or sub-section of mathematics, but it is mostly strongly defined by its applications to other research areas that have to deal with real world 'populations' of data. The research areas include: species group behaviour, voting and census data, and stock market variations. Statistical mechanics is defined as 'that branch of physics which studies macroscopic systems from a microscopic or molecular point of view' [10]. The term 'macro' is important because it refers to a collection of objects or quantities, and this is where the 'statistical' aspect of statistical mechanics comes from. In order to discuss deeper aspects of statistical mechanics, there are some key concepts that need to be explored first.

3.2.1 Probability Distributions

Say that we have a system with N possible unique states $i \in (1, N)$, each of which the system is equally likely to be found in at any given time. If we wish to know what the chance (or probability) of finding the system in any state i, this is simply given by Equation (3.5).

$$P(i) = \frac{1}{N} \tag{3.5}$$

Here $P(i)$ is known as the probability distribution i.e. a function that gives the probability of finding the system in state i. There are some important things to note about P namely: $P(i) \geq 0$ for all i, $\sum_i^N P(i) = 1$, and P can only have a finite number of discrete values, in this case $1/N$. These first two points are fundamental and true for all probability functions (they are part of what is known as the Kolmogorov axioms), while the last point is not fundamental and we will quickly encounter probability distributions that can have an infinite number of continuous values. This is one of the simplest examples of probability that we could give but it presents some key concepts that will become invaluable when studying more complex probability distributions.

An example of a continuous probability distribution, is the very well known Gaussian or Normal distribution. It was first described by several people including de Moivre, Leplace and Gauss [95]. It is well known because of its frequent occurrence in natural phenomena involving random variables, due in part to the central limit theorem [96]. Its most common form in modern times is given in Equation (3.6).

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{(x - \overline{x})^2}{2\sigma^2} \tag{3.6}$$

In Equation (3.6), $\overline{x}$ is the expected mean value of the continuous variable x, while σ is the standard deviation from the mean i.e. a measure of the spread of probability centred around the mean.

3.2.2 Ensembles

In our discussion about probability distributions, we were only talking about abstract systems, but let us now focus on more physical systems. Let us consider a system with N number of particles with occupy a volume V, and which have a total energy of E. If we wanted to study the dynamics of this system we could study this one system for a long time. This approach has its limitations, however, and we could in fact wait an eternity for the system to eventually occupy every possible state just once. Therefore, it is best that we try another approach and thankfully one is already presented

to us by the work of Gibbs [97]. Rather than studying one single system for some length of time, we rather study a collection of copies of this system at different states. These copies can be thought of as 'snapshots' of our single system, but at different times. It turns out that using this approach allows us to utilise powerful statistical tools to increase our understanding of the system [10]. This collection of copies of the system is called an *ensemble*. There are several different types of ensembles, each of which are defined by which quantities of the system are kept constant. In this case, we could choose to keep N, V and E constant for all our copies of the system. Doing this creates what is called a micro-canonical ensemble.

Let us now rather consider an ensemble in which we keep N, V and the temperature T constant. This is called the canonical ensemble [98]. We then have C number of copies of our system. Let us assume that our system has L number of possible states that it could be in. For each state l, it has an energy E_l, and there is c_l number of copies of our system in that state. We can therefore note some important relations in Equation (3.7).

$$\begin{aligned} \sum_{l}^{L} c_l &= C \\ \sum_{l}^{L} c_l E_l &= \epsilon \end{aligned} \tag{3.7}$$

In Equation (3.7), ϵ is the total energy for all the copies in our ensemble. We then consider the number of possible ways $W(\vec{c}) = W(c_1, c_2, c_3, ..., c_L)$ that we could arrange all our copies C, given some distribution of the number of copies in each state $\vec{c}$. We also assume that all our copies are distinguishable. This is given in Equation (3.8).

$$W(\vec{c}) = \frac{C!}{c_1! c_2! c_3! ...} = \frac{C!}{\prod_l c_l!} \tag{3.8}$$

Now we consider the probability that a copy is in state l. We find this by having the average number of copies in state l divided the total number of copies. This expression is given in Equation (3.9), where the sums are in terms of all the possible ways to construct $\vec{c}$.

$$P_l = \frac{\overline{c}_l}{C} = \frac{1}{C} \frac{\sum_{\vec{c}_i} W(\vec{c}_i) c_l(\vec{c}_i)}{\sum_{\vec{c}_i} W(\vec{c}_i)} \tag{3.9}$$

In Equation (3.9) the function $c_l(\vec{c}_i)$ donates the number of copies in the lth state, in distribution $\vec{c}_i$.

We now have the probability to find any given system in state l. We then write a simple expression for the average value of any mechanical property of the system, given that we know that value of M_l for any state l. This is given in Equation (3.10).

$$\overline{M} = \sum_l M_l P_l \tag{3.10}$$

Let us consider the distribution $\vec{c}^*$ that maximises our function $W(\vec{c})$ i.e. the most probable distribution, since there are so many ways to create this distribution. If we simply increase the number of copies $C \to \infty$, the value of $W(\vec{c}^*)$ will continue getting larger and dominate the other terms. We are therefore left with the approximation in Equation (3.11).

$$P_l = \frac{1}{C}\frac{\sum_{\vec{c}_i} W(\vec{c}_i)c_l(\vec{c}_i)}{\sum_{\vec{c}_i} W(\vec{c}_i)} \approx \frac{1}{C}\frac{W(\vec{c}^*)c_l^*}{W(\vec{c}^*)} = \frac{c_l^*}{C} \tag{3.11}$$

We now notice a similarity between Equation (3.11) to Equation (3.9). We therefore have a way to find the mean value of $\overline{c}_l$ by finding $\vec{c}^*$ that maximises $W(\vec{c})$. This can be done by using the method of Lagrange multipliers [99] on Equation (3.8) with the constraints given in Equation (3.7). By doing this we get the following expression in Equation (3.12).

$$\frac{\partial}{\partial c_l}\left(\ln W(\vec{c}) - \alpha\sum_i c_i - \beta\sum_i c_i E_i\right) = 0 \tag{3.12}$$

In Equation (3.12) we have a set of equations for $l = 1, 2, ..., L$ and two undetermined multipliers α and β. By looking at Equation (3.8), we realise that we can use Stirlings approximation [100] for $W(\vec{c})$ if we let $c_l \to \infty$. This result is given in Equation (3.13).

$$-\ln c_l^* - \alpha - 1 - \beta E_l = 0 \tag{3.13}$$

Rearranging Equation (3.13), we get the form of Equation (3.14).

$$c_l^* = e^{-\alpha'}e^{-\beta E_l} \tag{3.14}$$

Here $\alpha' = \alpha + 1$. To get an expression in terms of α, we can simply sum Equation (3.14) for all l and then use the first part of Equation (3.7) to get Equation (3.15).

$$e^{-\alpha'} = \frac{C}{\sum_l e^{-\beta E_l}} \tag{3.15}$$

Then we just substitute Equation (3.15) back into Equation (3.11) to get Equation (3.16).

$$P_l = \frac{c_l^*}{C} = \frac{e^{-\beta E_l}}{\sum_l e^{-\beta E_l}} \tag{3.16}$$

The term in the denominator of the right hand side of Equation (3.16) is known as the canonical partition function Q [10]. In conjunction with Equation (3.10), it can be used to calculate the average value of any mechanical property. More importantly, Q appears in lots of other thermodynamic expressions. It is typically given in the form of Equation (3.17).

$$Q(N, V, \beta) = \sum_l e^{-\beta E_l(N,V)} \tag{3.17}$$

The only question that remains is what is the true value of the variable β? To answer this question, we can use the partition function for its intended purpose i.e. calculating averages. First we can start off with energy E and calculating its average, which is given in Equation (3.18).

$$\overline{E} = \sum_i E_i P_i = \frac{\sum_i E_i e^{-\beta E_i}}{Q} \tag{3.18}$$

The average energy $\overline{E}$ for our ensemble of systems is actually equivalent to the average energy of a single system over time. This is the fundamental motivation for the Gibbs method of using ensembles. Therefore if we have the energy for a system, we can use it to find the pressure p. We can relate energy and pressure by considering the relation given in Equation (3.19), which shows the work down on the system when changing the volume of dV, assuming the number of particles is kept constant.

$$dE_i = -p_i dV \tag{3.19}$$

To calculate the average pressure $\overline{p}$ in Equation (3.20), we can again use the canonical partition function and Equation (3.10).

$$\overline{p} = \frac{\sum_i p_i e^{-\beta E_i}}{Q} = -\frac{\sum_i (\frac{\partial E_i}{\partial V}) e^{-\beta E_i}}{Q} \tag{3.20}$$

Next we differentiate Equation (3.18) with respect to V, remember that E_i is a function of V. The result of this is given in Equation (3.21).

$$\begin{aligned}\frac{\partial \overline{E}}{\partial V} &= \frac{\sum_i (\frac{\partial E_i}{\partial V}) e^{-\beta E_i}}{Q} - \beta \frac{\sum_i E_i (\frac{\partial E_i}{\partial V}) e^{-\beta E_i}}{Q} - \frac{\sum_i E_i e^{-\beta E_i}}{Q^2} \left(-\beta \left(\frac{\partial E_i}{\partial V} \right) e^{-\beta E_i} \right) \\ &= -\overline{p} + \beta \overline{Ep} - \beta \left(\frac{\sum_i E_i e^{-\beta E_i}}{Q} \right) \left(\frac{\sum_i p_i e^{-\beta E_i}}{Q} \right) \\ &= -\overline{p} + \beta \overline{Ep} - \beta \overline{E}\overline{p}\end{aligned} \tag{3.21}$$

We can do the same derivative for $\overline{p}$, but in terms of β. Doing this, we get the following in Equation (3.22).

$$\frac{\partial \overline{p}}{\partial \beta} = \overline{E}\overline{p} - \overline{Ep} \tag{3.22}$$

Putting Equation (3.21) and Equation (3.21) together, we get the result in Equation (3.23).

$$\frac{\partial \overline{E}}{\partial V} + \beta \frac{\partial \overline{p}}{\partial V} = -\overline{p} \tag{3.23}$$

There are known thermodynamic relationship [101] that relates temperate T, volume V, energy E and pressure P and one is given in Equation (3.24) along with a rearranged form in Equation (3.25)

$$\frac{\partial E}{\partial V} - T \frac{\partial p}{\partial T} = -p \tag{3.24}$$

$$\frac{\partial E}{\partial V} + \frac{1}{T} \frac{\partial p}{\partial \frac{1}{T}} = -p \tag{3.25}$$

Looking at Equation (3.25) and comparing it to Equation (3.23), we can see that there is now a relation for $\beta = \frac{constant}{T}$ where the constant can be found to be independent of the system in question. This constant is known as the Boltzmann constant k_B.

Our expression for the canonical partition function can now be written in the form given in Equation (3.26).

$$Q(N, V, T) = \sum_l e^{-\frac{1}{k_B T} E_l(N,V)} \tag{3.26}$$

If we consider the logarithm of Equation (3.26), we can notice that if we differentiate by T or by V, we can get some familiar results that are presented in Equation (3.27).

$$\begin{aligned}\frac{\partial \ln Q}{\partial T} &= \frac{1}{Q}\frac{\partial Q}{\partial T} = \frac{1}{Q}\sum_l \frac{\partial}{\partial T}\left(e^{-\frac{1}{k_B T}E_l(N,V)}\right) \\ &= \frac{1}{Q}\sum_l e^{-\frac{1}{k_B T}E_l(N,V)} E_l(N,V)\frac{k_B}{(k_B T)^2} \\ &= \frac{k_B}{(k_B T)^2}\frac{\sum_l E_l(N,V)e^{-\frac{1}{k_B T}E_l(N,V)}}{Q}\end{aligned} \tag{3.27}$$

Looking at Equation (3.27) and Equation (3.18), we can see that we can then express $\overline{E}$ as $\overline{E} = k_B T^2(\frac{\partial \ln Q}{\partial T})$.

It can be found that $Q(N,V,T)$ is directly related to the Helmholtz free energy A in the canonical ensemble [102]. This expression for A given in Equation (3.28).

$$A = -k_B T \ln Q \tag{3.28}$$

Equation (3.28) mirrors another important relation between the potential of mean force (PMF) and the probability distribution P given by Equation (3.29).

$$W = -k_B T \ln P \tag{3.29}$$

The PMF and the free energy can be considered equivalent to each other when we are dealing with a PMF that reduces the fundamental dimensionality of the system, and absorbing the entropic effects of the reduced dimensions.

We have now found ways to find the free energy using the probability distribution of an ensemble of systems. What follows is a brief digression on Hamiltonian mechanics due to its relevance on the later methods for calculating free energy differences.

3.3 The Hamiltonian

Classical mechanics has its origins in Newtonian mechanics, whose laws of motion continue to be used to this day [89]. The most important law for our consideration is Newton's second law, which states that an object's change in velocity (or more accurately an object's momentum) is directly related to force applied to that object. This is given simply in Equation (3.30).

$$\vec{F} = \frac{d\vec{p}}{dt} = \dot{\vec{p}} \tag{3.30}$$

As a brief reminder, the momentum is just the object mass multiplied by its velocity $\vec{p} = m\vec{v}$. Velocity is simply $\vec{v} = \frac{d\vec{r}}{dt}$ where $\vec{r}$ is the position vector of an object in cartesian space. $\vec{v}$ is also commonly written as $\dot{\vec{r}}$. This law has powerful applications, however, it does have its limitations. This is because getting an expression for the total force can be a challenge for some systems. As an alternative, Lagrange [103] (along with the assistance of others including Euler [104]) developed his own approach to classical mechanics. His version of Newton's second law is given in Equation (3.31).

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\vec{r}}}\right) = \frac{\partial L}{\partial \vec{r}} \tag{3.31}$$

L is a function called the Lagrangian and is defined as $L = K - U$,where K is the kinetic energy of the system and U is the potential energy. An incredibly powerful property of Langragian mechanics is that Equation (3.31) remains true regardless of the coordinate system. We can actually rewrite Equation (3.31) in a more general form as Equation (3.32).

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\vec{q}}}\right) = \frac{\partial L}{\partial \vec{q}} \tag{3.32}$$

$\vec{q}$ is a position vector in any general coordinate system and $\dot{\vec{q}}$ is referred to as the generalised velocity. This avoids the potentially complicated coordinate transformation that would be required if we were to instead use Newtonian mechanics. An object's equations of motion are now also given in terms of the object potential (in the form of the Lagrangian), which can be easier to express in certain systems.

There exists an alternate form of Lagrange's system, developed by William Rowan Hamilton [105], which rather than being in terms of the generalised velocity $\dot{\vec{q}}$, is instead in terms of the generalised momentum $\vec{p}$. This is defined as $\vec{p} = \frac{\partial L}{\dot{\vec{q}}}$. This change results in Newton's second law being in the form of two equations rather than just one. These two equation are given in Equation (3.33).

$$\frac{\partial H}{\partial \vec{p}} = \dot{\vec{q}}, \frac{\partial H}{\partial \vec{p}} = -\dot{\vec{p}} \tag{3.33}$$

H is referred to as the Hamiltonian and it defined as $H = K + U$ i.e. the total energy of the object. The vector $\dot{\vec{p}}$ is the time derivative of $\vec{p}$. At first this appears to be an undesirable alternative to Lagrange or Newton since we now actually have double the number of ordinary differential equations (ODEs) that we need to solve. Its change of focus to the total energy of the object (as apposed to force in Newton, and potential in Lagrange) actually turns out to make this form of mechanics incredibly powerful. So powerful in-fact,

that its applicability is not simply limited simply to classical mechanics, but actually extends itself to the seemingly unrelated field of quantum mechanics. This is demonstrated by its appearance in the central equation of quantum mechanics, the Schrödinger equation.

Now that we have a system of mechanics that can be applied in both classical and quantum mechanical systems, we can now consider some methods to calculate free energy, some of which will utilise the Hamiltonian.

3.4 Physics-based Free Energy Methods

Free energy differences are an important quantity to calculate when studying dynamic systems. In the field of chemistry in particular, it is essential to understanding reaction dynamics and determining whether a reaction is spontaneous or not [106]. In addition, free energy differences are used to determine which chemical structures are energetically favourable and which are not. In this work, the type of free energy that is referred to as just the 'free energy', is the Helmholtz free energy.

The Helmholtz free energy represents the amount of 'useful' work that a system at constant temperature can do. Studying the change in A when going from one state to another is therefore often more instructive that simply studying the change of internal energy. This is because it excludes the energy lost as heat, as well the the energy involved with creating the system, neither of which effects the dynamics of a system in equilibrium.

When studying complex systems, it is often highly informative to pick out key variables of the system that one believes are the most important quantities governing a given reaction or dynamic structure. This may be done by either selecting 'by-hand' based on ones knowledge of the system, or using some form of automated principle component analysis [107] [108]. These chosen variables are referred to as reaction coordinates or collective variables. For example, reaction coordinates could be the distance between two atoms, the angle between two molecules, or the puckering configuration of a molecular ring. Measuring how the system's free energy changes as these reaction coordinates are varied, is important for understanding what values of the reaction coordinates represent minimum energy states. A significant challenge is that these reaction coordinates are almost always impossible to measure experimentally, since they represent relations between individual atoms and molecules. We typically do not have the means to measure objects at this length scale without significantly altering the states they are in [4] [5]. We are therefore forced to use computational techniques such as MD in order to study interactions at these scales.

These computational techniques do come with their own limitations. To have a proper understanding of the free energy landscape, we must have measured

values for the free energy at all possible values of the reaction coordinates. This is nearly impossible when using techniques like MD since certain reaction coordinate values may require such high energies to be reached, that they are almost never sampled under normal thermal conditions. We therefore require the development of specialised methods in order to properly sample the full free energy landscape [109].

A recent review by Barros *et. al.* [110] on free energy methods developed for multi-scale simulations highlights the two main categories of free energy methods. The first category is those methods that allow the use of alchemical transformations, and therefore can calculate the free energy difference through non-physically possible transformations. The second category of methods are those that calculate free energy differences through strictly physical transformations.

Non-physical methods were the initial category to be developed, and they include: Thermodynamic Integration [111], Free Energy Perturbation [112], and Bennet Acceptance ratio [113].

Physics-based methods to calculate the free energy profile have also been developed and these include: Umbrella sampling [12], Adaptive Biasing [13], the Window Method [114], and Flat Histogram methods such as Metadynamics [115] or Local Elevation [116]. These methods are the ones most directly related to FEARCF and are therefore explored in detail in the sections below.

3.4.1 Umbrella Sampling

Umbrella sampling was first proposed by Torrie and Valleau [12], and was created to overcome issues with standard Monte Carlo (MC) simulations when they were unable to overcome large energy barriers in the potential landscape. To solve this, Torrie and Valleau proposed running Markov-Chain simulations with the probability given in Equation (3.34).

$$P(\vec{q}) = \frac{w(\vec{q})e^{-\frac{U(\vec{q})}{k_B T}}}{\int w(\vec{q'})e^{-\frac{U(\vec{q'})}{k_B T}} d\vec{q'}} \tag{3.34}$$

$w(\vec{q})$ is a weighting function designed to ensure that sampling is not impeded by energy barriers. Using the Markov-Chain method with the distribution in Equation (3.34) is equivalent to including a biasing potential to the Hamiltonian of the system of the form $U(\vec{q}) = -k_B T \ln w(\vec{q})$. Figure 3.1 shows the affect of umbrella sampling on the probability distribution of soft sphere fluid. The weighting function is specifically chosen to favour configurations with more negative energy.

With sufficient sampling achieved, calculation of thermodynamic properties is possible by sampling with the weighting function used to generate the

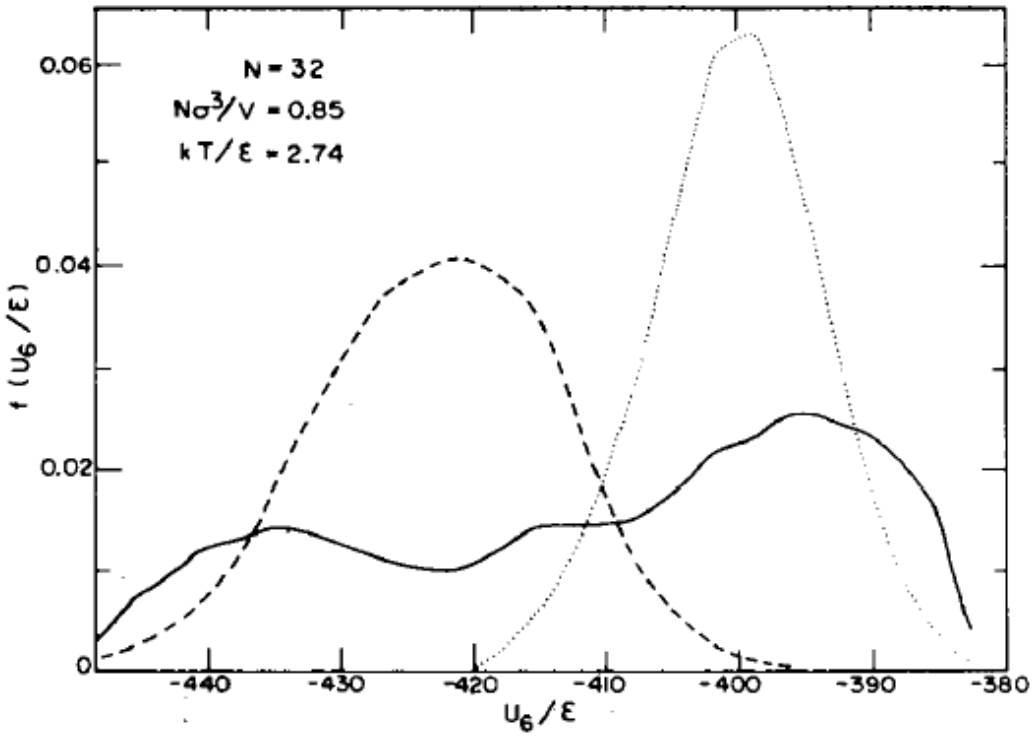


Figure 3.1: Results of umbrella sampling applied to 32 soft-sphere fluid. The curve with a solid line is probability distribution with umbrella sampling, the curve with a dotted line is unbiased probability distribution, and the curve with broken lines is representation of the weighting function. Reprinted from Journal of Computational Physics, Vol 23, Issue 2, G.M. Torrie,J.P. Valleau, Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella sampling, 187-199, Copyright (1977), with permission from Elsevier [12].

biased simulation. If A was needed to be calculated, this could be done using Equation (3.35).

$$< A >= \frac{< A/w >}{< 1/w >} \tag{3.35}$$

3.4.2 Adaptive Biasing

The problem with the above approach is that it requires some prior knowledge about the system's probability distribution in order to select an appropriate weighting function. To overcome this problem, the adaptive umbrella sampling method was developed by Mezei [13]. In this method the biasing force is actually derived from an estimate of the system probability distribution function that is iteratively generated after n iterations i.e. $U^n(\vec{\xi}) = k_b T \ln P^n(\vec{\xi})$ where $\vec{\xi}$ is some set of reaction coordinates. The means to calculate $P^n(\xi)$ is given in Equation (3.36).

$$P^n(\vec{\xi}) = \sum_{i=1}^{n} r_i(\vec{\xi}) N_i p_i(\vec{\xi}) \tag{3.36}$$

$$r_i(\vec{\xi}) = \frac{f_i(\vec{\xi})}{F_i(\vec{\xi})} \tag{3.37}$$

$$F_i(\vec{\xi}) = \sum_{j=1}^{i} f_j(\vec{\xi}) \tag{3.38}$$

Here $f_j(\xi)$, in Equation (3.38), is the number of sampled steps for the specific ξ value, $r_i(\xi)$ is a weighting that increases for ξ values that are highly sampled, $p_i(\xi)$ is the probability of ξ from the i^{th} iteration, and N_i is a normalisation constant that needs to be found by minimising the relative deviation square sum (see Mezei [13]). The weighting function is then updated, or 'adapted', based upon the approximation of the probability distribution.

3.4.3 Window Method

An alternative method is called the Window Method [114], in which a potential is actually built up as a sum of individual potentials, rather than adding a single potential term. This is useful when there are multiple energy barriers that need to be overcome. The total phase space is divided into n number of 'windows' and a harmonic function is then defined which is centred on each window. These harmonic functions take the place of the biasing potential by constraining the system within each window. This then gives a total biasing potential of the form given in Equation (3.39).

$$U(\vec{\xi}) = \sum_{i}^{n} k_i(\vec{\xi} - \vec{\xi}_i)^2 \tag{3.39}$$

In Equation (3.39), k_i is the unique spring constant and $\vec{\xi}_i$ is the centre of the i^{th} harmonic potential. The problem with this method is that for dimensions higher than two, the number of windows scales exponentially therefore making its application limited to low dimension systems.

3.4.4 Flat Histogram Methods

Wang and Landau [117] were the first to describe an algorithm that has resulted in a family of related methods, called flat histogram methods. The aim of all these methods is to create conditions in a molecular system that allows for all system energy states to be equalled sampled. When this sampling is plotted as a histogram, it will form a flat curve. This is the reason for the name flat histogram methods. Wang and Landau originally described the algorithm for MC simulations, but other methods have since been developed that can be applied in both MC and MD simulations.

Local Elevation

Hober, Torda, and van Gunsteren [116] described the local elevation method which uses a gaussian-shaped, sampling penalising function V_{pen} to allow the system to overcome high energy barriers. The general form of the penalising function is given in Equation (3.40). In this equation, χ is the current system conformation, χ^0 is the previously visited conformation, and n_{χ^0} is the number of times χ^0 has been visited before. The shape of the gaussian is given by the height k_{pen} and width w parameters.

$$V_{pen}(\chi) = k_{pen} n_{\chi^0} e^{-(\chi-\chi^0)^2/2w^2} \tag{3.40}$$

For multiple dimensions, the penalising function is expressed as a product of gaussian functions. This has the effect of reducing the effectiveness of the penalising function, for each dimension added.

Metadynamics

A more recent method developed by Liao and Parrinello, called Metadynamics [115], takes inspiration from both the window method and adaptive umbrella sampling. Like the window method, it uses a sum of individual potentials to form a net potential but instead of harmonic potentials, it uses inverted gaussian potentials. The choice of this potential form is because both the height and the width of gaussians can be adjusted. There is notably no defined separation into windows between the individual potentials unlike in

the window method. Rather the method is designed to be more flexible with variable distances between the potentials. The net biasing potential is given in Equation (3.41) in terms of the n reaction coordinates given by $\vec{\xi} = (\xi_1, \xi_2, ..., \xi_n)$.

$$U(\vec{xi}) = -\sum_{i}^{n}\sum_{k=1}^{N_i} W_{i,k} e^{-(\xi_i - \xi_{i,k})^2/2\sigma_{i,k}^2} \tag{3.41}$$

There are N_i gaussian potentials for each reaction coordinate, and each potential is defined by its mean $\xi_{i,k}$ and variance $\sigma_{i,k}^2$ along with a weight $W_{i,k}$. Metadynamics is made adaptive by the fact that during a simulation, the $W_{i,k}$, $\xi_{i,k}$ and $\sigma_{i,k}^2$ terms are updated every set number of steps. An example of the evolution of the net potential over the simulation is given in Figure 3.2. The simulation begins in the centre well, then gradually constructs gaussian potentials in order to escape this well into the left well. It then repeats this process until the net potential is able to overcome the energy barrier separating the right well from the centre and left well.

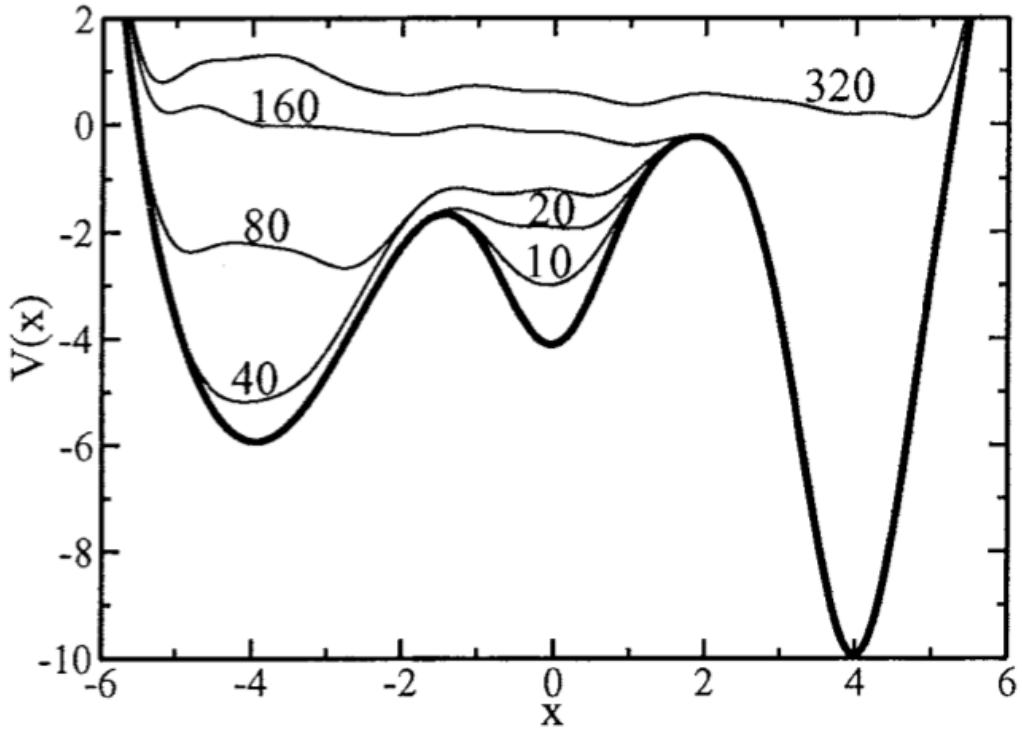


Figure 3.2: 1D example of the evolution of metadynamic's net bias potential over simulation time. The unbiased potential is given in bold, while the other curves show the biasing potential after the labeled number of dynamic iterations. Reprinted with permission from Proceedings of the National Academy of Sciences, Vol 99, Issue 20, A. Laio,M. Parrinello, Escaping free-energy minima, 12562-12566, Copyright (2022), National Academy of Sciences [115].

Figure 3.2 also illustrates why metadynamics falls under the category of flat histogram methods.

The problem with this method is that a large number of gaussian terms may be required to increase sampling at every point in phase space, and this problem is further magnified by number of reaction coordinates there are. In addition to this, Jacobian transformation are required to transform this biasing term, which is defined in terms of the 'lab' reference frame, into the molecular reference frame. These Jacobian terms become more complicated the more dimensions there are [118]. Metadynamics also runs in a singular simulation, which limits the ability of the method to be parallelised.

FEARCF

Another flat histogram method, FEARCF [16], will be explored in further detail later in this thesis. For now, it is sufficient to note that was distinguishes FEARCF from other flat histogram methods is its use of a numerical potential instead of analytical potentials such as gaussians or harmonic potentials.

With the establishment of the existence of methods to numerically calculate the free energy profile, we take another brief detour to discuss another useful numerical calculation.

3.5 Time Correlation Functions

An important method for studying non-equilibrium systems is that of time correlation functions [10]. This method essentially considers how a certain time dependant property changes over time. We have previously discussed how the Hamiltonian approach describes a system using only a set of generalised coordinates $\vec{q}$ and their associated generalised momenta $\vec{p}$. In a dynamic system, these quantities are time dependant and their evolution over time can be calculated using the Hamiltonian equations of motion, given in Equation (3.33). Any dynamic quantity L that we can calculate should be dependant upon $\vec{q}$ and $\vec{p}$. These are in turn are dependant upon time t so we can implicitly write L as a function of time i.e. $L(t)$. The time correlation function $C(t)$ of $L(t)$ is defined in Equation (3.42).

$$C(t) =< L(0) \cdot L(t) >= \int .. \int d\vec{p}d\vec{q}L(0) \cdot L(t)f(\vec{p},\vec{q}) \tag{3.42}$$

In Equation (3.42), $f(\vec{p},\vec{q})$ is the equilibrium distribution function in the phase space described by $\vec{p}$ and $\vec{q}$. Equation (3.42) is the general case where $L(t)$ can be a vector. If L is just a scalar, then $L(0) \cdot L(t)$ is instead just the normal scalar multiplication. It is also more accurate to call equation (3.42) the autocorrelation function since we can actually calculate the time-

correlation function of two completely different quantities, for example $M(t)$ and $L(t)$ and this written as $C(t) =< M(0) \cdot L(t) >$.

Let's consider a simple example to show how useful time correlation functions can be. Let us consider the time correlation function (or auto correlation function) of the velocity $\vec{v}$ of a particular particle in a multi-particle system. This is given by $C(t) =< v(0) \cdot v(t) >$. We expect that $C(t)$ would initially be highly correlated as the particle continues with its initial velocity, however, as it hits other particles the magnitude and direction of its velocity will change and therefore we expect the correlation to reduce to zero as $t \to \infty$. The shape and gradient of this change of correlation should be characteristic of the system and we could model it as an exponential function $C(t) \approx e^{-t/\tau}$ where τ is obviously related to the gradient of the exponential and has the same units as time. We call τ the time constant of the correlation function. τ gives a value to the speed of decorrelation of $v(t)$ and can therefore be used to characterise the system.

Another use of the velocity time correlation is the calculation of the diffusion coefficient [7] given in Equation (3.43).

$$D = \frac{1}{3} \int_0^\infty < v(0) \cdot v(t) > dt \tag{3.43}$$

In this chapter we have discussed in detail several methods for calculating the free energy of a molecular system. In the next chapter, we will explore concepts in software, libraries object-oriented programming, and graph theory. This is in preparation for the introduction of the free energy method, FEARCF, that this thesis is dedicated to developing, and how it is implemented in a software library.

Chapter 4

Software Library Design

In this chapter we detail some of the principles behind software libraries and then explore the concepts of object-orientated programming and graph theory. This is to motivate their inclusion in the design and integration of FEARCF into a software library.

4.1 Software Libraries

The first use of the term 'library' in the context of computing dates back to 1947 with Herman Goldstein and John van Neumann and their work on the IAS machine [119]. The first true software library, however, was implemented by Maurice Wilkes at Cambridge with his work on the EDSAC machine. This was actually a physical library of subroutines in the form of punch paper tape that would be loaded, along with a main program, into the machine [120].

In the modern sense, a software library is a collection of any number of subroutines, classes, objects, and data, that is partitioned independently from other code. A software library must also have a well-designed interface so that other programs may access and use specific parts or the whole library [121]. Libraries are designed to be assessed by any number of programs regardless of whether they are related to each other or not. Libraries therefore represent a form of modularity in code design, due to the ability to share and utilise a library independent from other programs. Libraries usually contain commonly used data or subroutines, which prevents programs written after the creation of the library from having to explicitly define and allocate these subroutines and/or data every time they are needed. Instead a call is made to the library in a specific manner, which prevents the writer of the new program from having to necessarily know how the library explicitly functions. The writer need only know the syntax of the interface.

Most programming languages come with a standard library for things like common mathematical functions, however, since the implementation of sub-

routines in FORTRAN II [122], users have been able to write and implement their own libraries. The inclusion of modules in Fortran 90 further encouraged the creation of custom libraries [123].

Libraries are typically distinguished by the way that they are implemented in desired programs. Libraries that have their objects files linked (typically to archive files in UNIX systems) and then compiled with the program in question, are referred to as static libraries [124]. This is because copies of the library object files are contained within the program executable, and are therefore not subject to change i.e. static. In order to change a part of the library, the program would need to be recompiled. Libraries that only have their object files called and linked during the running of a program executable, are referred to as dynamic libraries [125]. This is because in between runs of the program executable, the library object files may dynamically change without causing the program executable to be recompiled. The problem this presents, however, is that other static libraries in the program executable that depend on the dynamic library may no longer function due to these dynamic changes. This in turn causes what is known as 'dependency hell'. A visual illustration of the difference between static and dynamic libraries is given in Fig 4.1.

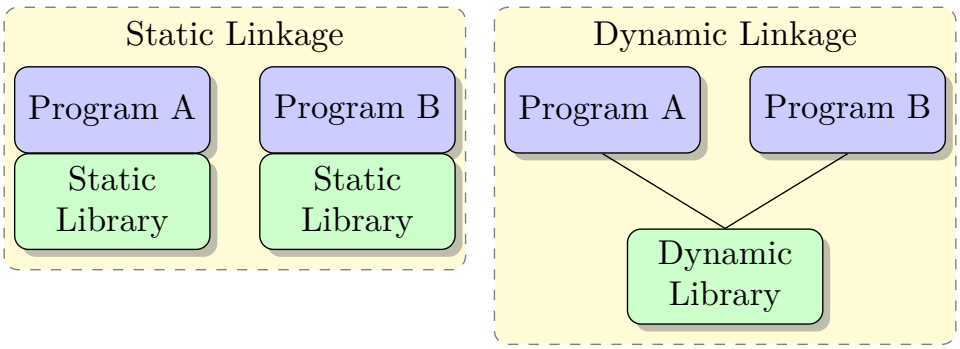


Figure 4.1: Illustration of difference between a statically linked library and a dynamically linked library.

Static libraries are typically faster at runtime since the object files are already contained within the executable and do not need to be retrieved. However, this means that every program is required to contain the library object files, in essence creating multiple copies. This is different to dynamic libraries which require only one set of object files, which can then be accessed by any number of programs during their runtime. But any issue that develops with the dynamic library may render all dependant program executables unusable, which is not a problem with a static library [124].

For FEARCF, the decision was made to implement it as a static method. The primary reason for this is that the FEARCF method remains proprietary,

therefore a distribution of precompiled executables is preferable to providing a dynamic library that casual users can access and possibly change. The FEARCF library is still maintained on an internal Git code repository for SCRU developers to access, submit suggestions, and track changes. In addition, because FEARCF is still in development, public releases of MD software packages do not come compatible with FEARCF as standard and therefore need to have their source code modified. This creates greater chance of dependency issues since any copy of the MD software needs to be modified by hand. It is therefore easier to simply provide users with a static, pre-compiled executable. The multi-dimensional splining that occurs within FEARCF also represents considerable compute time, particularly in classical simulations, therefore any possible speed advantage of a static library is desirable. It should be noted, however, the speedup of static versus dynamically compiled versions of the FEARCF library has not been explicitly tested.

Now that we have introduced the concept of software libraries, a discussion of the object-orientated programming paradigm is presented as a basis for the design of the FEARCF software library.

4.2 Object Orientated Programming

The formal beginnings of object-orientated programming (OOP) lie with the language Simula, developed by Kristen Nygaard and Ole-Johan Dahl at the Norwegian Computing Centre from 1962 onwards [126]. It should be noted, however, that some of the key concepts had been in development since the late 1950's at MIT [127]. Later developments in OOP include the languages SmallTalk, Lisp, Eiffel, and perhaps most importantly, C++, an extension of the C language centred wholly around OOP [128]. In the modern era, many of the most popular programming languages, such as Java and Python, incorporate OOP and procedural design in a hybrid approach. Many older languages, including Fortran, have also been subsequently modified to incorporate many of the design ideas central to OOP. In this section I will be exploring some of the core tenants of OOP and attempt to justify why they are suited to the FEARCF software library design.

To begin our discussion of OOP, we need to define two core terms namely: *classes* and *objects*. An object is the fundamental unit of OOP and is defined as a set of data, as well as the set of requisite functions or subroutines required to access, manipulate, and extract that data. Objects can be used to represent an almost endless number of possible items of interest including: individual atoms, a human beings in a database, or nodes in a complex network. Every object is importantly a member of a specific class. A class is a general description of the common data types and methods that can be used on those data types. These methods should include how objects can be created, as well as how objects might interact with each other. As an example

of the relationship between classes and objects, if we had an object defined as 'Tom' then we could say it belonged to the class 'human'. Every object can be thought of as an *instance* of a specific class.

When we are defining a class, we are also defining the attributes of each and every possible object that could fall under that class. This OOP approach means that we must have some idea of what objects we are expecting to handle in our code, and what commonalities they have. If we expect to have multiple objects that are similar, we can potentially save time by defining a class for these objects. Instead of defining the data types and related functions for each possible object (this is the 'procedural' method), we can instead define these attributes in a class and then create our objects as instances of this class.

Now that we have a clear understanding of the components of OOP and why we might choose to implement them, we can now discuss three other important traits of OOD: encapsulation, inheritance, and polymorphism.

Encapsulation is the concept of tying together a set of data with the functions used to access or manipulate it, so that they are considered as a single entity. This can allow for the functions to be written very explicitly for the data, as well as ensure that data is only accessed by these functions.

Inheritance is the ability to define classes as either a subset of another class or as a 'parent' of another class. The 'parent' may have all the data and functions of the 'child' class but with additional features. With inheritance, we can construct a network of interrelated classes. This presents another opportunity to save programming time by reusing features of a 'parent' class.

Polymorphism describes a type of function that is able to take different types of objects as inputs, identify what class of object it has been given, and then act accordingly. This again saves programming time by avoiding having to rewrite the same function for every possible class we have defined.

These are the core essentials of OOP and below will be discussed why OOP is a useful approach for designing the FEARCF software library.

4.2.1 Benefits of OOP for FEARCF

We now consider why OOP may be suitable for a program that aims to implement the FEARCF method in an optimised and intuitive manner. The principal components of FEARCF are the reaction coordinates. While the reaction coordinates for each system are unique, they will share certain qualities such as: being distances, angles, dihedrals, etc. In addition, all reaction coordinates will need to be defined using the atoms in the molecules, either directly or indirectly. Atoms themselves have shared qualities such as having mass and coordinates in cartesian space. Reaction coordinates and atoms therefore make good potential candidates for classes. This is because the

large number of molecular systems we wish to model will all require defining atoms and reaction coordinates, and defining these concepts over and over again for each type of system is a very inefficient way to write a program.

As a class, atoms require three types of data, namely: the atoms mass, the atomic coordinates, and the atomic forces. In terms of the subroutines for this class, the most obvious two that come to mind are a subroutine to set the values of a particular atom and a subroutine to read already stored values for a particular atom. These are referred to as 'set' and 'get' subroutines respectively.

Now we consider the potential classes that will be used for reaction coordinates. The reaction coordinates types include: distances, angles, dihedral angles, vectors, planes, centre of masses, and pucker angles. The data needed for these reaction coordinates are primarily the value of the reaction coordinate itself. For most reaction coordinates (distance, angle, etc.) this is just a singular scalar value, but for other (vector, plane which is define by its normal vector, etc.) it will need to be a 3D array representing a vector or a cartesian position. The reaction coordinates classes will also need to store force terms representing either the partial derivatives from the splining routine, or the product of partial derivative terms. The reaction coordinate classes will also require 'set' and 'get' subroutines.

A less obvious insight is that we will also need to define a class for the FEARCF as a whole. This is to allow for storage and access of all the partial derivatives from the splining routine, the chosen reaction coordinate values, and the complete set of atomic forces. These are needed to be kept separate from the atom and reaction coordinate classes because multiple objects may need to retrieve or write to these values. It is more efficient to have them stored as single arrays in a FEARCF class, rather than having duplicates for each created atom or reaction coordinate object. A set of subroutines are also needed to retrieve these values, write them to output files, and export these values.

The other important set of three subroutines for each type of reaction coordinate are those needed to define the reaction coordinate, calculate the reaction coordinate values, and then calculate the partial derivative term. While the obvious place to put these subroutines would be in the reaction coordinate classes themselves, this turns out to not be the right approach. Rather, these subroutines need to be a part of the FEARCF class. The reason for this is that these subroutines need to facilitate communication between classes i.e. a distance reaction coordinate and the two atoms used to define it. If defined inside a class, they would in turn need to define each class that they would ever need to communicate with, defeating the usage of classes in the first place. In addition, the FEARCF class contains a list of all the defined atoms/reaction coordinates and so can return an error if an attempt is made to communicate with one that has not been defined. These three sets of

subroutines are therefore defined as a set of methods within the FEARCF class.

We next consider how OOP is able to be implemented within the Fortran programming language that FEARCF was previously developed in.

4.2.2 OOP in Fortran

FEARCF was originally developed in CHARMM, which is primarily written in Fortran, specifically Fortran 95. Most of the legacy FEARCF code is therefore written in Fortran. The decision was then made to write the FEARCF software library in Fortran in order to avoid having to rewrite all of FEARCF in a new language. But with the decision to utilise OOP in building the structure of the FEARCF library, we must now consider the capacity that Fortran has for object-oriented programming.

Before the Fortran 90 standard, it was not possible to write OOP code in Fortran. But Fortran 90 introduced several key features that made OOP possible such as declarable data types and interfaces [123]. In Fortran 90, classes can be defined using the 'module' block. In a 'module' declaration, there is firstly a 'type' declaration which defines the principal object the class is associated with, as well as the variables contained in this object. This is then followed by a 'contains' statement which proceeds to list all the function and subroutines definitions that are required for this class. In this way, the 'module' block fulfils the encapsulation trait of OOP, since the object is directly connected to its associated subroutines and/or functions. This can be made more explicit because the default behaviour in Fortran is to set all object variables to public which allows for outside programs to access and modify them. If they are instead chosen to be private, the 'module' will require a specific function to be defined in order to allow for outside programs to create new object instances, as well as modify objects that already exist.

Related classes will often share common methods and Fortran's way of handling these common methods is the 'interface' block. In an external program, all the requisite classes can be loaded through a set of 'use module_name' statements. Once this is done, an 'interface' block can be created in which a method for each potential class type must be given. After this is done, the program no longer needs to call the specific method defined for each class but can simply call the interface which will correctly select the correct method depending on the object type it is called with. This is an example of polymorphism since it relies upon the common characteristics of related classes.

Fortran 90 does not have implicit support for class inheritance however [19]. Fortran 95 introduced the ability to reference other types as variables when creating a new type but this is still not true inheritance. Only in the Fortran 2003 standard was class inheritance introduced via the 'extend' command [129]. With CHARMM now written in Fortran 95 [130] and NWChem written

primarily in Fortran 90 [131], it was decided to not write the FEARCF library in Fortran 2003. The choice that was left was then whether to write it in Fortran 90 or 95. It was decided that the ability to declare types as variables in other types was not enough to warrant the use of Fortran 95. This is because while using types inside of other types might be useful, such as referring to the atoms that define a distance reaction coordinate, this would limit the atoms to only being usable for that particular reaction coordinate. If we wanted to use the same atoms to define multiple reaction coordinates, multiple objects would need to be created for each atom, one for each reaction coordinate type. This approach would defeat the purpose of OOP.

It is therefore been established that the version of Fortran that FEARCF is written in does have most of the capacity to be written in OOP. To further argue why OOP is a suitable paradigm for designing a FEARCF library, the section will discuss graph theory and how it relates to the definition of reaction coordinates in FEARCF.

4.3 Graph Theory

Graph theory is a branch of mathematics that was begun by the renowned mathematician Leonard Euler in 1736 with his treatment of the Seven Bridges of Königsberg problem [132]. Since then it has gone on to find applications in numerous fields including: physics, chemistry, biology, the social science, and computer science [133]. At its most basic level, graph theory is the study of two types of objects: vertices (or nodes) and the links between them called edges (or lines). Arrangements of these two objects can model a large variety of problems. The study and characterisation of the types of graphs one can construct is the motivation behind graph theory. An example of a basic graph is given in Figure 1.5.

There are several considerations to make when deciding what type of graph to use to model a particular problem [134]. There are two main types of graphs: directed and undirected. In directed graphs each edge has a specified direction pointing from one node to another. In undirected graphs, edges have no such orientation. In Figure 1.5 the basic graph is an example of an undirected graph. Directed graphs are suited for systems in which the objects that the nodes represent have a one way relation to each other e.g. node A affects node B but not the other way around. Directed graphs can also be used for models with bi-directional relationships. This is possible if one wants to highlight that the relationship between two nodes is different depending on the direction. An example of a directional graph is given in Figure 1.5.

Graphs may also be classified as trees if any two node are connected by exactly one path of edges and vertices, and they have no cycles i.e. nodes connected in a loop. This definition is for undirected graphs, so directed graphs whose underlying undirected graph satisfies this definition are known as poly-trees

or directed trees [135]. Trees may also be 'rooted' by selecting one node as the root, which in turn may form the basis for the orientation of the rooted tree. Trees are very commonly found in many data structures in computer science, as well in biological taxonomy and language hierarchy. An example of a simple tree is given in Figure 1.5.

Graphs may be 'coloured' i.e. their nodes or edges assigned different colours either randomly or according to some constraints. The colours themselves may have no meaning other than to differentiate each vertex or edge. The colouring also may represent some commonality between vertices or edges of the same colour. The use of coloured graphs was most famously used in attempts to prove the four colour theorem [136], a postulate that states only 4 colours are needed to properly colour any map. This ultimately became the first accepted mathematical proof involving computers by Appel and Haken [137]. An example of a coloured graph is given in Figure 1.5.

While graphs are primarily depicted visually, then can also be expressed in a more abstract manner. One example is that of the adjacency matrix A. This is a square matrix where the i^{th} row and i^{th} column represent the i^{th} node in the graph. The elements of the matrix represent the edges of the graph with the element a_{ij} being defined as 0 if no edge exists between the i^{th} and j^{th} node, or 1 if an edge does exist. Assuming there are no loops in the graph (edges that connect a node to itself), this means that the adjacency matrix is by definition a diagonal matrix. The adjacency matrix is also symmetric about the main diagonal. The dimensionality $n \times n$ of A indicates there are n nodes in the graph, and half the sum of all its elements is the number of edges in the graph. The adjacency matrix of the basic graph in Figure 1.5 is given below, with the rows and columns matching the labelling assigned in Figure 4.2. The labelling in this case is arbitrary, since A could simply be subject to row and column swapping in order to produce any other possible labelling.

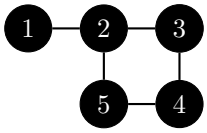


Figure 4.2: A graph with each node given a unique numerical label.

$$\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}$$

The definition of the adjacency matrix may be modified to be suitable for other types of graphs such as directed or weighted graphs. Other types of abstract representations related to adjacency matrices include: adjacency lists, distance matrices, incidence matrices, and degree matrices [138].

Now that we have introduced some basic graph theory concepts, we can explore previous work in chemistry that use graphs.

4.3.1 Graph Theory Applied to Chemistry

Graph theory has been seen as a useful tool for solving problems in chemistry for over a century. One of the earliest applications was the modelling of non-cyclic single bonded hydrocarbons (alkanes) as trees or rooted trees by Cayley [139]. In this model, the vertices represent the carbons in the alkanes molecules, while the edges represent the bonds between these carbons. The hydrogens are assumed to be whatever configuration completes the individual carbon valences and are therefore excluded. An example is given in Figure 4.3.

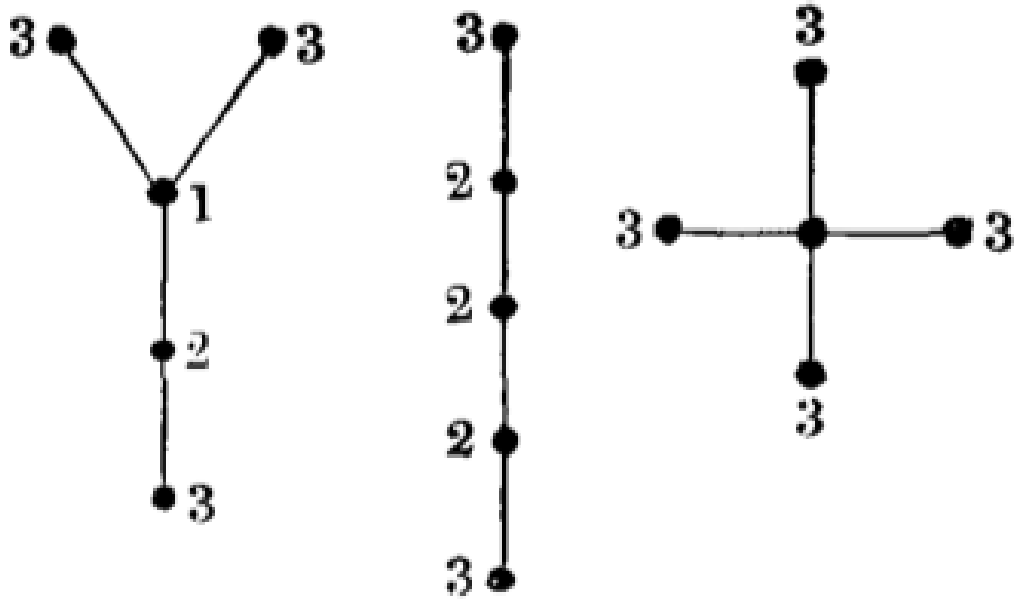


Figure 4.3: All three possible alkanes with five carbons as trees, with numbers indicating the number of hydrogens bonded to each carbon. Reprinted with permission from The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, Volume 47, Issue 314, A. Cayley, LVII. On the Mathematical Theory of Isomers, 444-447, Copyright (1874) Taylor & Francis [139].

The theory of trees was actually developed by Arthur Cayley himself and only much later applied to isomers [135]. Originally Cayley proved that the number of trees possible for n labelled trees is given by n^{n-2}. The number of trees for

n unlabelled vertices is not so simple however, and in fact no closed formula exists to predict this number. Instead, Cayley eventually turned his focus on trees as applied to chemical isomers. These present physical constraints that allow for possible prediction of the number of expected isomers. For alkanes there is a restriction on the number of bonds carbon may form, four, which in graph theory means that vertices are restricted from having degree (or valency) greater than four. The degree of a vertex is the number of edges connected to that vertex. Uses the notion of centred and bi-centered graphs, Cayley was able to correctly predict the number of alkane isomers up to 11 carbons. His values for 12 and 13 carbons, however, were later shown to be incorrect by Herrmann [140]. Further extensions of Cayley's work was done, most notably by Henze and Blair [141].

Cayley also proposed ways to calculate the number of isomers for alkyl derivatives. These are alkenes where one hydrogen is replaced with a bond to another element. This uses the notion of rooted trees since only one carbon in an alkyl will be short of a hydrogen. It is this method that Henze and Blair managed to use to calculate the number of alkene isomers of higher carbon numbers.

A more modern example of the use of graph theory is the Methyl Assignment by Graph Matching (MAGMA) tool developed by Pritišanac *et al.* [142] for the analysis of methyl based nuclear magnetic response (NMR) spectra. NMR spectroscopy is a technique used to study the structure of organic molecules in solution. The technique relies upon the specific resonance that some atomic nuclei experience when exposed to a near oscillating magnetic field within a larger constant magnetic field [143]. The resonant frequency is unique to each nuclei type, as long the nuclei in question has a non-zero nuclear spin. Otherwise the nuclei does not respond to NMR.

Normally NMR is restricted to small molecules of up to approximately 30 kDa, however, with the use of methyl-TROSY NMR, this restriction is lifted up to molecules of 1MDa [144]. This technique uses the ^{13}C isotope as a tag since the more common ^{12}C is not responsive to NMR. In proteins, the methyl containing amino acids (such as leucine, isoleucine, and valine) are tagged with ^{13}C and ^{1}H methyl groups, while the rest of the protein hydrogens are replaced with deuterium to ensure the methyl groups are the only resonant nuclei in the protein. The frequency spectra is then obtained, and the resonant frequencies need to be assigned to the respective atoms in the protein. This is the most difficult part, but several techniques such as FLAMEnGO2.0 and MAP-XSII have been developed based on Monte-Carlo methods [145] [146].

MAGMA uses graph theory by constructing two graphs. The first graph, called the 'data graph', is derived from NMR data and has vertices which represent the individual methyl resonances, while the edges represent inter-methyl correlations. The second graph, called the 'structure graph', is derived

from an actual structural model and has vertices which represent the carbon atoms in the methyl groups, while the edges represent methyl groups that are within a specific distance threshold of each other. The aim of MAGMA is to match these two graphs as closely as possible. Testing all possible arrangements would take an inordinate amount of time, and a further problem is that the data graph often has less edges due to limitations and effects leading to distortion in NMR data.

To address this, MAGMA uses two optimised graph matching algorithms, VF2 and McGregor, to eliminate the majority of possible arrangements. MAGMA also utilises an optimised ordering algorithm before and during the search, which significantly reduces the computational time required for a solution. And example of some matched graphs is given in Figure 4.4. Note how in the figure Peaks 1 to 3 are matched to specific residues of the protein by forming graphs that agree with each other.

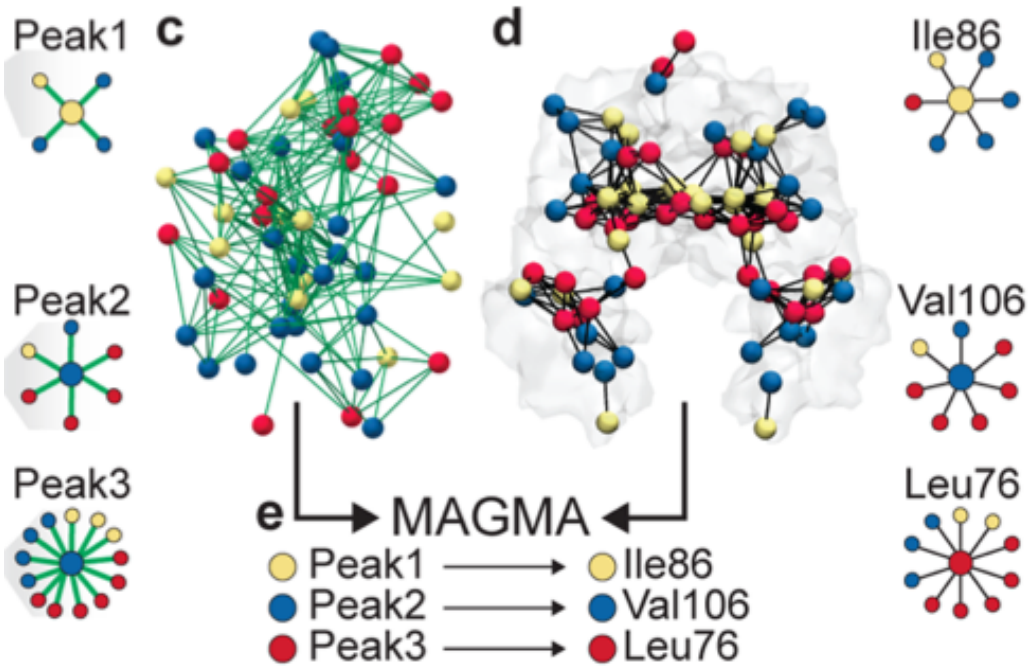


Figure 4.4: The two types of graphs that MAGMA constructs, the data graph (left) and the structure graph (right), that are then attempted to be matched. Reprinted with permission from Journal of American Chemical Society, Volume 139, Issue 28, I. Pritišanac *et. al.*, Automatic Assignment of Methyl-NMR Spectra of Supramolecular Machines Using Graph Theory, 9523-9533, Copyright (2017) American Chemical Society [142].

Now that we know graphs have been successfully used to study other problems in chemistry, we now discuss how graphs are utilised in FEARCF and how this use of graphs is made possible by the OOP paradigm in the FEARCF library.

4.3.2 Graphs in FEARCF

In this work the graphs that we can construct within FEARCF represent how reaction coordinates (RCs) are defined in terms atoms and other various defined quantities. The vertices represent several types of geometric quantities (atom coordinates, vectors, planes etc.) while edges represent the two relationships between any two quantities. To be precise, these relationships are: how a quantity is defined in terms of the other, and how the partial derivative of one quantity is defined with respect to the other. Since we will have variables of the same type (i.e. we will have multiple atoms), we can colour our vertices to represent the atom type as well as all possible RC types. If we wished, we could also select a colour for each edge type, since there might be multiple instances of one type of geometric quantity used to define another. In this work, however, we will refrain from colouring the edges and just colour the nodes.

We can construct an example graph representing a simple one dimensional case of an atom A_1 and a reaction coordinate ξ_1, defined as some function of the atomic coordinates $f(A_1)$. We can construct the simple coloured graph shown in Figure 4.5.

Figure 4.5: Coloured FEARCF graph for a single atom A_1 and a reaction coordinate ξ_1 defined in terms of A_1.

We could, if we wanted to, draw edges that represent the two types of relationships that connect different variables in FEARCF, namely the function f that relates A_1 to ξ_1 and the partial derivative of ξ_1 with respect to A_1. This is illustrated in Figure 4.6.

An early graphing approach defining RCs was implemented in the CHARMM module RXNCOR [130]. A limited precursor to FEARCF expanded on this to undertake QM/MM reactions relying on a pair of distance RCs in two dimensions with umbrella sampling [15]. The original implementation in CHARMM was limited by umbrella sampling because it is free energy method not ideally suited to free energy simulations of more than two dimensions. In addition, umbrella sampling is not efficiently parallelizable. The limit to two dimensions is due to the analytical nature of the biasing potential used in umbrella sampling that requires increasingly complex corrections with increasing di-

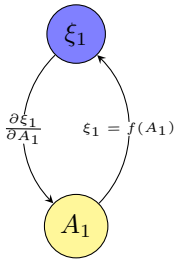


Figure 4.6: Coloured, directed FEARCF graph for a single atom A_1 and a reaction coordinate R_1 defined in terms of A_1, with directed edges representing the possible calculations that can be made between the two.

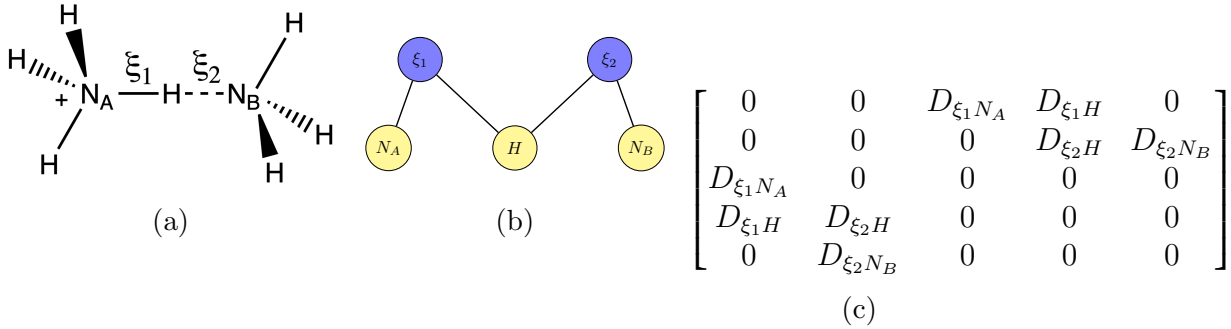
mensionality. An implementation of the adaptive umbrella sampling method in CHARMM, which uses WHAM to combine results from multiple parallel simulations, has not been developed further to date.

The shortcoming in the current CHARMM implementation of FEARCF is that the calculations of the variables and the partial derivatives are contained with two encompassing subroutines, *ASCEND* and *DESCEND*, respectively. These subroutines are not constructed using the OOP specific 'type' command, but rather the 'case' command contained within a *for* loop that cycles through the defined nodes. In contrast to this, the FEARCF library contains these calculations in separate subroutines within the FEARCF class. The reason for this is that it allows multiple calculations to be run at the same time, which cannot be done if the calculations are all contained within a sequential loop. The CHARMM implementation also writes explicit calculations for each variable, depending on what other type of variables are used to define this first variable. For example, a vector can be defined as the vector between two points in space, or it may be defined as the cross product of two other vectors. In CHARMM, there is a 'case' statement for each possibility. In the FEARCF library, however, there is still only one vector object. There is simply differing calculations within the same subroutine depending on what the defining objects are detected to be.

To make it more clear how the graphing representation is related to the OOP paradigm of FEARCF, abstract representations below are used to illustrate how FEARCF calculates atomic biasing potentials.

An associated matrix to the adjacency matrix can be defined, which we name the derivative matrix D. Rather than simply having 1s representing the edges, D has these replaced by the terms $D_{ij} = \frac{\partial i}{\partial j}$, which is the partial derivative of node i with respect to j. The atomic biasing forces can then be found by finding a path in the derivative matrix from each reaction coordinate, to each connected atom node. The net atomic biasing force on an atom can then be expressed as the sum of the products of the derivatives along each path

(this is detailed in the next Chapter). An update of the previously computed proton exchange reaction ($H_3N-H-NH_3$) in CHARMM [147], now defined using a graph representation of the reaction coordinates in FEARCF is shown in Figure 4.7.



(a) (b)

$$\begin{bmatrix} 0 & 0 & D_{\xi_1 N_A} & D_{\xi_1 H} & 0 \\ 0 & 0 & 0 & D_{\xi_2 H} & D_{\xi_2 N_B} \\ D_{\xi_1 N_A} & 0 & 0 & 0 & 0 \\ D_{\xi_1 H} & D_{\xi_2 H} & 0 & 0 & 0 \\ 0 & D_{\xi_2 N_B} & 0 & 0 & 0 \end{bmatrix}$$

(c)

Figure 4.7

The distance reaction coordinates (ξ_1 and ξ_2) defined in the molecular representation in Figure 4.7a, can be represented in a two layer graph shown in Figure 4.7b. The force computation in FEARCF is shown in the derivative matrix in Figure 4.7c. The atomic biasing force dependencies and net force on the hydrogen being exchanged can be mathematically defined and computed as the sum of two paths in the derivative matrix. This sum is given in Equation (4.1). Here i is an index over the set of reaction coordinates and j is an index over the edges that make up the path connecting the reaction coordinates to the atom.

$$F_H = \sum_i \frac{\partial W}{\partial \xi_i} \prod_j D_{ij} = \frac{\partial W}{\partial \xi_1} D_{\xi_1 H} + \frac{\partial W}{\partial \xi_2} D_{\xi_2 H} = \frac{\partial W}{\partial \xi_1}\frac{\partial \xi_1}{\partial H} + \frac{\partial W}{\partial \xi_2}\frac{\partial \xi_2}{\partial H} \tag{4.1}$$

Understanding the ability to construct any reaction coordinate set as a graph is key to understanding why an OOP approach is an optimal approach. In the conception of the FEARCF library, the graph that can be constructed for a reaction coordinate is directly related to how the reaction coordinate is structured in the OOP paradigm of the library. This is then also related to how FEARCF is able to calculate both the reaction coordinate value and the atomic biasing forces. Each node in a reaction coordinate graph represents an object defined in the library, while each edge represents the possible subroutines used to parse and calculate information between the two. These subroutines, in theory, need to be defined for any node type pairing. This is not actually the case, however, since there are strict conditions on which nodes can be used to define other nodes e.g. an angle node is only defined by two vector nodes, not point nodes. Any attempt to define a node with other node types that are not allowed will return an error. This then dramatically reduces the number of subroutines needing to be defined. And with

these subroutines defined, any arbitrary graph that can be constructed is then calculable by the FEARCF library. A procedural paradigm constructed FEARCF library would not be able to make this statement, because any unique graph would need to be defined explicitly within the library. With an OOP approach, the FEARCF library provides the tools for users to construct an infinite number of graphs and, therefore, an infinite number of possible reaction coordinate sets.

We have now laid out how OOP and graph theory are essential to the FEARCF software library by describing how the graph representation of reaction coordinates, along with OOP structures in the library, enable FEARCF to calculate any set of reaction coordinates that a user enters. In the next chapter, the FEARCF method will be presented in detail and the software library implementation of FEARCF detailed in full.

Chapter 5

FEARCF Library Development

In this chapter, the principle methodology of the FEARCF method will be given along with detailed equations showing how different reaction coordinates types are calculated, along with the partial derivatives needed to produce the atomic biasing forces. Then a detailed description of the FEARCF library is given along with explanations on how it is linked to the CHARMM and NWChem MD software packages.

5.1 The FEARCF Method

Presented below is the theoretical basis for the FEARCF free energy method.

To have complete understanding of a free energy volume (FEV) for a particular system, a simulation would ideally be able to sample all possible regions of reaction coordinate phase space ξ. This is not possible under normal simulation conditions, however, because some areas of free energy phase space require too high an energy for the simulation to visit by pure chance. We therefore need to introduce an additional potential $U(\xi)$, which we call a biasing potential, which when added to the actual system potential of the system itself $W(\xi)$, will produce a simulation that is able to sample all regions of free energy phase space. If we remember the Boltzmann relation, we can realise that the sum of $U(\xi)$ and $W(\xi)$ must be a constant in order for the probability of all values of ξ to be the same. For simplicity sake, we can choose this constant to be zero. We are left with the relation given in Equation (5.1).

$$U(\xi) + W(\xi) = 0 \tag{5.1}$$

This means that in the ideal case we have $U(\xi) = -W(\xi)$. The problem we now face is that we do not know the system potential $W(\xi)$, at least not in terms of the reaction coordinates. We therefore cannot even begin because

we don't know what to use for $U(\xi)$. We can realise however, that while we don't know $W(\xi)$ precisely, we can get an initial estimate $\widetilde{W}_1(\xi)$ for it. We can do this by simply running a simulation of our system with $U(\xi)$ initially set to zero i.e. $U_1(\xi) = 0$. With our first simulation done, we then have an estimate for the simulation probability density $\widetilde{P}_1(\xi)$. We can then calculate $\widetilde{W}_1(\xi)$ using the inverse Boltzmann relation given in Equation (5.2).

$$\widetilde{W}_1(\xi) = -k_B T \ln \widetilde{P}_1(\xi) \tag{5.2}$$

A quick reminder that k_B is the Boltzmann constant and T is the temperature.

Now if we run a second simulation of our system, we can then let $U_2(\xi) = -\widetilde{W}_1(\xi)$. After this second simulation, we now have another estimate of the probability density. But because the simulation was run using a biasing potential, it is not in fact an estimate of the probability density of our system $\widetilde{P}_2(\xi)$, but rather an estimate of the probability density of the biased system $\widetilde{P}'_2(\xi)$. Thankfully, because we know the biasing potential that caused $\widetilde{P}'_2(\xi)$, we can actually calculate $\widetilde{P}_2(\xi)$ using the expression in Equation (5.3), where C is a normalisation constant.

$$\widetilde{P}_2(\xi) = C\widetilde{P}'_2(\xi) \exp \frac{U_2(\xi)}{k_B T} \tag{5.3}$$

We then can calculate a second estimate for the system potential, $\widetilde{W}_2(\xi)$, by using $\widetilde{P}_2(\xi)$ and the same relationship given in Equation (5.2).

We repeat this process until we pass a convergence criteria. This criteria is that the ratio of sampling between least sampled to most sampled regions being at least 1:50, after which we assume that after n iterations $U_n(\xi) \approx -W(\xi)$ and we therefore have a good estimate for $W(\xi)$ in $\widetilde{W}_n(\xi)$. This in turn means that all of reaction coordinate phase space is being equally sampled, and we can then obtain a good estimate for the desired FEV [148].

The numerical nature of the free energy used in FEARCF has been described before by Naidoo [148]. The fact that $\widetilde{W}(\xi)$ is a numerically calculated potential is what distinguished it from previously discussed free energy methods.

5.2 Force Equations and Partial Derivatives

The manner in which the biasing potential $W(\xi)$ is used to calculate individual biasing forces on each of the atoms used to define ξ, is through a simple application of the properties of conserved potentials. Specifically, we reference the relation that states that the force $\vec{F}$ is the negative gradient of the scalar potential U i.e. $\vec{F} = -\nabla U$. The gradient is presented in terms of cartesian coordinates $\vec{x}$, however, in FEARCF the potential is defined in

terms of ξ. To resolve this discrepancy, we consider the chain rule which is given in Equation (5.4).

$$\nabla U = \frac{dU}{d\vec{x}} = \frac{\partial U}{\partial \xi}\frac{\partial \xi}{\partial \vec{x}} \tag{5.4}$$

The first term on the right hand side of Equation (5.4) is the gradient of the potential, which for numerical potentials can be numerically approximated using interpolation schemes such as cubic splining. The second term is the partial derivative of the ξ in terms of the cartesian coordinates of the atoms used to define ξ. In the case of multiple dimensions, this term is split into several partial derivatives which have known analytical solutions and are computable [149] [150].

To find the force from the free energy for a simple distance reaction coordinate r between two points a and b is relatively straight forward [149]. It is simply the partial derivative of the potential with respect to the distance, in the direction of the defined distance which is given in Equation (5.5).

$$\vec{F}_r = -\frac{\partial V}{\partial r}\hat{r} \tag{5.5}$$

Then, for the complete partial derivatives with respect to point a and b, they are simply the inverse of each other, to ensure the conservation of momentum. This is shown in Equation (5.6).

$$\vec{F}_a = \vec{F}_r \tag{5.6}$$

$$\vec{F}_b = -\vec{F}_a \tag{5.7}$$

For the force from the free energy for a direction reaction coordinate $\vec{r}$ between points a and b, the equation has the same first term as in Equation (5.5) but now has a second term for a partial derivative with respect to the unit vector $\hat{r}$. This partial derivative is in turn dotted with the 'anti-projector' matrix derived from the outer product of the unit vector from itself. This ensures that this second force term is purely in a direction perpendicular to the unit vector. The result is given in Equation (5.8).

$$\vec{F}_{\vec{r}} = \frac{\partial V}{\partial r}\hat{r} - \frac{1}{r}\frac{\partial V}{\partial \hat{r}}\left(\mathbb{1} - \hat{r}\otimes\hat{r}\right) \tag{5.8}$$

The force terms for point a and b are simply the same as given in Equation (5.6), except with $\vec{F}_{\vec{r}}$ instead of $\vec{F}_r$.

The force from the free energy for a bending angle θ defined between three points (can be atoms or centre of masses) i,j, and k [149] is given below in

Equation (5.9). The partial derivative of θ with respect to each point is given below in Equations (5.10), (5.11), and (5.12).

$$\vec{F}_i = -\frac{\partial V}{\partial \theta}\frac{\partial \theta}{\partial \vec{r}_i}, \vec{F}_j = -\frac{\partial V}{\partial \theta}\frac{\partial \theta}{\partial \vec{r}_j}, \vec{F}_k = -\frac{\partial V}{\partial \theta}\frac{\partial \theta}{\partial \vec{r}_k} \tag{5.9}$$

$$\frac{\partial \theta}{\partial \vec{r}_i} = -\frac{1}{r_{ij}}\left(\hat{r}_{ij} \times \left(\hat{r}_{kj} \times \hat{r}_{ij}\right)\right) \tag{5.10}$$

$$\frac{\partial \theta}{\partial \vec{r}_k} = -\frac{1}{r_{kj}}\left(\hat{r}_{kj} \times \left(\hat{r}_{kj} \times \hat{r}_{ij}\right)\right) \tag{5.11}$$

$$\frac{\partial \theta}{\partial \vec{r}_j} = -\frac{\partial \theta}{\partial \vec{r}_i} - \frac{\partial \theta}{\partial \vec{r}_k} \tag{5.12}$$

In the above equations, $\hat{r}_{ij}$ is the unit vector of the connecting vector from point j to point i, and equivalently the same is true for $\hat{r}_{kj}$.

For the force from the free energy for a dihedral angle ϕ defined between four points i, j, k and l [150], we firstly consider vectors connecting these points given in Equation (5.13).

$$\begin{aligned} \vec{r}_{ij} &= \vec{r}_i - \vec{r}_j \\ \vec{r}_{kj} &= \vec{r}_k - \vec{r}_j \\ \vec{r}_{lk} &= \vec{l}_i - \vec{k}_j \end{aligned} \tag{5.13}$$

We then consider the vectors $\vec{m}$ and $\vec{n}$, which are the normal to the planes that we can define in Equation (5.14) with our previous vectors in Equation (5.13).

$$\begin{aligned} \vec{m} &= \vec{r}_{ij} \times \vec{r}_{kj} \\ \vec{n} &= \vec{r}_{lk} \times \vec{r}_{kj} \end{aligned} \tag{5.14}$$

Now we give the force on each point given from the partial derivative of the biasing potential in Equation (5.15).

$$\begin{aligned} \vec{F}_i &= -\frac{\partial V}{\partial \phi} r_{kj} \frac{\vec{m}}{|m|^2} \\ \vec{F}_l &= -\frac{\partial V}{\partial \phi} r_{kj} \frac{\vec{n}}{|n|^2} \\ \vec{F}_j &= -\vec{F}_i + \left(\frac{\vec{r}_{kj} \cdot \vec{r}_{ij}}{r_{kj}^2}\right)\vec{F}_i + \left(\frac{\vec{r}_{kj} \cdot \vec{r}_{lk}}{r_{kj}^2}\right)\vec{F}_l \\ \vec{F}_k &= -\vec{F}_l - \left(\frac{\vec{r}_{kj} \cdot \vec{r}_{ij}}{r_{kj}^2}\right)\vec{F}_i - \left(\frac{\vec{r}_{kj} \cdot \vec{r}_{lk}}{r_{kj}^2}\right)\vec{F}_l \end{aligned} \tag{5.15}$$

For a pucker reaction of several pucker angles, there are several ways in which to define the angles themselves such as those defined by Cremer and Pople [151]. We have chosen to utilise the triangular tessellation definition, originally described by Pickett and Strauss for six membered rings [152], and then later generalised by Hill and Reilly [153]. This is due to its geometric nature being more suited to the graph structure of the library. Each angle θ_i is defined as $\frac{\pi}{2}$ minus the angle between the normal of the reference plane $\vec{n}$, and a vector $\vec{q}_i$ in the i^{th} side plane that is also perpendicular to the line of intersection $\vec{a}_i$ between the reference plane and the side plane. The equation for θ_i is given in Equation (5.16) while the definition of all the other vectors are given from Equation (5.17).

$$\theta_i = \frac{\pi}{2} - \arccos \frac{\vec{q}_i \cdot \vec{n}}{|q_i||n|} \tag{5.16}$$

$$\begin{aligned}
\vec{q}_i &= \vec{a}_i \times \vec{p}_i \\
\vec{p}_i &= \vec{r}_1 \times \vec{r}_2 \\
\vec{r}_1 &= \vec{x}_{side_2} - \vec{x}_{side_1} \\
\vec{r}_2 &= \vec{x}_{side_3} - \vec{x}_{side_2} \\
\vec{a}_i &= \vec{x}_{ref_2} - \vec{x}_{ref_1} \\
\vec{n} &= \vec{a}_i \times \vec{a}_j \\
\vec{a}_j &= \vec{x}_{ref_3} - \vec{x}_{ref_2}
\end{aligned} \tag{5.17}$$

In Equation (5.17), $\vec{x}_{side_j}$ is the position vector of the j^{th} atom that make up the i^{th} side plane, while $\vec{x}_{ref}$ is the position vector of the j^{th} atom that makes up the central reference plane. In some cases, these may be the same atom.

The force in terms of $\vec{n}$ is given in Equation (5.19) and the force in terms of vector $\vec{q}_i$ is given in equation (5.18).

$$\vec{F}_{\vec{q}_i} = \frac{\partial V}{\partial \theta_i} \frac{1}{\cos\theta_i} \frac{1}{|\vec{q}_i|} \left(\mathbb{1} - \hat{q}_i \otimes \hat{q}_i\right) \hat{n} \tag{5.18}$$

$$\vec{F}_{\vec{n}} = \frac{\partial V}{\partial \theta_i} \frac{1}{\cos\theta_i} \frac{1}{|\vec{n}|} \left(\mathbb{1} - \hat{n} \otimes \hat{n}\right) \hat{q}_i \tag{5.19}$$

Next we need to consider the the forces in terms of the vectors used to define $\vec{q}_i$, namely $\vec{p}_i$ and $\vec{a}_i$ which are given in Equations (5.20) and (5.21) respectively.

$$\vec{F}_{\vec{p}_i} = -\vec{a}_i \times \vec{F}_{\vec{q}_i} \tag{5.20}$$

$$\vec{F}_{\vec{a}_i} = \vec{p}_i \times \vec{F}_{\vec{q}_i} \tag{5.21}$$

We then consider the resultant force terms from $\vec{F}_{\vec{n}}$ for $\vec{a}_1$ and $\vec{a}_2$ which are given in Equation (5.22) and Equation (5.23).

$$\vec{F}_{\vec{a}_1} = -\vec{a}_2 \times \vec{F}_{\vec{n}} \tag{5.22}$$

$$\vec{F}_{\vec{a}_2} = \vec{a}_1 \times \vec{F}_{\vec{n}} \tag{5.23}$$

Considering next the resulting force terms for the side plain normals $\vec{p}_i$.

$$\vec{F}_{\vec{r}_{i_1}} = \vec{r}_{i_2} \times \vec{F}_{\vec{p}_i} \tag{5.24}$$

$$\vec{F}_{\vec{r}_{i_2}} = -\vec{r}_{i_1} \times \vec{F}_{\vec{p}_i} \tag{5.25}$$

The atoms that define the $\vec{a}$ and $\vec{r}$ are then given their forces in the same manner as Equation (5.6).

The above equations are all for variables chosen as reaction coordinates that define the free energy ,however, when a variable is simply used to define another variable or reaction coordinate, the force from the chosen reaction coordinates will need to be transferred and transformed down through each variable until it reaches the original atoms nodes which are the ultimate basis for all variable definitions.

For a distance r, defined between two points a and b, distributing the forces are simple

$$\begin{aligned} \vec{F}_a &= \vec{F}_r \\ \vec{F}_b &= -\vec{F}_a \end{aligned} \tag{5.26}$$

An identical distribution happens for when a direction $\vec{r}$ is defined between two points a and b

$$\begin{aligned} \vec{F}_a &= \vec{F}_{\vec{r}} \\ \vec{F}_b &= -\vec{F}_a \end{aligned} \tag{5.27}$$

When a direction $\vec{r}$ is defined as the cross product of two vectors (which can also be the normal vector of a plane) $\vec{a}$ and $\vec{b}$ i.e. $\vec{r} = \vec{a} \times \vec{b}$, then forces are distributed in the manner given in Equation (5.28)

$$\begin{aligned} \vec{F}_{\vec{a}} &= \vec{b} \times \vec{F}_{\vec{r}} \\ \vec{F}_{\vec{b}} &= -\vec{a} \times \vec{F}_{\vec{r}} \end{aligned} \tag{5.28}$$

If a defined point A is a centre of mass with total mass M, then the forces are distributed to each of the atoms a_i used to define it with suitable weighting according to their respective mass m_i. This is given in Equation (5.29).

$$\vec{F}_{a_i} = \frac{m_i}{M} \vec{F}_A \tag{5.29}$$

Now that all the equations needed to perform all calculations in FEARCF are given, a brief section outlining WHAM and its use in FEARCF for combining the results of parallel MD simulations is given.

5.3 WHAM

The parallelisation of the FEARCF method requires the probability distribution data from multiple simulations, each potentially with a different number of steps and different areas of reaction coordinate space sampled, to be combined in order to generate a new biasing potential. In order to do this, we utilise the weighted histogram analysis method (WHAM) [17] that allows for relative weighting of different simulation data. It involves solving the two equation given in Equation (5.30) and Equation (5.31).

$$p_k = \frac{\sum_i n_{i,k}}{\sum_i N_i f_i c_{i,k}} \tag{5.30}$$

$$f_i = \frac{1}{\sum_k e^{-\beta V_i(\omega_k)} p_k} \tag{5.31}$$

The above equations reference each other, therefore, they need to be solved iteratively until self consistency is achieved. In FEARCF, WHAM takes the output reaction coordinate trajectory files for each simulation, produces complete and weighted probability for each iteration. WHAM then uses the Boltzmann relation given in Equation (5.2) to produce a new associated biasing potential for the next iteration to utilise.

We now move on to describing in detail the implementation of the FEARCF method in the form of a software library.

5.4 Designing the FEARCF library

In FEARCF, the subject we are most interested in are the reaction coordinates. We cannot think of reaction coordinates in isolation however. In order to mathematically define our reaction coordinates, we need to in turn define other geometric quantities. The most important of which are the atoms of our system and their cartesian coordinates. From just these coordinates, we are able to define any number of reaction coordinates that we might wish. These reaction coordinates definitions can be visualised as graphs. Once we have the definitions for a set of reaction coordinates, and their relation to the individual atoms of a molecular system, we can then consider what else is required for FEARCF. In FEARCF, we obtain the derivatives of our free energy volume as a result of an interpolation function. These derivatives are, however, derivatives in terms of our reaction coordinates. The purpose of calculating these derivatives of the free energy, is to find the resultant biasing forces on the atoms in our system. The derivatives that we have, however, are not sufficient for this purpose. They still need to be transformed so that they become derivatives in terms of the atomic coordinates. The method in which this can be done is demonstrated by considering a simple chain rule example given in Equation (5.32).

$$\frac{\partial U}{\partial \vec{x}_i} = \frac{\partial U}{\partial \xi}\frac{\partial \xi}{\partial \vec{x}_i} \tag{5.32}$$

$\vec{x}_i$ are the atomic coordinates for atom i and ξ is a reaction coordinate defined using $\vec{x}_i$. The first term on the left hand side of Equation (5.32) is provided by the interpolation function of FEARCF. The second term is not. To calculate this second term, we need to find the derivative of ξ in terms of $\vec{x}_i$. This is usually easy enough since we already know the definition of ξ in terms of $\vec{x}_i$ and can therefore analytically solve this derivative. The difficulty comes in the fact that ξ may not being explicitly defined in terms of $\vec{x}_i$, but rather other variables. This difficulty can be overcome if we take the same chain rule approach as in Equation (5.32), but with the term $\frac{\partial \xi}{\partial \vec{x}_i}$. Say that ξ is defined by a variable b, which is in turn defined by $\vec{x}_i$. We can then express a chain rule expansion in Equation (5.33).

$$\frac{\partial \xi}{\partial \vec{x}_i} = \frac{\partial \xi}{\partial b}\frac{\partial b}{\partial \vec{x}_i} \tag{5.33}$$

Each term on the right hand side of the equation of Equation (5.33) is relatively simple to find since (almost) all the functional forms are known and differentiable. Therefore they can be defined beforehand, incorporated as subroutines within the FEARCF class, and calculated using the calculated values of $\vec{x}_i$, b, and ξ.

5.4.1 Graph Theory in FEARCF

The graphs constructed in FEARCF detail the relationship of the reaction coordinates to the atomic coordinates from which they are ultimately calculated. The vertices represent several types of quantities (atom coordinates, vectors, planes etc.) while the edges represent the mathematical relations between these quantities. In this way, our constructed reactions coordinates can be represented as coloured graphs.

At every integration step of a FEARCF simulation, the ξ values are calculated. To do this ,each RC node retrieves information from the immediate nodes that define it. If these nodes do not represent atoms, they will in turn need to retrieve information from the nodes that define them. This creates a chain of information retrieval that ends in the nodes used to define the atoms of the system. These contain the atomic coordinates retrieved from the MD package that the FEARCF library is compiled with. With these coordinates, all other nodes are able to perform the required calculations to ultimately find the values of all the nodes and the chosen RCs.

Once all RC values are known, the cubic or B-spline interpolation subroutine is used to calculate the derivatives of the FEV at these RC values in order to obtain the biasing force for the current FEARCF iteration. The derivatives are in terms of the RCs but can be transformed to be in terms of the atoms via the chain rule. The derivative received from the interpolation subroutine is sent to each immediate node of every RC node. These nodes then calculate the partial derivative term required to transform the received derivative to be in terms of that node variable. These nodes then send this derivative to their immediate nodes. This is repeated in a chain like manner until the nodes that represent the atoms have accumulated all the derivatives needed to represent the biasing force on each atom. The atomic biasing forces are then added to the forces retrieved by the MD package interfaced with the FEARCF library, so that the atomic velocities and positions are updated using both the system potential and the FEARCF biasing potential.

We now consider how these graphs are translated into the actual OOP structure of the FEARCF library. The nodes of a graph make ideal candidates for objects in a OOP paradigm since they act as containers for data about the variables the nodes represent (variable values, coordinates, mass, etc.) and objects also contain methods for accessing and modifying the data in the nodes. In addition, the role of the edges of the graph can be fulfilled by methods within the FEARCF class that contain the mathematical relations between different types of nodes, for both calculating the actual value of the nodes, as well as calculating the partial derivative of one node in relation to another. An important point to note is that these mathematical relations are general and can therefore be used in calculations for any graph that can be constructed with these types of nodes.

This relation between the reaction coordinate graphs and the OOP structure of the FEARCF library shows that the scope of the graph model extends beyond that of a visualisation tool, and is actually key to understanding the advantage of using an OOP approach for this particular free energy method.

5.4.2 Library Structure

The FEARCF library is written in Fortran 90 and is composed primarily of several interdependent modules. The most important module is the FEARCF class module that defines the FEARCF class, which contains all the information about the defined variables and atom coordinates. It also defines three important subroutines for each type of variable. The first subroutine type define is called when each variable is initially defined and allocates the appropriate array sizes and memory. This subroutine type also stores the list of 'children' for each specific variable, which are the other variables used to define this variable. This is to ensure that they can be easily referenced. The second subroutine type calc calculates the value of the variable itself at every time step, while the third subroutine type calc_f calculates the resultant force from the product of partial derivatives. A variable's subroutines may call other subroutines (or even itself in a recursive manner) related to other variables, depending on what variables were used to define the variable in question. The overall structure of the FEARCF class is given in Figure 5.1 and distinguishes the methods into public and private ones, with the private methods being the ones which represent communication between nodes within any defined graph.

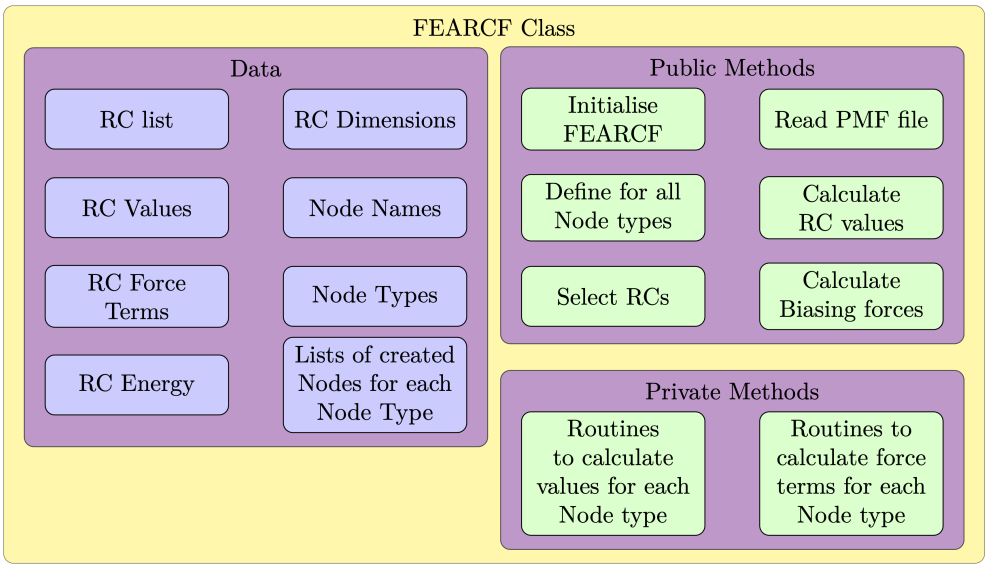


Figure 5.1: Class structure of FEARCF type defined in class_fearcf.

The FEARCF prepare module is used to process information received from

the chosen MD software, as well as prepare the calculated force values to be returned to the MD software. For each supported software package, four subroutines are needed to be defined. One to process the atomic coordinates, another to process the forces already calculated by the MD software, another to extract the atomic masses, and a final one to transform the forces calculated by FEARCF into the appropriate format to return to the main force calculations in the MD software.

The FEARCF defs module is where all the variable (node) types are defined. This includes defining all the value storage variables, as well as two subroutines for each variable type. The first subroutine set, is used to store the values (including the forces) once they have been calculated. The second subroutine get is used to retrieve values or forces, usually in order to calculate other variables. The 'set' and 'get' subroutines for each variable type and then placed within a Fortran interface, allowing for easier subroutine calls with the FEARCF class. An example of how a type is defined in Fortran is given below as well as how a method interface is defined in Fortran.

```
type angle
        real*8 theta ! . Value of angle
        real*8 fx,fy,fz,fmag ! . Force components and magnitude
end type angle

interface get_
        module procedure get_angle
end interface
```

In addition, an illustration of how the atom (or point) type is structured is given in Figure 5.2 while an illustration of how other reaction coordinates types are defined is given in Figure 5.3.

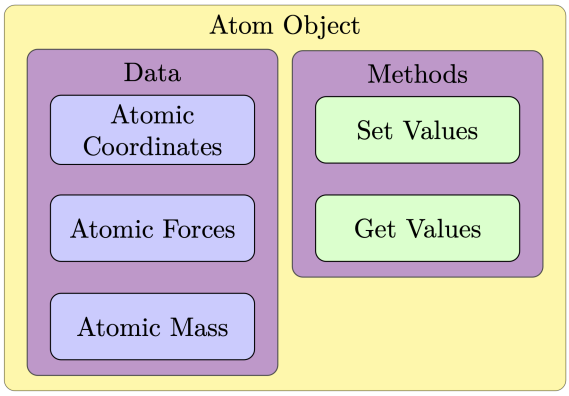


Figure 5.2: Object structure of atom type within the FEARCF library.

The FEARCF interface module is the main control code within FEARCF, keeping track of the number of iterations and then calling the appropriate

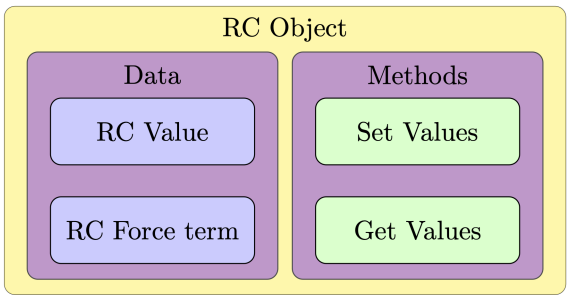


Figure 5.3: Object structure of general RC types within the FEARCF library.

.pmf and trajectory files. This module reads the file fearcf_input.dat, which is where the user defines all the required variables and selects the reaction coordinates. The interface module then makes all the required subroutine calls within the other modules for each possible MD software.

The module ndspline contains all the code needed for the interpolation of the free energy, including code for both normal cubic spline interpolation and cubic B-spline. Which interpolation method is used by FEARCF can be selected by the user in fearcf_input.dat

The module math_helper contains subroutines for commonly required maths objects, such as cross products, which can then be called by other modules.

An illustration of how the above modules are interconnected, along with the requisite input files i given in Figure 5.4.

interface is the first module that is called since it contains the two subroutines that are called by the MD software. The first is setup_fearcf and is called before the main MD loop has begun. Its only input is the sim variable which is of the form 'i-j' where i is the number of the current FEARCF iteration and j is the index number identifying which parallel simulation is now being started. sim is used so that FEARCF knows which *.pmf* file to read and which trajectory output file to write to. It firstly calls the initialise subroutine from the class module which generates an object of the FEARCF class type, and initialises all the required arrays. Next ,setup_fearcf reads the FEARCF input file called fearcf_input.dat, using the io module to assign a unique Fortran unit number.

An example of a simple fearcf_input.dat file defining a distance reaction coordinate between two atoms, along with selecting the B-spline method, is given below.

```
point atom1 1
point atom2 2
dist r1 atom1 atom2
set r1
```

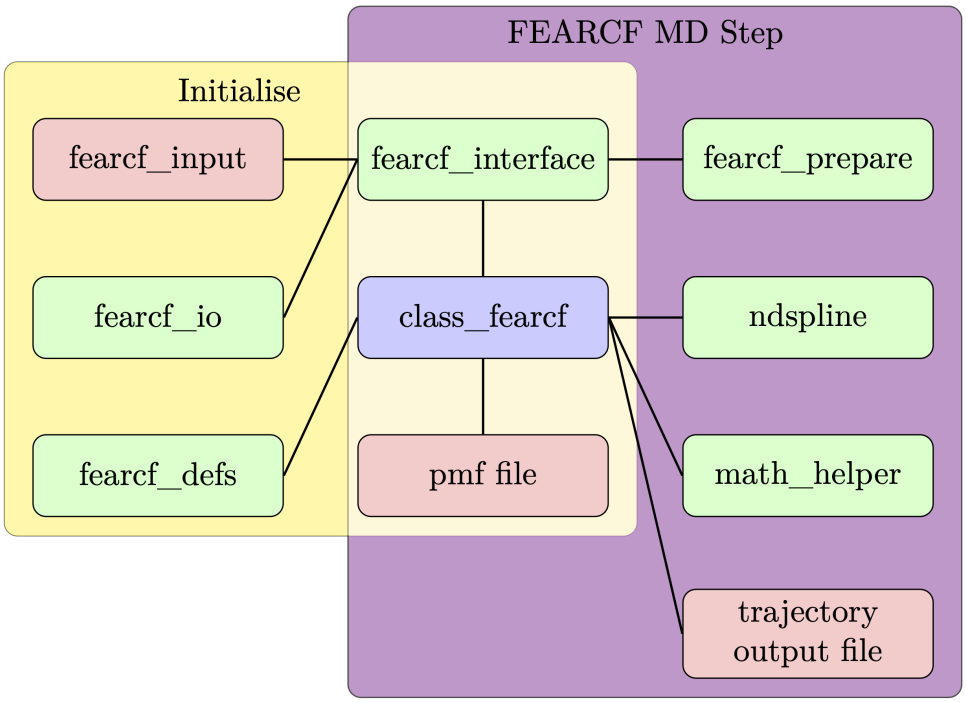


Figure 5.4: Diagram showing connections between the different modules making up the FEARCF library as well as the required input files. The central FEARCF class module is coloured blue, other sub-modules are coloured green, and input files are coloured red.

```
spline 0
```

The first two lines define the atoms by firstly giving their variable type as point, then assigning them a unique name, either atom1 or atom2, and then giving their atom number within the MD package coordinate set. The distance reaction coordinate is then defined by, again, first stating that the type is dist, assigning it a unique name r1, and then listing the points used to define it. The set command, followed by the desired reaction coordinate name, is then used to select that reaction coordinate for use in defining the free energy that will be then be calculated. Lastly, the spline command sets the spline method to be used, 0 for cubic B-spline or 1 for regular cubic spline.

As setup_fearcf reads each line of fearcf_input.dat, it calls the appropriate define subroutine from the FEARCF class module. The define subroutines all create objects of the classes defined in the defs module. When it reaches the set commands it calls the fearcf_set subroutine (also from class and not to be confused with 'set' subroutines within defs) and records which reaction coordinates the free energy volume will be in terms of. Once all the set commands have be read, it then calculates how many dimensions the free energy will be defined as.

setup_fearcf then calls the read_map subroutine from class that reads and records the values from the *.pmf* input file, again using the io module to assign a unique Fortran unit number. Finally, setup_fearcf calls set_traj from the class module which uses the io module to give the trajectory output file a unique Fortran unit number, as well as create the output file name 'traj-i-j' based on the sim variable.

During the main MD loop controlled by the MD software package the library is compiled in, the FEARCF method is started by a call to the fearcf_calc subroutine in the interface module. There is a unique fearcf_calc for each possible MD software and the reason for this will be demonstrated below, by comparing the implementation of the FEARCF library in NWChem and CHARMM.

All the fearcf_calc subroutines have, as input variables, a scalar for the total atom number, and arrays for atomic coordinates, atomic forces, and atomic masses. The fearcf_calc subroutine then calls three subroutines from the prepare module, one for the masses, one for the coordinates, and one for the forces. This is to get the mass, coordinate, and force data all into the same form. There are three unique subroutines for each possible MD engine. The reason for this is that the input arrays cannot assumed to be of a certain shape and therefore has to be set ahead of time depending on whatever array shape the MD software stores them in. If the FEARCF library was a dynamic library, this issue might be avoidable since it would only be linked at run time and therefore the array shapes in question could be retrieved. As a static library, however, it needs to have every possibility defined be-

fore being compiled, since this is before the arrays even exist. For example, NWChem [131] has its coordinate arrays as either $3 \times N$ or $N \times 3$ arrays (depending which MD module is being used), where N is the number of atoms. In CHARMM [130], the coordinates are in three individual N shape arrays for the x, y and z components, the same being true for the forces. In addition to this, CHARMM stores its coordinate arrays in Å and its forces in kcal/mol/Å, while NWChem stores all its units in Hartree atomic units e.g. bohr a_0 for length and hartree E_h for energy. The NWChem arrays therefore need to be multiplied by a conversion factor, since the FEARCF code is written in terms of the same units as CHARMM. The mass arrays do not need to be converted because they are only used to calculate centre of mass, in which the mass units cancel out.

Next, fearcf_calc calls the calc_crds subroutine from the class module, which then proceeds to calculate the values of the chosen reaction coordinates by calling the associated calc subroutine. In order to do this, it will in turn call other calc subroutines for other classes and then retrieve the calculated values. The end point of these calc calls should be the subroutine calc_point in which the atomic coordinates for that particular atom are found and stored in the appropriate point object. Once this is done and all variables have values, calc_crds then calls the spline subroutine from the ndspline module. spline calculates the derivative of the current free energy in terms of each reaction coordinate, using the interpolation method the user selected. These derivatives are then multiplied by a negative sign to transform them into the desired biasing potential. Lastly, calc_crds writes the reaction coordinate values, as well as the free energy value received from spline, to the output trajectory file.

The subroutine calc_frcs is then called by fearcf_calc from the class module. This subroutine calls the calc_f subroutine for each chosen reaction coordinate class. This subroutine takes derivative in terms of the reaction coordinate received from the spline subroutine and then calculates the partial derivative of the reaction coordinate in terms of its children variables. These two partial derivative terms are then combined and then stored in the associated objects for the children variables. Then, the calc_f subroutine is called for the same class as the children variables. This is repeated until the point_calc_f is called for each atom, at which point all the partial derivatives should have been combined and resulting in the correct derivative of the biasing potential in terms of each atom. This derivative is then added to the force value derived from the system potential that was retrieved from the MD software by the prepare module.

Lastly, fearcf_calc calls the apply_forces subroutine from the prepare module. This then takes the array that FEARCF had stored, reverses any unit conversations that was done, and then uses these values to repopulate the force array imported from the MD software.

5.4.3 Implementation in MD Software

A key feature of libraries is that they are easily imported into other pieces of software, therefore, they need to be well designed so that the required changes for importing software is as minimal as possible. To demonstrate how the FEARCF library fulfils this requirement, I will explain how the library has been incorporated firstly into the CHARMM software package [130], and then the NWChem software package [131]. For simplicity's sake, I have given a general description of how the FEARCF library is integrated into any MD software in Figure 5.5. This figure shows how FEARCF is integrated within the MD integration loop, importing atomic coordinates, forces, and masses, and then exporting biasing forces resulting from biasing potential.

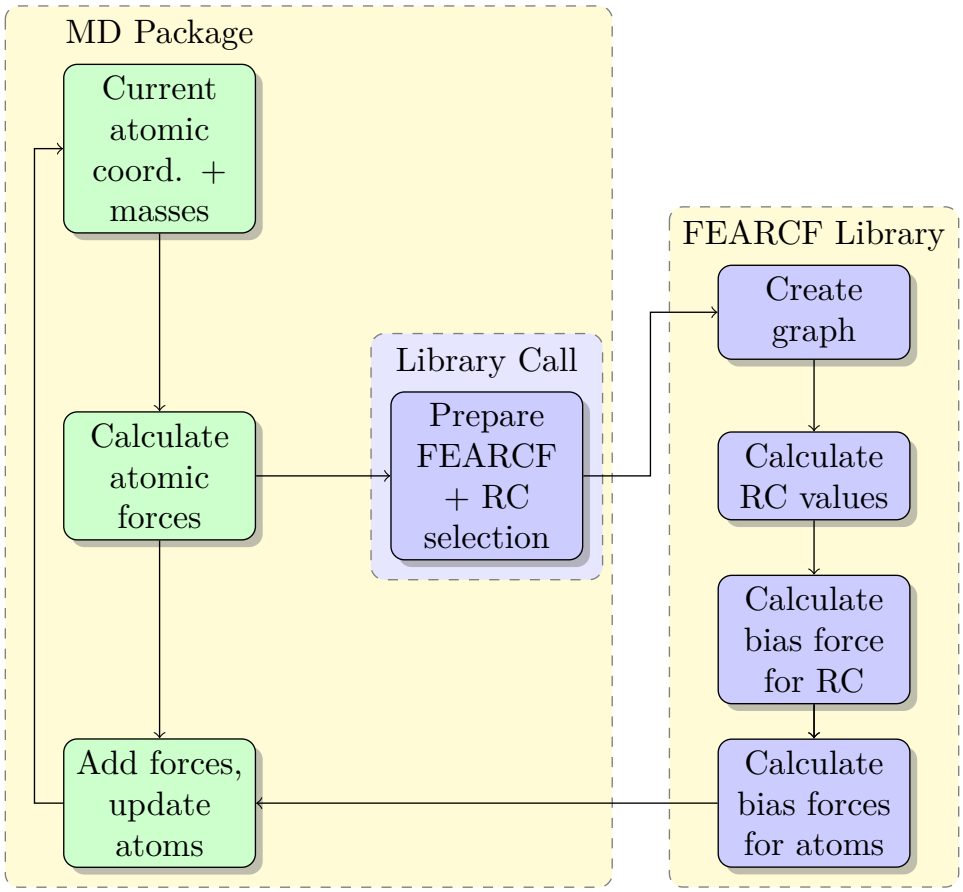


Figure 5.5: Illustration of the required connections between the FEARCF library and an associated MD package

CHARMM happens to have a set of subroutines which are expressly designed for users to modify and insert their own subroutine calls. These are contained within the usermod module that appears in the source file usersb.src. There are several subroutines within this module that are called at different points

within the main CHARMM program. For the FEARCF library, we have chosen to utilise two subroutines: usersb and usere. The subroutine usersb is able to be called by a user in a CHARMM input script by adding the *user* command anywhere in the script. In the regular CHARMM release, all this command does is output text confirming that usersb has been called and nothing else. We have chosen to use this subroutine as the one in which the setup_fearcf call is made. The reason for this is that it allows for the user to decide whether they wish to run their MD simulation as a FEARCF simulation or not, by choosing to include this *user* command. If the *user* command does not appear in an input script, FEARCF will not be run. The 'sim' input variable for setup_fearcf is retrieved as a shell environmental variable which is set by a job script used to run CHARMM.

When setup_fearcf runs, it sets the fearcf_is_initialised logical variable as true. This is important because in the second usermod subroutine that FEARCF uses usere, it is used as an if statement condition to decide whether the fearcf_calc subroutine should be called. The subroutine usere is normally called as part of the CHARMM energy evaluation and its regular function is to simply set the energy arrays as zero. For FEARCF it is used to access the atomic coordinates and masses which have been populated by CHARMM already, and then call the fearcf_calc subroutine. The atomic force arrays have not been populated yet, but they have been allocated shapes and at no point are they reset during the next MD step, so it is possible to use these arrays to store the biasing forces which are then later added to the forces calculated from the CHARMM force field.

NWChem unfortunately does not have this capacity. Therefore, the FEARCF library calls need to be inserted in several points within the NWChem program. The most difficult thing with NWChem is that it actually contains two separate MD integration routines or 'engines'. The first is referred to simply as 'NWMD' while the second is referred to as 'QMD'. Because of the separate engines, separate fearcf_calc subroutines are required to be defined due to the differing arrays shapes in each engine. In the 'NWMD' engine, the coordinate and force arrays are of the shape $N \times 3$, while in the 'QMD' these arrays are of the shape $3 \times N$. In addition, NWChem is specifically designed for parallelisation but the FEARCF library is currently not. CPU node ID flags are therefore used to ensure that only one CPU conducts the FEARCF calculation. If this was not the case, multiple biasing forces would then be added to the calculated atomic forces and convergence to a flat histogram distribution would be impossible.

The 'NWMD' engine is the more outdated MD engine of NWChem. It is still used for classical MD runs as well as QM/MM simulations, and therefore it cannot be ignored. It is composed of several subroutines including: md_input, md_main, and md_start. The FEARCF subroutine calls for setup_fearcf and fearcf_calc occur within md_main. Difficulty comes in the fact that NWChem

uses Global Arrays (GAs) [154], so subroutines are actually required to be written for FEARCF to retrieve values stored in GAs and deposit them in local arrays that FEARCF can manipulate. Another subroutine is also required to deposit the atomic forces back once the biasing forces have been added. In order to make the usage of FEARCF easier, a 'fearcf' input flag was also added the NWMD input block that is read from the NWChem input file. If the 'fearcf' command is included in the NWMD command block, then NWChem will set the associated 'fearcf' logical to `True`. This then leads to the setup_fearcf subroutine being called before the default equilibration sequence. In this case, the sim variable is extracted from the actual file name of the input file which should be of the form 'moleculename-i-j', where i and j are the appropriate integers for sim. This is because the NWChem input file is not a purely Fortran file like CHARMM, and therefore cannot retrieve shell environmental variables. The fearcf_calc subroutine is called just after the NWChem forces are calculated, during the main MD loop.

The 'QMD' engine is the more up-to-date MD engine and it actually provides a simpler implementation for the FEARCF library. The main subroutine for QMD is qmd_driver because that is the module that contains the actual MD loop. The retrieval of the GAs are actually handled by the qmd_driver itself, so no extra subroutines for FEARCF have to be written. Like NWMD, a 'fearcf' input flag was added to the QMD command block that features in the NWChem input script so the user can control whether they want to use FEARCF during a dynamics run or not. The setup_fearcf subroutine is called after all the regular initialisation steps have been conducted. Similarly to NWMD, the sim variable is extracted from the file name. The fearcf_calc subroutine call is inserted after the first force calculation, which occurs outside of the MD loop. fearcf_calc is called again after the NWChem force calculation that features within the main MD loop.

We have now described the FEARCF method, given all the equations need to perform all calculations, and described in detail how FEARCF has been incorporated into a software library. The implementation of the FEARCF library into both CHARMM and NWChem has also be detailed in full. The following three chapters will present results of free energy simulations to prove the success of this library implementation, as well as demonstrate its improvements over previous implementations of the FEARCF method.

Chapter 6

Application Case Study: Water Dimer

The chapter presents classical and quantum multidimensional free energy results for the water dimer system. This was only possible by the FEARCF library being implemented in both the CHARMM and NWChem MD software packages.

6.1 The Water Dimer

Water, while a simple molecule, exhibits a complex structure in both bulk liquid and solid states. A key reason for this is the intermolecular hydrogen bonds that form between water molecules. A single water molecule is able to form up to four hydrogen bonds at once, a large number for a molecule of such low molecular weight [31]. In ice, water forms a hexagonal crystal structure by forming inter-molecular bonds with four of its neighbours. Two H-bonds being formed with its oxygen as the acceptor, and two H-bonds being formed with each of the hydrogens acting as donors. These numerous hydrogen bonds are the reason for water's high boiling point and greater density in the liquid state when compared to the solid state. This is due to the hydrogen bonding network being retained in the liquid form, but being able to be packed denser than the hexagonal structure of the solid form due to the continual exchange of hydrogen bonds [25]. Previously, van Thiel *et al.* [155] described three potential configurations for a water dimer to form hydrogen bonds named: Chain (C), Cyclic (C_y) and Bifurcated (B).

The TIP3P model [156] has been used as a standard for modelling water in classical MD for many years and is the parameterisation basis for many atomic force fields. It is a classical model which treats the oxygen and hydrogen atoms as point charges. The model's simplicity is remarkably affective for modelling solute-solvent interactions with organic molecules. As a model of bulk water itself, however, it is severally limited [157]. Attempts have

been made to improve the TIP3P model, most notably the TIP4P [156] and TIP5P [158] models which add additional point charges in an attempt to more accurately model the non-classical electron charge distribution. These models have been shown to have their own limitations, and have therefore not replaced the TIP3P model [159]. In this work, I will present several TIP3P free energy volumes (FEV) that will give an insight into TIP3P's behaviour and limitations as a model for bulk water, which will then be contrasted with an *ab initio* water model.

6.2 Classical Results

Below are presented the free energy results of the classical TIP3P model.

6.2.1 Simulation Setup

All classical MD simulations are run in CHARMM 42b2 [130], using the CHARMM force field [54], run at 300K in periodic boxes of size 12Å each side, with non-bonded interactions cut-off at 13Å, the Velocity-Verlet integrator [7] with a Nose-Hoover NVT ensemble thermostat [63] with 1 heat bath, a tau value of 0.1ps and 10 sub-steps per time step. Each simulation is run with a 1fs time step, with no equilibrium steps, and a unique set of four seven-digit random seed numbers for the initial random assignment of velocities.

6.2.2 1D-FEV-Distance

First, let's consider one reaction coordinate, namely: the well-studied scalar distance between the oxygen atoms for the water dimer system which is given in Figure 6.1a. The graph that represents this reaction coordinate is then given in Figure 6.1b. The oxygen atoms are represented as points with which a distance r between the two waters is defined. This distance is the reaction coordinate. The value of node r is simply given by $r = \sqrt{\left(\vec{x}_{W_2(O)} - \vec{x}_{W_1(O)}\right)^2}$. The atomic biasing forces is computed through the addition of the derivative in terms of r and the unit vector $\hat{r}$. This is given in equation (6.1).

$$\begin{aligned} \vec{F}_r &= -\frac{\partial V}{\partial r}\hat{r} \\ \vec{F}_{W_1} &= \vec{F}_r \\ \vec{F}_{W_2} &= -\vec{F}_r \end{aligned} \tag{6.1}$$

The 1D classical water dimer FEARCF simulation is run for 50ps each iteration, with 128 simultaneous simulations in each iteration, each simulation run with a single core. Histograms for the reaction coordinate r comprised 41 bins over range of [0.5,10] Å.

The sampling along $\xi = r$ during the first FEARCF iteration, where the biasing potential = 0, is centered around 3 Å with the ratio of highest to lowest sampling being 19.3:1 (Figure 6.1c). After the 15th FEARCF iteration (Figure 6.1e), the sampling along r converges toward a flat histogram with a high-to-low sampling ratio of 2.5:1. The minimum free energy is found at r = 2.9 Å. Considering that the covalent bond length between the oxygen and the hydrogen in the TIP3P model is 0.9572 Å, the minimum well is presumed to be a combination hydrogen bond configurations with an average length 1.9428 Å.

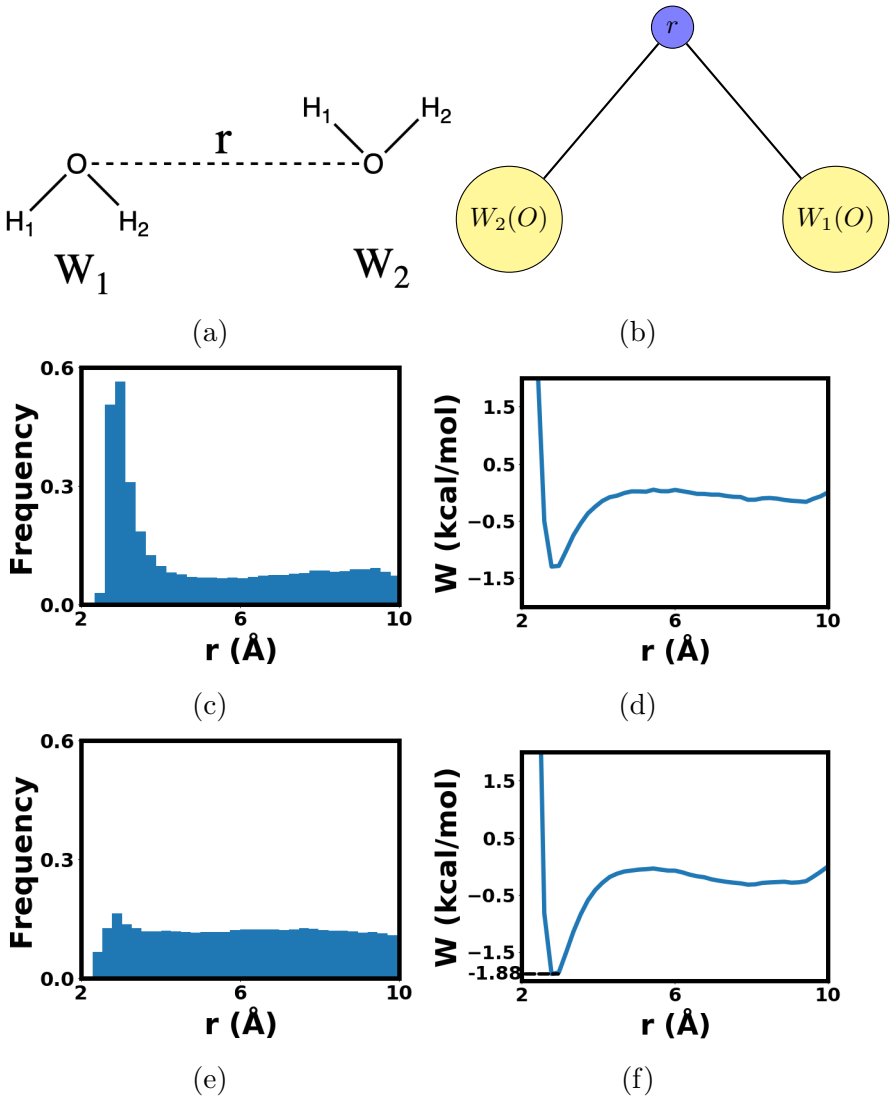


Figure 6.1: 1D classical water dimer FEARCF simulation a) 1D RC definition b) FEARCF RC graph representation c) Histogram showing sampling of 1st iteration d) FE surface of 1st iteration e) Histogram showing sampling of 15th iteration f) FE surface of 15th iteration.

The magnitude of the free energy well at -1.88 kcal/mol does agree with the

hydrogen bond strength being between that of van der Waals forces and covalent/ionic bonds [160]. However, this one dimensional FEV gives us no insight into the actual orientational preference of water dimer hydrogen bonds.

6.2.3 2D-FEV-Distance and Angle

Next, we show another common setup for investigating water dimer interactions, where two reaction coordinates (Figure 6.2a) include a distance (between the centre of mass of each water molecule) and an angle (between the two dipole moments of each water molecule). An illustration of the graphing construct for these reaction coordinates is given in Figure 6.2b. Since the distance is now between centres of masses, all of the atoms in the system are required to be represented as points. The oxygen atom and two hydrogens of each water is needed to define the centre of mass W_i. The first reaction coordinate is the scalar distance $\xi_1 = r$ between the centres of mass. The second reaction coordinate requires a definition of the dipole moment vector $\vec{b}_i$ for each water molecule. This is done by simply defining a vector that points from the oxygen atom to the centre of mass for each water molecule. While these are not the true dipole moment vectors, they are actually parallel to them. This then allows us to get the correct value for our second reaction coordinate, the angle ϕ.

To translate the data flow via edges shown in Figure 6.2b, $r = \sqrt{(\vec{x}_{W_2} - \vec{x}_{W_1})^2}$ and $\phi = \arccos\left(\vec{b}_1 \cdot \vec{b}_2 / \left|\vec{b}_1 \cdot \vec{b}_2\right|\right)$ are needed for the ξ nodes. In the case of the atomic biasing forces, a similar approach for r as in Equation (6.1) is used. However, for ϕ, the forces for the nodes are given in Equation (6.2).

$$
\begin{aligned}
\vec{F}_{W_1} &= -\frac{\partial V}{\partial \phi}\frac{\partial \phi}{\partial \vec{x}_{W_1}} \\
\vec{F}_{W_1(O)} &= -\frac{\partial V}{\partial \phi}\frac{\partial \phi}{\partial \vec{x}_{W_1(O)}} \\
\vec{F}_{W_2} &= -\frac{\partial V}{\partial \phi}\frac{\partial \phi}{\partial \vec{x}_{W_2}} \\
\vec{F}_{W_2}(O) &= -\frac{\partial V}{\partial \phi}\frac{\partial \phi}{\partial \vec{x}_{W_2(O)}} \\
\frac{\partial \phi}{\partial \vec{x}_{W_1}} &= -\frac{1}{|b_1|}\left(\hat{b}_1 \times \left(\hat{b}_2 \times \hat{b}_1\right)\right) \\
\frac{\partial \phi}{\partial \vec{x}_{W_2}} &= -\frac{1}{|b_2|}\left(\hat{b}_2 \times \left(\hat{b}_2 \times \hat{b}_1\right)\right) \\
\frac{\partial \phi}{\partial \vec{x}_{W_1(O)}} &= -\frac{\partial \phi}{\partial \vec{x}_{W_1}} \\
\frac{\partial \phi}{\partial \vec{x}_{W_2(O)}} &= -\frac{\partial \phi}{\partial \vec{x}_{W_2}}
\end{aligned}
\tag{6.2}
$$

For the atom nodes used to define the centre of masses, forces are simply weighted according to their atomic masses. The 2D classical water dimer FEARCF simulation is run for 50ps each iteration, with 128 simultaneous simulations in each iteration, and each simulation run with a single core. Histogram dimension for reaction coordinate r comprised 41 divisions over range of [0.5,10] Å, while the histogram dimension for reaction coordinate ϕ comprised 21 divisions, over a range of [-90,90] degrees. The sampling for both r and ϕ (Figure 6.2c) for the first FEARCF iteration, that has a zero biasing potential, has as high-to-low ratio for ($\xi_1 = r$;$\xi_2 = \phi$) of (8:1; 28:1), while after the convergence toward a flat histogram in 12 iterations (Figure 6.2e), the ratio is (1.14:1; 26:1).

While the same minimum energy distance is observed in Figure 6.2f as in the 1D case (Figure 6.1f), now the preference of the dipoles to orient themselves between 40° and 120° is made clear. A preference for orthogonal configurations hints at preferred hydrogen bonding configurations. Orthogonal dipole orientations more closely resemble linear chain compared with bifurcated hydrogen bonding. However, there is no single minima for the ϕ angle. This is consistent with an understanding that ϕ cannot distinguish between an orientation where the hydrogen bonding is donated from one water giving a bifurcated configuration, or when the waters are laying parallel alongside each other. As we have previously noted, dipole interactions are typically described by two degrees of freedom i.e. the separation of the centres and the angle between them. This is a sparse description and contains not much more information than just the one dimensional free energy $W(r)$.

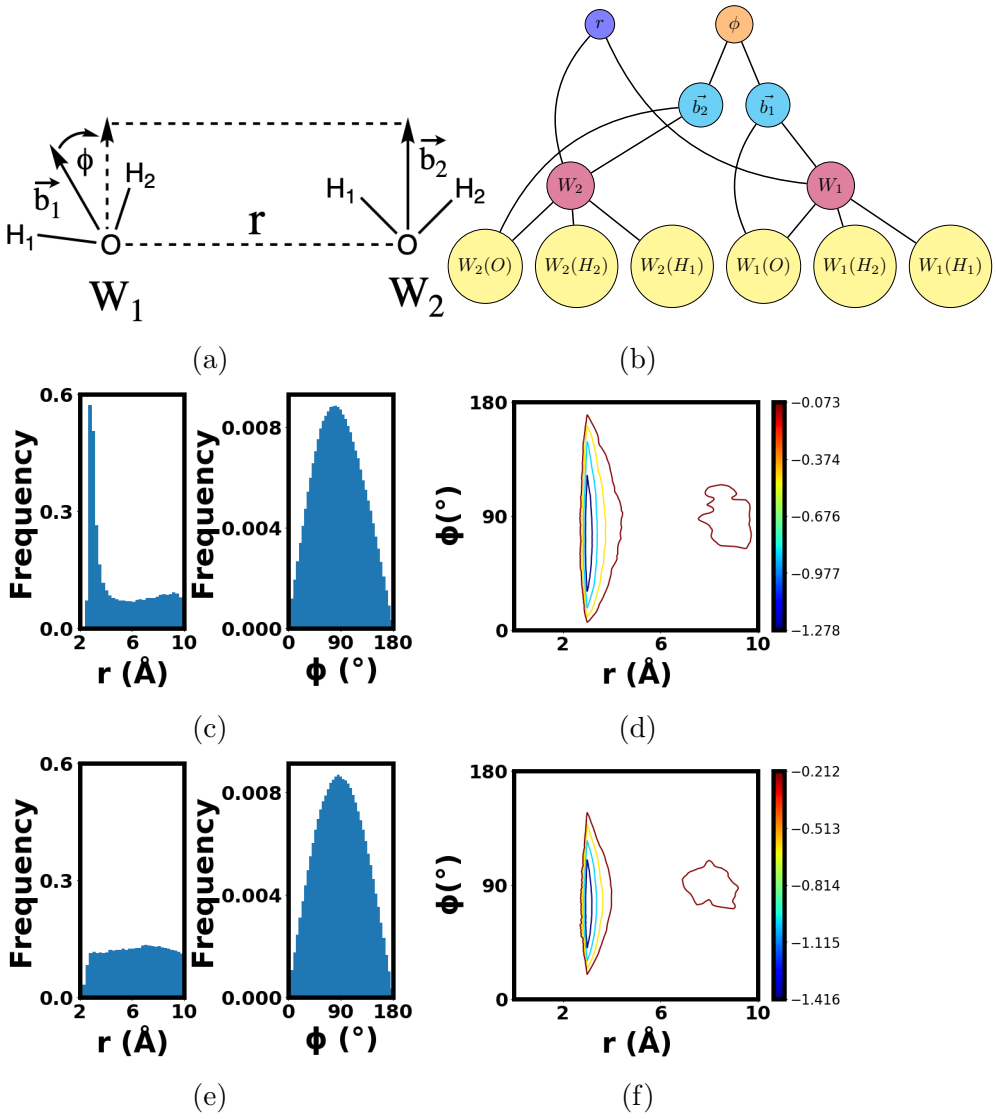


Figure 6.2: 2D classical water dimer FEARCF simulation a) 2D RC definitions b) FEARCF RC graph representation c) Histogram showing sampling of 1st iteration d) FE surface of 1st iteration with contours at 1/2 k_BT intervals e) Histogram showing sampling of 12th iteration f) FE surface of 12th iteration with contours at 1/2 k_BT intervals.

6.2.4 4D-FEV-Distance, Angles and Dihedral

Previous work showed that four reaction coordinates were a useful description for the water dimer system (reduced from a full description of six reaction coordinates), leading to the 4D $W(r, \theta_1, \theta_2, \phi)$ [18]. Here the reaction coordinates are the distance between the centres of masses (r), the molecular vector angles (θ_1 and θ_2), and their relative dihedral orientation (ϕ) as shown in Figure 6.3a. It is similar to the 2D system with the same definition for r, however, ϕ is now a dihedral angle. In addition, we have defined angles be-

tween the dipole moment vectors $\vec{b_i}$ and the vector version $\vec{r}$ of our previously defined distance r. A graph that best represents these reaction coordinates is illustrated in Figure 6.3b. Again, we need to define all the atoms in our system as points and then define the centre of masses m for each water molecule. With this alone, we can define our scalar distance reaction coordinate r as well as its vector $\vec{r}$ alternative. We then define the vectors parallel with the dipole moment vectors $\vec{b_i}$, which we then use to define the dihedral angle ϕ between them. This is possible because along with $\vec{r}$, they form two planes. In addition, we define two angles, θ_1 and θ_2, which are the angles between the dipole moment vectors $\vec{b_i}$ and $\vec{r}$.

The value for the node r in Figure 6.3b is the same as for the 2D case while for the other reaction coordinates $\theta_1 = \arccos\frac{\vec{b}_1\cdot\vec{r}}{|\vec{b}_1\cdot\vec{r}|}$, $\theta_2 = \arccos\frac{\vec{b}_2\cdot\vec{r}}{|\vec{b}_2\cdot\vec{r}|}$ and $\phi = \arccos\frac{\vec{m}\cdot\vec{n}}{|\vec{m}\cdot\vec{n}|}$. For ϕ, $\vec{m} = \vec{b}_1 \times \vec{r}$ and $\vec{n} = \vec{b}_2 \times \vec{r}$ are the normal to planes defined by each water molecule's dipole moment. The atomic biasing forces resulting from r, θ_1 and θ_2 are much the same for the 1D and 2D cases, however, for the dihedral angle ϕ, Equation (6.3) gives the forces sent via edges to the point nodes used to define the vector nodes $\vec{r}$, $\vec{b}_1$, and $\vec{b}_2$.

$$
\begin{aligned}
\vec{F}_{W_1} &= -\frac{\partial V}{\partial \phi}\,|r|\,\frac{\vec{m}}{|m|^2} \\
\vec{F}_{W_2} &= -\frac{\partial V}{\partial \phi}\,|r|\,\frac{\vec{n}}{|n|^2} \\
\vec{F}_{W_1(O)} &= -\vec{F}_{W_1} + \left(\frac{\vec{r}\cdot\vec{b}_1}{r^2}\right)\vec{F}_{W_1} + \left(\frac{\vec{r}\cdot\vec{b}_2}{r^2}\right)\vec{F}_{W_2} \\
\vec{F}_{W_2(O)} &= -\vec{F}_{W_2} - \left(\frac{\vec{r}\cdot\vec{b}_1}{r^2}\right)\vec{F}_{W_1} - \left(\frac{\vec{r}\cdot\vec{b}_2}{r^2}\right)\vec{F}_{W_2}
\end{aligned}
\tag{6.3}
$$

The 4D classical water dimer FEARCF simulation is run for 50ps each iteration, with 128 simultaneous simulations in each iteration, and each simulation run with a single core. Histogram dimension for reaction coordinate r comprised 41 divisions over a range of [0.5,10] Å, with the histogram dimensions for reactions coordinates θ_1 and θ_2 being 21 divisions over a range of [-90,90] degrees each, and the histogram dimension for reaction coordinate ϕ comprised of 21 divisions, over a range of [0,180] degrees. The sampling for all ξ_I in the first FEARCF iteration (Figure 6.3c) with no biasing forces has ($\xi_1 = r$;$\xi_2 = \theta_1$;$\xi_3 = \theta_2$;$\xi_4 = \phi$) with a high-to-low ratio for the distance of (57.4:1) while after the 15th FEARCF iteration (Figure 6.3e) approximate convergence was reached with high-to-low ratios of (1.8:1).

The preference for $\phi = 180°$ and $\theta_{1/2} = 52°/128°$ is due to 52° being half of the TIP3P parameter angle [156] formed between the two hydrogen-oxygen covalent bonds. At $\theta_1 = 52°$, one of the hydrogen atoms of the first water is

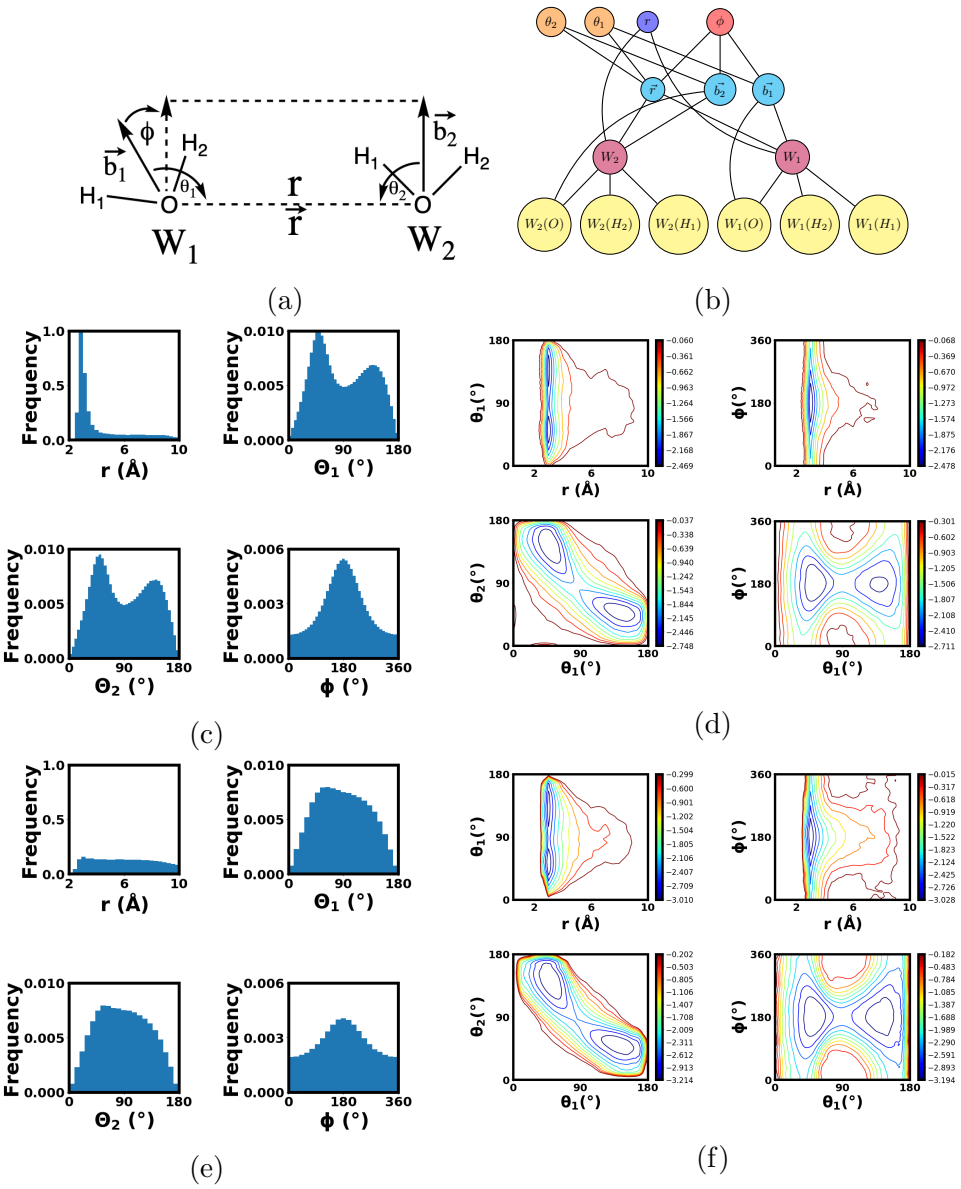


Figure 6.3: 4D classical water dimer FEARCF simulation a) 4D RCdefinitions b) FEARCF RC graph representation c) Histograms showing sampling of 1st iteration d) 2D Boltzmann averaged FE surfaces for 1st iteration with contours at 1/4 k_BT intervals e) Histograms showing sampling of 5th iteration f) 2D Boltzmann averaged FE surfaces for 5th iteration with contours at 1/2 k_BT intervals.

now in line with the vector $\vec{r}$ and facing the oxygen of the second water, while one of the hydrogens of the second water is also in line with $\vec{r}$ but facing away from the other oxygen. Lastly, $\phi = 180°$ indicates that the second hydrogen of each water molecule orients itself away from each other due to electrostatic repulsion.

The above results for the classical TIP3P are consistent with its parametisation but we now consider but the above results might look using an *ab initio* model of water.

6.3 QM Results

Below are presented free energy results using an *ab initio* model of water and DFT methods.

6.3.1 Simulation Setup

All *ab initio* water dimer simulations were run in NWCHEM version 7.0.2 [131] using the QMD Gaussian basis-set [73] module with 10ps per iteration, 30 simultaneous simulations, and eight cores for each simulation. The aug-cc-pVDZ basis set and the X3LYP exchange-correlation functional were used as inputs for DFT. This is following Plumley and Dannenberg [161] reporting that the aug-cc-pVDZ basis set and the X3LYP exchange-correlation functional reproduced similar binding energies to functionals with larger number of terms (like aug-cc-pVTZ) but without CP-corrections. In addition, they found that the bond distances for O-H and O-O were better modelled unlike some small functionals that over or underestimate the bond length. The water molecules were kept in proximity by utilising a cubic cavity of 12 Å that applies a spring potential to any atom that moves outside the cavity, with a spring constant of 1.23 N/m. This spring constant was calculated to ensure that a water molecule would not move more than 2 Å outside the cavity. The reduced iteration time is because of limitations in the memory management of NWChem for long dynamic runs.

6.3.2 1D-FEV-Distance

The 1D water dimer reaction coordinate was defined in the same manner as in the classical case. The results of a 30 iteration FEARCF simulation are given in Figure 6.4 with the initial sampling high-to-low ratio of $\xi_1 = r$ being 59:1 (Figure 6.4a) while the final ratio is 2:1 (Figure 6.4c). The most important thing to note is the difference in minimum energy between the classical model and the QM model. Figure 6.1f shows the classical model having a minimum bonding energy of -1.88 kcal/mol while Figure 6.4d shows the *ab initio* water having a minimum bonding energy of -3.92 kcal/mol. The QM result is closer to the experimental value reported by Rocher-Casterline *et. al.* [162] which gave a value of the dissociation energy of -3.15±0.03 kcal/mol. It is also interesting to note that both Figure 6.1f and 6.2f show a secondary energy minimum at around 8 Å which is not present in Figure 6.4d. This is possibly due to the effects of periodic boundary conditions which are not present in the QM case.

Another noticeable difference between the classical and the QM models is the location of the minimum energy. In the classical model (Figure 6.1f) the minimum energy is at r=2.78Å while the QM model has r=2.97Å. While the TIP3P model uses the Leonard-Jones potential to model the repulsive effect of the Pauli exclusion principle [156], DFT more accurately calculates the effect of electron orbital overlaps. The minimum contact distance for DFT 2.51Å is, however, slightly lower than the one from TIP3P 2.56Å.

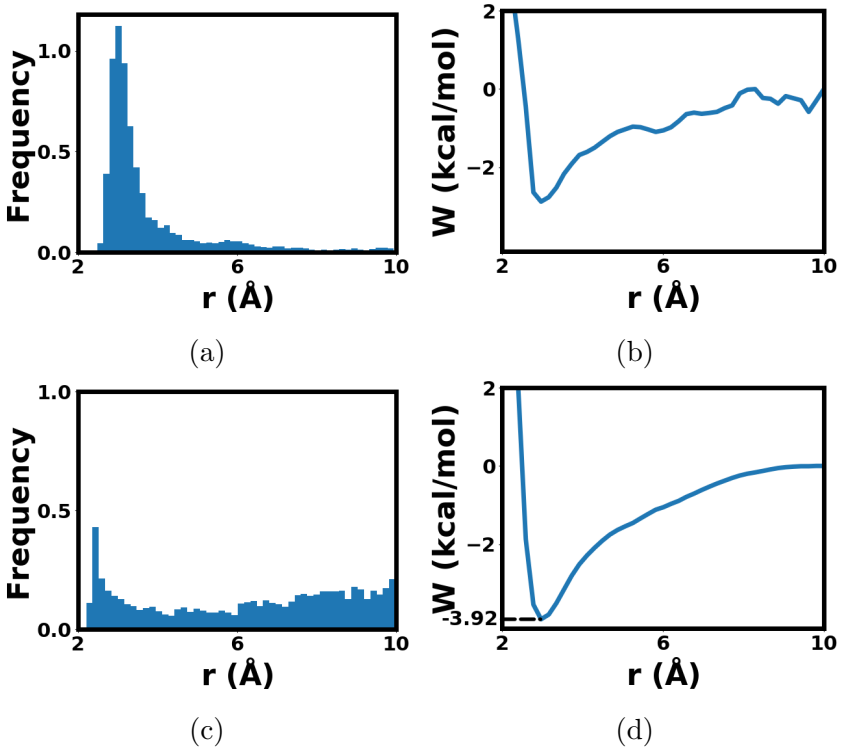


Figure 6.4: a) 1D RC definition for 1D water dimer system b) 1D water dimer RC graph representation c) Histogram showing sampling of 1st iteration d) FE surface of 1st iteration e) Histogram showing sampling of 30th iteration f) FE surface of 30th iteration

6.3.3 2D-FEV-Distance and Angle

The 2D water dimer reaction coordinates were defined in the same manner as in the classical case. The initial high-to-low sampling ratio in the first FEARCF iteration for the variables ($\xi_1 = r$,$\xi_2 = \phi$) was (30.4:1,33.8:1) (Figure 6.5a) while the final sampling ratio in the 30th FEARCF iteration is (1.8:1,23.5:1) (Figure 6.5d). Observing the final surface in Figure 6.5d we see there is a large minima from $\phi = (66°, 128°)$ and r=(2.97Å,3.21Å) which is noticeably different from the range shown in Figure 6.2f which was $\phi = (40°, 120°)$ and r=(2.94Å,3.2Å). Figure 6.5d is also noticeably asymmetric with a much stronger potential wall for $\phi < 90$. This could possibly be caused by a difference in repulsive mechanism when the dipoles are aligned

versus when they are not, which is not modelled by the TIP3P model. If the dipoles are aligned, the waters are either in a bifurcated bonding mode or stacked hydrogen-to-hydrogen and oxygen-to-oxygen. If the dipoles are anti-aligned, the waters either have the oxygens facing each other with hydrogens far apart or stacked with hydrogen-to-oxygen. The potential overlap of hydrogen molecular orbitals may be more intense than the $1/r^{12}$ term in the Leonard-Jones portion of the CHARMM force field allows, or the partial charges of TIP3P do not create the electrostatic repulsion at the close contact that is given by DFT and its set nuclear charges.

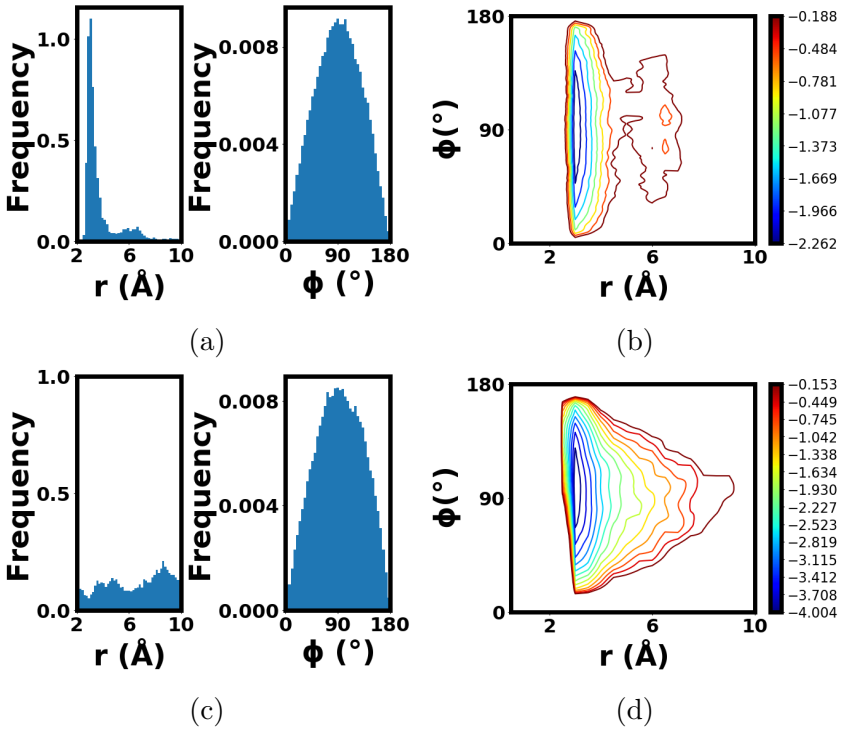


Figure 6.5: a) 2D RC definition for 2D water dimer system b) 2D water dimer RC graph representation c) Histogram showing sampling of 1st iteration d) FE surface of 1st iteration with contours at 1/2 k_BT intervals e) Histogram showing sampling of 30th iteration f) FE surface of 30th iteration with contours at 1/2 k_BT intervals

6.3.4 4D-FEV-Distance, Angles and Dihedral

The 4D water dimer reaction coordinates were defined in the same manner as in the classical case. The initial high-to-low sampling ratio in the first FEARCF iteration (Figure 6.6a) for ($\xi_1 = r$,$\xi_2 = \theta_1$,$\xi_3 = \theta_2$,$\xi_4 = \phi$) was (240:1,26:1,30.7:1,4.3:1), while the sampling ration in the 40th FEARCF iteration (Figure 6.6c) was (2.1:1,16.7:1,16.6:1,2.5:1). The separation between the 2 minimum energy wells in Figure 6.3f is considerably greater than the separation shown in Figure 6.6d, indicating that the exchange of hydrogen

bonds is much easier in the *ab initio* model than the classical model.

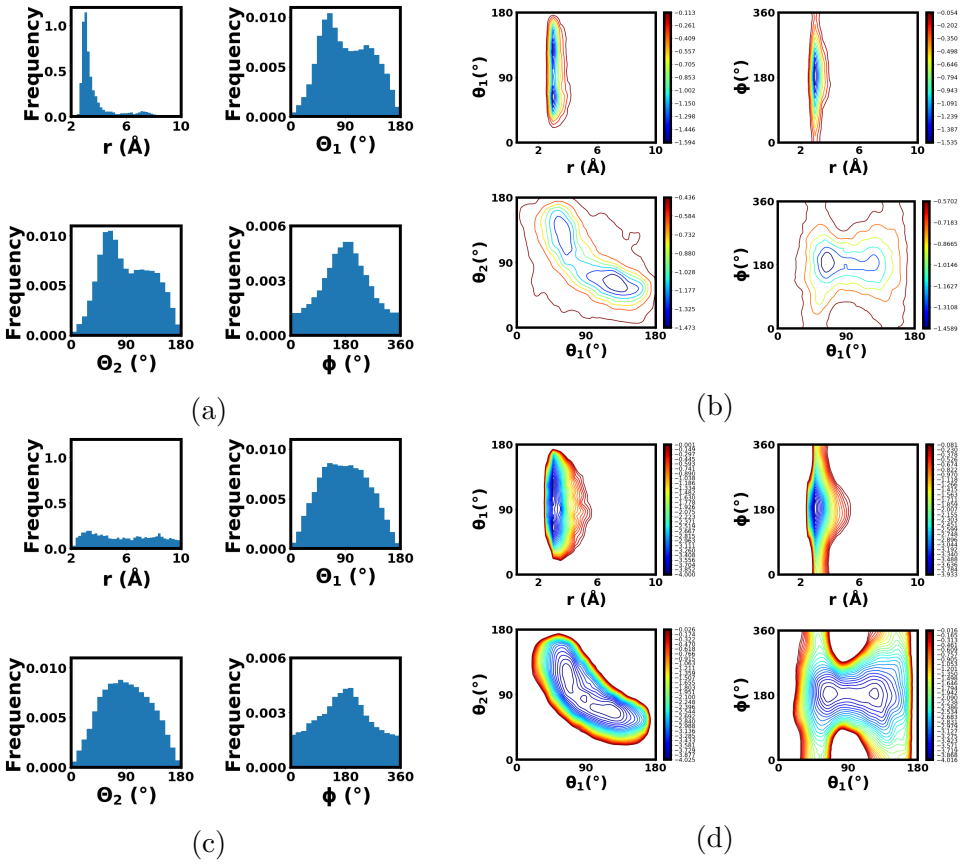


Figure 6.6: a) 4D RC definition for 4D water dimer system b) 4D water dimer RC graph representation c) Histograms showing sampling of 1st iteration d) 2D Boltzmann averaged FE surfaces for 1st iteration with contours at 1/4 k_BT intervals e) Histograms showing sampling of 40th iteration f) 2D Boltzmann averaged FE surfaces for 40th iteration with contours at 1/4 k_BT intervals.

6.3.5 3D-FEV Minimum energy path

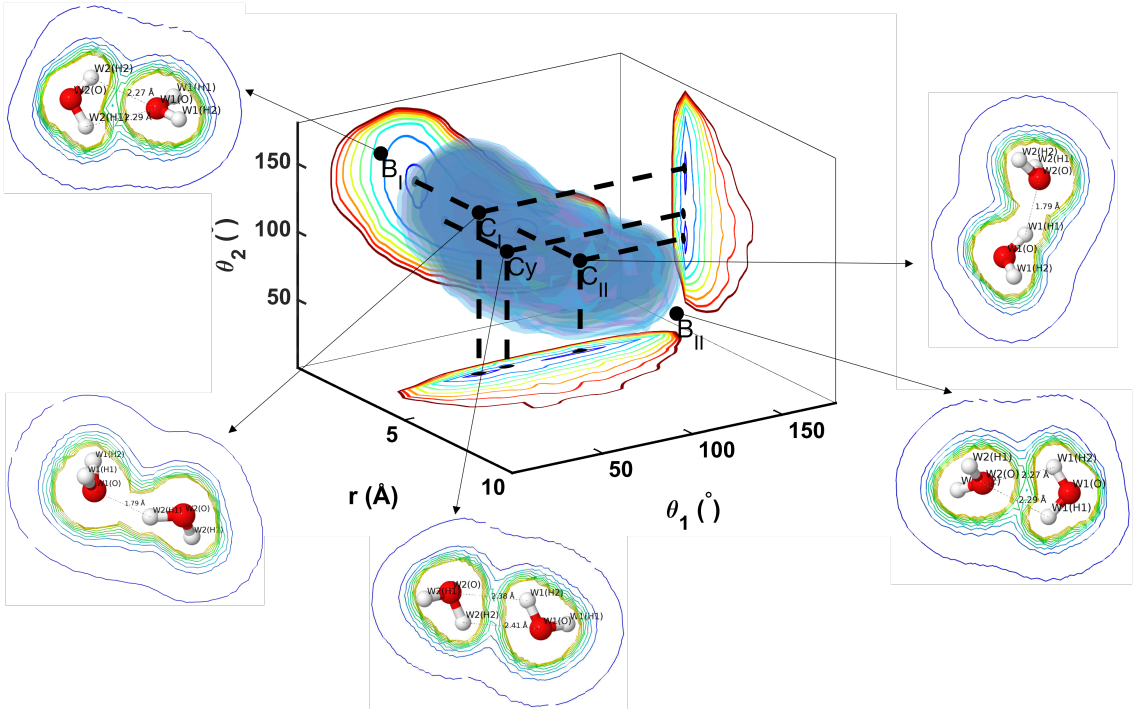
The three potential configurations for a water dimer to form hydrogen bonds, Chain (C), Cyclic (C_y), and Bifurcated (B), are illustrated in Figure 6.7b. There is a minima well observed in the 4D *ab initio* water dimer FEV that comprises of two configuration types: C and C_y. Since the system is symmetrical, two equivalent Chain configurations are seen; C_I (r = 2.96Å, θ_1 = 65.53°, θ_2 = 118.33°) and C_{II} (r = 2.90Å, θ_1 = 122.61°, θ_2 = 66.23°), with C_y (r = 2.90Å, θ_1 = 81.87°, θ_2 = 84.50°) midway between the two. While the r minima value agrees with the classical value, the θ_1 and θ_2 minima values do not. This is due to the decreased rigidity of the *ab initio* model allowing for the water molecule to deform when strongly hydrogen bonded and there is electron density sharing between the hydrogen donor and the oxygen acceptor atoms. Overlaid in Figure 6.7a are electron density contour plots of

C_I, C_{II}, C_y, B_I, and B_{II}. The closest contours are of density 0.03 au, where it is apparent that there is sharing of electrons in the chain cases, C_I and C_{II}, that does not occur for C_y, B_I, or B_{II}. This implies that the hydrogen bonds formed at C_I and C_{II} are not purely electrostatic in nature, but also involve deformation of the electron clouds of the hydrogen bond donor and acceptor (Figure 6.7b and Table 6.1). While the C and C_y configurations are located in the k_BT minima well, the B configurations are just outside of the $6k_BT$ broad envelop of hydrogen bonded configurations. The geometric and electronic values for the hydrogen bonding configurations derived from the FEV and subsequent electronic analysis are summarised in Table 6.1.

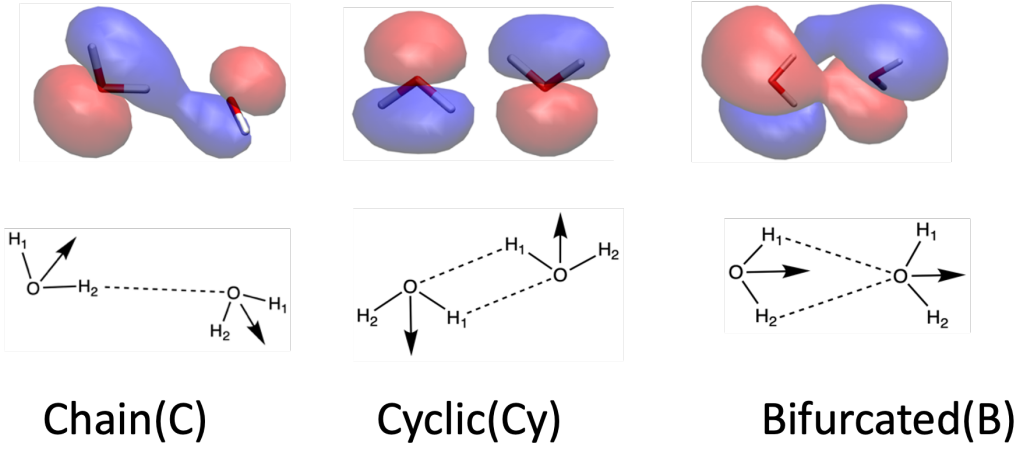
Table 6.1: Differences in hydrogen bond length, electron density and free energy for the three types of water dimer configurations.

HB Configuration	r(Å)	$\theta_1(^\circ)$	$\theta_2(^\circ)$	$\phi(^\circ)$	$R_{H(D)--O(A)}$(Å)	ρ(au)	ΔG(kcal/mol)
C	2.96	65.6	118.3	183.1	1.94	0.022	-3.92
Cy	2.9	81.9	84.5	180.6	2.28	0.010	-3.79
B	2.7	172.4	7.9	-	2.27	0.006	9.67

The 4D free energy hyper-surfaces (Figure 6.7a) provided clues to hydrogen bonding configurations of the water dimer, as well as the shape of the ensemble of configurations. Now we attempt to uncover the mechanisms of the dimer association within the hydrogen bond minima well, and the pathway from hydrogen bonding to dimer dissociation through a projection of trajectories onto a 2D FEV (Figure 6.8a). In the case of association, a trajectory that traverses within the $3k_BT$ minima FE surface well appears to librate about the chain type configurations, which is shown in the left panel of Figure 6.8a. It does this by passing through a cyclic configuration in-between the two low energy chain configurations. The local minima labelled C_I (Figure 6.7a) is defined by a hydrogen bond between W1(H2) and W2(O), with an overlap of the electron density highlighting the covalent nature of hydrogen bonds. Another local minima labelled C_{II}, where W2(H2) is the hydrogen bond donor and W1(O) the acceptor, is the symmetric equivalent of C_I. In between these two states is a pseudo transition state labelled as C_y, where the two molecules are straddled alongside each other, no longer share electron clouds, and where both molecules are hydrogen bond donors and acceptors. It is a is a mid-point between C_I and C_{II}.



(a)



(b)

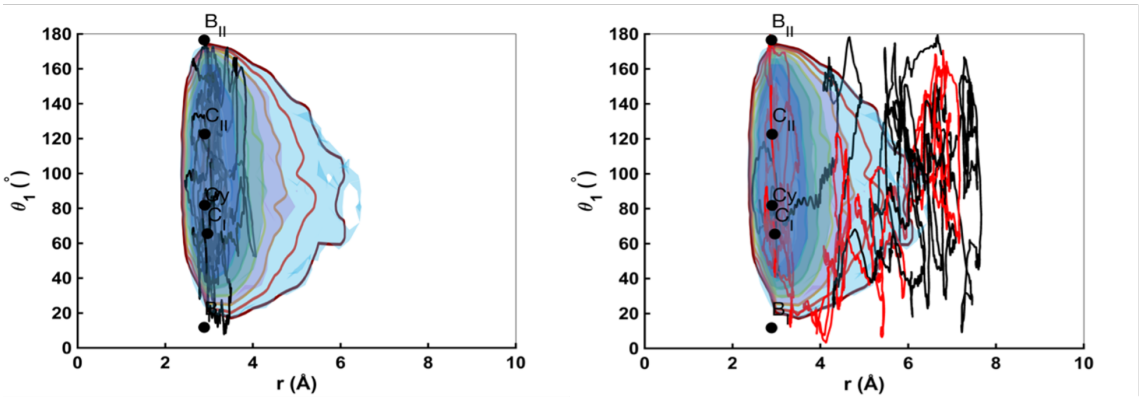
Figure 6.7: a) 3D FEV for 30th FEARCF iteration with iso-surfaces and contours, along with overlaid electron density contours at Chain, Cyclic, and Bifurcated configurations b) molecular orbital plot of $3a_1$ for proton donor and $1b_1$ for proton acceptor at 0.03 au for the 3 hydrogen bonding configurations with positive density in red and negative density in blue along with illustration of each configuration.

6.3.6 Dipole Autocorrelation Function and Time Series

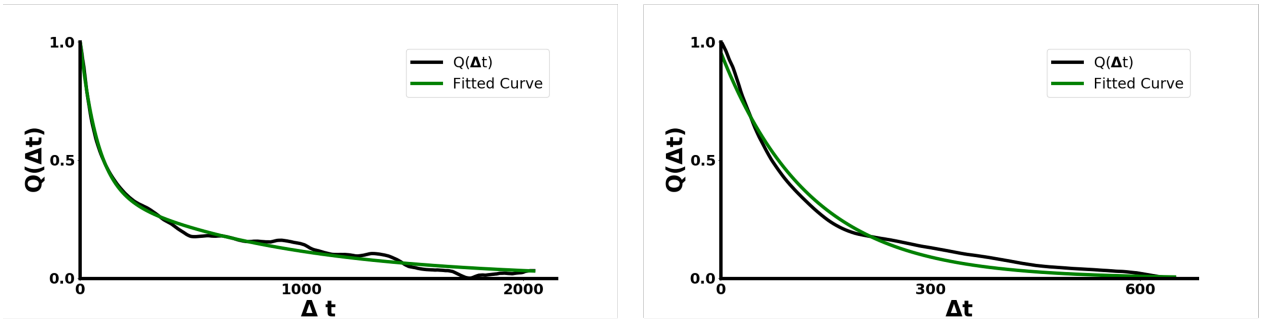
The nature and extent of the water association can also be measured through the dipole-dipole correlation function [163] given by $Q(t) =< D(0) \cdot D(t) >$,

where $D(t)$ is the total dipole moment of both water molecules at time t (left panel of Figure 6.8b). To fit the $Q(t)$ decay, two exponentials of the form $e^{-(t+x_1)/\tau_1} + e^{-(t+x_2)/\tau_2}$ were used. From this, the relaxation times and fitting parameters $\tau_1 = 75.7 \pm 1.2$,$\tau_2 = 798.2 \pm 6.4$,$x_1 = 36.93 \pm 0.87$,$x_2 = 729 \pm 11$) were calculated. The two exponential terms needed for the fit implies that within the association well there are two processes at play. The relaxation time is a measure of how fast a particular quantity is becoming uncorrelated [164]. Trajectories from the association well were analysed to unpack the two events (left panel Figure 6.8c). The faster relaxation time τ_1 is due to the librational motion of both dipoles within the minima well about the angles of C_I ($\theta_1 = 65.53°$, $\theta_2 = 118.33°$) and C_{II} ($\theta_1 = 122.61°$, $\theta_2 = 66.23°$). This observation for an isolated water dimer is consistent with the well-known phenomenon of water molecule libration in bulk liquid indicating that water libration in hydrogen bonding configurations is an innate molecular property rather than a condensed phase induced phenomenon [165]. The slower relaxation time τ_2 is due to the periodic migration from one minima well C_I/C_{II} to another C_{II}/C_I, via the transition configuration Cy (Table 6.1).

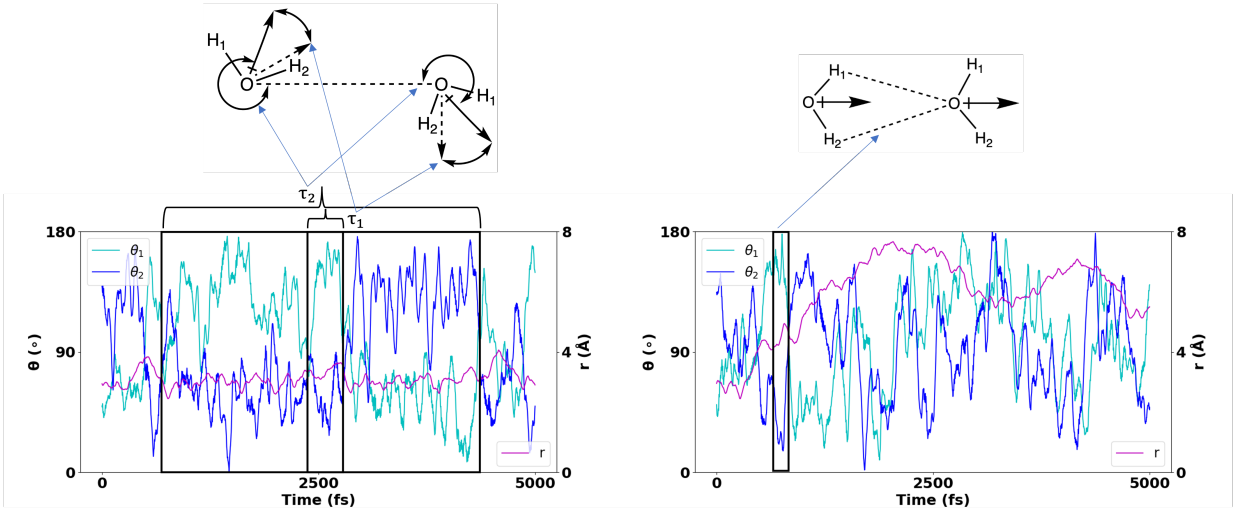
Turning our attention to the process of hydrogen bond dissociation (right panel Figure 6.8a), we investigate trajectories to understand if there is a systematic molecular process of the hydrogen bond between the waters breaking as they drift apart. After passing the energy barrier of $6k_BT$= 3.55kcal/mol, the water molecules remain more than 5Å apart, are longer hydrogen bonded, and do not have an enthalpic means to re-establish a hydrogen bond. The time correlation function can be fitted to a single exponential function (right panel Figure 6.8b) of the form $e^{\frac{-(t+x)}{\tau}}$, where τ is the relaxation time that required values of ($\tau = 127.0 \pm 1$,$x = 5.63 \pm 0.77$). Analysing the dissociation trajectories, two of which are projected onto the free energy surface (right panel in Figure 6.8a), shows that the exit out of the $6k_BT$ hydrogen bond zone is made via a bifurcated hydrogen bonding configuration. A closer look at this through an examination of the time series of the θ reaction coordinates (right panel Figure 6.8c) confirms that the gateway out of the hydrogen bond zone of $r > 4$Å is through the B_I and B_{II} configurations (red and black trajectories on right panel Figure 6.8a).



(a)



(b)



(c)

Figure 6.8: a) Examples of three water dimer MD trajectories that either remain associated or become disassociated b) averaged dipole-dipole auto-correlation plots of 16 similar trajectories for each case c) Time series for the θ and r RC values

We have presented results from both classical and quantum models of the water dimer, demonstrating the successful implementation of the FEARCF library into multiple MD software packages. In addition, we have shown the

success of using a multidimensional approach to free energy as opposed to a simple one dimensional approach. The next chapter will be dedicated to demonstrating the advantages of the OOP design of the FEARCF library.

Chapter 7

Application Case Study: Glucose Pucker

This chapter presents free energy results for ring puckering glucose. This is used to ensure equivalence with the previous implementation of ring puckering in CHARMM by Barnett [166], but in a more efficient manner provided by the OOP design of the FEARCF library.

7.1 The Glucose Sugar Ring and Puckering

Glucose is an important building block for many complex carbohydrates in organic organisms and one of its most important features is its high solubility in water [167]. It is known that the puckering of the glucose carbon ring is a key component in the mechanisms of certain enzymatic reactions involving glucose, but previous work as also suggested it has a role to play in glucoses solubility namely, due to the ability to mimic the tetrahedral structure of liquid water [166]. There are 38 canonical pucker conformers that a pyranose carbon ring like glucose may adopt consisting of various chair (C), half-chair (H), boat (B), skew (S), and envelope (E) shapes. Only a few are these conformers are stable [168]. We therefore will use our FEV analysis to understand which pucker configurations are most favoured for a classical glucose model in vacuum.

7.2 Classical Results

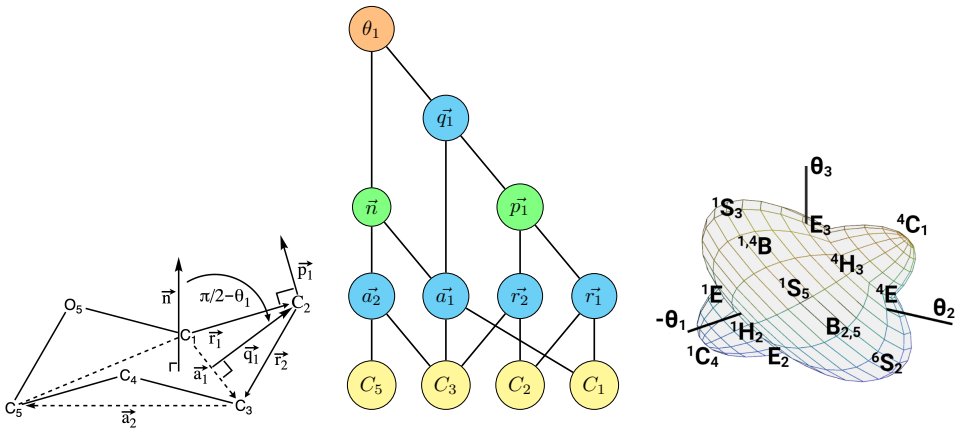
Below is presented the results generated using a classical model in glucose using the CHARMM force field.

7.2.1 Simulation Setup

All classical MD simulations are run in CHARMM 42b2 [130] with the CHARMM force field [54], run at 300K in a periodic box of size 14Å each side, with non-bonded interactions cut-off at 13Å, the Velocity-Verlet integrator [7] with a Nose-Hoover NVT ensemble thermostat [63] with 1 heat bath, a tau value of 0.1ps and 10 sub-steps per time step. Each simulation is run with a 1fs time step, with no equilibrium steps and a unique set of four seven-digit random seed numbers.

7.2.2 3D-FEV Ring Puckering

Figure 7.1a shows the triangular tessellation definition of the first pucker angle for a glucose ring [153]. Each angle θ_i is defined as $\pi/2$ minus the angle between the normal of the reference plane $\vec{n}$, and a vector $\vec{q}_i$ in the side plane that is also perpendicular to the line of intersection $\vec{a}_i$ between the central reference plane and the side plane. Figure 7.1b simplicity shows, for simplicity, the graph for just one pucker angle and the value of the θ_i is given by $\theta_i = \frac{\pi}{2} - \arccos \frac{\vec{q}_i \cdot \vec{n}}{|\vec{q}_i \cdot \vec{n}|}$.



(a) (b) (c)

Figure 7.1: Classical model of β-D-glucose a) definition of glucose ring pucker angle θ_1 b) 3D glucose ring pucker RC graph representation for θ_1 c) pucker conformations plotted onto a globe defined by the three pucker angles defined by a six membered ring like glucose

The edges from the θ_1 node in Figure 7.1b give forces in terms of $\vec{n}$ and $\vec{q}_i$, with the expression of these forms and the resultant force terms for each node, given in Equation (7.1). Note that the final atomic force terms all sum to zero, ensuring conservation of momentum for the system as a whole.

$$
\begin{aligned}
\vec{F}_{\vec{q}_1} &= \frac{\partial V}{\partial \theta_1} \frac{1}{\cos\theta_1} \frac{1}{|\vec{q}_1|} \left(\mathbb{1} - \widehat{q_1} \otimes \widehat{q_1}\right) \hat{n} \\
\vec{F}_{\vec{n}} &= \frac{\partial V}{\partial \theta_1} \frac{1}{\cos\theta_1} \frac{1}{|\vec{n}|} \left(\mathbb{1} - \hat{n} \otimes \hat{n}\right) \widehat{q_1} \\
\vec{F}_{\vec{p}_1} &= -\vec{a}_i \times \vec{F}_{\vec{q}_1} \\
\vec{F}_{\vec{a}_1} &= \vec{p}_i \times \vec{F}_{\vec{q}_1} - \vec{a}_2 \times \vec{F}_{\vec{n}} \\
\vec{F}_{\vec{a}_2} &= \vec{a}_1 \times \vec{F}_{\vec{n}} \\
\vec{F}_{\vec{r}_1} &= \vec{r}_2 \times \vec{F}_{\vec{p}_1} \\
\vec{F}_{\vec{r}_2} &= -\vec{r}_1 \times \vec{F}_{\vec{p}_1} \\
\vec{F}_{C_5} &= \vec{F}_{\vec{a}_2} \\
\vec{F}_{C_3} &= \vec{F}_{\vec{r}_2} - \vec{F}_{\vec{a}_2} + \vec{F}_{\vec{a}_1} \\
\vec{F}_{C_2} &= \vec{F}_{\vec{r}_1} - \vec{F}_{\vec{r}_2} \\
\vec{F}_{C_1} &= -\vec{F}_{\vec{r}_1} - \vec{F}_{\vec{a}_1}
\end{aligned}
\tag{7.1}
$$

The 3D classical glucose FEARCF simulation is run for 50ps each iteration, with 128 simultaneous simulations in each iteration, each simulation run with a single core. Histogram dimension each for reaction coordinate θ_i is comprised of 41 divisions, over range of [90,90] degrees for each pucker angle.

Figure 7.2a shows that during the 1st FEARCF iteration, where there is no biasing potential, the glucose molecule stays exclusively in the ${}^4C_1$ conformation which indicates that is a minimum energy conformation. This is confirmed when looking at the FEV in Figure 7.2b, where the only visible minima is at ${}^4C_1$. All other conformers are at much higher energies. This is consistent with the fact that chair conformers are known to be the most stable pucker conformers for cyclohexane. The question remains, however, why the other chair conformation ${}^1C_4$ is not being sampled.

Figure 7.2c shows the change in the conformational sampling during the 2nd FEARCF iteration, where now ${}^4C_1$ is not the only conformer being sampled, but the ${}^1S_3$ conformer is now being sampled as well. This shows that FEARCF is applying its biasing potential by clearly forcing the system to sample conformers that are not the global minima. Looking at the updated FEV in Figure 7.2d, we can now see that the iso-surfaces have been expanded, and now include a 8 kcal/mol that also appears around ${}^1S_3$, which is still higher than ${}^4C_1$. This evolution of the FEV from which FEARCF derives its biasing potential ensures that sampling will continue to move away from these volumes of lower energy, because they will be inversely be treated as volumes of higher energy according to the biasing potential.

Skipping ahead to 30 iterations, Figure 7.2e shows flat histogram sampling for all pucker conformations with a ratio of least sample to most sampled

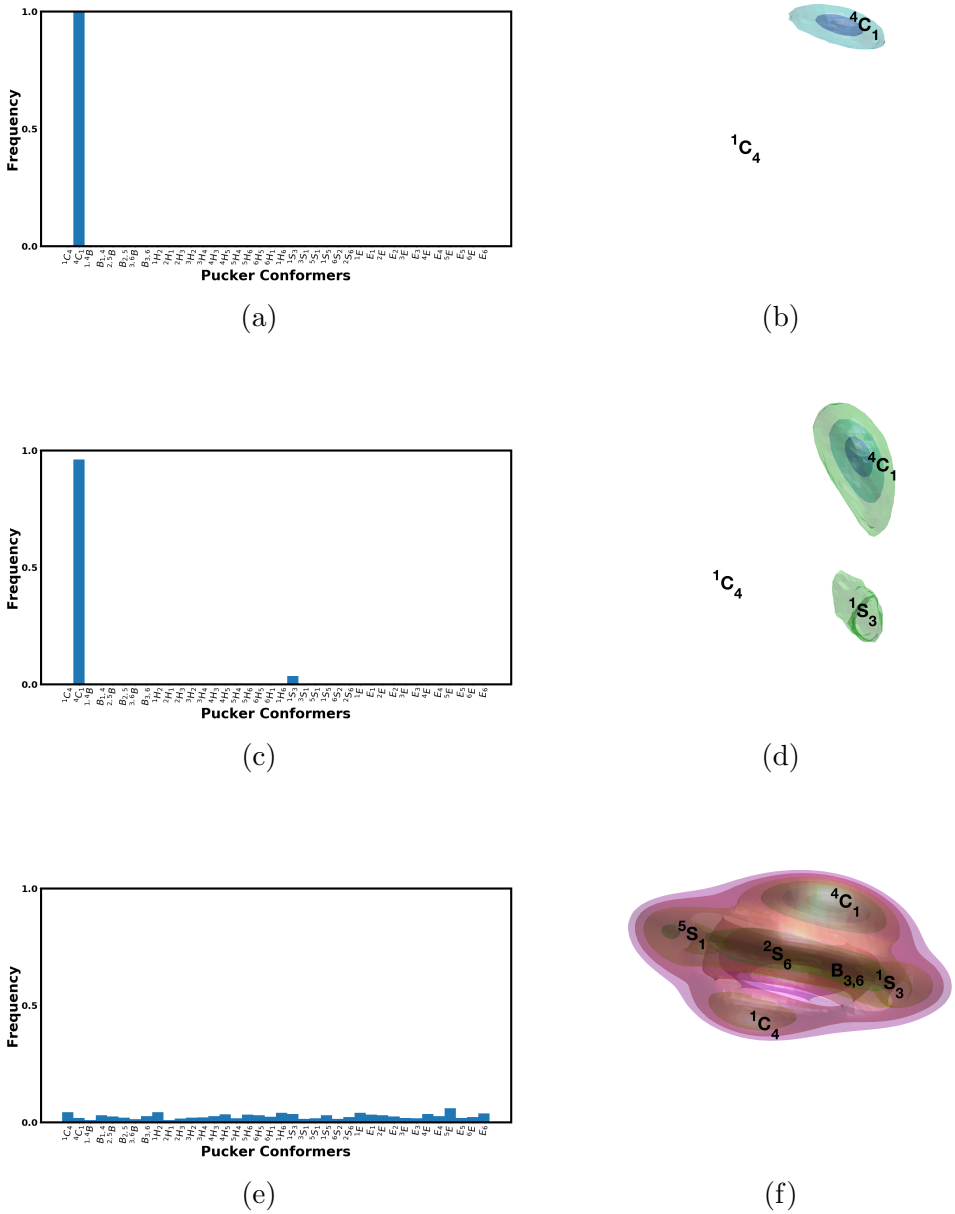


Figure 7.2: Result for glucose ring puckering a) Histogram showing canonical pucker conformation sampling of 1st iteration b) FE surface of 1st iteration with iso-surfaces at 1 and 4 kcal/mol c) Histogram showing canonical pucker conformation sampling of 2nd iteration d) FE surface of 2nd iteration with iso-surfaces at 1, 4, and 8 kcal/mol e) Histogram showing canonical pucker conformations sampling of 30th iteration f) FE surface of 30th iteration with iso-surfaces at 1,4,8,12,16, and 20 kcal/mol.

being 1:6.5. What this means is that the net potential acting on the system, a result of combining the CHARMM atomic force field and the numerical

biasing force from FEARCF, is approximately equal at all points in free energy space. Looking at Figure 7.2f, we can identify the lowest non-chair conformer at ${}^{2}S_{6}$ with a free energy value of 5.9 kcal/mol, however, the global minima remains at ${}^{4}C_{1}$. The next lowest energy state is $B_{3,6}$ at 7.8 kcal/mol, followed by ${}^{1}S_{3}$(7.9 kcal/mol), and ${}^{5}S_{1}$ (8 kcal/mol). We can actually see that ${}^{1}C_{4}$ has a free energy value approximately 9.4 kcal/mol greater than ${}^{4}C_{1}$. To understand this difference, we can examine the structure of the two chair conformers. For ${}^{4}C_{1}$ in Figure 7.4, the hydroxyl groups are all roughly level with each other, parallel with the ring, and pointing outwards. For the ${}^{1}C_{4}$ in Figure 7.4, they are forced to become perpendicular to the ring and therefore closer together, creating greater electrostatic repulsion.

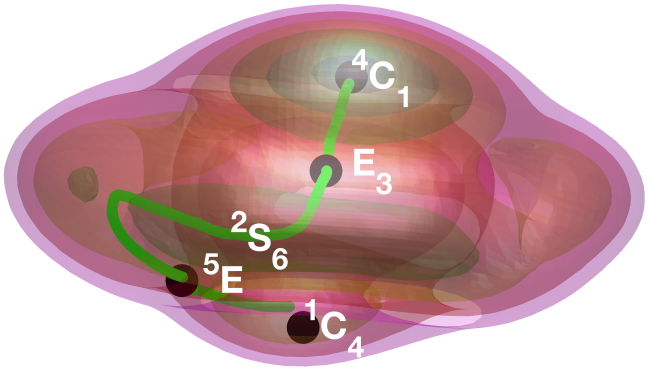


Figure 7.3: Classical glucose 3D FEV with calculated MEP with highlighted pucker conformations along the path.

We can then consider the minimum energy path (MEP) between the two chairs conformers. This is given in Figure 7.3, where the MEP is plotted on top of the 3D FEV. In Figure 7.4, the MEP is then plotted as a 1D plot of free energy versus distance from the ${}^{4}C_{1}$ conformer in phase space. The MEP was calculated using simple gradient descent, specifically Forward Euler Integration, to find local minima. Previous work has indicated E_3 as an important transition state from ${}^{4}C_{1}$ for glucose from during some enzymatic reactions [166], so it was chosen as the initial starting point for the gradient descent. Two gradient descents were then formed, one towards ${}^{4}C_{1}$, and one

in the opposite direction. The latter gradient descent terminated at the ${}^{2}S_{6}$ pucker conformation. Again using previous work [166], ${}^{5}E$ was chosen as the transition state from ${}^{1}C_{4}$ and the same gradient descents performed for E_{3}, with the first gradient descent terminating at ${}^{1}C_{4}$ and the second terminating at ${}^{2}S_{6}$.

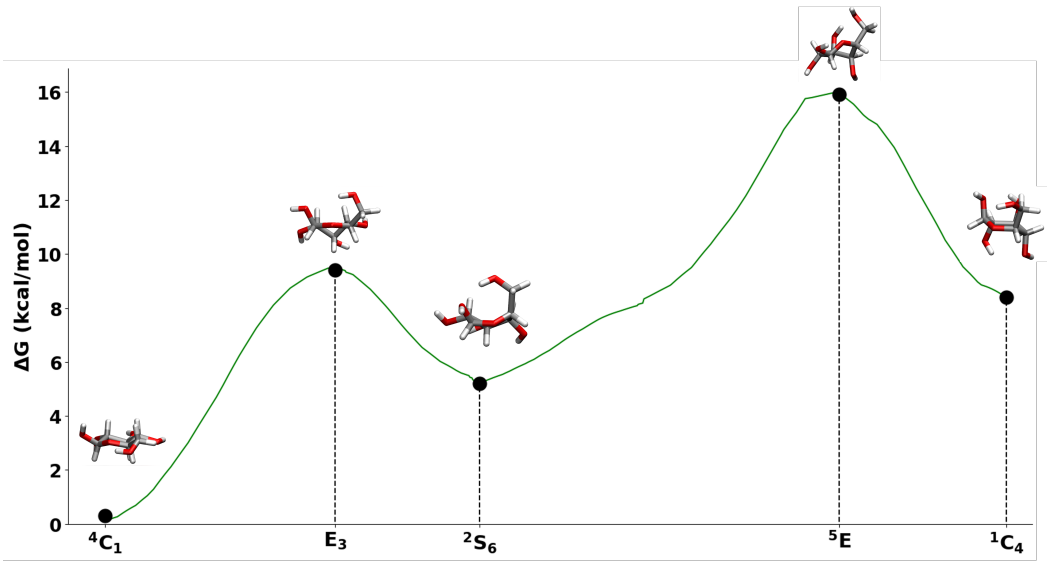


Figure 7.4: 1D plot of classical glucose calculated MEP with highlighted pucker conformations.

Now we have discovered ${}^{2}S_{6}$ as a midpoint in the transition from ${}^{4}C_{1}$ to ${}^{1}C_{4}$. We then look at its relative free energy and see that it is lower than ${}^{1}C_{4}$ and can again ask why this is. Looking at Figure 7.4 we can see that most of the hydroxyl groups are in line with the ring like in ${}^{4}C_{1}$. The exception to this is the hydroxyl bonded to the C6 carbon, which is now perpendicular to the ring. We can understand this progression more if we look at the transition states E_{3} and ${}^{5}E$. Figure 7.4 shows the progression of the C6 hydroxyl to its perpendicular orientation in ${}^{2}S_{6}$, while Figure 7.4 shows the subsequent rotation of the C1 and C3 associated hydroxyls.

Simply considering the orientation of the side-groups is not necessarily sufficient, however, and we therefore need to study changes in the atoms in the pyranose ring itself. In order to do this we need to determine the electronic structure of the bonds of the ring, which is not possible using classical theory. An *ab initio* ring puckering system is presented in the next chapter.

An important distinction between the work presented above and the previous work with ring pucker in FEARCF presented by Barnett [166], is that Barnett took his method of calculating the pucker angles, and resultant force equations, from work by Hill and Reilly [153]. He then implemented them in the FEARCF implementation in CHARMM. The limitation of the RXNCOR module as model for defining reactions coordinates with graphs became ap-

parent when Barnett was forced to write Fortran 'case' statements for each possible ring dimension from one to seven. This is due to the array structure of the 'ASCEND' and 'DESCEND' subroutines which require different 'case' statements to reference properly. This is not the case due to the OOP structure of the FEARCF library, and therefore the pucker angle and subsequent force calculations for any ring pucker dimensional case are possible with a combination of the existing vector, angle, and point subroutines within the FEARCF class. To ensure that the new OOP approach and the previous approach by Barnett were equivalent, the above free energy simulations for glucose were repeated with the previous implementation of FEARCF in CHARMM and the results were identical.

In this chapter we have shown the capability of the FEARCF library's OOP design to improve upon the previous graph-based reaction coordinates module in CHARMM, by simplifying the coding required to implement the triangular tessellation ring pucker angles as useable reaction coordinates within FEARCF. The next chapter will further show FEARCF's ability to perform free energy calculations at alternate levels of theory, as well as produce useful results in the context of enzyme catalysis.

Chapter 8

Application Case Study: OGT Reaction and Pucker

This chapter presents results of free energy from a QM/MM simulation of the OGT-GlcNAc enzyme-substrate complex, to demonstrate that FEARCF can produce free energy of this reaction system utilising this theory.

8.1 The OGT Enzyme

O-linked β-N-acetylglucosamine transferase (OGT) is an enzyme that is involved in cell signalling and recognition in mammalian cells [169]. The primary function of OGT is the post-translational modification of proteins by the addition of the sugar amide N-Acetylglucosamine (GlcNAc) to either serine or threonine residues via a glycosidic bond with their hydroxyl group [170]. This occurs within the cytoplasm, the nucleus, and the mitochondria. OGT is paired with another enzyme, N-acetyl-d-glucosaminidase (OGA), which reverses the effect of OGT by removing GlcNAc from the peptide residue it is bonded to [171]. The source of GlcNAc for OGT is uridine-diphosphate-N-acetylglucosamine (UDP-GlcNAc) which is the primary product of the hexosamine biosynthetic pathway. The transfer of GlcNAc from UDP-GlcNAc to the peptide is thought to occur via an ordered sequential bi-bi mechanism reaction, where the HSD498residue of OGT acts as a catalytic base, and LYS848 weakens the GlcNAc-UDP bond [172]. This reaction has been previously studied using FEARCF simulations and this proposed mechanism has been analysed using a combination of FEVs and MEPs. During this analysis, it was found that GlcNAc adopts an E_3 pucker conformation before it proceeds to the reaction transition state [173]. This is a potential aspect of OGT's catalytic mechanism that has not been well studied before and warrants further investigation. This chapter is focused on using FEARCF simulations to provide information about OGT's affect on GlcNAc ring pucker by comparing it to GlcNAc puckering in vacuum.

OGT has be shown to be involved in the development of insulin resistance and therefore associated with Diabetes [174]. More recently it has also been shown to possibly play a key role in cancer and tumour development. This is speculated to be due to differential gene expression but the underlying mechanism is still not fully understood [175]. The design and development of inhibitors for OGT are therefore important tools for further studying OGT's role in cancer, as well as possibly being potential drug candidates. In order to design these inhibitors, however, OGT's catalytic mechanism needs to be fully understood and therefore a study on the potential affect of OGT on GlcNAc's ring pucker is essential.

8.2 QM/MM results

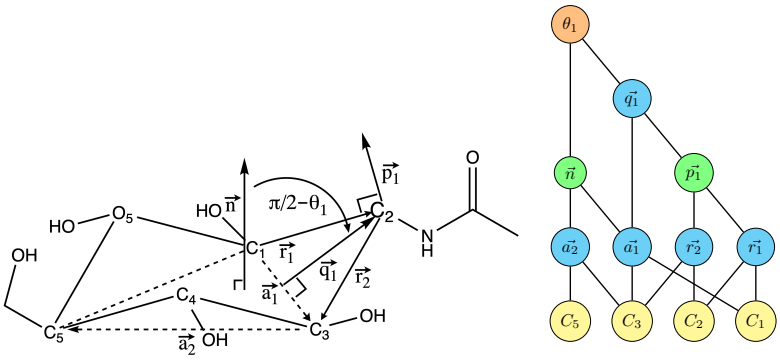
8.2.1 Simulation Setup

The FEARCF simulation of the GlcNAc ring pucker in the OGT enzyme, as well as in vacuum, was carried out using the Quantum Supercharger Library [176] [177] accelerating GAMESS-UK [178] and the CHARMM QM/MM routines [130].The dynamics simulations were run for 15 ps each, with a time step of 1fs at 300 K using the Leapfrog integrator [57]. All iterations were preceded by a 2 ps equilibrium step that did not include the biasing potential. Eight V100 GPUs were used in parallel to run one simulation each, with 15 total iterations for the enzyme and 25 iterations for the vacuum. The initial coordinates were chosen to be near the ${}^4C_1$ conformer. Histogram dimensions for each reaction coordinate were comprised of 41 divisions, over range of $[-90, 90]$ degrees, for each pucker angle.

8.2.2 GlcNAc in Vacuum

For GlcNAc, we have an identical definition for the ring pucker as was defined for glucose in Chapter 7 since they are both pyranose rings. Therefore Figure 8.1a is the same as Figure 7.1a with the exception of the amide group on the second carbon. This difference in side groups does not change anything in our constructed graph shown in Figure 8.1b, because none of these atoms are used to calculate the pucker angles.

Looking at Figure 8.2a we note, as with glucose, a very strong preference for the ${}^4C_1$ conformer. This is also reflected in the FEV in Figure 8.2b. After the sixth iteration, Figure 8.2c shows that we now have sampling at the E_3 conformer and this is shown in the FEV in Figure 8.2d. The 8 kcal/mol green isosurface now reaches E_3. We then allow it to run for several more iterations until the 25th iteration whose sampling is shown in Figure 8.2e. The sampling is not flat, however, we are not interested in sampling the whole reaction space, only the space between ${}^4C_1$ and E_3. Time and resource constraints also prevented further iterations.



(a) (b)

Figure 8.1: a) Definition of GlcNAc ring pucker angle θ_1 b) 3D GlcNAc ring pucker RC graph representation for θ_1.

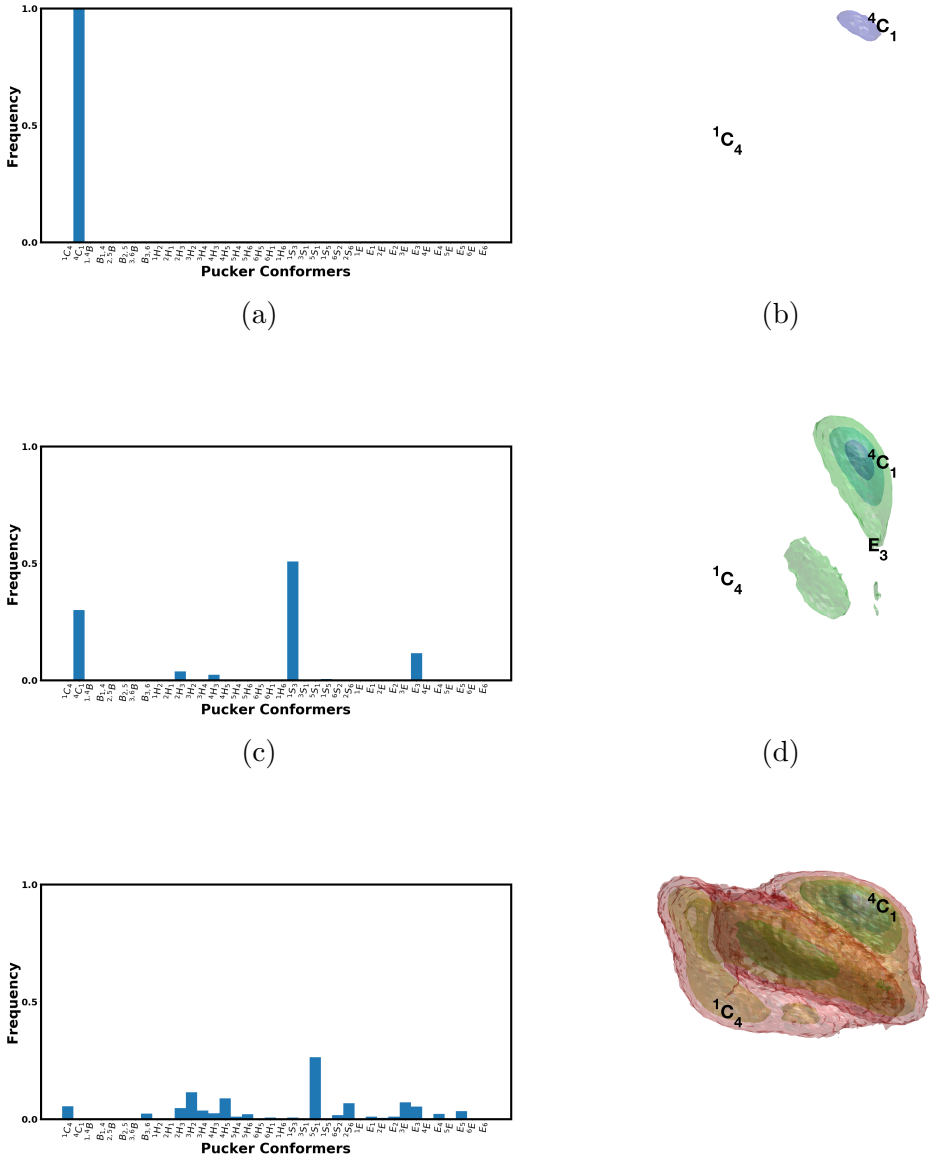


Figure 8.2: Result for GlcNAc ring puckering in vacuum a) Histogram showing canonical pucker conformation sampling of 1st iteration b) FE surface of 1st iteration with iso-surfaces at 1 and 4 kcal/mol c) Histogram showing canonical pucker conformations sampling of 6th iteration d) FE surface of 6th iteration with iso-surfaces at 1,4, and 8 kcal/mol e) Histogram showing canonical pucker conformations sampling of 25th iteration f) FE surface of 25th iteration with iso-surfaces at 1,4,8,12, and 16 kcal/mol.

8.2.3 GlcNAc in OGT

Looking at Fig 8.3a, as with the vacuum, we see a very strong preference for the $^4C_1$ conformer. This is also reflected in the FEV in Fig 8.3b. After the eighth iteration, Fig 8.3c shows that we now have sampling at the E_3 conformer and this is shown in the FEV in Fig 8.3d. We then allow it to run for several more iterations until the 15th iteration whose sampling is shown in Fig 8.3e. Again, the sampling is not flat, but we are still not interested in sampling the whole reaction space, only the space between $^4C_1$ and E_3.

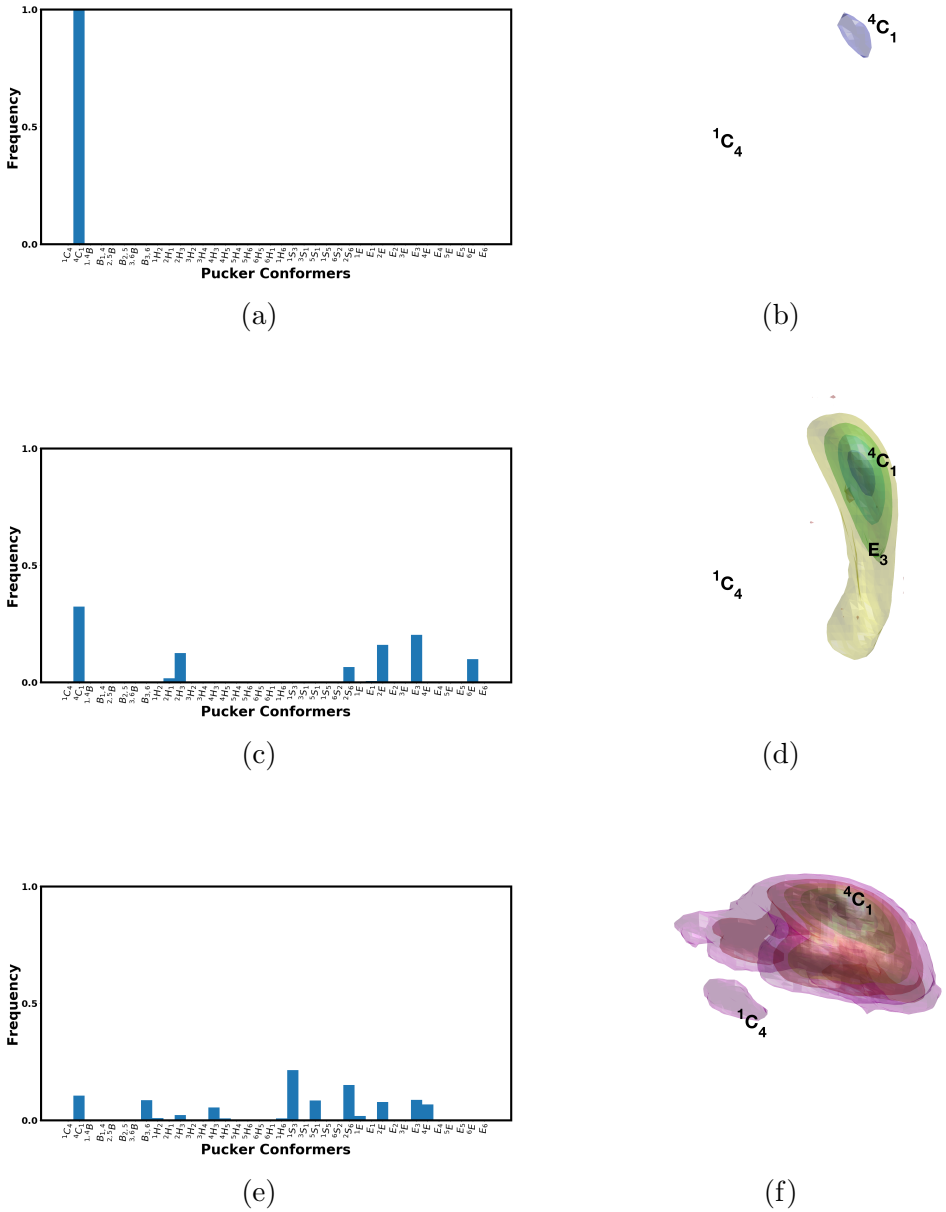


Figure 8.3: Result for GlcNAc ring puckering in OGT a) Histogram showing canonical pucker conformation sampling of 1st iteration b) FE surface of 1st iteration with iso-surfaces at 1 and 4 kcal/mol c) Histogram showing canonical pucker conformations sampling of 8th iteration d) FE surface of 8th iteration with iso-surfaces at 1,4, and 8 kcal/mol e) Histogram showing canonical pucker conformations sampling of 15th iteration f) FE surface of 15th iteration with iso-surfaces at 1,4,8,12,16, and 20 kcal/mol.

8.2.4 Comparison

We can now take the FEVs that we have generated for GlcNAc, both in vacuum and inside OGT, and compare them. The way we will do this is by

plotting a MEP from $^4C_1$ to E_3 on each FEV given in Fig 8.4a, and then compare the free energy values along this path. The calculation of the MEPs is done using gradient descent. The MEPs are then plotted as a function of distance from the $^4C_1$ conformer versus free energy. This is shown in Figure 8.4b. The electron density for the E_3 conformer in vacuum and in OGT was calculated by extracting coordinates from the FEARCF trajectories and using Gaussian [179] to calculate the electron density using the same level of theory used to run the FEARCF simulations. These electron densities are superimposed upon the molecular structure and given in Figure 8.4c.

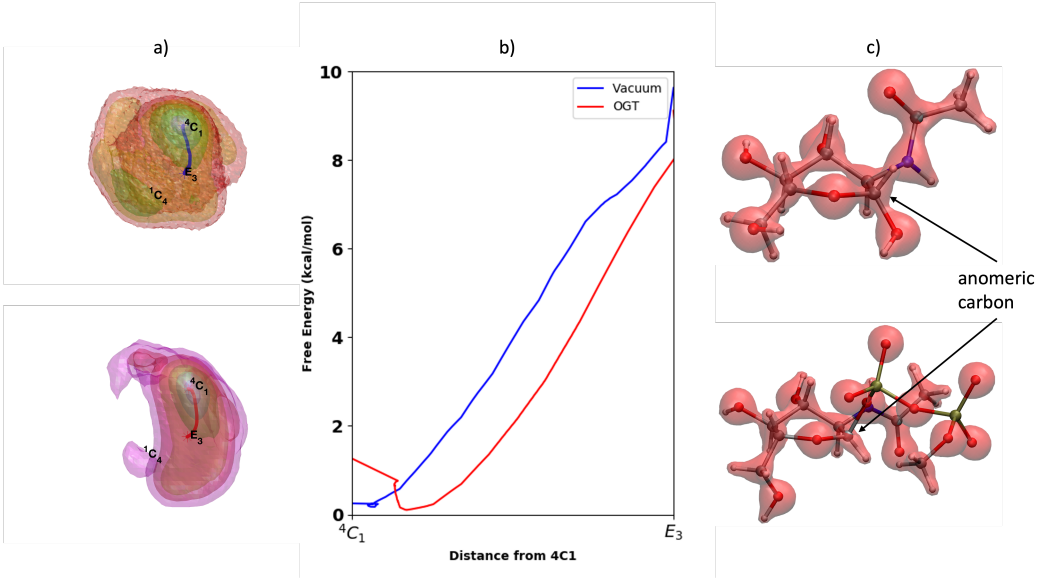


Figure 8.4: Comparison of GlcNAc in vacuum versus in OGT a) MEP between $^4C_1$ and E_3 plotted onto 3D FEV with vacuum MEP in blue and OGT MEP in red b) 1D plot of MEPs as distance from $^4C_1$ versus free energy c) electron density of GlcNAc in vacuum (top) and in OGT (bottom) with anomeric carbon highlighted.

Looking at the FEVs in Figure 8.4a, it is clear that in vacuum there is an iso-surface at 12 kcal/mol that connects the two chair conformers $^4C_1$ and $^1C_4$, while in OGT not even the 20 kcal/mol iso-surface connects them. In addition, the 1,4, and 8 kcal/mol iso-surfaces appear concentrated around the northern hemisphere of $^4C_1$, which constrains GlcNAc to the $^4C_1$ and E_3 transition when it is in OGT. This is likely due to the spatial and electrostatic constraints present within the OGT reaction cavity.

Now looking at the 1D plot of MEPs in Fig 8.4b, we can see that the $^4C_1$ conformer is actually less stable in OGT, since it has 1.02 kcal/mol greater free energy than in vacuum. During the transition from $^4C_1$ to E_3, being in OGT actually lowers the path energy for GlcNAc. The E_3 conformer in OGT also has a free energy 1.63 kcal/mol less than in vacuum.

If we look at possible explanations for this, we can take a look at the electron

density given in Fig 8.4c which shows that, at the anomeric carbon (C5), there is significantly less electron density when GlcNAc is in OGT versus when it is in vacuum. In addition, there is also a decrease in the electron density of the O5-C1 bond, which may be the cause of the instability which leads to the cleaving of the UDP from the GlcNAc during the first stage of the glycotransferase reaction.

This chapter presented results from a QM/MM free energy simulation of an enzyme and it's substrate, demonstrating the FEARCF library's applicability to molecular systems of this type and theoretical model.

Chapter 9

Conclusions

In the introduction of this work, the aim was stated to 'demonstrate how the graph theory inspired OOP design of the FEARCF software library is capable of performing multi-dimensional free energy simulations for various molecular systems at several levels of accuracy'. This aim has been accomplished by firstly illustrating how representing reaction coordinates as graphs is the basis for an analogous construction within an OOP paradigm.

This paradigm enables efficient and intuitive reuse of code which in turn allows for the use of more complex reaction coordinate sets. OOP also makes implementing new reaction coordinate types easier and less labour intensive. The packaging of this code into a software library, with a well defined interface, is what allows for free energy simulations to be run at various levels of theory. This was proven by compilation in both the CHARMM and NWChem MD software packages, which enabled the generation of free energy volumes using classical, *ab initio*, and QM/MM theory.

The free energy volumes produced for water illustrated the advantages of the multidimensional free energy approach as well as using high level quantum theory as opposed to parameterised classical models. The free energy volumes produced for glucose showed that the new OOP approach produces equivalent results as the previous procedural implementation of FEARCF, but with a simpler and more efficient coding approach. Finally, the free energy volumes produced for the GlcNAc and OGT enzyme-substrate complex demonstrated FEARCF's applicability to more complex systems, such those with reaction dynamics whose mechanisms can be studied with free energy.

Further development of the methods presented in this work is certainly possible. The parallisation of parts of the FEARCF library is one possibility, thanks to its graph-based nature providing a potential guide. This could be done by calculating the number of branches emanating from each rooted tree, the path length between each reaction coordinate node and atom node, and then allocating processor jobs accordingly. Additional reaction coordinate

types may also be added to the library, using previously implemented types and methods to save on the amount of additional code needing to be written. Within the current capabilities of the library, the used of 3D vectors as reaction coordinates has not fully been explored yet. Preliminary results, however, are promising and represent a type of free energy volume that could easily be mapped onto actual molecular structures. This would provide more intuitive means in which to draw conclusions about the molecular system from the free energy. This is something which, to our knowledge, has not been explored before in the realm of multidimensional free energy calculations.

Bibliography

[1] Kenneth G. Wilson. Grand challenges to computational science. *Future Generation Computer Systems*, 5(2):171–189, 1989.

[2] The Royal Swedish Academy of Sciences. The nobel prize in chemistry 1998, 1998.

[3] The Royal Swedish Academy of Sciences. The nobel prize in chemistry 2013, 2013.

[4] Margherita Maiuri, Marco Garavelli, and Giulio Cerullo. Ultrafast spectroscopy: State of the art and open challenges. *Journal of the American Chemical Society*, 142(1):3–15, 2020.

[5] Dexter B Northrop. Steady-state analysis of kinetic isotope effects in enzymic reactions. *Biochemistry*, 14(12):2644–2651, 1975.

[6] Andreas Vitalis and Rohit V. Pappu. *Chapter 3 Methods for Monte Carlo Simulations of Biomacromolecules*, volume 5, pages 49–76. Elsevier, 2009.

[7] Michael P Allen and Dominic J Tildesley. *Computer simulation of liquids*. Oxford university press, 2017.

[8] Christopher J Cramer. *Essentials of computational chemistry: theories and models*. John Wiley & Sons, 2013.

[9] Hans Martin Senn and Walter Thiel. Qm/mm methods for biomolecular systems. *Angewandte Chemie International Edition*, 48(7):1198–1229, 2009.

[10] D.A. McQuarrie. *Statistical Mechanics*. University Science Books, 2000.

[11] Ruth Nussinov and Chung-Jung Tsai. Free energy diagrams for protein function. *Chemistry & Biology*, 21(3):311–318, 2014.

[12] G. M. Torrie and J. P. Valleau. Nonphysical sampling distributions in monte carlo free-energy estimation: Umbrella sampling. *Journal of Computational Physics*, 23(2):187–199, 1977.

[13] Mihaly Mezei. Adaptive umbrella sampling: Self-consistent determination of the non-boltzmann bias. *Journal of Computational Physics*, 68:237–248, 1987.

[14] Alessandro Laio and Michele Parrinello. Escaping free-energy minima. *Proceedings of the National Academy of Science*, 99:12562, 2002.

[15] Kevin J Naidoo and JW Brady. Calculation of the ramachandran potential of mean force for a disaccharide in aqueous solution. *Journal of the American Chemical Society*, 121(10):2244–2252, 1999.

[16] Christopher B. Barnett and Kevin J. Naidoo. Free energies from adaptive reaction coordinate forces (fearcf): an application to ring puckering. *Molecular Physics*, 107(8-12):1243–1250, 2009.

[17] Shankar Kumar, John M. Rosenberg, Djamal Bouzida, Robert H. Swendsen, and Peter A. Kollman. The weighted histogram analysis method for free energy calculations on biomolecules. i. the method. *Journal of Computational Chemistry*, 13, 1992.

[18] J. Strümpfer and K. J. Naidoo. Computing free energy hypersurfaces for anisotropic intermolecular associations. *J Comput Chem*, 31(2):308–16, 2010. 1096-987x Strümpfer, Johan Naidoo, Kevin J Journal Article United States J Comput Chem. 2010 Jan 30;31(2):308-16. doi: 10.1002/jcc.21317.

[19] Ed Akin. *Object-Oriented Programming via Fortran 90/95*. Cambridge University Press, 2003.

[20] Nenad Trinajstic. *Chemical graph theory*. CRC press, 2018.

[21] F. Franks. *The Solvent Properties of Water*, pages 1–54. Springer US, Boston, MA, 1973.

[22] Bimal Krishna Banik and Biswa Mohan Sahoo. *13 - Reactions in water: Synthesis of biologically active compounds*, pages 491–521. Elsevier, 2020.

[23] JK Gregory, DC Clary, K Liu, MG Brown, and RJ Saykally. The water dipole moment in water clusters. *Science*, 275(5301):814–817, Feb 1997.

[24] Roberto Scipioni, Diedrich A. Schmidt, and Mauro Boero. A first principles investigation of water dipole moment in a defective continuous hydrogen bond network. *The Journal of Chemical Physics*, 130(2):024502, 2022/05/15 2009.

[25] Emiliano Brini, Christopher J. Fennell, Marivi Fernandez-Serra, Barbara Hribar-Lee, Miha Lukšič, and Ken A. Dill. How water's properties are encoded in its molecular structure and energies. *Chemical Reviews*, 117(19):12385–12414, 10 2017.

[26] P M Wiggins. Role of water in some biological processes. *Microbiological reviews*, 54(4):432–449, 12 1990.

[27] William Stillwell. *Chapter 1 - Introduction to Biological Membranes*, pages 3–15. Elsevier, 2016.

[28] Philip L. Yeagle. *Chapter 8 - Lipid Dynamics in Membranes*, pages 155–188. Academic Press, Boston, 2016.

[29] Biman Bagchi. *Biological water*, pages 81–96. Cambridge Molecular Science. Cambridge University Press, 2013.

[30] Elliott J Stollar and David P Smith. Uncovering protein structure. *Essays in Biochemistry*, 64(4):649–680, 5/15/2022 2020.

[31] Tanya M Raschke. Water structure and interactions with protein surfaces. *Current Opinion in Structural Biology*, 16(2):152–159, 2006.

[32] Naoko Akiya and Phillip E. Savage. Roles of water for chemical reactions in high-temperature water. *Chemical Reviews*, 102(8):2725–2750, 08 2002.

[33] Dmitry Yu. Murzin. *Chapter 11. Reactions involving water: hydration, dehydration, etherification, hydrolysis, and esterification*, pages 281–292. De Gruyter, 2015.

[34] Helen C. Hailes. Reaction solvent selection: The potential of water as a solvent for organic transformations. *Organic Process Research & Development*, 11(1):114–120, 01 2007.

[35] A Lovegrove, C H Edwards, I De Noni, H Patel, S N El, T Grassby, C Zielke, M Ulmius, L Nilsson, P J Butterworth, P R Ellis, and P R Shewry. Role of polysaccharides in food, digestion, and health. *Critical reviews in food science and nutrition*, 57(2):237–253, 01 2017.

[36] James N. BeMiller. *1 - Monosaccharides*, pages 1–23. AACC International Press, 2019.

[37] Davide Alocci, Marie Ghraichy, Elena Barletta, Alessandra Gastaldello, Julien Mariethoz, and Frederique Lisacek. Understanding the glycome: an interactive view of glycosylation from glycocompositions to glycoepitopes. *Glycobiology*, 28(6):349–362, 5/15/2022 2018.

[38] Zachary Shriver, S. Raguram, and Ram Sasisekharan. Glycomics: a pathway to a class of new and improved therapeutics. *Nature Reviews Drug Discovery*, 3(10):863–873, 2004.

[39] Peter H. Seeberger. Monosaccharide diversity, 2015.

[40] Momcilo Miljkovic. *Carbohydrates: synthesis, mechanisms, and stereoelectronic effects*. Springer Science & Business Media, 2009.

[41] T. Bruce Grindley. *Structure and Conformation of Carbohydrates*, pages 3–55. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[42] Andrei R. Ionescu, Attila Bérces, Marek Z. Zgierski, Dennis M. Whitfield, and Tomoo Nukada. Conformational pathways of saturated six-membered rings. a static and dynamical density functional study. *The Journal of Physical Chemistry A*, 109(36):8096–8105, 09 2005.

[43] 1916-2003. Laidler, Keith J. (Keith James). *Chemical kinetics*. Harper & Row, New York, 3rd ed. edition, 1987.

[44] Stephen J. Klippenstein, Vijay S. Pande, and Donald G. Truhlar. Chemical kinetics and mechanisms of complex systems: A perspective on recent theoretical advances. *Journal of the American Chemical Society*, 136(2):528–546, 01 2014.

[45] Richard I. (1951-). Masel. *Chemical kinetics and catalysis*. John Wiley & Sons, New York [etc.], 2005.

[46] Fritz 1868-1934. Haber. *Thermodynamik technischer Gasreaktionen; Sieben Vorlesungen,*. R. Oldenburg, München, 1905.

[47] Michael Reese, Cory Marquart, Mahdi Malmali, Kevin Wagner, Eric Buchanan, Alon McCormick, and Edward L. Cussler. Performance of a small-scale haber process. *Industrial & Engineering Chemistry Research*, 55(13), 2016.

[48] Trevor 1944 Palmer. *Understanding enzymes*. Ellis Horwood series in biological chemistry and biotechnology. Ellis Horwood ;, Chichester, West Sussex; New York, 2nd ed. edition, 1985.

[49] Lizbeth Hedstrom. Enzyme specificity and selectivity. In *Encyclopedia of Life Science*. John Wiley & Sons, Ltd, 02 2010.

[50] Duncan E Scott, Alessio Ciulli, and Chris Abell. Coenzyme biosynthesis: enzyme mechanism, structure and inhibition. *Natural product reports*, 24(5):1009–26, 2007.

[51] R. A. Freedland and Stephanie Briggs. *Regulation of enzyme activity*, pages 11–15. Springer Netherlands, Dordrecht, 1977.

[52] Florencia Rago, Daniel Saltzberg, Karen N. Allen, and Dean R. Tolan. Enzyme substrate specificity conferred by distinct conformational pathways. *Journal of the American Chemical Society*, 137(43):13876–13886, 11 2015.

[53] Chris 1943 Woodford and Chris 1950 Phillips. *Numerical methods with worked examples : Matlab edition*. Springer Science+Business Media, Dordrecht ;, 2nd ed. edition, 2012.

[54] K Vanommeslaeghe, E Hatcher, C Acharya, S Kundu, S Zhong, J Shim, E Darian, O Guvench, P Lopes, I Vorobyov, and Jr Mackerell, A D. Charmm general force field: A force field for drug-like molecules compatible with the charmm all-atom additive biological force fields. *Journal of computational chemistry*, 31(4):671–690, 03 2010.

[55] Kevin. Burrage. *Parallel and sequential methods for ordinary differential equations*. Oxford science publications. Clarendon Press ;, Oxford; New York, 1995.

[56] Loup Verlet. Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Physical Review*, 159(1):98–103, 07 1967.

[57] Benedict J. Leimkuhler, Sebastian Reich, and Robert D. Skeel. *Integration Methods for Molecular Dynamics*, pages 161–185. Springer New York, New York, NY, 1996.

[58] P. Schofield. Computer simulation studies of the liquid state. *Computer Physics Communications*, 5(1):17–23, 1973.

[59] Philippe H. Hünenberger. *Thermostat Algorithms for Molecular Dynamics Simulations*, pages 105–149. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[60] Giovanni Bussi, Davide Donadio, and Michele Parrinello. Canonical sampling through velocity rescaling. *The Journal of Chemical Physics*, 126(1):014101, 2022/05/18 2007.

[61] William G. Hoover, Anthony J. C. Ladd, and Bill Moran. High-strain-rate plastic flow studied via nonequilibrium molecular dynamics. *Physical Review Letters*, 48(26):1818–1820, 06 1982.

[62] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak. Molecular dynamics with coupling to an external bath. *The Journal of Chemical Physics*, 81(8):3684–3690, 2022/05/18 1984.

[63] D. J. Evans and B. L. Holian. The nose–hoover thermostat. *The Journal of Chemical Physics*, 83(8):4069–4074, 2022/05/18 1985.

[64] Ruslan L. Davidchack, Richard Handel, and M. V. Tretyakov. Langevin thermostat for rigid body dynamics. *The Journal of Chemical Physics*, 130(23):234101, 2022/05/18 2009.

[65] Henry F Schaefer III. *Quantum chemistry: the development of ab initio methods in molecular electronic structure theory*. Courier Corporation, 2012.

[66] Yusuf Atalay, Eric Paquet, and Herna L. Viktor. Computational methods for ab initio molecular dynamics. *Advances in Chemistry*, 2018:9839641, 2018.

[67] D. R. Hartree. The wave mechanics of an atom with a non-coulomb central field. part ii. some results and discussion. *Mathematical Proceedings of the Cambridge Philosophical Society*, 24(1):111–132, 1928.

[68] V. Fock. Näherungsmethode zur lösung des quantenmechanischen mehrkörperproblems. *Zeitschrift für Physik*, 61(1):126–148, 1930.

[69] J. C. Slater. Note on hartree's method. *Physical Review*, 35(2):210–211, 01 1930.

[70] J. C. Slater. The theory of complex spectra. *Physical Review*, 34(10):1293–1322, 11 1929.

[71] J. C. Slater. Atomic shielding constants. *Physical Review*, 36(1):57–64, 07 1930.

[72] S. F. Boys and Alfred Charles Egerton. Electronic wave functions - i. a general method of calculation for the stationary states of any molecular system. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 200(1063):542–554, 2022/05/18 1950.

[73] Peter M. W. Gill. *Molecular integrals Over Gaussian Basis Functions*, volume 25, pages 141–205. Academic Press, 1994.

[74] Jiří Čížek. On the correlation problem in atomic and molecular systems. calculation of wavefunction components in ursell-type expansion using quantum-field theoretical methods. *The Journal of Chemical Physics*, 45(11):4256–4266, 2022/10/06 1966.

[75] Chr. Møller and M. S. Plesset. Note on an approximation treatment for many-electron systems. *Physical Review*, 46(7):618–622, 10 1934.

[76] Christopher J Cramer. *Essentials of computational chemistry: theories and models*. John Wiley & Sons, 2013.

[77] Kieron Burke and Lucas Wagner. Dft in a nutshell. *International Journal of Quantum Chemistry*, 113:96–101, 2013.

[78] L. H. Thomas. The calculation of atomic fields. *Mathematical Proceedings of the Cambridge Philosophical Society*, 23(5):542–548, 1927.

[79] Enrico Fermi. Statistical method to determine some properties of atoms. *Rend. Accad. Naz. Lincei*, 6(602-607):5, 1927.

[80] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Physical Review*, 136(3B):B864–B871, 11 1964.

[81] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Physical Review*, 140(4A):A1133–A1138, 11 1965.

[82] John P. Perdew, J. A. Chevary, S. H. Vosko, Koblar A. Jackson, Mark R. Pederson, D. J. Singh, and Carlos Fiolhais. Atoms, molecules, solids, and surfaces: Applications of the generalized gradient approximation for exchange and correlation. *Physical Review B*, 46(11):6671–6687, 09 1992.

[83] J. Harris. Adiabatic-connection approach to kohn-sham theory. *Physical Review A*, 29(4):1648–1659, 04 1984.

[84] Gerrit Groenhof. *Introduction to QM/MM Simulations*, pages 43–66. Humana Press, Totowa, NJ, 2013.

[85] Martin J Field, Paul A Bash, and Martin Karplus. A combined quantum mechanical and molecular mechanical potential for molecular dynamics simulations. *Journal of Computational Chemistry*, 11(6):700–733, 1990.

[86] Dean M Philipp and Richard A Friesner. Mixed ab initio qm/mm modeling using frozen orbitals and tests with alanine dipeptide and tetrapeptide. *Journal of computational chemistry*, 20(14):1468–1494, 1999.

[87] Hai Lin and Donald G. Truhlar. Qm/mm: what have we learned, where are we, and where do we go from here? *Theoretical Chemistry Accounts*, 117(2):185, 2006.

[88] Jean-Philippe Ansermet and Sylvain D. Brechet, editors. *Foundations*, pages 1–2. Cambridge University Press, Cambridge, 2019.

[89] J. R. Taylor and S. L. L. J. R. Taylor. *Classical Mechanics*. G - Reference,Information and Interdisciplinary Subjects Series. University Science Books, 2005.

[90] James Clerk 1831-1879. Maxwell. *Theory of heat.* Greenwood Press, Westport, Conn., 3d ed. edition, 1970.

[91] Alfred Wehrl. General properties of entropy. *Reviews of Modern Physics*, 50(2):221–260, 04 1978.

[92] Hermann von Helmholtz. On the thermodynamics of chemical processes,. *Physical memoirs.*, 1(1):43–97, 1888.

[93] J. Willard Gibbs. *A method of geometrical representation of the thermodynamic properties of substances by means of surfaces.* Transaction of the Conneticut Academy, [New Haven], 1873.

[94] H. P. Lehmann, X. Fuentes-Arderiu, and L. F. Bertello. Glossary of terms in quantities and units in clinical chemistry (iupac-ifcc recommendations 1996). *Pure and Applied Chemistry*, 68(4):957–1000, 1996.

[95] Stephen M. Stigler. *The history of statistics : the measurement of uncertainty before 1900.* Belknap Press of Harvard University Press, Cambridge, Mass., 1986.

[96] Aidan Lyon. Why are normal distributions normal? *The British Journal for the Philosophy of Science*, 65(3):621–649, 2022/05/19 2014.

[97] Josiah Willard Gibbs. *Elementary principles in statistical mechanics : developed with especial reference to the rational foundation of thermodynamics.* Charles Scribner's Sons ; Edward Arnold, New York; London, 1902.

[98] Marc Baus and Carlos F. Tejero, editors. *Canonical Ensemble*, pages 89–119. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[99] Maurice D. Weir, Joel. Hass, and 1914-2006. Thomas, George B. Jr. (George Brinton). *Thomas' calculus.* Always Learning. Addison-Wesley, Boston, 12th ed. edition, 2010.

[100] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics.* Cambridge University Press, Cambridge, 2009.

[101] Yunus A. Çengel and Michael A. Boles. *Thermodynamics : an engineering approach.* McGraw-Hill Education, New York, eighth edition. edition, 2015.

[102] Robert A. Alberty. Use of legendre transforms in chemical thermodynamics (iupac technical report). *Pure and Applied Chemistry*, 73(8):1349–1380, 2001.

[103] J. L. Lagrange. *Mécanique analytique.* Number v. 1 in Mécanique analytique. Ve Courcier, 1811.

[104] L. N. Hand and J. D. Finch. *Analytical Mechanics.* Cambridge University Press, 1998.

[105] M G Calkin. *Lagrangian and Hamiltonian Mechanics.* WORLD SCIENTIFIC, 2022/05/19 1996.

[106] Trine Holst Sørensen, Nicolaj Cruys-Bagger, Kim Borch, and Peter Westh. Free energy diagram for the heterogeneous enzymatic

hydrolysis of glycosidic bonds in cellulose *. *Journal of Biological Chemistry*, 290(36):22203–22211, 2022/05/19 2015.

[107] Stanislav M. Avdoshenko and Dmitrii E. Makarov. Reaction coordinates and pathways of mechanochemical transformations. *The Journal of Physical Chemistry B*, 120(8):1537–1545, 03 2016.

[108] Best Robert B. and Hummer Gerhard. Reaction coordinates and rates from transition paths. *Proceedings of the National Academy of Sciences*, 102(19):6732–6737, 2022/05/19 2005.

[109] Lawrence R. Pratt and Dilip Asthagiri. *Potential Distribution Methods and Free Energy Models of Molecular Solutions*, pages 323–351. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[110] Emilia P. Barros, Benjamin Ries, Lennard Böselt, Candide Champion, and Sereina Riniker. Recent developments in multiscale free energy simulations. *Current Opinion in Structural Biology*, 72:55–62, 2022.

[111] Daan Frenkel and Berend Smit. Understanding molecular simulation : from algorithms to applications. 2nd ed. *Physics Today*, 50, 01 1996.

[112] Robert W. Zwanzig. High-temperature equation of state by a perturbation method. i. nonpolar gases. *The Journal of Chemical Physics*, 22(8):1420–1426, 2022/06/14 1954.

[113] Charles H Bennett. Efficient estimation of free energy differences from monte carlo data. *Journal of Computational Physics*, 22(2):245–268, 1976.

[114] David A. Pearlman and Peter A. Kollman. A new method for carrying out free energy perturbation calculations: Dynamically modified windows. *The Journal of Chemical Physics*, 90(4):2460–2470, 2022/05/17 1989.

[115] Alessandro Barducci, Giovanni Bussi, and Michele Parrinello. Well-tempered metadynamics: A smoothly converging and tunable free-energy method. *Physical Review Letters*, 100:020603, 2008.

[116] Thomas Huber, Andrew E. Torda, and Wilfred F. van Gunsteren. Local elevation: A method for improving the searching properties of molecular dynamics simulation. *Journal of Computer-Aided Molecular Design*, 8(6):695–708, 1994.

[117] Fugao Wang and D. P. Landau. Efficient, multiple-range random walk algorithm to calculate the density of states. *Physical Review Letters*, 86(10):2050–2053, 03 2001.

[118] Arak Mathai. *Jacobians of Matrix Transformations and Functions of Matrix Argument*. Springer, New York, NY, 01 1997.

[119] Herman H. Goldstine. *The Computer from Pascal to von Neumann*. Princeton University Press SN - 9781400820139, 2008.

[120] M. V. Wilkes, editor. *The EDSAC Computer*, 1951.

[121] Thomas Sterling, Matthew Anderson, and Maciej Brodowicz. *Chapter 10 - Libraries*, pages 313–345. Morgan Kaufmann, Boston, 2018.

[122] J. Backus. The history of fortran i, ii, and iii. *IEEE Annals of the History of Computing*, 20(4):68–78, 1998.

[123] Michael. Metcalf and John Ker. Reid. *Fortran 90/95 explained*. Oxford University Press, Oxford ;, 2nd ed. edition, 1999.

[124] J. R. Levine and B. A. John R Levine. *Linkers and Loaders*. Operating Systems Series. Elsevier Science, 2000.

[125] Richard John Anthony. *Chapter 5 - The Architecture View*, pages 277–382. Morgan Kaufmann, Boston, 2016.

[126] Kristen Nygaard and Ole-Johan Dahl. *The development of the SIMULA languages*, pages 439–480. Association for Computing Machinery, 1978.

[127] John McCarthy and Michael I. Levin. Lisp 1.5 programmer's manual. Cambridge, Mass., 1979.

[128] R. Khurana. *Object Oriented Programming With C++, 1E*. Vikas Publishing House Pvt Limited, 2009.

[129] Michael. Metcalf, John Ker. Reid, and Malcolm 1957 Cohen. *Fortran 95/2003 explained*. Numerical mathematics and scientific computation. Oxford University Press, Oxford ;, 2004.

[130] B. R. Brooks, 3rd Brooks, C. L., Jr. Mackerell, A. D., L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R. W. Pastor, C. B. Post, J. Z. Pu, M. Schaefer, B. Tidor, R. M. Venable, H. L. Woodcock, X. Wu, W. Yang, D. M. York, and M. Karplus. Charmm: the biomolecular simulation program. *Journal of computational chemistry*, 30(10):1545–1614, 2009. 19444816[pmid] PMC2810661[pmcid].

[131] E. Aprà, E. J. Bylaska, W. A. de Jong, N. Govind, K. Kowalski, T. P. Straatsma, M. Valiev, H. J. J. van Dam, Y. Alexeev, J. Anchell, V. Anisimov, F. W. Aquino, R. Atta-Fynn, J. Autschbach, N. P. Bauman, J. C. Becca, D. E. Bernholdt, K. Bhaskaran-Nair, S. Bogatko, P. Borowski, J. Boschen, J. Brabec, A. Bruner, E. Cauët, Y. Chen, G. N. Chuev, C. J. Cramer, J. Daily, M. J. O. Deegan, T. H. Dunning, M. Dupuis, K. G. Dyall, G. I. Fann, S. A. Fischer,

A. Fonari, H. Früchtl, L. Gagliardi, J. Garza, N. Gawande, S. Ghosh, K. Glaesemann, A. W. Götz, J. Hammond, V. Helms, E. D. Hermes, K. Hirao, S. Hirata, M. Jacquelin, L. Jensen, B. G. Johnson, H. Jónsson, R. A. Kendall, M. Klemm, R. Kobayashi, V. Konkov, S. Krishnamoorthy, M. Krishnan, Z. Lin, R. D. Lins, R. J. Littlefield, A. J. Logsdail, K. Lopata, W. Ma, A. V. Marenich, J. Martin del Campo, D. Mejia-Rodriguez, J. E. Moore, J. M. Mullin, T. Nakajima, D. R. Nascimento, J. A. Nichols, P. J. Nichols, J. Nieplocha, A. Otero-de-la Roza, B. Palmer, A. Panyala, T. Pirojsirikul, B. Peng, R. Peverati, J. Pittner, L. Pollack, R. M. Richard, P. Sadayappan, G. C. Schatz, W. A. Shelton, D. W. Silverstein, D. M. A. Smith, T. A. Soares, D. Song, M. Swart, H. L. Taylor, G. S. Thomas, V. Tipparaju, D. G. Truhlar, K. Tsemekhman, T. Van Voorhis, Á. Vázquez-Mayagoitia, P. Verma, O. Villa, A. Vishnu, K. D. Vogiatzis, D. Wang, J. H. Weare, M. J. Williamson, T. L. Windus, K. Woliński, A. T. Wong, Q. Wu, C. Yang, Q. Yu, M. Zacharias, Z. Zhang, Y. Zhao, and R. J. Harrison. Nwchem: Past, present, and future. *The Journal of Chemical Physics*, 152(18):184102, 2022/05/19 2020.

[132] Norman Biggs, E Keith Lloyd, and Robin J Wilson. *Graph Theory, 1736-1936*. Oxford University Press, 1986.

[133] Leslie R Foulds. *Graph theory applications*. Springer Science & Business Media, 2012.

[134] John Adrian Bondy and Uppaluri Siva Ramachandra Murty. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[135] A. Cayley. Xxviii. on the theory of the analytical forms called trees. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 13(85):172–176, 1857.

[136] undefined Tait. 10. remarks on the previous communication. *Proceedings of the Royal Society of Edinburgh*, 10:729–729, 1880.

[137] K. Appel and W. Haken. Every planar map is four colorable. part i: Discharging. *Illinois Journal of Mathematics*, 21(3):429–490, 9 1977.

[138] Gary. Chartrand. *Introductory graph theory*. Dover, New York, unabridged and corr. edition, 1985.

[139] Cayley. Lvii. on the mathematical theory of isomers. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 47(314):444–447, 1874.

[140] F Hermann. Ueber das problem, die anzahl der isomeren paraffine der formel cnh2n+ 2 zu bestimmen. *Berichte der deutschen chemischen Gesellschaft*, 13(1):792–792, 1880.

[141] Henry R. Henze and Charles M. Blair. The number of isomeric hydrocarbons of the methane series. *Journal of the American Chemical Society*, 53(8):3077–3085, 08 1931.

[142] Iva Pritišanac, Matteo T. Degiacomi, T. Reid Alderson, Marta G. Carneiro, Eiso Ab, Gregg Siegal, and Andrew J. Baldwin. Automatic assignment of methyl-nmr spectra of supramolecular machines using graph theory. *Journal of the American Chemical Society*, 139(28):9523–9533, 2017.

[143] C. S. Johnson. Diffusion ordered nuclear magnetic resonance spectroscopy: principles and applications. *Progress in Nuclear Magnetic Resonance Spectroscopy*, 34(3):203–256, 1999.

[144] Stefan Schütz and Remco Sprangers. Methyl trosy spectroscopy: A versatile nmr approach to study challenging biological systems. *Progress in Nuclear Magnetic Resonance Spectroscopy*, 116:56–84, 2020.

[145] Fa-An Chao, Jonggul Kim, Youlin Xia, Michael Milligan, Nancy Rowe, and Gianluigi Veglia. Flamengo 2.0: An enhanced fuzzy logic algorithm for structure-based assignment of methyl group resonances. *Journal of Magnetic Resonance*, 245:17–23, 2014.

[146] Yingqi Xu and Stephen Matthews. Map-xsii: an improved program for the automatic assignment of methyl resonances in large proteins. *J Biomol NMR*, 55(2):179–187, Feb 2013.

[147] Ramkumar Rajamani, Kevin J Naidoo, and Jiali Gao. Implementation of an adaptive umbrella sampling method for the calculation of multidimensional potential of mean force of chemical reactions in solution. *J Comput Chem*, 24(14):1775–1781, Nov 2003.

[148] Kevin Naidoo. Fearcf a multidimensional free energy method for investigating conformational landscapes and chemical reaction mechanisms. *Science China-Chemistry*, 54:1962, 2011.

[149] William C. Swope and David M. Ferguson. Alternative expressions for energies and forces due to angle bending and torsional energy. *J. Comput. Chem.*, 13

[150] Robert Tuzun, DW Noid, and Bobby Sumpter. Computation of internal coordinates, derivatives, and gradient expressions: Torsion and improper torsion. *Journal of Computational Chemistry*, 21:553–561, 05 2000.

[151] D. Cremer and J. A. Pople. General definition of ring puckering coordinates. *Journal of the American Chemical Society*, 97(6):1354–1358, 03 1975.

[152] Herbert L. Strauss and Herbert M. Pickett. Conformational structure, energy, and inversion rates of cyclohexane and some related oxanes. *Journal of the American Chemical Society*, 92(25):7281–7290, 12 1970.

[153] Anthony D. Hill and Peter J. Reilly. Puckering coordinates of monocyclic rings by triangular decomposition. *Journal of Chemical Information and Modeling*, 47(3):1031–1035, 05 2007.

[154] Jarek Nieplocha and Robert Harrison. Shared memory programming in metacomputing environments: The global array approach. *The Journal of Supercomputing*, 11(2):119–136, 1997.

[155] Mathias Van Thiel, Edwin D. Becker, and George C. Pimentel. Infrared studies of hydrogen bonding of water by the matrix isolation technique. *The Journal of Chemical Physics*, 27(2):486–490, 1957.

[156] William L. Jorgensen, Jayaraman Chandrasekhar, Jeffry D. Madura, Roger W. Impey, and Michael L. Klein. Comparison of simple potential functions for simulating liquid water. *The Journal of Chemical Physics*, 79(2):926–935, 2022/05/20 1983.

[157] Pekka Mark and Lennart Nilsson. Structure and dynamics of the tip3p, spc, and spc/e water models at 298 k. *The Journal of Physical Chemistry A*, 105(43):9954–9960, 11 2001.

[158] Michael W. Mahoney and William L. Jorgensen. A five-site model for liquid water and the reproduction of the density anomaly by rigid, nonpolarizable potential functions. *The Journal of Chemical Physics*, 112(20):8910–8922, 2022/05/20 2000.

[159] Yijin Mao and Yuwen Zhang. Thermal conductivity, shear viscosity and specific heat of rigid water models. *Chemical Physics Letters*, 542:37–41, 2012.

[160] M. A. Dvorak, R. S. Ford, R. D. Suenram, F. J. Lovas, and K. R. Leopold. van der waals vs. covalent bonding: microwave characterization of a structurally intermediate case. *Journal of the American Chemical Society*, 114(1):108–115, 01 1992.

[161] Joshua A. Plumley and J. J. Dannenberg. A comparison of the behavior of functional/basis set combinations for hydrogen-bonding in the water dimer with emphasis on basis set superposition error. *Journal of computational chemistry*, 32(8):1519–1527, 2011. 21328398[pmid] PMC3073166[pmcid].

[162] Blithe E. Rocher-Casterline, Lee C. Ch'ng, Andrew K. Mollner, and Hanna Reisler. Communication: Determination of the bond dissociation energy (d0) of the water dimer, (h2o)2, by velocity map imaging. *The Journal of Chemical Physics*, 134(21):211101, 2011.

[163] Cui Zhang and Giulia Galli. Dipolar correlations in liquid water. *The Journal of Chemical Physics*, 141(8):084504, 2022/05/20 2014.

[164] Graham. Williams. Use of the dipole correlation function in dielectric relaxation. *Chemical Reviews*, 72(1):55–69, 02 1972.

[165] DW James and RF Armishaw. Structure of aqueous solutions: Infrared spectra of the water librational mode in solutions of monovalent halides. *Australian Journal of Chemistry*, 28(6):1179–1186, 1975.

[166] Christopher B. Barnett and Kevin J. Naidoo. Stereoelectronic and solvation effects determine hydroxymethyl conformational preferences in monosaccharides. *The Journal of Physical Chemistry B*, 112(48):15450–15459, 12 2008.

[167] Lourdes A. Alves, João B. Almeida e Silva, and Marco Giulietti. Solubility of d-glucose in water and ethanol/water mixtures. *Journal of Chemical & Engineering Data*, 52(6):2166–2170, 11 2007.

[168] Robert S Shallenberger. *Advanced sugar chemistry: principles of sugar stereochemistry*. AVI Publishing Company, 1982.

[169] William A. Lubas, David W. Frank, Michael Krause, and John A. Hanover. O-linked glcnac transferase is a conserved nucleocytoplasmic protein containing tetratricopeptide repeats*. *Journal of Biological Chemistry*, 272(14):9316–9324, 1997.

[170] Andrew J Clarke, Ramon Hurtado-Guerrero, Shalini Pathak, Alexander W Schüttelkopf, Vladimir Borodkin, Sharon M Shepherd, Adel F M Ibrahim, and Daan M F van Aalten. Structural insights into mechanism and specificity of o-glcnac transferase. *The EMBO Journal*, 27(20):2780–2788, 2022/05/20 2008.

[171] Eun Ju Kim, Dae Ook Kang, Dona C. Love, and John A. Hanover. Enzymatic characterization of o-glcnacase isoforms using a fluorogenic glcnac substrate. *Carbohydrate Research*, 341(8):971–982, 2006.

[172] Michael B. Lazarus, Yunsun Nam, Jiaoyang Jiang, Piotr Sliz, and Suzanne Walker. Structure of human o-glcnac transferase and its complex with a peptide substrate. *Nature*, 469(7331):564–567, 2011.

[173] Kevin J. Naidoo, Tomás Bruce-Chwatt, Tharindu Senapathi, and Malcolm Hillebrand. Multidimensional free energy and accelerated quantum library methods provide a gateway to glycoenzyme conformational, electronic, and reaction mechanisms. *Accounts of Chemical Research*, 54(22):4120–4130, 2021.

[174] Xiaoyong Yang, Pat P. Ongusaha, Philip D. Miles, Joyce C. Havstad, Fengxue Zhang, W. Venus So, Jeffrey E. Kudlow, Robert H. Michell,

Jerrold M. Olefsky, Seth J. Field, and Ronald M. Evans. Phosphoinositide signalling links o-glcnac transferase to insulin resistance. *Nature*, 451(7181):964–969, 2008.

[175] Jahanshah Ashkani and Kevin J. Naidoo. Glycosyltransferase gene expression profiles classify cancer types and propose prognostic subtypes. *Scientific Reports*, 6(1):26451, 2016.

[176] Karl A Wilkinson, Paul Sherwood, Martyn F Guest, and Kevin J Naidoo. Acceleration of the gamess-uk electronic structure package on graphical processing units. *Journal of computational chemistry*, 32(10):2313–2318, 2011.

[177] C Alicia Renison, Kyle D Fernandes, and Kevin J Naidoo. Quantum supercharger library: Hyper-parallel integral derivatives algorithms for ab initio qm/mm dynamics. *Journal of Computational Chemistry*, 36(18):1410–1419, 2015.

[178] Martyn F. Guest *, Ian J. Bush, Huub J. J. Van Dam, Paul Sherwood, Jens M. H. Thomas, Joop H. Van Lenthe, Remco W. A. Havenith, and John Kendrick. The gamess-uk electronic structure package: algorithms, developments and applications. *Molecular Physics*, 103(6-8):719–747, 03 2005.

[179] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, G. A. Petersson, H. Nakatsuji, X. Li, M. Caricato, A. V. Marenich, J. Bloino, B. G. Janesko, R. Gomperts, B. Mennucci, H. P. Hratchian, J. V. Ortiz, A. F. Izmaylov, J. L. Sonnenberg, Williams, F. Ding, F. Lipparini, F. Egidi, J. Goings, B. Peng, A. Petrone, T. Henderson, D. Ranasinghe, V. G. Zakrzewski, J. Gao, N. Rega, G. Zheng, W. Liang, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, K. Throssell, J. A. Montgomery Jr., J. E. Peralta, F. Ogliaro, M. J. Bearpark, J. J. Heyd, E. N. Brothers, K. N. Kudin, V. N. Staroverov, T. A. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A. P. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, J. M. Millam, M. Klene, C. Adamo, R. Cammi, J. W. Ochterski, R. L. Martin, K. Morokuma, O. Farkas, J. B. Foresman, and D. J. Fox. Gaussian 16 rev. c.01, 2016.