A COMPARATIVE STUDY BETWEEN THE CUBIC SPLINE AND B-SPLINE INTERPOLATION METHODS IN FREE ENERGY CALCULATIONS

Dissertation presented to the

UNIVERSITY OF CAPE TOWN

In fulfilment of the requirements for the degree of

MASTER OF SCIENCE

by

Hikmet Emre Kaya

KYXHIK001

Supervisor: Professor Kevin J. Naidoo

SCIENTIFIC COMPUTING RESEARCH UNIT

Department of Chemistry

UNIVERSITY OF CAPE TOWN

JUNE 2019

University of Cape Town

The copyright of this thesis vests in the author. No quotation from it or information derived from it is to be published without full acknowledgement of the source. The thesis is to be used for private study or non-commercial research purposes only.

Published by the University of Cape Town (UCT) in terms of the non-exclusive license granted to UCT by the author.

University of Cape Town

ABSTRACT

A Comparative Study between Cubic Spline and B-Spline Interpolation Methods in Free Energy Calculations

Numerical methods are essential in computational science, as analytic calculations for large datasets are impractical. Using numerical methods, one can approximate the problem to solve it with basic arithmetic operations. Interpolation is a commonly-used method, inter alia, constructing the value of new data points within an interval of known data points. Furthermore, polynomial interpolation with a sufficiently high degree can make the data set differentiable. One consequence of using high-degree polynomials is the oscillatory behaviour towards the endpoints, also known as Runge's Phenomenon. Spline interpolation overcomes this obstacle by connecting the data points in a piecewise fashion. However, its complex formulation requires nested iterations in higher dimensions, which is time-consuming. In addition, the calculations have to be repeated for computing each partial derivative at the data point, leading to further slowdown. The B-spline interpolation is an alternative representation of the cubic spline method, where a spline interpolation at a point could be expressed as the linear combination of piecewise basis functions. It was proposed that implementing this new formulation can accelerate many scientific computing operations involving interpolation. Nevertheless, there is a lack of detailed comparison to back up this hypothesis, especially when it comes to computing the partial derivatives.

Among many scientific research fields, free energy calculations particularly stand out for their use of interpolation methods. Numerical interpolation was implemented in free energy methods for many purposes, from calculating intermediate energy states to deriving forces from free energy surfaces. The results of these calculations can provide insight into reaction mechanisms and their thermodynamic properties. The free energy methods include biased flat histogram methods, which are especially promising due to their ability to accurately construct free energy profiles at the rarely-visited regions of reaction spaces. Free Energies from Adaptive Reaction Coordinates (FEARCF) that was developed by Professor Kevin J. Naidoo has many advantages over the other flat histogram methods.

Because of its treatment of the atoms in reactions, FEARCF makes it easier to apply interpolation methods. It implements cubic spline interpolation to derive biasing forces from the free energy surface, driving the reaction towards regions with higher energy. A major drawback of the method is the slowdown experienced in higher dimensions due to the complicated nature of the cubic spline routine. If the routine is replaced by a more straightforward B-spline interpolation, sampling and generating free energy surfaces can be accelerated.

The dissertation aims to perform a comparative study between the cubic spline interpolation and B-spline interpolation methods. At first, data sets of analytic functions were used instead of numerical data to compare the accuracy and compute the percentage errors of both methods by taking the functions themselves as reference. These functions were used to evaluate the performances of the two methods at the endpoints, inflections points and regions with a steep gradient. Both interpolation methods generated identically approximated values with a percentage error below the threshold of 1%, although they both performed poorly at the endpoints and the points of inflection. Increasing the number of interpolation knots reduced the errors, however, it caused overfitting in the other regions. Although significant speed-up was not observed in the univariate interpolation, cubic spline suffered from a drastic slowdown in higher dimensions with up to 10^3 in 3D and 10^5 in 4D interpolations. The same results applied to the classical molecular dynamics simulations with FEARCF with a speed-up of up to 10^3 when B-spline interpolation was implemented. To conclude, the B-spline interpolation method can enhance the efficiency of the free energy calculations where cubic spline interpolation has been the currently-used method.

DECLARATION

I declare that this dissertation, titled A COMPARATIVE STUDY BETWEEN THE CUBIC SPLINE AND B-SPLINE INTERPOLATION METHODS IN FREE ENERGY CALCULATIONS, 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.

Hikmet Emre Kaya

ACKNOWLEDGEMENTS

I am thankful for receiving financial aid from the University of Cape Town (UCT); the South African Research Chair Initiative (SARChI) and the Computational Chemistry bursary fund.

I would like to offer my sincere gratitude to my supervisor, Professor Kevin J. Naidoo. He has provided extensive guidance with his academic experience and knowledge as well as his inspiring personality. Thank you for having the confidence in me even at the times when I had self-doubts.

I am also thankful to Dr Christopher Barnett for helping me to learn the fundamentals of computational science and encouraging me to become an inquisitive researcher.

Many thanks to my colleagues at the Scientific Computing Research Unit (SCRU): for giving constructive feedback, and sharing their knowledge and experience with me.

I would like to offer special thanks to Lydia Dreyer for taking care of the administrative duties that I always deemed intimidating. Thank you for helping me to get through the stressful registration and submission processes.

Thanks to the administrative staff of the Department of Chemistry for their friendliness and effective guidance.

I feel blessed and grateful to have the instant support of my family even though they are 12 hours away from Cape Town by plane.

Huge thanks to all my friends who have always been supportive and understanding during the last two years.

ABBREVIATIONS

A: Helmholtz Free Energy

AM-1: Austin Model 1

CHARMM: Chemistry at Harvard Macromolecular Mechanics

CHARMM-GUI: CHARMM Graphic User Interface

CPU: Central Processing Unit

CV: Collective Variable

FEARCF: Free Energies from Adaptive Reaction Coordinate Forces

FEP: Free Energy Perturbation

FS: Ferrenberg-Swendsen Method

G: Gibbs Free Energy

HPC: High Performance Scientific Computing

K: Kelvin

LE: Local Elevation

MC: Monte Carlo Methods

MD: Molecular Dynamics

MM: Molecular Mechanics

NS: Nanosecond

NVT: Canonical Ensemble

PCA: Principal Component Analysis

PES: Potential Energy Surface

PI-TI: Parameter Interpolated Thermodynamic Integration

PMF: Potential of Mean Forces

POLYFIT: Matlab's Polynomial Curve Fitting

QM: Quantum Mechanics

RMS: Root Mean Square

SCRU: Scientific Computing Research Unit at UCT

SPC: Simple Point Charge

TI: Thermodynamic Integration

TIP3P: Three Site Transferrable Intermolecular Potential

VD: Voronoi Diagram

VM: Vandermonde Matrix

WHAM: Weighted Histogram Analysis Method

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: NUMERICAL METHODS IN COMPUTER SIMULATIONS

1.1 COMPUTATIONAL SCIENCE

From chemistry and nuclear engineering, to environmental studies and seismology, many scientific fields require the development of numerical models and simulations to solve complex problems. Computational science has become increasingly important as a multidisciplinary area of study that can be applied to these fields and provide solutions through advanced computing techniques. Such techniques involve a large number of floating-point calculations, a large storage area, as well as frame works, architectures, and industry-level compilers, all of which are executed on high-performance computers.

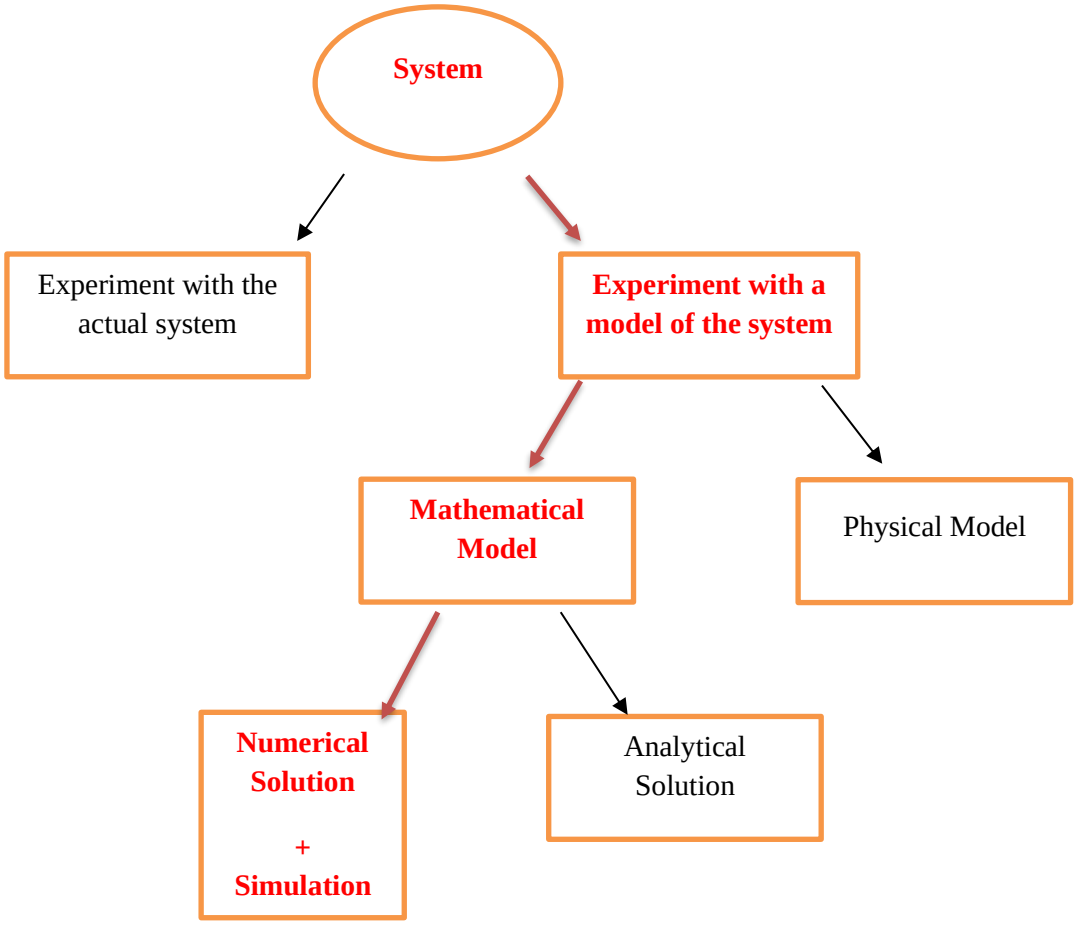


Figure 1.1: A flow diagram of the study of a system through experimental, theoretical and computational methods. The highlighted sections refer to the role of computational science.

The figure is adapted from the copyright-free original image
https://upload.wikimedia.org/wikipedia/commons/d/d6/Ways_to_study_a_system.png

High-performance computers stand out for interlinked architectures that consist of many individual computers – called nodes – to make up a cluster. The main purpose of a cluster is to solve a large problem through the collective work of individual nodes that can work together and communicate [1].

Many research groups make use of computational science by evaluating large datasets, simulations and modelling using High Performance Computing.

1.2 NUMERICAL METHODS IN COMPUTATIONAL SCIENCE

For large mathematical models, finding an analytical solution and performing direct calculations such as derivation and integration is time-consuming. Numerical methods are techniques that approximate these complex calculations to basic mathematical operations that can be solved in a shorter time frame by high-performance computers[2].

The most commonly used numerical methods are:

- Finding the roots of a complex equation
- Solving the system of linear algebraic equations
- Interpolation and regression analysis
- Numerical differentiation and integration
- Solutions of differential equations

A fundamental task of scientific calculations is to interpret a dataset by constructing a representative model that can predict the value at any point along the data space. Among the numerical methods listed above, interpolation is the most useful for this task as it creates connectivity between discretized data points.

The most preferred interpolation methods are those that provide continuity as well as the ability of continuous differentiation. Linear interpolation is fast and straightforward although it is not differentiable at the points where the gradient changes. Polynomial interpolation is preferable over linear interpolation since it is continuous, differentiable and easy to evaluate. Given n+1 data points, one can fit a polynomial of order n:

$$y = a_0 + a_{1x} + \cdots + a_n x^n \ (1.1)$$

The constants a_n can be found by solving the simultaneous linear equations. One major issue is that a single interpolation curve diverges unrealistically towards the edges of the interval if the degree of the polynomial is higher than three. Often referred to as Runge's Phenomenon [3], this issue makes the use of polynomials impractical. Spline interpolation is an alternative method that incorporates many small polynomials to interpolate between data points [4].

The accuracy of an interpolation method is depicted by the error it generates.

The terms used for measuring errors are true error and absolute relative true error. True error is the difference between the exact value and the approximated value.

$$True\ Error = Exact\ Value - Approximate\ Value \ (1.2)$$

True error is only a measure of the magnitude of difference between the two values; however, it does not indicate the impact of the error. An error of ~10^{-6} might seem to be small, but if the values were as small as ~10^{-4} – 10^{-5},[5] the impact of the error would still be significant. Relative true error is a better indication of the accuracy, shown as:

$$Relative\ True\ Error = \frac{True\ Error}{Exact\ Value} = \frac{Exact\ Value - Approximate\ Value}{Exact\ Value} \ (1.3)$$

1.3 COMPUTER SIMULATIONS OF REACTIONS

Investigating the reaction mechanism of chemical reactions is crucial in chemistry. Several experimental methods were previously developed to study reaction mechanisms. Nobel Prize-winner physical chemist Ahmed Zewail developed an ultrafast laser spectroscopy to study simple reactions with few molecules[6]. Another method called Kinetic Isotope Effect[7] replaced one of the atoms in the reactants by an isotope to observe the change in the reaction rate. However, such methods did not have the capacity to study the formation of a new chemical bond or the breaking of an already existing bond in a time scale of nanoseconds to femtoseconds.

More efficient methods were sought after to investigate reaction mechanisms based on statistical mechanics, where free energy and thermodynamic properties were calculated from the positions the atoms.

At the centre of these calculations lies the free energy expressed as a function of the molecular geometry. Characterizing the free energy of a chemical process can provide valuable information about the thermodynamic properties of that process.

Molecular Mechanics (MM) and Quantum Mechanics (QM) are used to generate PES based on the conformation and the intramolecular interactions.

Classical MM methods use force fields to predict the potential energy of a molecule as a function of its conformation. This allows prediction of atom velocities and positions, as well as equilibrium geometries, transition states and relative energies between different molecules. A force field is the collection of functions of the nuclear coordinates and parameters associated with those functions.

In the force field of a molecule, the total energy is expressed as a sum of Taylor expansion for every pair of bonded atoms (E_{str}) , and additional potential energy terms coming from bending (E_{bend}), torsional energy (E_{tors}), Van der Waals energy (E_{vdw}), electrostatics (E_{el}) and cross terms (E_{cross}).

$$E = E_{str} + E_{bend} + E_{tors} + E_{vdw} + E_{el} + E_{cross}$$

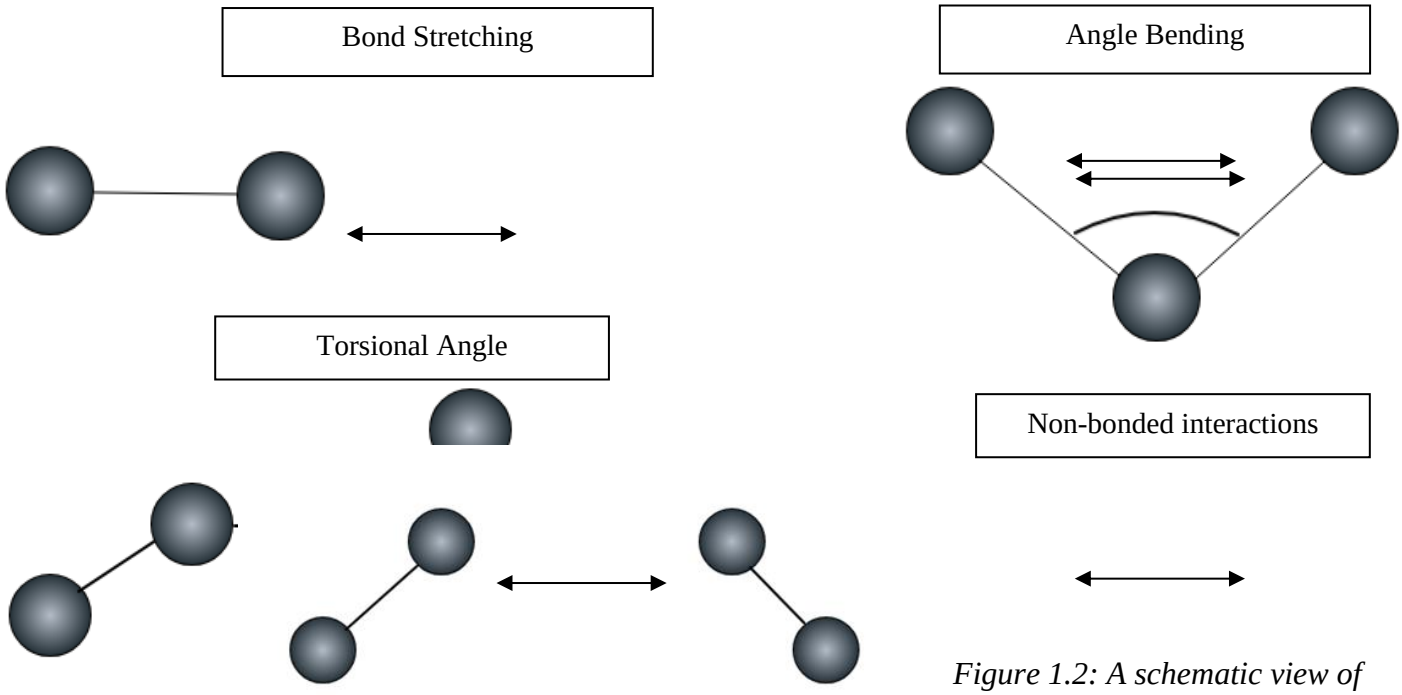


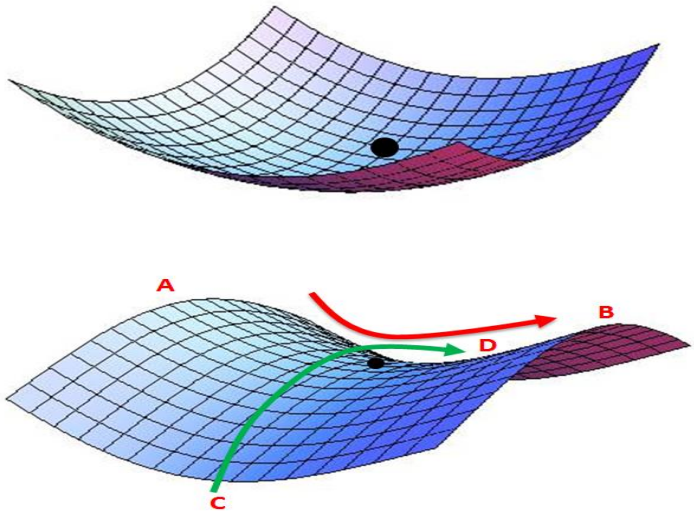
Figure 1.2: A schematic view of force field interactions

The classical MM force field is not sufficient to describe the behaviour of electrons. Because there are the making of new bonds and the breaking of existing bonds involved, a chemical reaction cannot be modelled correctly when the electron structure is neglected. Therefore, the explicit treatment of electrons is essential.

Various methods tackle this issue by either defining a bond-order based reactive force field[8] or using Quantum mechanical (QM) methods to incorporate nuclear and electronic interactions between the particles based on the Schrödinger equation[9]. These methods are used to define molecular properties such as the equilibrium geometries, vibrational frequencies, excited states, polarizability, activation barriers of reactions and the transition state structure.

1.4 MINIMUM ENERGY PATHWAY AND FREE ENERGY METHODS

The most critical points of a PES are minima, transition states and higher-order saddle points[10].



Copyright-free Original Image:

Figure 1.3: Potential Energy Surface and notable paths & phase points along the surface

Copyright-free Original Image:
https://upload.wikimedia.org/wikipedia/commons/4/42/Minima_and_Saddle_Point.png

The black dot on the upper image represents the minimum energy. The dot in the lower image is a transition state that is a maximum along the path C-D and a minimum along the path A-B. The path C-D containing the transition state is the minimum energy path, along which the ideal reaction mechanism can be constructed.

Free energy methods are useful for calculating the free energy difference between two states on the Potential Energy Surface. The applications of free energy provide a broad understanding of properties such as protein folding, solubility, and the ligand binding affinities. [11]

Free energy methods are classified as equilibrium and non-equilibrium methods. Equilibrium methods such as Thermodynamic Integration and Free Energy Perturbation apply equilibrium simulations of configurations step by step using a scaling parameter. On the other end, non – equilibrium approaches introduce a bias into the system, switching to subsequent values of the scaling parameter even if equilibrium has not been attained[12]**.**

Computing absolute free energies along the minimum energy path is also essential. For instance, the free energy profile of a transition state can be constructed to design a transition state analogue as an inhibitor of a protein-ligand interaction[13]. Sampling methods, among several computational and experimental methods, can be used to achieve free energy surfaces that represent the absolute free energy of the transition state.

Boltzmann sampling methods are used to compute the free energy profile of a system in equilibrium by averaging over a large number of microstates generated by Monte Carlo Methods or Molecular Dynamics[14] and assigning a probability to each microstate. Non-Boltzmann methods employ ensemble with the probabilities and energies proportional to the density of the states[15]. The first method implemented was called the Umbrella Sampling[16] with the consequent methods derived from it. These methods include the Local Elevation Method[17], Meta-dynamics[18], Adaptive Umbrella Sampling[19] and Free Energies from Adaptive Reaction Coordinate Forces[20]. Detailed information about Non-Boltzmann sampling methods were included in chapter 3 and 4.

1.5 NUMERICAL METHODS IN FREE ENERGY CALCULATIONS

Numerical approximations are commonly used in free energy calculations. The atoms and molecules in a chemical reaction can attain an infinite number of possible configurations.

Calculating the free energy difference between two states would require the computation of the energy values over all the possible configurations. For an infinitely large number of collections, finding an exact solution is nearly impossible due to the intricacy of the calculation. Therefore, only a finite set of configurations can be used, and the results are only representative estimates [21].

From integration to approximated molecular representations, numerical methods are encountered at various stages of free energy calculations.

An example of the use of numerical methods is the Thermodynamic Integration (TI) that performs a numerical integration using a finite number of samples. TI uses a coupling parameter λ that varies from 0 to 1 as the system progresses slowly from the initial state to the final state. During the integration, the trapezoid rule is used to obtain an approximate integral[22].

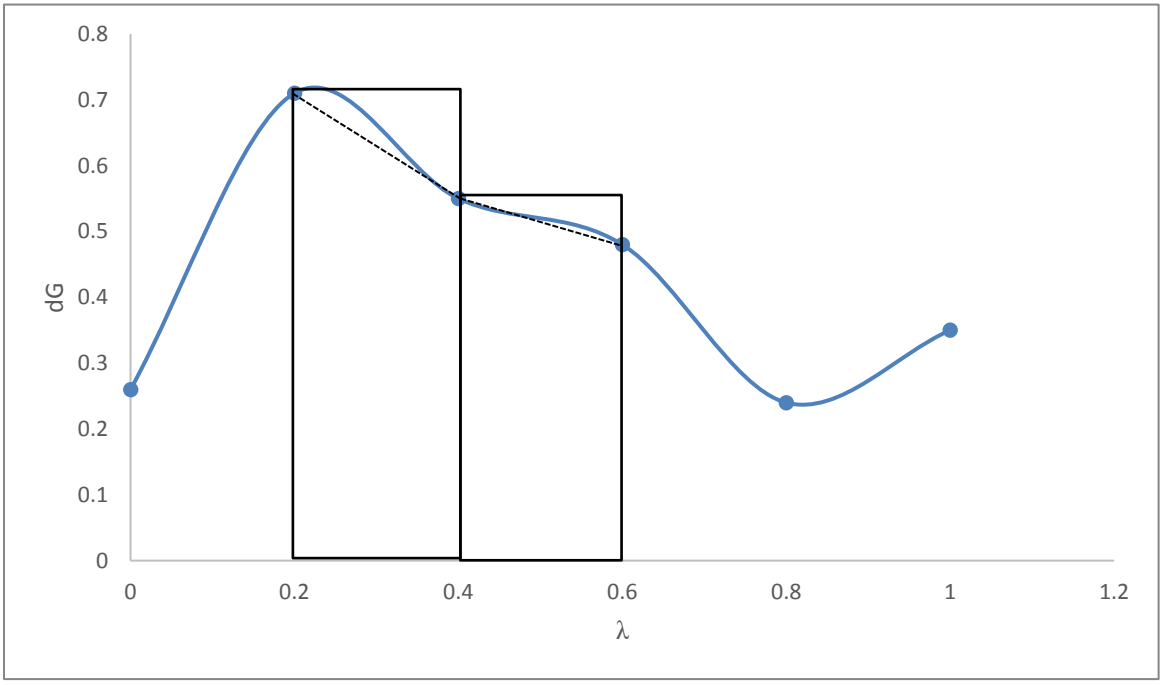


Figure 1.4: A graph illustrating the use of the trapezoidal rule.
The dashed lines connecting the points create adjacent trapezoids, areas of which can be calculated and summed up to give an approximate measure of the area under the curve.

Interpolation is a particularly important component of free energy calculations as it provides connectivity between finite number of discrete estimates, such as atomic distances and energy

states. Implementing fast and effective interpolation methods in free energy calculations yields realistic free energy diagrams in a shorter time frame.

The concepts of high-performance computing, numerical methods, and free energy calculations have a strong correlation. Considering this correlation, the study focuses on the investigation of two interpolation methods using free energy calculations.

1.6 THESIS SYNOPSIS

This dissertation reviews interpolation methods, demonstrating their uses in equilibrium and non-equilibrium calculations. A flat histogram method, FEARCF, is subsequently used to compare the effects of two interpolations: the cubic spline interpolation and B-spline interpolation.

Below is a brief outline of the topics that are covered in the next chapters.

Chapter 2:

- Theoretical information about interpolation methods, emphasizing the advantages of cubic spline interpolation.
- Introduction of an alternative method called B-spline interpolation providing simplified calculations and a faster algorithm.

Chapter 3:

- The methods for calculating free energy differences.
- The importance of Non-Boltzmann sampling for computing absolute free energies from the density of states.
- The use of interpolation in these methods is addressed.

Chapter 4:

- Detailed descriptions of the Non-Boltzmann flat histogram methods.
- Particular focus on the method FEARCF and its use of cubic spline interpolation.

Chapters 5:

- The comparison of the performances of the two interpolation methods using analytic functions

Chapter 6:

- The comparison between the same interpolation methods using the numerical data obtained from the FEARCF method. Accuracy and speed are the main criteria of evaluation.

1.7 REFERENCES

1. Eadline, D., *High Performance Computing for Dummies*. 2009: Wiley Publishing, Inc.
2. Jain, M.K., *Numerical methods for scientific and engineering computation*. 2003: New Age International.
3. Epperson, J.F., *On the Runge example.* The American Mathematical Monthly, 1987. **94**(4): p. 329-341.
4. Reinsch, C.H., *Smoothing by spline functions.* Numerische mathematik, 1967. **10**(3): p. 177-183.
5. Hiestand, J., *Numerical methods with VBA programming*. 2008: Jones & Bartlett Learning.
6. Zewail, A.H., *Laser femtochemistry.* Science, 1988. **242**(4886): p. 1645-1653.
7. Schramm, V.L., *Enzymatic transition states and transition state analog design*. 1998, Annual Reviews 4139 El Camino Way, PO Box 10139, Palo Alto, CA 94303-0139, USA.
8. Fano, G., L.F. Landovitz, and S. Technica, *Mathematical methods of quantum mechanics*. 1971: McGraw-Hill New York;.
9. Wales, D.J. and T.V. Bogdan, *Potential energy and free energy landscapes*. 2006, ACS Publications.
10. Ytreberg, F.M., R.H. Swendsen, and D.M. Zuckerman, *Comparison of free energy methods for molecular systems.* The Journal of chemical physics, 2006. **125**(18): p. 184114.

11. Cossins, B.P., et al., *Assessment of nonequilibrium free energy methods.* The Journal of Physical Chemistry B, 2009. **113**(16): p. 5508-5519.

12. Tropsha, A. and J. Hermans, *Application of free energy simulations to the binding of a transition-state-analogue inhibitor to HTV protease.* Protein Engineering, Design and Selection, 1992. **5**(1): p. 29-33.

13. Hastings, W.K., *Monte Carlo sampling methods using Markov chains and their applications.* 1970.

14. Murthy, K. *Non-Boltzmann Ensembles and Monte Carlo Simulations*. in *Journal of Physics: Conference Series*. 2016. IOP Publishing.

15. Torrie, G.M. and J.P. Valleau, *Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella sampling.* Journal of Computational Physics, 1977. **23**(2): p. 187-199.

16. Huber, T., A.E. Torda, and W.F. Van Gunsteren, *Local elevation: a method for improving the searching properties of molecular dynamics simulation.* Journal of computer-aided molecular design, 1994. **8**(6): p. 695-708.

17. Laio, A. and F.L. Gervasio, *Metadynamics: a method to simulate rare events and reconstruct the free energy in biophysics, chemistry and material science.* Reports on Progress in Physics, 2008. **71**(12): p. 126601.

18. Mezei, M., *Adaptive umbrella sampling: Self-consistent determination of the non-Boltzmann bias.* Journal of Computational Physics, 1987. **68**(1): p. 237-248.

19. Naidoo, K.J., *FEARCF a multidimensional free energy method for investigating conformational landscapes and chemical reaction mechanisms.* Science China Chemistry, 2011. **54**(12): p. 1962-1973.

20. Jacucci, G. and A. Rahman, *Comparing the efficiency of metropolis Monte Carlo and molecular-dynamics methods for configuration space sampling.* Il Nuovo Cimento D, 1984. **4**(4): p. 341-356.

21. Radmer, R.J. and P.A. Kollman, *Free energy calculation methods: a theoretical and empirical comparison of numerical errors and a new method qualitative estimates of free energy changes.* Journal of Computational Chemistry, 1997. **18**(7): p. 902-919.

22. Jorgensen, W.L., et al., *Comparison of simple potential functions for simulating liquid water*. The Journal of chemical physics, 1983. **79**(2): p. 926-935.

CHAPTER 2: REVIEW OF INTERPOLATION METHODS

2.1 IMPORTANCE OF INTERPOLATION

Many scientific research fields rely heavily on analysing experimental data obtained in the form of discrete data points. However, constructing a smooth and continuous model is necessary to interpret values that fall between discrete experimental data points along the data space.

Interpolation is the most reliable method to date because it provides a quick and easy evaluation. Using interpolation, any given data can be represented with an analytical function that forms a continuous curve through all the points. A potential side benefit is that the tabulated data can be differentiated and integrated to obtain valuable information from the data set.

Interpolation methods can be grouped depending on the level of continuity that they provide. Piecewise Constant Interpolation is a discontinuous method, while polynomial interpolation has a level of continuity and differentiability depending on the degree of the polynomial. Each interpolation method is beneficial in various scientific applications, yet not all of them are able to provide continuous curves that are twice differentiable. The methods that do provide such continuity will often have oscillatory errors because of high degree polynomials. Recent methods offer solutions to these issues by employing piece-wise interpolant curves called spline.

This chapter aims to introduce the aforementioned interpolation methods, discussing their advantages and drawbacks.

The primary focus will be the comparison between a single interpolant for all points and a piecewise spline interpolant between consecutive points.

2.2 INTERPOLATION TYPES

The explanation of interpolation methods will be in the increasing order of continuity, starting from a discontinuous interpolation method and progressing towards continuous and twice-differentiable interpolation methods.

2.2.1 PIECEWISE CONSTANT INTERPOLATION

In the piecewise constant interpolation method, one chooses the nearest data value and assigns the same value to that point. [23].

For a given x value, the data point x_i that minimizes $|x - x_i|$ is found and defined such that $f(x) = f(x_i) = y_i$

Piecewise constant interpolants are discontinuous at the data points and consequently non-differentiable.

This method is rarely used in one-dimensional interpolation but can be useful in the interpolation of multivariate cases. One such example is the Voronoi Diagram (VD), in which a Euclidean plane is partitioned into regions based on distances to the set of points. Each region has a data point called a seed that is the closest to all the other points in that particular region [24]. VD is broadly used in applications of natural sciences, health, and engineering. In one study, inter alia, they are used to calculate rainfall distribution in an area based on measurements at multiple points in that area[25]. Another area is medical diagnosis where tissue models and structures are constructed to detect diseases[4]. [26, 27] [26, 27] [26, 27] [26, 27]

OBJOBJ[27]

2.2.2 LINEAR INTERPOLATION

The linear interpolation method fits a curve by rendering linear polynomials between data points of a discrete set. The linear interpolant of two points is a straight line between these two points.

Given a set of data points $(x_i, f(x_i)); i = 1, \dots, N$, the linear polynomial $P(x) = P_i(x)$ with $x \in [x_i, x_{i+1}]$ can be written as follows;

$$P_i(x) = f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i) \quad (2.1)$$

Linear interpolation is of differentiability class C^0, meaning that it is continuous, but not differentiable [28].

Linear interpolation has been commonly used in population statistics. An example of its real-life application is the estimation of the population size at a given year based on the surrounding two years for which the population sizes are known. Furthermore, linear

interpolation calculations are widely used in computer graphics; however, the error of the approximation can be very large if there are large amplitude differences between points [29].

2.2.3 POLYNOMIAL INTERPOLATION

Polynomial interpolation can reduce the approximation errors by forming a smoother curve, thereby eliminating discontinuities. It interpolates a given data set by a polynomial passing through all of the data points with the lowest order possible[30].

Given a set of n+1 data points (x_i, y_i), the ideal polynomial interpolant has a degree of at most n and the following properties.

- $P(x_i) = y_i,\ i = 0,1, \dots, n$
- The polynomial is unique.

The uniqueness of the polynomial can be demonstrated using the proof [31] explained below.

Suppose that besides $p(x_i)$, there exists another polynomial denoted by $q(x_i)$ that has no more than n degrees and passes through the n+1 data points. Then, the subtraction of these two polynomials is $r(x) = p(x) - q(x)$. The polynomial r(x) must have the same degree as p and q since it is the difference between these two.

Looking at r(x) values at the n+1 data points; $r(x_i) = p(x_i) - q(x_i) = 0$. This means that r(x) has n+1 roots. In other words, r(x) can be written in the following form.

$$r(x) = A\ (x - x_0)(x - x_1)(x - x_2) \dots (x - x_n)\ (2.2)$$

where A is a constant.

The leading term of *r(x)* will be in the form of Ax^{n+1}, which contradicts with the fact that *r(x)* is a polynomial of degree n at most. The only way both conditions can be satisfied is for *A* to be equal to zero. This makes $r(x)\ =\ p(x)\ -\ q(x)$ equal to zero. The resultant is:

$$p(x) = q(x)(2.3)$$

The most straightforward way of constructing a polynomial interpolant given a set of points $(x_0, x_1, \dots x_n)$and a set of basis polynomials $(p_0, p_1, \dots, p_n)$ is to write it as the linear combination of the polynomials and a set of coefficients $(a_0, a_1, \dots, a_n)$ as follows;

$$p(x) = a_0 p_0(x) + a_1 p_1(x) + \cdots + a_n p_n(x) \ (2.4)$$

This equation can be expanded, and it holds true for all the points (x_0, f_0).

$$p(x_0) = a_0 p_0(x_0) + a_1 p_1(x_0) + \cdots + a_n p_n(x_0) = f_0 \ (2.5)$$

$$p(x_1) = a_0 p_0(x_1) + a_1 p_1(x_1) + \cdots + a_n p_n(x_1) = f_1 \ (2.6)$$

$$p(x_n) = a_0 p_0(x_n) + a_1 p_1(x_n) + \cdots + a_n p_n(x_n) = f_n \ (2.7)$$

The set of equations can be re-written as a linear system.

$$\begin{pmatrix} p_0(x_0) & p_1(x_0) & p_2(x_0) & \dots & p_n(x_0) \\ p_0(x_1) & p_1(x_1) & p_2(x_2) & \dots & p_n(x_1) \\ . & . & . & & . \\ . & . & . & & . \\ . & . & . & & . \\ p_0(x_n) & p_1(x_n) & p_2(x_n) & \dots & p_n(x_n) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ . \\ . \\ . \\ a_n \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ . \\ . \\ . \\ f_n \end{pmatrix} \ (2.8)$$

assuming $x_i \neq x_j \ for \ i \neq j$

If $p_i(x) = \ M_i(x) = x^i, i = 0,1, \dots, n$, the interpolating polynomial can be written in the form of a Monomial Basis[32] as shown below.

$$p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_n x^n \ (2.9)$$

The linear system associated with this polynomial would be,

$$\begin{pmatrix} 1 & x_0 & x_0^2 & x_0^3 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & x_1^3 & \dots & x_1^n \\ & & \dots & & & \\ & & \dots & & & \\ & & \dots & & & \\ 1 & x_n & x_n^2 & x_n^3 & \dots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ . \\ . \\ . \\ a_n \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ . \\ . \\ . \\ f_n \end{pmatrix} \ (2.10)$$

The nxn matrix in the equation above is called the Vandermonde Matrix (VM) [33]**.**

The computation of the coefficients $a_0, \dots, a_n$ and the polynomial interpolant requires the Lower-Upper Decomposition method to be applied on VM [34]

2.2.3.1 RUNGE'S PHENOMENON

It was initially hypothesised that the precision of a polynomial interpolant would increase with the increasing order. According to the Weierstrass approximation theorem[35] – for any

continuous function $f(x)$ on a real interval $[a, b]$, and for every value of Ɛ>0 – there exists a polynomial $P_n(x)$ of n[th] order such that:

$$|f(x) - P_n(x)| < \varepsilon \quad (2.11)$$

for all x ϵ [a,b]

The theorem suggests that there is a polynomial that will result in uniform convergence for the value of ε arbitrarily close to zero. However, it does not imply that the approximation error ε will be zero as n approaches infinity.

On the contrary, German mathematician Carl David Tolme Runge discovered that polynomial interpolants of higher degrees could cause oscillations near the endpoints of the interval [3].

A typical example Runge used to demonstrate the effect of increasing the order was the function $f(x) = \frac{1}{1+25x^2}$ [36]. Figure 2.1 illustrates three polynomial interpolants constructed with n degrees, interpolating $f(x)$ at $n + 1$ equidistant points via Polyfit[37]. The polynomial with the higher degree oscillates more towards the edges and interpolates the function more poorly.

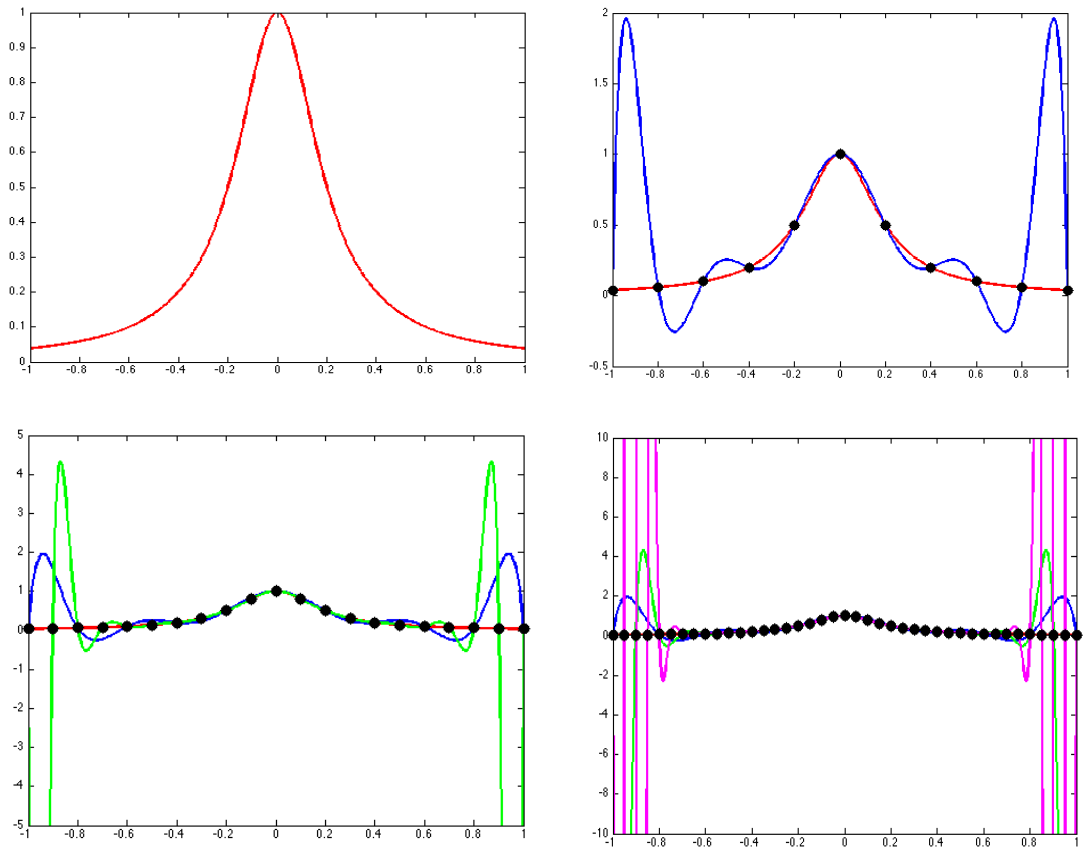


Figure 2.1: A set of graphs depicting the oscillatory performance of polynomial interpolants towards the endpoints.

The oscillation increases with the increasing degree of the polynomial.

Original image: https://math.boisestate.edu/~calhoun/teaching/matlab-tutorials/lab_11/html/lab_11.html

A more mathematical explanation states that the error between the analytical function and the interpolating function within an interval (a,b) is given by the formula:

$$f(x) - P_n(x) = R_n(x) = \frac{f^{n+1}(\varepsilon)}{(n+1)!}\prod_{i=0}^{n}(x - x_i) \quad (2.12)$$

For Ɛ ϵ (-1,1). Then, the maximum error can be written as follows:

$$\max_{-1\le x\le 1}|f(x) - P_n(x)| \le \max_{-1\le x\le 1}\frac{|f^{n+1}(\varepsilon)|}{(n+1)!}\max_{-1\le x\le 1}\prod_{i=0}^{n}|x - x_i| \quad (2.13)$$

This formula implies that there is a value of n, for which the difference between the interpolating polynomial and the real polynomial will be infinitely large. Therefore, the oscillatory error resulting from a high-degree polynomial can reach infinity as n goes to infinity.

The Runge's Phenomenon holds true only when the points are equally spaced[38].

2.2.3.2 SOLUTIONS TO RUNGE'S OSCILLATIONS

To reduce the polynomial interpolation error towards the endpoints, adjustment of the points was considered.

Chebyshev Method[39] was used to approximate a function using unequally distributed nodes to overcome Runge's Phenomenon. The idea is to place the nodes such that they will cluster towards the endpoints.

The Chebyshev nodes for an interval [a,b] are constructed by drawing a semicircle with a perimeter of [ab] and dividing it into n equal arcs. Then, the mid-points of the arcs are projected onto the interval.

The generalized formula for the calculation of such nodes on an interval [a,b] is:

$$x_i = \frac{1}{2}(a+b) + \frac{1}{2}(b-a)\cos\left(\frac{2i-1}{2n}\pi\right) \quad (2.14)$$

The positions of the nodes can be optimised further. The Remez Algorithm[40] takes Chebyshev nodes of an interval as the starting point and keeps relocating them iteratively until the error is minimised.

First, the polynomial approximation of the function f is obtained at the points of Chebyshev, denoted as $R(x)$. A system of linear equations is formed as follows:

$$R(x_{ie}) + (-1)^{ie}E = f(x_{ie}) \; (2.15-a)$$

where $R(x_{ie}) = c_0 + c_1 x_{ie} + c_2 x_{ie}^2 + \cdots + c_N x_{ie}^N$ (2.15-b) for the Chebyshev nodes x_{ie} and E is the absolute maximum error, which is also unknown

Solving the system of equations, both the set of coefficients and the value E are generated. Using the set of coefficients, the roots of the polynomial $R(x_{ie}) = c_0 + c_1 x_{ie} + c_2 x_{ie}^2 + \cdots + c_N x_{ie}^N = 0$ are found.

With this new polynomial, the set of local maximum errors $|f(x_{ie}) - R(x_{ie})|$ are computed between each consecutive root. If all the errors are equal to E and they alternate in sign, the best approximation polynomial – the minimax approximation polynomial – is obtained. Otherwise, the roots found in equation 2.15-b are used as the updated nodes to be substituted in equation 2.15-a for the next iteration.

2.3 SPLINE INTERPOLATION

Even though the node-adjustment methods provide a better approximation, alternative interpolation methods can still employ equidistant points avoiding high-degree polynomials. Instead of a n-degree polynomial, for n+1 points, one can apply n polynomials with fewer degrees. Each polynomial is called a *spline* and connecting pairs of points using splines is called *spline interpolation*.

Spline, as a concept, was first used in elasticised rulers that had the ability to bend. Using these rulers, one would pass through fixed points called knots. These rulers were widely used in technical drawings in the design and construction of ships, where the aim was to model the shape of the ship from the elastic ruler through interpolation between the fixed knots[41].

For n+1 knots $[(x_i, y_i): i = 0,1, \dots, n)]$, interpolation between the pairs of consecutive knots require spline polynomials satisfying the condition $y = p_i(x), i = 1,2, \dots, n$.

These spline polynomials ensured that the following conditions were met:

1- The shape of the ruler had minimum amount of bending provided that the ruler passed through the knots
2- The first and the second derivatives throughout the ruler, including the knots, were continuous. The second condition can be represented as follows:

$$\begin{cases} p_i'(x_i) = p_{i+1}'(x_i) \\ p_i''(x_i) = p_{i+1}''(x_i) \end{cases} \qquad 1 \le i \le n-1 \quad (2.16)$$

It was discovered that a spline satisfying these conditions had a degree of 3 or higher[42].

2.3.1 CUBIC SPLINE INTERPOLATION

The spline function that satisfies the conditions above with the minimum degree is called a cubic spline. In the *cubic spline interpolation* method, every spline is a cubic polynomial expressed as follows:

$$P_i(x) = a_i x^3 + b_i x^2 + c x_i + d \ (2.17)$$

There are several advantages to cubic spline polynomials over previously mentioned methods:

- They are computationally easy to handle, derive and integrate because of their low degree.
- Spline interpolation provides control over the data set by putting certain parameters, *viz* the number and the positioning of the knots, at the programmer's disposal.
- A cubic spline has the differentiability class of C^2 [43] ensuring the continuity at the knots as well as the continuity of the first and the second derivatives.

There are certain principles for the construction of cubic splines. The positioning of the knots is essential for the best approximation that represents the overall trend of the data. To minimise the computational time, it is important to have as few knots as possible; however, it is also important to contain at least 4 or 5 points per interval since fewer points per interval will cause overfitting. In addition, there can only be a maximum of one extremum point (a local maximum or a local minimum) and one inflection point (where the second derivative

changes sign) per interval. The extremum points should be close to the midpoints of the intervals while the inflection points should be close to the knots[44].

The computation of the coefficients is crucial for the construction of the cubic splines. The computation for two splines interpolating three points is demonstrated as an example.

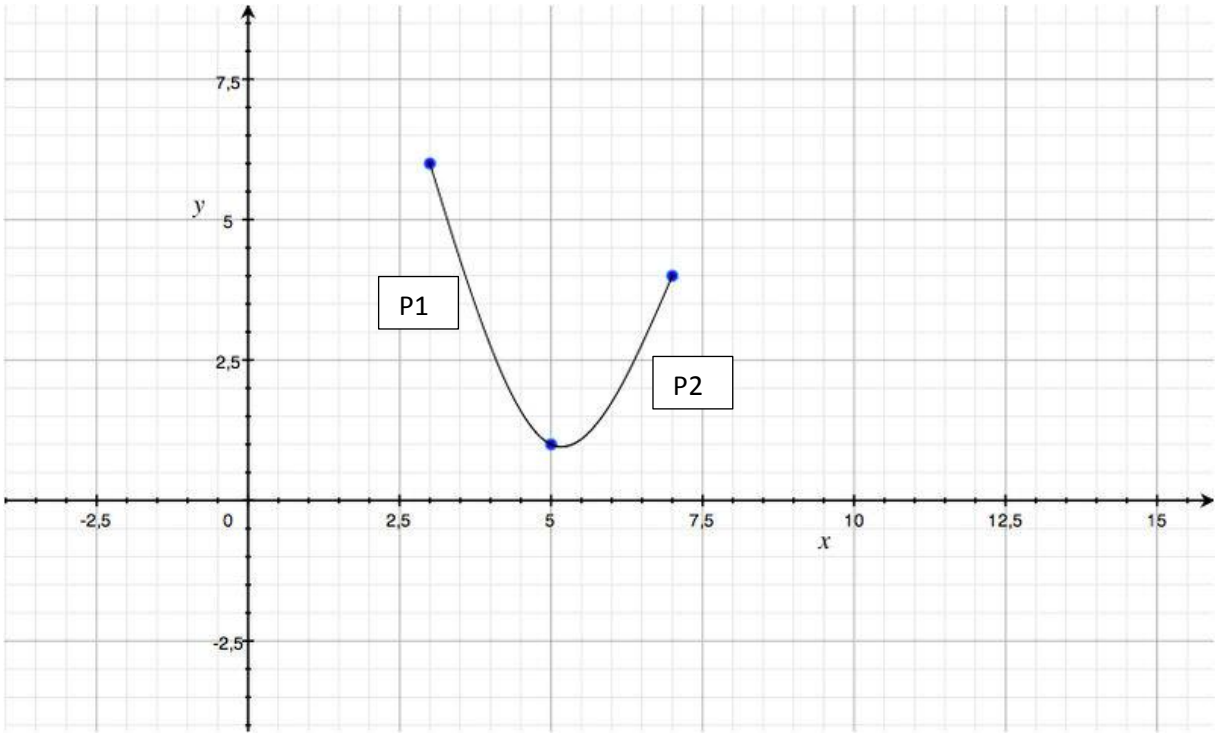


Figure 2.2: A graph showing two cubic spline polynomials p_1 and p_2

The representation of the two polynomials and their coefficients given three points (x_0 y_0), (x_1 y_1), (x_2 y_2) is:

$$p_1(x) = a_1x^3 + b_1x^2 + c_1x + d_1 \ (2.18)$$
$$p_2(x) = a_2x^3 + b_2x^2 + c_2x + d_2 \ (2.19)$$

The two equations can be defined with four conditions.

Condition 1: defines the values of the polynomials at the given points.

$$p_1(x_0) = y_0 \ (2.20)$$

$$p_1(x_1) = y_1 \ (2.21)$$

$$p_2(x_1) = y_1 \ (2.22)$$

$$p_2(x_2) = y_2 \ (2.23)$$

Condition 2: states that the derivatives of the two polynomials are equal at point $x=x_1$

$$p_1'(x_1) = p_2'(x_1) \ (2.24)$$

Condition 3: states that the second derivatives are also equal at $x=x_1$

$$p_1''(x_1) = p_2''(x_1) (2.25)$$

Condition 4: assumes that the left endpoint of the first polynomial and the right endpoint of the second polynomial have their second derivatives set to zero.

$$p_1''(x_0) = 0 \ (2.26) \qquad p_2''(x_2) = 0 \ (2.27)$$

Overall, the system has eight equations and eight unknown coefficients, which can be solved via the matrix.

$$a_1x_0^3 + b_1x_0^2 + c_1x_0 + d_1 = y_0$$

$$a_1x_1^3 + b_1x_1^2 + c_1x_1 + d_1 = y_1$$

$$a_2x_1^3 + b_2x_1^2 + c_2x_1 + d_2 = y_1$$

$$a_2x_2^3 + b_2x_2^2 + c_2x_2 + d_2 = y_2$$

$$0x_1^3 + 3(a_1 - a_2)x_1^2 + 2(b_1 - b_2)x_1 + (c_1 - c_2) = 0$$

$$0x_1^3 + 0x_1^2 + 6(a_1 - a_2)x_1 + 2(b_1 - b_2) = 0$$

$$0x_0^3 + 0x_0^2 + 6(a_1)x_0 + 2(b_1) = 0$$

$$0x_2^3 + 0x_2^2 + 6(a_2)x_2 + 2(b_2) = 0 \ (2.28)$$

Although the cubic spline approach is more flexible and avoidant of the oscillatory errors of polynomial interpolations, it has its own disadvantages. For each spline in the data space, one must calculate 4 coefficients, inferring that one has to solve 4n equations with 4n unknowns. This is time-consuming and computationally expensive for a large number of points[45]. An alternative method was developed by focusing on the second derivatives at the knots followed by interpolating at the point of interest.

The formulation for cubic spline interpolation in the book "Numerical Recipes For Fortran" [46] begins by defining a general spline function for the interval [x_j, x_{j+1}] that is twice differentiable.

$$y = Ay_j + By_{j+1} + Cy_j'' + Dy_{j+1}'' \quad (2.29)$$

The first two terms $y = Ay_j + By_{j+1}$ define linear interpolation in the interval between x_j and x_{j+1} where

$$A = \frac{x_{j+1} - x}{x_{j+1} - x_j} \quad (2.30) \qquad B = 1 - A = \frac{x - x_j}{x_{j+1} - x_j} \quad (2.31)$$

are written as a special case of Lagrange interpolation formula. The third and the fourth terms constitute a cubic polynomial that has zero values at x_j and x_{j+1}, and second derivatives ranging from y_j to y_{j+1}. C and D are defined in terms of A and B respectively.

$$C = \frac{1}{6}(A^3 - A)\left(x_{j+1} - x_j\right)^2 \quad (2.32) \qquad D = \frac{1}{6}(B^3 - B)\left(x_{j+1} - x_j\right) \quad (2.33)$$

Taking the first and the second derivative of the polynomial y:

$$\frac{dy}{dx} = \frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{3A^2 - 1}{6}\left(x_{j+1} - x_j\right)y_j'' + \frac{(3B^2 - 1)}{6}\left(x_{j+1} - x_j\right)y_{j+1}'' \quad (2.34)\ and$$

$$\frac{d^2y}{dx^2} = Ay_j'' + By_{j+1}'' \quad (2.35)$$

Substituting x_j makes A=1 and B=0, while A=0 and B=1 for x_{j+1}. Thus, the form of polynomial equation 2.29 ensures that the spline $y''$is continuous at the point x_j.

The only missing step is the computation of the second derivatives y_j''. Because the second derivatives are continuous across x_j, the first derivatives at x_j must be equal for the intervals (x_{j-1}, x_j) and $\left(x_j, x_{j+1}\right)$. In other words, the equation (2.34) holds true for both intervals. Substituting and rearranging gives

$$\frac{x_j - x_{j-1}}{6}y_{j-1}'' + \frac{x_{j+1} - x_{j-1}}{3}y_j'' + \frac{x_{j+1} - x_j}{6}y_{j+1}'' = \frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{y_j - y_{j-1}}{x_j - x_{j-1}} \quad (2.36)$$

For $j = 2, \dots, N-1$, there are N-2 equations, but N unknowns for $y_j''; j = 1,2, \dots, N$, therefore two more conditions are needed: the boundary condition at x_1, and at x_N.

There are two ways of setting the boundary conditions: setting both y_1'' and y_N'' equal to zero, alternatively, setting these values from the equation (2.34) with the values y_1' and y_N'. The set of equations indicates that each value of y_j'' is only coupled to its neighbours both $y_{j\pm1}''$, meaning that the system of equations can be solved via the tridiagonal matrix shown below:

$$\begin{pmatrix} \frac{x_2-x_1}{6} & \frac{x_3-x_1}{3} & \frac{x_3-x_2}{6} & 0 & \dots & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & \frac{x_3-x_2}{6} & \frac{x_4-x_2}{3} & \frac{x_4-x_3}{6} & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots & \dots & 0 & \frac{x_{n-2}-x_{n-3}}{6} & \frac{x_{n-1}-x_{n-3}}{3} & \frac{x_{n-1}-x_{n-2}}{6} & 0 \\ 0 & \dots & \dots & \dots & \dots & \dots & \dots & 0 & \frac{x_{n-1}-x_{n-2}}{6} & \frac{x_n-x_{n-2}}{3} & \frac{x_n-x_{n-1}}{6} \end{pmatrix} \begin{pmatrix} y_1'' \\ y_2'' \\ y_3'' \\ \dots \\ y_{n-2}'' \\ y_{n-1}'' \\ y_n'' \end{pmatrix}$$

$$= \begin{pmatrix} m_2 \\ m_3 \\ m_4 \\ \dots \\ m_{n-3} \\ m_{n-2} \\ m_{n-1} \end{pmatrix} \; with\; m_{n-1} = \frac{y_n - y_{n-1}}{x_n - x_{n-1}} - \frac{y_{n-1} - y_{n-2}}{x_{n-1} - x_{n-2}} \quad (2.37)$$

Since $x_n - x_{n-1} = x_{n-1} - x_{n-2}$ for equidistant nodes, the tridiagonal matrix is symmetrical[47]

Once the second derivatives y_j'' are obtained from equation 2.37, a y value can be calculated for any given x value using equations 2.29-33 by substituting x_j, y_j and y_j'' values. The derivative y' at that value can be calculated from equation 2.34.

The formulation of multidimensional cubic spline interpolation is more complicated. The calculation must be broken down to one-dimensional arrays to apply the equations 2.29-37. The formulation for the two-dimensional cubic spline method is explained based on "Numerical Recipes in Fortran"[46] and the code written by J. Strümpfer[48].

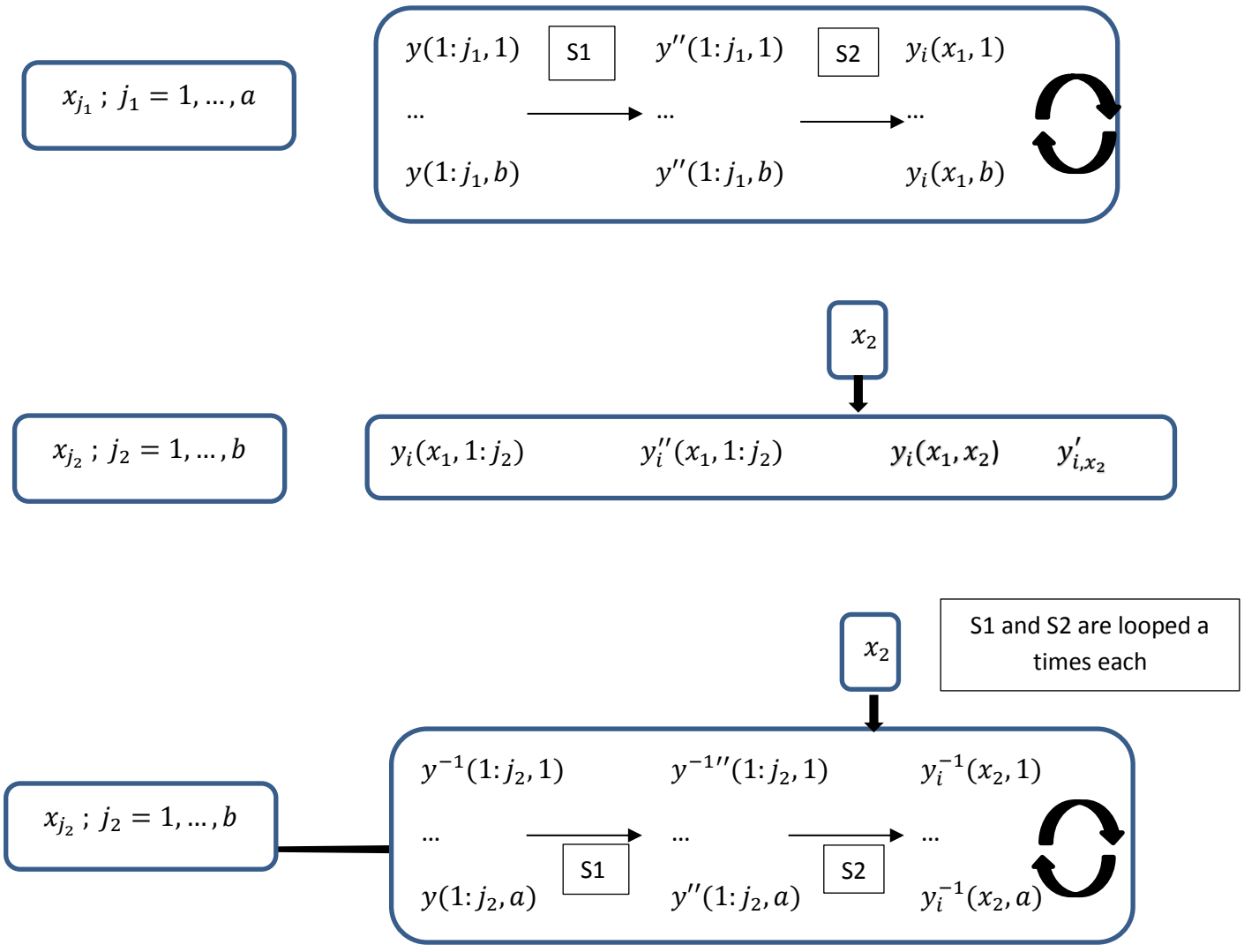
Two arrays of independent values $x_{j_1}, j_1 = (1, \dots, a)$ and $x_{j_2}, j_2 = (1, \dots, b)$ and a matrix of y values $y = [y_{j_1}, y_{j_2}]$ are given. The aim is to find a value of $y(x_1, x_2)$, where x_1 and x_2 are two random values in the respective intervals $[1, a]$ and $[1, b]$.

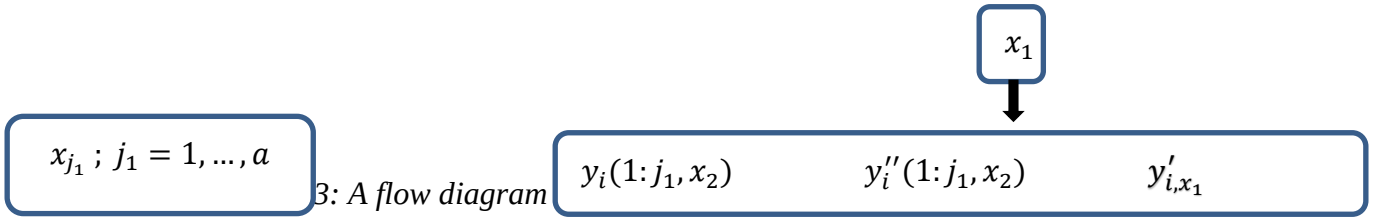
First, the matrix y is broken down into b arrays each having length, a. For each y_{j_1} array, a set of second derivatives y_{j_1}'' is computed using the linear equation system 2.37. Consequently, a

single interpolation is performed by substituting the arrays x_{j_1}, y_{j_1} and y''_{j_1} (each of length a) into the equations 2.29-33. This calculation is performed b times for all b arrays of y_{j_1}.

At the end of b iterations, an array of interpolated values y_{i_2} is obtained for $i_2 = 1, \dots, b$. Then, the array of x_{j_2} and the set of values y_{i_2} can be substituted into the equation 2.37 to calculate the second derivatives y''_{i_2}. This is followed by a single interpolation to obtain the final interpolated value of $y(x_1, x_2)$ by substituting the values x_1, x_2 and y''_{i_2} into equations 2.29-33. The partial derivative y'_{j_2} is calculated with respect to x_2 using equation 2.34.

To calculate the partial derivative with respect to x_{j_1}, this nested interpolation must be followed again. This time, the matrix y must be inverted and broken down to a number of arrays. Thereafter, one-dimensional interpolation is performed a times between x_{j_2} and each of the arrays y_{j_2} in a manner similar to that described in the previous two paragraphs, generating a set of interpolated values y_{i_1} for $i_1 = 1, \dots, a$. Thus, the final interpolation can be performed with respect to the x_{j_1} values and the interpolated set of y_{i_1} values using the equations 2.29-37 one last time. In the end, the partial derivative y'_{i_1} is computed.





3: A flow diagram

For higher dimensions, this becomes more complex, as the aim is to obtain one- dimensional arrays for which the basic interpolation is applicable. For a three-dimensional matrix of $y_{j_1 j_2 j_3}$, with $j_1 = 1, \dots, a\ j_2 = 1, \dots, b\ and\ j_3 = 1, \dots, c$, a two-dimensional matrix of initially interpolated values $y_{i_2 i_3}$ is obtained through b second-derivative calculations and b interpolations in each row. Before computing the elements of the next row, another second-derivative-calculation and interpolation is performed to obtain a single interpolated value y_{i_3} corresponding to that row.

An array of interpolated values y_{i_3} with length c is obtained after $c(b+1)$ iterations. This array is finally used to compute the final $y_{i_1 i_2 i_3}$ value and its derivative with respect to j_1. The entire procedure has to be repeated twice more to compute every partial derivative. Overall, the steps described for a two-dimensional cubic spline interpolation are repeated for all the elements of the third dimension.

From a computational perspective, a multi-dimensional cubic spline algorithm includes nested loops. In addition, a separate set of second derivatives have to be calculated before every intermediate interpolation step and the final interpolation step. As a consequence, the algorithm would lead to a drastic increase in the sequential computational time.

2.3.2 B-SPLINE INTERPOLATION

2.3.2.1 BASIS FUNCTIONS

The set of all cubic splines with fixed knots forms a vector space, for which a set of functions form a basis. A spline interpolation can be written as a unique linear combination of these basis functions, which is called a B-spline interpolation.

The most fundamental B-spline basis has a zero degree and is called a box function[49].

$$B_0(x) = \begin{cases} 1, & |x| < \frac{1}{2} \\ \frac{1}{2}, & |x| = \frac{1}{2} \\ 0, & |x| > \frac{1}{2} \end{cases} \quad (2.38)$$

A- B spline basis of degree n can be obtained by the recursive convolution of the above function with the B-spline basis of degree n-1.

The general recursive formula developed by C. De Boor[50] for the expression of basis functions of the function space $S_{k,j}(\Delta_n(a,b))$ is as follows:

$$B_j^{k,\Delta}(x) = \frac{x - x_{j-1-k}}{x_{j-1} - x_{j-1-k}} B_{j-1}^{k-1,\Delta}(x) + \frac{x_j - x}{x_j - x_{j-k}} B_j^{k-1,\Delta}(x) (2.39)$$

for k=0, $B_j^{0,\Delta}$ basis functions are:

$$B_j^{0,\Delta} = \begin{cases} 1 & x_{j-1} \le x < x_j \\ 0 & otherwise \end{cases} \quad (2.40)$$

In these equations, j is in the interval $j = 1, 2, \dots, n, \dots, n+k$ and $B_j^{k,\Delta}$ is the basis for the function space $S_{k,k-1}(\Delta_n(a,b))$ with n degrees.

Within the function space, a spline interpolation at the point x, $s_n^{k,k-1}$, can be written as a linear combination of the B-spline basis functions.

$$s_n^{k,k-1}(x) = \sum_{j=1}^{n+k} c_j B_j^{k,\Delta}(x) \quad (2.41)$$

Where c_j are weighing coefficients.

2.3.2.2 IMPLEMENTATION ON CUBIC B- SPLINE WITH EQUIDISTANT KNOTS

For equidistant knots [x_0, x_1, x_2, …, x_n], each interval is defined as;

$$x_i = a + ih \ (2.42) \qquad h = \frac{b-a}{n} \ (2.43) \qquad i = 0,1, \dots, n \rightarrow \qquad a = x_0, b = x_n (2.44)$$

The set of basis functions U = [u_1, u_2, …, u_{n+3}] shown in figure 2.4 can be obtained by substituting equidistant nodes into the recursive functions as described by Habermann[51] and illustrated below, Figure 2.3.

$$u_k(x) = \phi(t) = \begin{cases} (2-|t|)^3 & 1 \le |t| \le 2 \\ 4 - 6|t|^2 + 3|t|^3 & |t| \le 1 \\ 0 & elsewhere \end{cases} \quad (2.45)$$

$$t = \frac{x-a}{h} - (k-2)\ (2.46) \quad k = 1,2,\dots,n+3 (2.47)$$

The basis function described above will be a piecewise cubic polynomial that is non-zero between -2 and 2.

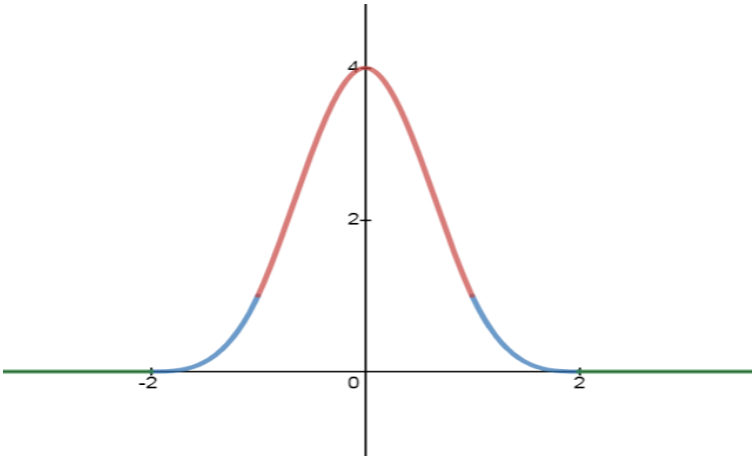


Figure 2.4: A graph showing the basis function to be used in this study.
The curve was created on the website Desmos using its piecewise function feature.

Then, the spline interpolation at point x, $s \in S_3(\Delta_n(a,b))$, can be written in the form

$$s(x) = \sum_{k=1}^{n+3} c_k u_k(x)\ (2.48)$$

When the knots and data points are spaced equally, the basis function in every interval has an identical shape. Consequently, any one of them can be obtained by translating one basis function along the x-axis[52].

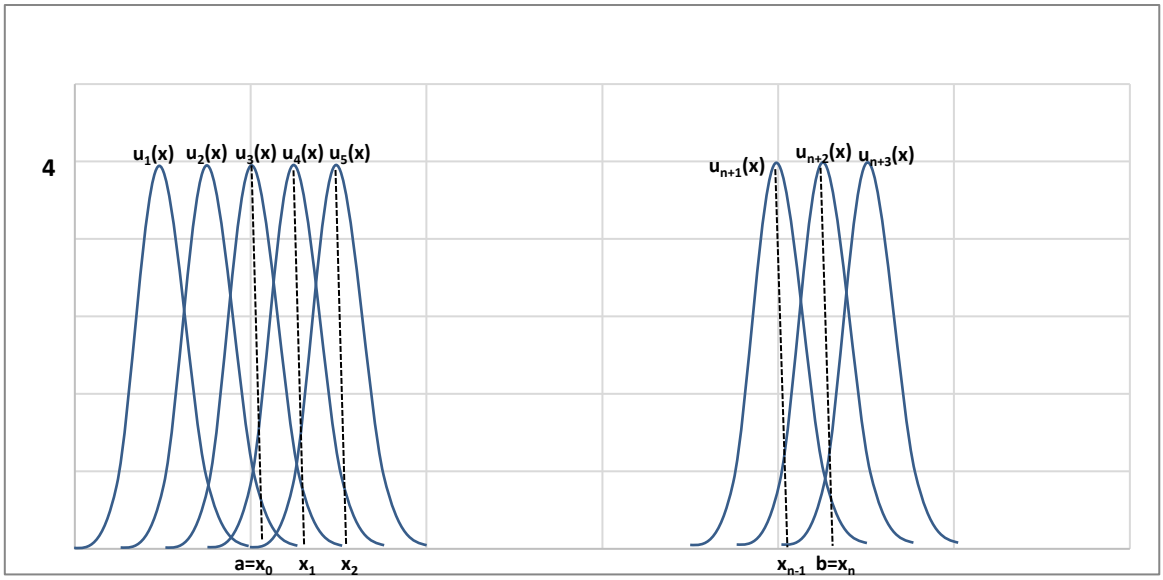


Figure 2.5: A graph showing identical basis functions constructed with equidistant knots, making up the basis of the spline.
The diagram is an adaptation, based on Habermann [51]

The biggest challenge is to find the coefficients of the linear system. As explained in the work of Habermann[51], the initial conditions of the linear system are defined as follows:

$$s''(x_0) = \sum_{i=1}^{3} c_i u_i''(x_0) = \alpha \quad (2.49)$$

$$s''(x_i) = \sum_{k=l}^{m} c_k u_k(x_i) = y_i \; with \; l = \left[\frac{x_i - a}{h}\right] + 1, m = \min(l+3, n+3),$$
$$i = 0,1,\dots,n \quad (2.50)$$

$$s''(x_n) = \sum_{i=n+1}^{n+3} c_i u_i''(x_n) = \beta \quad (2.51)$$

where α and β are constants; u_k is the basis function for the k[th] interval; c_k is the corresponding coefficient. The algorithm implements the natural splining described in the work of De Boor [21] and sets the constants $\alpha = \beta = 0$ to minimise the overall curvature [53].

Given the conditions 2.49-51, the system of equations can be written as a matrix:

$$\begin{pmatrix} u_1''(x_0) & u_2''(x_0) & u_3''(x_0) & 0 & \dots & \dots & 0 \\ u_1(x_0) & u_2(x_0) & u_3(x_0) & 0 & \dots & \dots & 0 \\ 0 & u_2(x_1) & u_3(x_1) & u_4(x_1) & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & u_n(x_{n-1}) & u_{n+1}(x_{n-1}) & u_{n+2}(x_{n-1}) & 0 \\ 0 & \dots & \dots & 0 & u_{n+1}(x_n) & u_{n+2}(x_n) & u_{n+3}(x_n) \\ 0 & \dots & \dots & 0 & u_{n+1}''(x_n) & u_{n+2}''(x_n) & u_{n+3}''(x_n) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \dots \\ c_{n+1} \\ c_{n+2} \\ c_{n+3} \end{pmatrix} = \begin{pmatrix} \alpha \\ y_0 \\ y_1 \\ \dots \\ y_{n-1} \\ y_n \\ \beta \end{pmatrix} \quad (2.53)$$

If the nodes are equidistant, the set of $u_k(x_i), u_k''(x_0)\ and\ u_k''(x_n)$ can be computed. Using the equations 2.45-47 creates the following matrix:

$$\begin{pmatrix} 1 & -2 & 1 & 0 & \dots & \dots & 0 \\ 1 & 4 & 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & 4 & 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 1 & 4 & 1 & 0 \\ 0 & \dots & \dots & 0 & 1 & 4 & 1 \\ 0 & \dots & \dots & 0 & 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \dots \\ c_{n+1} \\ c_{n+2} \\ c_{n+3} \end{pmatrix} = \begin{pmatrix} \frac{\alpha h^2}{6} \\ y_0 \\ y_1 \\ \dots \\ y_{n-1} \\ y_n \\ \frac{\beta h^2}{6} \end{pmatrix} \quad (2.54)$$

The first two and the last two rows lead to the equations

$$c_2 = \left(\frac{1}{6}\right)\left(y_0 - \frac{\alpha h^2}{6}\right) \quad (2.55), \qquad c_{n+2} = \frac{1}{6}\left(y_n - \frac{\beta h^2}{6}\right) \quad (2.56)$$

When the first and the last rows are omitted, the following tridiagonal matrix is formed:

$$\begin{pmatrix} 4 & 1 & 0 & \dots & \dots & \dots & \dots & 0 \\ 1 & 4 & 1 & 0 & \dots & \dots & \dots & 0 \\ 0 & 1 & 4 & 1 & 0 & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & 0 & 1 & 4 & 1 & 0 \\ 0 & \dots & \dots & \dots & 0 & 1 & 4 & 1 \\ 0 & \dots & \dots & \dots & \dots & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} c_3 \\ c_4 \\ \dots \\ \dots \\ \dots \\ c_n \\ c_{n+1} \end{pmatrix} = \begin{pmatrix} y_1 - c_2 \\ y_2 \\ y_3 \\ \dots \\ y_{n-3} \\ y_{n-2} \\ y_{n-1} - c_{n+2} \end{pmatrix} \quad (2.57)$$

The matrix above can be solved given the y values and the computed values of c_2 and c_{n+2}. Finally, c_1 and c_{n+3} can be computed from the equations:

$$c_1 = \frac{\alpha h^2}{6} + 2c_2 - c_3 \quad (2.58), \qquad c_{n+3} = \frac{\beta h^2}{6} + 2c_{n+2} - c_{n+1} \quad (2.59)$$

The B-spline formulation can be generalised in higher dimensions.

The extended algorithm named the bicubic B-spline interpolation is the combination of the coefficient matrix $c_{i_1 i_2}$ and basis functions u_{i_1} and v_{i_2}. Given the two points $(x1, x2)$

$$s(x1, x2) = \sum_{i_1=1}^{n_1+3} \sum_{i_2=1}^{n_2+3} c_{i_1 i_2} u_{i_1}(x1)\, v_{i_2}(x2) \quad (2.60)$$

For a single value i_2, notated q, the spline $s_q(x)$ that satisfies the condition $s_q(x1) = y_{i_1 q}$ is the linear combination of $u_{i_1}(x1)$ and some coefficients $c^*_{i_1}$. For every value of $q = 0,1, \dots, n_2$; the following condition holds:

$$s_q(x) = \sum_{i_1=1}^{n_1+3} c^*_{i_1 q} u_{i_1}(x) \quad (2.61)$$

$c^*_{i_1 q}$ can be calculated using the equations 2.53-59. These coefficients will satisfy the condition $s_{i_1}(x2) = c^*_{i_1 q}$. For a single value of i_1, each coefficient $c^*_{i_1 q}$would be a linear combination of the basis functions v_{i_2} and the coefficients $c_{i_1 i_2}$. For all values of i_1:

$$s_{i_1 2}(x2) = \sum_{i_2=1}^{n_2+3} c_{i_1 i_2} v_{i_2}(x2) \quad for\ i_1 = 1, \dots, n_1 + 3 \quad (2.62)$$

The system can be solved using the equations 2.53-59 to obtain the coefficient matrix $c_{i_1 i_2}$.

2.4 COMPARING CUBIC AND B SPLINE INTERPOLATIONS

Despite its advantages, the cubic spline interpolation has disadvantages in regard to the computational cost. The method of interpolation through the equations 2.29-37 can be straightforward in a single dimensional case, but it is much more strenuous in multiple dimensions. The B-spline formulation eliminates this complexity by implementing a one-time coefficient calculation before the interpolation. Once the coefficients are computed, the basis functions can be combined with them, linearly, to compute the value at any point. For uniform knots, the basis functions are identical for each interval, making the linear combination easier. The biggest advantage is related to the partial derivatives, which can be calculated by repeating the calculations with the derivatives of the basis functions. The ability of the B-spline algorithm to calculate all the partial derivatives at once is a significant improvement over the cubic spline algorithm.

Given the simplifications of the B-Spline method, its implementation can lead to significant speedup. However, this hypothesis requires an extensive comparison between the two methods, which has not been done before. Firstly, it must be established that using B-splining instead of cubic splining does not affect the accuracy. The same confirmation is needed for

the partial derivatives. Finally, the elapsed time of interpolation must be measured for both methods. As far as the current research goes, there is a lack of proof that presents an answer to these questions in a single study.

This comparison can be performed in two ways: through a series of analytical functions and a numerical data.

The comparison with analytical functions is crucial to understand the change in performance when certain parameters are modified. In addition, derivatives cannot be computed from a numerical data because it lacks an analytical function. Therefore, the first part of this study focused on the performance of the cubic spline interpolation and the B-spline interpolation using different data sets of a variety of analytic functions. The analytic functions were grouped as univariate and multivariate. The two univariate functions are; a periodic cosine function and a polynomial, while the multivariate function was a mixture of the two functions.

The second part of the study focused on the applications of interpolation. The aim was to work with free energy calculations, particularly flat histogram methods. Interpolation is a crucial approximation method in free energy methods, where different energy states are discontinuous and discretized.

The next two chapters review the free energy methods and their implementations of interpolation. This will provide an insight into the theory of the comparison and its application to flat histogram methods.

2.5 RESOURCES

1. Parker, J.A., R.V. Kenyon, and D.E. Troxel, *Comparison of interpolating methods for image resampling.* IEEE Transactions on medical imaging, 1983. **2**(1): p. 31-39.
2. Fairfield, J., *Segmenting dot patterns by Voronoi diagram concavity.* IEEE transactions on pattern analysis and machine intelligence, 1983(1): p. 104-110.
3. C.s.v. Arnold E. Perham, F.L.P., *Voronoi Diagrams and Spring Rain.* The Mathematics Teacher. **105**(2): p. 126.
4. !!! INVALID CITATION !!! .

5. Sudbø, J., R. Marcelpoil, and A. Reith, *New algorithms based on the Voronoi Diagram applied in a pilot study on normal mucosa and carcinomas.* Analytical Cellular Pathology, 2000. **21**(2): p. 71-86.

6. Courant, R. and F. John, *Introduction to calculus and analysis I*. 2012: Springer Science & Business Media.

7. Neumaier, A., *Computer graphics, linear interpolation, and nonstandard intervals*. 2009.

8. Sauer, T., *Polynomial interpolation of minimal degree.* Numerische Mathematik, 1997. **78**(1): p. 59-85.

9. Jia, Y.-B., *Polynomial Interpolation.* 2017.

10. Gander, W., *Change of basis in polynomial interpolation.* Numerical linear algebra with applications, 2005. **12**(8): p. 769-778.

11. Klinger, A., *The vandermonde matrix.* The American Mathematical Monthly, 1967. **74**(5): p. 571-574.

12. Oruç, H. and G.M. Phillips, *Explicit factorization of the Vandermonde matrix.* Linear Algebra and its Applications, 2000. **315**(1-3): p. 113-123.

13. Stone, M.H., *The generalized Weierstrass approximation theorem.* Mathematics Magazine, 1948. **21**(5): p. 237-254.

14. Epperson, J.F., *On the Runge example.* The American Mathematical Monthly, 1987. **94**(4): p. 329-341.

15. Lin, H. and L. Sun, *Searching globally optimal parameter sequence for defeating Runge phenomenon by immunity genetic algorithm.* Applied Mathematics and Computation, 2015. **264**: p. 85-98.

16. Edwards, A. and J. Dennis, *Polyfit-A computer program for fitting a specified degree of polynomial to data points.* NRPB-M, 1973. **11**: p. 1-18.

17. Boyd, J.P., *Defeating the Runge phenomenon for equispaced polynomial interpolation via Tikhonov regularization.* Applied Mathematics Letters, 1992. **5**(6): p. 57-59.

18. Tchebychev, P.L., *Théorie des mécanismes connus sous le nom de parallélogrammes*. 1853: Imprimerie de l'Académie impériale des sciences.

19. Remez, E.Y., *Sur la détermination des polynômes d'approximation de degré donnée.* Comm. Soc. Math. Kharkov, 1934. **10**(4163): p. 196.

20. Segner, D., *The shape of the human face recorded by use of contour photography and spline function interpolation.* The European Journal of Orthodontics, 1986. **8**(2): p. 112-117.

21. De Boor, C., et al., *A practical guide to splines*. Vol. 27. 1978: springer-verlag New York.

22. Unser, M., *Splines: A perfect fit for signal and image processing.* IEEE Signal processing magazine, 1999. **16**(ARTICLE): p. 22–38.

23. Foley, T.A. and G.M. Nielson, *Knot selection for parametric spline interpolation*, in *Mathematical methods in computer aided geometric design*. 1989, Elsevier. p. 261-CP4.

24. Siauw, T. and A. Bayen, *An introduction to MATLAB® programming and numerical methods for engineers*. 2014: Academic Press.

25. Press, W.H., et al., *Numerical recipes in Fortran 90.* The art of scientific computing,(Cambridge, 1996), 1992.

26. Schmeisser, G., *A real symmetric tridiagonal matrix with a given characteristic polynomial.* Linear algebra and its applications, 1993. **193**: p. 11-18.

27. Rajamani, R., K.J. Naidoo, and J. Gao, *Implementation of an adaptive umbrella sampling method for the calculation of multidimensional potential of mean force of chemical reactions in solution.* Journal of computational chemistry, 2003. **24**(14): p. 1775-1781.

28. Ruijters, D., B.M. ter Haar Romeny, and P. Suetens, *Efficient GPU-based texture interpolation using uniform B-splines.* Journal of Graphics Tools, 2008. **13**(4): p. 61-69.

29. De Boor, C., *On calculating with B-splines.* Journal of Approximation theory, 1972. **6**(1): p. 50-62.

30. Habermann, C. and F. Kindermann, *Multidimensional spline interpolation: Theory and applications.* Computational Economics, 2007. **30**(2): p. 153-169.

31. Lu, Y., et al., *A B-spline curve extension algorithm.* Science China Information Sciences, 2016. **59**(3): p. 32103.

32. Newbery, A. and T.S. Garrett, *Interpolation with minimized curvature.* Computers & Mathematics with Applications, 1991. **22**(1): p. 37-43.

CHAPTER 3: INTERPOLATION IN FREE ENERGY CALCULATIONS

3.1 INTRODUCTION

Free energy methods based on statistical mechanics require approximations, as the number of possible configurations can be infinite. For an approximated system, it is necessary to make a connection between discrete energy states. This is where interpolation methods can be useful.

The concept behind interpolation stems from the need to derive thermodynamic properties from free energy surfaces. An energy surface with discontinuities between its energy states cannot be differentiated without interpolation methods.

This chapter reviews the free energy methods that calculate relative free energy differences, namely Free Energy Perturbation and Thermodynamic Integration, as well as the flat histogram methods that calculate absolute free energies from the density of states. Particular focus is on their implementations of various interpolation methods.

3.2 FREE ENERGY DEFINITIONS

The free energy of a thermodynamic system is the amount of internal energy available for work. An example of this is when a person exerts a force to push an object, using the internal energy coming from metabolism. Although energy is conserved, not all of it goes into pushing the object. In fact, part of the internal energy gets released in the form of heat. The remainder that is used to push the object is the free energy.

Free energy is an essential component of reaction mechanisms, providing valuable information about other thermodynamic properties of the system. The minimum energy pathway is the most important information, inter alia, from which the reaction mechanism can be constructed. Thermodynamic properties such as the rate of reactions, equilibrium constants, solvation properties, drug binding properties and many more can be calculated from the free energy calculations[54, 55].

There are different thermodynamic variables for free energy depending on the type of constraint applied to the system.

Helmholtz free energy (A)[56] is the maximum work under constant volume and temperature. Its state function can be expressed:

$$\mathrm{A} = \mathrm{U} - \mathrm{TS} \quad (3.1)$$

where **U** is the internal energy, **T** is the absolute temperature and **S** is the entropy. According to the first law of thermodynamics, the change in internal energy is

$$dU = dQ + dW \quad (3.2)$$

Where *dQ* is the energy term of heat and *dW* is the work done on the system. According to the second law of thermodynamics in case of a reversible process

$$dQ = TdS \quad (3.3) \qquad dW = -pdV \quad (3.4)$$

Substituting equations 3.3 and 3.4 into 3.2

$$\mathrm{dA} = -\mathrm{SdT} - \mathrm{PdV} \quad (3.5)$$

where A=f(T,V)

Gibbs free energy (G)[57] is the maximum work under constant pressure and temperature. The state function for the Gibbs free energy is

$$\mathrm{G} = \mathrm{H} - \mathrm{TS} \quad (3.6)$$

where H is the enthalpy with $\mathrm{H} = \mathrm{U} + \mathrm{pV}$.

Substituting equations (3.3) and (3.4) and taking derivatives on both sides;

$$\mathrm{dH} = \mathrm{TdS} + \mathrm{VdP} \quad (3.7)$$

Finally, substituting this equation into the equation for Gibbs Free energy

$$dG = VdP - SdT \quad (3.8)$$

where G = f(T,P)

For experimental studies in laboratories, it is much easier to manipulate volume rather than pressure, therefore, **G** is more suitable. On the other hand, computational studies involve systems in constant unit volumes with periodic boundary conditions (PBC)[58]. Therefore, the calculations are presented in **(A)**

3.3 FREE ENERGY CALCULATIONS IN STATISTICAL MECHANICS

Statistical mechanics methods are used to explain the macroscopic behaviour of thermodynamic systems using the microscopic properties. Combining the rules of thermodynamics with statistics, statistical mechanics has become an essential component of free energy calculations.

3.3.1 PARTITION FUNCTIONS

Partition functions are the key elements connecting microscopic properties to macroscopic properties.

In a large thermodynamic system, there are certain microstates that can be occupied by the molecules. A partition function defines the number of configurations found in an energy state. Assuming that the system has a fixed volume with a fixed number of particles in contact with a temperature bath (NVT- canonical ensemble), partition functions can be added up to generate the canonical partition function Z in terms of each microstate s and the energy of that micro-state E_s [59]

$$Z = \sum_s e^{-\beta E_s} \quad (3.9)$$

with

$$\beta = \frac{1}{k_B T} \quad (3.10)$$

where k_B is the Boltzmann's constant.

For continuous microstates, the canonical partition function can be re-written in terms of the position r as follows:

$$Z = \int dr\, e^{-\beta E(r)} \quad (3.11)$$

Using the canonical partition function, one can calculate the expectation value of any thermodynamic observable *O(r)* (particle number, pressure, temperature, density, potential energy, etc.)

$$< O >= \frac{\int dr\, O(r) e^{-\beta E(r)}}{\int dr\, e^{-\beta E(r)}} \quad (3.12)$$

This equation is significant because it calculates the expected average value of that observable at a particular energy level.

3.4 COMPUTER SIMULATIONS IN FREE ENERGY CALCULATIONS

Calculating the absolute free energy of a system requires knowledge of the partition functions for all the energy states, which is near impossible. However, it is possible to calculate the free energy difference. This property has gained significance in the construction of the reaction mechanisms.

The expectation value of a particular state with energy U is

$$U = \frac{\iint E(r) \exp(-\beta E(r))\, dr}{\iint \exp(-\beta E(r))\, dr} = \iint E(r) P(r) dr \quad (3.13)$$

Where *P* is the probability of being at a particular point in phase space and is denoted as

$$P(r) = Z^{-1} \exp(-\beta E(r)) \quad (3.14)$$

with

$Z = \int dr\, e^{-\beta E(r)}$ being the partition function for the system.

A can be expressed in terms of the partition function as

$$A = -k_B T ln Z \quad (3.15)$$

The advantage of calculating free energy difference between two states is that one can take the ratio of the two corresponding partition functions.

$$\Delta A = A_1 - A_0 = -k_B T ln\left(\frac{Z_1}{Z_0}\right) \quad (3.16)$$

Calculating free energy differences requires an initial structure, a force field and a sufficient scanning of the phase space. Because the phase space is vast in large systems, the computation of the energy values at every state requires an enormous amount of time. However, a significant portion of the phase points would have unfavourable positional coordinates. Therefore, the calculations can be simplified by picking only the physically favourable points of the phase space. [60]

Monte Carlo (MC) and Molecular Dynamics (MD) are computer simulation methods that discover high probability and low energy points of the system to sample all the probable regions in the phase space.

MC[14] sampling obtains an ensemble of the probable configurations by applying a random walk in between these configurations. At each MC simulation, a random move is attempted towards another energy state. If the energy of the attempted state is more favourable, the move is accepted. If the attempted state has a less favourable energy, the acceptance of the move depends on the transition probability, which decreases as the energy of the attempted state increases. The procedure is repeated until all the possible configurations are visited.

Classical MD[58] simulations facilitate the time-dependent evolution through trajectories derived from Newton's equations. MD simulation is essential in calculating time-dependent observables, such as those measured in drug-binding studies, pharmacokinetics, and pharmacodynamics[61].

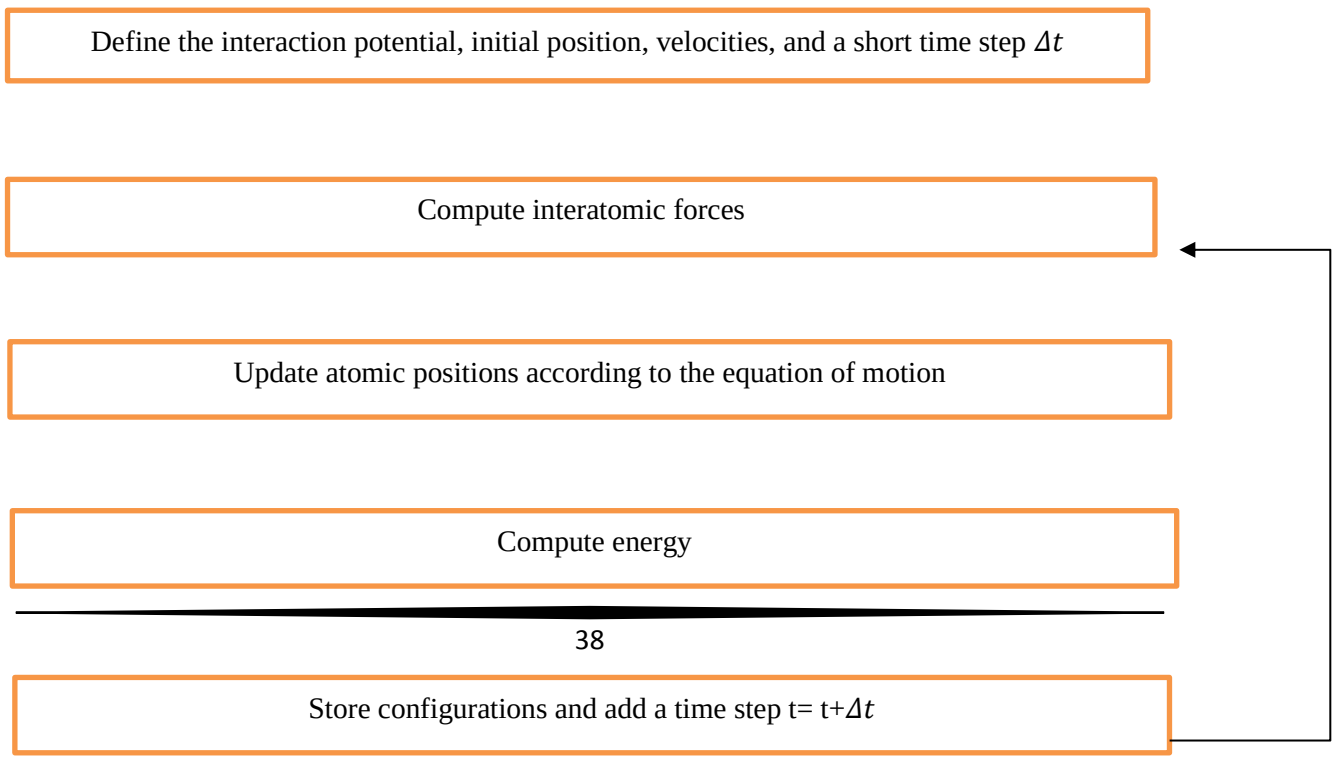


Figure 3.1: A flow diagram of a Classical Molecular Dynamics algorithm that solves Newton's equations of motion

If the average energies are computed iteratively at certain intervals, the energy difference can be determined as an average over the probable configurations.

$$\langle U\rangle_1 - \langle U\rangle_0 = \frac{1}{M_1}\sum_i^{M_1} E(t_i) - \frac{1}{M_0}\sum_i^{M_0} E(t_i) = \langle E\rangle_1 - \langle E\rangle_0 \quad (3.17)$$

Where M_0 and M_1 are the total number of times the energy values were sampled.

3.5 FREE ENERGY METHODS AND INTERPOLATION

The issue with the aforementioned method is that the energy values fluctuate for large systems. Therefore, the standard deviation of the ensemble averages would also be large and high ensemble errors of energies would occur[60]. Free energy methods can be used to make approximations by gradually calculating the ensemble averages between two energy levels that are very close to each other.

3.5.1 FREE ENERGY PERTURBATION

It is often difficult to extract the ratio of two partition functions, but a partition function can be rearranged in terms of another to make the calculation easier.

Given two partition functions $Z_0 = \int dr\, e^{-\beta E_0(r)}$ and $Z_1 = \int dr\, e^{-\beta E_1(r)}$

Z_1 can be re-written by multiplying the integral by 1 as follows:

$$Z_1 = \int dr\, e^{-\beta E_1(r)} e^{-\beta\left(E_0(r)-E_0(r)\right)} \rightarrow \int dr\, e^{-\beta E_0(r)} e^{-\beta\left(E_1(r)-E_0(r)\right)} \quad (3.18)$$

Then, the ratio of Z_1/Z_0 becomes

$$\frac{Z_1}{Z_0} = \frac{\int dr\, e^{-\beta E_0(r)} e^{-\beta\left(E_1(r)-E_0(r)\right)}}{\int dr\, e^{-\beta E_0(r)}} \quad (3.19)$$

The equation above has the form $< O >= \frac{\int dr\, O(r) e^{-\beta E(r)}}{\int dr\, e^{-\beta E(r)}}$; therefore, it is equal to the expectation value of the free energy difference averaged over the zeroth state.

$$\frac{Z_1}{Z_0} = < e^{-\beta(E_1(r)-E_0(r))} >_0 \quad (3.20)$$

The equation above was first discovered by Zwanzig[62] in 1954 and the method was called Free Energy Perturbation (FEP)

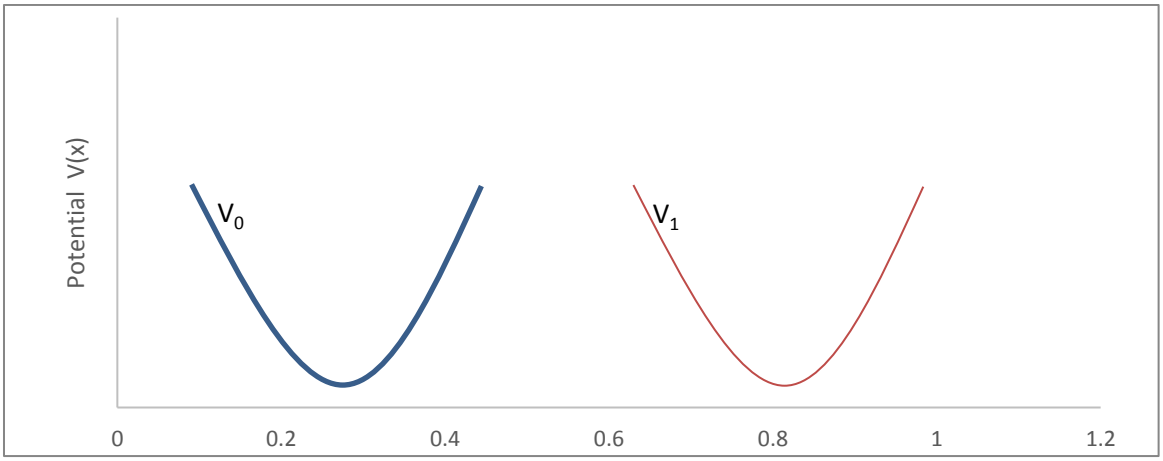
The equation tells us that the energy difference between two states can be written in the form of an averaged function obtained by sampling only the initial state.

$$\Delta A(A \to B) = A_B - A_A = -k_B T ln < e^{-\beta(E_1(r)-E_0(r))} >_0 \quad (3.21)$$

One can do a simulation of the system 0 and accumulate the average $e^{-\beta(E_1(r)-E_0(r))}$. This is followed by taking the natural logarithm of the observable and multiplying by $-k_BT$.

3.5.2 COUPLING PARAMETER

This calculation is feasible when there is a high overlap between the two states. Otherwise, the sampling probabilities of the energy levels contradict the expectation values[63]. For example, a small overlap region is sampled rarely even if the expectation value is high, resulting in a low probability. This affects the accuracy of free energy calculations negatively.



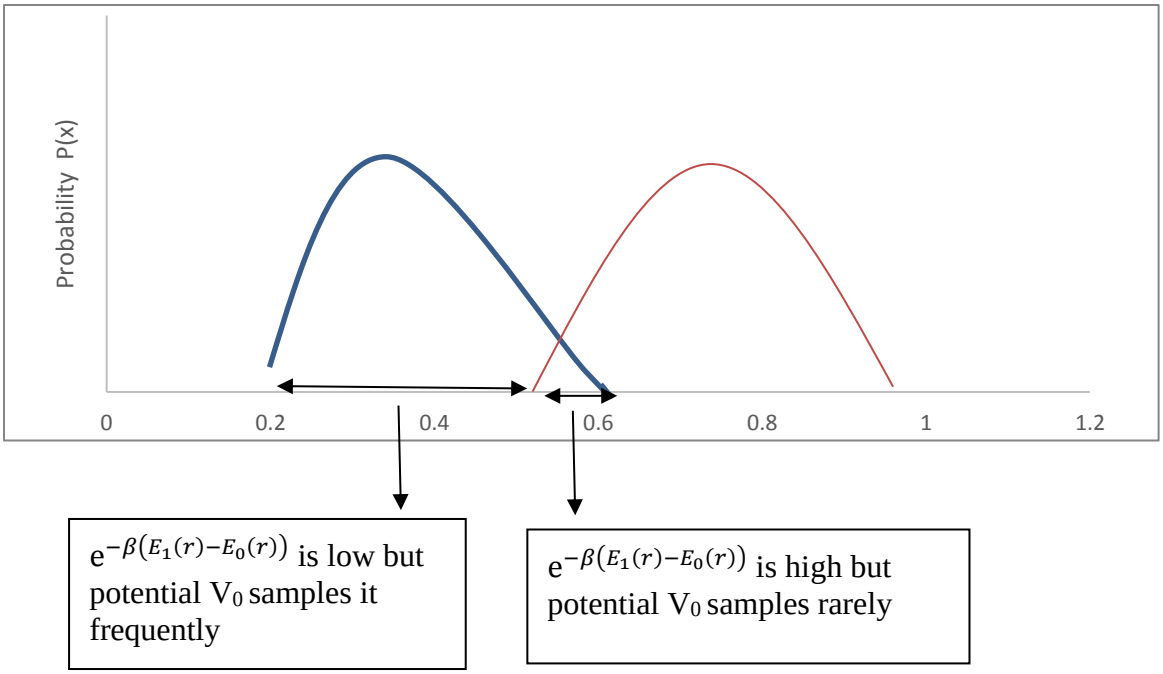


Figure 3.2: A diagram showing the deficiencies in the FEP method when the potentials have little to no overlap

Considering that free energies are state functions and that they do not depend on how the system acquired a particular state, one can take any path between the two states. If the path is broken down to smaller windows, intermediate states that have bigger overlaps can be generated– whether they are physically meaningful or not[64].

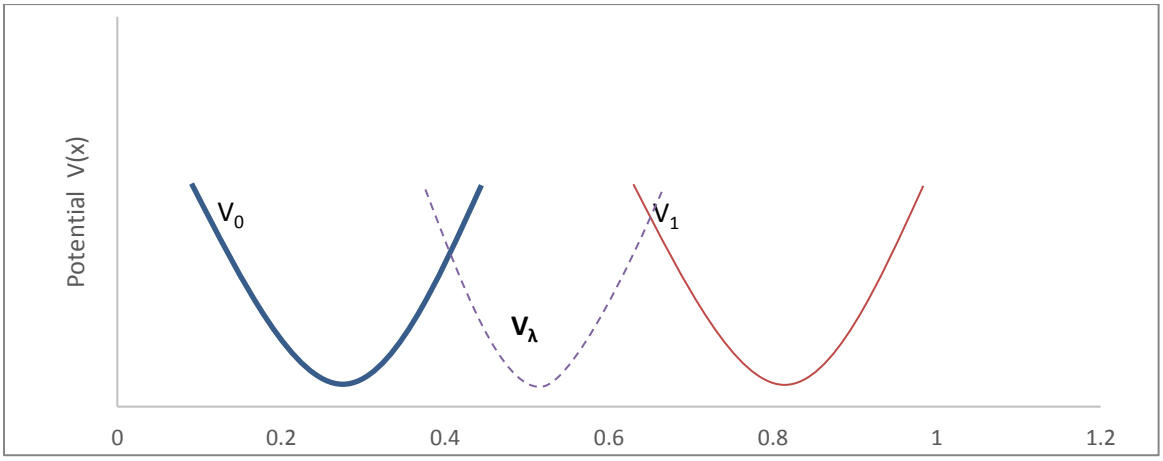


Figure 3.3: A diagram showing the FEP performed with a coupling parameter that generates middle state(s) with higher overlaps with both potentials V_0 and V_1

The smaller parts are expressed in terms of a coupling parameter, λ, that ranges from 0 to 1. In that case, the energy can be written as a function of λ. A common method for generating

intermediate states is to use linear interpolation to interpolate between the energy states E_a and E_b as follows:

$$E(\lambda) = \lambda E_B + (1-\lambda)E_A \quad (3.22)$$

Then, the equation for the free energy difference can be re-written as

$$\Delta A = A_B - A_A = \sum_0^1 k_B T ln < \exp\left(\frac{E_{\lambda+d\lambda} - E_\lambda}{k_B T}\right) >_\lambda \quad (3.23)$$

Interpolation of intermediate states through a coupling parameter is computationally more feasible than following more complicated physical pathways.

Linear interpolation was used in some of the previous FEP studies. In a previous work, point charges were linearly interpolated to compute side chain charges for intermediate states. [65]

3.5.3 THERMODYNAMIC INTEGRATION

The Thermodynamic Integration (TI) method is a derivation of the free energy perturbation method. It computes the free energy difference between two states by changing the coupling parameter slowly enough so that the system is in equilibrium and reversible at all times[66].

The free energy can be written as a function of the coupling parameter as follows:

$$A(\lambda) = -k_B T ln Q(\lambda) \quad (3.24)$$

Taking the derivative with respect to the coupling parameter gives

$$\frac{\partial A(\lambda)}{\partial \lambda} = -\frac{k_B T}{Q}\frac{\partial Q(\lambda)}{\partial \lambda} = -\frac{k_B T}{Z}\frac{\partial Z(\lambda)}{\partial \lambda} \quad (3.25)$$

From the equation $Z = \int dr\, e^{-\beta E(r)}$;

$$\frac{\partial Z(\lambda)}{\partial \lambda} = \int dr\, \frac{\partial e^{-\beta E_\lambda(r)}}{\partial \lambda} = \int dr \left(-\beta \frac{\partial E_\lambda}{\partial \lambda}\right) e^{-\beta E_\lambda(r)} \quad (3.26)$$

Substituting into the equation 3.25 and realizing that β and $k_B T$ cancel each other;

$$\frac{\partial A(\lambda)}{\partial \lambda} = -\frac{k_B T}{Z}\frac{\partial Z(\lambda)}{\partial \lambda} = \frac{\int dr \left(\frac{\partial E_\lambda}{\partial \lambda}\right) e^{-\beta E_\lambda(r)}}{\int dr\, e^{-\beta E(r)}} \quad (3.27)$$

The expression on the right side is the expectation value of the observable $< \frac{\partial E_\lambda}{\partial \lambda} >_\lambda$, therefore,

$$\frac{\partial A(\lambda)}{\partial \lambda} = < \frac{\partial E_\lambda}{\partial \lambda} >_\lambda \quad (3.28)$$

This equation relates the free energy change to the change in potential. If the intervals of the coupling parameter were taken arbitrarily small, one could integrate the expectation value over the interval between 0 and 1;

$$A(\lambda) = \int_0^1 < \frac{\partial E_\lambda}{\partial \lambda} >_\lambda d\,\lambda \quad (3.29)$$

This equation, theorised by Kirkwood[67] in 1935, gives rise to the Thermodynamic Integration (TI) method. Finally, recalling the equation $E(\lambda) = \lambda \mathrm{E_B} + (1-\lambda)\mathrm{E_A}$, one can rewrite the inside of the integral as follows:

$$\frac{\partial E_\lambda}{\partial \lambda} = E_B - E_A \rightarrow A(\lambda) = \int_0^1 < V_1 - V_0 >_\lambda d\,\lambda \quad (3.30)$$

By changing the coupling parameter slowly along the interval, one can define a thermodynamic path or a gradient of $< \frac{\partial E_\lambda}{\partial \lambda} >_\lambda$ for each interval between two states and integrate over each interval along that path. This is advantageous because one can calculate the energy directly from the work required to change the coupling parameter[68].

Similar to the FEP, a more recent version of the TI method also benefits from the interpolation of the coupling parameter λ to generate intermediate states. In the method Parameter-Interpolated Thermodynamic Integration (PI-TI) [69], the potential energy surface is expressed in terms of the coupling value of λ

$$U_{PI(\lambda)} = U\left(P^\lambda\right) \quad (3.31)$$

$$P^\lambda = P^0 + \lambda(\mathrm{P^1} - P^0) \quad (3.32)$$

Recent studies adopted this method to transform a system from an initial state to a final state along the atomic charges [13].

$$q_{a(\lambda)} = q_{a(0)} + \lambda\left(\mathrm{q_{a(1)}} - q_{a(0)}\right) \quad (3.33)$$

Bond and angle force constants and equilibrium constants were also interpolated at intermediate states in the same study. Different interpolation methods such as natural cubic spline, linear interpolation, and B-spline interpolation were employed according to the continuity level required. It was shown by the developers of the method [13] that alchemical pathways generated from PI-TI were more linear and uniform compared to the TI algorithm without interpolation. In other words, transforming a system between two states required fewer intermediate states in PI-TI compared to the number required in the standard TI method.

In a study by Shyu and Ytreberg, interpolation methods were compared for the accurate estimation of free energy differences from the TI-generated data. The interpolation methods were used to calculate the derivatives $\frac{\partial A(\lambda)}{\partial \lambda} = < \frac{\partial E_\lambda}{\partial \lambda} >_\lambda$ with minimum integration errors. Test systems with analytical solutions were interpolated using polynomial and cubic spline interpolations. It was found that polynomial interpolation approximated the energy surface more accurately for data points less than or equal to 12; for a higher number of data points $\left\{\lambda, \frac{\partial A(\lambda)}{\partial \lambda}\right\}$, spline interpolation provided more stable results. Furthermore, Chebyshev nodes were used to adjust the distances between data points and to enhance accuracy. [70]

3.5.4 FLAT HISTOGRAM METHODS

Other than the initial and the final states, knowing the exact reaction pathway along the potential energy surface could provide valuable mechanistic insight into reactions. The second rule of thermodynamics states that the energy profile of a chemical reaction tends to decrease and approach a minimum value. This is similar to getting to the other side of a mountain. Instead of ascending and descending, one spends much less energy by going around the mountain along the least strenuous path.

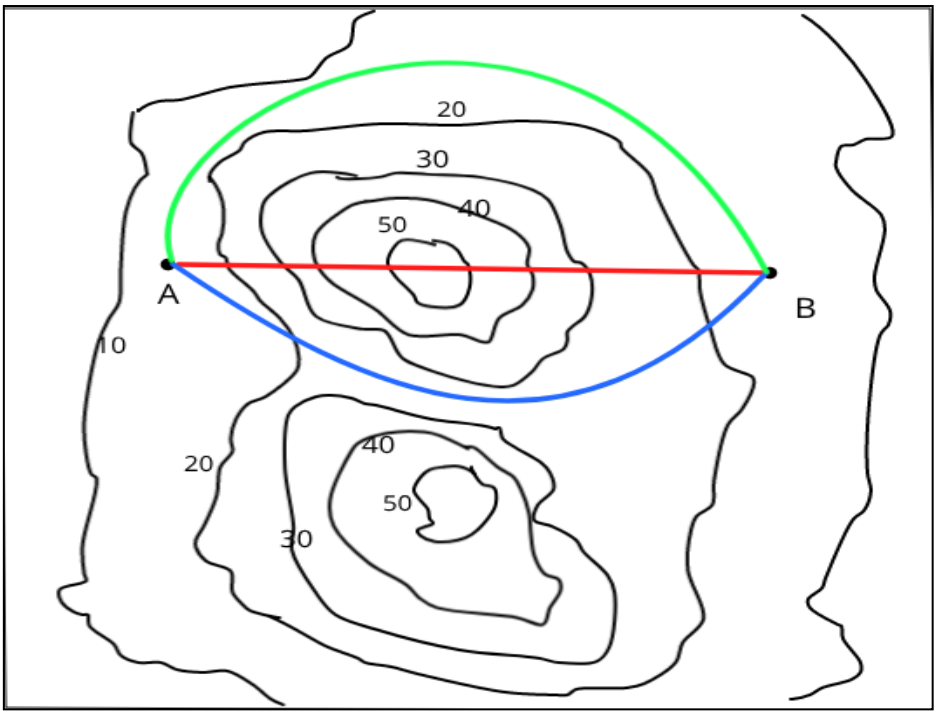


Figure 3.4: A diagram of different possibilities for getting from point A to B. If the diagram was a potential energy surface, the green path would be the minimum free energy pathway.

The knowledge of the minimum free energy path is necessary to find the probability of being at a particular point along that path. This becomes crucial for specific intermediate states such as the absolute minimum or a transition state with a high energy barrier. To handle such chemistry problems, it is more practical to express free energy in terms of a reaction coordinate[71]. A reaction coordinate can be any bond distance, angle or dihedral that changes throughout the chemical process. When the energy of the system is calculated as a function of the reaction coordinate, it is called the potential of mean force (PMF). The general formula for the PMF is

$$A(q) = -k_B T ln P(q) \quad (3.34)$$

Where q is the reaction coordinate of interest and P is the probability of that reaction coordinate to have a particular value. Then, the free energy of moving from one position to another along that reaction coordinate is calculated as follows:

$$\Delta A(q) = -k_B T ln\left(\frac{P(q_1)}{P(q_0)}\right) \quad (3.35)$$

This equation is useful to make a connection between the probability distribution of a reaction coordinate and the free energy of the movement along that coordinate.

The probabilities can be evaluated by dividing the reaction coordinate into discrete bins so that the snapshots of conformations can be grouped.

For the construction of the free energy diagram for a system, it is important to obtain adequate sampling of all relevant energy levels. The requirement for a reasonable sampling is to have a sampling ratio of 1:50 between the regions with the lowest and the highest probabilities[72]. However, it is often not possible to take a sufficient number of snapshots from regions with higher energy because the probability of sampling that region is very low. Many chemical reactions involving the breaking and forming of bonds have high activation energies that cannot be reached with the available thermal energy during an MD simulation[73].

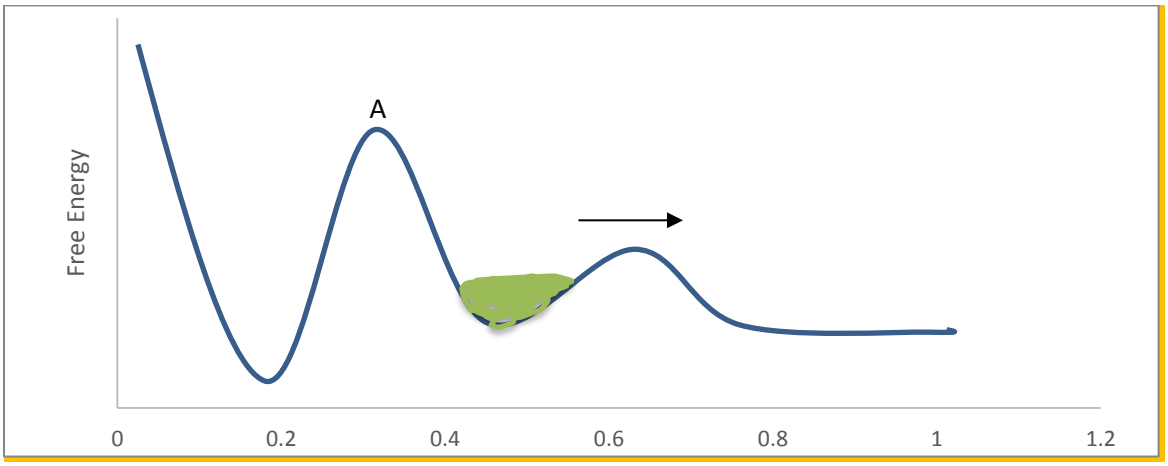


Figure 3.5: Sketch of a free energy surface sampled with Classical MD. The rare event of A is unlikely to be sampled.

The probability of sampling a configuration with energy *E* is

$$P_E = \rho(E)p(E) \quad (3.36)$$

where *ρ(E)* denotes the number or the density of the states with energy *E* and *p(E)* is the probability of each state with the energy *E*.

Boltzmann distribution measures the probability *p(E)* as a function of the state's energy and the overall temperature, where each probability is proportional in the form

$$p(E_i) \propto e^{-\frac{E_i}{kT}} \quad (3.37)$$

where $p(E_i)$ is the probability that the system is in state i, and E_i is the energy of state i. Equation 3.37 implies that the states with higher energies will always have a lower probability of being occupied compared to the states with lower energy. As a result, critical regions with extremely high energies will always be seldom sampled by Boltzmann distribution[15]. Therefore, the study turns the focus towards non-Boltzmann sampling methods that generate flat histograms.

The main idea behind flat histogram methods is to formulate a linear relationship between p(E) and the frequency of the states such that

$$p(E) \propto \frac{1}{\rho(E)} \quad (3.38)$$

Thus, one can obtain a constant probability, P_E, for visiting all the energy levels of E, forming a flat histogram. If equal amounts of times were spent at the lowest and the highest energy levels, crossing the energy barriers would become easier.

The earliest version of flat histogram methods was developed by Wang-Landau[74], where a Monte Carlo-based random walk was performed to acquire all the available energy levels. The visiting of all the energy levels was made possible by a modified function that was updated and reduced at each step to reduce the errors. This was continued until all the accessible states were visited. However, this method fell short when it came to systems where two or more configuration spaces were separated by high energy barriers.

In the meantime, the idea of adding an external potential term was employed in a method called umbrella sampling[75]. The method iteratively added quadratic harmonic potentials, which added up to the inverse of the potential of the mean force so that the energy barrier would be cancelled. Similar methods implemented different forms of biasing potentials. The local elevation method[17] used a memory dependent repulsive potential, whereas Metadynamics[76] implemented Gaussian functions to overcome the barrier. More advanced methods like Adaptive Umbrella Sampling[19] and Free Energies from Adaptive Reaction Coordinate Forces (FEARCF) [20] came up with the idea of a biasing potential term that adapted according to all the sampling results obtained up to that point.

Later on, it was discovered that a more realistic sampling required taking samples at multiple points of the reaction coordinate in separate simulations. This facilitated the invention of reweighting methods that combined the probability distributions from multiple simulations.

Ferrenberg-Swendsen Method[77] and the Weighted Histogram Analysis Method (WHAM)[78] were used to combine multiple probability distributions obtained at different regions of the reaction space.

Flat histograms require interpolation of data for ensuring continuity since a histogram consists of discrete bins. In Wang- Landau-based simulations with long-range spin models, unequally distributed energy levels and the gaps around the ground state required either a linear interpolator or a B-spline interpolator. It was observed that the acceptance rate was unrealistically reduced without any interpolator. In the same study, the microcanonical inverse temperature was calculated by taking the derivative of the microcanonical entropy with respect to the energy using spline interpolation. Because the density of states obtained in the early iterations was jagged, spline interpolation was needed to obtain a smoother density of states proportional to the microcanonical entropy. [79]

Interpolation was also used in improving the accuracy of biasing potentials. In a study that combined adaptive biasing with Hamiltonian Replica Exchange[80], PMFs were linearly interpolated to obtain the negative mean forces along the reaction coordinate between points of measurement. It was found that the discontinuities arising from linear interpolation did not disrupt the results[81]. By the same logic, the FEARCF method calculated driving forces – which acted on the atoms to update their positions for further simulation – were derived by using cubic spline functions to interpolate the PMFs. [20]

When calculating the free energy from a reaction coordinate λ, multiple simulations can be run with different choices of λ. Then, free energy at a value of λ other than those chosen for simulation can be interpolated.[82]

Chapter 4 provides a more detailed explanation of the flat histogram methods and points out the objectives of the study involving these methods.

3.6 REFERENCES

1. Shukla, D., et al., *Activation pathway of Src kinase reveals intermediate states as targets for drug design.* Nature communications, 2014. **5**: p. 3397.

2. Sheppard, D., R. Terrell, and G. Henkelman, *Optimization methods for finding minimum energy paths.* The Journal of chemical physics, 2008. **128**(13): p. 134106.

3. Helmholtz, H., *On the thermodynamics of chemical processes.* Physical Memoirs Selected and Translated from Foreign Sources, 1882. **1**: p. 43-97.

4. Gibbs, J.W., *A Method of Geometrical Representation of the Thermodynamic Properties by Means of Surfaces.* Transactions of Connecticut Academy of Arts and Sciences, 1873: p. 382-404.

5. Alder, B.J. and T.E. Wainwright, *Studies in molecular dynamics. I. General method.* The Journal of Chemical Physics, 1959. **31**(2): p. 459-466.

6. Ott, J.B. and J. Boerio-Goates, *Chemical Thermodynamics: Advanced Applications: Advanced Applications*. 2000: Elsevier.

7. Chatfield, D., *Christopher J. Cramer: Essentials of Computational Chemistry: Theories and Models.* Theoretical Chemistry Accounts: Theory, Computation, and Modeling (Theoretica Chimica Acta), 2002. **108**(6): p. 367-368.

8. Hastings, W.K., *Monte Carlo sampling methods using Markov chains and their applications.* 1970.

9. Paquet, E. and H.L. Viktor, *Molecular dynamics, monte carlo simulations, and langevin dynamics: a computational review.* Biomed Res Int, 2015. **2015**: p. 183918.

10. Zwanzig, R.W., *High-temperature equation of state by a perturbation method. I. Nonpolar gases.* The Journal of Chemical Physics, 1954. **22**(8): p. 1420-1426.

11. Pohorille, A., C. Jarzynski, and C. Chipot, *Good practices in free-energy calculations.* The Journal of Physical Chemistry B, 2010. **114**(32): p. 10235-10253.

12. Bhati, A.P., et al., *Rapid, accurate, precise, and reliable relative free energy prediction using ensemble based thermodynamic integration.* Journal of chemical theory and computation, 2016. **13**(1): p. 210-222.

13. Meng, Y., D. Sabri Dashti, and A.E. Roitberg, *Computing alchemical free energy differences with Hamiltonian replica exchange molecular dynamics (H-REMD) simulations.* Journal of chemical theory and computation, 2011. **7**(9): p. 2721-2727.

14. Apte, P.A. and I. Kusaka, *Direct calculation of solid-liquid coexistence points of a binary mixture by thermodynamic integration.* The Journal of chemical physics, 2005. **123**(19): p. 194503.

15. Kirkwood, J.G., *Statistical mechanics of fluid mixtures.* The Journal of Chemical Physics, 1935. **3**(5): p. 300-313.

16. van Gunsteren, W.F., X. Daura, and A.E. Mark, *Computation of free energy.* Helvetica Chimica Acta, 2002. **85**(10): p. 3113-3129.

17. Giese, T.J. and D.M. York, *A GPU-accelerated parameter interpolation thermodynamic integration free energy method.* Journal of chemical theory and computation, 2018. **14**(3): p. 1564-1582.

18. Shyu, C. and F.M. Ytreberg, *Use of polynomial interpolation to reduce bias and uncertainty of free energy estimates via thermodynamic integration.* arXiv preprint arXiv:0809.0882, 2008.

19. Quaytman, S.L. and S.D. Schwartz, *Reaction coordinate of an enzymatic reaction revealed by transition path sampling.* Proceedings of the National Academy of Sciences, 2007. **104**(30): p. 12253-12258.

20. Beveridge, D.L. and F. DiCapua, *Free energy via molecular simulation: applications to chemical and biomolecular systems.* Annual review of biophysics and biophysical chemistry, 1989. **18**(1): p. 431-492.

21. Murthy, K. *Non-Boltzmann Ensembles and Monte Carlo Simulations*. in *Journal of Physics: Conference Series*. 2016. IOP Publishing.

22. Landau, D., S.-H. Tsai, and M. Exler, *A new approach to Monte Carlo simulations in statistical physics: Wang-Landau sampling.* American Journal of Physics, 2004. **72**(10): p. 1294-1302.

23. Kästner, J., *Umbrella sampling.* Wiley Interdisciplinary Reviews: Computational Molecular Science, 2011. **1**(6): p. 932-942.

24. Huber, T., A.E. Torda, and W.F. Van Gunsteren, *Local elevation: a method for improving the searching properties of molecular dynamics simulation.* Journal of computer-aided molecular design, 1994. **8**(6): p. 695-708.

25. Barducci, A., M. Bonomi, and M. Parrinello, *Metadynamics.* Wiley Interdisciplinary Reviews: Computational Molecular Science, 2011. **1**(5): p. 826-843.

26. Mezei, M., *Adaptive umbrella sampling: Self-consistent determination of the non-Boltzmann bias.* Journal of Computational Physics, 1987. **68**(1): p. 237-248.

27. Naidoo, K.J., *FEARCF a multidimensional free energy method for investigating conformational landscapes and chemical reaction mechanisms.* Science China Chemistry, 2011. **54**(12): p. 1962-1973.

28. Ferrenberg, A.M. and R.H. Swendsen, *New Monte Carlo technique for studying phase transitions.* Physical review letters, 1988. **61**(23): p. 2635.

29. Kumar, S., et al., *The weighted histogram analysis method for free-energy calculations on biomolecules. I. The method.* Journal of computational chemistry, 1992. **13**(8): p. 1011-1021.

30. Reynal, S. and H.-T. Diep, *Fast flat-histogram method for generalized spin models.* Physical Review E, 2005. **72**(5): p. 056710.

31. Affentranger, R., I. Tavernelli, and E.E. Di Iorio, *A novel Hamiltonian replica exchange MD protocol to enhance protein conformational space sampling.* Journal of Chemical Theory and Computation, 2006. **2**(2): p. 217-228.

32. Zeller, F. and M. Zacharias, *Adaptive biasing combined with Hamiltonian replica exchange to improve umbrella sampling free energy simulations.* Journal of chemical theory and computation, 2014. **10**(2): p. 703-710.

33. Souaille, M. and B. Roux, *Extension to the weighted histogram analysis method: combining umbrella sampling with free energy calculations.* Computer physics communications, 2001. **135**(1): p. 40-57.

CHAPTER 4: FLAT HISTOGRAM METHODS FOR NON-BOLTZMANN SAMPLING

4.1 OVERVIEW

In Chapter 3, the concepts of free energy calculations and flat histogram methods were introduced. The correct application of flat histogram methods requires an in-depth understanding of efficient sampling through numerical approximations.

Flat histogram methods are essential because the readily available thermal energy of the system is not sufficient to cross the high energy barriers. This requires an additional biasing potential to guide the trajectories towards poorly sampled areas over time. Some flat histogram methods do this by adding potential terms to the Hamiltonian definition to manipulate the probability distribution, while others derive forces to act on the atoms to bias the atomic positions into the regions of rare events. Once all the regions are equally sampled (hence the name “flat histogram”) the probability distribution becomes the inverse of the accumulated free energy surface. This enables a random walk throughout the entire surface without running into any barrier. Consequently, the free energy profile of the chemical process can be constructed.

Free Energies from Adaptive Reaction Coordinate Forces (FEARCF)[20] developed by Professor Kevin Naidoo fits into the latter category of deriving forces from energy. It stands out among the other methods for its simplicity, time-saving properties, and dimensionality. However, it is also worth noting how the method benefits from interpolation methods to derive forces. Therefore, it constitutes a perfect case for comparing various interpolation methods.

Before mentioning the formulation of the FEARCF method, it is necessary to talk about certain flat histogram methods that planted the underlying ideas. This chapter will focus on the first method that generated a reasonable sampling and further improvements made to reach the regions with high activation barriers. Following this, the FEARCF method will be explained. Particular focus will be placed on how the method successfully employs interpolation methods as a way of approximating derivatives of numerical data.

4.2 WANG-LANDAU SAMPLING

One of the earlier mentions of flat histogram methods was in 2000 by Wang and Landau[74], who estimated the density of states *g(E)* by multiple random walks for different ranges of energy. In this Monte Carlo algorithm, the probability distribution is modified until a flat histogram is obtained, then it converges to its true value. The random walk is performed on an energy space that is the reciprocal of the density of states.

In the beginning, the probability density of all energies is assumed to be 1 since density of states is unknown at that point. Using an Ising model[83], a random walk is performed by randomly flipping spins and comparing the energies before and after. The probability of transitioning from one energy level to the other is;

$$p(E_1 \rightarrow E_2) = \min\left[\frac{g(E_1)}{g(E_2)}, 1\right] \quad (4.1)$$

Once the transition occurs, the new density $g(E_2)$ is updated through multiplication by a modification factor, which is usually the Euler's Number, *e*.

$$g(E_2)^* = g(E_2)f \quad (4.2)$$

The multiplication by the factor f makes the process memory-dependent. When there is a proposal to transition into an already-visited state, it is refused according to the Metropolis-Hastings algorithm[14]. This encourages the system to explore new energy states.

The enhancement of the density of states allows the random walk to reach all the energy levels and obtain a flat accumulated histogram.

At the end of the first series of random walks, the multiplying factor is reduced by taking its square root to reduce the error.

$$f_1 = \sqrt{f_0} \quad (4.3)$$

The histogram is reset to *H(E) = 0* in the meantime and the random walk is restarted with the new value of f_1. The reduction process continues until *f* approaches 1, which means that convergence has occurred.

The relationship between two energy levels in the case of successful convergence is as

follows:

$$\frac{1}{g(E_i)}p(E_i \to E_{i+1}) = \frac{1}{g(E_{i+1})}p(E_{i+1} \to E_i) (4.4)$$

With all the density of the states, the free energy can be calculated

$$A(T) = -k_B T ln\left(\sum_E g(E)e^{-\beta E}\right) (4.5)$$

The Wang-Landau Sampling method was used in constructing the folding mechanism of proteins as well as the solution of numerical integrals. However, it is important to reduce *f* correctly to avoid errors in the individual energy calculations and the mean value of the energy. [84, 85]

4.3 EMPLOYING A BIASING POTENTIAL

4.3.1 UMBRELLA SAMPLING

One of the first approaches to overcome the sampling deficiencies was the Umbrella Sampling method[16], in which a biasing potential called **umbrella potential** was applied. The biasing potential is a function of the reaction coordinate *q*. When applied, it forces the sampling away from its previous location, so areas that were not sampled become accessible.

If the PMF had an analytical function, one would be able to take its inverse and add it as a biasing potential so that the sampling property is uniform throughout *q*. The alternative solution is to add small umbrella potentials iteratively until a flat histogram is obtained. Quadratic potentials are very suitable for this purpose.

$$U(q, q_0) = \frac{1}{2}k(q - q_0)^2 (4.6)$$

In this quadratic equation, q_0 is a point on the reaction coordinate surface in the direction of the sampling

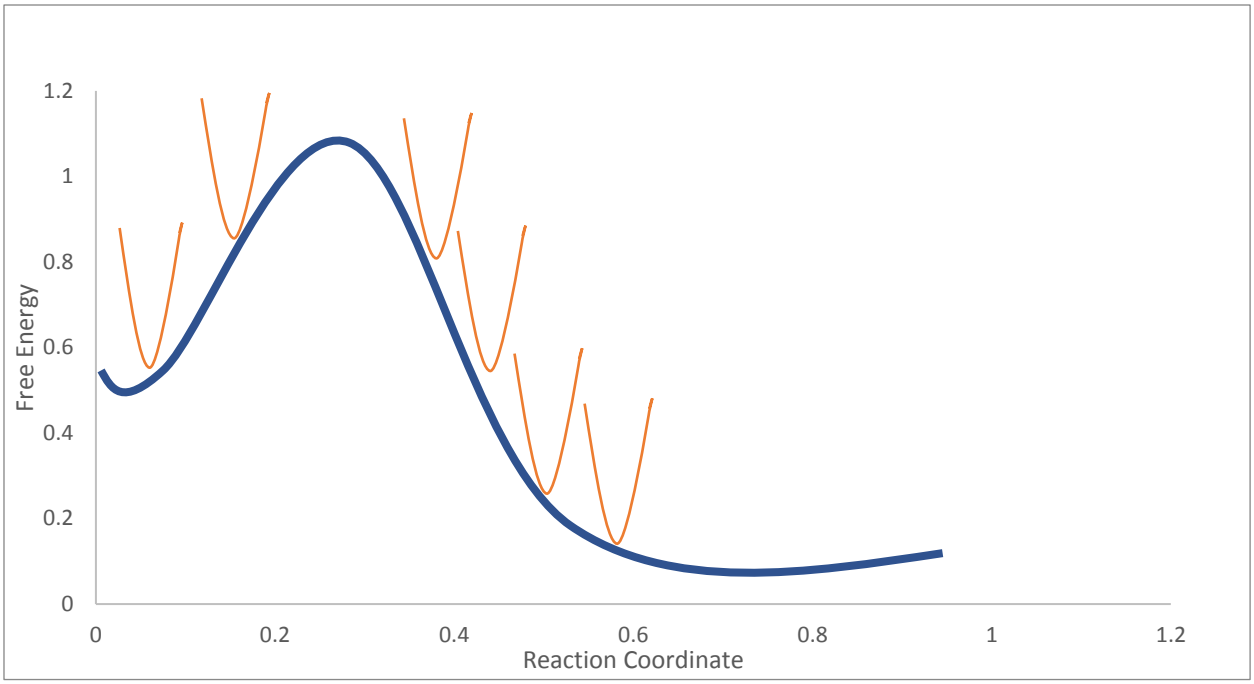


Figure 4.1: A graph showing umbrella potentials applied on the free energy surface along the reaction coordinate

With the addition of this quadratic potential, the sampling on the next iteration clusters around the new q_0 value and generate a modified probability function $P^*(q)$. The new PMF according to this function is as follows:

$$W(q, q_0) = -k_B T ln P^*(q) - U(q, q_0) - k_B T ln \langle \exp\left(-\frac{U(q)}{k_B T}\right)\rangle_* \quad (4.7)$$

The problem with this method is that the probability function from a single q_0 is unlikely to include the entire span of the reaction coordinate. Therefore, one needs to repeat this procedure by changing q_0 to obtain a complete PMF. This procedure requires a proper choice of the reaction coordinate that is computationally convenient to sample. The only solution within the scope of umbrella sampling is to come up with a generalized reaction coordinate that combines multiple reaction coordinates[86].

4.3.2 LOCAL ELEVATION

Local Elevation (LE) [17] is a free energy method that adds a memory-dependent potential energy term to drift the simulation away from already-sampled configurations by making them less favourable energetically. This increases the chance of sampling unexplored configurations.

The total potential energy is the sum of the physical potential energy surface and bias energy such that

$$U_{tot} = U_{phys} + U_{bias}^{LE} \quad (4.8)$$

Initially, the biasing term U_{bias}^{LE} is zero. Repulsive potential energy functions are added at each time step.

$$U_{bias}^{LE}(Q;(n+1)\Delta t) = U_{bias}^{LE}\big(Q;(n)\Delta t + k_{LE}F(Q - Q_{n+1})\big) \quad (4.9)$$

k_{LE} is the scaling constant and $F(Q\text{-}Q_{n+1})$ is the repulsive function that penalizes the visited areas of the phase space. Then, the overall biasing potential is;

$$U_{bias}^{LE}(Q;(n)\Delta t) = \sum_{i=1}^{n} k_{LE}F(Q_i) \quad (4.10)$$

The approach was first developed and tested by Huber et.al[17], who used Gaussian functions to penalize the visited conformations.

$$F(q,q_0) = k_{mem}n_{q^0}\exp\left(-\frac{(q-q^0)^2}{2w^2}\right) \quad (4.11)$$

q is the current conformation, q^0 is the previously sampled conformation, $n_q{}^0$ is the number of times q^0 was sampled before, k is the height and w is the width of the Gaussian function. Truncated polynomials having widths of the grid-spacing were also used to construct repulsive functions to avoid discontinuous biasing potentials or non-periodic biasing potentials for periodic coordinates such as angles[87].

LE is particularly useful for finding a set of low-energy structures. The major drawback of this method is the requirement of storing every sampled conformation, which limits its usage to small systems. With a large set, a slowdown occurs during the comparison between the current conformation and the previously stored ones.

4.3.3 METADYNAMICS

Metadynamics is another method that adds a biasing potential to enhance sampling. Similar to the LE method, a memory-dependent bias potential is constructed by summing up Gaussians. This ensures that the previously sampled configurations are not visited in the next

simulation.[18]

The Metadynamics biasing potential is written in terms of functions of the reaction coordinate *F(q)*. When the reaction coordinate is divided into d number of bins, the biasing potential can be written as follows:

$$V_G(F,t) = \int_0^t dt' \omega \exp\left(-\sum_{i=1}^{d} \frac{\left(F_i(q) - F_i\left(q(t')\right)\right)^2}{2\sigma_i^2}\right) \quad (4.12)$$

σ is the width of the Gaussian for the i[th] bin, t' is the time interval at which a new Gaussian is added and ω is the energy rate, which is expressed as the Gaussian height divided by the Gaussian addition interval.

$$\omega = \frac{W}{t'} \quad (4.13)$$

The Gaussian potentials are deposited in each step to create a growing overall-bias potential. When one free energy well is sufficiently filled, the next Gaussian addition pushes the system towards the next local minimum. This continues until all local minimum wells are sufficiently filled for a random walk.

Metadynamics has several advantages as a biasing potential method. Like the local elevation method, it accelerates the sampling of high energy regions and even facilitates new reaction pathway discoveries without any prior knowledge of the energy surface. As opposed to local elevation, it takes the biasing factor into account and constructs the unbiased free-energy surface[88].

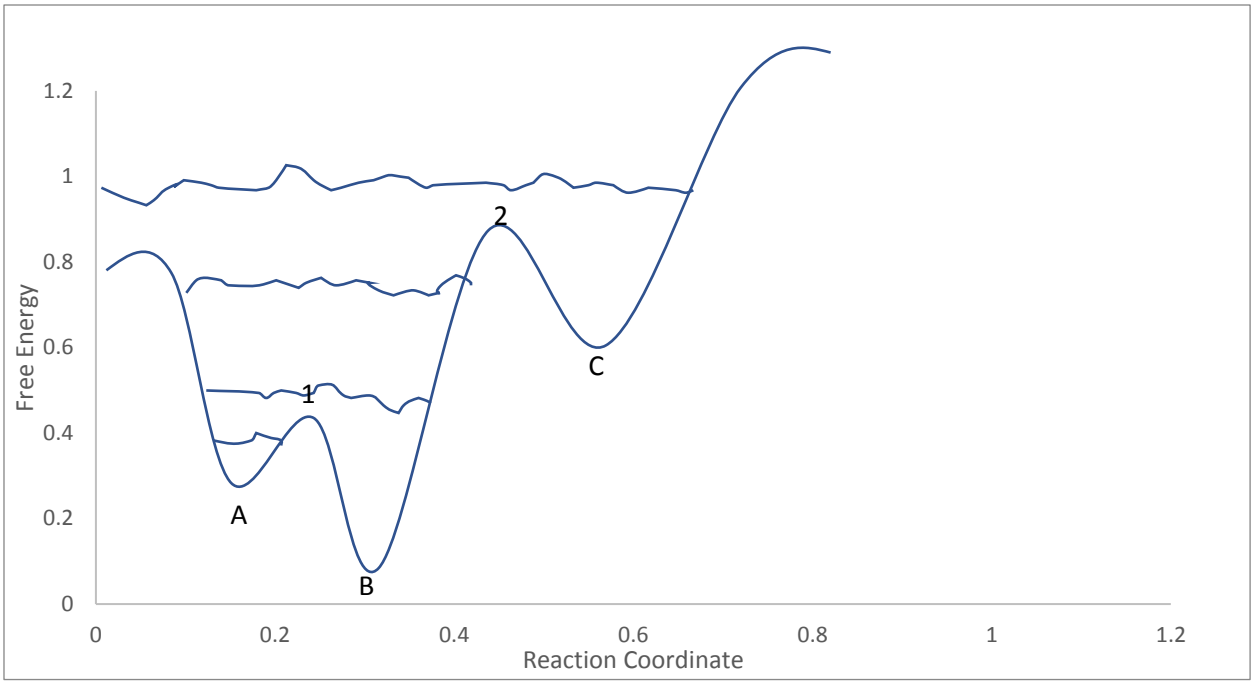


Figure 4.2: A sketch that shows the filling of minimum energy wells in Metadynamics. The sampling that starts at point A has to fill in the well completely before crossing point 1 over to the minimum energy well B. The Gaussians are then supposed to completely fill the region up to the point to before being able to sample the minimum point C.

There are certain drawbacks of Metadynamics. Most importantly, the biasing potential oscillates before converging and causes the overfilling of the free energy surface. Therefore, it is not always clear when to terminate the simulation[76].

One method to minimize the error is to decrease the growth rate by decreasing the Gaussian height. This method is called Well-tempered Metadynamics[89]. The Gaussian height is updated at each step as follows:

$$W = \omega t' \exp\left(-\frac{V_G(F,t)}{k_B \Delta T}\right) \quad (4.14)$$

W_0 is the initial Gaussian height per time, τ_G is the time interval between consecutive Gaussian depositions and ΔT is the temperature change parameter that determines the deviation of the trajectory from the free-energy minima.

In Well-Tempered Metadynamics, the overall biasing potential does not converge to the exact negative of the free energy surface, it only converges partially. As the simulation time goes to

infinity, the biasing potential converges to

$$V_G(F, t \to \infty) = -\frac{\Delta T}{T + \Delta T} F(S) + C \quad (4.15)$$

Where *C* is an arbitrary constant. The extent of convergence and overfilling can be determined by adjusting ΔT.

4.4 NOVEL APPROACHES TO BIAS THE POTENTIAL

The biasing potential can be adapted according to the accumulated results of the previous samplings.

4.4.1 ADAPTIVE UMBRELLA SAMPLING

The idea behind Adaptive Umbrella Sampling is to adapt the biasing potential until convergence is achieved. The aim of the method is to take adequate samples throughout the reaction coordinate with the adaptive potential eventually converging to the inverse of the PMF.

The earliest definition of the Adaptive Umbrella Sampling method was made by Mezei et.al[19]. In this study, a novel approach called the self-consistent method was presented. The word self-consistent emerged from the fact that the biasing potential was generated directly from the simulation instead of an external potential term such as Gaussian.

The potential energy function was defined as the sum of the original potential and the additional term.

$$V = E + E_W \quad (4.16)$$

In the first simulation, the additional potential term $E_w(q)$ – a function of the reaction coordinate q – is zero since there is no PMF data in the beginning. The set of sampled values of q, S_n, is also zero. The probability distribution according to these conditions is equal to $P_0(q) = 1$.

With the initial E_w value, a simulation is run for a reasonable length. For each grid along λ, probability estimates obtained from only the current iteration are calculated using the following equation:

$$< p_n^k >_E = \frac{< p_n^k e^{\frac{E_W}{kT}} >_V}{< e^{\frac{E_W}{kT}} >_V} \quad (4.17)$$

With k is the bin number; n is the iteration number; p_n^k is the probability estimate obtained at the k^{th} window from the n^{th} iteration. A combination of individual probability estimates with appropriate weighing and normalization gives an overall probability distribution

$$P_n^k = \sum_{i=1}^{n} r_i^k N_i p_i^k \quad (4.18)$$

where r_i^k is the set of weights and N_i is the set of normalization factors formulated in the following formula:

$$r_n^k = \frac{f_n^k}{\sum_{j=1}^{n} f_j^k} \quad (4.19)$$

where f_n^k is the set of the number of configurations each generating a certain probability from the set of probability estimates p_n^k. Equations 4.18 and 19 imply that the grid points that are sampled more frequently are weighed more, contributing more to the overall distribution function.

The additional potential term for the next iteration is set as the inverse of the PMF obtained from the probability distribution.

$$E_w(q_k) = kTln(P_n^k) \quad (4.20)$$

With this new additional term, the potential energy function V is updated. New values of q are chosen for the next set of simulations to encourage the sampling of unexplored regions.

Mezei et al used the following formulation for picking the next set of grid points:

$$q^c(q_k, S) \epsilon S, |q_k - q^c(\lambda, S)| = \min_{q' \epsilon S} |q_k - q'| \quad (4.21)$$

For the choice of each new starting position, the grid point should be outside the previously sampled points, but closest to any of those points. When this is done, the additional potential term at these new points is assigned as in

$$E_w(q_k) => E_w\left(q^c(q_k, S_n)\right) \quad (4.22)$$

The value of E_w at the point in S_n nearest to q_k becomes the new potential term.

4.4.2 REWEIGHTING METHODS

Earlier studies of Monte Carlo simulations provided only average quantities of thermodynamic properties based on a single point on the parameter space. If a simulation was performed at a certain temperature, the simulation data would be valid only for that particular temperature. There were several attempts to obtain information over a range of values of a parameter[90, 91]. However, these required multiple MC simulations which was computationally expensive[77].

Reweighting methods were developed to obtain results from a simulation at a different temperature without additional simulations.

Given a system with two energy levels E_0 and E_1 and the corresponding partition functions Z_0 and Z_1 with the free energy difference

$$\Delta A = -k_B T ln\left(\frac{Z_0}{Z_1}\right) \quad (4.23)$$

For any observable O, $\langle O \rangle_1$ – averaged with E_1 – can be determined by the data generated with E_0. The procedure is as follows:

$$\langle O \rangle_1 = \frac{\int dr\ O(r) e^{-\beta E_1(r)}}{\int dr\ e^{-\beta E_1(r)}} \quad (4.24)$$

Since $Z_0 = \int e^{-\beta E_0(r)}$, we can rewrite the above equation as follows:

$$\langle O \rangle_1 = \frac{Z_0^{-1} \int dr\ O(r) e^{-\beta(E_1 - E_0)}\ e^{\left(-\beta E_0(r)\right)}}{Z_0^{-1} \int dr\ e^{-\beta(E_1 - E_0)} e^{\left(-\beta E_0(r)\right)}} \quad (4.25)$$

Which can be written as an average of E_0. The resultant is

$$\langle O \rangle_1 = \frac{\langle O e^{-\beta(E_1 - E_0)} \rangle_0}{\langle e^{-\beta(E_1 - E_0)} \rangle_0} \quad (4.26)$$

A sufficient sampling from the energy level E_1 is necessary for equation 4.26. A high overlap between the energy distributions of E_0 and E_1 is needed to obtain an estimate of $\langle O \rangle_1$ from data generated with E_0.

The estimated error was found to increase exponentially with the volume of the system[92]

4.4.3 FERRENBERG-SWENDSEN METHOD

If the parameter of interest is far away from some of the parameter values used in the simulations, the errors coming from those values will be large.

The Ferrenberg-Swendsen (FS) method combines multiple histograms to reconstruct a final histogram of the density of the states with a minimized error. The contribution of each run to the overall probability distribution depends on the magnitude of errors or variance. The method determines a reweighting scheme and gives more consideration to the runs with higher overlaps[77]. This is particularly helpful for combining multiple histograms δ_i at different temperatures T_i to generate a probability distribution at a new temperature T_1. To obtain the distribution at T_1 from an individual run, the conversion between the two distributions could be written as follows:

$$\delta_i(E;T_1) = \frac{(\delta(E,T_i)e^{-(\beta-\beta_i)E})}{\int(\delta(E,T_0)e^{-(\beta-\beta_i)E}dE)} = \frac{Z(T_i)}{Z(T_1)}\delta(E,T_i)e^{-(\beta-\beta_i)E} \quad (4.27)$$

The equation relates the histogram obtained at a run temperature to that at a target temperature T_1. According to FS, it is possible to make a linear combination of multiple histograms by using weighing factors in the following equation:

$$\delta(E;T_1) = \sum_i w_i(E)\delta_i(E;T_1) = \sum_i w_i(E)\frac{Z_i(T_i)}{Z(T_1)}\delta(E,T_i)e^{-(\beta-\beta_i)E} \quad (4.28)$$

With the weighing functions

$$\sum_i w_i(E) = 1 \quad (4.29)$$

The weighing functions, w_i, depend on the energy E, the number of simulations and the number of histograms. The constraint 4.29 is used to determine the set of weighting functions that will minimize the prediction error.

The statistical error function for each window *i* would be

$$\left(\sigma^{i}\right)^{2} = \frac{\langle e^{-\beta E_i}\rangle e^{-(\beta E_i)}}{f_i(E_i)} \quad (4.30)$$

The optimum weight function can be written in terms of the number of counts in a bin, *f(E)*, and the total number of counts, f_{tot}, using Lagrange multipliers for the minimization of the error**.** The variance in each histogram measure is dependent on both the energy and the number of counts in the i^{th} bin. [93]

$$w_i = \frac{f_i e^{-\beta E}}{\langle e^{-\beta E}\rangle} \quad (4.30)$$

Substituting the weights in equation 4.28 gives

$$\delta^{*}(E;T_1) = \frac{\sum_i \ f_i(E)e^{-\beta E}}{\sum_i f_{tot,i} e^{(\beta_i A_i - \beta_i E)}} \quad (4.31)$$

$$e^{\beta_i A_i} = \frac{1}{\sum_E \delta^{*}(E;T_i)} \quad (4.32)$$

$\delta^{*}(E;T_1)$ is the un-normalized probability distribution, A_i is the free energy for run *i*, $f_i(E)$ is the number of counts of energy *E* for the i^{th} run, $f_{tot} = \sum_k f(E_k)$. Then, the normalized probability distribution is as follows:

$$\delta(E;T_1) = \frac{\delta^{*}(E;T_1)}{\sum_E \delta^{*}(E;T_1)} \quad (4.33)$$

The free energy for each run, A_i, is calculated iteratively by taking the initial value $A_i = 0$ and continuing until the value converges. The final energy values are used to calculate the final probability distribution, from which the PMF can be calculated.

4.4.4 WHAM

The FS method was generalized as the Weighted Histogram Analysis Method (WHAM) to other simulation settings and ensembles. In such cases, the probability distributions will be multi-dimensional. The demonstration was made by Kumar Et. Al[78], where the energy function for each run was written as a linear combination of component energy functions V_j and coefficients λ_i as in

$$E_i = \sum_j \lambda_{i,j} V_j$$

The WHAM equations for this energy expression are

$$\delta^*(\boldsymbol{V}, \xi; T) = \frac{\sum_i f_i(V, \xi) \exp\left(-\beta \sum_j \lambda_j V_j\right)}{\sum_i f_{tot,i} \exp\left(\beta_i A_i - \beta_i \sum_j \lambda_{i,j} V_j\right)} \quad (4.34)$$

$$e^{(-\beta_i A_i)} = \sum_E \delta^*(\boldsymbol{V}, \xi; T) \quad (4.35)$$

$$\delta(\boldsymbol{V}, \xi; T) = \frac{\delta^*(\boldsymbol{V}, \xi; T)}{\sum_E \delta^*(\boldsymbol{V}, \xi; T)} \quad (4.36)$$

$\boldsymbol{V}$ denotes the vector for the component energies V_j. The notations $\lambda_{i,j}$ correspond to the coefficient of V_j in the i^{th} run. The formula indicates that the histograms are functions of both the component energies and the order parameter ξ.

While WHAM relies on histograms containing discrete bins with many samples, its equations hold true even when the bin width is made arbitrarily small. The Multistate Bennett Acceptance Ratio (MBAR)[94] method can predict the free energy at a state that is not sampled. The method is widely used to compute absolute free energies of states formed through alchemical transformations without the need to do binning. It is shown that MBAR provides the lowest variance for alchemical free energy calculations[95].

4.4.5 FEARCF

The Adaptive Umbrella sampling was initially used to construct the Ramachandran W(Φ,ψ) two-dimensional conformational PMF for maltose in an aqueous solution[96]**.** Furthermore, it was proven that the method can be coupled with WHAM and hybrid quantum and classical methods for generating multidimensional PMF for reactions in aqueous solutions[48]. Finally, the method was generalized for multidimensional free energy volumes and renamed as the Free Energies from Adaptive Reaction Coordinate Forces.

FEARCF generates forces from probability distributions of one or more reaction coordinates and uses these forces to drive the simulation towards poorly sampled regions. This continues iteratively until an equal ratio of sampling of the global minimum and the highest energy state is achieved.

The first clear explanation of the method was made using an example of Claisen rearrangement, where a carbon-carbon bond forming occurs and facilitates the rearrangement of other bonds and conformations[20]. At the end of the reaction, the molecule Chorismate is converted into a Prephenate catalysed by an enzyme Chorismate mutase[97]**.** The free energy calculations were performed using two reaction coordinates, both of which were bond distances.

The effective Hamiltonian of the system was written as a sum of the initial system Hamiltonian, H^0, and the driving potential, V.

$$H_{i+1} = H^0 + V_{i+1}(\xi_1, \xi_2) \quad (4.37)$$

Because there is no prior data, the driving potential (V_1) is set to zero. At the end of every iteration, the driving potential is updated, similar to the method of adaptive umbrella sampling. At any given i^{th} simulation, the driving potential is updated as the negative of the PMF, which is a function of the unbiased probability distribution averaged over all the simulations up to that point.

$$-V_{i+1}(\xi_1, \xi_2) = W_i(\xi_1, \xi_2) = -\frac{1}{\beta} ln\, \overline{\mathrm{P^0_i}}\ (\xi_1, \xi_2) \quad (4.38)$$

The notation $\overline{P^0_i}$ stands for the accumulative unbiased probability distribution, which is the weighted average of the unbiased probability distributions from the first simulation to the i^{th} simulation. The weighing is performed through the WHAM equations.

$$\overline{\mathrm{P^0_i}}\ (\xi_1, \xi_2) = C \sum_{k=1}^{i} w_k^0(\xi_1, \xi_2) P_k^0(\xi_1, \xi_2) \quad (4.39)$$

C is the normalization factor and $P_k^0\ (\xi_1, \xi_2)$ is the unbiased probability distribution obtained at the k^{th} simulation. Each distribution can be expressed in terms of the driving potential as follows;

$$P_k^0(\xi_1, \xi_2) = \frac{f_k(\xi_1, \xi_2)}{F_k} e^{\beta(V_k(\xi_1, \xi_2) - A_k)} \quad (4.40)$$

where F_k is the number of all data points sampled during the k^{th} simulation and f_k is the number of data points for the bin containing the current positions of the two reaction

coordinates. Also, A_k is the relative free energy of the k^{th} simulation compared to other simulations. The relative free energy can be obtained by calculating the minimized statistical error.

In the first iteration, the PMF is obtained from equilibrium dynamics since the external potential is unknown and there are no forces exerted. As the iterations proceed, the PMF is updated by populating the discrete bins in a cumulative way by collecting snapshots of conformations.

To update the reaction coordinate positions for the next iteration, the driving forces are calculated by taking the partial derivative of $W_i(\xi_1, \xi_2)$ with respect to the reaction coordinates.

$$F(\xi_i) = -\frac{\partial W(\xi_1, \xi_2)}{\partial \xi_i} \quad (4.41)$$

Forces are applied in Cartesian space at each atom that is involved in the reaction coordinate. These forces can be obtained by applying a chain rule. For an atom x_A

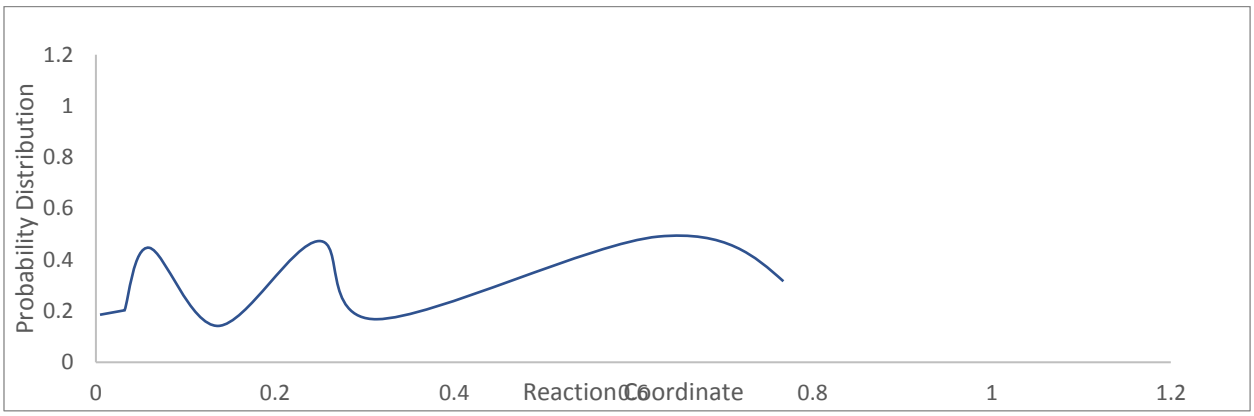
$$F(x_A) = -\frac{\partial W(\xi_1, \xi_2)}{\partial \xi_i} \frac{\partial \xi_i}{\partial x_A} \quad (4.42)$$

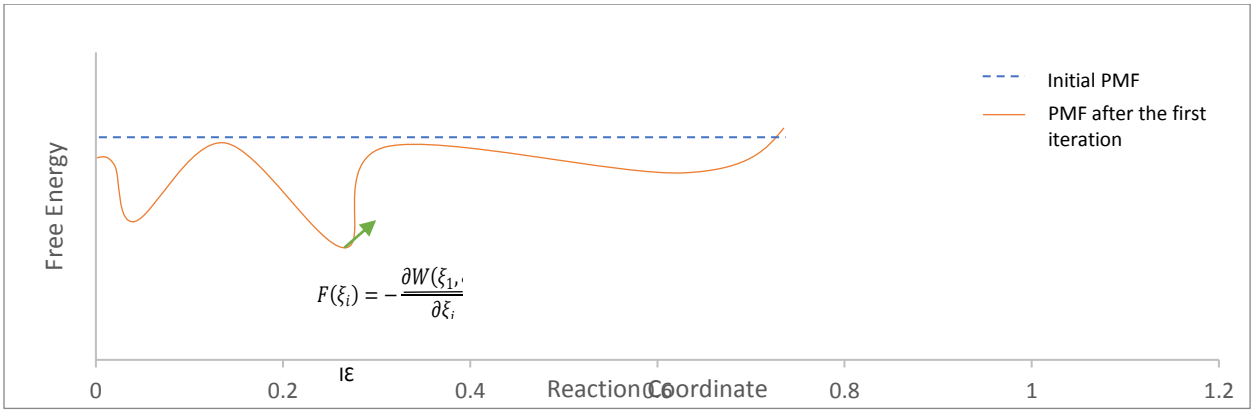
The PMF's are obtained from discrete histograms; however, the driving forces $F(\xi_i)$ are described as the gradient of $\partial W(\xi_1, \xi_2)$ along the reaction coordinates. This definition requires a continuous PMF profile, but the discretized bins of histograms are discontinuous and not differentiable. So, the forces along the reaction coordinates are calculated by interpolation along the histograms at each simulation step to smoothly connect the bins.

Among the interpolation methods reviewed in Chapter 2, the most eligible method is the spline interpolation. Many reactions involve rare events such as transition states, which can lead to a jagged free energy surface with jumps between data points. A single polynomial interpolation with a high degree leads to oscillations between the bins. A more useful approach is to use interpolants of a smaller degree between each bin. Therefore, the method used in FEARCF is the cubic spline interpolation, which interpolates between consecutive bins in a piece-wise fashion using cubic polynomials.

The derived forces are applied to update the atomic positions along the reaction coordinate.

The driving forces are updated at each step according to the current PMF to avoid the previously-visited conformations. Eventually, the atoms acquire such positions that the simulations will adequately sample the rare events occurring at these positions. For the example of the Claisen rearrangement, this would be the distance at which the C-O bond breaks or the distance at which the C-C bond forms. Such crucial positions are gradually reached by applying the forces in either direction to expand or shorten the bond lengths between atoms.





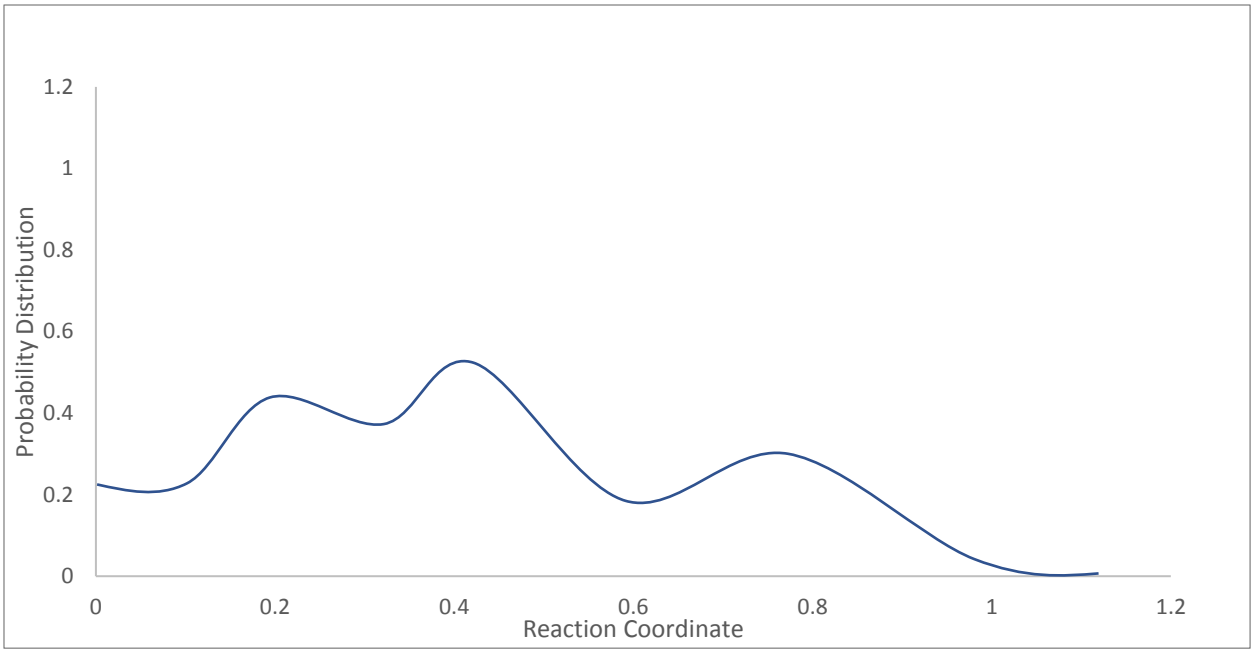


Figure 4.3: A sketch illustrating the beginning of a FEARCF iteration. The initial PMF is unbiased. The probability distribution is used to obtain the first biased PMF surface from which forces are constructed through derivation via cubic spline. These forces are used to update atom positions, so the next probability distribution will be more inclusive of the high energy regions.

At the end of each simulation, a convergence check is performed to determine whether or not to continue with the next iterations. The convergence would be achieved when equal sampling throughout the reaction coordinate(s) is observed. In that case, the final PMF generated corresponds to the true free energy profile, thus indicating convergence.

For chemical reactions with rare events, the accepted ratio of sampling between the rare event and the minimum energy level is 1:50[98], although ratios as promising as 1:1.7 have previously been achieved[72]**.**

4.5 ADVANTAGES OF FEARCF

The FEARCF method has a few important advantages compared to the other flat histogram methods. It boasts certain features that make it less time-consuming and more suitable for higher dimensions. This section discusses the advantages of FEARCF in terms of computational science and the application of numerical methods.

4.5.1 EMBARASSINGLY PARALLEL SIMULATIONS

A major advantage of HPC systems is the ability of parallel computing where multiple computer resources are used simultaneously to solve complex computational problems.

In parallel computing, the problem is broken into discrete parts that execute the same instructions simultaneously on different processors. In the end, the outcomes of the parallel processors are correlated and coordinated to give an overall output.

The simplest type of parallel algorithm is one that requires no communication between the simultaneous processes. Thus, each process can perform their own computations independently. The term "embarrassingly parallel" was first used in literature by MATLAB's creator Cleve Moler[99].

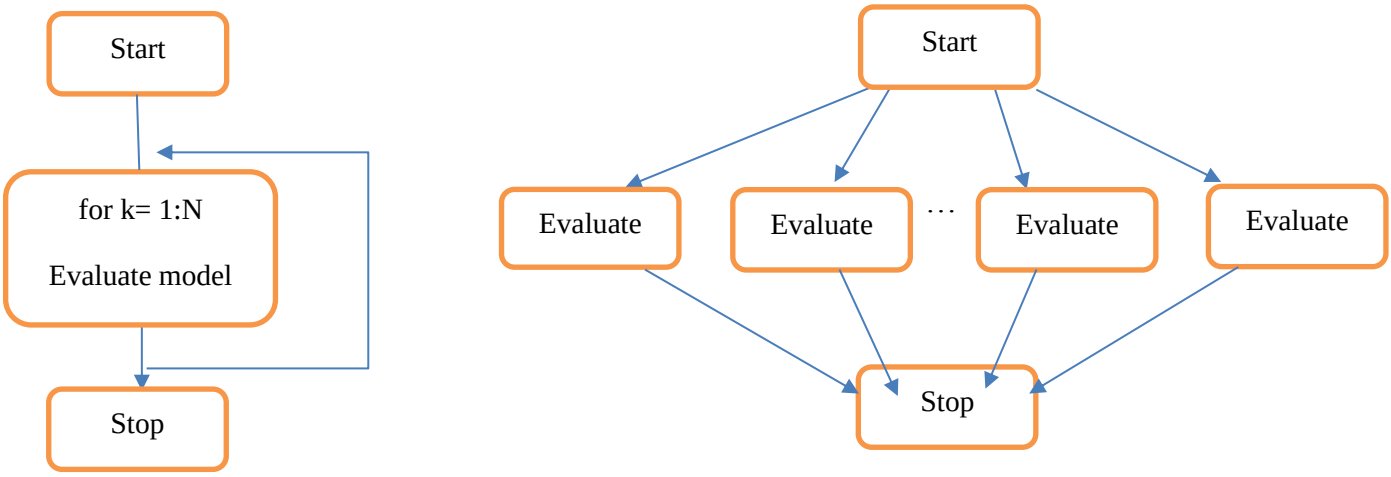


Figure 4.4: A flow diagram of the comparison between serial computation and parallel computation

One of the biggest advantages of FEARCF is that multiple histograms can be spread over multiple compute units and run simultaneously. This is necessary since spanning the reaction coordinate in one iteration requires many simulations. If run sequentially, this process could take days or even weeks.

The embarrassingly parallel computation allows the separation of one iteration into many simulations that run simultaneously and independent of each other. The histograms generated from parallel tasks are combined and weighed to get the ultimate probability distribution for that iteration.

4.5.2 THE CHOICE OF COLLECTIVE VARIABLES

One of the biggest challenges in flat histogram methods is the choice of reaction coordinates also called Collective Variables (CV). One must be able to select the fewest number of CVs possible, yet sufficient to define the important conformational changes.

In classical MD methods, reactions are simulated on a Cartesian coordinate system with Newton's equations of motion because Cartesian-based all-atom dynamics are mathematically simple to construct compared to complex Lagrange equations [100]. However, using a non-linear reaction coordinate leads to complications in the Cartesian system. The projection of the free energy on a non-linear reaction coordinate leads to an additional term in the PMF to translate the Cartesian coordinates into the internal coordinates through the determinant of the Jacobian Matrix. Although Cartesian-based dynamics are mathematically simple, they require complex differential equation solvers when Jacobian corrections are added[101]**.** The computational speed of the umbrella method significantly suffers from this obstacle when non-linear reaction coordinates are involved. This reduces the feasibility of umbrella sampling with collective variables containing angular reaction coordinates.

Another method to generate a subset of essential coordinates is Principal Component Analysis (PCA) that represents the fluctuations in the Cartesian coordinates of the atoms[102]**.** Metadynamics and Well-Tempered Metadynamics have employed this method to construct a more informed set of collective variables. Through PCA performed in a shorter and unbiased MD simulation beforehand, the largest fluctuations in the atom positions are found and used to generate the essential reaction coordinates.[103]

Although PCA of Cartesian coordinates can generate collective variables, they fall short in representing internal and non-linear coordinates on their own[104]. In previous studies, PCA carried out using only Cartesian coordinates did not succeed in depicting the cyclic motions[105]. PCA can account for the angular motion, but this requires the conversion of cyclic coordinates into a linear metric coordinate space. Novel methods such as dihedral Principal Component Analysis (dPCA) [106] were developed. However, dPCA brings in additional computation time and resource usage for the separate PCA of dihedrals.

The FEARCF method eliminates the need for Jacobian corrections by applying biasing forces. The forces are applied on the Cartesian coordinates of the atoms whether the reaction coordinate is Cartesian or an internal coordinate. This conversion is done by applying a chain

rule of derivation[107] with the underlying cubic spline interpolation that makes the PMFs twice differentiable. The direct application of forces omits the necessity to calculate the Jacobian term. Since the FEARCF is faster due to this simplification, it is currently possible to sample reactions with up to six reaction coordinates

4.6 OBJECTIVES OF THE THESIS

In Chapter 2, the spline methods were covered in detail introducing the cubic spline and the B-spline interpolation methods, concluding that the B-spline method is faster. Although the time difference in a single interpolation can be negligibly small, a B-spline implementation can lead to a more significant speedup in iterative and multidimensional interpolations. FEARCF constitutes a perfect example since forces are derived from multiple PMF measures through a high number of iterations. Replacing the cubic spline routine with a B-spline routine could decrease the computation time significantly. Furthermore, if the B-spline interpolation yields identical results to the cubic spline routine, the free energy surface at the end should be similar if not identical.

Since the PMF surfaces are obtained numerically, direct observation of the forces remains a challenge. Working with analytic functions enables a direct comparison between the two spline methods.

The dissertation aims to compare the accuracy and speed of the cubic spline interpolation and the B-spline interpolation using analytic functions. Then, the comparison is translated into the FEARCF sampling of a chemical reaction for further validation.

4.7 RESOURCES

1. Naidoo, K.J., *FEARCF a multidimensional free energy method for investigating conformational landscapes and chemical reaction mechanisms.* Science China Chemistry, 2011. **54**(12): p. 1962-1973.

2. Landau, D., S.-H. Tsai, and M. Exler, *A new approach to Monte Carlo simulations in statistical physics: Wang-Landau sampling.* American Journal of Physics, 2004. **72**(10): p. 1294-1302.

3. Tsai, S.-H., F. Wang, and D. Landau, *Critical endpoint behavior in an asymmetric Ising model: Application of Wang-Landau sampling to calculate the density of states.*

Physical Review E, 2007. **75**(6): p. 061108.

4. Hastings, W.K., *Monte Carlo sampling methods using Markov chains and their applications.* 1970.

5. Belardinelli, R. and V. Pereyra, *Wang-Landau algorithm: A theoretical analysis of the saturation of the error.* The Journal of chemical physics, 2007. **127**(18): p. 184105.

6. Belardinelli, R., S. Manzi, and V. Pereyra, *Analysis of the convergence of the 1/ t and Wang-Landau algorithms in the calculation of multidimensional integrals.* Physical Review E, 2008. **78**(6): p. 067701.

7. Torrie, G.M. and J.P. Valleau, *Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella sampling.* Journal of Computational Physics, 1977. **23**(2): p. 187-199.

8. Beutler, T.C. and W.F. van Gunsteren, *Umbrella sampling along linear combinations of generalized coordinates. Theory and application to a glycine dipeptide.* Chemical physics letters, 1995. **237**(3-4): p. 308-316.

9. Huber, T., A.E. Torda, and W.F. Van Gunsteren, *Local elevation: a method for improving the searching properties of molecular dynamics simulation.* Journal of computer-aided molecular design, 1994. **8**(6): p. 695-708.

10. Hansen, H.S., X. Daura, and P.H. Hünenberger, *Enhanced conformational sampling in molecular dynamics simulations of solvated peptides: Fragment-based local elevation umbrella sampling.* Journal of chemical theory and computation, 2010. **6**(9): p. 2598-2621.

11. Laio, A. and F.L. Gervasio, *Metadynamics: a method to simulate rare events and reconstruct the free energy in biophysics, chemistry and material science.* Reports on Progress in Physics, 2008. **71**(12): p. 126601.

12. Abrams, C. and G. Bussi, *Enhanced sampling in molecular dynamics using metadynamics, replica-exchange, and temperature-acceleration.* Entropy, 2014. **16**(1): p. 163-199.

13. Barducci, A., M. Bonomi, and M. Parrinello, *Metadynamics.* Wiley Interdisciplinary

Reviews: Computational Molecular Science, 2011. **1**(5): p. 826-843.

14. Barducci, A., G. Bussi, and M. Parrinello, *Well-tempered metadynamics: a smoothly converging and tunable free-energy method.* Physical review letters, 2008. **100**(2): p. 020603.

15. Mezei, M., *Adaptive umbrella sampling: Self-consistent determination of the non-Boltzmann bias.* Journal of Computational Physics, 1987. **68**(1): p. 237-248.

16. Salzburg, Z., et al., *Application of the Monte Carlo method to the lattice-gas model.* J. Chem. Phys, 1959. **30**: p. 65.

17. Chesnut, D.A. and Z.W. Salsburg, *Monte Carlo procedure for statistical mechanical calculations in a grand canonical ensemble of lattice systems.* The Journal of Chemical Physics, 1963. **38**(12): p. 2861-2875.

18. Ferrenberg, A.M. and R.H. Swendsen, *New Monte Carlo technique for studying phase transitions.* Physical review letters, 1988. **61**(23): p. 2635.

19. Ferrenberg, A.M., D. Landau, and R.H. Swendsen, *Statistical errors in histogram reweighting.* Physical Review E, 1995. **51**(5): p. 5092.

20. Pohorille, A. and C. Chipot, *Free energy calculations: theory and applications in chemistry and biology*. 2007: Springer.

21. Kumar, S., et al., *The weighted histogram analysis method for free-energy calculations on biomolecules. I. The method.* Journal of computational chemistry, 1992. **13**(8): p. 1011-1021.

22. Naidoo, K.J. and J. Brady, *Calculation of the Ramachandran potential of mean force for a disaccharide in aqueous solution.* Journal of the American Chemical Society, 1999. **121**(10): p. 2244-2252.

23. Rajamani, R., K.J. Naidoo, and J. Gao, *Implementation of an adaptive umbrella sampling method for the calculation of multidimensional potential of mean force of chemical reactions in solution.* Journal of computational chemistry, 2003. **24**(14): p. 1775-1781.

24. Ireland, R.E. and R.H. Mueller, *Claisen rearrangement of allyl esters.* Journal of the American Chemical Society, 1972. **94**(16): p. 5897-5898.

25. Strümpfer, J. and K.J. Naidoo, *Computing free energy hypersurfaces for anisotropic intermolecular associations.* Journal of computational chemistry, 2010. **31**(2): p. 308-316.

26. Barnett, C.B. and K.J. Naidoo, *Free Energies from Adaptive Reaction Coordinate Forces (FEARCF): an application to ring puckering.* Molecular Physics, 2009. **107**(8-12): p. 1243-1250.

27. Moler, C., *Matrix computation on distributed memory multiprocessors.* Hypercube Multiprocessors, 1986. **86**(181-195): p. 31.

28. Stocker, U., D. Juchli, and W.F. van Gunsteren, *Increasing the time step and efficiency of molecular dynamics simulations: optimal solutions for equilibrium simulations or structure refinement of large biomolecules.* Molecular Simulation, 2003. **29**(2): p. 123-138.

29. Vaidehi, N. and A. Jain, *Internal coordinate molecular dynamics: A foundation for multiscale dynamics.* The Journal of Physical Chemistry B, 2015. **119**(4): p. 1233-1242.

30. Pearson, K., *LIII. On lines and planes of closest fit to systems of points in space.* The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 1901. **2**(11): p. 559-572.

31. Sutto, L., S. Marsili, and F.L. Gervasio, *New advances in metadynamics.* Wiley Interdisciplinary Reviews: Computational Molecular Science, 2012. **2**(5): p. 771-779.

32. Sicard, F. and P. Senet, *Reconstructing the free-energy landscape of Met-enkephalin using dihedral principal component analysis and well-tempered metadynamics.* The Journal of chemical physics, 2013. **138**(23): p. 06B610_1.

33. Mu, Y., P.H. Nguyen, and G. Stock, *Energy landscape of a small peptide revealed by dihedral angle principal component analysis.* Proteins: Structure, Function, and Bioinformatics, 2005. **58**(1): p. 45-52.

34. Altis, A., et al., *Dihedral angle principal component analysis of molecular dynamics simulations.* The Journal of chemical physics, 2007. **126**(24): p. 244111.

35. Rogers, I.L., *Measuring the effects of reaction coordinate and electronic treatments in the QM/MM reaction dynamics of Trypanosoma cruzi trans-sialidase*. 2016, University of Cape Town.

CHAPTER 5: COMPARISON OF SPLINE INTERPOLATION METHODS WITH ANALYTICAL FUNCTIONS

5.1 OVERVIEW

In Chapter 2, the main advantages of spline interpolation methods were emphasized. In chapters 3 and 4, it was concluded that replacing the cubic spline algorithm with the B-spline algorithm can enhance the speed of the flat histogram method FEARCF. Before introducing such a replacement, the performances of the two spline methods will be systematically compared. The dual purpose of this comparison is to confirm that the B-spline method interpolates with an equivalent accuracy and that it performs faster while doing so. A set of analytic functions is used to undertake this task.

Chapter 5 explains the algorithms of both methods, paying attention to the iterations and conditional statements within the subroutines. Then, a comparison is illustrated in terms of accuracy and speed using three analytical functions.

5.2 METHODOLOGY

5.2.1 THE CUBIC SPLINE ALGORITHM

The cubic spline algorithm was implemented based on the formulation described in Chapter 2. A subroutine was coded for calculating the array of y_j'' given x_j, y_j, y_1' and y_N'. The second derivatives at the endpoints were initially determined by either setting both y_1'' and y_N'' equal to zero or calculating them from the first derivatives (using equation 2.34). The algorithm has a self-deterministic way of defining the boundary conditions by specifying a threshold value, which would set the values to zero if the first derivatives at the endpoints were equal to or larger than that value. The algorithm sets this threshold value to $1x10^{30}$.[46]

A second subroutine calculates y and y' values for any given x using the arrays of x_j, y_j and y_j''. The flow diagram of the algorithm is shown in Figure 5.1

The algorithm for a two-dimensional implementation is illustrated in Figure 5.2. Two arrays of independent values $x_{j_1}, j_1 = (1, \dots, a)$ and $x_{j_2}, j_2 = (1, \dots, b)$ and a matrix of y values $y = [y_{j_1}, y_{j_2}]$ are given. The algorithm determines a value for $y(x_1, x_2)$, where x_1 and x_2 are two

intermediate values in the respective intervals $[1, a]$ and $[1, b]$, but are different from the sets of x_{j_1} and x_{j_2}

The second derivatives are computed in the subroutine S1, followed by interpolation using the second derivatives in the second subroutine S2. The iterations in the algorithm are depicted using circular arrows, also indicating how many times the loop runs.

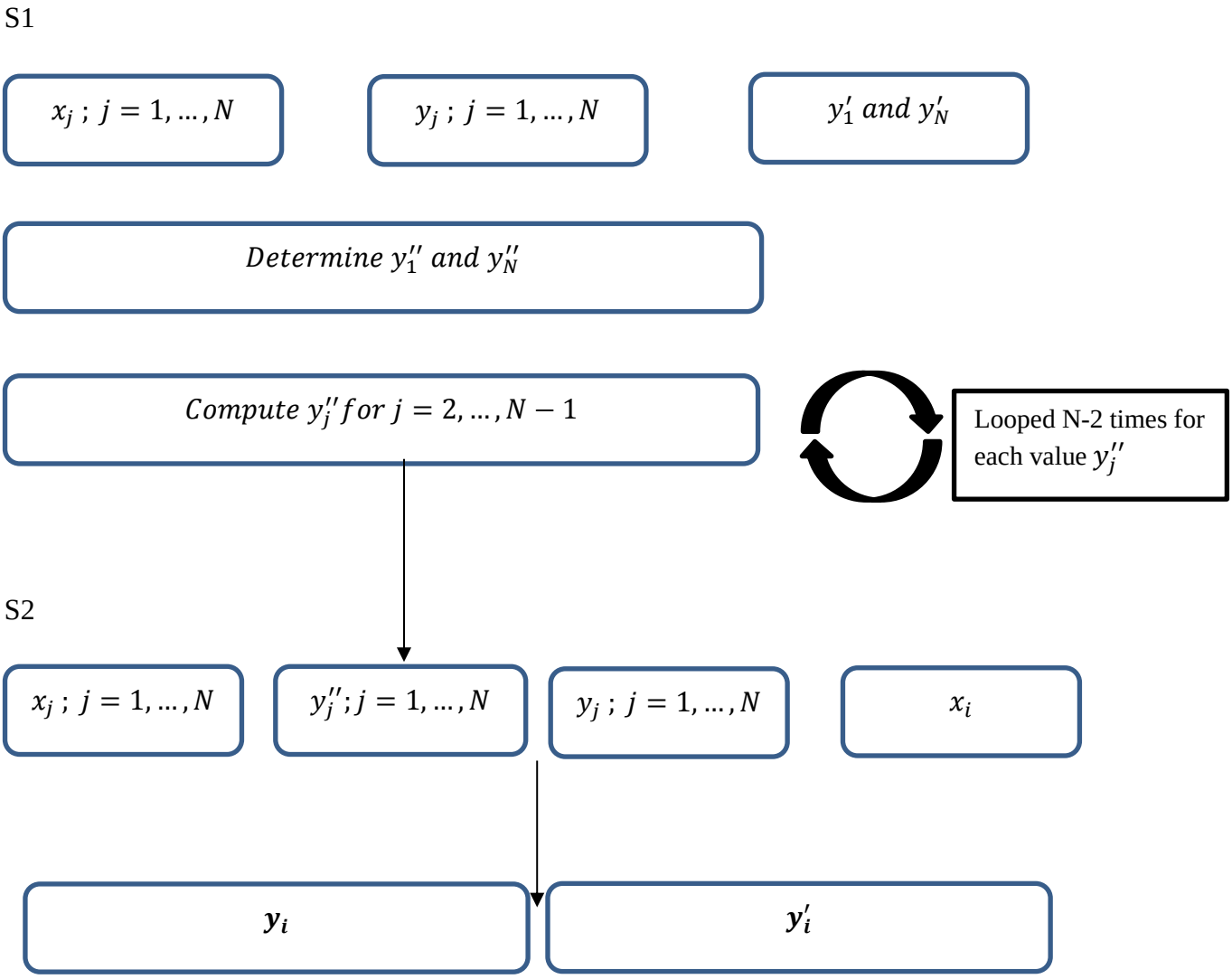


Figure 5.1: A flow diagram of the Cubic Spline Algorithm of a single array

The algorithm for the two-dimensional cubic spline algorithm was explained in Chapter 2, section 2.3-1.

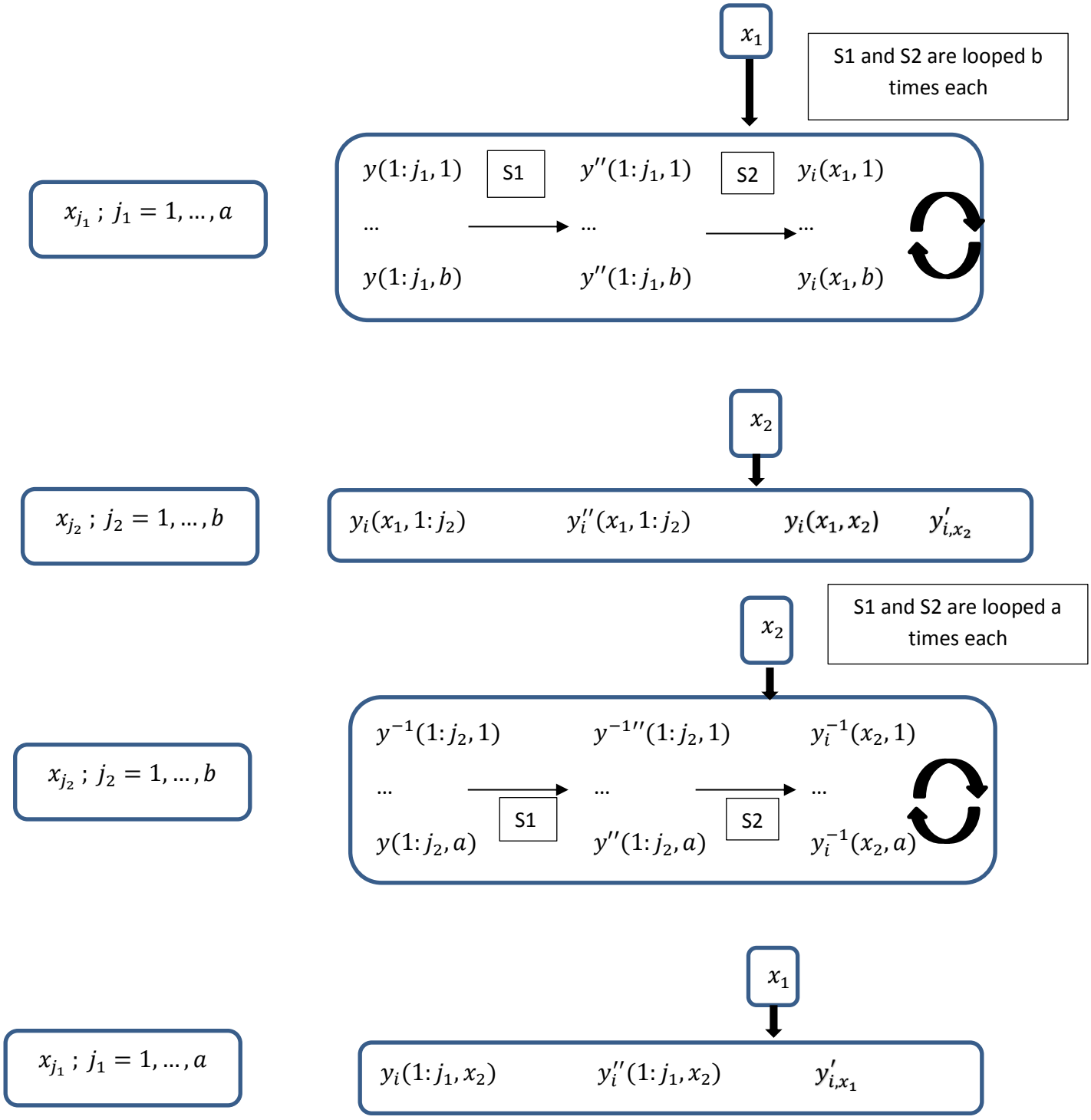


Figure 5.2: A flow diagram of the two-dimensional Cubic Spline Algorithm

5.2.2 B-SPLINE INTERPOLATION

The B-Spline algorithm was written following the Habermann et.al [51] formulation. The four subroutines were: one that calculated the coefficients from the set of y values through an equation system of a tridiagonal matrix (called SB1), one that defined the basis functions

(SB2), one that defined their derivatives (SB3) and one that formulated the linear combination of the coefficients and basis values (SB4).

The one-dimensional algorithm is shown in Figure 5.3. The two-dimensional coefficient calculation and linear combination are shown in Figures 5.4 and 5.5 respectively.

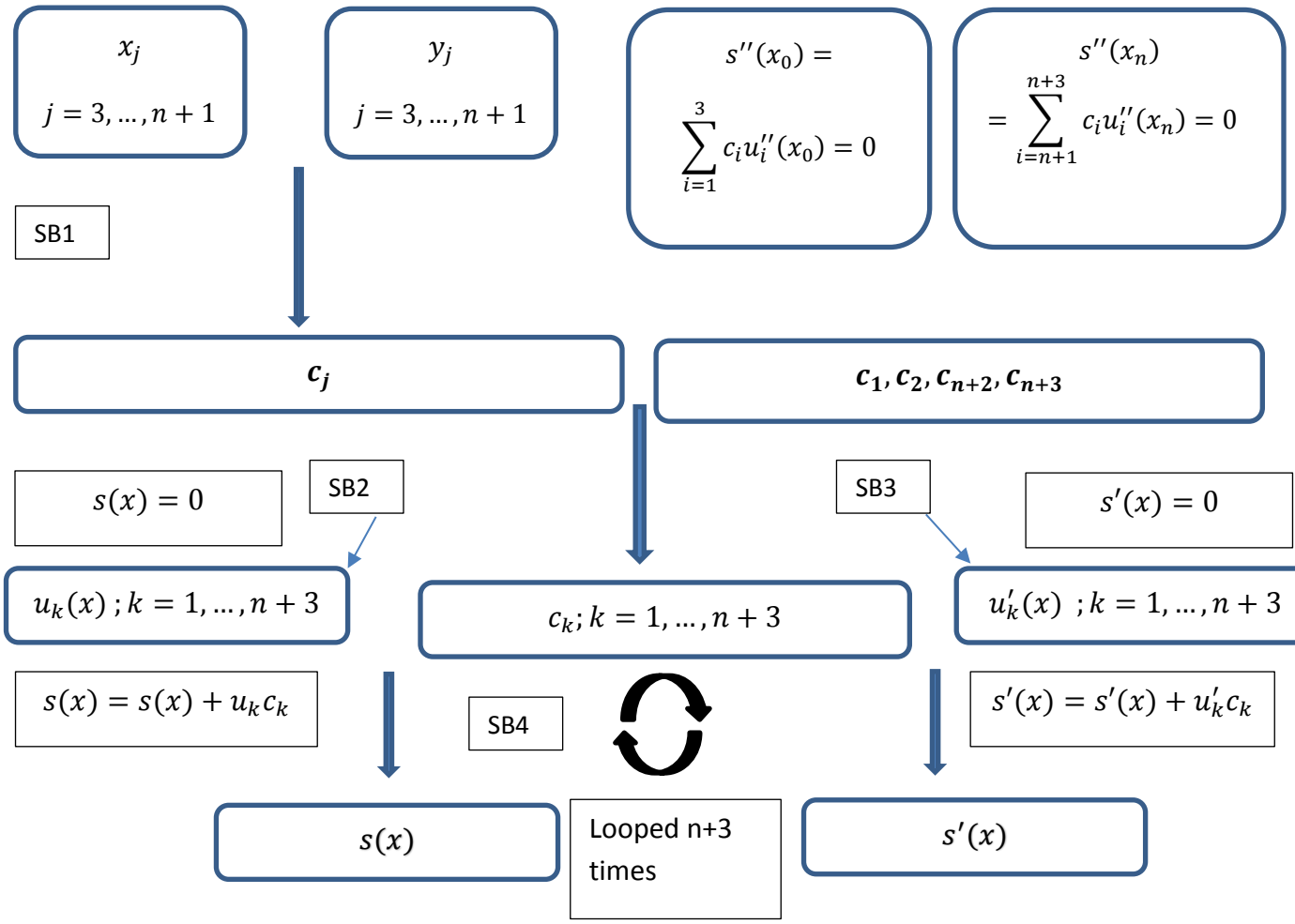


Figure 5.3: A flow diagram of the B-spline algorithm that interpolates a value of x

The calculation for the two-dimensional coefficient matrix is explained in Chapter 2, section 2.3-2.

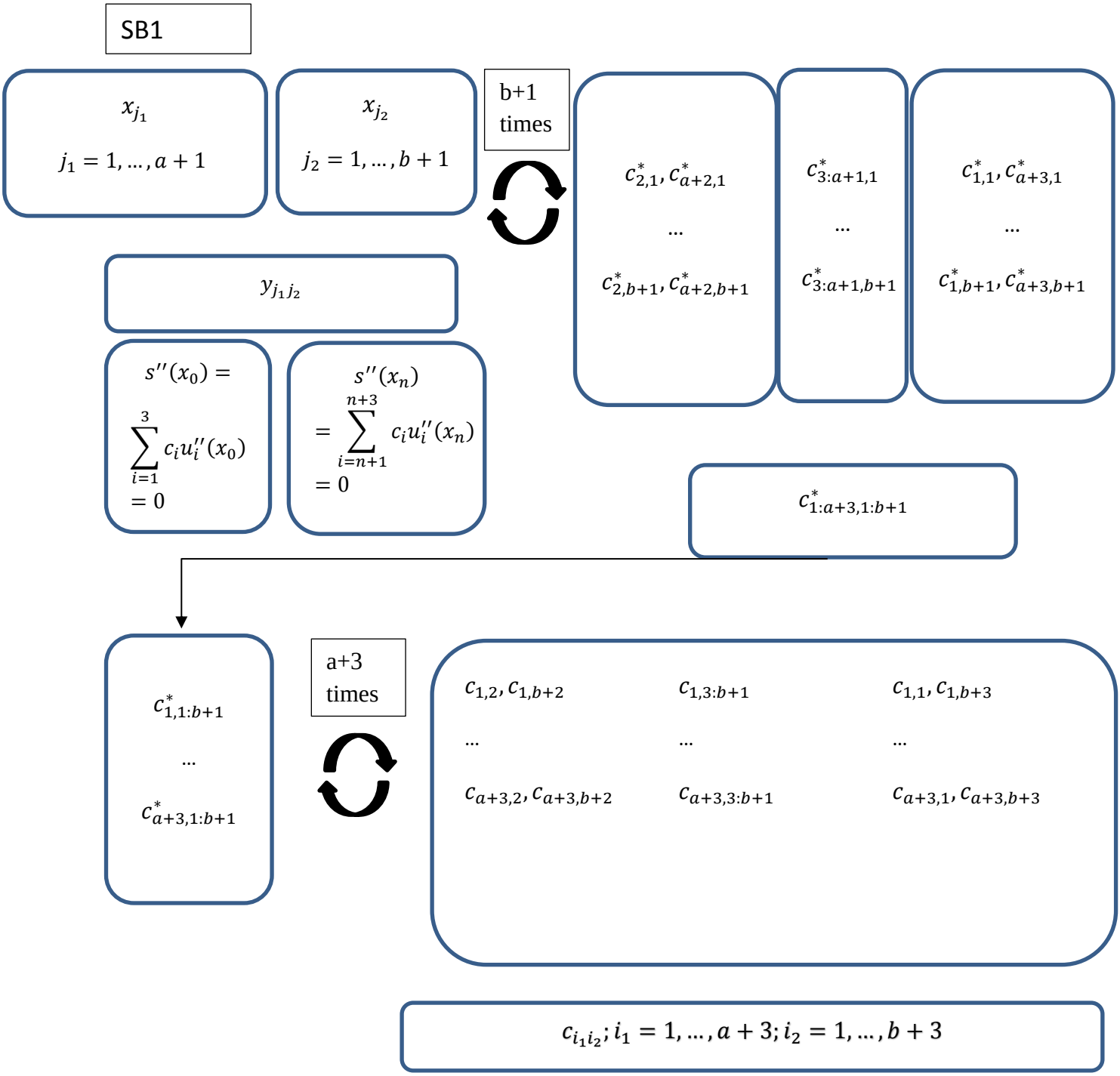


Figure 5.4: A flow diagram of the coefficient calculation for two-dimensional B-spline interpolation

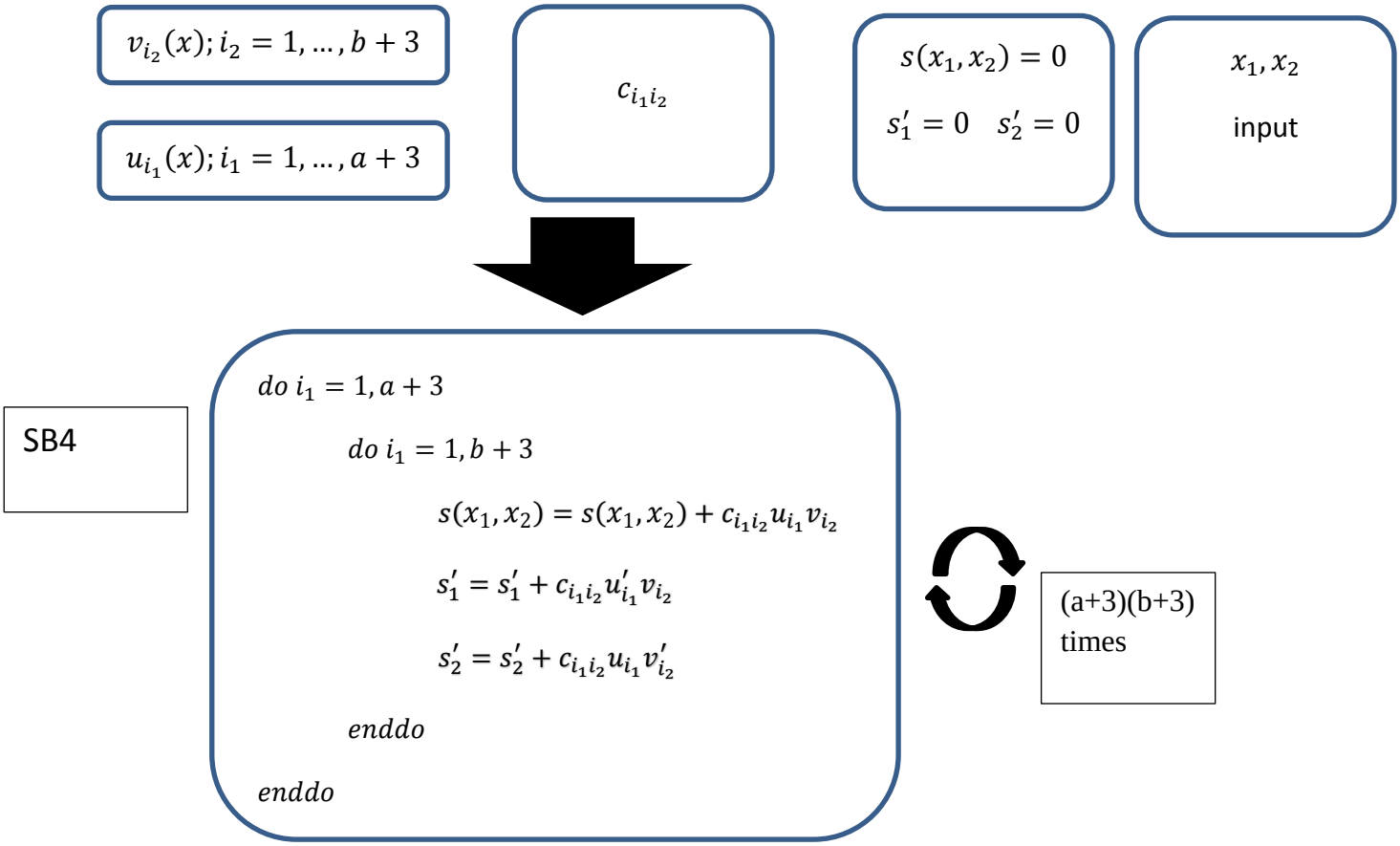


Figure 5.5: A flow-diagram of the linear combination step for two-dimensional B-spline interpolation

5.2.3 DATA GENERATION AND CALLING SPLINES

To generate the grid points, the maximum and minimum values of the interval, the number of points in the grid, and the analytic function forming the grid were computed in a single subroutine. In the case of a multivariate spline, the maximum & minimum values and the grid points were computed in every dimension.

Three analytic functions were used to construct a set of grid points. The aim was to understand the performance of interpolation at various gradients, inflection points and jagged regions. Firstly, two univariate functions with the features described above were considered. A cosine function, $y = \cos(x)$ and a polynomial $y = \frac{x^2}{(x-10)^2+1}$ are decent choices for comparison because they both have a big range of gradient and big data-jumps at their inflection points. In the case of multidimensional interpolation, the two-dimensional analytic

function $z = \cos(xy) + \frac{y^2}{(x-10)^2+1}$ was used to compare the performances of the two methods in a jagged multidimensional surface.

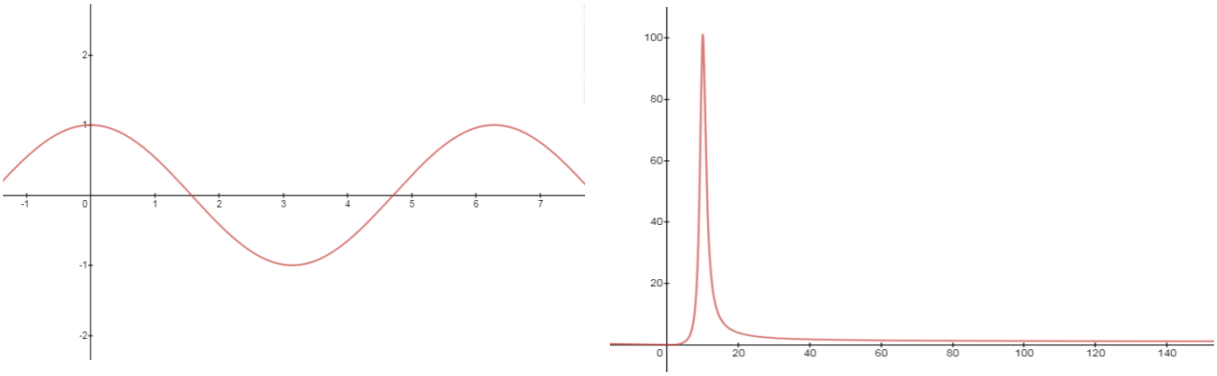


Figure 5.6: The Univariate Functions that were used for comparing two spline interpolation grids. The graphs were created using the graphing website Desmos

Figure 5.7 illustrates the processing of the grid-data by the main program. Inside the main subroutine, the set of grid points (x, y) are mapped onto separate arrays for the x and the y values. The subroutine also requires the entry of an x value for which an interpolated value is to be calculated. Once the input values are provided, the program calls the spline subroutines for the interpolation and gives an output of the interpolated value and the partial derivatives.

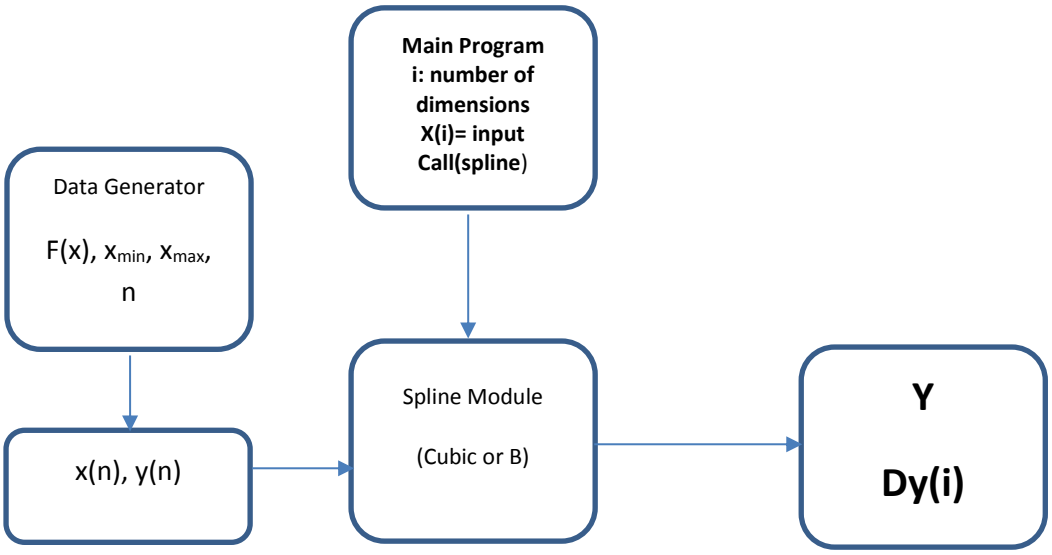


Figure 5.7: A flow diagram depicting the connection between the main subroutine, the grid-generating subroutine and the spline subroutines.

5.3 ACCURACY

5.3.1 METHOD OF ASSESMENT

The ideal way of assessing the accuracy of an interpolation method is to generate a set of data points by first calculating the values using an analytical function and interpolating at the same points. When these two sets are graphed and overlaid, significant overlap is expected if the method is accurate.

A set of values were initially calculated from an analytical function and graphed using MATLAB[108]. Using the spline subroutines and the equidistant grids – generated in FORTRAN using the same analytical function – the interpolated values of the same data points were computed. Finally, the graph for the interpolated values was overlaid with that of the analytical values. The procedure was repeated for the other univariate function and the multivariate function. Figures 5.8-a and d illustrate the overlap between the analytic values and the cubic-spline-interpolated values for the two univariate functions, while figures 5.8-b and e show the overlap between the analytic values and the B-spline-interpolated values.

The percentage error of interpolation at the data points was calculated using the equation 5.1;

$$Error\% = 100 * \frac{real\ value - interpolated\ value}{real\ value} \quad (5.1)$$

Two sets of percentage errors were calculated for the two splining methods and their graphs were overlaid – as shown in Figure 5.8-c&f – to see if they provided the same accuracy.

The accuracy of the derivation required the same approach with the exception that the first derivatives of the analytical functions were used. The overlap of analytic values with interpolated values, as well as the percentage errors for both functions, are shown in Figure 5.9-a – f.

5.3.2 RESULTS

Figures 5.8-a – f illustrate the interpolation of 501 data points using 51 equidistant grid points so that ten data points are interpolated in each interval.

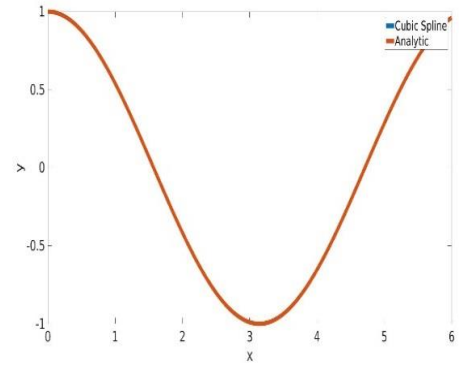
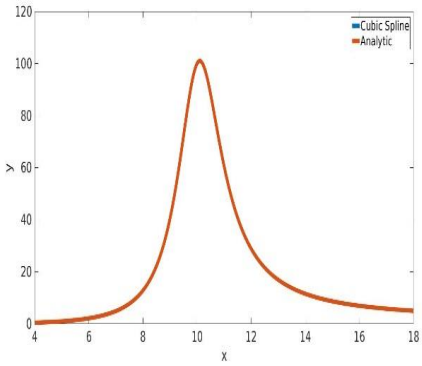
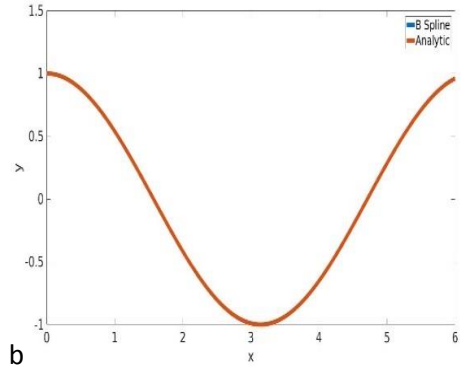
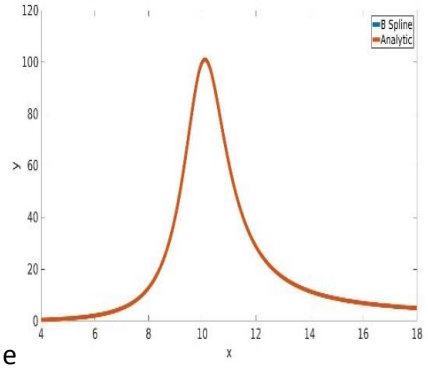
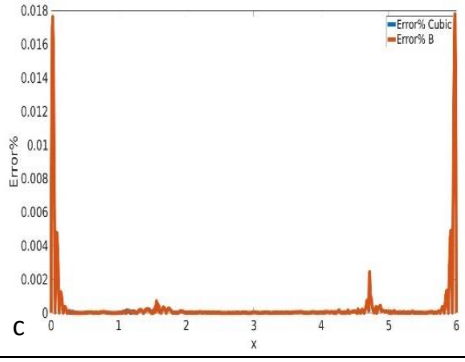
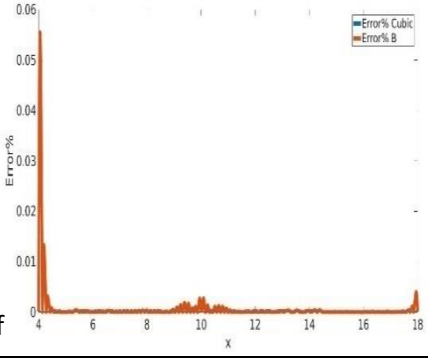
	$y = \cos(x)$	$y = \frac{x^2}{(x-10)^2+1}$
Analytic Data Cubic Spline Data	 a	 d
Analytic Data B-Spline Data	 b	 e
Error%	 c	 f

Figure 5.8: A set of graphs comparing the accuracy of the Cubic Spline and B-Spline interpolation methods on univariate function datasets.

Figures 5.9-a – f illustrate the interpolation of the derivatives.

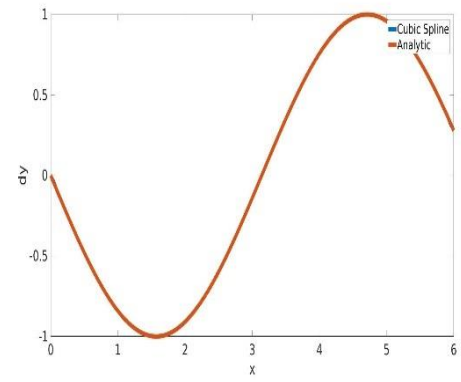
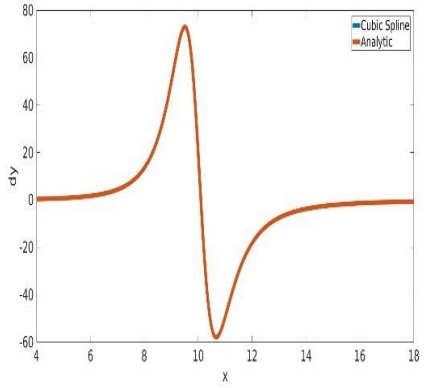
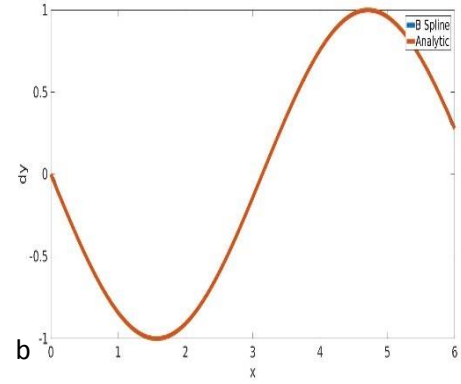
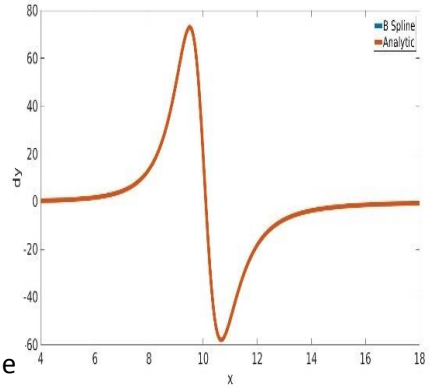
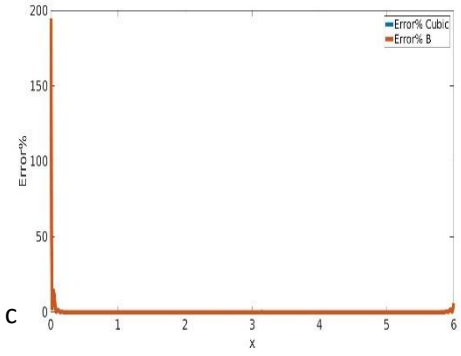
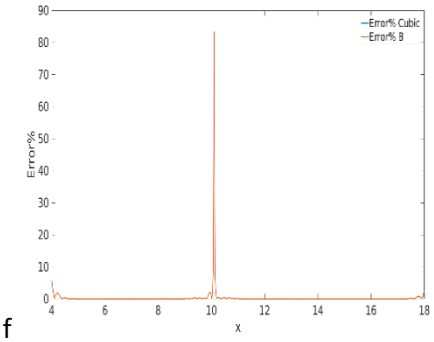
	$y = \cos(x)$	$y = \frac{x^2}{(x-10)^2+1}$
Analytic Data Cubic Spline Data	 a	 d
Analytic Data B-Spline Data	 b	 e
Error%	 c	 f

Figure 5.9: A set of graphs comparing the accuracy of the Cubic Spline and B-Spline interpolation methods in taking derivatives on univariate function datasets.

In the case of the multivariate function, the contour plots generated from the analytical values and the two interpolation methods are shown in Figure 5.10. The analytic derivative values and the interpolated values were overlaid for the two partial derivatives as shown in Figure 5.11

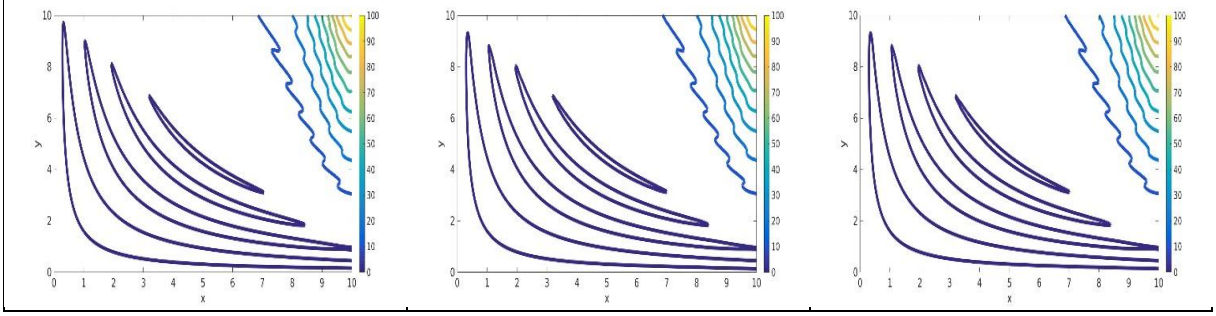
Analytic Data	Cubic Spline Data	B-Spline Data
		

Figure 5.10: A set of contour plots comparing the accuracy of the Cubic Spline and B-Spline interpolation methods on the multivariate function dataset

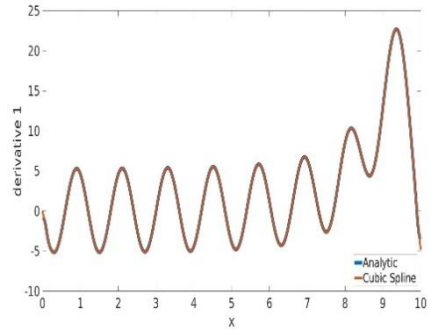
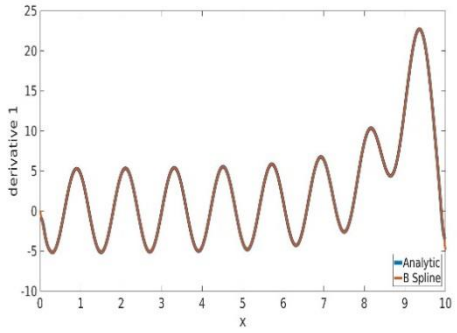
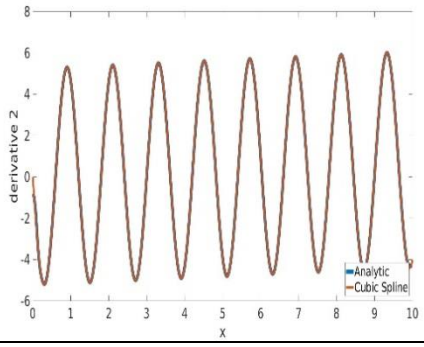
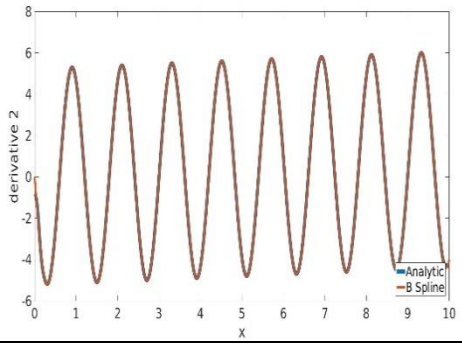
	Cubic Spline	**B-Spline**
Derivative 1		
Derivative 2		

Figure 5.11: A set of graphs comparing the accuracy of the Cubic Spline and B-Spline interpolation methods in taking partial derivatives of the multivariate function dataset

The results indicate that both interpolation methods are highly accurate in one dimension and two dimensions. The complete overlap of the two percentage error curves indicates that the methods are equally accurate. This statement also applies to the derivatives and to each partial

derivative of the multivariate function. Overall, the B-spline method can be used in place of the cubic spline method without sacrificing accuracy since they both have equal performance.

Despite the equal performance of accuracy, the methods exhibit deficiencies at certain regions.

The poor performance of both methods at the endpoints can be recognized from the percentage error graphs, where the error% is above 10%. This can be attributed to the assumption of the natural spline stating that the second derivatives at the endpoints are zero. This assumption is related to the case of the elastic band mentioned in Chapter 2. The elastic spline band that is constrained to go through given knots has certain strain energy increasing with the fluctuation of the knot positions. Because there are no knots beyond the end-knots, the constraint is no longer valid. Without an external force, the band becomes linear beyond the endpoints to minimize the overall strain energy, making the second derivatives at these points equal to zero. By the same logic, the second derivatives are assumed to be zero to minimize the overall curvature of the splines. This is applicable if the gradient near an endpoint is smooth, but it leads to unrealistic extrapolation beyond an endpoint that has a steep gradient nearby [109].

Another issue regarding the endpoints is that the B-spline method performs more poorly than the cubic spline method. The cubic spline interpolation has a self-deterministic way of calculating the second derivatives at the endpoints to account for the errors. The second derivative is calculated from the first derivative if the gradient is smaller than a threshold value; the value is set to zero otherwise. On the other hand, the B-spline algorithm sets these values to zero and uses the natural splines at the endpoints regardless of the gradient. Overall, the cubic spline method's self-deterministic approximation is more efficient at the endpoints. If the B-spline method applies a similar approach, the errors at the endpoints can be reduced.

Another region with poor performance is the steepest region of a function, where the concavity changes. For $y = \cos(x)$, the second derivative $y'' = -\cos(x)$ is zero at $\frac{\pi}{2} \cong 1.57$ and at $\frac{3\pi}{2} \cong 4.71$. For $y = \frac{x^2}{(x-10)^2+1}$, the second derivative $y'' = \frac{2(20x^3-303x^2+10201)}{(x^2-20x+101)^3}$ is zero at the points $x \cong 9.51$ and $x \cong 10.67$. Indeed, the percentage error is higher at these points compared to the rest of the values – excluding the endpoints. This confirms the deficiency of both interpolation methods at the points of inflection.

The poor performance at a steep gradient has been observed in other works that employed either cubic [110, 111] or B-spline methods [112]. In a study conducted by Zhang and Martin[113]**,** the overshoot and oscillation of the cubic spline interpolation near a steep gradient was attributed to the Gibbs Phenomenon[114], which states that the partial Fourier sums of a function near a discontinuity diverge from the true value of the function. In an earlier study, it was shown that the Gibbs Phenomenon can also apply to spline interpolation with equal grid spacing[115]. These studies concluded that the cubic spline interpolation oscillates near the points of discontinuity with an overshoot proportional to the height of the discontinuity. In addition, the oscillation was maximized as the number of grid points was increased to infinity. Although the analytical functions shown in this chapter are not discontinuous, the set of grid points form discontinuity due to the jumps between consecutive grid points at a steep gradient.

Minimizing the errors at the endpoints and the inflection points is essential for optimizing the accuracy. One way to reduce the errors is to adjust the number of grid points.

For the function $y = \frac{x^2}{(x-10)^2+1}$, 501 points were interpolated using 51 grid points. Then, the number of grid points was changed to 21, 101 and 401 to see the change in percentage error. The results for the values and their derivatives are shown in Figures 5.12 and 5.13 respectively.

The percentage errors at the endpoints and the inflection points diminish with the increasing number of grid points. On the other hand, using an excessive number of grid points leads to oscillations, generating increased errors at the non-critical regions. For instance, the range of error near x=5 and x=8 is 2-3x10^{-4} when 101 points are used; however, the error increases to nearly 5x10^{-4} when 401 points are used. The same trend can be observed in the percentage errors of the derivatives. The percentage of errors at the endpoint and the inflection points decrease to 2-4 % as the number of knots increase; however, small fluctuations can be seen at around x=14 when 401 points are used. This indicates that both spline interpolation methods suffer from over-fitting when the grid point-spacing is too narrow.

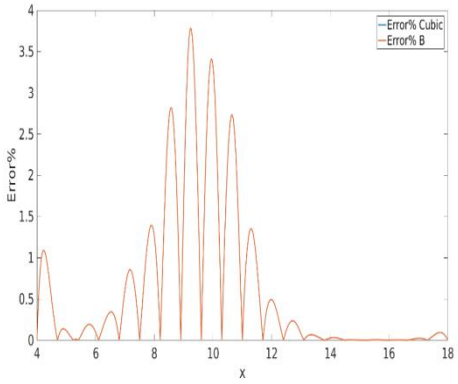
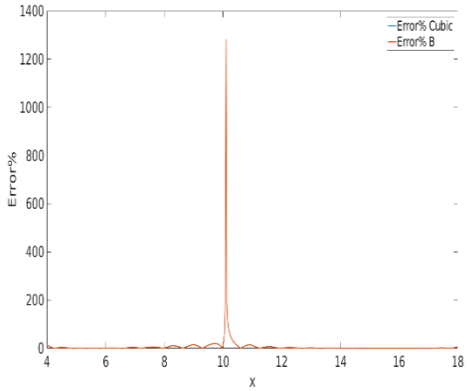
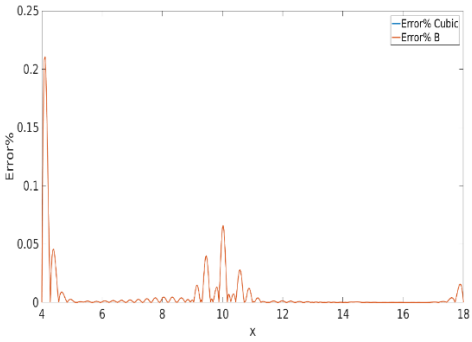
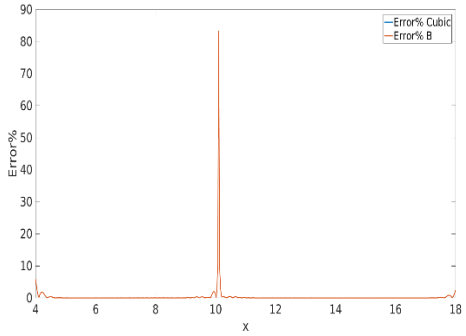
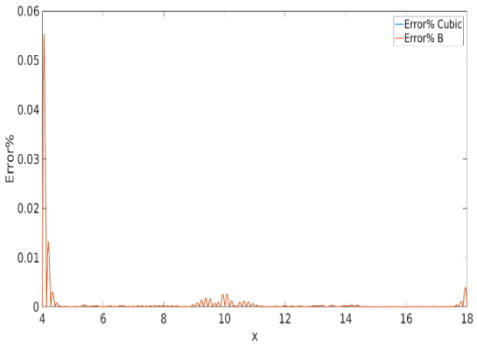
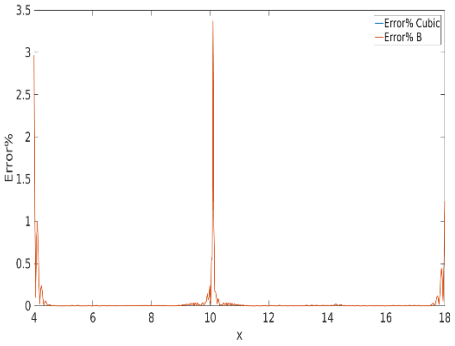
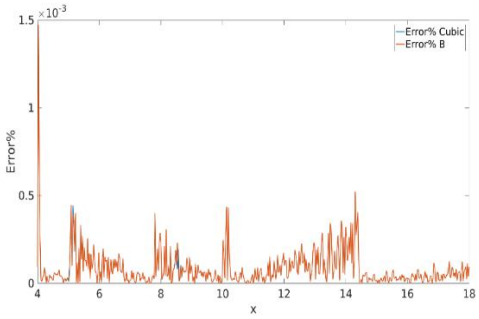
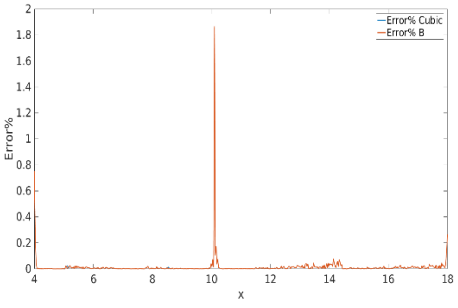
Knots	$y = \frac{x^2}{(x-10)^2+1}$	$y' = -\frac{2x(10x-101)}{(x^2-20x+101)^2}$
21		
51		
101		
401		

Figure 5.12: A set of graphs depicting the percentage errors obtained from the two spline interpolation methods depending on the number grid points.

The problem of overfitting can be resolved by finding the optimum grid-spacing as emphasized in previous studies. For linear spline interpolation, it was shown that datasets with rapid changes and steep gradients required more grid points, whereas smooth regions required a more scattered grid point positioning[116]**.** A similar conclusion was reached in cubic and B-spline interpolations in a study conducted by Foley and Nielson[44].

When a larger number of grid points were used, each interval would have fewer data points to interpolate due to narrower spacing. This approach is useful for the points along steep regions because a spline curve captures the fewer data points more easily; however, the same flexibility affects the smoother parts. In other words, the spline curves would try to pick up more details in every interval, leading to more oscillations at regions where detailed fitting is not necessary.

Maximum accuracy in splining depends on the correct choice of the number of grid points to capture the overall behaviour of the dataset as well as the local fluctuations and sharp changes[44]. Research suggests that the optimum grid size must be determined such that there is a minimum of 4-5 data points to interpolate in every interval[117]**.**

5.4 SPEEDUP

5.4.1 METHOD OF ASSESMENT

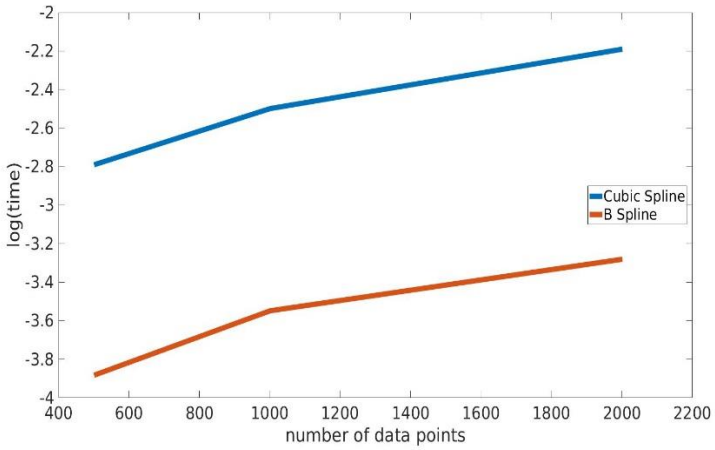
A comparison of the speed between the cubic spline and B-spline methods was conducted using the built-in Fortran "cpu_time" function. The elapsed time for the interpolation of all the data points was estimated. For each case, the estimation was repeated five times and the average time values were taken.

5.4.2 RESULTS

In Chapter 2, it was hypothesized that the B-Spline interpolation is faster compared to the cubic-spline interpolation because of the simplifications in its formulation. To test this hypothesis, the three analytical functions must be tested. More importantly, it is valuable to understand the effects of parameters such as the number of data points, the number of grid points, and the dimensionality on the computational speed. The elapsed time was evaluated in three different scenarios for the three parameters.

To assess the change of speed with the increasing number of data points, 501 points were initially interpolated using 51 grid points. Following this, the size of the data set was

increased to 1001 and 2001 keeping the number of the grid points constant. The results are shown in Table 5.1

Number of Data Points	Time Elapsed (s) for 51 grid points for $y = \cos(x)$	
	Speed Up	
501	12.4	
1001	11.2	
2001	12.3	

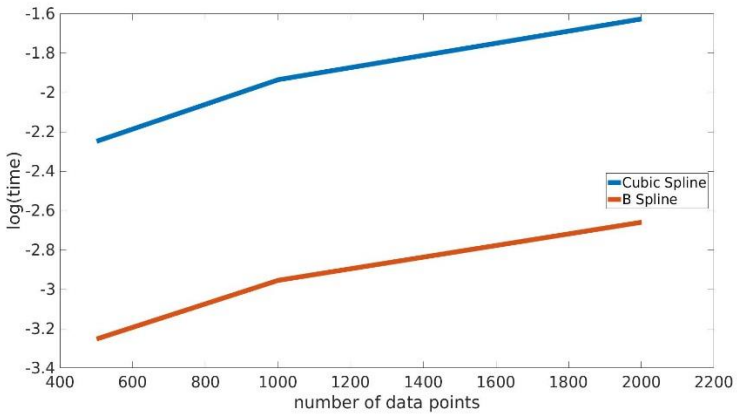
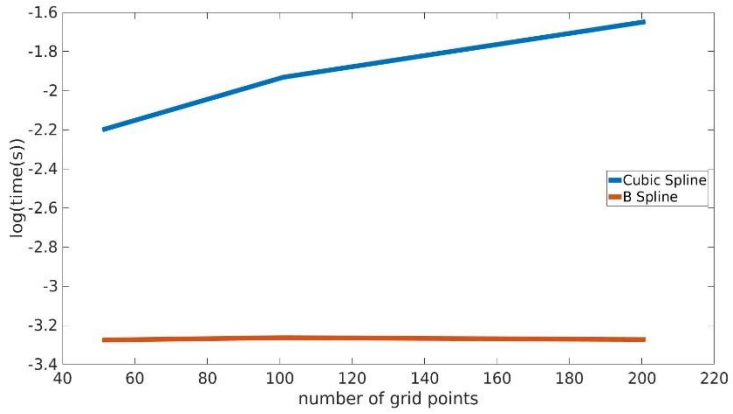
Number of Data Points	Time Elapsed (s) with 51 grid points for $y = \frac{x^2}{(x-10)^2-1}$	
	Speed Up	
501	10.09	
1001	10.45	
2001	10.78	

Table 5.1: A table illustrating the change of speed and speedup with the increasing number of data points.

The elapsed time for both interpolation methods increased linearly with the size of the data set. Therefore, the speedup achieved by using the B-spline method was not affected by the number of data points.

For the assessment of the speed depending on the number of grid points, 2001 data points were interpolated using 51 grid points. Keeping the data set fixed, the number of grid points was increased to 101 and 201. The results are shown in Table 5.2

Number of Grid Points	Time Elapsed (s) for interpolating 2001 data points for $y = \cos(x)$	
	Speed Up	
51	11.9	
101	21.5	
201	42.2	

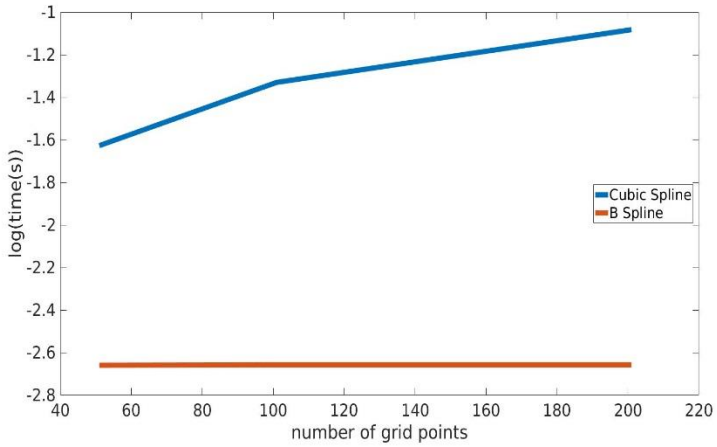
Number of Grid Points	Time Elapsed (s) for interpolating 2001 data points for $y = \frac{x^2}{(x-10)^2-1}$	
	Speed Up	
51	10.78	
101	21.27	
201	37.64	

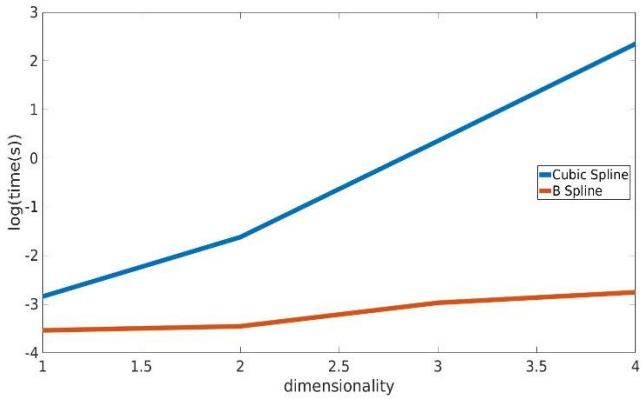
Table 5.2: A table illustrating the change of speed and speedup with the increasing number of grid points.

Increasing the number of grid points had a bigger impact on the speedup than the previous scenario. The time that elapsed in the cubic spline interpolation method increased linearly with the number of grid points. On the contrary, increasing the number of grid points did not affect the elapsed time in B-spline interpolation. As a result, the speedup coming from the B-spline method increased linearly with the number of grid points.

The final scenario involved the effect of dimensionality. Firstly, 51 points were interpolated using 11 grid points for the two univariate functions. Then, the procedure was extended up to four dimensions with the same number of data points and grid points in every dimension. For instance, the four-dimensional extension for the univariate function $y = \cos(x)$ would be,

$y = \cos(x_1) + \cos(x_2) + \cos(x_3) + \cos(x_4)$ with a data set of size 51^4 interpolated with 11^4 grid points. The results are shown in Table 5.3

Dimension	Time Elapsed (s) for $y = \cos(x_1) + \cdots \cos(x_i)$	
	Speed Up	
i=1	4.97 ~ 2^2	
i=2	68.01 ~ 2^6	
i=3	$2.1\text{x}10^3$ ~ 2^{11}	
i=4	$1.28\text{x}10^5$ ~ 2^{17}	



Dimension	Time Elapsed (s) for $y = \frac{x_1^2}{(x_1-10)^2-1} + \cdots + \frac{x_i^2}{(x_i-10)^2-1}$	
	Speed Up	
i=1	4.17~ 2^2	
i=2	58.5~ 2^6	
i=3	$2.02\text{x}10^3$ ~ 2^{11}	
i=4	$1.34\text{x}10^5$ ~ 2^{17}	

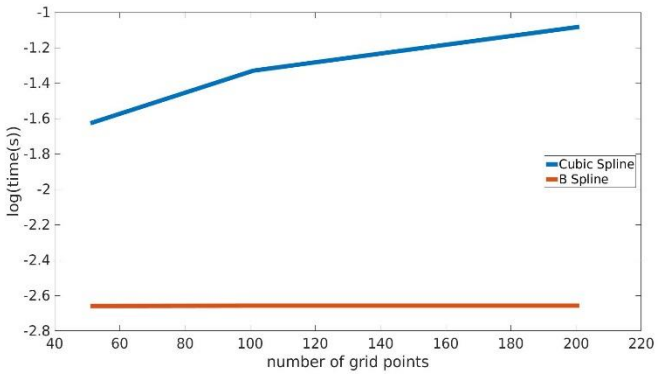


Table 5.3: A table illustrating the change of speed and speedup with the increasing dimensionality

The speed up between the B-spline and cubic spline interpolation increased exponentially as the number of dimensions was increased from one to four.

Changing the number of data points does not affect the speed up. If the methods calculate the value at one point through one interpolation, they both have to perform as many interpolations as the number of data points.

The major advantage of B-Spline interpolation is that changing the number of grid points does not affect the computational speed, while the speed of the cubic splining is drastically reduced

with the increasing number of grid points and the increasing dimensionality. The iterative structures of both spline methods and their subroutines can provide a better understanding of this slowdown.

The subroutines that have a larger number of iterations become more time-dominating. These are the subroutines where a system of equations is solved by employing a tridiagonal matrix. Through the system of equations, the cubic spline algorithm calculates the second derivatives at the grid points, while the B-spline algorithm calculates the coefficients needed for the linear combination technique. When the number of grid points is changed, this affects the number of iterations in the subroutines S1 and SB1. The cubic spline algorithm has to calculate a larger number of second derivatives and the B-spline algorithm has to compute a higher number of coefficients.

Although both methods use the same approach of tridiagonal matrices, they differ in the computation of the non-zero elements inside the matrix. The cubic spline algorithm computes the non-zero elements on each row iteratively, whereas the B-spline method pre-determines three non-zero elements in the beginning and applies them in every row. This calculation requires the storage of only two elements: the diagonal element and the sub-diagonal element that is symmetrical about the diagonal[47]. On the other hand, the tridiagonal matrix in the cubic spline method requires the storage of the elements in every row. This leads to a higher use of memory on the chip, hence the slowdown.

The determination of the second derivatives at the endpoints is another factor contributing to the time difference. The cubic spline method uses a conditional statement that depends on the value of gradient while the B-spline method assumes that they are equal to zero regardless of the gradient. The conditional approach that involves an “if-else” determination leads to a further slowdown because it involves an extra amount of calculation and memory usage [118]. The small difference caused by the determination of endpoints can be trivial in a single interpolation, but repeated one-dimensional interpolations will be affected significantly.

Multidimensional interpolation involves the iterative usage of subroutines leading to further slowdown for the cubic spline method with an intertwined mechanism, where both subroutines S1 and S2 are used sequentially and iteratively. To compute one x_1 array of the two-dimensional x_1x_2 grid, S1 is run to get the second derivatives and S2 is performed to get an interpolated value representing that array. To compute an interpolated value for all the x_1

arrays, S1 and S2 must be run repeatedly until the entire grid is scanned. The set of interpolated values are passed on for the final interpolation – using S1 and S2 once more – to get the interpolated value and its first partial derivative. To obtain the other partial derivative, this iterative procedure has to be repeated starting from the arrays in the direction of x_2. For higher dimensions, the iterations become more nested to reduce the dimensionality of the grid to a two-dimensional problem. Furthermore, there would be a higher number of partial derivatives to calculate, requiring the repetition of the calculations for every partial derivative. Therefore, the cubic spline interpolation method is not feasible for multidimensional problems as it involves an iterative use of nested loops.

The main advantage of the B-spline algorithm is the ability to compartmentalize the two subroutines. The coefficients are calculated in the subroutine SB1 only once regardless of the number of dimensions. This makes the final interpolation more straightforward. In addition, the B-spline algorithm has the ability to compute all the partial derivatives in one round since they can be directly calculated by using the derivatives of the basis functions instead of the functions themselves.

The overall advantage of the B-spline interpolation depends on the specific case of implementation. If the numerical data is only one dimensional and one wants to calculate an interpolated value without the derivative, the slowdown in the cubic spline interpolation is not a major issue. By contrast, using the B-spline algorithm provides a more practical implementation for a multi-dimensional array and calculation of the partial derivatives.

5.5 RESOURCES

1. Press, W.H., et al., *Numerical recipes in Fortran 90.* The art of scientific computing,(Cambridge, 1996), 1992.

2. Habermann, C. and F. Kindermann, *Multidimensional spline interpolation: Theory and applications.* Computational Economics, 2007. **30**(2): p. 153-169.

3. Mathews, J.H. and K.D. Fink, *Numerical methods using MATLAB.* Vol. 3. 2004: Pearson Prentice Hall Upper Saddle River, NJ.

4. Adams, K., *Smooth interpolation of zero curves.* Algo Research Quarterly, 2001. **4**(1/2): p. 11-22.

5. Gilsinn, D.E., M.A. McClain, and C. Witzgall, *Non-Oscillatory Splines on Irregular Data.*

6. Zhang, Z., J. Tomlinson, and C. Martin, *Splines and linear control theory.* Acta Applicandae Mathematica, 1997. **49**(1): p. 1-34.

7. Gang, X. and W. Guo-Zhao, *Extended cubic uniform B-spline and α-B-spline.* Acta Automatica Sinica, 2008. **34**(8): p. 980-984.

8. Zhang, Z. and C.F. Martin, *Convergence and Gibbs' phenomenon in cubic spline interpolation of discontinuous functions.* Journal of Computational and Applied mathematics, 1997. **87**(2): p. 359-371.

9. Richards, F., *A Gibbs phenomenon for spline functions.* Journal of approximation theory, 1991. **66**(3): p. 334-351.

10. Schoenberg, I., *Cardinal interpolation and spline functions.* Journal of Approximation theory, 1969. **2**(2): p. 167-206.

11. Jamrozik, J., J. Bohmanova, and L. Schaeffer, *Selection of locations of knots for linear splines in random regression test-day models.* Journal of Animal Breeding and Genetics, 2010. **127**(2): p. 87-92.

12. Foley, T.A. and G.M. Nielson, *Knot selection for parametric spline interpolation*, in *Mathematical methods in computer aided geometric design*. 1989, Elsevier. p. 261-CP4.

13. Wold, S., *Spline functions in data analysis.* Technometrics, 1974. **16**(1): p. 1-11.

14. Schmeisser, G., *A real symmetric tridiagonal matrix with a given characteristic polynomial.* Linear algebra and its applications, 1993. **193**: p. 11-18.

15. Openshaw, S. and I. Turton, *High Performance Computing and the Art of Parallel Programming: An Introduction for Geographers, Social Scientists and Engineers*. 2005: Routledge.

CHAPTER 6: THE COMPARISON OF THE SPLINE INTERPOLATION METHODS IN FEARCF

6.1 OVERVIEW OF THE COMPARISON

FEARCF[20] is an ideal test-case for performing a comparative study of spline interpolation methods. Its main advantage is the direct application of the driving forces on the Cartesian coordinates unlike the other flat histogram methods. The derivation of these forces is straightforward due to the spline interpolation method that can easily be extended into multiple dimensions. Furthermore, it has a time-saving embarrassingly parallel algorithm. These features enable the investigation of reaction mechanisms in greater detail with more reaction coordinates as opposed to the flat histogram methods that operate only in one or two dimensions.

The feasibility of the cubic spline method was evaluated, and the B-spline method was proposed as an alternative method in Chapter 5. In this chapter, the spline comparison is taken one step further and incorporated into the free energy calculations of FEARCF, which currently uses a cubic spline routine.

Using FEARCF, the free energy profile of a simulated reaction mechanism was constructed. The proton exchange between ammonium and ammonia was simulated in a vacuum using Molecular Mechanics methods and Quantum Mechanical Methods. The first simulation aimed to capture the vibrational movements such as stretching, bending, and torsion, while the second simulation aimed to construct the free energy of the bond breaking & forming. The aim of testing spline interpolation with both MM and QM methods was to demonstrate the overall time-contribution of interpolation in the reaction sampling. The sampling procedure has many steps some of which are more time-consuming than the others. Therefore, speeding up the numerical interpolation does not have a significant contribution if there are other time-dominating steps. It is essential to demonstrate the interpolation in different case scenarios where atoms are treated differently to comprehend when the B-spline speedup would be the most useful.

6.2 FEARCF

The current FEARCF source code exists as a library of files each performing a different task for the execution of the sampling.

6.2.1 FEARCF LIBRARY AND RECENT MODIFICATIONS

There have been some significant changes in the FEARCF package since its initial development to make the FEARCF more up-to-date and precise.

The FEARCF library used in previous works[72] was only able to define bond distances as reaction coordinates; however, it was recently upgraded at the Scientific Computing Research Unit to work with angular reaction coordinates. This enabled the investigation of vibrations, bending and torsion angles as well as the movement of the centre of masses. This improvement in FEARCF made the method perfectly relatable to the spline comparison discussed in Chapter 5 because the numerical interpolation of free energy surfaces with cyclic and non-cyclic coordinates provide further support for the interpolation of analytic functions containing cosine (non-linear) and polynomial (linear) terms.

After incorporating the B-spline subroutines into the library, the modified FEARCF was compiled with the CHARMM software that was used to get force fields and perform an MD simulation. In this dissertation, the version CHARMM41 was used.

6.2.2 FEARCF SOURCE CODE

The FEARCF source code was written in Fortran 90, including modules with a specific task assigned to each of them:

- File 1: the FEARCF data type is defined as well as the reaction coordinates. The current version of FEARCF allows a maximum of six reaction coordinates.
- File 2: has a module that contains general information about FEARCF and the data types.
- File 3: is the subroutine that generates an interface in which the user can define the atoms of interest and the reaction coordinates.
- File 4: is the module that facilitates the successful processing of the data and defines the possible errors that might arise.

- File 5: contains the module that calculates the forces and applies them to the atoms to update the atomic coordinates.
- File 6: defines the essential formulas for PMF generation and force calculations.
- File 7: is the module that formulates the cubic splining.

The working mechanism of the source code is illustrated in Figure 6.1

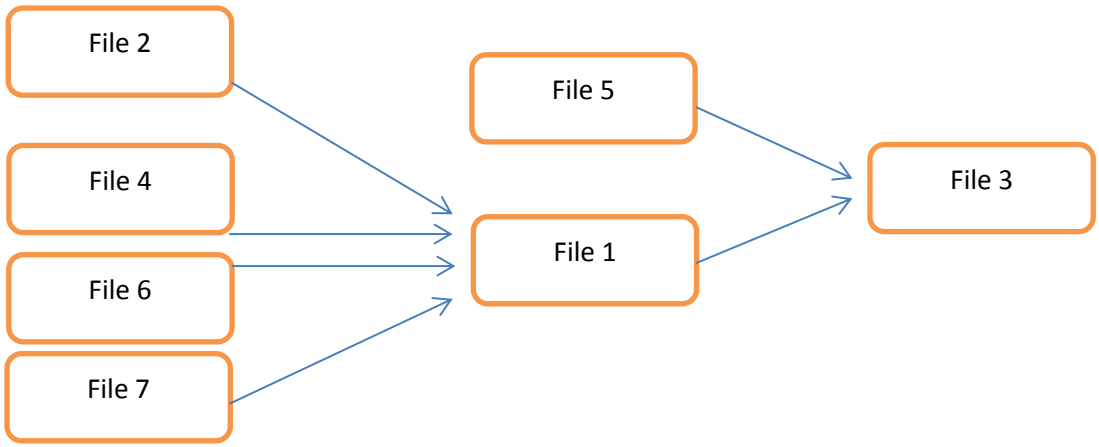


Figure 6.1: A flow diagram depicting the connections between the FEARCF modules

File 3 has the information about the FEARCF definitions and reaction coordinates to be used. This information is passed onto File 1 that calls the files 2,4,6 and 7 to perform the required calculations.

6.2.3 MODIFYING THE SPLINE MODULE IN FEARCF

The major modification for this study was implemented in File 7 that formulated the cubic spline interpolation of discrete bins of histograms. The B-spline algorithm was included in the new spline module as formulated and illustrated in Chapters 2 and 5.

Because the dissertation aims to compare the two splining methods, it was necessary to contain both spline algorithms in the source code instead of the complete replacement of the cubic spline with the B-spline algorithm. A conditional statement was created to make it quicker to switch on-off between the two spline methods. This enables to user to determine the spline type to be used in a separate input file by specifying the condition. The format and logic of the input file is explained further in section 6.2.5.4.

The same modification was also be applied in any file where the spline module is called. In File 1, the argument defining the spline was extended to introduce the conditional statement. File 3 that contains the setup and interface was modified accordingly.

6.2.4 CHARMM FORCE FIELDS

The CHARMM 41b2 release, which is one of the most recent releases, was used. This is the FORTRAN 95 conversion of CHARMM.

6.2.5 RUNNING CHARMM WITH FEARCF

6.2.5.1 REACTION MECHANISM

To test the effect of splining routine in the accuracy and speed of the sampling with the FEARCF method, the protonation of ammonia into ammonium was sampled. The cation Ammonium is obtained by the protonation of ammonia.

Ammonium is an essential part of many inorganic compounds that play an important role as a human metabolite. Ammonium is also an important source of nitrogen for plant species growing on soil with low oxygen levels[119]. Improving the sampling efficiency of this fundamental reaction mechanism can provide insight into many other complex reaction mechanisms involving the conversion between ammonia and ammonium.

Because ammonia is a weak base, it can react with Brønsted acids that contain proton donors.

$H^+ + NH_3 \rightarrow NH_4^+$

For the particular reaction sampling process, a conjugated structure of ammonium and ammonia was minimized in a vacuum. The atoms are labelled and shown in Figure 6.2

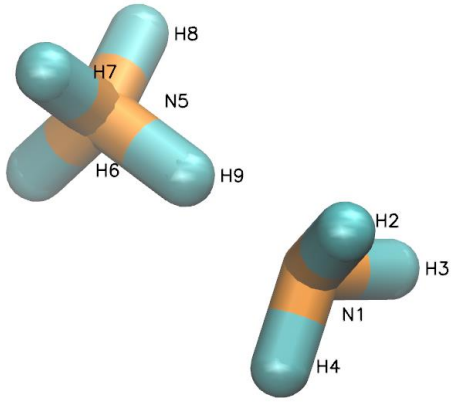


Figure 6.2: A Virtual Molecular Dynamics Screenshot of NH_3+NH_4^+ proton exchange with the atoms labelled.

6.2.5.2 CLASSICAL MOLECULAR DYNAMICS

CHARMM-Gui[120] was used to model and obtain the coordinate files.

Periodic boundary conditions were set such that the dimensions of the unit cell in the lattice were 24.5 x 24.5 x 23.5 Å with a cut-off distance of 60 Å.

To keep ammonia and ammonium within a certain proximity of each other, a distance restraint was applied between N1 of NH_3 and H9 of NH_4 such that the two atoms were restrained to be 6.8 Å apart with a harmonic force constant of 500 kcal mol^{-1} $Å^{-2}$.

The FEARCF library was called before the dynamics to sample the interaction.

A $2x10^{-2}$ ns classical molecular dynamics simulation was performed for 20000 steps using a time step of 1 fs. The Verlet algorithm[121] was employed with a temperature of 298.15 K and a cut-off radius of 22Å with outer and inner cut-off distances set to 20 and 18 Å respectively. The coordinates were updated and written at every 100 steps, while the energy data was updated and written at every 500 steps. The root mean square (RMS) fluctuations of the major energy values were updated at every 1000 steps.

6.2.5.3 QM METHODS (AM-1)

The proton exchange between NH_3 and NH_4^+ requires the bond between H9 and N5 to break while another bond forms between N1 and H9. Molecular mechanics methods are inadequate for determining the electronic structure changes during the bond breaking – forming. Therefore, the atoms were treated quantum mechanically in another set of simulations for testing the modified FEARCF.

Because the system was small, all the atoms were included as part of the quantum mechanical region. Austin Model 1, AM1[122] was used as the semi-empirical method for the quantum calculation of the electronic structure. The Self Consistent Field- convergence was set as 10^{-7} with the overall charge of +1.

During the energy update, the inner cut-off distance was 10 Å, the outer cut-off distance was 12 Å, while the cut-off radius was 14 Å.

A $2x10^{-2}$ ns classical molecular dynamics simulation was performed for 20000 steps using a time step of 1 fs and the Verlet algorithm with a temperature of 298.15 K. The rest of the procedure was the same as that for the classical MD simulation.

6.2.5.4 FEARCF INTERFACE

Before the FEARCF interface, the reaction coordinates used to be specified within the CHARMM input file. With the creation of the interface module, the information about the reaction can be specified separately. This improvement makes FEARCF more user-friendly and practical.

An input file for the FEARCF specifications was created. The file contains information regarding the atoms of interest, reaction coordinates – which can be bond distance, bond angle, torsion or dihedral angle and centre of mass – and the spline type.

The input file gets called within the CHARMM input file before the reaction dynamics. This facilitates either classical dynamics or semiempirical QM methods to simulate the reaction so that FEARCF can start taking snapshots of the reaction coordinates. During the dynamics, the input written by the user gets passed onto File 3 in the FEARCF library and subsequently into the other modules for processing. The conditional statement in the spline routine enables the user to specify the spline module in the input file so that File 7 can perform either the cubic spline or the B-spline interpolation as requested by the user. A flow diagram of the process is shown in Figure 6.3

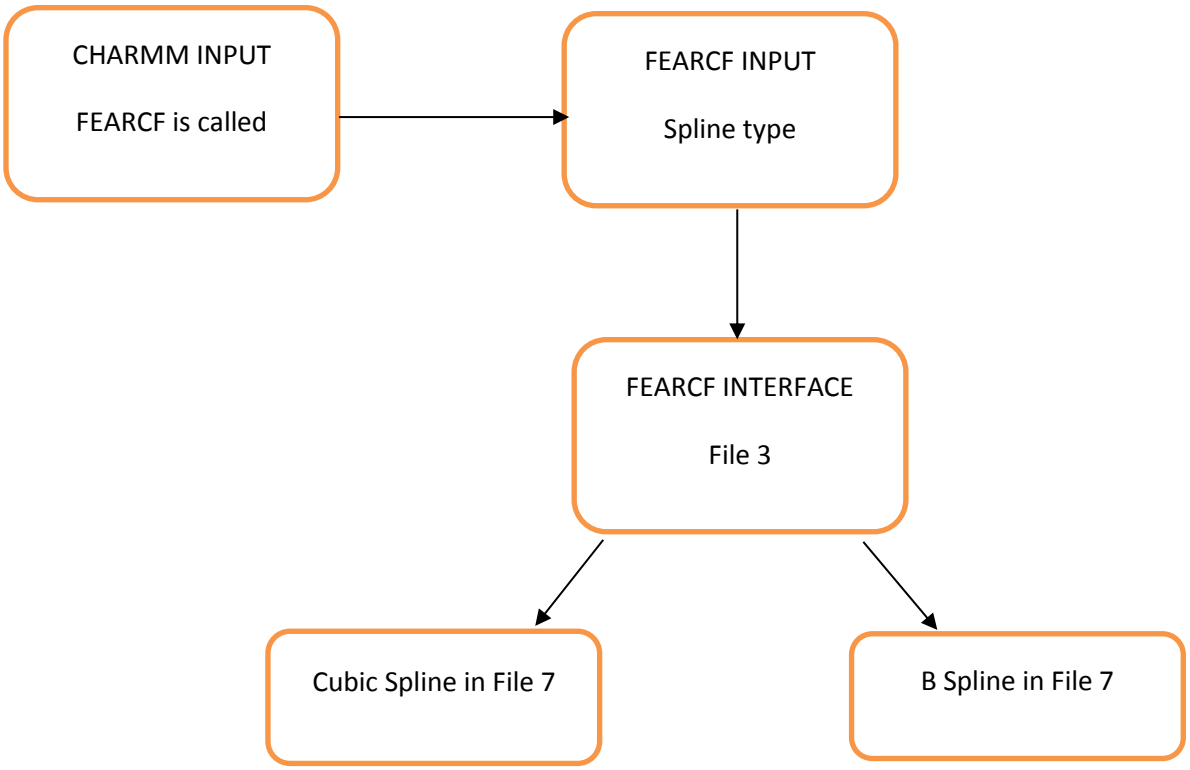


Figure 6.3: A flow diagram illustrating the decision-making process of the splining method to be used.

The sampling was performed first with distance coordinates and then with cyclic coordinates. The distance coordinates are the bond distances that vary as bonds stretch, break or form. The cyclic coordinates include bond angles and torsion angles.

In the case of distance coordinates, three scenarios were included. In the first one, the only reaction coordinate was the bond distance N5-H9. Then, the bond distance N1-H9 was added for two-dimensional sampling, followed by the addition of the bond distance N5-H7 for three-dimensional sampling. These reaction coordinates correspond to the spline interpolation test of the polynomial analytical function from Chapter 5 such that three reaction coordinates correspond to the three-dimensional polynomial function.

A similar approach was followed with the cyclic reaction coordinates. In the first scenario, the angle H6-N5-H7 was taken as the reaction coordinate. The second scenario involved two angles H6-N5-H7 and H7-N5-H9. Finally, the torsion angle between the central nitrogen atom N1 and the bond between the atoms N5-H7 was added. Because the coordinates are cyclic, these scenarios can be attributed to the spline comparison from Chapter 5 involving the cosine

function. Using three reaction coordinates corresponds to the three-dimensional cosine function.

The aim was to observe the time difference between the two interpolation methods and as the number of reaction coordinates – the dimensionality – increased.

A total of 20 FEARCF iterations were run for each scenario. The free energy simulations were carried out using 10 independent parallel reactions simulated on 10 cores using an embarrassingly parallel approach. WHAM[78] was used to overlap the histograms of parallel simulations and calculate the unbiased probability distribution. Each new iteration took the final coordinates of the previous iteration as the starting coordinates.

6.3 RESULTS

In classical molecular dynamics, the bending and stretching of the bonds were simulated using molecular mechanics methods of CHARMM Force Field. The atoms were restrained to a distance of 6.8 Angstrom and so that they would not move far away from each other and the simulation box. The dynamic bonds throughout one FEARCF iteration were visualized via Visual Molecular Dynamics[123] software as shown in Figure 6.4.

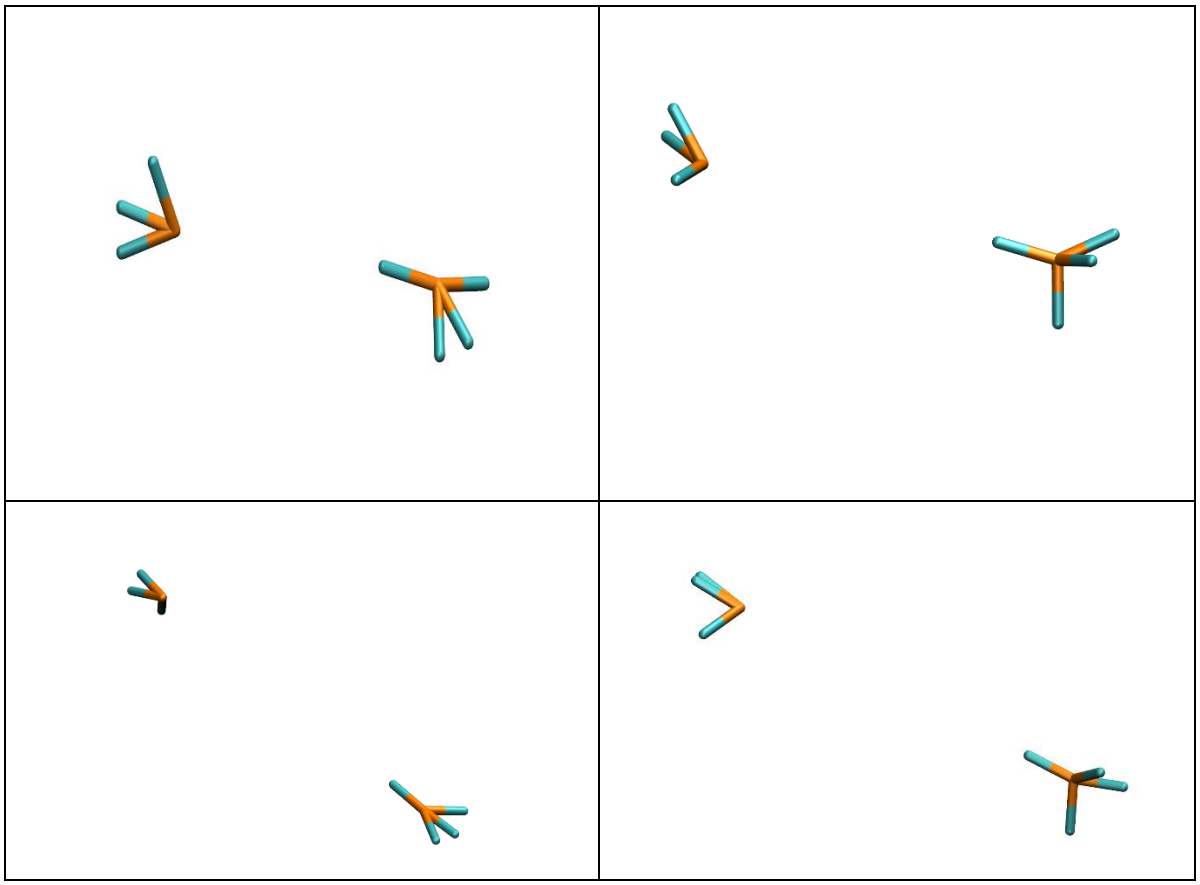


Figure 6.4: Snapshots from dynamic bonds of the interaction between Ammonium-Ammonia simulated with CHARMM Force Field.

The proton exchange was then simulated using the semiempirical QM method AM1. The motion of the dynamic bonds throughout a FEARCF iteration was visualized in Visual Molecular Dynamics software as shown in Figure 6.5.

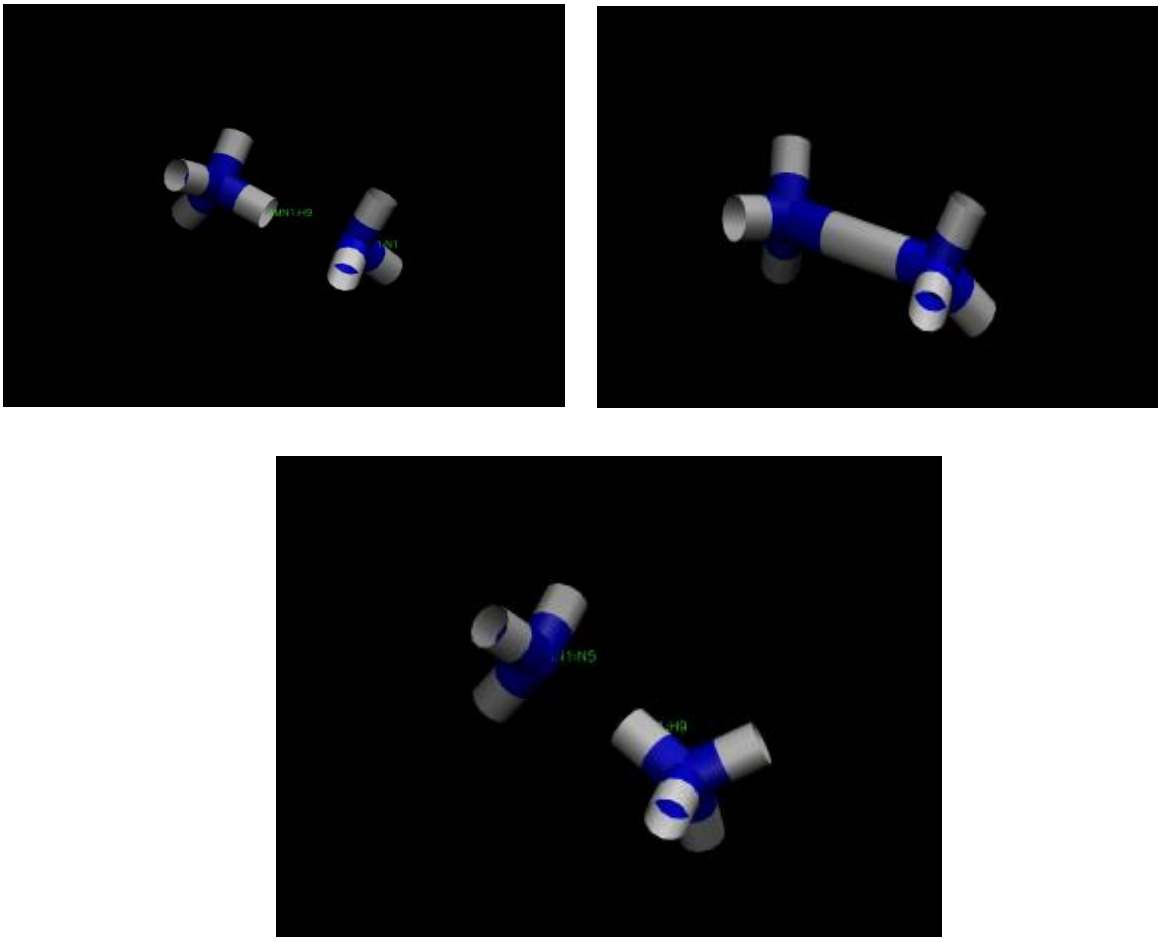


Figure 6.5: Snapshots from dynamic bonds of the interaction between Ammonium-Ammonia simulated with semiempirical QM method of AM1

6.2.1 EFFECT OF SPLINING ON PMF SURFACES

The aim of comparing cubic spline and B spline interpolation methods in the context of FEARCF was to observe the effect of the splining type on the PMF surfaces generated. In Chapter 5, changing the spline type did not affect the accuracy. Therefore, it is expected that the two PMF surfaces generated from the two interpolation methods would yield similar surfaces.

To plot the PMF surface as contours, two noncyclic reaction coordinates and two cyclic reaction coordinates were chosen: The distances between the bonds N1-H9 and N5-H9 were sampled for the first contour plot, while the angle between the bonds N5-H6 & N5-H7 and the torsion angle of atom N1 with respect to the N5-H7 bond were sampled for the second. Separate simulations were carried out for 2D noncyclic coordinates and 2D cyclic coordinates, both with classical MD and QM methods. For each scenario, two PMF surfaces were constructed by modifying the spline type.

The contour plots of the surfaces are shown in Figures 6.6 and 6.7.

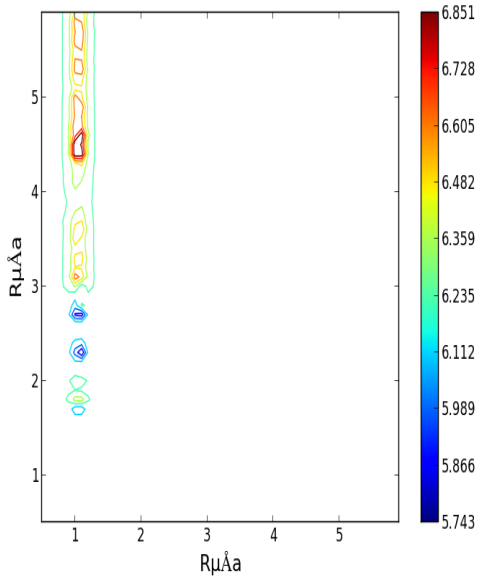
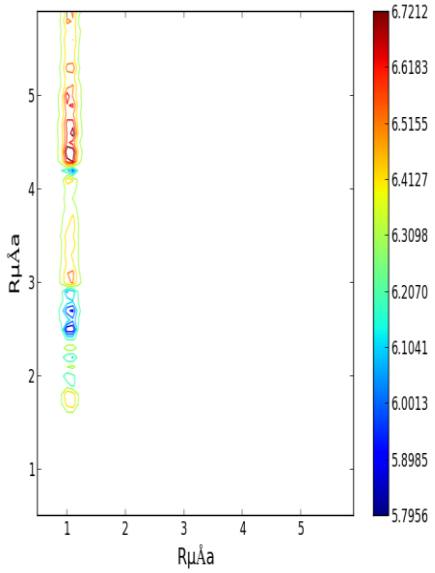
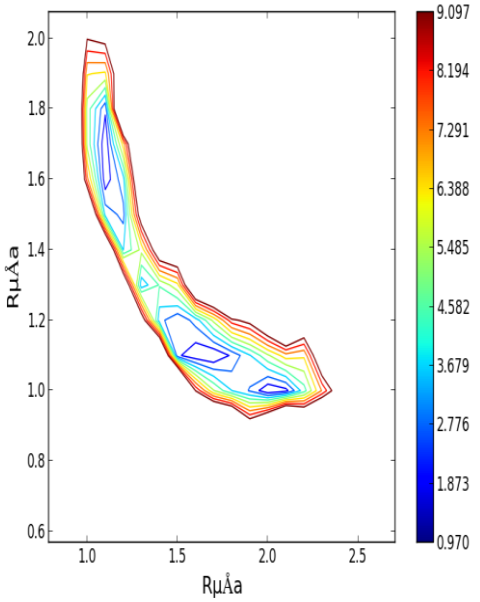
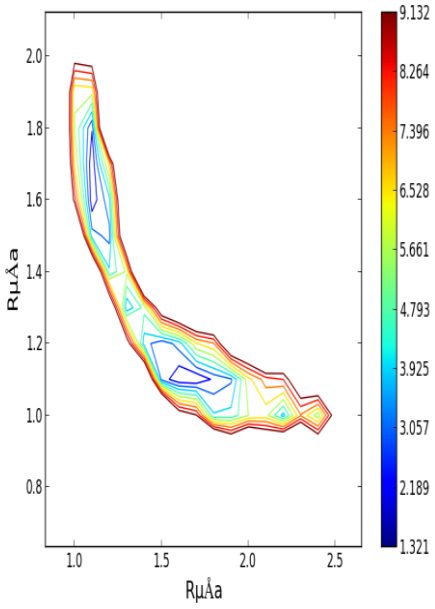
	CUBIC SPLINE	B-SPLINE
MM		
QM		

Figure 6.6: PMF surfaces of Ammonia-Ammonium interaction constructed from FEARCF sampling of Classical MD and QM runs with non-cyclic reaction coordinates

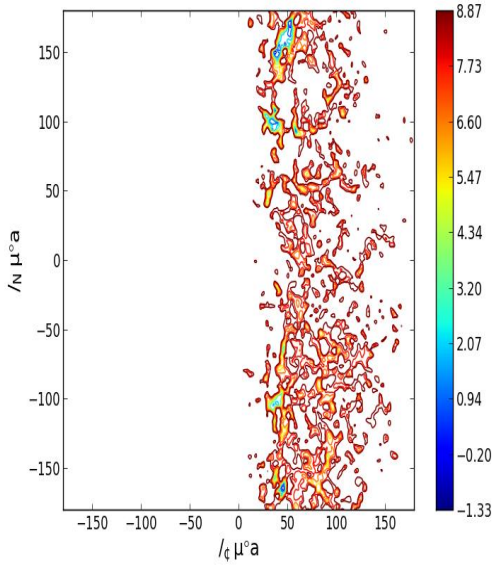
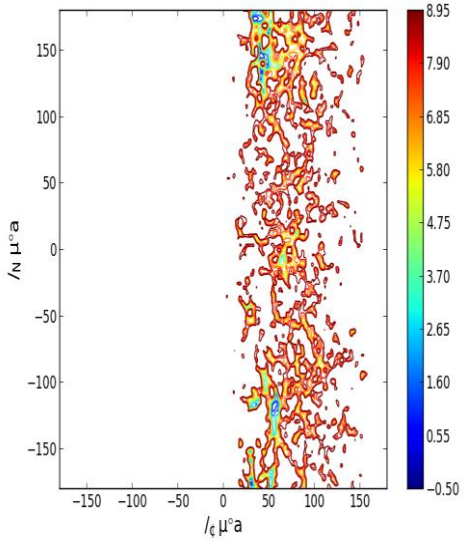
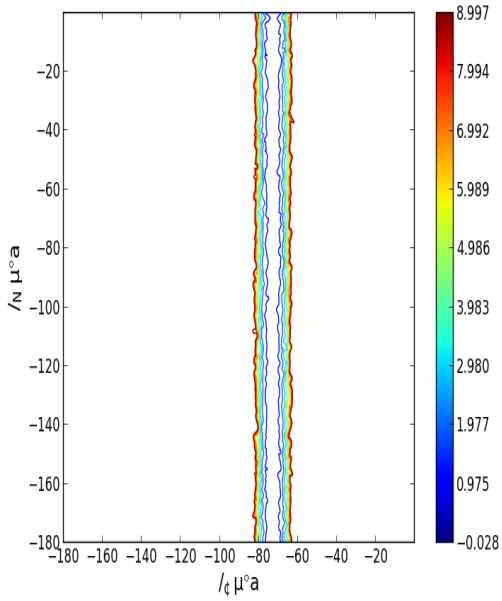
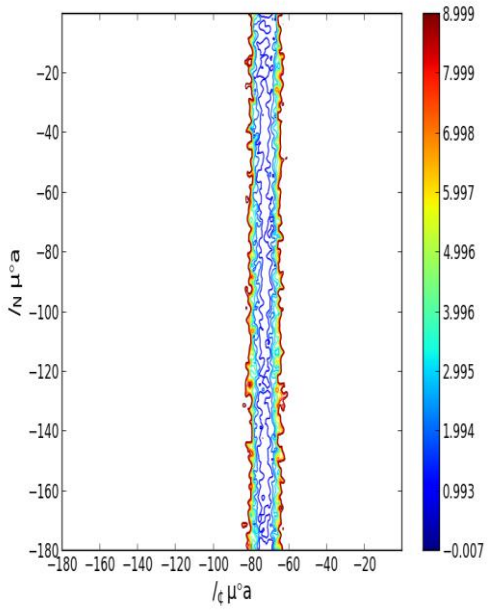
	CUBIC SPLINE	B-SPLINE
MM		
QM		

Figure 6.7: PMF surfaces of Ammonia-Ammonium interaction constructed from FEARCF sampling of Classical MD and QM runs with cyclic reaction coordinates

The initial biasing force before the sampling is set to zero. After the first iteration, several simulations are performed to obtain multiple histograms of probability distributions with discrete bins. These are combined through the Weighted Histogram Analysis Method (WHAM) to generate the average free energy surface. At this point, the values of free energies at every grid point are updated and the forces needed to drive the reaction for the next iteration are obtained by taking partial derivatives of the updated free energy surface. The forces are applied on the Cartesian coordinates of the atoms directly. This enables direct implementation of FEARCF in multiple dimensions without the addition of Jacobian terms for cyclic reaction coordinates. Therefore, it is easier to make a comparison of the two splining methods in multidimensional surfaces without any limitation.

Determining the accuracy of the spline methods in the free energy calculations was challenging because there was no reference structure to compare the interpolated surfaces to. In Chapter 5, the set of interpolated points were overlaid with the values obtained from the analytic functions. This is not the case in Chapter 6 since the free energy surface is the product of a numerically-generated probability distribution. Therefore, the surfaces cannot be expressed analytically. Nevertheless, a thorough comparison of the two interpolation methods was performed using analytic functions in Chapter 5. The results showed high accuracy, while the deficiencies at the endpoints and the points of inflection were emphasized. Most importantly, both surfaces demonstrated identical performance, indicating that the B-spline implementation does not reduce the accuracy. This claim can be confirmed by looking at the Figures 6.6 and 6.7, where both spline methods yielded similar PMFs.

6.2.2 EFFECT OF SPLINING ON RUN TIME

Changing the spline method has a bigger impact on the duration of the FEARCF iterations. In Chapter 5, it was proven that B-spline was significantly faster than the cubic spline. However, the ultimate goal is to confirm the same speedup effect in the free energy calculations with iterative flat histogram methods and large data sets. More importantly, it has to be ensured that speeding up the interpolation does contribute to the runtime of the entire MD simulation.

The CPU time for a single iteration of FEARCF was computed. The results are tabulated for the distance reaction coordinates and the angular reaction coordinates in both classical MD runs and QM runs as shown in Table 6.1 and Table 6.2 respectively.

Reaction Coordinate	Cubic (s)	B (s)	Speed Up
1 Bond Distance	0.802	0.680	1.18
2 Bond Distances	19.474	0.748	26.03
3 Bond Distances	1884	1.3161	1432
Reaction Coordinate	Cubic (s)	B (s)	Speed Up
1 Angle	0.93	0.85	1.09
2 Angles	20.01	1.02	19.62
2 Angles + 1 Dihedral	1899.60	1.63	1165.4

Table 6.1: Tabulated results for the comparison of the runtime in a FEARCF iteration depending on the splining method – simulated with classical MD methods.

Reaction Coordinate	Cubic (s)	B (s)	Speed Up
1 Bond Distance	8.08	7.95	1.02
2 Bond Distances	27.25	8.05	3.39
3 Bond Distances	2022	8.44	239.57
Reaction Coordinate	Cubic (s)	B (s)	Speed Up
1 Angle	8.13	8.05	1,01
2 Angles	27.09	8.28	3,27
2 Angles + 1 Dihedral	1911.00	8.91	214.48

Table 6.2: Tabulated results for the comparison of the runtime in a FEARCF iteration depending on the splining method – simulated with QM methods.

In both classical MD and QM methods, the effect of modifying the spline routine is more obvious in higher dimensions. Almost no speedup was acquired when only one reaction coordinate was used for sampling. As soon as another reaction coordinate was added, speedups of 20-fold and 3-fold were achieved for classical MD and QM respectively. The speedup effect was even more drastic when a third reaction coordinate was added, increasing up to 10^3 for classical MD and $2x10^2$ for QM.

The enhanced time differences in higher dimensions can be attributed to the iterative mechanism of the cubic spline. The two-dimensional cubic spline routine in FEARCF starts off by computing the second derivatives and free energy values with respect to the grid points

of one of the reaction coordinates. The interpolated free energy profile is then projected on the other reaction coordinate and used to interpolate for the two-dimensional free energy profile. This requires the application of the second-derivative-generation and interpolation subroutines repeatedly; yet, it only gives the force in one direction. To compute all the components of the force, the same interpolation algorithm must be applied, starting with a different reaction coordinate each time. On the other hand, the B-spline algorithm calculates the coefficients corresponding to the histogram bins and uses the same coefficients throughout the rest of the iteration, generating the forces in all directions at once. Overall, replacing the cubic spline routine with the B-spline reduces the sampling time.

Another important interpretation regarding the results is the decrease in the speedup when the QM method is used. There are other calculations during a Molecular Dynamics simulation affecting the overall runtime. The treatment of the chemical system determines how long an MD simulation takes. Quantum effects are completely neglected in classical MD simulations, where the atoms are simply considered as rigid balls while the bonds are represented as springs. This simplified approach is accurate when it comes to bonds, angles, dihedrals and improper dihedrals of molecules; however, it is inadequate for simulating electronic structure changes such as bond breaking and bond forming. Quantum mechanical calculations are particularly challenging, especially for calculating electron-electron repulsion. For instance, in one-electron models such as Hartree-Fock[124], with dimension N, $O(N^2)$ variables are stored and $O(N^3)$ arithmetic operations are performed. For more accurate computations of the repulsion terms, methods such as Coupled Cluster[125] would require the storage of $O(N^4)$ variables and $O(N^6)$ operations [126]**.** Although AM1 is a semi-empirical method that uses approximations for the repulsion terms via predetermined parameters, it only decreases the computational time to an extent.

As a result, the calculations regarding the electron structure take up a significant amount of computation time in free energy calculations. Even though the speed up stemming from the replacement of cubic spline with B-spline is still effective, its impact on the overall runtime is reduced since the QM calculations are more time-consuming then MM calculations.

6.3 CONCLUDING REMARKS

Although the comparison of the two spline interpolation methods provided significant results using analytical functions, it was essential to demonstrate the effect of splining on numerical

datasets. Among the flat histogram methods, FEARCF was the most ideal method because of its successful implementation of the cubic spline routine. Replacing this routine with the B-spline interpolation gave results similar to those obtained in Chapter 5. The PMF surfaces obtained from the interpolation methods were similar. In addition, the use of B-spline enhanced the speed of the sampling, especially when multiple reaction coordinates were used. The overall speedup effect of the B-spline method differed depending on the treatment of the atoms and the time dominating step in molecular dynamics. Concluding, the implementation of the B-spline interpolation provides a more effective alternative to the cubic spline interpolation and it can be used to enhance the speed of reaction sampling.

6.4 RESOURCES

1. Eadline, D., *High Performance Computing for Dummies*. 2009: Wiley Publishing, Inc.
2. Jain, M.K., *Numerical methods for scientific and engineering computation*. 2003: New Age International.
3. Epperson, J.F., *On the Runge example.* The American Mathematical Monthly, 1987. **94**(4): p. 329-341.
4. Reinsch, C.H., *Smoothing by spline functions.* Numerische mathematik, 1967. **10**(3): p. 177-183.
5. Hiestand, J., *Numerical methods with VBA programming*. 2008: Jones & Bartlett Learning.
6. Zewail, A.H., *Laser femtochemistry.* Science, 1988. **242**(4886): p. 1645-1653.
7. Schramm, V.L., *Enzymatic transition states and transition state analog design*. 1998, Annual Reviews 4139 El Camino Way, PO Box 10139, Palo Alto, CA 94303-0139, USA.
8. Van Duin, A.C., et al., *ReaxFF: a reactive force field for hydrocarbons.* The Journal of Physical Chemistry A, 2001. **105**(41): p. 9396-9409.
9. Fano, G., L.F. Landovitz, and S. Technica, *Mathematical methods of quantum mechanics*. 1971: McGraw-Hill New York;.
10. Wales, D.J. and T.V. Bogdan, *Potential energy and free energy landscapes*. 2006, ACS Publications.
11. Ytreberg, F.M., R.H. Swendsen, and D.M. Zuckerman, *Comparison of free energy methods for molecular systems.* The Journal of chemical physics, 2006. **125**(18): p. 184114.
12. Cossins, B.P., et al., *Assessment of nonequilibrium free energy methods.* The Journal of Physical Chemistry B, 2009. **113**(16): p. 5508-5519.
13. Tropsha, A. and J. Hermans, *Application of free energy simulations to the binding of a transition-state-analogue inhibitor to HTV protease.* Protein Engineering, Design and Selection, 1992. **5**(1): p. 29-33.
14. Hastings, W.K., *Monte Carlo sampling methods using Markov chains and their applications.* 1970.
15. Murthy, K. *Non-Boltzmann Ensembles and Monte Carlo Simulations*. in *Journal of Physics: Conference Series*. 2016. IOP Publishing.
16. Torrie, G.M. and J.P. Valleau, *Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella sampling.* Journal of Computational Physics, 1977. **23**(2): p. 187-199.
17. Huber, T., A.E. Torda, and W.F. Van Gunsteren, *Local elevation: a method for improving the searching properties of molecular dynamics simulation.* Journal of computer-aided molecular design, 1994. **8**(6): p. 695-708.

18. Laio, A. and F.L. Gervasio, *Metadynamics: a method to simulate rare events and reconstruct the free energy in biophysics, chemistry and material science.* Reports on Progress in Physics, 2008. **71**(12): p. 126601.
19. Mezei, M., *Adaptive umbrella sampling: Self-consistent determination of the non-Boltzmann bias.* Journal of Computational Physics, 1987. **68**(1): p. 237-248.
20. Naidoo, K.J., *FEARCF a multidimensional free energy method for investigating conformational landscapes and chemical reaction mechanisms.* Science China Chemistry, 2011. **54**(12): p. 1962-1973.
21. Jacucci, G. and A. Rahman, *Comparing the efficiency of metropolis Monte Carlo and molecular-dynamics methods for configuration space sampling.* Il Nuovo Cimento D, 1984. **4**(4): p. 341-356.
22. Radmer, R.J. and P.A. Kollman, *Free energy calculation methods: a theoretical and empirical comparison of numerical errors and a new method qualitative estimates of free energy changes.* Journal of Computational Chemistry, 1997. **18**(7): p. 902-919.
23. Parker, J.A., R.V. Kenyon, and D.E. Troxel, *Comparison of interpolating methods for image resampling.* IEEE Transactions on medical imaging, 1983. **2**(1): p. 31-39.
24. Fairfield, J., *Segmenting dot patterns by Voronoi diagram concavity.* IEEE transactions on pattern analysis and machine intelligence, 1983(1): p. 104-110.
25. C.s.v. Arnold E. Perham, F.L.P., *Voronoi Diagrams and Spring Rain.* The Mathematics Teacher. **105**(2): p. 126.
26. !!! INVALID CITATION !!! .
27. Sudbø, J., R. Marcelpoil, and A. Reith, *New algorithms based on the Voronoi Diagram applied in a pilot study on normal mucosa and carcinomas.* Analytical Cellular Pathology, 2000. **21**(2): p. 71-86.
28. Courant, R. and F. John, *Introduction to calculus and analysis I*. 2012: Springer Science & Business Media.
29. Neumaier, A., *Computer graphics, linear interpolation, and nonstandard intervals*. 2009.
30. Sauer, T., *Polynomial interpolation of minimal degree.* Numerische Mathematik, 1997. **78**(1): p. 59-85.
31. Jia, Y.-B., *Polynomial Interpolation.* 2017.
32. Gander, W., *Change of basis in polynomial interpolation.* Numerical linear algebra with applications, 2005. **12**(8): p. 769-778.
33. Klinger, A., *The vandermonde matrix.* The American Mathematical Monthly, 1967. **74**(5): p. 571-574.
34. Oruç, H. and G.M. Phillips, *Explicit factorization of the Vandermonde matrix.* Linear Algebra and its Applications, 2000. **315**(1-3): p. 113-123.
35. Stone, M.H., *The generalized Weierstrass approximation theorem.* Mathematics Magazine, 1948. **21**(5): p. 237-254.
36. Lin, H. and L. Sun, *Searching globally optimal parameter sequence for defeating Runge phenomenon by immunity genetic algorithm.* Applied Mathematics and Computation, 2015. **264**: p. 85-98.
37. Edwards, A. and J. Dennis, *Polyfit-A computer program for fitting a specified degree of polynomial to data points.* NRPB-M, 1973. **11**: p. 1-18.
38. Boyd, J.P., *Defeating the Runge phenomenon for equispaced polynomial interpolation via Tikhonov regularization.* Applied Mathematics Letters, 1992. **5**(6): p. 57-59.
39. Tchebychev, P.L., *Théorie des mécanismes connus sous le nom de parallélogrammes*. 1853: Imprimerie de l'Académie impériale des sciences.
40. Remez, E.Y., *Sur la détermination des polynômes d'approximation de degré donnée.* Comm. Soc. Math. Kharkov, 1934. **10**(4163): p. 196.
41. Segner, D., *The shape of the human face recorded by use of contour photography and spline function interpolation.* The European Journal of Orthodontics, 1986. **8**(2): p. 112-117.

42. De Boor, C., et al., *A practical guide to splines*. Vol. 27. 1978: springer-verlag New York.
43. Unser, M., *Splines: A perfect fit for signal and image processing.* IEEE Signal processing magazine, 1999. **16**(ARTICLE): p. 22–38.
44. Foley, T.A. and G.M. Nielson, *Knot selection for parametric spline interpolation*, in *Mathematical methods in computer aided geometric design*. 1989, Elsevier. p. 261-CP4.
45. Siauw, T. and A. Bayen, *An introduction to MATLAB® programming and numerical methods for engineers*. 2014: Academic Press.
46. Press, W.H., et al., *Numerical recipes in Fortran 90.* The art of scientific computing,(Cambridge, 1996), 1992.
47. Schmeisser, G., *A real symmetric tridiagonal matrix with a given characteristic polynomial.* Linear algebra and its applications, 1993. **193**: p. 11-18.
48. Rajamani, R., K.J. Naidoo, and J. Gao, *Implementation of an adaptive umbrella sampling method for the calculation of multidimensional potential of mean force of chemical reactions in solution.* Journal of computational chemistry, 2003. **24**(14): p. 1775-1781.
49. Ruijters, D., B.M. ter Haar Romeny, and P. Suetens, *Efficient GPU-based texture interpolation using uniform B-splines.* Journal of Graphics Tools, 2008. **13**(4): p. 61-69.
50. De Boor, C., *On calculating with B-splines.* Journal of Approximation theory, 1972. **6**(1): p. 50-62.
51. Habermann, C. and F. Kindermann, *Multidimensional spline interpolation: Theory and applications.* Computational Economics, 2007. **30**(2): p. 153-169.
52. Lu, Y., et al., *A B-spline curve extension algorithm.* Science China Information Sciences, 2016. **59**(3): p. 32103.
53. Newbery, A. and T.S. Garrett, *Interpolation with minimized curvature.* Computers & Mathematics with Applications, 1991. **22**(1): p. 37-43.
54. Shukla, D., et al., *Activation pathway of Src kinase reveals intermediate states as targets for drug design.* Nature communications, 2014. **5**: p. 3397.
55. Sheppard, D., R. Terrell, and G. Henkelman, *Optimization methods for finding minimum energy paths.* The Journal of chemical physics, 2008. **128**(13): p. 134106.
56. Helmholtz, H., *On the thermodynamics of chemical processes.* Physical Memoirs Selected and Translated from Foreign Sources, 1882. **1**: p. 43-97.
57. Gibbs, J.W., *A Method of Geometrical Representation of the Thermodynamic Properties by Means of Surfaces.* Transactions of Connecticut Academy of Arts and Sciences, 1873: p. 382-404.
58. Alder, B.J. and T.E. Wainwright, *Studies in molecular dynamics. I. General method.* The Journal of Chemical Physics, 1959. **31**(2): p. 459-466.
59. Ott, J.B. and J. Boerio-Goates, *Chemical Thermodynamics: Advanced Applications: Advanced Applications*. 2000: Elsevier.
60. Chatfield, D., *Christopher J. Cramer: Essentials of Computational Chemistry: Theories and Models.* Theoretical Chemistry Accounts: Theory, Computation, and Modeling (Theoretica Chimica Acta), 2002. **108**(6): p. 367-368.
61. Paquet, E. and H.L. Viktor, *Molecular dynamics, monte carlo simulations, and langevin dynamics: a computational review.* Biomed Res Int, 2015. **2015**: p. 183918.
62. Zwanzig, R.W., *High-temperature equation of state by a perturbation method. I. Nonpolar gases.* The Journal of Chemical Physics, 1954. **22**(8): p. 1420-1426.
63. Pohorille, A., C. Jarzynski, and C. Chipot, *Good practices in free-energy calculations.* The Journal of Physical Chemistry B, 2010. **114**(32): p. 10235-10253.
64. Bhati, A.P., et al., *Rapid, accurate, precise, and reliable relative free energy prediction using ensemble based thermodynamic integration.* Journal of chemical theory and computation, 2016. **13**(1): p. 210-222.

65. Meng, Y., D. Sabri Dashti, and A.E. Roitberg, *Computing alchemical free energy differences with Hamiltonian replica exchange molecular dynamics (H-REMD) simulations.* Journal of chemical theory and computation, 2011. **7**(9): p. 2721-2727.
66. Apte, P.A. and I. Kusaka, *Direct calculation of solid-liquid coexistence points of a binary mixture by thermodynamic integration.* The Journal of chemical physics, 2005. **123**(19): p. 194503.
67. Kirkwood, J.G., *Statistical mechanics of fluid mixtures.* The Journal of Chemical Physics, 1935. **3**(5): p. 300-313.
68. van Gunsteren, W.F., X. Daura, and A.E. Mark, *Computation of free energy.* Helvetica Chimica Acta, 2002. **85**(10): p. 3113-3129.
69. Giese, T.J. and D.M. York, *A GPU-accelerated parameter interpolation thermodynamic integration free energy method.* Journal of chemical theory and computation, 2018. **14**(3): p. 1564-1582.
70. Shyu, C. and F.M. Ytreberg, *Use of polynomial interpolation to reduce bias and uncertainty of free energy estimates via thermodynamic integration.* arXiv preprint arXiv:0809.0882, 2008.
71. Quaytman, S.L. and S.D. Schwartz, *Reaction coordinate of an enzymatic reaction revealed by transition path sampling.* Proceedings of the National Academy of Sciences, 2007. **104**(30): p. 12253-12258.
72. Barnett, C.B. and K.J. Naidoo, *Free Energies from Adaptive Reaction Coordinate Forces (FEARCF): an application to ring puckering.* Molecular Physics, 2009. **107**(8-12): p. 1243-1250.
73. Beveridge, D.L. and F. DiCapua, *Free energy via molecular simulation: applications to chemical and biomolecular systems.* Annual review of biophysics and biophysical chemistry, 1989. **18**(1): p. 431-492.
74. Landau, D., S.-H. Tsai, and M. Exler, *A new approach to Monte Carlo simulations in statistical physics: Wang-Landau sampling.* American Journal of Physics, 2004. **72**(10): p. 1294-1302.
75. Kästner, J., *Umbrella sampling.* Wiley Interdisciplinary Reviews: Computational Molecular Science, 2011. **1**(6): p. 932-942.
76. Barducci, A., M. Bonomi, and M. Parrinello, *Metadynamics.* Wiley Interdisciplinary Reviews: Computational Molecular Science, 2011. **1**(5): p. 826-843.
77. Ferrenberg, A.M. and R.H. Swendsen, *New Monte Carlo technique for studying phase transitions.* Physical review letters, 1988. **61**(23): p. 2635.
78. Kumar, S., et al., *The weighted histogram analysis method for free-energy calculations on biomolecules. I. The method.* Journal of computational chemistry, 1992. **13**(8): p. 1011-1021.
79. Reynal, S. and H.-T. Diep, *Fast flat-histogram method for generalized spin models.* Physical Review E, 2005. **72**(5): p. 056710.
80. Affentranger, R., I. Tavernelli, and E.E. Di Iorio, *A novel Hamiltonian replica exchange MD protocol to enhance protein conformational space sampling.* Journal of Chemical Theory and Computation, 2006. **2**(2): p. 217-228.
81. Zeller, F. and M. Zacharias, *Adaptive biasing combined with Hamiltonian replica exchange to improve umbrella sampling free energy simulations.* Journal of chemical theory and computation, 2014. **10**(2): p. 703-710.
82. Souaille, M. and B. Roux, *Extension to the weighted histogram analysis method: combining umbrella sampling with free energy calculations.* Computer physics communications, 2001. **135**(1): p. 40-57.
83. Tsai, S.-H., F. Wang, and D. Landau, *Critical endpoint behavior in an asymmetric Ising model: Application of Wang-Landau sampling to calculate the density of states.* Physical Review E, 2007. **75**(6): p. 061108.
84. Belardinelli, R. and V. Pereyra, *Wang-Landau algorithm: A theoretical analysis of the saturation of the error.* The Journal of chemical physics, 2007. **127**(18): p. 184105.

85. Belardinelli, R., S. Manzi, and V. Pereyra, *Analysis of the convergence of the 1⁄t and Wang-Landau algorithms in the calculation of multidimensional integrals.* Physical Review E, 2008. **78**(6): p. 067701.
86. Beutler, T.C. and W.F. van Gunsteren, *Umbrella sampling along linear combinations of generalized coordinates. Theory and application to a glycine dipeptide.* Chemical physics letters, 1995. **237**(3-4): p. 308-316.
87. Hansen, H.S., X. Daura, and P.H. Hünenberger, *Enhanced conformational sampling in molecular dynamics simulations of solvated peptides: Fragment-based local elevation umbrella sampling.* Journal of chemical theory and computation, 2010. **6**(9): p. 2598-2621.
88. Abrams, C. and G. Bussi, *Enhanced sampling in molecular dynamics using metadynamics, replica-exchange, and temperature-acceleration.* Entropy, 2014. **16**(1): p. 163-199.
89. Barducci, A., G. Bussi, and M. Parrinello, *Well-tempered metadynamics: a smoothly converging and tunable free-energy method.* Physical review letters, 2008. **100**(2): p. 020603.
90. Salzburg, Z., et al., *Application of the Monte Carlo method to the lattice-gas model.* J. Chem. Phys, 1959. **30**: p. 65.
91. Chesnut, D.A. and Z.W. Salsburg, *Monte Carlo procedure for statistical mechanical calculations in a grand canonical ensemble of lattice systems.* The Journal of Chemical Physics, 1963. **38**(12): p. 2861-2875.
92. Ferrenberg, A.M., D. Landau, and R.H. Swendsen, *Statistical errors in histogram reweighting.* Physical Review E, 1995. **51**(5): p. 5092.
93. Pohorille, A. and C. Chipot, *Free energy calculations: theory and applications in chemistry and biology*. 2007: Springer.
94. Shirts, M.R., *Reweighting from the mixture distribution as a better way to describe the Multistate Bennett Acceptance Ratio.* arXiv preprint arXiv:1704.00891, 2017.
95. Tan, Z., et al., *Theory of binless multi-state free energy estimation with applications to protein-ligand binding.* The Journal of chemical physics, 2012. **136**(14): p. 04B608.
96. Naidoo, K.J. and J. Brady, *Calculation of the Ramachandran potential of mean force for a disaccharide in aqueous solution.* Journal of the American Chemical Society, 1999. **121**(10): p. 2244-2252.
97. Ireland, R.E. and R.H. Mueller, *Claisen rearrangement of allyl esters.* Journal of the American Chemical Society, 1972. **94**(16): p. 5897-5898.
98. Strümpfer, J. and K.J. Naidoo, *Computing free energy hypersurfaces for anisotropic intermolecular associations.* Journal of computational chemistry, 2010. **31**(2): p. 308-316.
99. Moler, C., *Matrix computation on distributed memory multiprocessors.* Hypercube Multiprocessors, 1986. **86**(181-195): p. 31.
100. Stocker, U., D. Juchli, and W.F. van Gunsteren, *Increasing the time step and efficiency of molecular dynamics simulations: optimal solutions for equilibrium simulations or structure refinement of large biomolecules.* Molecular Simulation, 2003. **29**(2): p. 123-138.
101. Vaidehi, N. and A. Jain, *Internal coordinate molecular dynamics: A foundation for multiscale dynamics.* The Journal of Physical Chemistry B, 2015. **119**(4): p. 1233-1242.
102. Pearson, K., *LIII. On lines and planes of closest fit to systems of points in space.* The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 1901. **2**(11): p. 559-572.
103. Sutto, L., S. Marsili, and F.L. Gervasio, *New advances in metadynamics.* Wiley Interdisciplinary Reviews: Computational Molecular Science, 2012. **2**(5): p. 771-779.
104. Sicard, F. and P. Senet, *Reconstructing the free-energy landscape of Met-enkephalin using dihedral principal component analysis and well-tempered metadynamics.* The Journal of chemical physics, 2013. **138**(23): p. 06B610_1.
105. Mu, Y., P.H. Nguyen, and G. Stock, *Energy landscape of a small peptide revealed by dihedral angle principal component analysis.* Proteins: Structure, Function, and Bioinformatics, 2005. **58**(1): p. 45-52.

106. Altis, A., et al., *Dihedral angle principal component analysis of molecular dynamics simulations.* The Journal of chemical physics, 2007. **126**(24): p. 244111.
107. Rogers, I.L., *Measuring the effects of reaction coordinate and electronic treatments in the QM/MM reaction dynamics of Trypanosoma cruzi trans-sialidase*. 2016, University of Cape Town.
108. Mathews, J.H. and K.D. Fink, *Numerical methods using MATLAB*. Vol. 3. 2004: Pearson Prentice Hall Upper Saddle River, NJ.
109. Adams, K., *Smooth interpolation of zero curves.* Algo Research Quarterly, 2001. **4**(1/2): p. 11-22.
110. Gilsinn, D.E., M.A. McClain, and C. Witzgall, *Non-Oscillatory Splines on Irregular Data.*
111. Zhang, Z., J. Tomlinson, and C. Martin, *Splines and linear control theory.* Acta Applicandae Mathematica, 1997. **49**(1): p. 1-34.
112. Gang, X. and W. Guo-Zhao, *Extended cubic uniform B-spline and α-B-spline.* Acta Automatica Sinica, 2008. **34**(8): p. 980-984.
113. Zhang, Z. and C.F. Martin, *Convergence and Gibbs' phenomenon in cubic spline interpolation of discontinuous functions.* Journal of Computational and Applied mathematics, 1997. **87**(2): p. 359-371.
114. Richards, F., *A Gibbs phenomenon for spline functions.* Journal of approximation theory, 1991. **66**(3): p. 334-351.
115. Schoenberg, I., *Cardinal interpolation and spline functions.* Journal of Approximation theory, 1969. **2**(2): p. 167-206.
116. Jamrozik, J., J. Bohmanova, and L. Schaeffer, *Selection of locations of knots for linear splines in random regression test-day models.* Journal of Animal Breeding and Genetics, 2010. **127**(2): p. 87-92.
117. Wold, S., *Spline functions in data analysis.* Technometrics, 1974. **16**(1): p. 1-11.
118. Openshaw, S. and I. Turton, *High Performance Computing and the Art of Parallel Programming: An Introduction for Geographers, Social Scientists and Engineers*. 2005: Routledge.
119. Joye, K., *Carnivorous plants.* 1989.
120. Jo, S., et al., *CHARMM-GUI: a web-based graphical user interface for CHARMM.* Journal of computational chemistry, 2008. **29**(11): p. 1859-1865.
121. Grubmüller, H., et al., *Generalized Verlet algorithm for efficient molecular dynamics simulations with long-range interactions.* Molecular Simulation, 1991. **6**(1-3): p. 121-142.
122. Dewar, M.J., et al., *Development and use of quantum mechanical molecular models. 76. AM1: a new general purpose quantum mechanical molecular model.* Journal of the American Chemical Society, 1985. **107**(13): p. 3902-3909.
123. Humphrey, W., A. Dalke, and K. Schulten, *VMD: visual molecular dynamics.* Journal of molecular graphics, 1996. **14**(1): p. 33-38.
124. Hartree, D.R. *The wave mechanics of an atom with a non-Coulomb central field. Part I. Theory and methods*. in *Mathematical Proceedings of the Cambridge Philosophical Society*. 1928. Cambridge University Press.
125. Monkhorst, H.J., *Calculation of properties with the coupled-cluster method.* International Journal of Quantum Chemistry, 1977. **12**(S11): p. 421-432.
126. De Jong, W.A., et al., *Utilizing high performance computing for chemistry: parallel computational chemistry.* Physical Chemistry Chemical Physics, 2010. **12**(26): p. 6896-6920.

CHAPTER 7: CONCLUSION

The dissertation is an interdisciplinary study that combines aspects of computational science and chemistry. It demonstrates the comparison between two numerical methods on an application of chemistry under the catalysis of High Performance Computing.

Numerical methods are valuable in computational science where the mathematical model of a system cannot be solved analytically. Interpolation is a commonly used numerical method, as it aids the prediction of the value at any point along a discretized dataset. The efficiency of an interpolation varies according to the degree of the interpolant and the continuity level it provides. Low degree interpolants such as linear polynomials lack the differentiability, while high degree polynomials deviate from the data values at critical regions such as the endpoints. Cubic spline interpolation was proven to compensate for the deficiencies of the aforementioned methods, as it is twice differentiable and avoidant of oscillatory errors.

Despite the advantages of cubic spline interpolation, it experiences a significant slowdown in a multidimensional numeric data due to the large number of iterations in its algorithm.

Alternatively, the B-spline interpolation can represent the a cubic spline interpolation as the linear combination of basis functions and their coefficients. It has a reduced amount of iterations because the calculations of the coefficient and the linear combination are segregated, yet it yields equivalent results. Therefore, it can replace the cubic splining for faster performance with an equally-accurate approximation.

Free energy methods are ideal to substantiate this hypothesis as they frequently make use of numerical methods. As reviewed in the dissertation, free energy calculations take advantage of interpolation for various purposes. The emphasis of the study is on flat histogram methods that generate free energy diagrams of chemical reactions. If interpolated, a flat histogram method can be used to extract valuable interpolation about the critical points of the reaction.

The method Free Energies from Adaptive Reaction Coordinate Forces developed by Prof. Kevin J. Naidoo uses cubic spline interpolation to derive biasing forces. The disadvantage of the cubic splining in higher dimensions holds true for FEARCF since sampling with multiple reaction coordinates slows down the process. Considering this, the study aimed to test the impact of B-splining to find out whether it was eligible for replacement.

The first part of the study compared the two interpolation methods to solve for a set of analytic models with known functions that featured various gradients and jagged regions. For all the functions, the methods yielded equal approximation with identical relative error. The maximum – which occurred at the endpoints and the inflection points – was reduced to $1.5x10^{-3}$ % for the interpolated values and 2% for the derivatives with the increasing number of interpolating knots. However, overfitting occurred at the non-critical regions with an excessive number of knots. Besides the accuracy test, the speed of interpolation was also compared. The cubic spline method experienced a significant slowdown with the increased number of knots, while the duration of the B-spline remained the same. Furthermore, the impact of the speedup was elevated in multiple dimensions.

In the second part of the study, the comparison was translated into the free energy calculations involving FEARCF, which sampled the proton exchange between ammonia and ammonium. The reaction was first simulated using only classical molecular dynamics, followed by the simulation with quantum mechanical methods. The reason for running two different types of simulations was to see the effect of splining on the overall reaction dynamics in different chemical problems.

For all the simulations, the two PMF surfaces obtained from the two interpolation methods were similar. In addition, the B-spline method was 20 times faster in two dimensions and 1800 times faster in three dimensions during classical molecular dynamics. The speedup was reduced in quantum mechanical calculations as the duration of the splining was less contributive due to the time-dominating calculation of electron repulsion terms.

To conclude, the study provides an extensive comparison between two spline interpolation methods in terms of accuracy and speed, concluding that the B-spline interpolation is a more feasible alternative to the cubic spline interpolation.