BIRZEIT UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

Convergence of Conformational Sampling For BPTI in a 30ns Molecular Dynamics Simulation

by

Waseem Ahmad Mohmmad AL-Hausani

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENTIFIC COMPUTING

Approved, Thesis
Committee:

Dr. Wa'el Karain

Dr. Mazen Hamad

Dr. Wasfi AL-Kafri

Birzeit, Palestine

November, 2007

ABSTRACT

Studying protein intramolecular motion gives an important knowledge about its function. In this work, the convergence of the conformational sampling space was performed on a 30ns long molecular dynamics simulation of Bovine Pancreatic Trypsin Inhibitor (BPTI). The overlap between the principle component vectors for different time windows of the simulation time were calculated using two methods. The first method measures the total overlap between covariance matrices of different time segments. It shows a lack of convergence with steady values throughout the simulation. The second method examines the subspace overlap between eigenvectors of different time windows of simulation. This method shows high convergence between 4ns long time fragments. The most of atomic fluctuations of BPTI C_α atoms can be seen in 4ns time intervals. The B-factor and RMSD were calculated in water to check the validity of the simulation results. They give good agreement with the experiment.

Acknowledgments

First of all, I would like to express my gratitude to my supervisor Dr.Wa'el Karain for all his support, guidance, constructive criticism and ideas as well as encouragement to achieve the success of this work. He was always very approachable.

I am grateful to Dr. Wasfi Alkfri and Dr. Mazen Hamad for their guidance, support and help in writing my thesis.

I would also like to thank Dr.Hassan Shebli, and all physics department members at Birzeit University for their guidance, help, and support.

I also owe my gratitude to my parents for their encouragement and support, and to my brothers for their love, engorgement and inspiration.

Contents

Abstract ii

Acknowledgments iii

List of Figures viii

List of Tables x

1 Introduction 1

2 Theoretical Background and Used Techniques 4

2.1 Studying Proteins motion 4

2.2 Aims of Computer Simulations 6

2.3 Theoretical Techniques used in Studying Protein Dynamics 7

2.4 Molecular Dynamics and Classical Laws of Motion 9

2.5 Stochastic Dynamics 13

2.6 Numerical Solution of the Molecular Dynamics Equations of Motion 14

2.6.1 Original Verlet Algorithm 15

2.6.2 The Verlet Leapfrog Algorithm 16

2.6.3 The Verlet Velocity Algorithm 17

2.6.4 The Beeman's Algorithm 17

2.7 Statistical Mechanics 18

2.8 Molecular Modeling 21

2.8.1 Molecular Mechanics 21

2.8.2 Empirical Potential energy functions 22

2.9 Analysis Techniques 27

2.9.1 Root Mean Square Deviation 27

2.9.2 Essential Dynamics 28

2.9.3 Principal Component Analysis Technique 29

2.9.4 The Essential Dynamics Method for Protein Dynamics Analysis 34

3 Computing Tools and Methodology 40

3.1 Molecular Dynamics Tools Used 40

3.1.1 NAMD Molecular Dynamics Software 40

3.1.2 Visual Molecular Dynamics (VMD) Software 41

3.1.3 Catdcd Tool 42

3.1.4 FlipDCD Tool 42

3.1.5 MATLAB: High-Performance Computation Software 43

3.2 Molecular Dynamics Simulation 44

3.2.1 Preparation of the BPTI Molecule 45

3.2.2 Solvation and Neutralization of BPTI in Water . . . 48

3.2.3 Minimization Phase 50

3.2.4 Heating Phase 50

3.2.5 Equilibration Phase 52

3.2.6 Production Phase 52

3.2.7 Analysis of MD Simulation Trajectories 53

4 Results and Analysis 59

4.1 B-Factor 59

4.2 Root Mean Square Deviation (RMSD) 61

4.3 Convergence of Conformational Sampling 63

5 Discussion of Results and Conclusion 72

A Files Formats 76

A.1 The PDB file format 76

A.2 The PSF file format 77

A.3 The DCD file format 78

B Configuration File 82

C M-Files Used to Analysis Data trajectory 87

C.1 Calculation of the Covariance Matrix 87

C.2 The First Method to Calculate the Overlap 88

C.3 The Second Method to Calculate the Overlap 89

C.4 M-file Used to Calculate the B-factor 90

References **91**

List of Figures

2.1 An amino acid structure. 4

2.2 Simulations as a bridge between theory and experiment. . . 7

2.3 A protein molecule. Dark dots represent backbone atoms. Springs represent the bonds between the atoms. 22

2.4 The backbone dihedral angles Φ, Ψ and ω[16]. 25

3.1 BPTI Molecule after ionization in a water box (solution). . . 51

4.1 Simulated B-factor for the C_α atoms of BPTI compared with the B-factor obtained from crystallographic structure refinement based on X-ray data. 60

4.2 Root mean square deviation (RMSD) of the BPTI configurations with respect to the initial configuration, as a function of simulation time. 62

4.3 Calculated covariance overlap using equation2.64. (a) The overlap between different simulation fragments with 100ps length for each time fragment along the 30ns simulation. (b) Covariance overlap between 500ps time fragments. 66

4.4 Calculated covariance overlap using equation2.64. (a) The overlap between different simulation fragments with 1ns length for each time fragment along the 30ns simulation. (b) Covariance overlap between 4ns time fragments. 67

4.5 Calculated covariance overlap using equation2.64 between 10ns time windows along the 30ns simulation. 68

4.6 Calculated subspace overlap using equation2.59. (a) The overlap between 30 eigenvectors 100ps long fragments along the 30ns simulation. (b)Overlap between 30 eigenvectors from 500ps long fragments along the simulation time. . . . 69

4.7 Calculated overlap using equation2.59. (a) The overlap between 30 eigenvectors from 1ns long fragments. (b) Overlap between 30 eigenvectors of 4ns long fragments. 70

4.8 Calculated overlap using equation2.59. (a) The overlap between 30 eigenvectors of 7ns long fragments. (b) Overlap between 30 eigenvectors of each 10ns time fragments along the simulation. 71

A.1 The PDB file format example 77

A.2 The PSF file format example for BPTI 80

A.3 The DCD file format example 81

List of Tables

1.1 The characteristic times for common molecular events. . . . 2

A.1 The PDB format for coordinate records 79

Chapter 1
Introduction

The aim of this work is to investigate how long a simulation should be to obtain good insight about protein (BPTI) global motions. To do so the simulation running for a 30ns was performed. The Essential Dynamics or PCA method was used to analysis the trajectory resulted from the simulation to find the overlap between eigenvectors of protein subsets along the motion as an indicator of the convergence of protein sampling along the simulation.

The aim of calculating the overlap between protein subspaces is to find the time necessary to perform the dynamics of BPTI. This time can range from picoseconds scale to nanoseconds time scale. The table 1.1[1] lists the characteristic times for common molecular events.

B. Hess[2] used the PCA method for studying the convergence of sampling in protein (Iysozyme and HPr) simulations. He found that 14ns is not enough time to obtain the convergence of sampling, and show that the PCA and covariance calculations is a good indicator for bad sampling[2]. Faraldo-Gómez et. al.[3] also used the same method for studying the convergence of the sampling of conformational space accomplished in molecular

Event	Time
Bond stretch	1 to 20 fs
Elastic domain modes	100 fs to several ps
Water reorientation	4ps
Inter-domain bending	10ps to 100ns
Globular protein tumbling	1ns to 10ns
Aromatic ring flipping	100μs to several seconds
Allosteric shifts	2μs to several seconds
Local denaturation	1ms to several seconds

Table 1.1 The characteristic times for common molecular events.

dynamics simulations of membrane proteins along 10ns.

This work follow the same method (PCA) in analysis the trajectory produced by simulation of BPTI along 30ns time scale. The overlap between eigenvectors for different size time-windows along the simulation time show high convergence at 4ns time-windows size. This consider as a new insight about the BPTI motion convergence time by molecular dynamics simulation.

Outline of this thesis

The second chapter of this thesis is concerned with the theoretical background about Molecular Dynamics technique which used in the simulation.

Also it contains a brief introduction about Essential Dynamics or Principle Component Analysis technique and overlap method.

The third chapter describe the used tools in running the simulation and analysis the results like NAMD software, VMD package, and MATLAB. The procedure used in running the simulation and in data analysis are included in the late sections of this chapter.

Chapter four is the main chapter of this thesis because it is contains all the results obtained by the simulation and its analysis.

Finally, chapter five presents the discussion of results and the conclusion of my work and the future work which should do to continue the work in protein (BPTI) dynamics.

Chapter 2
Theoretical Background and Used Techniques

2.1 Studying Proteins motion

Proteins are complex macromolecules made up of amino acids (Fig.2.1), consisting of thousands of atoms[4]. All amino acids have a central atom (carbon atom) which is called C_α. Twenty different side-chains can be attached to the C_α atom giving twenty types of amino acids[5]. Proteins are essential parts of all living organisms because all processes within living cells depend on protein activity, e.g., transport, storage, signaling, and defence.[6, 5].

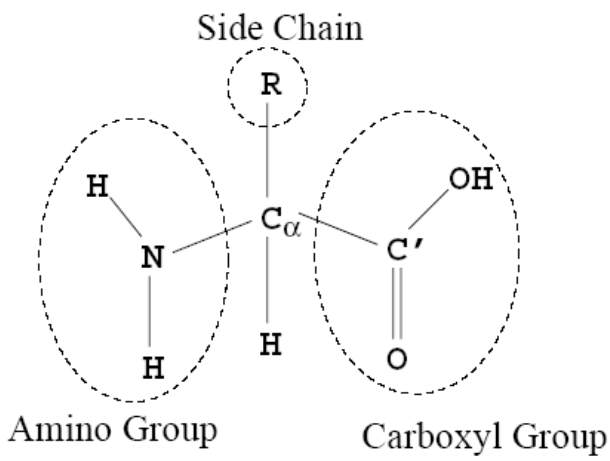

Figure 2.1 An amino acid structure.

Understanding protein biological functions requires studying its atomic motions. Scientific evidence suggests that biological processes, such as enzyme and virus activity, depends on atomic correlated motions[6, 7]. These motions range from tenths of Angstroms on the femtosecond time scale to several seconds[2].

Bovine pancreatic trypsin inhibitor (BPTI) is widely used in studying intramolecular motions of protein atoms. It is one of the smallest and simplest proteins and contains 57 amino acid residues[8]. It is also, one of the most extensively studied proteins and has been investigated structurally using x-ray crystallography and Nuclear Magnetic Resonance (NMR) spectroscopy[9].

Both experimental techniques (X-ray crystallography, Nuclear Magnetic Resonance (NMR) and Neutron scattering) and theoretical techniques (e.g. Monte Carlo (MC) and Molecular Dynamics (MD)) are used to study structure , dynamics and thermodynamics of biological molecules (e.g. protein) and their complexes[10]. Theoretical techniques provides a complete knowledge about conformational changes of proteins at the nanosecond time scale. This is very useful considering that currently no experimental techniques allows the study of protein dynamics at the nanosecond time scale[6].

Current processing speeds of computers provide a realistic option to

study protein molecules to obtain dynamics information in the picosecond to nanosecond time scale.

2.2 Aims of Computer Simulations

Simulations play an important role in studying natural phenomena especially in a conceptually difficult field such as physics. As to the role of molecular dynamics, it can bring to life the entire invisible universe of the atom[11].

Computer simulations aim to understand structure and macroscopic properties of molecules and the microscopic interactions between atoms in molecules[12]. Molecular dynamics is one of the most powerful computer techniques that offer new information of dynamical properties of molecules where conventional experiments cannot find these properties of molecules and atoms. Computer simulations act as a bridge between theory and experiment (Fig.2.2). They may be used to test a theory or to test the model by comparing simulation results with experimental results. Moreover, computer simulations can be used to carry out studies on intramolecular actions that are impossible to study experimentally[12].

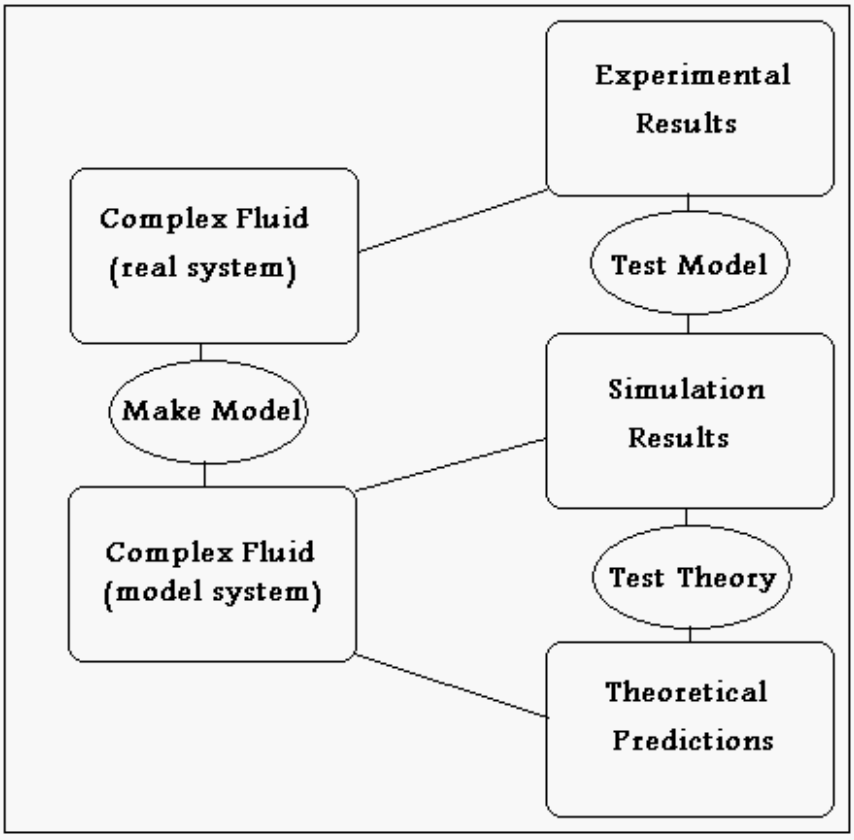

Figure 2.2 Simulations as a bridge between theory and experiment.

2.3 Theoretical Techniques used in Studying Protein Dynamics

Theoretical methods are used for generating and evaluating representative molecular conformations for peptides and small proteins, based on molecular mechanics energy functions[13]. Monte Carlo (MC), Molecular Dynamics (MD) and Lengeven Dynamics (LD) are considered as a distinct methods seeking for minimum energy conformer to generate a conforma-

tional states[13].

MC Method is a stochastic technique based on the use of random numbers and probability statistics, used in many fields of science (biochemistry, nuclear physics, biophysics,...etc) for studying complex and large systems with hundreds or thousands of atoms (e.g. proteins), which can be sampled in a number of random configurations that describe the whole system [14]. This means that MC for proteins simulation samples conformation space without a realistic dynamics trajectory (time-independent quantities)[15].

MD is a computational Method for simulating the motion of a system of many particles[16].MD simulation is a realistic method describe the collective motion of the molecule in terms of Newton's equations(Hamiltonian, Lagrangian, or direct expression) of motion[13],it is also allow to simulate the behavior of a molecular system under different conditions of temperature, pressure, and other parameters[17].

LD is a stochastic method used explicitly for the solvent model by replacing Newton's equations with the Lengeven equation to approximate the effects of neglected degrees of freedom for the solvent[13, 15].

Today, MD is a popular technique used widely to study dynamics of solvated proteins, protein-DNA complexes, lipid systems and folding of small proteins[10].

2.4 Molecular Dynamics and Classical Laws of Motion

Molecular Dynamics simulations are the main theoretical techniques used to study protein dynamics. Increased computer power and improving algorithms have extended accessible system sizes and time scales. simulations of large systems like protein surrounded by 3000 water molecules are possible[4, 18]. The computer power has increased by 10^5 since the first molecular simulation of a protein.

In a MD simulation, classical mechanics is used to determine the interaction forces between atoms of the molecule. These forces of interactions are calculated by Newton's second law which state that the time rate of change of the velocity (acceleration) is directly proportional to the force and inversely proportional to the mass of the particle[19].

$$F_i = m_i a_i \tag{2.1}$$

where F_i is the force exerted on particle (atom) i, m_i is the mass of particle i and a_i is the acceleration of particle i. Atomic positions along the simulation time x_i are calculated from equation(2.2):

$$a_i = \frac{d^2 x_i}{dt^2} \tag{2.2}$$

The force can also be expressed as the gradient of the potential energy,

$$F_i = -\nabla_i V \tag{2.3}$$

by combining the two equations (2.2) and (2.3) we get:

$$-\frac{\partial V}{\partial x_i} = m_i a_i = m_i \frac{\partial^2 x_i}{\partial t^2} \tag{2.4}$$

where V is the potential energy of the system. Newtons equation of motion now relate the derivative of the potential energy to the changes in position x_i as a function of time.

Determining the force acting on each atom of the molecule makes finding the velocity, acceleration and position (micro states) of this atom easy along the simulation time. This means that the state of the system or the macroscopic properties of the molecule can be calculated at any time by statistical ensemble averages as explained in section (2.7) . An example of one-dimensional motion can be shown as the following:

$$F = m \cdot \frac{dv}{dt} = m \cdot \frac{d^2x}{dt^2} \tag{2.5}$$

takeing a special case where the acceleration is constant

$$a = \frac{dv}{dt} \tag{2.6}$$

the velocity can be found by solving the simple differential equation(2.6)

and the result gives

$$v = at + v_0 \tag{2.7}$$

also since

$$v = \frac{dx}{dt} \tag{2.8}$$

the position x can found by solving the differential equation (2.8) to get

$$x = \int v dt \tag{2.9}$$

By subsituting the velovity from equation (2.7) in equation (2.9), we can get a new equation which gives the position x as a function of acceleration a, initial position x_0, and initial velocity v_0 as shown by equation (2.10):

$$x = \frac{1}{2}at^2 + v_0 t + x_0 \tag{2.10}$$

This equation (2.10) can calculate the trajectory if the initial values of positions and velocities of the protein molecule atoms and acceleration are known.

The experimental techniques which deal with studying protein structure such as x-ray diffraction and NMR spectroscopy, offer the needed starting positions of the protein molecule atoms. An initial distribution of velocities of atoms is obtained by guessing random velocities from a Maxwellian distribution at a given temperature.

The total energy of the molecule is a function of the atomic positions along the trajectory:

$$E = K + U = \frac{1}{2}\sum_{i=1}^{N} m_i \parallel r_i \parallel^2 + U(r_1, r_2, ...r_n) \tag{2.11}$$

where U is the potential energy function, K is the kinetic energy, and r the position coordinates (x,y,z) of the atom i.

Another way to determine the interaction forces between the atoms in the molecule is the Hamiltonian formulation. The Hamiltonian equation is defined in terms of particle positions r_i and momenta p_i[20].

$$H = \sum_{i=1}^{3N} \frac{p_i^2}{2m_i} + U(q_1, q_2, ...q_{3N}) \tag{2.12}$$

where q is the positions of the N particles. Hamiltonian equations of motion are:

$$\dot{q}_i = \frac{\partial H}{\partial p_i} \tag{2.13}$$

$$\dot{p}_i = -\frac{\partial H}{\partial q_i} \tag{2.14}$$

Hamiltonian formulation is more convenient than Newton's equation because it is straightforward to prove energy conservation, but we use New-

ton's equation of motion in protein motion problem because it is direct in calculating the the trajectory of motion.

2.5 Stochastic Dynamics

Stochastic dynamics on the protein simulation produced by a collision of the protein molecule atoms with the solvent molecule atoms. This dynamics represented by Langevin equation(2.15) in which two force terms are added to Newton's second law to approximate the effects of neglected degrees of freedom[13, 15]. Langevin dynamics can search conformations better than Newtonian molecular dynamics[15].

$$m_i \ddot{\mathbf{r}}_i(t) = \mathbf{F}_i(t) - \beta_i \mathbf{v}_i(t) + \mathbf{f}_i(t) \tag{2.15}$$

where m is the mass of the particle (atom), $\mathbf{F}_i$ is the systematic force exerted exerted on on atom i by other atoms, β is the friction constant, and $\mathbf{f}_i$ is the random force which measure the collisional effect of the solvent on the solute.

The random force $\mathbf{f}_i(t)$ in protein simulations is completely uncorrelated at different times:

$$\langle \mathbf{f}_i(t) \cdot \mathbf{f}_i(\acute{t}) \rangle = 6 k_B T \beta_i \delta(t - \acute{t}) \tag{2.16}$$

where k_B is the Boltzmann's constant, T is the temperature, and $\delta(t - \acute{t})$ is the Dirac delta function.

2.6 Numerical Solution of the Molecular Dynamics Equations of Motion

For N atoms of a protein molecule, Newton's second law (equation 2.4) represents $3N$ second-order, ordinary differential equations of motion where the potential energy is the function of atomic positions of atoms in the molecule. These equations of motion have no analytical solutions, and so numerical methods are needed to solve them[21].

Finite-difference methods replace differential (such as dx and dt) with finite differences (Δx and Δt), which means to replace differential equations with finite-difference equations over a small finite time Δt. Several finite-difference methods originate from the truncated Taylor expansion[21].

$$r(t + \Delta t) = r(t) + \dot{r}(t) + \frac{1}{2}\ddot{r}\Delta t^2 + ... \tag{2.17}$$

$$v(t + \Delta t) = v(t) + \dot{v}(t) + \frac{1}{2}\ddot{v}\Delta t^2 + ... \tag{2.18}$$

$$a(t + \Delta t) = a(t) + \dot{a}(t) + \frac{1}{2}\ddot{a}\Delta t^2 + ... \tag{2.19}$$

where r is the position of atoms (x,y,z components), $\dot{r}$ is the velocity v, and $\ddot{r}$ is the acceleration a. The most famous finite-difference methods or algorithms used are:

- Euler's Method.
- Runge-Kutta Method.
- Predictor-Corrector Method.
- Verlet Method.

Verlet Method is the chosen numerical method to used in Molecular Dynamics for the numerical solution of the Newton's equations of motion, because it is reflect the symmetry of Newton's equations. Verlet method has many algorithms developed to solve ordinary differential equations, the following sections explain these algorithms.

2.6.1 Original Verlet Algorithm

It is one of the most popular algorithms because of its simplicity. To obtain the main equations of verlet algorithms let us expand the coordinates at times $t + \Delta t$ and at $t - \Delta t$ by Taylor expansion[22]:

$$r(t + \Delta t) = r(t) + v(t) + \frac{1}{2}a\Delta t^2 + ... \tag{2.20}$$

$$r(t - \Delta t) = r(t) - v(t) + \frac{1}{2}a\Delta t^2 + ... \tag{2.21}$$

by adding the two equations (2.20) and (2.21), the main equations of the Verlet algorithm are:

$$r(t + \Delta t) = 2r(t) - r(t - \Delta t) + a(t)\Delta t^2 \tag{2.22}$$

The verlet main algorithm stores coordinates and accelerations per time step. Velocities are not present explicitly in this algorithm[22].

2.6.2 The Verlet Leapfrog Algorithm

The Verlet leapfrog algorithm is an economical version of the basic Verlet algorithm. The main equations defining this algorithm are[23]:

$$r(t + \Delta t) = r(t) + v(t - \frac{1}{2}\Delta t)\Delta t \tag{2.23}$$

$$v(t + \frac{1}{2}\Delta t) = v(t - \frac{1}{2}\Delta t) + a(t)\Delta t \tag{2.24}$$

$$v(t) = \frac{1}{2}[v(t - \frac{1}{2}\Delta t) + v(t + \frac{1}{2}\Delta t)] \tag{2.25}$$

This algorithm needs only to store one set of positions and velocities of the atoms which means it is simpler to program than the basic version of

the verlet algorithm. The velocities given by equation (2.24) are half-step velocities calculated at time $t + \frac{1}{2}\Delta t$, while the positions are calculated at time $t + \Delta t$. This means that the velocities and positions leap over each other[23].The full-step velocities could be obtained by the equation (2.25).

2.6.3 The Verlet Velocity Algorithm

Verlet velocity is a modification of the basic verlet algorithm. It calculates the atomic positions and velocities at the same time t. The basic equations of this algorithm are:

$$r(t + \Delta t) = r(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2 \tag{2.26}$$

$$v(t + \Delta t) = v(t) + \frac{1}{2}[a(t) + a(t + \Delta t)]\Delta t \tag{2.27}$$

The advantage of this algorithm is that it requires less memory than the other versions of verlet algorithm because one set of positions and velocities need to be carried at any one time t.

2.6.4 The Beeman's Algorithm

Beeman's algorithm is a numerical integration method closely related to verlet integrations. It is more accurate in calculating positions and velocities of atoms in molecular dynamics simulations than the other verlet

integrations. According to Beeman's algorithm the translational positions and velocities of atoms are given by equations[24]

$$r(t+\Delta t) = r(t) + v(t)\Delta t + \frac{2}{3}a(t)\Delta t^2 - \frac{1}{6}a(t-\Delta t)\Delta t^2 \tag{2.28}$$

$$v(t+\Delta t) = v(t) + \frac{1}{3}a(t+\Delta t)\Delta t + \frac{5}{6}a(t)\Delta t - \frac{1}{6}a(t-\Delta t)\Delta t \tag{2.29}$$

This algorithm provides a greater precision and energy conservation. However, it is computationally more expensive[25].

2.7 Statistical Mechanics

Statistical mechanics (also called statistical thermodynamics) is the link between the properties of molecules (microscopic behavior) and the behavior of macroscopic matter. So, if the distribution of the particles of the system among its energy states is known, the macroscopic properties of the system can be predicted[26, 27].

A molecular dynamics simulation yields a trajectory that contains the positions of each atom along the simulation time. These are used to calculate the velocities of the atoms making up the molecule as a function of time. Statistical mechanics is then used to convert this microscopic infor-

mation (position and velocity) to macroscopic properties such as pressure, energy and heat capacity.

To calculate macroscopic states from microscopic states the time independent statistical averages are calculated. These averages are defined in terms of ensemble averages. An ensemble average is defined as an average taken over a large number of replicas of the system considered simultaneously[10]. An ensemble average is given by the equation (2.30):

$$\langle A \rangle_{ensemble} = \int \int dp^N dr^N A(p^N, r^N) \rho(p^N, r^N) \tag{2.30}$$

where p is the momenta and r the positions of the system. The integration is over all possible variables of r and p. The probability density of the ensemble is given by equation (2.31):

$$\rho(p^N, r^N) = \frac{1}{Q} \exp[-H(p^N, r^N)/K_B T)] \tag{2.31}$$

where H is the Hamiltonian, T is the temperature, K_B is Boltzmanns constant and Q is the partition function.

$$Q = \int \int dp^N dr^N \exp[-H(p^N, r^N)/K_B T)] \tag{2.32}$$

Calculating the double integral in equation(2.32) is very difficult over all possible states of the system, but in a MD simulation the positions of atoms

are calculated as a function of time along the simulation, so it is easy to calculate a time average of A as shown by equation (2.33):

$$\langle A \rangle_{time} = \lim_{t \to \infty} \frac{1}{\tau} \int_{t=0}^{t} A(p^N(t), r^N(t))dt \approx \frac{1}{M} \sum_{t=1}^{M} A(p^N, r^N) \tag{2.33}$$

where τ is the simulation time, M is the number of time steps along the simulation and $A(p^N, r^N)$ is the instantaneous value of A. The problem now is how one could calculate the ensemble averages from time averages by a MD simulation. The ergodic hypothesis solved this problem because it states that the time average equals the ensemble average (equation (2.34)):

$$\langle A \rangle_{ensemble} = \langle A \rangle_{time} \tag{2.34}$$

According to the ergodic hypothesis any physical quantity can be calculated from the MD trajectory as an average of this quantity values along the simulation time. For example the average potential energy along trajectory time is calculated as shown by equation(2.35):

$$V = \langle V \rangle = \frac{1}{M} \sum_{i=1}^{M} V_i \tag{2.35}$$

where M is the number of configurations in the molecular dynamics trajectory and V_i is the potential energy of each configuration.

Also the kinetic energy of the system is calculated as shown in equation (2.36):

$$K = \langle K \rangle = \frac{1}{M} \sum_{j=1}^{M} \{ \sum_{i=1}^{N} \frac{m_i}{2} v_i \cdot v_i \} \tag{2.36}$$

where M is the number of configurations in the simulation, N is the number of atoms in the system, m_i is the mass of the particle i and v_i is the velocity of particle i.

2.8 Molecular Modeling

Protein conformational energy can be calculated using the approximations of potential energy functions[16]. Many potential energy functions are developed which is mathematical functions of the atomic coordinates, in this section the basic techniques used to predicting protein structure based on the minimum energy assumptions will introduce.

2.8.1 Molecular Mechanics

Molecular mechanics (MM) is a series of mathematical and classical physics assumptions used to model the structure of large and complex molecules in a practical manner [28]. Molecular mechanics studies molecules containing thousands of atoms such as organics,oligonucleotides, peptides, and saccharides (metallo-organics and inorganics in some cases)[29]. To cal-

culate the forces acting on these atoms the molecules are treated as a collection of particles held together by simple harmonic forces. Atoms are considered as spheres, and bonds between atoms as springs (Fig.2.3)[4, 30, 29].

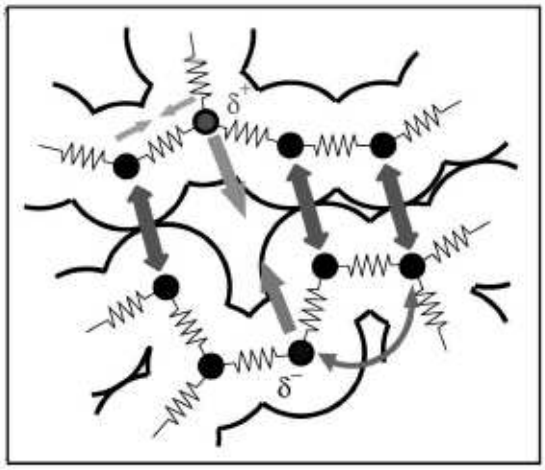

Figure 2.3 A protein molecule. Dark dots represent backbone atoms. Springs represent the bonds between the atoms.

2.8.2 Empirical Potential energy functions

In molecular mechanics, energy consists of bonded (stretching, bending and torsion) and non-bonded (van der Waals and electrostatic) terms[4, 7, 12, 30, 29, 31, 32]:

$$E = E_{(Bonded)} + E_{(Non-Bonded)} \tag{2.37}$$

$$E = E_{stretching} + E_{bending} + E_{torsion} + E_{vanderwaals} + E_{electrostatic} \tag{2.38}$$

Molecular mechanics calculates the energy of a molecule and then adjusts the energy through changes in bond lengths and angles to obtain the minimum energy structure[32].

A force field is a set of functions used to calculate the potential energy of the molecule. Most force fields are composed of empirically determined interaction terms. A number of these force fields have been developed for biomolecules, specifically for proteins and DNA/RNA. The most common are CHARMM, GROMOS96, AMBER, and OPLS[4, 7]. AMBER force field is commonly used to study protein dynamics[31]. This force field describes interactions between the molecule atoms from covalent chemical bonds and non-bonds (non-covalent) interactions. Details of each bond function are explained as follows:

Bonded Interactions

For protein molecules the intramolecular bonding interactions include terms of the following types:

1. **Stretching Energy:** is a harmonic potential representing the interaction between any two atoms in the protein molecule separated by one covalent bond[29, 31]. This energy is based on the Hooke potential.

$$E_{stretch} = \sum_{pairs} k_s(r - r_0)^2 \tag{2.39}$$

where r_0 is an initial bond length (equilibrium distance), k_s is the force constant for the bond , and r is the new distance between two atoms. Both r_0 and k_s are unique for each bond and depend on the atom types[32].

2. **Bending Energy:** is the energy required to bend a bond from its equilibrium angle θ_0. This energy is based on Hook's law in which the bending bond is modeled by a spring [32].

$$E_{bend} = \sum_{angles} k_\theta(\theta - \theta_0)^2 \tag{2.40}$$

Both θ_0 and k_θ are constant values that depend on the type of atoms constituting the angle (e.g. C-C-C, C-O-C, C-C-H, etc.)[29]

3. **Torsion Angle Energy:** is the energy of torsion required to rotate about single bonds only because double and triple bonds are too rigid to rotate[32]. So it models the presence of steric barriers between atoms separated by 3 covalent bonds (1,4 pairs)[10]. The dihedral angle rotation is shown in fig.2.4.

$$E_{dihedral} = \sum_{torsions} k_\phi(1 + cos(n\phi - \delta)) \tag{2.41}$$

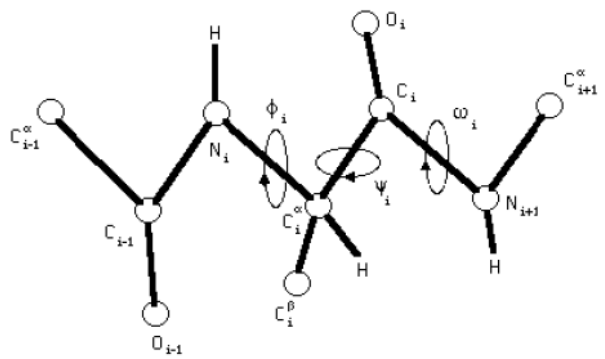

Figure 2.4 The backbone dihedral angles Φ, Ψ and ω[16].

The constant k_ϕ is the rotation force constant, n is the coefficient of symmetry (n=1,2,3), and ϕ is the dihedral angle about the bond. Both k_ϕ and n depend on the type of bonds between atoms (e.g. C-C-C-C, C-O-C-N, H-C-C-H, etc). The constant k_ϕ controls the amplitude of the potential curve, while the n parameter controls its periodicity[10, 29, 32, 33]

Non-Bonded Interactions

The most important non-bonded energy terms in protein MD are the Van der Waals and the electrostatic interactions of atom pairs.

1. **Van der Waals Energy:** is the interaction energy of the atomic pair

(i and j) is treated with Lennard-Jones potential function as shown in equation(2.42)[33]. This interaction between two atoms arises from a balance between repulsive and attractive forces[29].

$$E_{VdW,ij} = \sum_{Non-bondedpairs} (-\frac{A_{ij}}{r_{ij}^6} + \frac{B_{ij}}{r_{ij}^{12}}) \tag{2.42}$$

where A and B are constants that depend on the type of atoms involved in the non-bonded interactions (e.g. C:C, O:C, O:H, etc.), r_{ij} is the distance (in Angstroms) between the two nuclei, and the $-\frac{A_{ij}}{r_{ij}^6}$ part is the attractive part and the $+\frac{B_{ij}}{r_{ij}^{12}}$ part is the repulsive part of the interaction. When the atomic pair distance is very small, the two atoms are very close together, and the function is highly repulsive. But when the two atoms are far apart, the interaction is attractive between them. The interatomic attraction decreases as the separation distance approaches infinity[33].

2. **Electrostatic Energy:** the electrostatic interaction energy of the atomic pair (i and j) is represented by the coulomb potential function (equation 2.43)when the signs of the partial charge are[33].

$$E_{electrostatic,ij} = \sum_{Non-bonded(pairs)} \frac{kQ_iQ_j}{4\Pi\varepsilon r_{ij}} \tag{2.43}$$

where Q_i and Q_j are the partial atomic charges for atoms i and j separated by a distance r_{ij}, ε is the relative dielectric constant and K is a units conversion constant in Kcal/mol.

2.9 Analysis Techniques

The MD simulation trajectories are huge sets of data. Efficient computational methods and statistics are needed to analyze and compress these huge data. This is done by reducing the number of dimensions without much loss of information, and by highlighting similarities and differences of data during simulation time[34]. The Essential Dynamics (ED) method (see section 2.9.2) and other statistical calculations like, Root Mean Square Deviation (RMSD) (see section 2.9.1), and variance and covariance calculations, form an efficient computational method used to analyze and compress the large data produced by MD simulations. This method is a standard tool used to investigate the important biomolecular motions[35].

This section contains the most common computational techniques used to analyze protein motion trajectories produced by MD simulations.

2.9.1 Root Mean Square Deviation

The Root Mean Square Deviation (RMSD) is a numerical statistic measure of the difference of two structures[36]. For protein dynamics, RMSD

is used to calculate the distance between two aligned conformations of C_α atoms (carbon backbone atoms)[7].

$$RMSD = \sqrt{\frac{\sum_{i=1}^{N}(r_i(t_1) - r_i(t_2))^2}{N}} \tag{2.44}$$

where N represents the number of atoms, r_i is the position of atom i (x,y,z coordinates) at time t.

$$RMSD = \sqrt{\frac{\sum_{i=1}^{N}((\Delta x_i)^2 + (\Delta y_i)^2 + (\Delta z_i)^2)}{N}} \tag{2.45}$$

2.9.2 Essential Dynamics

The Essential Dynamics technique (ED) is equivalent to the Principal Components Analysis (PCA) of atomic displacements in an ensemble of structures. It has been used to divide the conformational space of motion into two major subspaces: the essential subspace explaining the atomic fluctuations, and the subspace in which the fluctuations have random characteristics (local fluctuations)[6, 35].

Essential dynamics analysis is used to distinguish large concentrated internal motions from small, random, and local internal motions[37]. Also ED analysis is used to remove all of non-internal motions like rotation and translation of protein molecule by fitting its atoms to a reference structure.

The ED or principal component analysis is based on construction and diagonalization (find eigenvectors and eigenvalues) of the covariance matrix of positional fluctuations[6, 7].

In short, the ED method can be summarized by two major steps[7, 37, 34, 38, 39]:

1. Eliminating the overall rotational and translational motions from the simulation data, because only internal motion of protein atoms are of interest. This is done by fitting and aligning atom trajectories of the protein to a reference frame.

2. Building a covariance matrix and computing its eigenvectors and eigenvalues. This allows for a description of internal (essential) motions of protein atoms by using PCA method.

2.9.3 Principal Component Analysis Technique

Principal component analysis (PCA) is a valuable applied linear algebra technique. PCA has been used in many diverse fields ranging from neuroscience to computer graphics. It can identify the dominant variables and mechanisms that describe and control the structure and processes underlying specific data sets[38, 39].

In PCA the values of data (observations) are placed in a 2-dimensional

matrix of size $(n \times p)$ where the rows represent one index and columns represent the other. PCA was initially developed to explain the variance-covariance structure of a set of variables by linearly combining the original variables[40].

$$X = [x_1, x_2, \ldots, x_p] \tag{2.46}$$

X is the trajectory variables (data set). Through PCA, a set of uncorrelated linear combinations can obtained in the following matrix[40]:

$$Y = A^T X \tag{2.47}$$

where $Y = (y_1, y_2, \ldots, y_p)^T$, y_1 is the first principal component (PC), y_2 is the second PC and so on. A is an orthogonal matrix with size $p \times p$ and $A^T A = I$. The eigenvectors and eigenvalues can be computed from the data matrix in which fewer eigenvectors than the number of original data vectors are required to sufficiently represent the data, which allows for data compression. [39].

Variance and Covariance

Variance is a measure of the spread of data in a data set. It is simply the standard deviation (SD) of data.

$$SD = \sqrt{\frac{\sum_{i=1}^{N}(X_i - \overline{X})^2}{N-1}} \tag{2.48}$$

$$var = \frac{\sum_{i=1}^{N}(X_i - \overline{X})^2}{N-1} \tag{2.49}$$

X_i represents the data set variables $(x_1, x_2, \ldots, x_N)$, $\overline{X}$ is the mean of data, and N is the number of data variables. It is obvious from the equation (2.49) that variance is the standard deviation squared. The variance formula could also be written as:

$$var(X) = \frac{\sum_{i=1}^{N}(X_i - \overline{X})(X_i - \overline{X})}{N-1} \tag{2.50}$$

The variance operates on one-dimensional data sets only. However, many data sets are multi-dimensional. A useful measure for multi-dimensional data sets is covariance, which is used to find out how much the dimensions of a data set vary from the mean with respect to each other[34].

$$cov(X, Y) = \frac{\sum_{i=1}^{N}(X_i - \overline{X})(Y_i - \overline{Y})}{N-1} \tag{2.51}$$

The covariance (Eq.2.51) is used for two dimensional data sets. If a data set is three-dimensional data (x, y, z), the covariance would be calculated between every two dimensions in the set, like covariance between x and y,

y and z, and so on. The covariance between x and x, y and y or z and z gives the variance of x, y and z respectively[34].

The covariance measures the degree of the correlation between two variables (two dimensions data set). The covariance value is an indicator of how the two variables relate to each other [34, 38]:

1. If the covariance value is positive $cov(X) > 0$, that means the two variables increase together.

2. If the covariance value is negative $cov(X) < 0$, that means that one variable increases while the other decreases.

3. If the covariance value is zero $cov(X) = 0$, that means the two variables are independent of each other (uncorrelated).

The Covariance Matrix

For an n-dimensional data set, the number of different covariance values that can be calculated are $\frac{n!}{(n-2)!\times 2}$ (n: number of variables in data set). The covariance matrix is a square matrix in which each entry in it is the covariance between two separate dimensions[34]. To calculate the covariance for an n-dimensional data set, all data measurements are ordered in $m \times n$ matrix X, with each row of the matrix containing the number of dimensions

or variables, and each column containing the set of measurements for one variable.

$$X = \begin{pmatrix} x_{11} & \ldots\ldots & x_{1n} \\ x_{21} & \ldots\ldots & x_{2n} \\ \vdots & \ldots\ldots & \vdots \\ x_{m1} & \ldots\ldots & x_{mn} \end{pmatrix} \tag{2.52}$$

The covariance matrix entries are calculated as follows:

$$c_{ij} = \frac{1}{n-1} \sum_{k=1}^{n} (x_{ik} - \overline{x}_i)(x_{jk} - \overline{x}_j), i, j = 1, ..., m \tag{2.53}$$

The c_{ij} value is the ij^{th} element of the covariance matrix $C_{n \times n}$, where the produced covariance matrix is a square $n \times n$ matrix (n is the number of variables in data set or its the number of columns in data matrix). The properties of this covariance matrix can be summarized as follows[38]:

1. C is a square symmetric matrix.

2. The diagonal entries of C are the variance of each variable in the data set with itself.

3. The non-diagonal entries of C are the covariance values between different measurement types.

2.9.4 The Essential Dynamics Method for Protein Dynamics Analysis

Essential Dynamics aims to identify a new reference frame such that only a subset of coordinates is sufficient to describe the over-all dynamics of the protein molecules[41]. This essential subspace, responsible for the majority of the fluctuations, is given by ten to twenty principal components[6]. The other remaining degrees of freedom corresponding to constrained harmonic oscillations can be neglected.

The first step in essential dynamics analysis, when applied to protein trajectories produced by MD simulations, is to remove all rotational and translational motions. This is done by fitting and aligning the protein atom coordinates along the trajectory to a reference frame. This is done by a translation of the center of mass of every configuration (frame) to the origin. This is followed by a least squares rotational fit of the atoms onto a reference structure(reference frame)[6].

The second step is to calculate the covariance matrix of the protein atoms positional fluctuations. Protein covariance matrix contains only backbone carbon (C_α) atoms trajectories because they are more stable than the proteins floppy side-chains and give a good indication of the proteins spatial conformation[36]. Moreover, Proteins C_α atoms contain all the im-

portant information of the protein large concerted motions.

A set of eigenvectors and eigenvalues of protein motion can be obtained by diagonalization of the covariance matrix. Each eigenvector describes a collective mode of protein motion that is not linearly correlated with any other in the system. The fluctuation order of this motion is given by the corresponding eigenvalues[3]

Fitting and Aligning the Structure

The first step in an essential dynamics analysis of protein atoms trajectory, is fitting the frames of the trajectory by a least squares fit method onto a reference frame. This step removes overall translational and rotational motion since only internal motions of the protein molecules are interesting to analyze[42, 43].

Let x_n and y_n (n representing the number of atoms)be two vector sets of atomic coordinates containing only backbone atoms (C_α) of the two protein structures. A least squares procedure is used to fit y_n to x_n (reference frame) as follows:

1. Optimal alignment based on atom positions is defined by minimizing the "distance" between two frames G_1 and G_2, which is the sum of the squared distance between corresponding atoms[44].

$$d(G_1, G_2) = \sum_{i=1}^{n} (a_{1i} - a_{2i})^2 \tag{2.54}$$

2. In protein alignment we try to place every frame as close as possible to the reference frame. To do so we need a rigid transformation R_i to minimize the distance $d(R_i(G_i), G_{ref}), i = 1, ..., M$ (M number of geometries or frames that to be aligned), where G_{ref} is the reference frame,G_i represents the frame to be aligned. R_i optimal can be computed according to the Kasbah method in which all frames translate in such a way that their barycenters $b_i = \frac{1}{n}\sum_{j=1}^{n} a_{ij}$ coincide with barycenters of the reference frame[44, 45].

3. Building a symmetric 3×3 matrix:

$$R_i = \sum_{k=1}^{n} a_{ref,k} \otimes a_{i,k} \tag{2.55}$$

4. The optimal rotation for the frame G_i can be found by finding eigenvectors and eigenvalues of $R_i^T R_i$ matrix.

5. Now the rotation matrix can be computed from the normalized eigenvectors p_i and eigenvalues q_i:

$$U = \sum_{i=1}^{3} q_i \otimes p_i \tag{2.56}$$

This method is fast for small proteins (around 600 residues), but it is slow for large proteins. The adaptive selection algorithm used appears to have computational complexity of order N^2[46].

Diagonalization the Covariance Matrix

The covariance matrix is constructed from the C_α atomic positions along the simulation time according to the equation (2.57):

$$C_{ij} = \langle (X_i - \langle X_i \rangle)(X_j - \langle X_j \rangle) \rangle \tag{2.57}$$

where X are the x,y and z coordinates of atoms, $\langle X \rangle$ are the ensemble average over time. The diagonalization of this symmetric matrix gives a set of eigenvectors and eigenvalues in which the transformation matrix T contains the eigenvectors (principal components of C) as columns, and the resulting diagonal matrix A contains the corresponding eigenvalues[6, 47]:

$$A = T^T C T \tag{2.58}$$

The eigenvectors represent a direction in a high-dimensional space, describing concerted fluctuations of protein atoms (C_α)[47]. These eigenvec-

tors are sorted according to their eigenvalues. The eigenvectors with the large eigenvalues approximate the sum of protein fluctuations. Those with small eigenvalues represent the most constrained degrees of freedom[6].

The Overlap of Protein Fluctuations

Overlap of two vectors is the volume of the intersection between them. The overlap between two subsets of a protein simulation (A and B) is a common PCA approach used to explore the same conformational space[3].In other words, it can be used as a measure for the convergence of the sampled space[2]. To do so, we use the expression:

$$\Psi_{A,B} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}(v_{Ai} \cdot v_{Bj})^2 \tag{2.59}$$

where Ψ is the subspace overlap, v_A and v_B are two subset of eigenvectors of each ensemble (A and B), and n is the number of eigenvectors used. The eigenvectors are chosen to represent the significant proportion of the fluctuations in the simulation[3]. An overlap value of 0 occurs when two eigenvector subsets are completely dissimilar, and 1 when they are identical.

The other approach for quantifying the similarity between conformational spaces obtained from two time-windows of the simulation uses all the eigenvectors of the covariance matrix and their corresponding eigenvalues, and is calculated as[2]:

$$s(A,B) = 1 - \frac{d(A,B)}{\sqrt{trA + trB}} \tag{2.60}$$

where s is the overlap between two covariance matrices A and B, tr is the trace of a matrix and d(A,B) is the difference between the covariance matrices A and B defined as:

$$d(A,B) = \sqrt{tr[(A^{1/2} - B^{1/2})^2]} \tag{2.61}$$

where

$$A^{1/2} = Rdiag(\lambda_1^{1/2}, \lambda_2^{1/2} \ldots, \lambda_n^{1/2})R^T \tag{2.62}$$

so

$$d(A,B) = [\sum_{i=1}^{n}(\lambda_{Ai} + \lambda_{Bi}) - 2\sum_{i=1}^{n}\sum_{j=1}^{n}\sqrt{\lambda_{Ai} + \lambda_{Bi}}(R_{Ai} \cdot R_{Bj})^2]^{1/2} \tag{2.63}$$

where R is the matrix its columns being the eigenvectors R_i, λ_i are the eigenvalues and diag() is a diagonal matrix. Now the overlap s can be defined as:

$$s(A,B) = [\frac{\sum_{i=1}^{n}(\lambda_{Ai} + \lambda_{Bi}) - 2\sum_{i=1}^{n}\sum_{j=1}^{n}\sqrt{\lambda_{Ai} + \lambda_{Bi}}(R_{Ai} \cdot R_{Bj})^2}{\sum_{i=1}^{n}(\lambda_{Ai} + \lambda_{Bi})}]^{1/2} \tag{2.64}$$

The overlap s is one if two matrices A and B are identical,and is zero when the sampled subspaces are completely orthogonal.

Chapter 3
Computing Tools and Methodology

This chapter contains a brief introduction to each tool used in running the BPTI simulation, and in analyzing the resulting trajectory. Also it contains an explanation of the methodology used to perform a molecular dynamics simulation of BPTI.

3.1 Molecular Dynamics Tools Used

The computational tools used in this research are divided into two main categories. First, the tools used in preparing and running molecular dynamics simulations of protein (NAMD and VMD), and the second are the tools used in principal component analysis of the resulting protein trajectory (Catdcd, Flipdcd, and MATLAB)

3.1.1 NAMD Molecular Dynamics Software

NAMD (NAnoscale Molecular Dynamics) is a parallel, object-oriented molecular dynamics software. It was developed by the Theoretical Biophysics Group at Illinois Beckman Institute (University of Illinois). It is supported by the National Institutes of Health Resource for Macromolecular Modeling and Bioinformatics, and the National Science Foundation. NAMD is designed for high-performance simulations of biological systems

on a number of different hardware platforms[48, 49].

NAMD is based on Charm++ parallel objects, used in other molecular dynamics simulation programs (e.g. X-PLOR, CHARMM, GROMACS, AMBER, ...). It computes atomic trajectories by solving equations of motion numerically using empirical force fields. The main advantage of NAMD over other MD packages, is the ability of NAMD to run efficiently on parallel processors for simulating large molecules.

3.1.2 Visual Molecular Dynamics (VMD) Software

VMD is a molecular graphics software developed as NAMD by the Theoretical and Computational Biophysics group at the University of Illinois and supported by grants from the National Institutes of Health and the National Science Foundation. VMD is used in research in different disciplines like biology, physics, chemistry and biophysics. VMD is designed for the visualization and analysis of biomolecular systems such as proteins[36].

VMD is available free of charge for end-users at:
http//:www.ks.uiuc.edu/reserach/vmd/. It supports Microsoft Windows XP/NT, MAC and Unix platforms. The main feature of VMD is its ability to animate and analyze the trajectory of MD simulation.

3.1.3 Catdcd Tool

A DCD file is a binary format file used by CHARMM, X-PLOR and NAMD. It contains the coordinate trajectories produced by one of the simulation packages. The DCD format is structured as a FORTRAN UNFORMATTED data type[48]. The output file is always binary because it requires much less space than text file, and reading it is much faster than reading text format.

Catdcd tool is used just like the Unix "cat" command to concatenate DCD files into a single DCD file. It is also used to specify which atoms should be written into the output DCD file[50]. In this work Catdcd was used to separate C_α (protein backbone atoms) coordinate trajectories from the water in the final output DCD file.

3.1.4 FlipDCD Tool

The binary format of the DCD trajectories produced by simulations depends on the the machine operating system type (Unix, MS Windows,...etc) which is running the simulation. The byte ordering or endianism is different from one machine type to another. For example, Unix uses little endianism, while Windows uses big endianism. To overcome the problem of a platform mobility a conversion tool is needed to convert from little endianism to big

endianism and viceversa.

FlipDCD is a tool used for converting binary trajectory files between little and big endian formats. The utility program FilpDCD is provided with the NAMD Linux/Intel version to reformat the DCD files endianism[48]. It reverses the endianism by memory mapping the DCD file and converting the endianism in-place. This allows the use of a Windows PC to read DCD trajectories generated on a Sun platform (Unix), and for a Sun platform to read trajectory files produced on a PC cluster running Linux. Moreover, FlipDCD can be used to report the endian status of DCD files without regard for the origin of the DCD files[51].

3.1.5 MATLAB: High-Performance Computation Software

MATLAB is a high-performance language for technical computing used for numerical computation and graphics. In other words, MATLAB is a software designed for matrix computation, solving systems of linear equations, computing eigenvalues and eigenvectors and so forth[52]. It was introduced in the late 1970s by Cleve Moler at the University of New Mexico to provide students an easy access to matrix software developed by LINPACK and EISPACK projects. MATLAB now is one of the products of The MathWorks corporation used widely in many fields of research like natural sciences and engineering to perform the numerical analysis and image

processing.

In this work MATLAB is used as the main analysis tool. It is used to read the binary DCD files (trajectory files). It is also used to compute the covariance matrix by using the MATLAB's function cov(X), where X is the data matrix. The eigenvalues and eigenvectors are found for the covariance matrix in MATLAB by using the the function eig(A) which returns eigenvalues of matrix A and function [V,E]=eig(A) which returns the matrix V containing the eigenvectors of matrix A as columns, and diagonal matrix E containing the eigenvalues of matrix A. Moreover, all the graphs in this work were plotted using the MATLAB graphical capabilities.

3.2 Molecular Dynamics Simulation

To run a MD simulation of BPTI using NAMD package, four files are needed[7]:

1. A BPTI file (6PTI) from the Protein Data Bank (PDB)[53], which contains the initial atomic coordinates and velocities for protein.

2. A force field parameter file which contains all the needed information about the bonded and non-bonded interactions between protein molecule atoms. The force field has four types: CHARMM, X-PLOR, AMBER and GROMACS. In this work the CHARMM force field type

was used.

3. Protein Structure File(PSF) of BPTI generated from the initial PDB file and the parameter files using VMD software. This file contains the structural information of the protein.

4. NAMD configuration file which contains all the simulation options and commands needed to run the simulation.

Running the simulation for a molecular system follows these steps[10]:

1. Preparation of the protein molecule for simulation by generating the needed PDB and PSF protein files.

2. Minimization which is important for the stability of the simulation.

3. System Heating to reach the desired temperature.

4. Equilibration the system to avoid any distortion in the results.

5. Running the simulation and getting the final data.

6. Analyzing the resultting MD simulation data.

3.2.1 Preparation of the BPTI Molecule

In this phase, all the files required to run the simulation using NAMD package are generated. The PDB and PSF files of BPTI are generated from

6PTI (Initial BPTI file from data bank) by using psfgen within VMD. A parameter file which contains the description of bond, angle, torsion energy and non-bonded interactions, and a topology file which contains the types of molecule atoms, are needed in the preparation of PDB and PSF files[7]. The PSF and PDB files of BPTI are built as follows:

1. The 6PTI.pdb original file is split into two segments, $6PTI\_protein.pdb$ and $6PTI\_water.pdb$. All non-protein atom records are removed using the grep command in VMD as follows[48]:

   - **grep -v 'ĤETATM' 6PTI.pdb > output/6PTI_protein.pdb**
   - **grep 'HOH' 6PTI.pdb > output/6PTI_water.pdb**

2. Run the psfgen package within VMD1.8.4 as follows[48]:

   - **psfgen**

     Run the psfgen commands within VMD shell script .

   - **topology toppar/top_all22_prot.inp**

     Read the topology definitions for the residues from a folder called toppar[48].

   - **segment BPTI pdb output/6PTI_protein.pdb**

     Build protein segment of BPTI and adds the hydrogen atoms.

- **patch DISU BPTI:5 BPTI:55**
- **patch DISU BPTI:14 BPTI:38**
- **patch DISU BPTI:30 BPTI:51**
- **pdbalias atom ILE CD1 CD**

  In the residue ILE, the atom CD is called CD1 in the pdb file. The "pdbalias atom" command is used here to define the correct name.
- **coordpdb output/6PTI_protein.pdb BPTI**
- **guesscoord**

  This command guesses the missing coordinates locations of many atoms, particularly hydrogens atoms[48].
- **writepdb output/bpti.pdb**

  Writes the coordinates of all BPTI atoms (including hydrogen atoms).
- **writepsf output/bpti.psf**

  Writes the structure file of BPTI.

The PSF and PDB files of BPTI are ready to run the simulation in vacuum. For solution we need to solvate and ionize the BPTI in a water box[7].

3.2.2 Solvation and Neutralization of BPTI in Water

Most activities of proteins occur in solution environment. Thus to mimic the real environment of BPTI, we need to solvate it in water. VMD was used to build the PSF and PDB files of BPTI in a water box. This is done as follows :

- **package require solvate**

  Invoke the solvate package to use with VMD commands.

- **solvate BPTI.psf BPTI.pdb -t 6 -o** $BPTI\_water$

  This command allows the solvate package to place the BPTI files (BPD and PSF) in a water box with enough size to avoid the interaction between the protein and its image in the next cell of the Periodic Boundary Conditions (PBC)[7]. The option -t define the dimensions of the water box. The option -o exports the final BPTI files with the water box.

- **set everyone [atomselect 0 all]**

  Atomselect is the command to access information about the atoms in a molecule. It takes the Id of the molecule (in our case 0) and number of frames to select (all frames in our command).

- **measure minmax $everyone**

  Returns the minimum and maximum coordinates (x,y,z) of the protein atoms in the water.

- **measure center $everyone**

  Return the center of the selected atoms (center of the water box).

The next step after creating the water box and putting the protein molecule in it, is adding a number of ions to the solvent to make its charge zero. The net charge of all atoms should be determined to now the number of ions needed to neutralize the charge of the system. The autoionize package withen VMD calculate the net charge of the system and adds Na and Cl ions to the solvent. Ion positions are random, but there are minimum distances between ions and molecule as well as between any two ions. If an ion concentration is specified, autoionize will also attempt to neutralize the total charge of the system. This package is used to ionize the BPTI files as shown in the following commands:

- **package required autoionize**

  Run the autoionize package within VMD.

- **autoionize -psf** *BPTI_water.psf* **-pdb** *BPTI_water.pdb* **-is 0.05**

This command tells the autoionize package to compute the net charge of the solvent, and to add the appropriate number of ions (Na) and (Cl) to the solvent to make its charge zero. The option -is defines the desired concentration (mol/l).

The resulting file after the ionization is ready to start the simulation in solution (Fig.3.1).

3.2.3 Minimization Phase

Energy minimization of the protein structure should be done before starting a molecular dynamics simulation in order to remove any strong Van Der Waals interactions, which might cause local structural distortion and lead to an unstable simulation[10]. Energy minimization of solvated BPTI is done with the protein fixed in its energy minimized positions for 5 ps using NAMD.

3.2.4 Heating Phase

This phase of the MD simulation aims to raise the temperature of the system (protein and water) to the desired one. The initial velocities of the BPTI at low temperature (0 K) are assigned to each atom, in the system. To raise the temperature of BPTI from zero to 300 K, new velocities are assigned at a slightly higher temperature while the simulation is running,

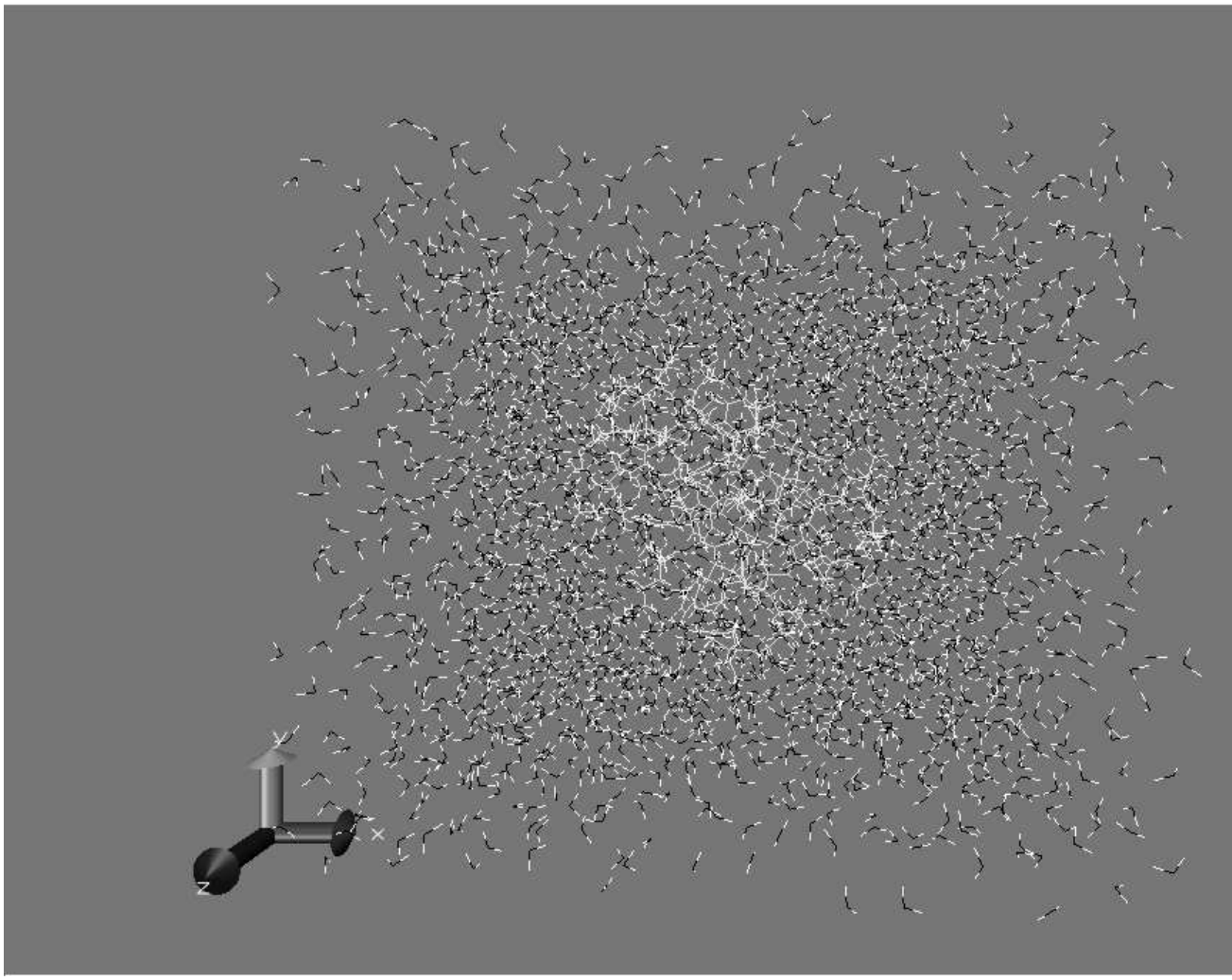

Figure 3.1 BPTI Molecule after ionization in a water box (solution).

and the simulation is allowed to continue until the desired temperature (300 K) is reached[10].

3.2.5 Equilibration Phase

This phase aims to make sure that the simulation is stable. This means that the properties of the simulated system like the pressure, the energy and temperature, are stable with respect to time. In this work, the BPTI was equilibrated to 40 ps using the temperature reassignment method[7]. If the temperature of the BPTI solvation increases or decreases, the velocities can be scaled such that the temperature returns to its desired value[10].

3.2.6 Production Phase

This phase is the final step in which the simulation of BPTI in solution runs for 30 ns at 300 K. To run the simulation, a parameter file is needed (Appendix B). It specifies the time step, electrostatic, constrains, fixed atoms, periodic boundary conditions, input files, calculated thermodynamics parameters and output files.

The resulting file is the DCD file which store the coordinates (x,y,z) of each atom in the system every 50 fs. This file is the trajectory of the solvated protein motion along the simulation time (30 ns). It has 600000 conformations (frames), and a total size of 57 GB (Giga Byte).

3.2.7 Analysis of MD Simulation Trajectories

The resulting DCD trajectory file contains all the information about protein motion and its water environment. To analyze this trajectory we first need to separate the C_α atoms coordinates from the water in the DCD trajectory. The C_α atoms in BPTI contain the interesting information about the motion of the protein molecule in solvent[36, 47]. The catdcd tool is used to separate these C_α atoms coordinates from the DCD trajectory file. Using catdcd to do the separation needs an index file for C_α atoms. This file is prepared by writing the following VMD commands :

- **set ca [atomselect 0 "name CA"]**
- **$ca get index**
- **$ca writepdb ca.pdb**

The index of C_α resulting from VMD is saved in a file named index.txt using the unix command cat: cat> index.txt. The catdcd tool can used to separate C_α form the total trajectory by writing the following line in the unix prompt directory where the catdcd is installed:

- **./catdcd -i index.txt -o calpha.dcd/D/dcdresults/traj.dcd**

Catdcd produce a new DCD file called calpha.dcd that contains only the C_α atom coordinates. This new DCD file can be opened by VMD with

the ca.pdb file. The motion of BPTI as seen on VMD has a rotational and translational part. The only motion of interest is the internal motion. So, there is a need to fit and align the protein frames of motion to a reference frame.

To do the fitting and alignment, we choose a reference frame after the simulation becomes stable and reaches the desired temperature (300 K). In this work the reference frame chosen was at 1ns from the simulation time. Fitting and alignment to the chosen reference frame is done by running the following script[36] in VMD:

- **proc fit-align {{mol top}}** {
- **# use frame 20000 for the reference**
- **set reference** [**atomselect $mol all frame 20000**]
- **# the frame being compared**
- **set compare** [**atomselect $mol all**]
- **# $mol is the molecule ID on VMD**
- **set num_steps** [**molinfo $mol get numframes**]
- **for {set frame 20000} {$frame < $num_steps} {incr frame}** {

- **# get the correct frame**
- **$compare frame $frame**
- **# compute the transformation**
- **set trans_mat [measure fit $compare $reference]**
- **# do the alignment**
- **$compare move $trans_mat**
- **}**
- **}**

After running the fitting and aligning script, the protein is checked on the VMD display window, to make sure that it is fitted and aligned The fitted coordinates are saved in a new DCD file. The new fitted DCD file is little endian. To analyze this file under Windows platforms (big endian)we need to change the endianism of the file. The flipdcd tool which distributed with NAMD package is used to convert the endianism of the DCD trajectory as explained in section 3.1.4. The command is:

- **./flipdcd DCDfile-name.dcd**

The first analysis technique of the fitted trajectory of BPTI can applied by calculating the RMSD values of all simulated protein frames or structures. This is done by running the following script[36] on the fitted DCD file opened by VMD:

- **# Prints the RMSD of the protein atoms between each timestep**
- **proc print_rmsd {{mol top}}** {
- **# use frame 20000 for the reference**
- **set reference [atomselect $mol all frame 20000]**
- **# the frame being compared**
- **set compare [atomselect $mol all]**
- **set f [open rms1.txt a]**
- **# $mol is the molecule ID on VMD**
- **set num_steps [molinfo $mol get numframes]**
- **for {set frame 20000} {$frame < $num_steps} {incr frame}** {
- **# get the correct frame**

- **$compare frame $frame**
- **# compute the transformation**
- **set trans_mat [measure fit $compare $reference]**
- **# do the alignment**
- **$compare move $trans_mat**
- **# compute the RMSD**
- **set rmsd [measure rmsd $compare $reference]**
- **# print the RMSD**
- **puts $f "$rmsd"**
- **}**
- close **$f**
- **}**

After running the print_rmsd script the RMSD values are stored in the text file rms1.txt, then plotted using MATLAB (Fig.4.2).

The next step is to find the covariance matrix of the data matrix (coordinates matrix). MATLAB's function COV(X) is used to calculate the

covariance matrix, where X is the DCD file trajectory which is read in MATLAB by using the M-file named readdcd. The B-factor is calculated from the diagonal element of the MD simulation covariance matrix (Appendix C.4) as in the equation(3.1):

$$B_i = \frac{8}{3}\Pi^2(c_{ii}) \tag{3.1}$$

where c_{ii} is the covariance matrix diagonal element and B_i is the B-factor for atom i.

The eigenvectors and eigenvalues of the covariance matrix are calculated by using the MATLAB function [V,E]=eig(A), where A is the covariance matrix and V is the resulting eigenvectors matrix (columns of the matrix are the eigenvectors) and E is the eigenvalues diagonal matrix. The overlap of protein fluctuations now can be calculated by two methods: the first is the total overlap between covariance matrices (equation 2.64)of simulation subintervals time by using MATLAB (Appendix C.2). The other method is calculating the overlap between the eigenvectors for approximately 86% of fluctuations by the squared inner product of eigenvectors subspaces(equation (2.59)) along the simulation time using MATLAB (Appendix C.3).

Chapter 4
Results and Analysis

4.1 B-Factor

The atomic B-factor is one of the few measurements for protein flexibility that can be obtained from experimental data (X-ray diffraction data) [3]. To validate simulation results the B-factor is calculated from the data resulting from the simulation using equation (3.1), and compared to the B-factor from the experimental data for BPTI protein which is available from the Protein Data Bank[53].

The calculated B-factor values from the simulation compared with the experimental crystallographic B-factors are shown in Fig.4.1. The simulation B-factor values agree with the experimental crystallographical values except two regions. The first is between residues 10 and 20, and the second region is between residues 34 and 42. Its also clear from the figure (Fig.4.1) that the simulated B-factor value increases with time especially in the disagreement region between simulation and experiment.

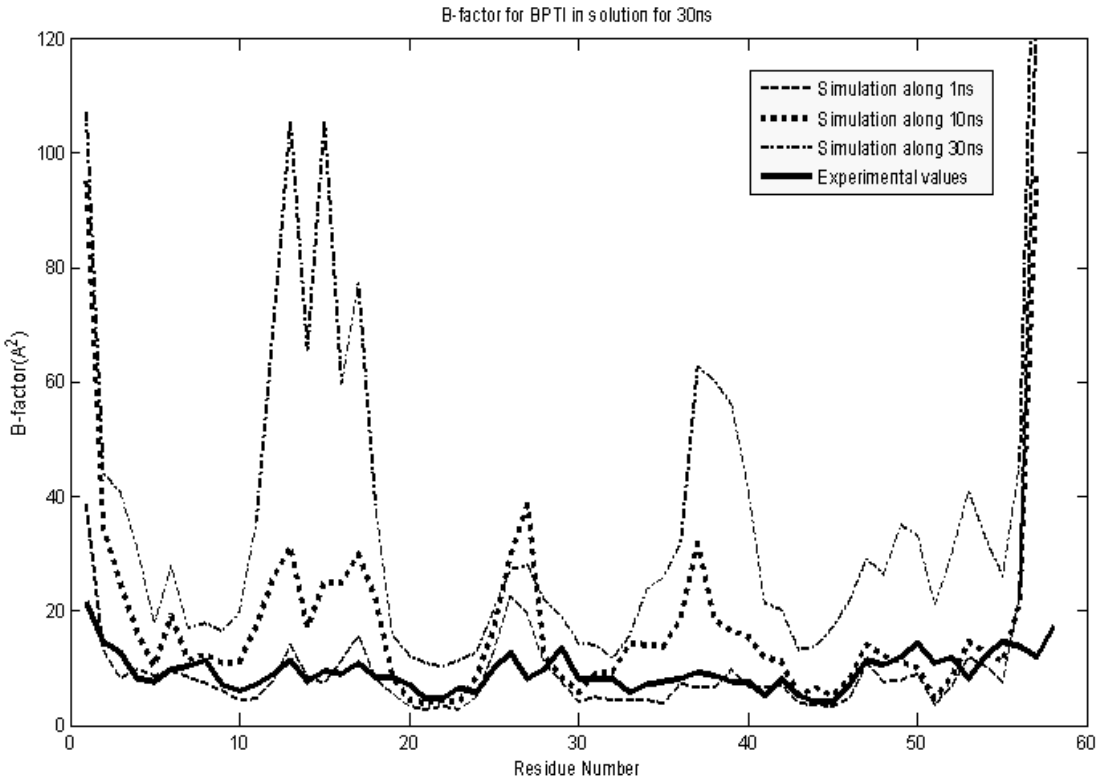

Figure 4.1 Simulated B-factor for the C_α atoms of BPTI compared with the B-factor obtained from crystallographic structure refinement based on X-ray data.

4.2 Root Mean Square Deviation (RMSD)

The changes in the conformations of the BPTI along the simulation time are analyzed by calculating the root mean square deviation (RMSD) of molecular dynamics structures with respect to the initial structure (the initial structure is chosen after 1ns of the simulation). As shown in Fig.4.2 the stability of the molecular dynamics simulation of BPTI along 30ns is achieved after 10ns. Beyond this point the RMSD value for BPTI in water simulation remains roughly arround 2.25 Å.

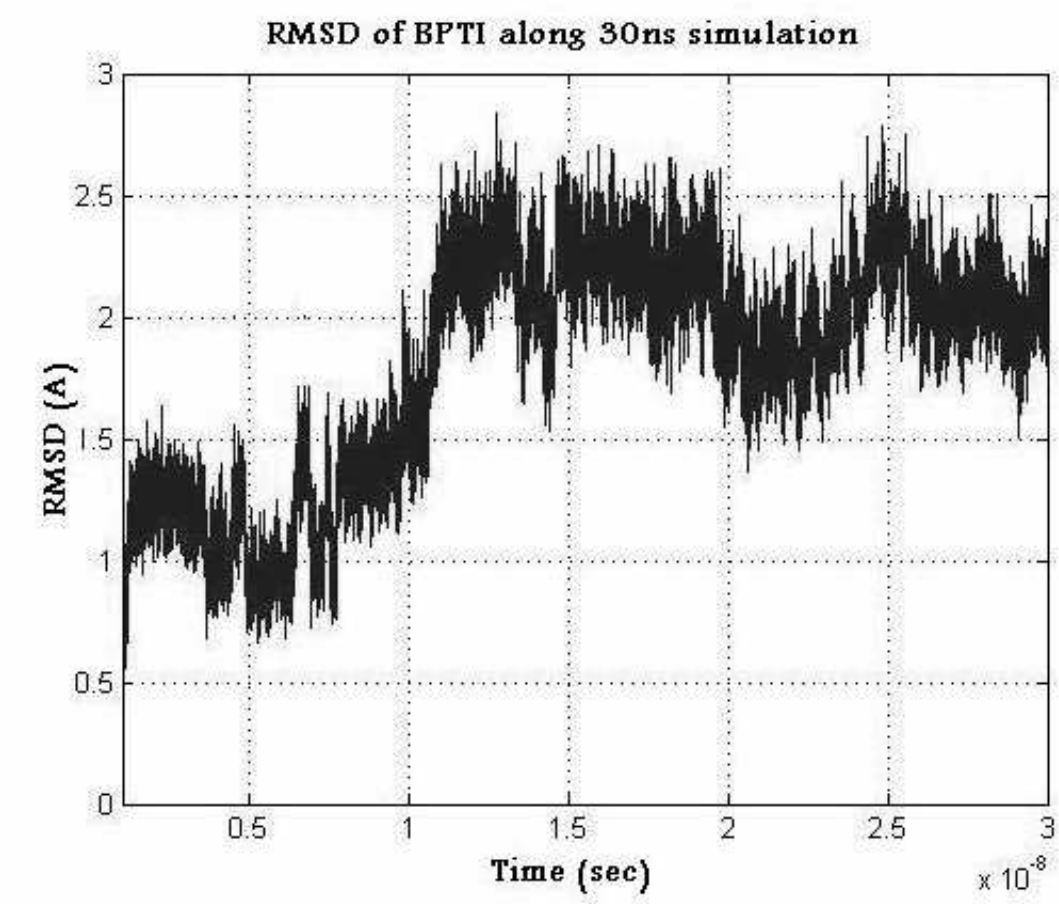

Figure 4.2 Root mean square deviation (RMSD) of the BPTI configurations with respect to the initial configuration, as a function of simulation time.

4.3 Convergence of Conformational Sampling

To study the convergence of conformational sampling for BPTI along 30ns MD simulation, the overlap between different fragments of the simulation trajectory are calculated. The trajectory of BPTI is divided into equal length fragments. The covariance matrix is calculated for each fragment. The similarity between these covariance matrices is calculated using the total overlap between protein fluctuations (equation (2.64)).

If the covariance overlap between different time windows of the simulation is close to 1 this will be a good indicator to predict the time dependence of the conformational sampling for BPTI[3]. If not, the total overlap is highly less than 1 (lack of convergence) and the conformational space is steadily throughout the simulation, this means the chosen time windows length is still smaller than time needed to obtain the convergence of the protein sampling.

The covariance overlap as shown in Fig.4.3-a is calculated between the first 100ps (reference fragment), and all other 100ps fragments along the 30ns simulation (300 fragments each one length is 100ps). The average value of the overlap between these fragments and the first one is about 50%. The average overlap for 500ps fragments (Fig.4.3-b) is approximately 50%. The overlap for a larger time fragments (1ns, 4ns and 10 ns) gives

a value similar to the 100ps time fragments overlap. For 1ns fragments (Fig.4.4-a) and 4ns fragments (Fig.4.4-b) the overlap is a little over 50%. The 10ns fragments show a similar behavior(Fig.4.5).

The overlap for different time windows (different trajectory fragments) using the covariance overlap method (total overlap) is about 50% throughout the simulation regardless of the time size of the fragments used to calculate the overlap. This means either the simulation time is much less than the time needed for convergence sampling for BPTI, or the method used to calculate the overlap between protein trajectory fragments is not suitable for measuring conformational sampling.

The covariance matrix of the trajectory coordinates contains the total information about the dynamics of the BPTI backbone atoms (C_α) along the simulation time. A large part of this information is about random motion (local fluctuations). This is not what we look for when trying to get a clear idea about convergence sampling for BPTI. The essential subspace, dealing with correlated motions, explaining the atomic motion (atomic fluctuations) is the interesting part. For this reason we need another method to calculate the overlap between different time windows of the simulation without considering this large amount of random motion in protein molecule.

The PCA technique is used to get the essential subspace which is respon-

sible for the majority of the atomic fluctuations. The eigenvectors with the large eigenvalues represent most of the atomic fluctuations. That means, the overlap between different time windows can be analyzed using subspace overlap method (equation 2.59) which is the inner square product between subspace eigenvectors. Only eigenvectors with the largest eigenvalues are included.

The overlap is calculated between the eigenvectors subspace of protein fragments, where each subspace contains 30 eigenvectors of high eigenvalues representing 86% of the protein fluctuations. The subspace overlap between protein fragments where each fragment length is 100ps is about 57% as shown in Fig.4.6-a. The subspace overlap increases with the length of the protein fragments as shown in Fig.4.6-b where each fragment is 500ps and the overlap is varies between 80% and 70%. The overlap varies between 85% and 70% for protein fragments of 1ns (Fig.4.7-a). The average subspace overlap is about 82% between protein fragments in which each fragment length is 4ns (Fig.4.7-b).

The overlap between protein fragments longer is more than 4ns decreases to vary between 80% and 60%. The overlap between 7ns long fragments is about 70% (Fig.4.8-a). This value drops to about 60% between 10ns long fragments. (Fig.4.8-b).

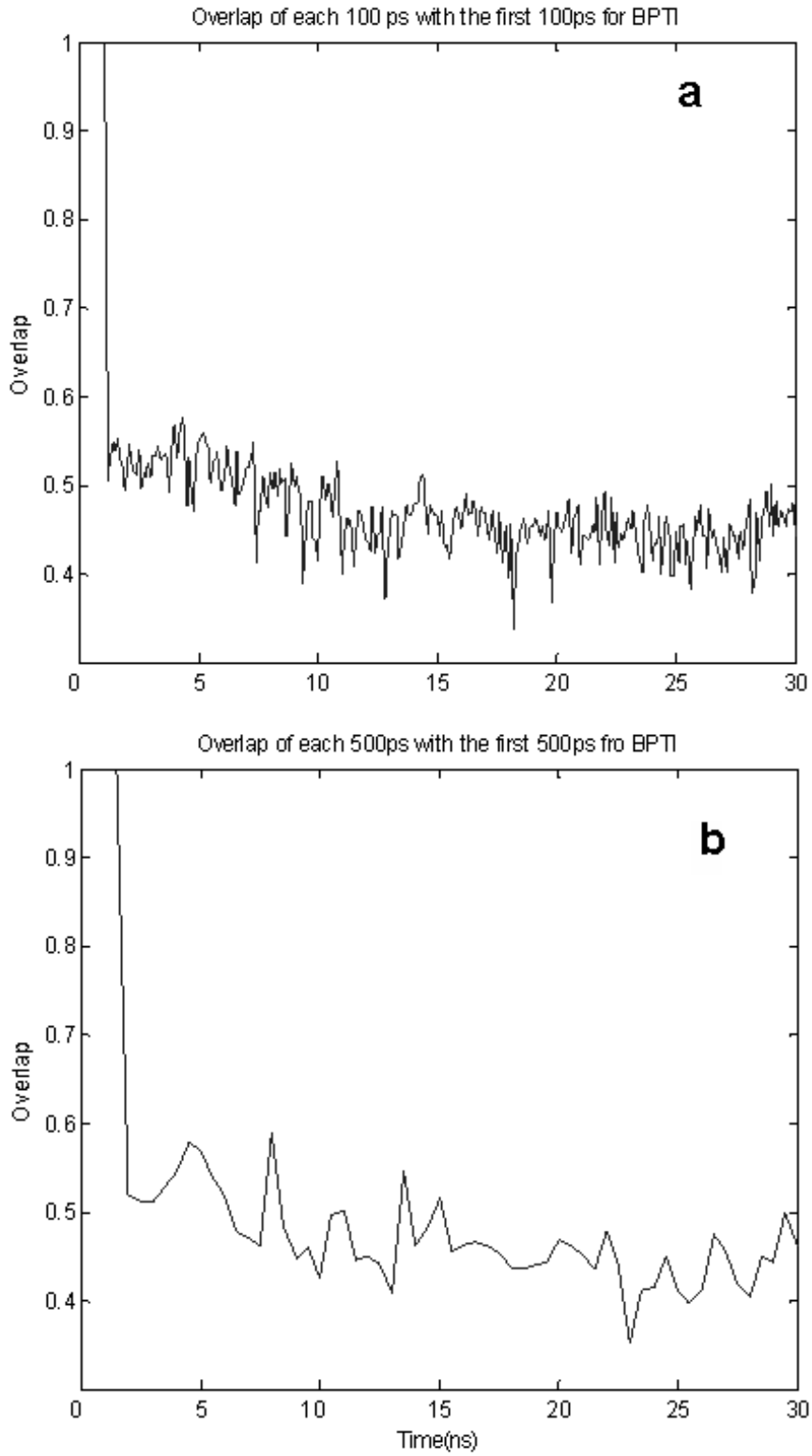

Figure 4.3 Calculated covariance overlap using equation2.64. (a) The overlap between different simulation fragments with 100ps length for each time fragment along the 30ns simulation. (b) Covariance overlap between 500ps time fragments.

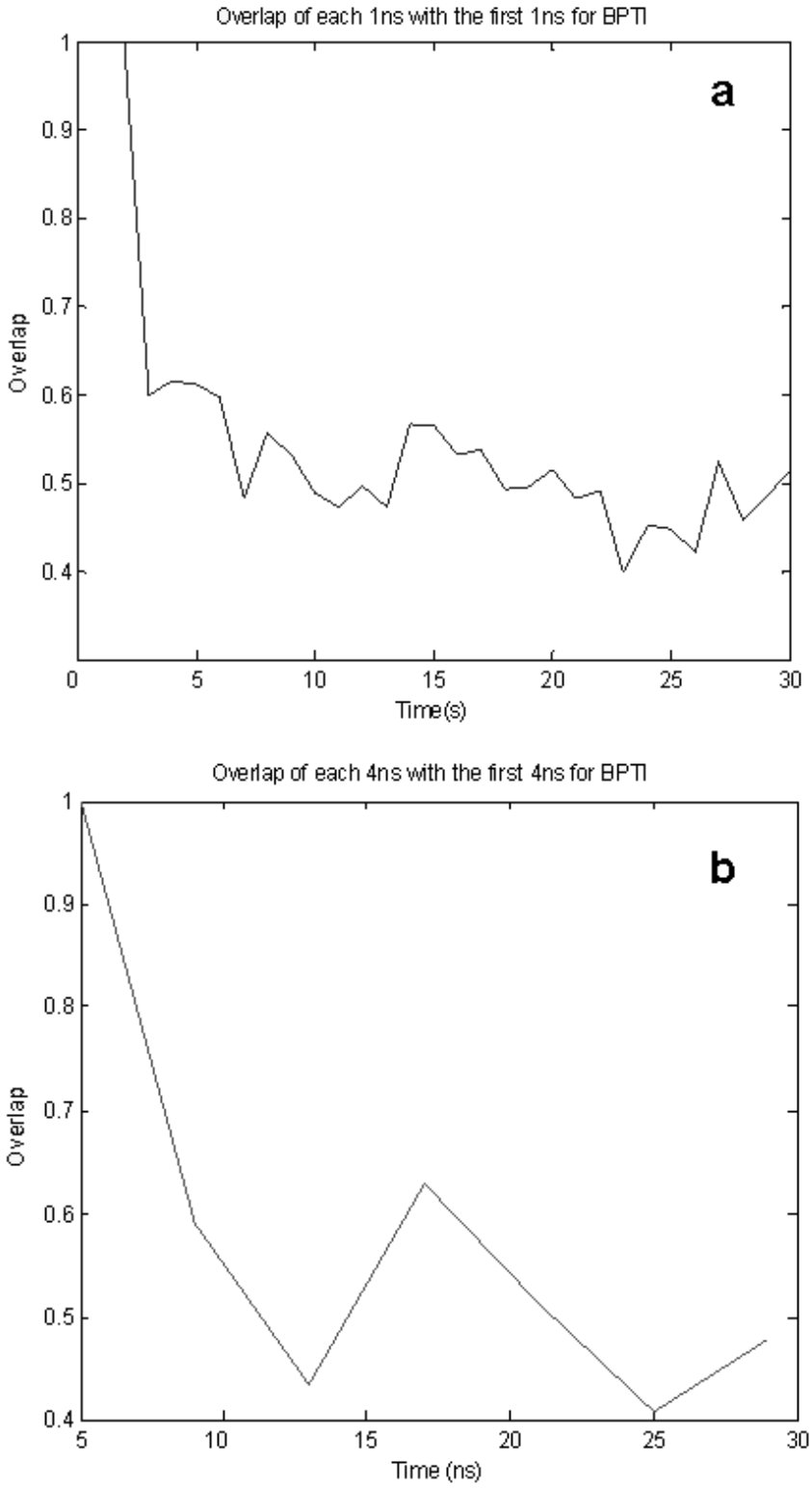

Figure 4.4 Calculated covariance overlap using equation2.64. (a) The overlap between different simulation fragments with 1ns length for each time fragment along the 30ns simulation. (b) Covariance overlap between 4ns time fragments.

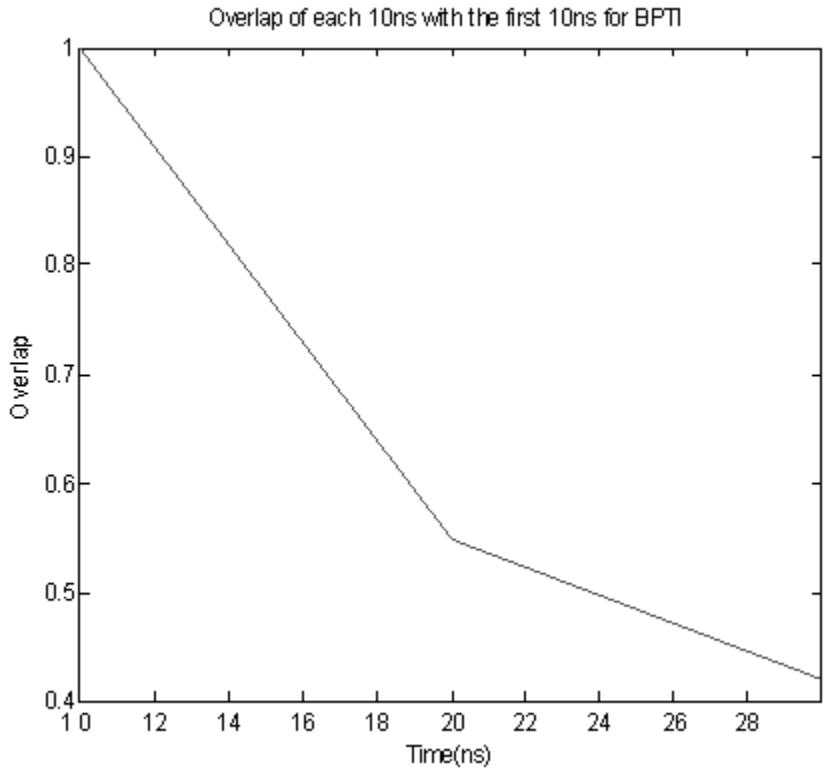

Figure 4.5 Calculated covariance overlap using equation2.64 between 10ns time windows along the 30ns simulation.

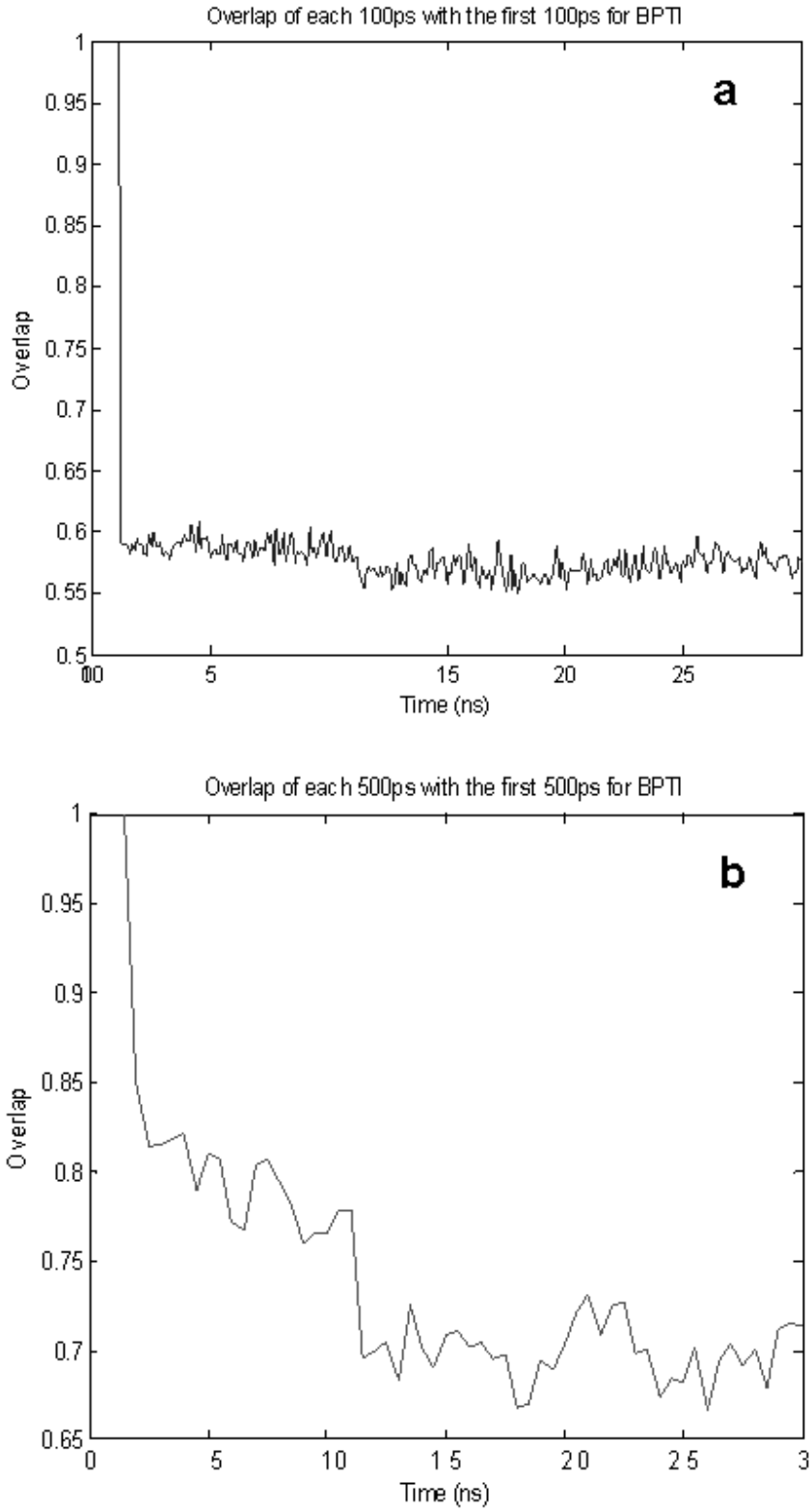

Figure 4.6 Calculated subspace overlap using equation2.59. (a) The overlap between 30 eigenvectors 100ps long fragments along the 30ns simulation. (b)Overlap between 30 eigenvectors from 500ps long fragments along the simulation time.

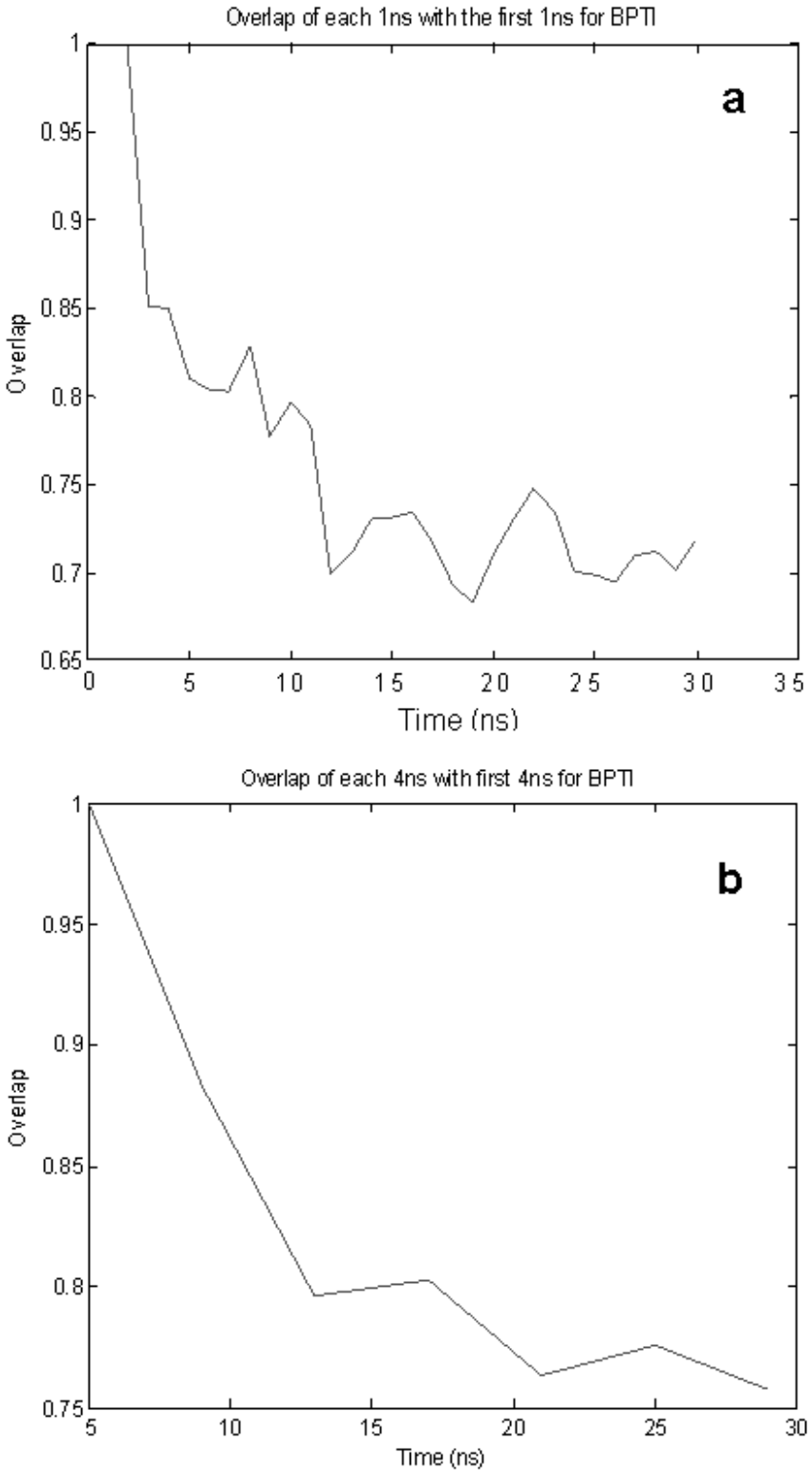

Figure 4.7 Calculated overlap using equation2.59. (a) The overlap between 30 eigenvectors from 1ns long fragments. (b) Overlap between 30 eigenvectors of 4ns long fragments.

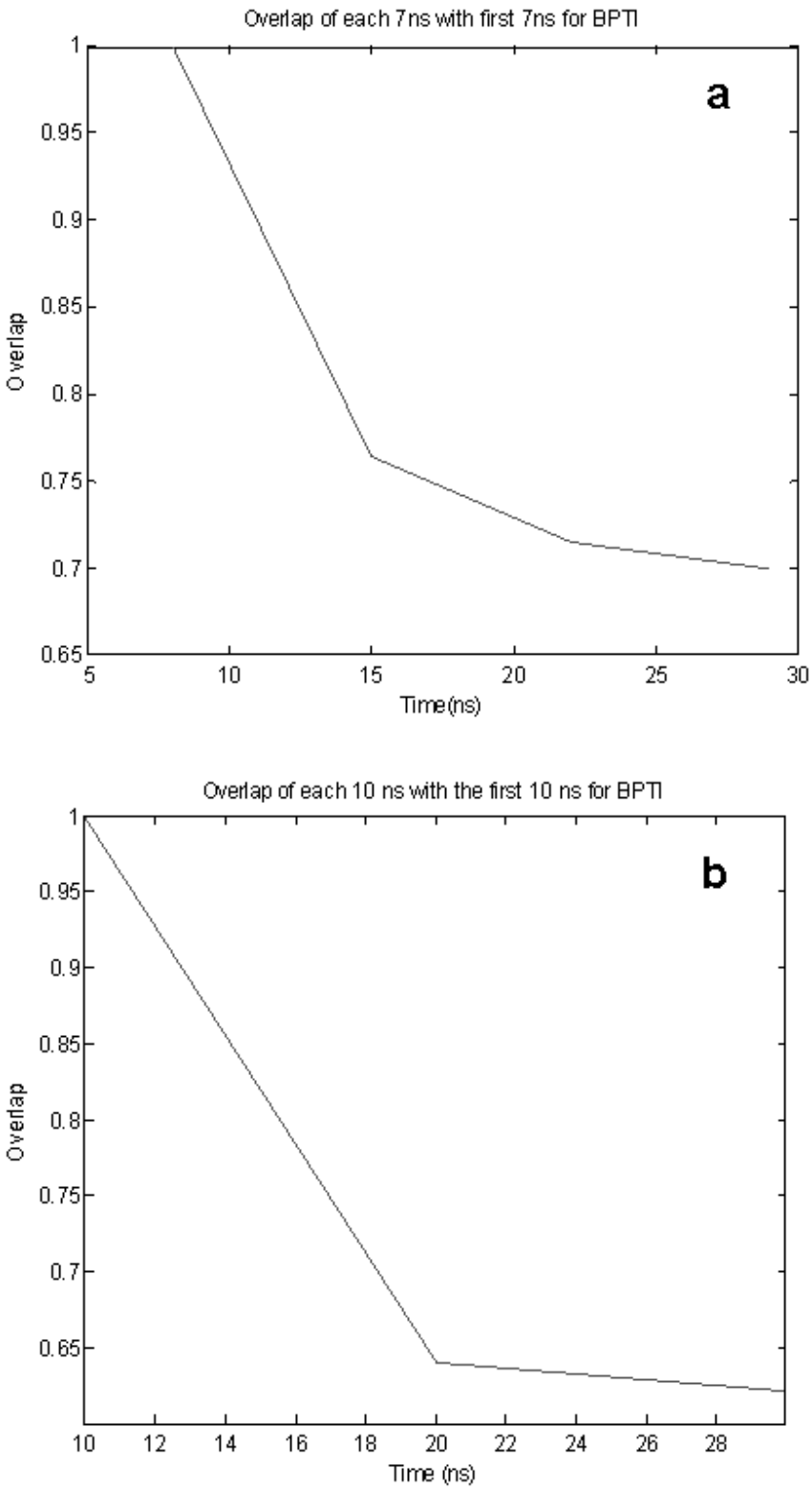

Figure 4.8 Calculated overlap using equation2.59. (a) The overlap between 30 eigenvectors of 7ns long fragments. (b) Overlap between 30 eigenvectors of each 10ns time fragments along the simulation.

Chapter 5
Discussion of Results and Conclusion

In this work, a 30ns long simulation for the BPTI protein was performed under the Linux environment using NAMD and VMD softwares. This is the longest simulation for this protein so far. The analysis for data trajectories were done using MATLAB program.

The first analysis on the data was calculating the B-factor and compared it values with the experimental crystallographical B-factor values which shows a qualitative agreement between them. This agreement validate our simulation results with respect to experimental results. Many other simulations made on BPTI shows similar results for B-factor[7, 54]. They obtained that B-factors calculated from the simulation (for BPTI and other proteins) are considerably larger than the experimental values (show Hunenberger[54] and AL-'Ajarmeh[7]). Further more, Molecular dynamics on protein simulations show only qualitative agreement for B-factor with experiment but not quantitatively[55].

The second analysis was calculating the RMSD values for atomic positions of C_α atoms. These converged to 2.25Å for BPTI in water.

Principal component analysis (PCA) of the C_α trajectories was used.

The overlap between different time windows of the simulation was calculated to study the convergence of conformational space for BPTI. Two methods were used to find the overlap. The first one finds the total overlap between covariance matrices for different fragments of the same time durations (100ps, 500ps, 1ns, 4ns and 10ns). This method was used by Hess[2] to measure the convergence of the protein (HPr and Ioyszyme) sampled space. He found that covariances have not converged after 14ns. The second method calculates the overlap between eigenvectors of high motion fluctuations (high eigenvalues percentage). Using this method to find the convergence of conformational sampling for membrane proteins along 10ns simulation also were not converged (overlap was about 55%.)[3].

In this work Eigenvector subspaces of different time length (100ps, 500ps, 1ns, 4ns, 7ns and 10ns) were analyzed. The results of calculating overlap using the two methods mention before are:

1. The total overlap between covariance matrices of different time windows of BPTI simulation gave a steady overlap value throughout the simulation. This is due to the inclusion of all motion,including the random motion.

2. The atomic fluctuations of C_α atoms only interested in studying convergence of conformational sampling (local fluctuations or random

motion are not interested). The PCA technique is used to get only the interested motion by calculating the eigenvectors of the data covariance matrix.

3. The overlap between eigenvectors with high eigenvalues which represent a high percentage of correlated atomic fluctuations was calculated for different time windows to see the convergence time BPTI along 30ns. The number of chosen eigenvectors is 30 that represent 86% to 90% of atomic fluctuations.

4. The value of overlap calculated using the second method shows an increases in overlap with the size of the time windows from 100ps to 4ns. Then the value of overlap decreases for time windows larger than 4ns.

5. The higher value of subspace overlap occurres when the length of time windows is 4ns. This means that similar atomic fluctuations for BPTI occurs along the simulation over the period of 4ns time.

6. The MD simulations made for BPTI with different time scales from hundreds of picoseconds[54] to 20ns[7] in addition to this work in which the simulation long is 30ns, shows that the positional fluctuations for BPTI are not yet completly converged.

7. According to the table 1.1 there are many molecular events that need a times up to seconds. Thus a longer simulation is needed to see the other types of motion.

Future work on studying protein dynamics should concern on long time simulations to get more insights about protein intramolecular motion. Development in computer hardware and software in the future as expected will offer an opportunity to do more long time simulations on protein.

Appendix A
Files Formats

A.1 The PDB file format

The Protein Data Bank (PDB) file format was created in the $1970's$ as a standard representation for macromolecular structure data derived from X-ray diffraction and NMR studies. The data contained in the PDB file include atomic coordinates, bibliographic citations, primary and secondary structure, information, and crystallographic structure factors and NMR experimental data. All details about this file format are available in the ***Protein Data Bank Contents Guide: Atomic Coordinate Entry Format Description Version 2.3, July 9, 1998*** .
(URL: `http://www.wwpdb.org/documentation/format2.3-0108-a4.pdf`)

The table below shows the PDB file format for the coordinate records (ATOM and HETATM):

An example of sample records for a PDB file format shown in fig.A.1.

```
SEQRES   1 A  471  5HP LYS ASP ALA ASN PHE ALA SER GLY ARG ASN SER ILE

MODRES 1CLV 5HP A    1  GLU  PYROGLUTAMIC ACID
HET    5HP  A   1       8

 HETNAM     5HP PYROGLUTAMIC ACID

FORMUL   1  5HP    C5 H7 N1 O3

HETATM    1  N   5HP A   1      29.020   7.713   8.323  1.00 17.69           N
HETATM    2  CA  5HP A   1      30.380   8.263   8.128  1.00 16.55           C
HETATM    3  C   5HP A   1      30.667   8.643   6.676  1.00 13.70           C
HETATM    4  O   5HP A   1      31.514   9.493   6.417  1.00 14.12           O
HETATM    5  CB  5HP A   1      31.390   7.193   8.612  1.00 16.19           C
HETATM    6  CG  5HP A   1      30.495   5.943   8.987  1.00 16.93           C
HETATM    7  CD  5HP A   1      29.101   6.476   8.787  1.00 19.39           C
HETATM    8  OD  5HP A   1      28.089   5.796   9.037  1.00 22.92           O
ATOM      9  N   LYS A   2      29.983   7.994   5.735  1.00 14.51           N
ATOM     10  CA  LYS A   2      30.178   8.269   4.313  1.00 13.28           C
ATOM     11  C   LYS A   2      28.999   8.963   3.640  1.00 16.12           C
ATOM     12  O   LYS A   2      29.027   9.224   2.435  1.00 17.54           O
ATOM     13  CB  LYS A   2      30.534   6.982   3.574  1.00 13.33           C
ATOM     14  CG  LYS A   2      31.829   6.365   4.059  1.00 14.70           C
ATOM     15  CD  LYS A   2      32.140   5.082   3.331  1.00 17.22           C
ATOM     16  CE  LYS A   2      33.340   4.422   3.957  1.00 17.71           C
ATOM     17  NZ  LYS A   2      33.629   3.104   3.340  1.00 20.50           N
```

Figure A.1 The PDB file format example

A.2 The PSF file format

The Protein Structure File (PSF) is generated by using CHARMM force field (topology file). The PSF file contains bonds, angle, dihedral, improper forces between atoms. The fig.A.2 show an example of a PSF file example for BPTI.

A.3 The DCD file format

The coordinate trajectories produced by NAMD are in binary format. Fortran data type of the trajectory DCD file is structured as in fig.A.3.

Where HDR = 'CORD' or 'VELD' for coordinates and velocities, respectively:

ICNTRL(1)=NFILE ! number of frames in trajectory.

fileICNTRL(2)=NPRIV ! number of steps in previous run.

ICNTRL(3)=NSAVC ! frequency of saving.

ICNTRL(4)=NSTEP ! total number of steps. NFILE=NSTEP/NSAVC.

ICNTRL(8)=NDEGF ! number of degrees of freedom.

ICNTRL(9)=NATOM-NFREAT ! number of fixed atoms.

ICNTRL(10)=DELTA4 ! coded time step.

ICNTRL(11)=stoi(XTLTYP,ALPHABET) ! coded crystallographic ! group (or zero).

ICNTRL(20)=VERNUM ! version number.

Field NO.	Column range	FORTRAN format	Description
1	1 - 6	A6	Record ID (eg. ATOM, HETATM)
2	7 - 11	I5	Atom serial number
-	12 - 12	1X	Blank
3	13 - 16	A4	Atom name (eg ” CA ” , ” ND1”)
4	17 - 17	A1	Alternative location code (if any)
5	18 - 20	A3	3-Letter amino acid code for residue
-	21 - 21	1X	Blank
6	22 - 22	A1	Chain identifier code
7	23 - 26	I4	Residue sequence number
8	27 - 27	A1	Insertion code (if any)
-	28 - 30	3X	Blank
9	31 - 38	F8.3	Atom’s x-coordinate
10	39 - 46	F8.3	Atom’s y-coordinate
11	47 - 54	F8.3	Atom’s z-coordinate
12	55 - 60	F6.2	Occupancy value for atom
13	61 - 66	F6.2	B-value (thermal factor)
-	67 - 67	1X	Blank
14	68 - 70	I3	Footnote number

Table A.1 The PDB format for coordinate records

```
PSF
     1 !NTITLE
 REMARKS original generated structure x-plor psf file

1101 !NATOM
    1 BPTI 1   ARG  N    NH3  -0.300000     14.0070      0
    2 BPTI 1   ARG  HT1  HC    0.330000      1.0080      0
    3 BPTI 1   ARG  HT2  HC    0.330000      1.0080      0
    4 BPTI 1   ARG  HT3  HC    0.330000      1.0080      0
    5 BPTI 1   ARG  CA   CT1   0.210000     12.0110      0
    ⋮
 1115 !NBOND: bonds
    1     5     2     1     3     1     4     1
    5     6     7     5     7     8     7     9
    ⋮
 675 !NTHETA: angles
    1     5     6     1     5    25     2     1     5
    2     1     4     2     1     3     3     1     5
    ⋮
2345 !NPHI: dihedrals
    1     5     7    10     1     5     7     8
    1     5     7     9     1     5    25    27
    ⋮
139 !NIMPHI: impropers
   18    19    22    16    25     5    27    26
   27    25    31    28    39    31    41    40
    ⋮
0 !NDON: donors
0 !NACC: acceptors
0 !NNB
```

Figure A.2 The PSF file format example for BPTI

```
 HDR    NSET   ISTRT  NSAVC  5-ZEROS NATOM-NFREAT   DELTA  9-ZEROS
`CORD' #files  step 1  step   zeroes  (zero)       timestep  (zeroes)
                                interval

 C*4    INT    INT    INT    5INT   INT         DOUBLE  9INT
==============================================================================
NTITLE            TITLE

INT (=2)          C*MAXTITL
                  (=32)
==============================================================================
NATOM   #atoms  INT
==============================================================================
X(I), I=1,NATOM                    (DOUBLE)  Y(I), I=1,NATOM  Z(I), I=1,NATOM
==============================================================================
```

Figure A.3 The DCD file format example

Appendix B
Configuration File

```
#namd configuration file for Running the simulation of PBTI in water.

# molecular system.

Structure /root/namd/NAMD_2.5_Linux-i686/water/ionized.psf

coordinates /root/namd/NAMD_2.5_Linux-i686/water/ionized.pdb

temperature 0

#force field

paraTypeCharmm on

parameters toppar/par_all22_prot.inp

parameters toppar/par_all27_prot_lipid.inp

restartname water_bpti.pdb

restartfreq 50

binaryrestart no

#output

outputenergies 50

outputtiming 50

outputMomenta 50

outputPressure 50
```

```
xstFreq 50

dcdFreq 50

wrapAll on

wrapNearest on

timestep 1

nonbondedFreq 2

stepspercycle 20

fullElectFrequency 2

#Approximations

switching on

switchdist 8.5

cutoff 10

pairlistdist 11.5

cellBasisVector1 50.336 0 0

cellBasisVector2 0 43.5166 0

cellBasisVector3 0 0 43.662

cellOrigin 9.546 4.504 4.056

margin 5

pme on

pmeGridsizeX 32
```

```
pmeGridsizeY 32

pmeGridsizeZ 64

#basic simulation

exclude scaled1-4

1-4scaling 0.4

#fix bpti atoms

fixedatoms on

fixedAtomsForces on

#bpti file fixed add water

fixedatomsfile /root/namd/NAMD_2.5_Linux-i686/water/fix.pdb

fixedAtomsCol B

langevinDamping 10

langevinTemp 300

langevinHydrogen no

langevinPiston on

langevinPistonTarget 1.01325

langevinPistonPeriod 200

langevinPistonDecay 100

langevinPistonTemp 300

useGroupPressure yes
```

```
useFlexibleCell yes

useConstantRatio yes

#output

binaryoutput no

outputname equil_out_wat_fix

#run one step to get into scripting mode

minimize 0

#langevinPiston turn off until later

langevinPiston off

#minimization

minimize 5000

output min_wat_fix

#heating

run 5000

output heat_fix

#equilibration 10ps

langevinPiston on

run 10000

output equil_fix

# min all atoms
```

```
langevinPiston off

fixedAtoms off

minimize 10000

output min_all_wat

#heat all

run 10000

output heat_all_wat

#equilibration 40ps

langevinPiston on

run 40000

output equil_all_wat

langevinPiston off

langevin off

run 30000000

output endsim
```

Appendix C
M-Files Used to Analysis Data trajectory

C.1 Calculation of the Covariance Matrix

```
%This M-file used to calculate the covariance matrix co of the

%data matrix A (trajectory) which its size is r*c.

function co =covariance(A,r,c)

co =zeros(c,c);

%%%%%%%%%%%%%%%%% Columns Averages %%%%%%%%%%%%%%

avg= zeros(1,c);

for i = 1:c

      for j = 1:r

          avg (1,i)= avg(1,i)+A(j,i);

     end

          avg(1,i)=avg(1,i)/r;

end

%%%%%%%%%%%%%%%%% Variance Calculations %%%%%%%%%%%%%%

for i =1:c

      for j = 1:r

          A(j,i)=A(j,i)-avg(1,i);

      end
```

```
end
%%%%%%%%%%%%%%%% Dot Products %%%%%%%%%%%%%%
for i=1:c
    for k=1:c
        for j = 1:r
            co(i,k)=co(i,k)+(A(j,i)*A(j,k));
        end
    end
end
```

C.2 The First Method to Calculate the Overlap

```
%This M-file calculate the total overlap S between two covariance
% matrices produced from the MD trajectory for BPTI.
function meth1=overlap(xyz1,xyz2)
% xyz1 is the data matrix (trajectory) for first interval of
%simulation time.
% xyz2 is the other data matrix for another interval of
%simulation time.
A=cov(xyz1);
B=cov(xyz2);
[Ra,Ea] = eig(A);
```

[Rb,Eb]= eig(B);

$c = Ra * ((Ea).^0.5) * Ra'$;

$d = Rb * ((Eb).^0.5) * Rb'$;

$s = 1 - (sqrt(trace((c-d)^2))/sqrt(trace(A) + trace(B)))$;

C.3 The Second Method to Calculate the Overlap

% This M-file calculate The overlap between two sets of n

%orthonormal vectors (eigenvectors).

function meth2=overlap(A,B,n)

% A is the first eigenvectors subset (v_1,....,v_n).

% B is the second eigenvectors subset (w_2,....,w_n).

% n is the number of eigenvectors.

meth2=zeros(n,n);

for i =1:n;

    for r = 1:n;

        for j = 1:n;

            test(i,r) = $test(i,r) + A(j,i) * B(j,r)$;

        end

            test(i,r)=$(test(i,r))^2$;

    end

end

ovlp =sum(sum(meth2))/n

C.4 M-file Used to Calculate the B-factor

% This M-file Using to calculate the B-factor from covariance matrix.

function B=factor(co);

% co is the covariance matrix.

% B is a produced array contains the b-factor value for each atom.

A=diag(co);

B= zeros(57,1);

for i= 1:3:169;

$B(i) = sum(A(i) + A(i+1) + A(i+2)) * 8 * pi^2/3;$

end

References

1. CHARMM Principles. Accelrys, Inc. September 18, 1998. (URL: `http://www-bio.unizh.ch/docu/acc_docs/doc/charmm_principles/Ch04_mol_dyn.FM5.html#679031`

2. Berk Hess. Convergence of Sampling in Protein Simulation. ***Physical Review E***, Volume 65, 031910. 2002.

3. José D. Faraldo-Gómez, Lucy R. Forrest, Marc Baaden, Peter J. Bond, Carmen Domene, George Patargias, Jonathan Cuthbertson and Mark S.P. Sansom. Conformational Sampling and Dynamics of Membrane Proteins From 10-Nanosecond Computer Simulations. ***PROTEINS: Structure, Function, and Bioinformatics*** (2004) 57: pp. 783791.

4. Helmut Grubmüller. Proteins as Molecular Machines: Force Probe Simulations. John von Neumann Institute for Computing, Jülich, NIC Series, Vol. 23, pp. 401-422, 2004. (URL: `http://www.fz-juelich.de/nic-series/volume23`).

5. Sugato Basu. Data Structures for a Mini-Threding Algorithm for Protein Structure Predction. Master Thesis, Unvrsity of California (Santa Cruz), 2000.

6. B.L. de Groot. Native state protein dynamics : a theoretical approach, Doctoral Thesis, Groningen University, (1999).

7. Basem AL-'Ajarmeh. Comparison of Residue Motion Correlation in BPTI Using Vacum and Solvent Molecular Dynamics Simulations. Master thesis, Birzeit University, Palestine, 2005.

8. Markus Dittrich, Chalermpol Kanchanawarin. Case Study: BPTI. (URL:`http://www.ks.uiuc.edu/Training/CaseStudies/pdfs/bpti.pdf`).

9. Ascenzi P. Bocedi A. Bolognesi M. Spallarossa A. Coletta M. De Cristofaro R. Menegatti E. The bovine basic pancreatic trypsin inhibitor (Kunitz inhibitor): a milestone protein. ***Curr. Protein Peptide Sci.*** 4, 231-251. (2003).

10. Roland Stote, Annick Dejaegere, Dmitry Kuznetsov, Laurent Falquet. Molecualr Dynamics Simulation Tutorial. October 26, 1999, version 1.0. (URL:http://www.ch.embnet.org/MD_tutorial).

11. D. C. Rapaport. The art of molecular dynamics simulation .Cambridge, U.K.: Cambridge University Press, 2004. p3-4.

12. Michael P. Allen. Introduction to Molecular Dynamics Simulation. John von Neumann Institute for Computing, Jülich, NIC Series, Vol. 23, pp. 1-28, 2004.

13. Volker Eyrich , Tatyana Polenova, and Angelo Rossi. Conformational Search: M.D. M.C. and S.D. Columbia University - Spring 2002 (URL:http://mcdermott.chem.columbia.edu/biophys_2002/search /backgroundII.html).

14. J. Woller. The Basics of Monte Carlo Simulations. (1996). (URL:http://www.chem.unl.edu/zeng/joy/mclab/mcintro.html).

15. Peter J. Steinbach. Introduction to Macromolecular Simulation. ***Biophysics Textbook Online*** (BTOL), (2003). (URL:http://www.biophysics.org/education/steinbach.pdf)

16. L. Ellis, K. Chow. Protein Structure Prediction From Primary Sequence. ***Biophysics Textbook Online*** (BTOL), (1998). (URL:http://www.biophysics.org/education/ellis.pdf)

17. F. Suits, M. C. Pitman, J. W. Pitera, W. C. Swope, R. S. Germain. Overview of molecular dynamics techniques and early scientific results from the Blue Gene project.***IBM Journal of Research and Devlopment***, Blue Gene, Volume 49, Number 2/3, (2005).

18. Doucet, Jean-Pierre and Jacques, Webber. Computer-Aided Molecular Design. Academic press inc, U.S.San Diego, CA 92101, p127.

19. Slobodan Danko Bosanac. Dynamics of Particles and the Electromganetic Field. World Scientific Series in Contemporary Chemical Physics - Vol. 24. Sept 2005.

20. Daniel A. Beard. A Molecular Modelers Guide to Statistical Mechanics. ***Biophysics Textbook Online*** (BTOL), (2001). (URL:http://www.biophysics.org/education/beard.pdf)

21. J. M. Haile. Molecular dynamics simulation : elementary methods. New York: Wiley,1997.

22. Boris Veytsman and Michael Kotelyanskii. Molecular Dynamics I: Numerical Integration. Finite Difference methods. November 17, 1997. (http://www.plmsc.psu.edu/ www/matsc597c-1997/simulations/ Lecture5/)

23. Alex S. Ct, Bill Smith and Philip J. D. Lindan, A Molecular Dynamics tutorial: Integration algorithms.Daresbury Laboratory 2001. (http://www.compsoc.man.ac.uk/ lucky/Democritus/Theory/ verlet.html)

24. Igor P. Omelyan, Numerical integration of the equations of motion for rigid polyatomics: The matrix method. Institute for Condensed Matter Physics, National Ukrainian Academy of Sciences, 1 Svientsitsky St., UA-290011 Lviv, Ukraine. 17 Jan 1999. (URL: arxiv.org/pdf/physics/9901026)

25. Bryan Van Der Ende. Molecular Dynamics Simulations of Wild-Type and Mutant Neu Transmembrane Domains. Master Thesis, University of Guelph. December, 1999.

26. Francis W. Sears and Gerahard L. Salinger. Thermodynamics, Kinetic Theory, and statistical Thermodynamics, Third edition. Addison-Wesley Publishing Company.p302 and p307.

27. William Graham. Computational statistical mechanics. Amsterdam: Elsevier, 1991. p1-3.

28. Daniel P. Stevens, Eric Renner and Roberto Gomperts. Nanotechnology Research: Enabling Technologies for Innovative New Materials. Principal Scientist, SGI; Milan Mehta, Intel. May 2005. (URL:http://www.jrti.com/pdf/adv_matsci.pdf)

29. National Institutes of Health (NIH), Center for Molecular Modeling(CMM). Molecular Mechanics. (URL: http://cmm.cit.nih.gov/modeling/guide_documents/ molecular_mechanics_document.html)

30. Kim Baldridge, Kim's Research Group. Molecular Mechanics. San Diego Supercomputer Center. (URL: http://www.sdsc.edu/ kimb/eff.html)

31. David Curcó , Francisco Rodriguez-Ropero and Carlos Alemán. Force-field parametrization of retroinverso modified residues: Development of torsional and electrostatic parameters. ***Journal of Computer-Aided Molecular Design*** (2006) 20: pp. 1325.

32. Thomas W. Shattuck, Colby College Molecular Mechanics Tutorial. MOE Version, December 2004.
(URL: `http://www.colby.edu/chemistry/CompChem/MMtutor.pdf`
)

33. David A.C. Beck and Valerie Daggett, Methods for molecular dynamics simulations of protein folding/unfolding in solution. ***ELSEVER, Methods*** (2004) 34: pp. 112120.

34. Lindsay I Smith. A tutorial on Principal Components Analysis. February 26, 2002.
(URL:
`http://csnet.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf`)

35. Janne T.A. Saarela , Kari Tuppurainen, Mikael Peräkylä, Harri Santa, Reino Laatikainen. Correlative motions and memory effects in molecular dynamics simulations of molecules: principal components and rescaled range analysis suggest that the motions of native BPTI are more correlated than those of its mutants. ***Biophysical Chemistry*** (2002) 95: pp. 4957.

36. Theoretical and Computational Biophysics Group, University of Illinois and Beckman Institute. VMD User's Guide, Version 1.8.5 August 21, 2006.
(URL:`http://www.ks.uiuc.edu/Research/vmd/`)

37. Zhiyong Zhang, Yongjin Zhu, Yunyu Shi, Molecular dynamics simulations of urea and thermal-induced denaturation of S-peptide analogue. ***Biophysical Chemistry*** (2001) 89: pp. 145-162.

38. Jonathon Shlens, A Tutorial on Principal Component Analysis. December 10, 2005; Version 2.
(URL:`http://www.cs.cmu.edu/ elaw/papers/pca.pdf`)

39. Paual R. Harasti, Roland List, Principal Component Analysis of Doppler Radar Data. Part I: Geometric Connections between Eigen-

vectors and the Core Region of Atmospheric Vortices. ***Journal of atmospheric sciences*** (2005) 26: pp. 4027-4042.

40. Lee-Ing Tong, Chung-HoWang and Hung-Cheng Chen. Optimization of multiple responses using principal component analysis and technique for order preference by similarity to ideal solution. ***Int J Adv Manuf Technol*** (2005) 27: pp. 407414.

41. Caterina Arcangeli, Anna Rita Bizzarri, Salvatore Cannistraro. Concerted motions in copper plastocyanin and azurin: an essential dynamics study. ***Biophysical Chemistry*** (2001) 90: pp. 45-56.

42. G. Vriend, R.W.W Hooft and D. van Aalten. ESSENTIAL DYNAMICS ANALYSIS (ESSDYN). (URL:`http://www.csb.yale.edu/userguides/graphics/whatif/html/chap75.html`)

43. Torsten Becker, Jennifer A. Hayward, John L. Finney, Roy M. Daniel, and Jeremy C. Smith. Neutron Frequency Windows and the Protein Dynamical Transition. ***Biophysical Journal*** (2004) 87: pp. 14361444.

44. Johannes Schmidt-Ehrenberg, Daniel Baum and Hans-Christian Hege. Visualizing Dynamic Molecular Conformations. ***IEEE Visualization*** 2002 Oct. 27 - Nov. 1, 2002, Boston, MA, USA.

45. Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. ***Acta Crystallographica*** A,(1976) 32: pp. 922923.

46. David Parker. A Survey of Classification Methods for Protein Domain Motions and Flexibility. Biochemistry 218 Final Project, Department of Mechanical Engineering Stanford University, Stanford Ca. March 10, 2003. p7.

47. Caterina Arcangeli, Anna Rita Bizzarri, Salvatore Cannistraro. Molecular dynamics simulation and essential dynamics study of mutated plastocyanin: structural, dynamical and functional effects of a disulfide bridge insertion at the protein surface. ***Biophysical Chemistry*** (2001) 92: pp. 183-199.

48. M. Bhandarkar, R. Brunner, C. Chipot, A. Dalke, S. Dixit, P. Grayson, J. Gullingsrud, A. Gursoy, D. Hardy, J. Henin, W. Humphrey, D. Hurwitz, N. Krawetz, S. Kumar, M. Nelson, J. Ph. NAMD Users Guide. Version 2.6, August 30, 2006.

49. James Phillips, Gengbin Zheng, Laxmikant Kalé. NAMD: Biomolecular Simulation on Thousands of Processors.
(URL:
http://www.psc.edu/training/scaling/James_Phillips_namd_v1.pdf)

50. Theoretical and Computational Biophysics Group, University of Illinois at Urban-Champagin. CatDCD - Concatenate DCD files. 20 Mar 2006.
(URL: http://www.ks.uiuc.edu/Development/MDTools/catdcd/)

51. Jim Phillips. FlipDCD - DCD file endianism converter. 20 Mar 2006.
(URL: http://www.ks.uiuc.edu/Development/MDTools/flipdcd/)

52. Mark S. Gockenbach. A Practical Introduction to Matlab (Updated for Matlab 5). Sep 8, 1999.
(URL: http://www.math.mtu.edu/ msgocken/intro/intro.html)

53. H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne: The Protein Data Bank. Nucleic Acids Research, 28 pp. 235-242 (2000).
(URL: http://www.pdb.org/.)

54. P.H. Hünenberger, A.E.Mark and W.F van Gunsteren. Fluctuation and Cross-correlation Analysis of Protein Motions Observed in Nanosecond Molecular Dynamics simulations. ***J.Mol.Biol.*** (1995)252, pp. 492-503.

55. Kil Ho Eom. The Mechanical Modeling of Proteins. PhD Thesis, The University of Texas at Austin. 2005.