Computational Mechanics Analysis of the Hidden Conformational Dynamics within a Molecular Trajectory

LARRY ENIWEI GODWIN

Masters by Research in Mathematics of Complex Systems

ASTON UNIVERSITY

October, 2013

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without proper acknowledgement.

ASTON UNIVERSITY

Computational Mechanics Analysis of the Hidden Conformational Dynamics within a Molecular Trajectory

LARRY ENIWEI GODWIN

Masters by Research in Mathematics of Complex Systems, 2013

Thesis Summary

In this research, the conformational dynamics of a peptide, VPAL, is investigated. Molecular Dynamic Simulation (MD) and Computational Mechanics (CM) methods are both applied to study this dynamic behaviour of VPAL. MD was used to simulate the microscopic behaviour, while the analysis was done using CM. From the analysis and the result it is observed that, at certain time scale (140ps) the conformational dynamics of VPAL dynamics is Markovian, but as the time step reduces ($\triangle t^s$ < 140ps) it becomes Non-Markovian, which is clearly described by the ϵ-machine constructed from the dynamics at different $\triangle t^s$.

Keywords: Protein Folding, Peptide Dynamics, Conformational Dynamics of Proteins/Peptides, Molecular Dynamic Simulation (MD), Computational Mechanics (CM), Causal States, Metastable States, GROMACS, Verlet Algorithm, ϵ-machine, Biomolecular Simulation, Microscopic Interactions and Trajectories.

Copyright ©2013, Larry E. Godwin, All Rights Reserved.

ABSTRACT

In this research, the conformational dynamics of a peptide, VPAL, is investigated. Molecular Dynamic Simulation (MD) and Computational Mechanics (CM) methods are both applied to study this dynamic behaviour of VPAL. MD was used to simulate the microscopic behaviour, while the analysis was done using CM. From the analysis and the result it is observed that, at certain time scale (140ps) the conformational dynamics of VPAL dynamics is Markovian, but as the time step reduces ($\triangle t^s < 140$ps) it becomes Non-Markovian, which is clearly described by the ϵ-machine constructed from the dynamics at different $\triangle t^s$.

ACKNOWLEDGEMENTS

Firstly, I would like to thank Prof. Emmanuel for always been there for me, all the time. I would like to thank my supervisor Dr. Dmitry Nerukh, for his time, efforts, guide, patience, understanding, humility and encouragement with all the academic rigour; and with his constant assistance to the actualization of this work

.

Also, my appreciation goes to the director of the Non-Linearity Complex Research Group (NLCRG), Dr. Amit K. Chattopadhyay for his clever decisions over issues and helpful academic assistances. My appreciation goes to all my MSc Lecturers and Staff for always been available in times of need. More so, I would want to thank all my friends in NLCRG and fellow MRes/Phd students, Chloe, Nerupa, Jack, Lizi, Iain, Xi and all others, for their encouragement, amusement, kindness, care and attitude towards me.

I would like to thank the National Information Technology Development Agency (NITDA) for her financial sponsorship. In addition, I appreciate the endless effort of the 2012/2013 NITDA Scholars, to making sure that we receive our allowances.

I am grateful to all my friends, colleagues, and the brethren in Christ, who have always been helpful when needed.

Finally, my gracious appreciation to my precious family for their love and care till this day. Special thanks to my mum for her wondrous love and financial assistances.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

Symbols	Meaning
CM	Computational Mechanics
MD	Molecular Dynamics
GROMACS	Molelcular Dynamic Software Package
PBC	Periodic Boundary Condition
VPAL	Valine-Proline-Alanine-Leucine is a peptide molecule

LIST OF SYMBOLS

Symbols	Meaning
T	Temperature
m or M	Mass of Particle
ps	Picoseconds
μs	Microseconds
$\overleftarrow{s}$	Past or history string sequence
$\overrightarrow{s}$	Future string sequence
S	Set of all causal states
$\mathcal{S}_i$	The i^{th} causal state
$P(A\|B)$	The conditional probability of A given B
A	Alphabet Set
A_N	Total number of Alphabet symbols
N	Total number of symbols in a sequence of strings
Λ	Minimum acceptable history of words in a given sequence of strings
L	Maximum acceptable history of words in a given sequence of strings
s^L	Set containing all possible histories
$\overleftarrow{s}^L_t$	Subset history of the set of string histories, s^L, at time t
C	Carbon
C_α or M	The alpha carbon atom of the dihedral backbone of the polypeptide
H	Hydrogen
N	Nitrogen
O	Oxygen
$X^{(-\|+\|nill)}$	A negative, positive or stable state atom
ϕ	The ϕ Dihedral Angle
ψ	The ψ Dihedral Angle
k_B	Boltzmann Constant
h	Plank Constant
λ	De Broglie Thermal Wavelength

t^s	Time Step for conformational dynamics of the simulation

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

It is fascinating how the study of an atom lead to different discoveries in the 20th century until this day. Harnessing the power within the atom has led to many discoveries like nuclear power plants, new drugs for various kind of illness, etc [1]. Atoms are considered to be the smallest unit of an element (Hydrogen, Nitrogen, Oxygen, etc.) that preserve its chemical properties [2]. Hence, a thorough understanding of the structure and properties of an atom is a way by which we can understand how the objects (liquid, solid and gas) around us behave.

Molecules are made up of one or more atoms and are the smallest units of a compound (protein, carbohydrate, alcohol, etc.). Therefore, in other to better describe/model how these compounds behave in various environments, reactions and composition, we need to first understand how they are formed in their lowest state. In addition, we need to analyse how changes in the microscopic level affects their formation, reactions and state. For a proper analysis, one has to critically examine the atoms that make up the molecules of these compounds and how they behave given certain criterion or constrains.

1.1 Background

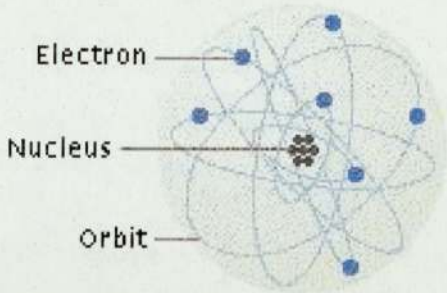


Figure 1.1: A Simple Model of the Atom.

Figure (1.1) shows the basic structural representation of an atom. It shows how the electrons revolve round a nucleus (contains the protons and neutrons) in its orbital. With this structure in mind, we can say an atom is smallest particle of matter (anything that occupy space and has mass). Knowing this, we can consider their behaviour to be described by physical laws which are applicable to matter. Newtons Second Law of Motion is one of such laws (in the domain of Classical Mechanics) that is used to describe the motion of particles which can be approximated classically. However, atoms are supposed to be treated as Quantum particles because of their sizes. But this depends on De Broglie Thermal Wavelenght (λ) [3] which classify atoms to be treated as classical or quantum depending on λ as it relates to l (the mean nearest neighbour separation value of the atoms).

$$\lambda = \sqrt{\frac{2\pi h^2}{Mk_BT}}. \tag{1.1}$$

For atoms to be treated as classical, then $\lambda << l$ (i.e λ should be far less than l). Now this can be true if the atom has a large mass (M) or the temperature (T) is high in equation (1.1). In addition, particles which are treated as quantum

generally are considered at a low temperature. Hence its acceptable to apply classical theories on certain types of atoms.

The Second Law of Motion [4, 5] is used in determining the trajectories of the atoms during Molecular Dynamic Simulation (MD) [6]. The law states that, *the total force* $\mathbf{F}$ *acting on a body is equal to the rate of change of its linear momentum* $\mathbf{p}$ *in an initial frame of reference.*

$$\mathbf{F} = \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} \tag{1.2}$$

But, $\mathbf{p} = m\dot{\mathbf{r}} = m\frac{\mathrm{d}x}{\mathrm{d}t}$. Where m and r are the constant mass and position vector of the body respectively. Therefore, equation 1.0 becomes:

$$\mathbf{F} = m\frac{\mathrm{d}^2\mathbf{r}}{\mathrm{d}t}, \tag{1.3}$$

$$= m\ddot{\mathbf{r}}. \tag{1.4}$$

For solution to N-particles problem, the equation(1.4) is expressed as follows:

$$\mathbf{F} = \sum_{i=1}^{N} m_i\ddot{\mathbf{r}}_i, \tag{1.5}$$

$$\mathbf{F}_i = m_i\ddot{\mathbf{r}}_i. \tag{1.6}$$

Equation(1.6) is further explained in Chapter 3.2.1, with is algorithmic implementation discussed in detail as regards to MD. *A computer simulation that is done on atoms of molecules to study their behaviour and interactions at different times, with the assumption that they can be classically approximated is known as MD simulation* [7, 8].

In this research, two major methods, MD and Computational Mechanics(CM) [9] where applied in simulating numerically, and then analysing the trajectories of the atoms respectively. The notion of CM is to understand the basic building blocks of any system by means of some state space representation models. The

patterns of the conformational dynamics of VPAL is investigated using CM. This is done in order to create a model that will reconstruct the conformational changes at different times.

The significance of this project is to intuitively understand some fundamental observation of the folding process in proteins which might lead to other findings that will efficiently explain the folding process, a reliable and efficient drug design or other microbiological substance. The following sections discuss further on the specifics of what this project is investigating.

1.2 Motivation

Proteins are self-organize, that is their conformations are polymorphic in nature depending on the structure of their 3-dimensional formation. The notion of self-organization in dynamic systems can be a bottleneck if theirs is no proper averaging process on how the patterns observed in the folding can be extracted [10]. therefore, it is needful to design a method which properly splits the dynamics and also estimate correctly this self-organizing nature of the protein conformation. Though dihedral angles are used in a way to describe the 3-Dimensional structure of proteins, yet there is no clear approach on how the different dynamic states could explicitly be separated to depict the different observed states or clusters for a Markov Model. This situation has been addressed [10, 11] and was adopted in this research, yet how does this separation of states describe the conformational changes or reconstruct the self-organizing behaviour of proteins has not been extensively validated. Hence, this research further investigates this behaviour as a follow-up to a previous research [12] which addresses this problem.

1.3 Statement of The Problem

The problem of Protein Folding is still unsolved by the research community. The question of how, why, and what makes it fold, is still progressively been considered

widely in the world today as an important research. However, tremendous efforts have been made to unfold the mystery behind it, but few has shown remarkable results [13]. The question of whether the solvent has an effect on its solute (i.e. peptide) as regards to the folding process and how it affects it is still a problem and many more. In fact, its a difficult problem to even analyse due to its separate chemistry and physics aspects of the protein as regards to its chemical reaction (i.e. as an organic compound); and its physics (e.g. its free energy, atomic motion and configurations).

Before this problem can be addressed as a whole, some related problems should be considered, which led to this research. For simplicity we shall state the problem in words. In this research we address the following question: *How can the trajectories of the atoms of peptide dynamics describe the different conformational states of peptides conformations, how probable are the transitions between these states, and how can the entire configuration changes be reconstructed from this observed transitional probabilistic changes.*

To address this problem we applied two different methods, MD and CM. The trajectories of the VPAL peptide were generated by MD, and analysed using CM. More details of this is explained in Chapter 3, 4 and 5 of this report.

1.4 The Scope of Study

In this project, we analyse the microstates observed from the Ramachandran Plot of the dihedrals angles for each time step of the MD simulation. Microstates are the state for each time step. These states are grouped into macrostates based on the time spent in them by the system. The analysis is done using an ϵ-machine implemented by the CSSR algorithm to describe the self-organizing nature of the conformational changes of the peptide dynamics by means of the different states known as causal states.

1.5 The Limitation of Study

In this section we explain some limitations in this research which also influence the scope of this work. Properties of the experiment data and parameter that has direct impact on the simulation are stated, and briefly discussed.

- The Number of Atoms

  The number of atoms (i.e. > 3 atoms) is very important in such research because the problem is an N-Body Problem. This means the more the atoms the more complex the system will be in terms of interaction, computation, and other measures. Though the system of atoms might have an initial state which defines its configurations, but in time this configuration becomes very complex due to several factors (quantum effect, physical, chemical etc.) that change this configuration. VPAL is just about 33 atoms, but real proteins are several thousands of atoms. In fact, we noticed that previously the experiment done on this work [12], with fewer atoms shows much different time step variance when considered in this case. That is, the ϵ-machine starts converging at 0.3ps, given even a less amount of data when compared with ours, 4.0ps. This shows that more atoms might lead to varying result and explanations, since there might be several factors to what really causes the drift in the results. Hence, one cannot adopt this same intuitive explanation to real proteins (i.e. > 1000 atoms) until the experiments are carried out for several different types of proteins with similar results.

- The Simulation Time

  In this research we used about 1μ simulation time, which sometimes will not be enough for the convergence for all the data (e.g. at $t^s = 0.4ps$ did not converged because more that is needed.).

1.6 Project Layout

In Chapter 1, the research is introduced with a detailed discussion on what it entails. The importance of the amino acid residues of VPAL and its chemical structure is discussed in Chapter 2. Briefly, the way the simulation is done and tools used are described as well the algorithms that computes the trajectories of the atoms in Chapter 3. The analysis is explained in Chapter 4, with concise mathematical details. In Chapter 5, the result is discussed, while in Chapter 6, the work is concluded and summarized. Appendix A contains some code fragments, and its followed by the Bibliography which contains the references inferred by this write-up.

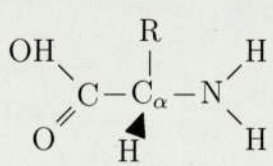
CHAPTER 2

THE VPAL STRUCTURE

There are 20 amino acids out of which all proteins formed . In this chapter we discuss some important features and structure of VPAL (Valine-Proline-Alanine-Leucine), by considering its residual constituents of amino acids.

2.1 Amino Acid

Organic compounds with the Amino (NH_2^-) and Carboxylic ($COOH^-$) groups attached to an R-group (e.g. CH3) are known as amino acids 2.0. They have high melting point and are soluble in water.

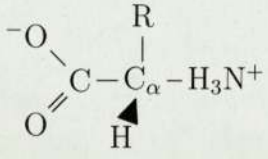


(2.0)

Amino Acid

In addition, the amino group is a base while the carboxyl group is an acid. Hence, amino acids are known as acid-base compounds, meaning they behaves as an acid as well as a base depending on the chemical reaction. More so, they exist in a particular form known as the **Zwitterion Form**, where the carboxylic group loses hydrogen (i.e. *deprotonate*) and amino group gains hydrogen (i.e. *protonate*);

which is known as the *amphiprotic* properties of amino acids, due to the presence of $COOH^-$ and NH_2^- functional groups. In this form the molecule of the amino acid contains equal number of positive and negative charges, which makes the net charge zero. Therefore, the zwitterion form of VPAL peptide is used in this research.



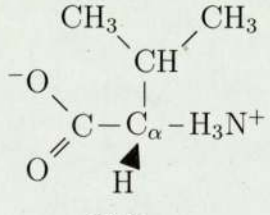
(2.1)

Zwitterion Form of an Amino Acid

The central carbon in the amino acids is asymmetric (i.e. it has 4 different groups attach to it). As a result, the acids exist in two forms known as the L-Form and D-Form. The former is considered the mostly, because they are manufactured in cells and incorporated into proteins, while the latter is a mirror of the former and its not normally found in proteins; which some are seen on cell walls of some bacteria which are not proteins.

Amino acids are classified as essential (i.e. amino acids that can't be synthesized by the organism body) and non-essentials (i.e. the body of the organism can synthesize the amino acid); grouped as hydrophilic (likely to stay in water) or hydrophobic (not likely to stay in water) based on their chemical behaviour. We consider here the four amino residues of our interest and categorize them accordingly. For more details on amino acids, see the reference on chemistry [2, 14]. The following are the residues that made up the VPAL and their zwitterion structures.

- **Valine**: is an essential hydrophobic amino acid.

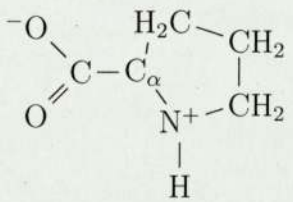


(2.2)

Valine

- **Proline**: is a non-essential aliphatic hydrophobic imino acid (i.e. called

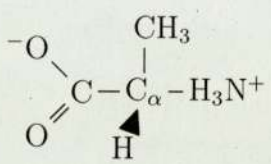
imino acid (NH_2^+) and not amino acid (NH_3^+)).



(2.3)

Proline

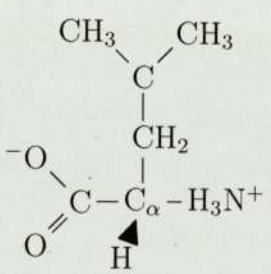
- **Alanine**: is a non-essential and hydrophobic amino acid just like proline.



(2.4)

Alanine

- **Leucine**: is also an essential hydrophobic amino acid.



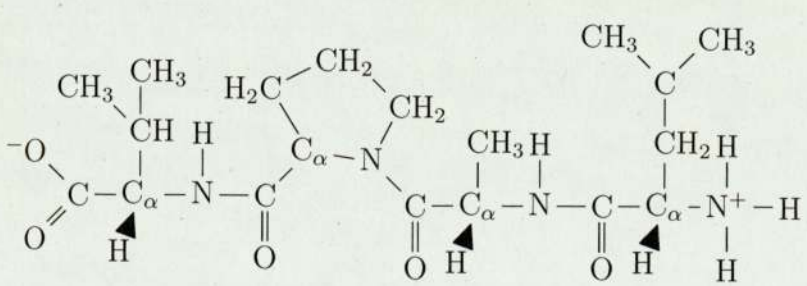
(2.5)

Leucine

2.2 VPAL

The combination of the above residues in a chemical reaction of their zwitterion form is as follows:

$$\text{Valine} + \text{Proline} + \text{Alanine} + \text{Leucine} \longrightarrow \text{VPAL} \tag{2.6}$$



(2.7)

Zwitterion Form of a VPAL Molecule

2.3 Dihedral Angles

Due to the present of the *peptide* bond (partial double covalent bond) the proteins backbone are very stable. Therefore, its important to describe the protein along its backbone. The angles ϕ and ψ along the planes of the peptide backbone, describes the structure of a peptide/protein molecule and are angles between N-C_α, and C_α-C respectively. The angles are could rotate 360 degrees, but not all of the angles are allowed, due to steric hindrances (i.e. situation that occurs due to bond rotation at specific angles, and courses the atoms to be at very close proximity that leads to instability of the peptide molecule) the angles are not allowed to take some specific values. In addition, the angles also take into account the 3-dimentional structure of the peptide molecule, which makes them very informative in describing its conformational form or changes[14].

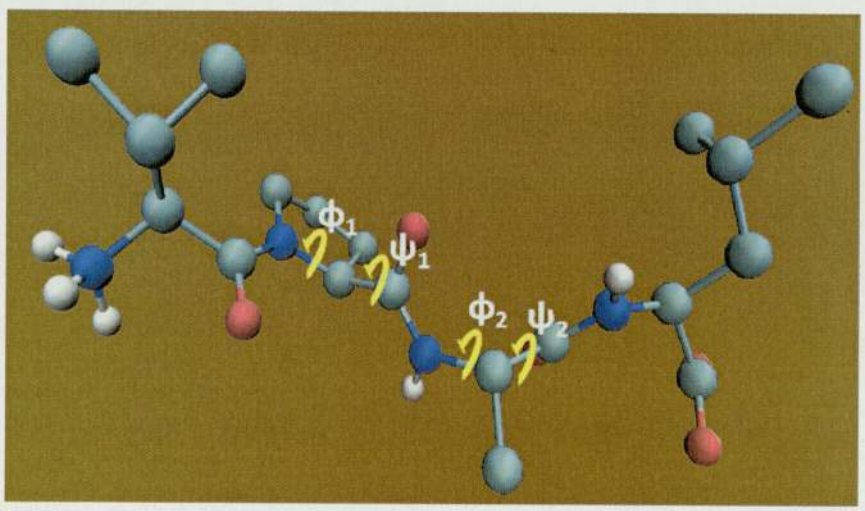


Figure 2.1: 3D Image of a VPAL Molecule showing the Dihedral Angles

It is noted that the dihedral angles (ϕ and ψ) of the first and the last residue of any protein or peptide is not considered useful in terms of the molecules conformation or conformational dynamics. This is due to the fast rotations of the bond angles (i.e. ϕ and ψ) of these residues [15, 16]. Therefore, the dihedral angles of Valine and Leucine residues of the VPAL molecules is not considered in this research as shown in Chapter 4.

2.4 Ramachandran Plot

Virtualizing a protein conformation distribution is done using a Ramachandran plot of the dihedrals angles. Since the angles exist between two or more amino acid residues of a peptide molecule, the plot helps describes the molecule along the backbone based on the positions of these residues in the 3-D space. Note, the Ramachandran plot is a 2D plot that describes a 3D molecule (2.2).

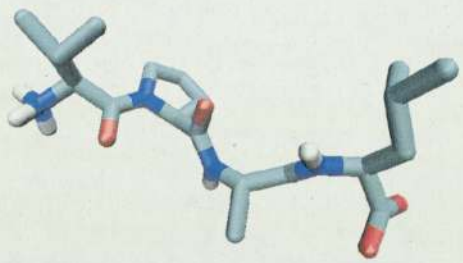


Figure 2.2: 3D Image of a VPAL Molecule. Colour scheme are carbon are cyan, hydrogen are gray, nitrogen are blue and oxygen are white

In addition, the plot helps to distinguish between acceptable and unacceptable peptide conformation distribution[17, 2].

The Ramachandran plot is used extensively in our research to describe the various clusters of VPAL molecule that is observed from the conformational changes of its macrostates. Chapter 4 and 5 have more details on the graphical representation of the plots.

CHAPTER 3

MOLECULAR DYNAMICS

3.1 Molecular Dynamic Simulation

The interaction of more than two bodies is what leads to N-particles problem [6, 5]. It was first observed on the solar system and seems to be insolvable analytically for $N > 3$. Due to the enormous amount of atoms or particles in real physical system, it is impossible to resolve the complex behaviour or properties analytically given this model (3.1).

$$m_i \ddot{\mathbf{r}}_i = \mathbf{F}_i. \tag{3.1}$$

MD is a computer simulation on how the motions of atoms and molecules affect the state or behaviour of a molecular system. In MD the N-particle problem is addressed at the microscopic level of particles, called atoms. During the simulation, the atoms and molecules are allowed to interact based on this model (3.1) for a period of time as defined in the system [6]. This method has proven to be very efficient in analysing disorder systems (glass dynamics, liquids, etc.) in various ways, to predict and help in making better models that explains the dynamics of these disordered systems [18, 13, 11, 19]. This kind of simulation can be done with any program that implements the MD specifications in terms of it

microscopic interaction calculation, which involves:

- Initialization of the coordinates and velocities of the atoms.
- Computing the Newtons equations and Potentials of the atoms.
- Finally, measuring other relevant properties and behaviour of the atoms, like the energy, correlation function etc.

This means the program should be able to implement the three major systematic stages of a given MD simulation. Hence, for the course of this research we considered using GROMACS as the choice of MD simulator for VPAL.

3.2 Simulation Tools

GROMACS is computer software that is designed specifically for MD simulations and it is written in C programming language. The term GROMACS is derived from, GROningen Machine for Chemical Simulation [20] because it was developed by the department of Biophysical Chemistry of the University of Groningen. It is now maintained as open source software by various research institutions and interested individuals across the continents. The package (software) has widely been used from the early 80s and ever since various versions has been created as well; currently the newest stable version is 4.6.2. Figure 3.1 shows the GROMACS flowchart [21]. The diagrams describe how simulation is carried out in the package. However, the level that concerns us in this research is the mdrun level, where the trajectory files are generated and the main process of the simulation is carried out. Briefly the parameters and theoretical framework of generating atoms trajectories are discussed in the following subsections. Please see the GROMACS Developers and User Guide for other relevant information concerning its theoretical considerations and how they are implemented within the software package.

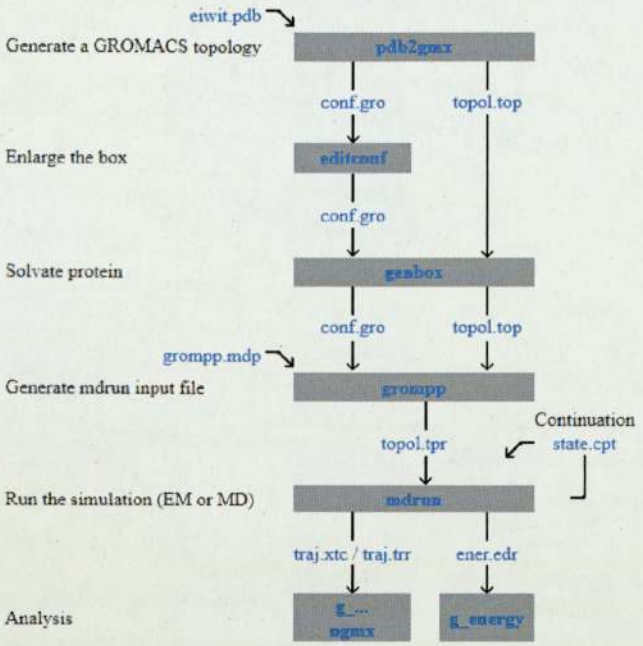


Figure 3.1: GROMACS simulation flowchart

However, some modifications where made to extract the trajectories of specific atoms for the purpose of analysis. Detail instructions on how it is done is found in Appendix B.

3.2.1 Theory Behind Generating The Trajectory

GROMACS implements an efficient approximation integrator called Verlet Algorithm, which is used to integrate Newton Second Law of motion for the purpose of solving for the *position* r *and velocity* v *of particles at time* t.

The algorithm is designed numerically as follows:
Lets consider the position of particles at a specific time, $(t + \triangle t)$ to be defined as $r_i(t + \triangle t)$. We shall consider two cases of the algorithm (i.e. when there is/not velocity dependent) to make clear our explanation.

- **Dependent**

  Using Taylor Series to expand the function, $r_i(t + \triangle t)$:

$$\mathbf{r}_i(t+\triangle t) = \mathbf{r}_i(t) + \triangle t\dot{\mathbf{r}}_{ii}(t) + \frac{1}{2}\triangle t^2\ddot{\mathbf{r}}_{ii}(t) + ...\text{highier order terms}, \quad (3.2)$$

$$\approx \mathbf{r}_i(t) + \triangle t\dot{\mathbf{r}}_{ii}(t) + \frac{1}{2}\triangle t^2\ddot{\mathbf{r}}_{ii}(t). \quad (3.3)$$

From equation (1.6), the Force ($\mathbf{F}$) and Velocity($\dot{\mathbf{r}}$)at time t are:

$$\mathbf{F}_i(t) = m_i\ddot{\mathbf{r}}_i(t). \quad \text{and} \quad \mathbf{v}_i(t) = \dot{\mathbf{r}}. \quad (3.4)$$

Therefore:

$$\ddot{\mathbf{r}}_i(t) = \frac{\mathbf{F}_i(t)}{m_i}. \quad (3.5)$$

Now putting (3.5) and $\mathbf{v}_i(t)$ into (3.3):

$$\mathbf{r}_i(t+\triangle t) \approx r_i(t) + \triangle t\mathbf{v}_i(t) + \frac{1}{2m_i}\triangle t^2\mathbf{F}_i(t). \quad (3.6)$$

- **Independed**

  Lets take the time to be $t-\triangle t$, hence by using the Taylor Series on $\mathbf{r}_i(t-\triangle t)$, and substituting $\mathbf{v}_i(t)$ and $\mathbf{F}_i(t)$, we have:

$$\mathbf{r}_i(t-\triangle t) \approx \mathbf{r}_i(t) - \triangle t\dot{\mathbf{r}}_i(t) + \frac{1}{2}\triangle t^2\ddot{\mathbf{r}}_{ii}(t), \quad (3.7)$$

$$\approx \mathbf{r}_i(t) - \triangle t\mathbf{v}_i(t) + \frac{1}{2m_i}\triangle t^2\mathbf{F}_i(t). \quad (3.8)$$

Combining equation(3.3) and (3.8)

$$\mathbf{r}_i(t+\triangle t) + \mathbf{r}_i(t-\triangle t) = \mathbf{r}_i(t) + \triangle t\mathbf{v}_i(t) + \frac{1}{2m_i}\triangle t^2\mathbf{F}_i(t) + \mathbf{r}_i(t) \quad (3.9)$$

$$- \triangle t\mathbf{v}_i(t) + \frac{1}{2m_i}\triangle t^2\mathbf{F}_i(t), \quad (3.10)$$

$$= 2\mathbf{r}_i(t) + \frac{1}{m_i}\triangle t^2\mathbf{F}_i(t). \quad (3.11)$$

Finally, the computation of the trajectory becomes:

$$\mathbf{r}_i(t + \triangle t) = 2\mathbf{r}_i(t) - \mathbf{r}_i(t - \triangle t) + \frac{1}{m_i}\triangle t^2 \mathbf{F}_i(t). \tag{3.12}$$

Note: This algorithm is important for a proper estimation of the required energy of the system. It is time reversible, it has a symmetrical potential and conserve angular momentum, which are properties that a good numerical integrator should have[3, 8]. In addition, the Position-Verlet equation (3.12) is shown to be efficiently stable as the time step scales to infinity [22].

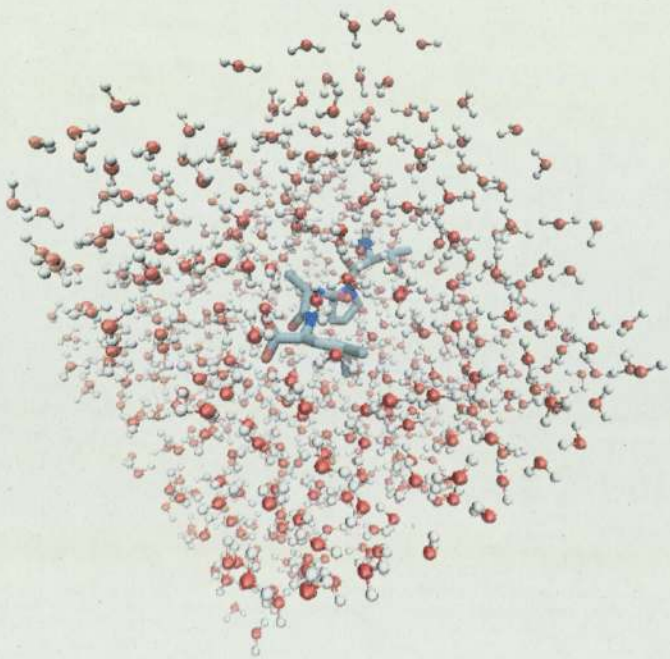


Figure 3.2: VPAL in Explicit Water

3.2.2 Parameters of Simulation

The simulation of the VPAL peptide molecule is done in explicit water as shown in figure (3.2). MD simulation needs to run at certain temperature depending on the type of simulation. This simulation uses the Berendsen Thermostat to keep the temperature at $300\,K$, in other to ensure that the system is kept at equilibrium. As mentioned above, a Verlet algorithm is used as the integrator with intervals of

$\triangle t = 0.002ps$. The peptide and water molecules make up a total of 1917 atoms, together in a box of dimension, $25 \times 22 \times 22$ Armstrong(Å). Due to finite size of data which leads to a problem of boundary effect in MD simulations, we use Periodic Boundary Conditions (PBC) during simulation, to ensure that the system behaves infinitely. Long and short range interactions of the atoms were taking into consideration when the simulation was carried out.

The Particle Mesh Ewald (PME) summation algorithm is used for computing the electrostatic interaction energies of the atoms during simulation. This is preferred knowing the system is periodic. Basically, the methods is used to sum efficiently the electrostatic energies interaction of the atoms, by computing the short-range and long-range interactions separately; finally taking the sum of both as the resulting total energy interaction of the atoms within the system.

In addition, the simulation time was kept at 500 million steps (i.e $1\,\mu s$ in terms so the integrator steps, $0.002\,ps$), and the Gromos 53a6 [23] as the molecular force field, since it is optimized for bimolecular interactions with water molecules.

CHAPTER 4

ANALYSIS OF MD DATA

Both Ramachandran Plot and Computational Mechanics are used in analysing the dihedral angles of the peptide backbone that was generated at each time step ($\triangle t^s$) of the simulation. Therefore, this chapter is divided into two major sections for a proper description of how the analysis is done.

4.1 Dihedral Angles of the VPAL Backbone

The generated coordinates of the resulted trajectories of the peptide dynamics is used to calculate the dihedral angles which describes the conformation at the $\triangle t^s$.

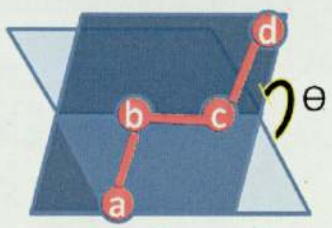


Figure 4.1: 4 bonded atoms (a, b, c and d) use to illustrate the Dihedral Angle (ϕ and ψ) of a given peptide or protein.

The ϕ and ψ of VPAL (2.1) is considered as described by the diagram (4.1). These dihedral angles for each $\triangle t^s$ is computed as follows:

$$\vec{Q} = \vec{\zeta}_{ab} \times \vec{\zeta}_{bc} \tag{4.1}$$

$$\vec{W} = \vec{\zeta}_{bc} \times \vec{\zeta}_{cd} \tag{4.2}$$

Where $\vec{\zeta_{ij}}$ are **bond vectors**

$$\cos\theta_{abcd} = \frac{\vec{Q} \cdot \vec{W}}{|\vec{Q}||\vec{W}|} \tag{4.3}$$

$$\sin\theta_{abcd} = \frac{(\vec{Q} \times \vec{W}) \cdot \zeta_{ab}}{|\vec{Q}||\vec{W}||\zeta_{ab}|} \tag{4.4}$$

$$\tan\theta_{abcd} = \frac{\sin\theta_{abcd}}{\cos\theta_{abcd}} \tag{4.5}$$

$$\theta_{abcd} = \arctan\left(\frac{\sin\theta_{abcd}}{\cos\theta_{abcd}}\right) \tag{4.6}$$

The dihedral angles for each t^s is:

$$\theta_{(abcd)_{t^s}} = \arctan\left(\frac{\sin\theta_{(abcd)_{t^s}}}{\cos\theta_{(abcd)_{t^s}}}\right) \tag{4.7}$$

Where the dihedral angle, θ, means either ϕ or ψ.

4.2 Virtual Clusters

Clustering is a way of understanding data by means of statistical analysis that groups data points according to their attributes or behaviours. Therefore, it is applied in this research to understand the behaviour of the data on a Raman-

chandran Plot as explained in Chapter 2, then proceed with the CM analysis. The following figures (4.2, 4.3,..., 4.6) show the behaviour of the data as the time increases.

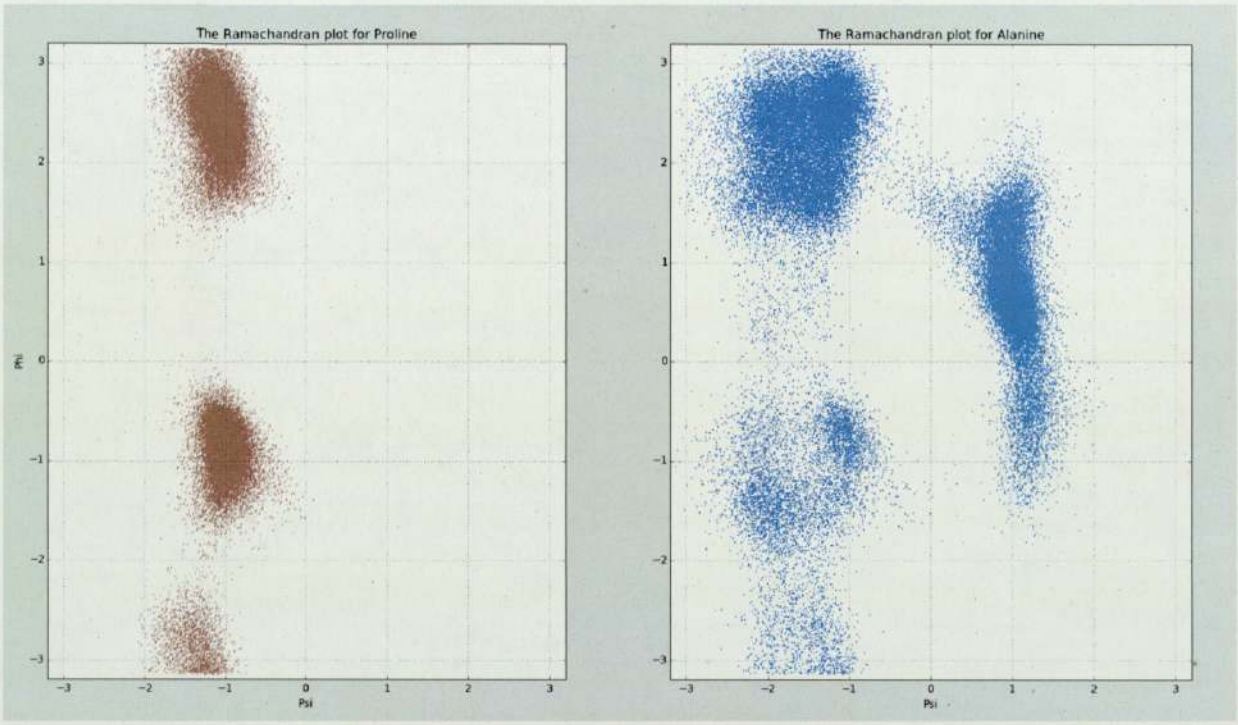


Figure 4.2: Ramachandran plot for 50000 data points.

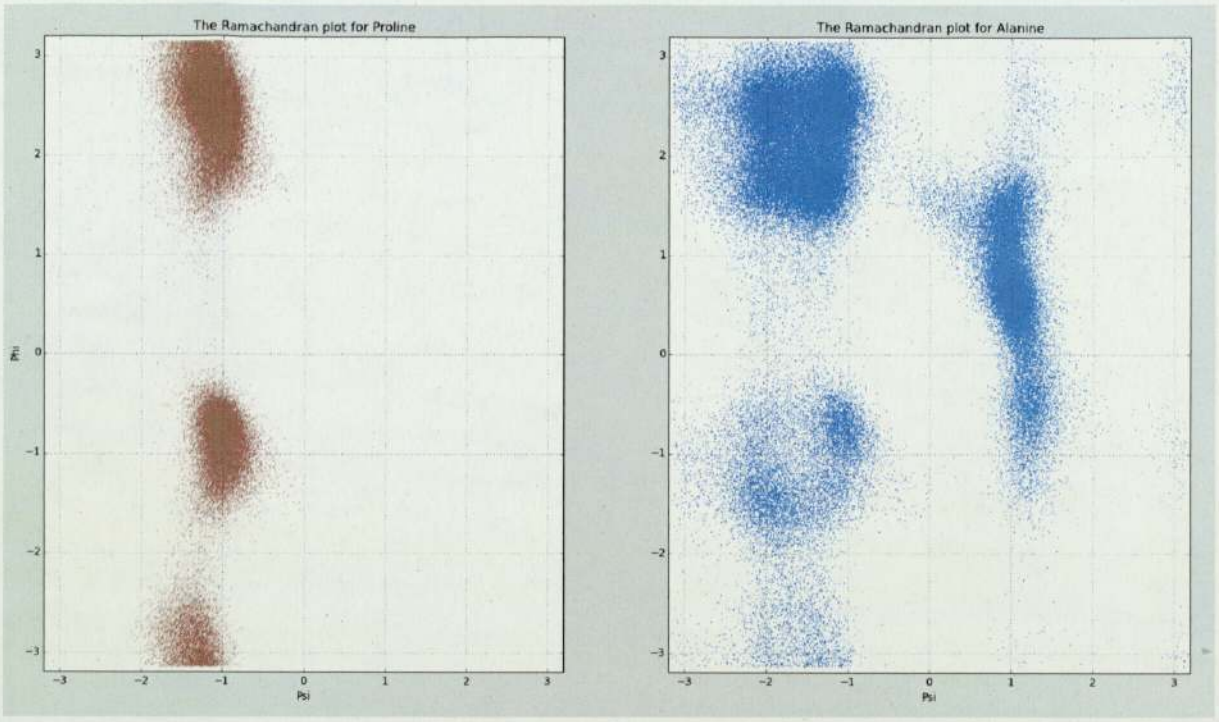


Figure 4.3: Ramachandran plot for 100000 data points.

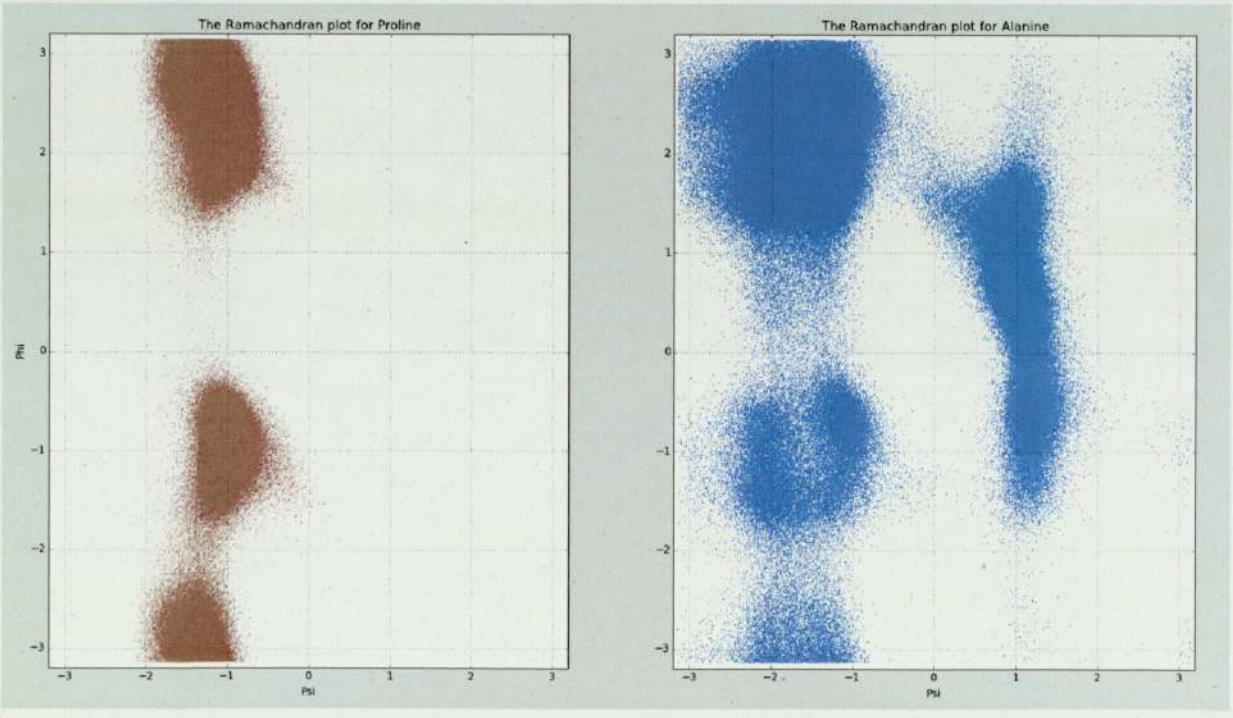


Figure 4.4: Ramachandran plot for 500000 data points.

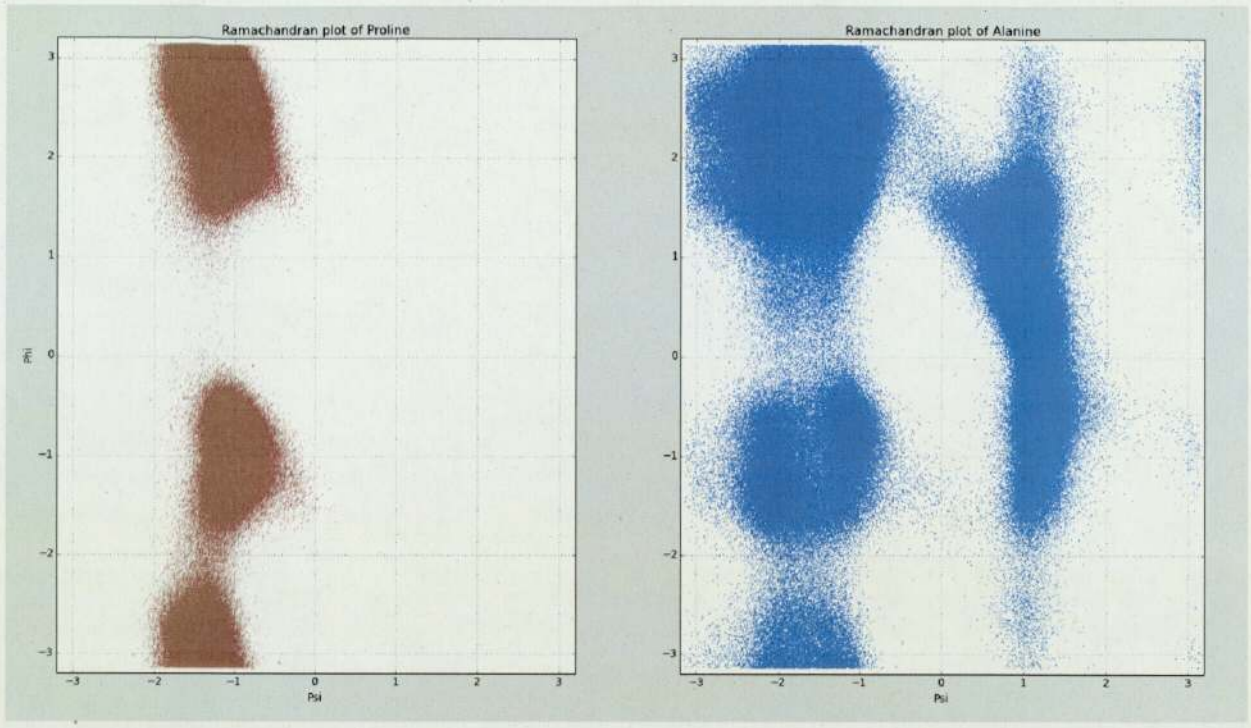


Figure 4.5: Ramachandran plot for 1000000 data points.

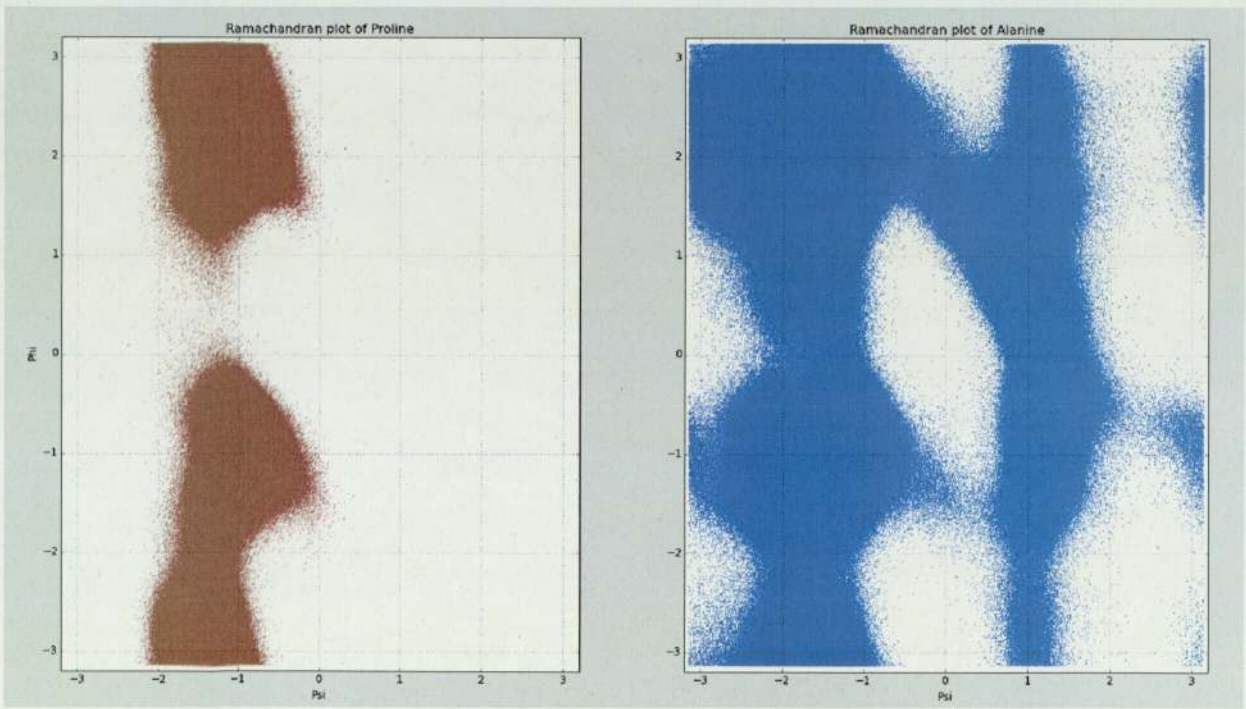


Figure 4.6: Ramachandran plot for 10000000 data points.

Note: These are radian plots with five observed clusters as the time increases.

4.2.1 Partitioning of Ramachandran Plot

The dihedral angles obtained from the trajectory of the simulation is plotted on a Ramanchandran Plot as shown in the figure 4.6. This is then partitioned 4.7 to depicted the conformational forms of the peptide. Each partition is symbolized using A_i,B_i and C_i; $i = 1, 2$. Where i represent an area of a specific peptide residue.

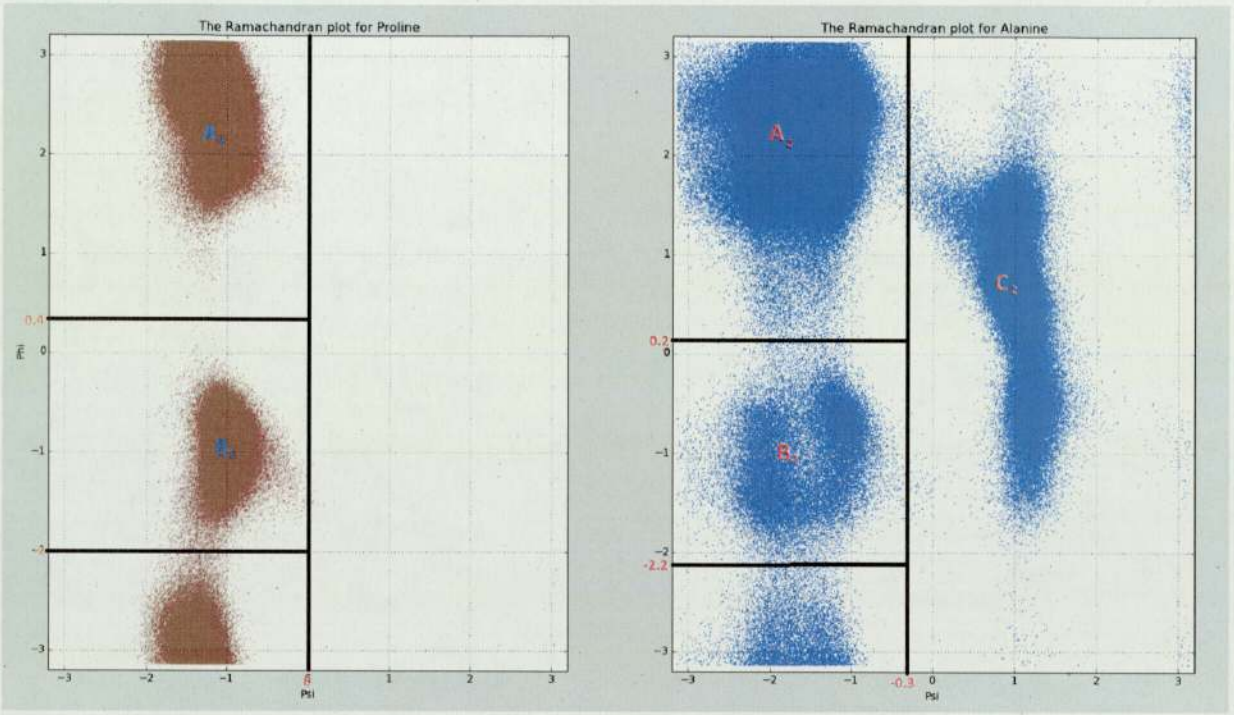


Figure 4.7: Ramachandran plot showing the partitioning scheme, and symbols assigned to each clusters of the 500000 data points.

After partitioning, symbols were assigned to data point that fall into four major group of metastable conformations. These groups are formed by combining different partitions [10]. Figure 4.8 display various clusters of each pair of the dihedral angles that are combined to form the metastable conformational states. Table (4.1) shows the symbols and their respective metastable states.

Symbols	Clusters
1	$A_1B_2 + B_1B_2$
2	A_1A_2
3	B_1A_2
4	A_1C_2

Table 4.1: A table showing the assignment of symbols and their clusters.

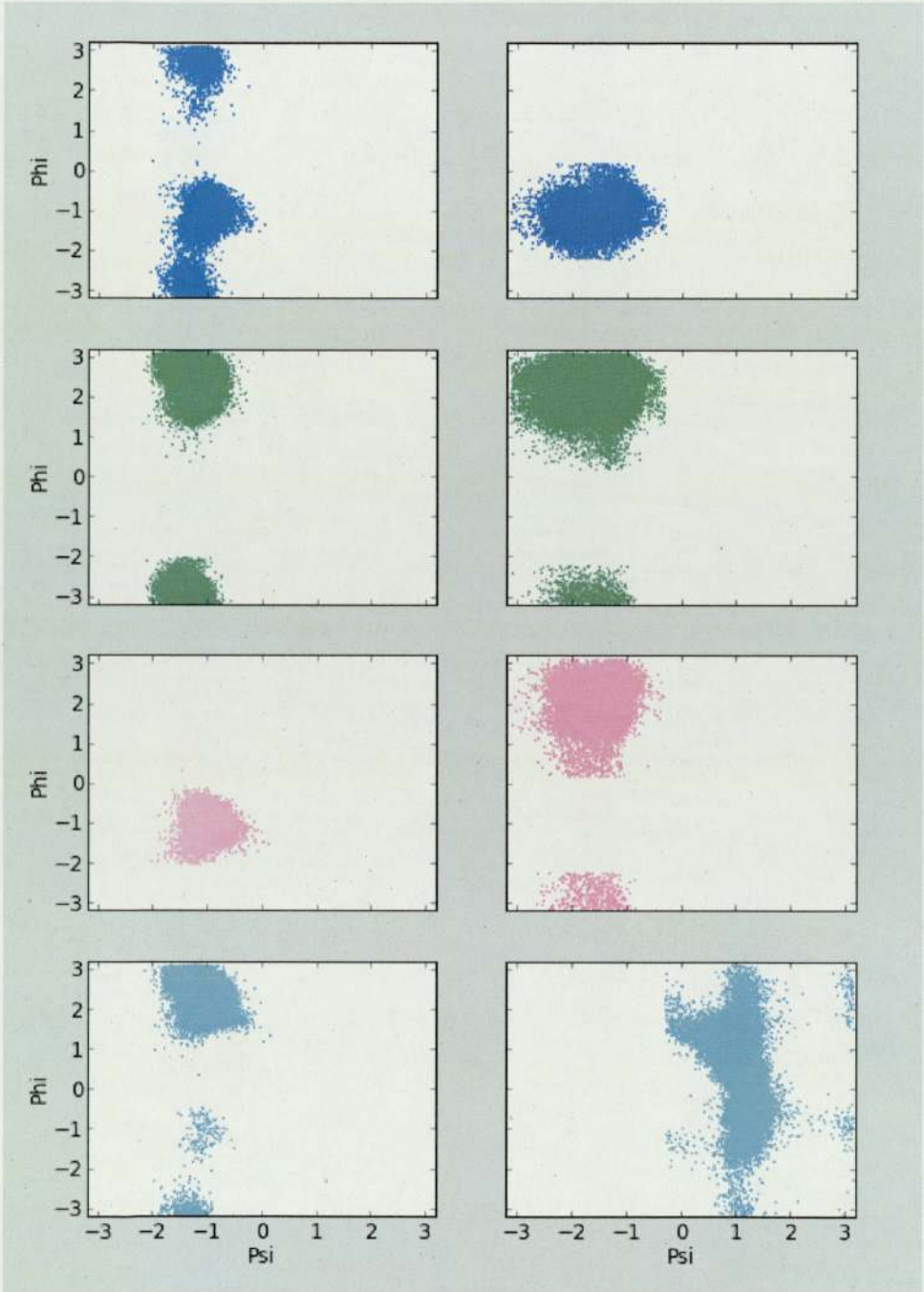


Figure 4.8: A Ramachandran plot, showing the pair of clusters in each metastable clusters.

Figure(4.9) displays a single plot of the entire metastable conformations. The overlap are easily spotted on the graph, and it is investigated in the next sections.

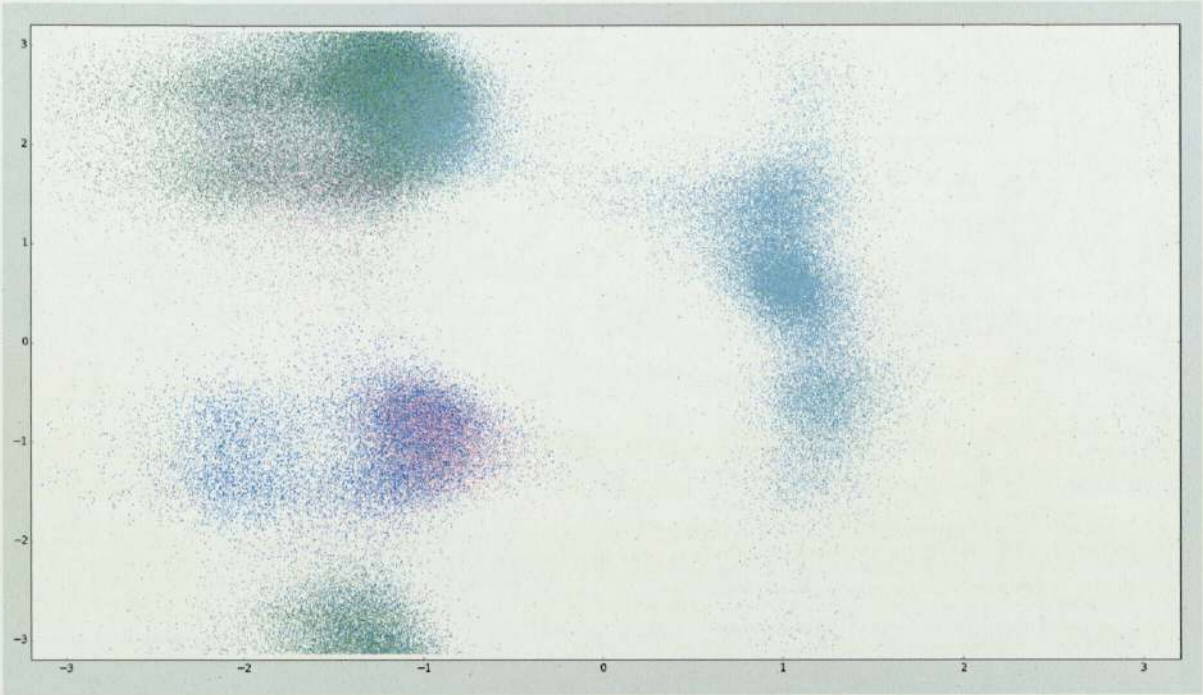


Figure 4.9: A plot showing the different clusters on a 2D metric space.

4.3 Causal State and ϵ-Machine

In this section the complexities of the patterns within the peptide conformation is addressed, by looking at the past observations of the conformational processes to infer the future changes observed in time.

A brief introduction of Computation Mechanics (CM) and how it is applied in this research is explained in detail. In addition, some theories and lemma can be found in [24], which details the mathematical rigorous proofs of its correctness. However, we are not going to expressly give any mathematical proof, but state the concepts and explains particularly the intuitive theories of Causal States and ϵ-Machine which are the expressive ideas of CM.

The trajectory contains coordinates of continuous values for each atom at different time step ($\triangle t^s$) of the simulation time. This values are then discretised

in various selected ($\triangle t^s$), from which an infinite string $\{s_i | i = 0, 1 \cdots, N\}$ is generated from a finite symbols of Alphabet ($\mathcal{A}$) based on certain criteria which designate specific symbols 4.1 for different coordinates at time t_i as describe by the Ramachandran Plot (4.7). This system can be termed a stochastic process since it is probable for any alphabet to be generate at a specific time.

Now that the system has been explicitly redefined as a machine which generates strings from a finite $\mathcal{A}$ at time ${t^s}_i$. The next step is to make meaning out of the generated strings and that is where CM (a variant of Hidden Markov Model) is applied in this research analysis. CM is briefly described mathematically in the following subsections.

4.3.1 The System

Given an infinite string, $\overleftrightarrow{S} = \ldots s_{-4}s_{-3}s_{-2}s_{-1}s_0s_1s_2s_3s_4 \ldots$ of random variables from a finite set of symbols $\{s\}$ generated by an Alphabet ($\mathcal{A}$) at different time, t^s. At any given t^s, the set $\{\overleftrightarrow{S}\}$ can be split into two sequence a past ($\overleftarrow{s}$) and future ($\overrightarrow{s}$). That is at any t, the sequence can be denoted as:

- The Past :

$$\overleftarrow{s_{t^s}} = \ldots s_{-4}s_{-3}s_{-2}s_{-1}. \tag{4.8}$$

- The Future:

$$\overrightarrow{s_{t^s}} = s_0s_1s_2s_3s_4 \ldots \tag{4.9}$$

4.3.2 Morph

From the sequence of strings given in the past the future can be inferred. The morph is a term that is used to describe the conditional probability of the future

given the past.

$$P(\overrightarrow{s}|\overleftarrow{s}) \tag{4.10}$$

4.3.3 Causal States

The future morph at different times is know as causal states [24] and its described in this section. The probability distribution of a specific past $\overrightarrow{s}_t^L$ of length L and time t^s over all possible futures $\overrightarrow{s}$ is:

$$P(\overrightarrow{s}|\overleftarrow{s}_{t^s}^L) \quad 0 < L < \infty. \tag{4.11}$$

This means when there is no define length (i.e $L = 0$) of consecutive strings, then there is no history(i.e. an empty string is generated, λ).

$$P(\overrightarrow{s}|\overleftarrow{s}_{t^s}^0) = P(\overrightarrow{s}|\lambda), \tag{4.12}$$

$$= P(\overrightarrow{s}). \tag{4.13}$$

The equivalent classes which are created by means of the equivalent relation,$\sim\in$ $\overleftarrow{s}_{t^s}^L$, forms the causal states,$\{\mathcal{S}_i \in \mathbf{S}|i = 1, 2\cdots, \mathbf{N}_s\}$ of the system.

$$\overleftarrow{s}_{t^s}^L \sim \overleftarrow{s}_{t^{s'}}^L \quad \Longleftrightarrow \quad P(\overrightarrow{s}|\overleftarrow{s}_{t^s}^L) = P(\overrightarrow{s}|\overleftarrow{s}_{t^{s'}}^L) + \alpha \,\forall\, \overrightarrow{s}. \tag{4.14}$$

Where α is the negligible factor for which the two distribution are taken to be equal. The causal state are important feature as notices from the model, because they take into account both the microstates and macrostates (i.e. more than one microstates) of the peptide conformational dynamics. They form a Markov process, but are constructed from a non-Markov dynamic and can recursively be calculated[25].

4.3.4 ϵ-machine

The process of labelling the probabilistic transition of causal states with the function ϵ is called an ϵ-machine, which is describes mathematically in this section.

From the set of causal states $\{\mathbf{S}\}$, the transitional probabilities $T_{ij}^{(S)}$ between states i and j is given as follows from the set of symbols $s \in \mathbf{A}$:

$$T = \Sigma_{s \in A} T^{(s)}. \tag{4.15}$$

and the component T_{ij} for the causal states i and j is,

$$T_{ij} = P(\mathcal{S}_j | \mathcal{S}_i). \tag{4.16}$$

T is a matrix of probabilistic transitions with its components, T_{ij}. The matrix T is normalized, hence the probability for exiting a state is 1, (i.e. $\Sigma_j T_{ij} = 1$). Sometimes the system might run for a long time and create a chains of probability, $P(\mathcal{S}_j)$ of a given casual state, defined as follows:

$$P(\mathcal{S}_j) = \sum_{i=1}^{\mathbf{N}_s} P(\mathcal{S}_i) T_{ij} \tag{4.17}$$

The ϵ-machine describes the systems minimal configuration or patterns, from the causal states and transition probabilities and it does depends on the behaviour of the systems dynamics by which the symbols where generated. Also, other useful properties of the system can be infer from the ϵ-machine, like statistical complexity, entropy rate, etc [9, 26, 27].

4.4 CSSR Implementation

Causal State Splitting Reconstruction (CSSR) Algorithm, is a non recursive predictor algorithm for discrete sequences that implements CM [28]. It has proven to be very efficient and estimated accurately the internal configuration, and represent

properly the causal states and e-machine of tested systems [12, 24, 29]. There are basically three phases to the algorithm, the initializing of state, testing of null hypothesis, and finally, the recursive calculation of the e-machine. It implements very much every mathematical details described above and others[24]. Therefore, it is applied in this work as a tool for our analysis.

A finite set of the simplest states that describes the statistical behaviour of the actual data string generated by the system, is outputted by the algorithm. This states forms the minimum Markov chain that is capable of generating the actual sequence $\{\mathbf{s}\}$ at dynamic time $\triangle t^s$. At certain time and particular probability, a symbol is outputted from a specific state that comprises of strings of symbols $\{s_i\}$ which defines what state the system is currently at. This means, the Markov Chain is at the state $\{s^L\}$ if it ends in a sequence of symbols $\{s^L\}$ at a time t^s.

To produce the estimated causal states that predicts the discrete sequence from the trajectory of the system, the algorithm needs to be supplied with string sequence $\{\mathbf{s}\}$ and a subset of length (L) of the string (i.e. $\{s^L_{t^s} \in \mathbf{s}\}$). Also, the Maximum Length (L) and Significant Level (σ) should be chosen with caution as express by the equation (4.18), which we took into consideration in this work [28].

$$\Lambda \leq L \leq \frac{\log_2 N}{log_2 A_N} \tag{4.18}$$

Where N is the total length of the string of symbols, Λ is the minimum acceptable word history, and A_N is the total number of alphabet symbols.

CHAPTER 5

RESULTS AND DISCUSSION

After a successive MD simulation of VPAL and analysing the generated trajectory, results were obtained which is discoursed in this chapter. These results are explained in three subsection based on their graphical representations. The transition state diagram are fairly complex to explain if all it describes is considered, therefore only the meaning that is related to this work is explained.

5.1 Dependence of Result on σ and l

In order to chose which significant level (σ) and maximum length (l) to use, a matrix was developed that correspond to each σ and l at different time step, $\triangle t^s$. This is to ensure that the list l is chosen. 8 major matrices are chosen for the basis of the discussion on the results archived. It is clear how the system converges approximately to different states, which is called a **plateau** (a series of a particular state that spreads across columns and rows). In addition, the matrix shows that as the number of steps reduces the corresponding states increases, which clearly illustrate the effect of the conformational dynamics of the system. Hence, the different matrices are listed to give a proper virtuallization on which of σ and l is used to generate the ϵ-machine .

l	1	2	3	4	5	6	7	8	9	10
σ										
0.1000	4	5	5	8	36	120	215	359	539	594
0.0656	4	5	5	8	36	120	215	357	531	552
0.0430	4	4	4	7	53	123	210	421	520	571
0.0282	4	4	4	7	31	116	213	417	521	577
0.0185	4	4	4	4	4	4	4	4	4	4
0.0122	4	4	4	4	4	4	4	4	4	4
0.0080	4	4	4	4	4	4	4	4	4	4
0.0052	4	4	4	4	4	4	4	4	4	4
0.0034	4	4	4	4	4	4	4	4	4	4
0.0023	4	4	4	4	4	4	4	4	4	4
0.0015	4	4	4	4	4	4	4	4	4	4
0.0010	4	4	4	4	4	4	4	4	4	4

140ps

At 140ps steps it shows that the conformational dynamics of the peptide molecule becomes Markovian, since it requires just a history of length one (i.e $l = 1$ for a plateau of 4's) to infer to the four converge states.

l	1	2	3	4	5	6	7	8	9	10
σ										
0.1000	4	6	6	8	57	129	195	344	531	687
0.0656	4	6	6	6	48	100	186	329	496	674
0.0430	4	6	6	6	59	16	192	309	458	607
0.0282	4	6	6	6	59	16	192	309	458	607
0.0185	4	5	5	5	5	105	203	307	574	644
0.0122	4	5	5	5	5	5	5	5	5	5
0.0080	4	5	5	5	5	5	5	5	5	5
0.0052	4	5	5	5	5	5	5	5	5	5
0.0034	4	5	5	5	5	5	5	5	5	5

0.0023	4	5	5	5	5	5	5	5	5	5
0.0015	4	5	5	5	5	5	5	5	5	5
0.0010	4	5	5	5	5	5	5	5	5	5

100ps

System converges with 5 states as shown in the matrix. This reveals that for 100ps the system is Non-Markovian since $l = 2$, whit an additional state.

l	1	2	3	4	5	6	7	8	9	10
σ										
0.1000	4	8	11	18	36	58	114	202	325	423
0.0656	4	8	11	18	36	58	114	206	307	421
0.0430	4	7	9	17	36	79	119	220	316	443
0.0282	4	7	8	8	12	13	142	199	217	401
0.0185	4	7	8	8	12	13	142	199	217	401
0.0122	4	7	8	8	12	13	142	199	271	408
0.0080	4	6	8	8	18	15	130	229	280	383
0.0052	4	6	6	6	6	6	6	6	6	6
0.0034	4	6	6	6	6	6	6	6	6	6
0.0023	4	6	6	6	6	6	6	6	6	6
0.0015	4	6	6	6	6	6	6	6	6	6
0.0010	4	6	6	6	6	6	6	6	6	6

50 steps

$l = 2$ with a gradually increase in states and complexity of the system dynamics.

l	1	2	3	4	5	6	7	8	9	10
σ										
0.1000	4	11	11	22	51	88	133	186	271	378
0.0656	4	11	11	14	36	75	91	166	220	334
0.0430	4	10	10	13	46	105	131	190	296	395
0.0282	4	10	10	10	10	88	124	205	269	408

0.0185	4	10	10	10	10	10	10	10	10	10
0.0122	4	10	10	10	10	10	10	10	10	10
0.0080	4	10	10	10	10	10	10	10	10	10
0.0052	4	10	10	10	10	10	10	10	10	10
0.0034	4	9	12	13	30	62	124	190	303	393
0.0023	4	9	12	13	30	62	124	190	303	393
0.0015	4	9	12	13	30	62	124	190	303	393
0.0010	4	9	12	13	30	62	124	190	303	393

20ps

$l = 2$, but more complex dynamics with about 10 states

l	1	2	3	4	5	6	7	8	9	10
σ										
0.1000	4	13	14	18	19	78	136	217	266	370
0.0656	4	12	13	15	16	77	124	221	301	365
0.0430	4	11	12	12	15	67	125	182	242	303
0.0282	4	11	12	12	12	63	107	171	242	352
0.0185	4	11	12	12	12	12	12	12	12	12
0.0122	4	11	11	11	11	11	11	11	11	11
0.0080	4	11	11	11	11	11	11	11	11	11
0.0052	4	11	11	11	11	11	11	11	11	11
0.0034	4	11	11	11	11	11	11	11	11	11
0.0023	4	11	11	11	11	11	11	11	11	11
0.0015	4	11	11	11	11	11	11	11	11	11
0.0010	4	11	11	11	11	11	11	11	11	11

15ps

With a small difference in time from 20ps to 15ps, the system converges to 11 states. This shows the conformational state are highly sensitive to the dynamics of the system even at $l = 2$

l	1	2	3	4	5	6	7	8	9	10
σ										
0.1000	4	13	22	28	47	89	167	253	394	505
0.0656	4	12	22	23	26	50	151	251	366	440
0.0430	4	12	21	23	29	75	174	268	399	443
0.0282	4	12	20	19	19	78	144	244	322	422
0.0185	4	12	19	18	18	64	153	202	286	410
0.0122	4	12	18	17	17	66	132	203	293	342
0.0080	4	12	18	17	17	66	132	203	293	342
0.0052	4	12	16	16	16	70	131	202	289	343
0.0034	4	12	16	16	16	70	131	202	289	343
0.0023	4	12	16	16	16	70	131	202	289	343
0.0015	4	12	16	16	19	73	132	194	301	337
0.0010	4	12	16	16	19	73	132	194	301	337

8ps

Now is getting even more interesting as the systems needs additional histories (i.e. $l = 3$), leading to 16 states and more complexity.

l	1	2	3	4	5	6	7	8	9	10
σ										
0.1000	4	15	27	36	59	114	195	332	467	666
0.0656	4	15	26	33	51	82	174	301	429	592
0.0430	4	14	25	29	45	81	171	285	466	628
0.0282	4	14	25	29	45	81	162	283	455	637
0.0185	4	14	24	27	43	86	180	271	443	650
0.0122	4	14	24	27	38	82	171	273	417	620
0.0080	4	14	23	26	39	82	161	267	403	571
0.0052	4	14	23	24	37	73	153	264	384	573
0.0034	4	13	24	24	37	73	154	256	374	574
0.0023	4	13	23	24	37	58	159	265	405	632

0.0015	4	13	21	24	37	60	158	262	416	633
0.0010	4	13	22	25	30	47	135	241	423	599

4ps

At this point the system has almost converged to either 23 or 24 states and very complex. However, we shall consider the 24 states with $l = 2$, since at $\sigma = 0.0034$ its form a plateau.

l	1	2	3	4	5	6	7	8	9	10
σ										
0.1000	4	16	50	112	183	271	404	617	978	1720
0.0656	4	16	51	109	166	241	343	552	899	1669
0.0430	4	16	49	103	164	241	347	531	907	1691
0.0282	4	16	50	97	159	235	341	501	867	1660
0.0185	4	16	50	91	154	236	332	490	830	1585
0.0122	4	16	50	90	145	220	309	468	808	1525
0.0080	4	16	50	84	139	211	318	493	814	1560
0.0052	4	16	50	85	135	207	328	494	785	1534
0.0034	4	16	51	84	131	204	322	482	770	1545
0.0023	4	16	51	81	134	206	324	477	744	1505
0.0015	4	16	50	83	128	192	291	444	752	1516
0.0010	4	16	50	83	129	193	294	433	735	1485

0.4ps

At 0.4ps the system has not converged, which means more data is needed for it to possibly converge at this time step.

5.2 E-Machine Reconstruction

The state machines that correspond to each matrices is given as follows: The state machine in figure 4.2 clearly shows that the highest probabilities of each state is when a transition is done to itself. Hence, they are metastable states.

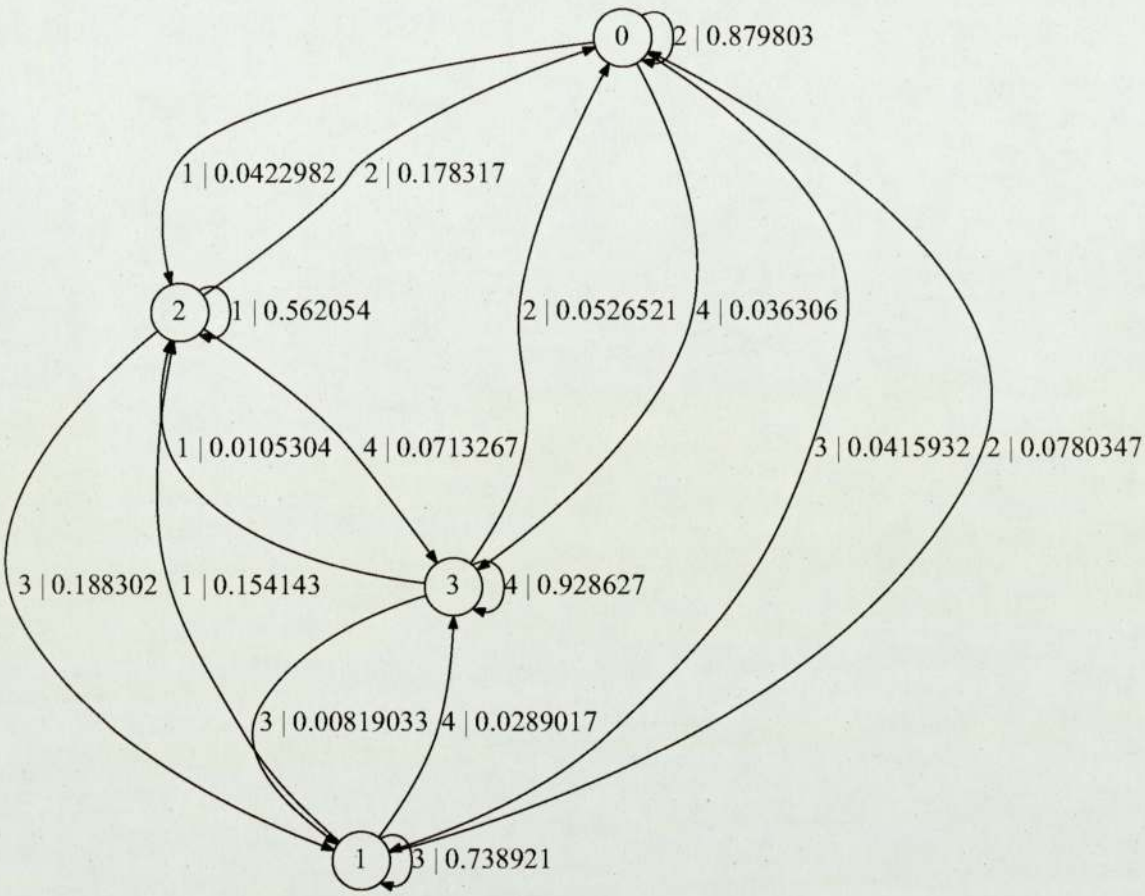


Figure 5.1: 4 States formed for time step 140ps

From figure 5.1, a new state(4) can be seen that describes the transition of the symbols from state (1) to other states. However, the transition it makes to state 2(symbol 1) is mostly probable, but other options are available as well.

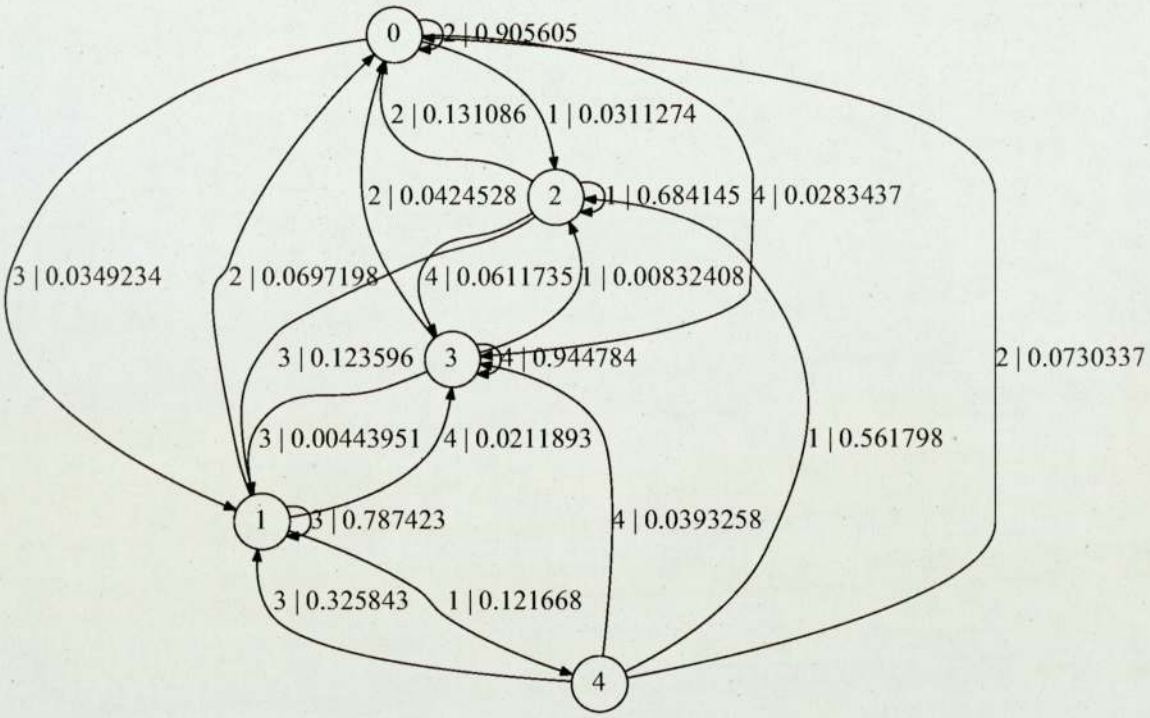


Figure 5.2: 5 States for time step 100ps

The 4 metastable states are more clearly separated (probability of transiting to themselves closer to 1) as shown in figure 4.4. That means the transitions are decoupled from metastable dynamics and described by the states 2 and 3

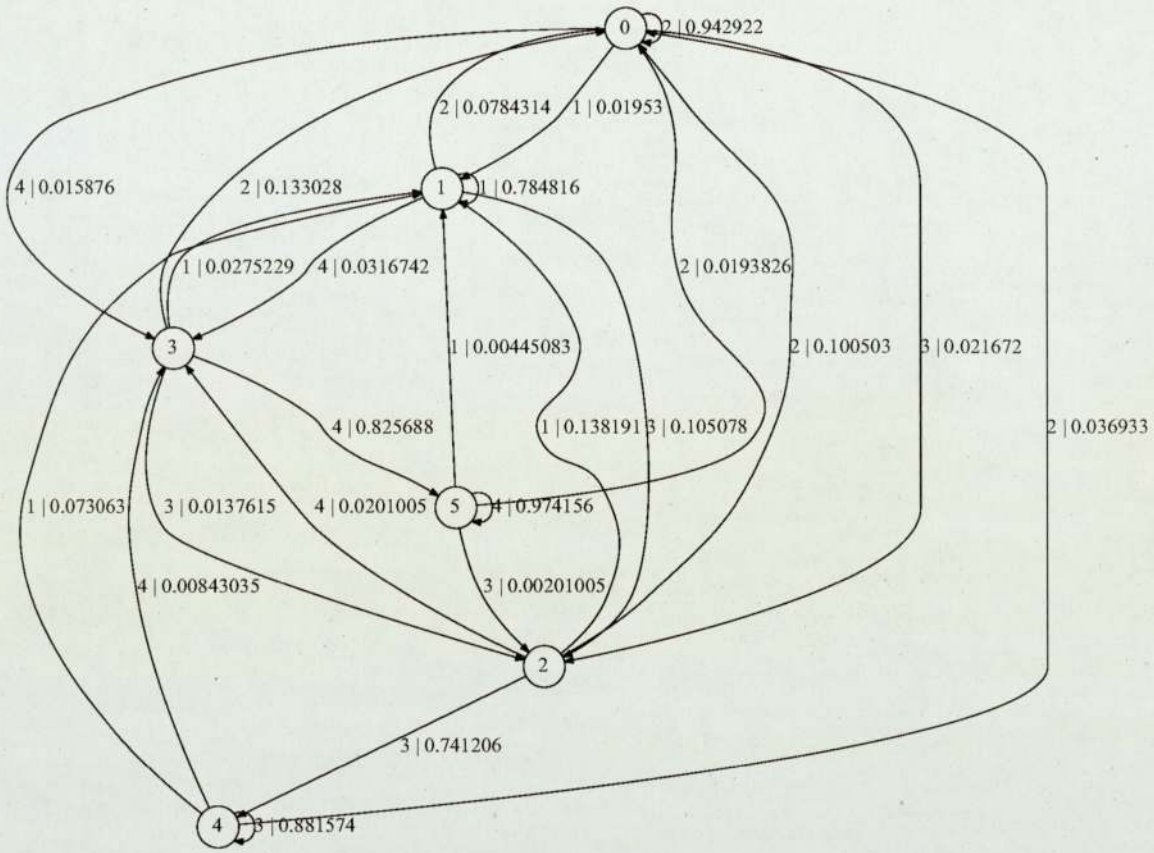


Figure 5.3: 6 States for time step 50ps

The matastable states are 0, 1, 4 and 9, with several other states that describe the various possible route of the different transition and metastable states.

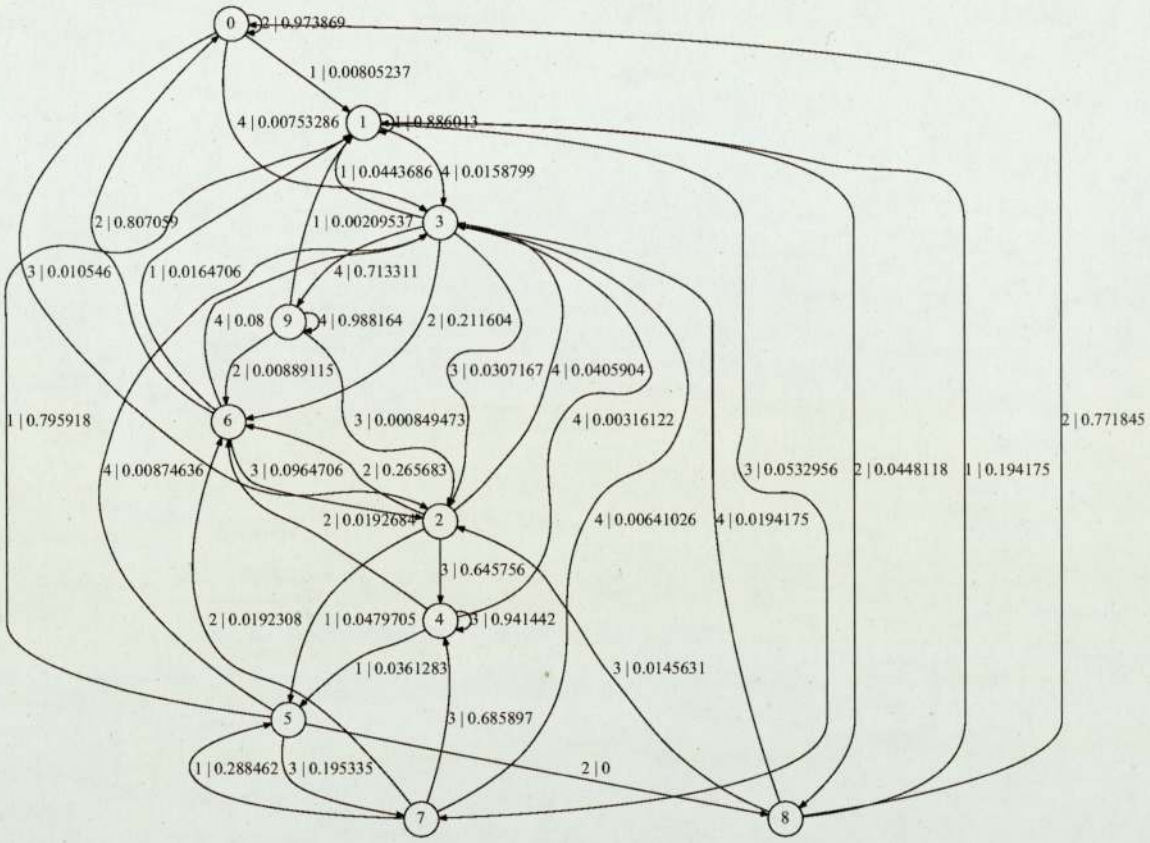


Figure 5.4: 10 States for time step 20ps

In figure 5.4 it is clear how the probability of the metastable state closely gets to 1. Also it can be noticed that state (8) is a transient state to state (4) whit a probability of above 60% to output symbol(4). Others can be spotted as wells. This means the system those not spend more time on some states, but transit immediate to other state, which clearly explains the drift in the conformation of the peptide dynamic.

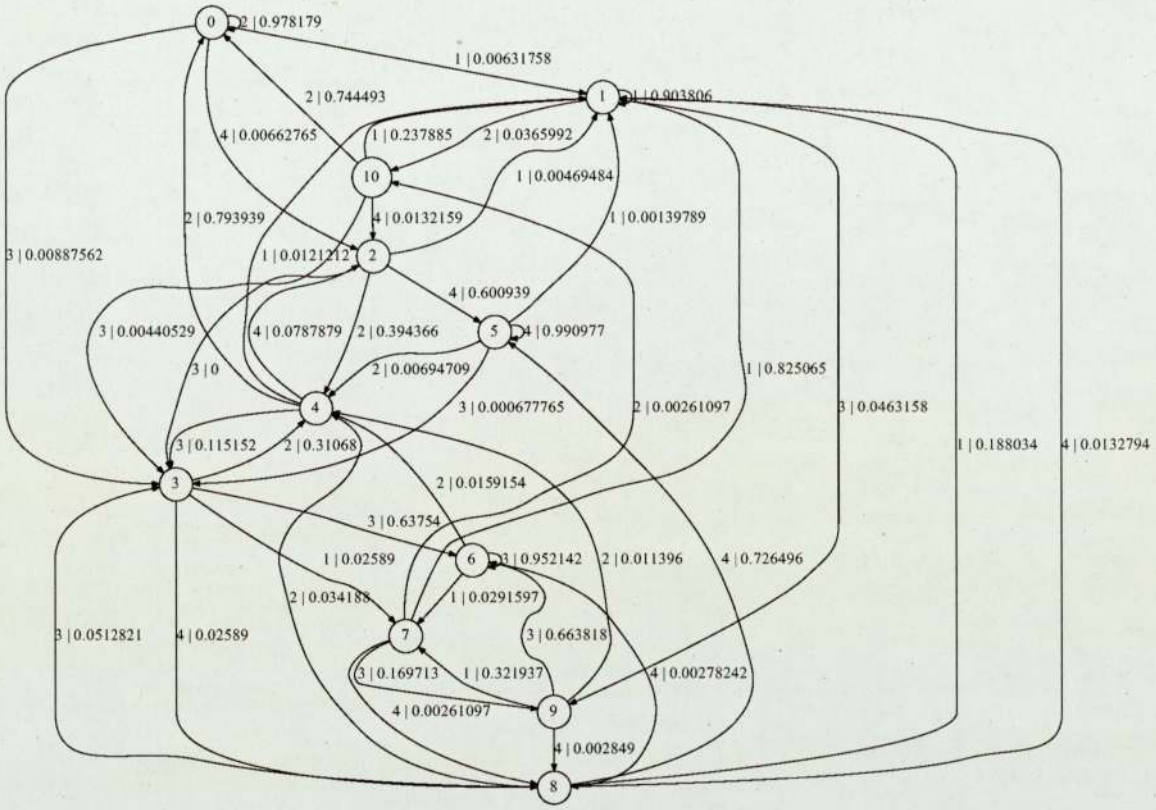


Figure 5.5: 11 States for time step 15ps

State (10) virtually explains the transition of other state to the metastable sate (5). The system spend less time at this state (10), before outputting a symbol (4) with about 50% probable, making it a transient state that route all other transition received by itself to the metastable state (5)

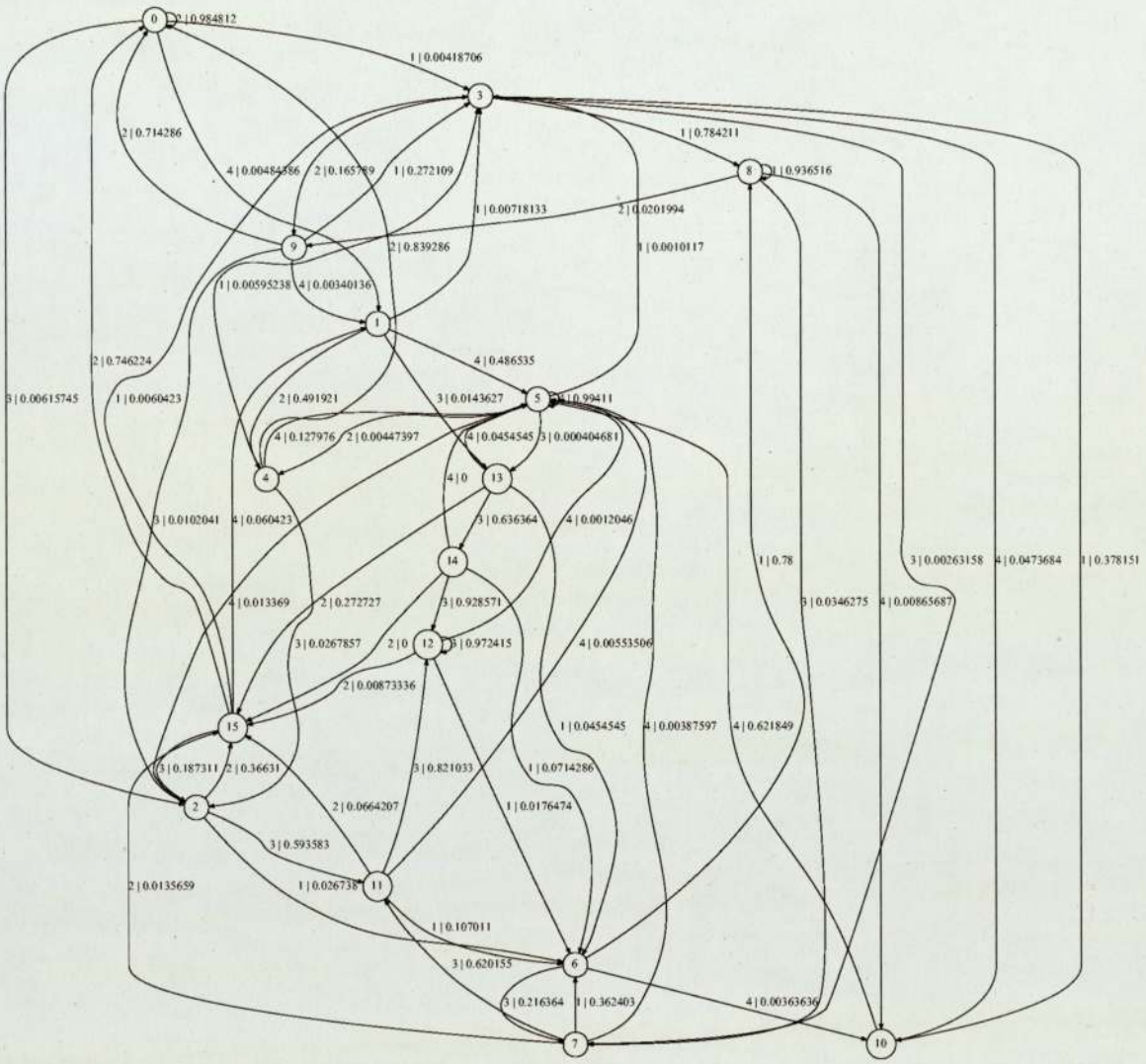


Figure 5.6: 16 States for time step 8ps

Figure 5.7 shows the transitions of the time at the list time $\triangle t^s$ by which the ϵ-machine converges for the amount of data used (i.e about 250000 data points). Also it require a deep analysis, which will be considered in the future.

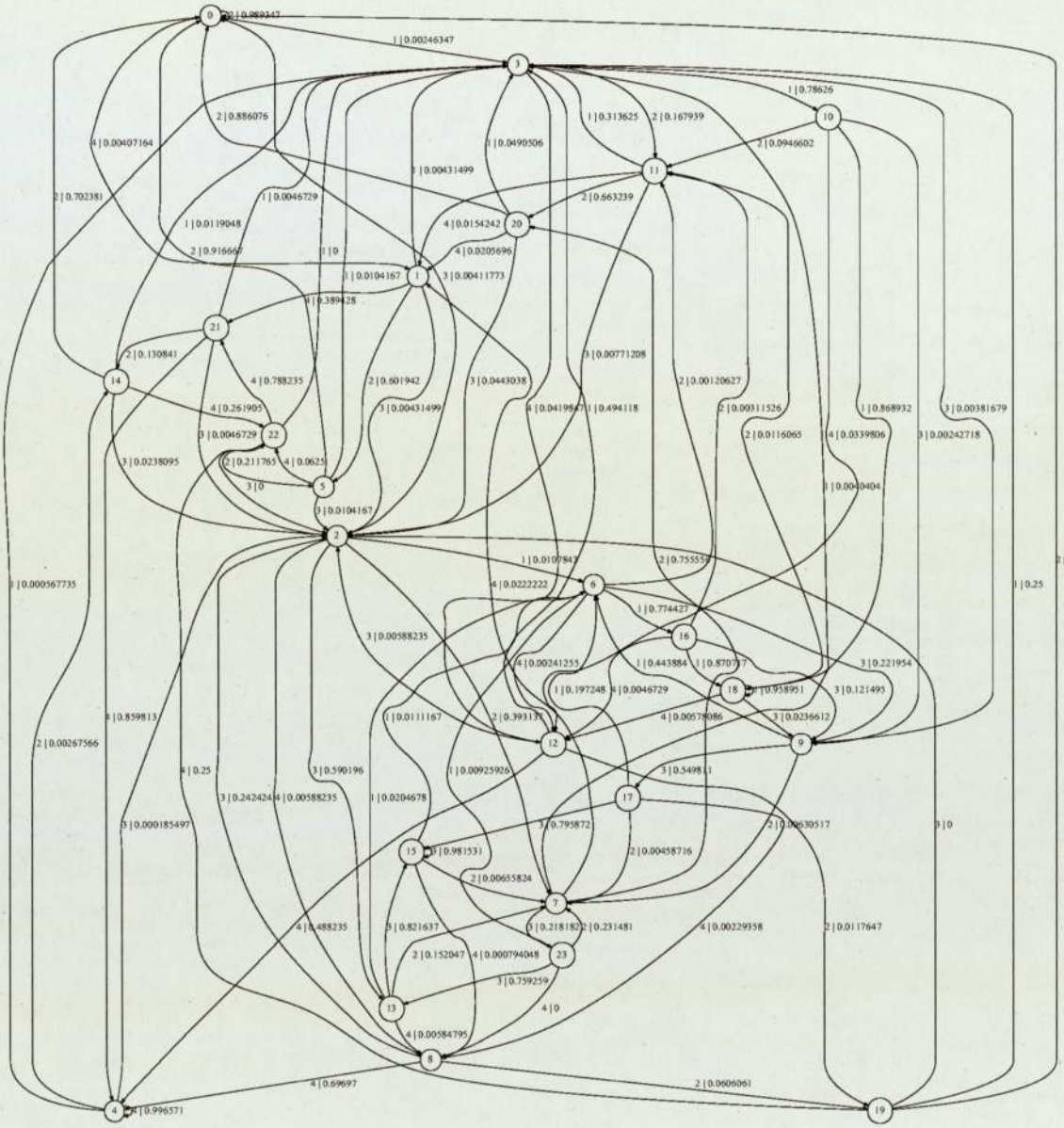


Figure 5.7: 24 States for time step 4ps

A tabular description of the used σ and l with the resulted states is show in Figure 5.8.

Time Step (ps)	Significant Level (σ)	Maximum Length (l)	States		
			Causal	Metastable	Transient
140	0.1000	1	4	4	0
100	0.0185	2	5	4	1
50	0.0080	2	6	4	2
20	0.0430	2	10	4	6
15	0.0430	2	11	4	7
8	0.0052	3	16	4	12
4	0.0034	3	24	4	20
0.4	-	-	Not Converged	Not Converged	Not Converged

Figure 5.8: A table showing the number of states given σ and l

5.3 Ramachandran Plot Diagram

The plots for each ϵ-machine are described by the Ramanchandran Plot. Clearly it is observed that the causal states are probably going to converge into the metastable states at some points. Also, it shows the likely metastable states that the causal states will converge to.

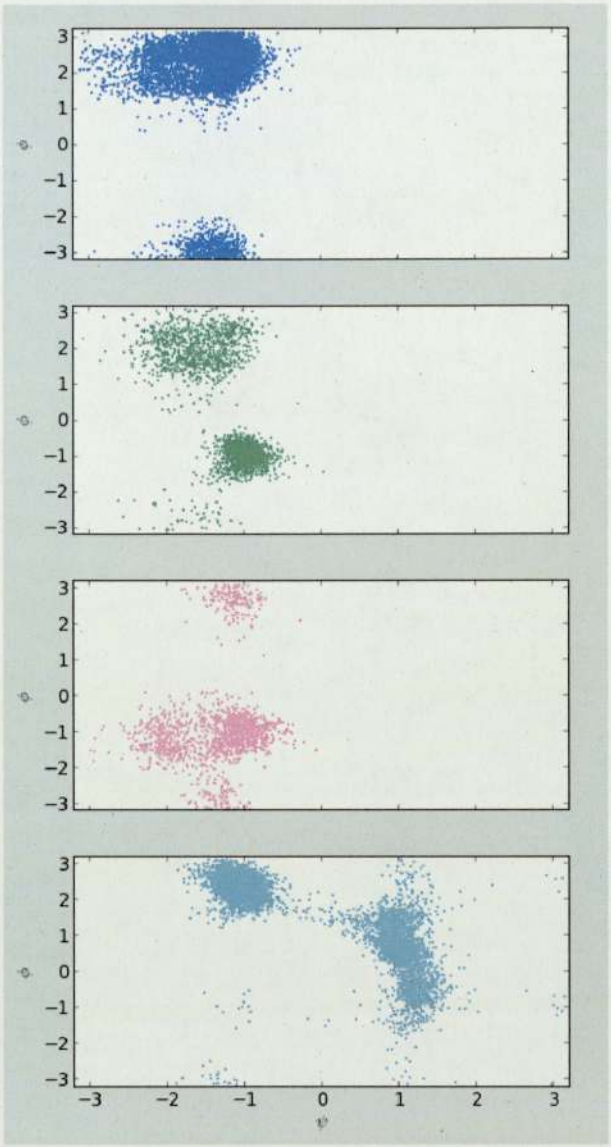


Figure 5.9: 4 States formed from 140ps

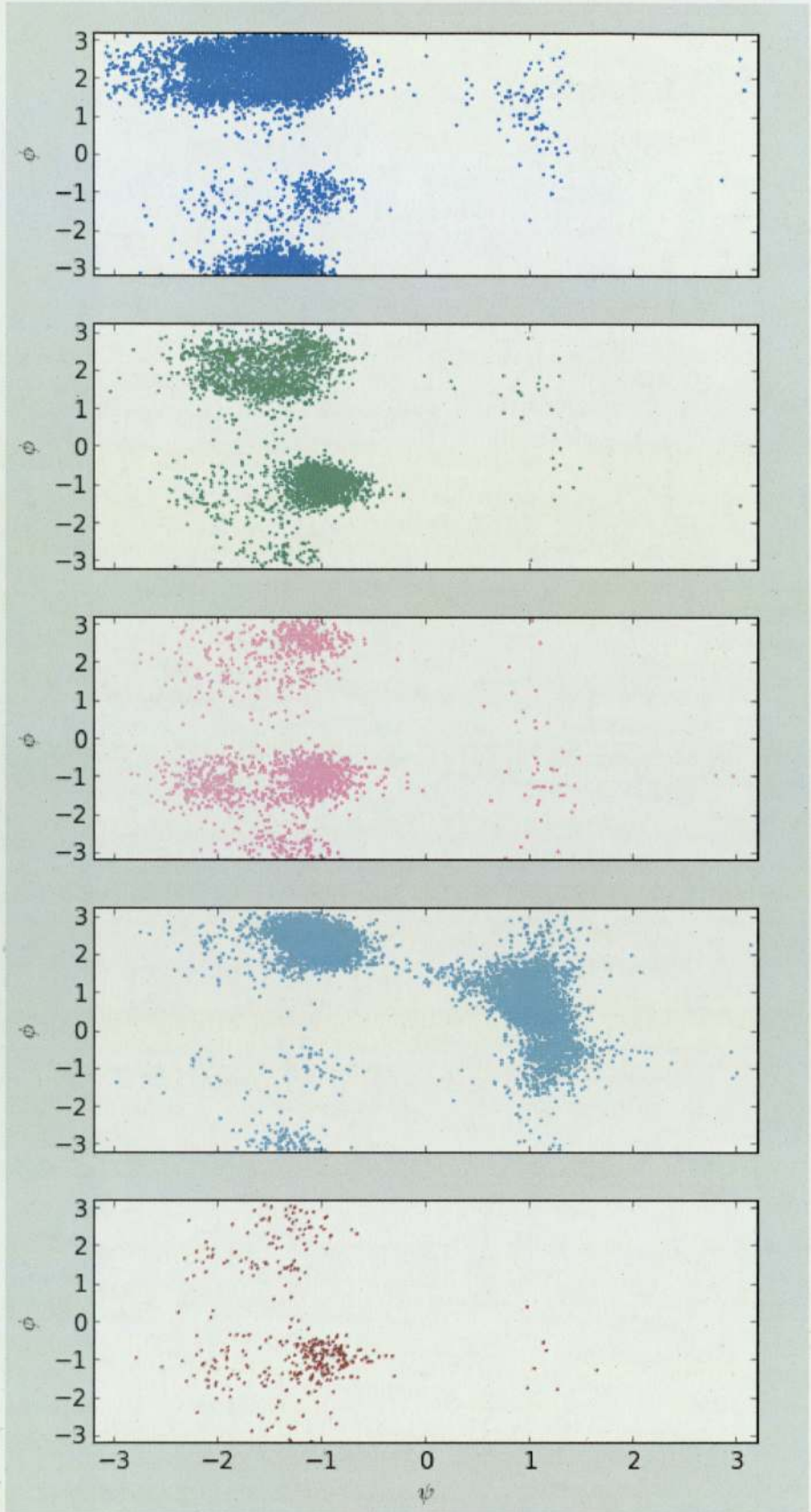


Figure 5.10: 5 States formed from 100ps

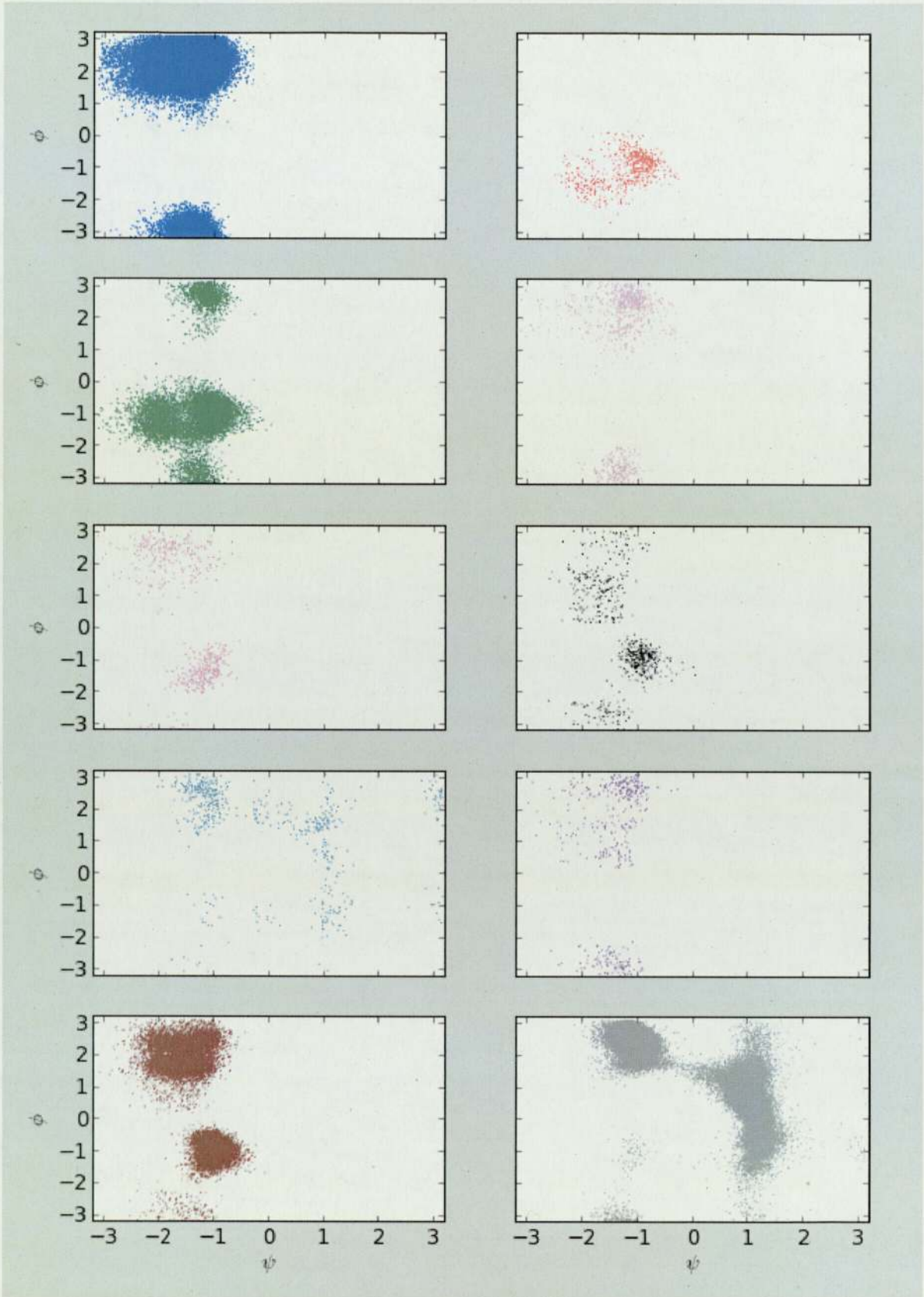


Figure 5.11: 10 States formed at 20ps

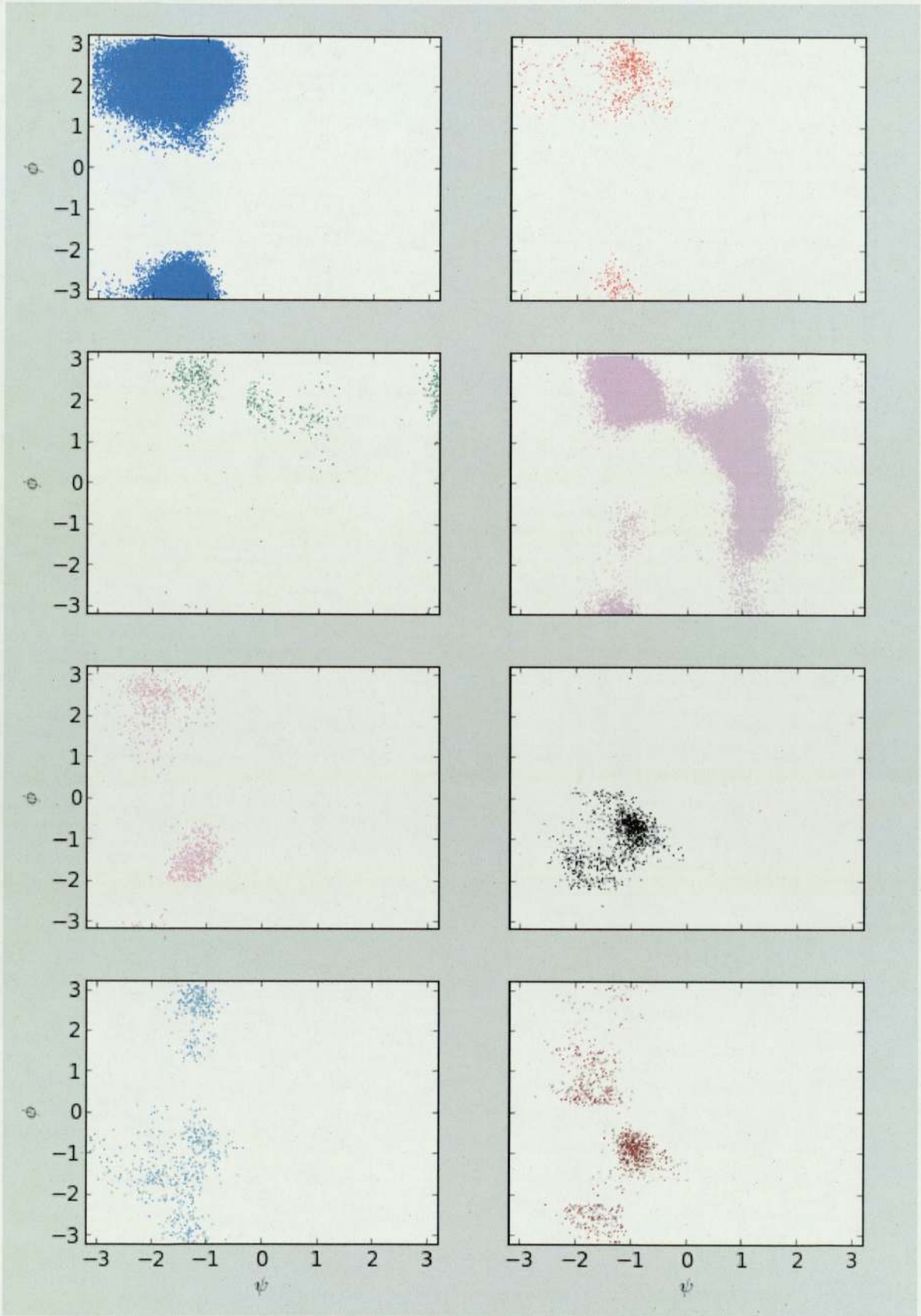


Figure 5.12: 1 to 8 States Plotted at 8ps

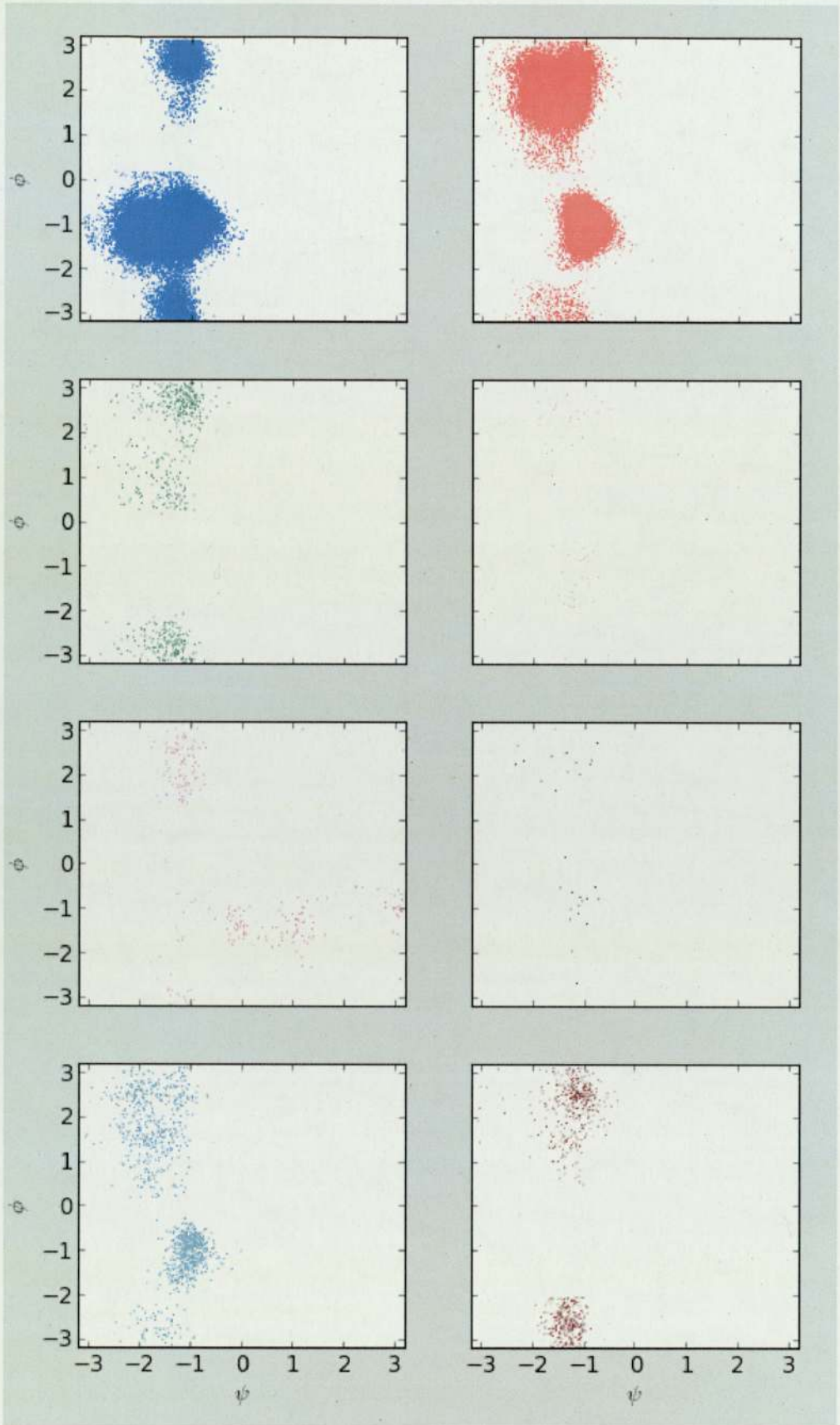


Figure 5.13: 9 to 16 States Plotted at 8ps

CHAPTER 6

CONCLUDING REMARKS

6.1 Summary

By observing the data on Ramachandran Plots, only 4 metastable clusters are shown, but with the help of CM very complicated dynamics (e.g. Figure 5.7) are recovered. The Markovian requirement is about 2.8ps, meaning it needs large time step. Finally, the ϵ-machine clearly splits the metastable moments from the transitions and allows smaller $\triangle t^s$.

6.2 Future Research Work

The work presented in this thesis can be extended in the following directions.

- Use more data for the simulation. Hence running for about 4μs compared to the 1μs used in this work.
- Investigate in details the transitions between states
- Detailed analyses of how the transitions relate to the clusters of the causal states as seen in Chapter 5.
- The anomalies of the conformational dynamics can also be investigated using

D-Markov [29]. This method combines Symbolic Dynamics, Finite State Automata and Pattern Recognition techniques to detect anomalies. In fact, is a subset or approximation technique of ϵ-machine.

REFERENCES

[1] C. Freudenrich, "How atoms works," 2013. [Online]. Available: http://www.howstuffworks.com/atom.htm

[2] P. Atkins, T. Overton, J. Rourke, M. Weller, and F. Armstrong, "Shriver and atkins inorganic chemistry," 2010.

[3] H. J. C. Berendsen, *Simulating the Physical World: Hierarchical Modeling from Quantum Mechanics to Fluid Dynamics)*, 1st ed. The Edinburgh Building, Cambridge CB2 2RU, UK: Cambridge University Press, 2007.

[4] C. Freudenrich, "Schaum's outline: Theory and problems of langrangian dynamics," 1967.

[5] L. Zhigilei, "Molecular dynamics: Introduction to atomistic simulations," 2010.

[6] D. Rapaport, *The Art of Molecular Dynamics Simulation*, 1st ed. Cambridge University Press, 2004.

[7] A. R. Leach, *Molecular Modelling: Principles and Applications*, 2nd ed. Henry Ling Limited, Dorset Press, Dorchester DT1 1HD: Prentice Hall, 2001.

[8] A. Hinchliffe, *Molecular Modelling for Beginners*, 1st ed. Atrium, Southern Gate, Chichester, West Sussex PQ19 8SQ, UK: John Wiley and Sons Ltd, 2003.

[9] J. P. Crutchfield, "The calculi of emergence: Computation, dynamics, and induction," vol. 75, 1993., p. 1154.

[10] S. Ruzhyska, M. N. Jaccobi, C. H. Jesen, and D. Nerukh, "Identification of metastable sates in peptide's dynamics," *Journal of Chemical Physics.*, vol. 133, no. 164102., October. 2010.

[11] C. H. Jensen, D. Nerukh, and R. C. Glen., "Sensitivity of peptide conformational dynamics on clustering of a classical molecular dynamic trajectory." *Journal of Chemical Physics.*, vol. 128, no. 115107., January. 2008.

[12] D. Nerukh, "Non-markov state model of peptide dynamics," *Journal of Molecular Liquids*, vol. 178, no. 084104, pp. 65–70, December. 2012.

[13] P. L. Freddolino, C. B. Harrison, Y. Liu, and K. Schulten, "Challenges in protein-folding simulations," *Nature Physics*, vol. 6, October. 2010. [Online]. Available: http://www.nature.com/naturephysics/DOI:10.1038/NPHYS1713.

[14] D. Voet and J. G. Voet, *Biochemistry*, 4th ed. John Wiley and Sons, INc, 2011.

[15] S. A. Mugilan, "Sequence, structure and conformational analysis of protein databases," *Journal of Advanced Bioinformatics Applications and Research*, pp. 183–192, 2011. [Online]. Available: http://bipublication.com/files/JABAR-v2i3201103.pdf

[16] M. Hong, J. D. Gross, W. Hu, and R. G. Griffin, "Determination of the peptide torsion angle ϕ by ^{15}n chemical shift and $^{13}c^{\alpha} -^{1} ha$ dipolar tensor correlation in solid-state mas nmr," *Journal of Magnetic Resonance*, pp. 163–177, 1998. [Online]. Available: http://web.mit.edu/fbml/cmr/griffin-group/pdf-publications/hong.jmr135.pdf

[17] G. E. Sims, I.-G. Choi, and S.-H. Kim, "Protein conformational space in higher order $\phi - \psi$ maps," 2005. [Online]. Available: http://www.pnas.org/cgi/doi/10.1073/pnas.0408746102

[18] K. A. Beauchamp, R. McGibbon, Y.-S. Lin, and V. S. Pande, "Simple few-state models reveal hidden complexity in protein folding," *Journal of Biophysics and Computational Biology.*, May. 2012. [Online]. Available: http://www.pnas.org/cgi/doi/10.1073/pnas.1201810109

[19] N. jie Deng, W. Dai, and R. M. Levy, "How kinetics within the unfolded state affects protein folding: An analysis based on markov state models and an ultra-long md trajectory," *Journal of Physical Chemistry*, May. 2013. [Online]. Available: http://dx.doi.org/10.1021/jp401962k

[20] H. Berendsen, D. van der Spoel, and R. van Drunen, "Gromacs: A message-passing parallel dolecular dynamics implementation," *Journal of Computer Physics Communications*, pp. 43–56, 1995.

[21] G. D. Group, "Gromacs flow chart," 2013. [Online]. Available: http://manual.gromacs.org/current/online/flow.html

[22] P. F. Batcho and T. Schlick, "Special stability advantages of position-verlet over velocity-verlet in multiple-time step integration," *Journal of Chemical Physics*, no. 9, 2001. [Online]. Available: http://www.biomath.nyu.edu/index/papdir/fulllengths/pap_2_83.pdf

[23] H. J. C. Berendsen and N. van der Vegt, *Journal of Physical Chemistry*, vol. 110, no. 35, pp. 17 616–17 626., 2006.

[24] D. P. Feldman, "Computational mechanics of classical spin systems," Ph.D. dissertation, University of California, 1998.

[25] D. Nerukh, C. H. Jensen, and R. C. Glen., "Identifying and correcting non–markov states in peptide conformational dynamics," *Journal of Chemical Physics.*, vol. 132, no. 084104., Febuary. 2010.

[26] D. Feldman, "A brief introduction to: Information theory, excess entropy and computational mechnanics," October. 2002. [Online]. Available: http://hornacek.coa.edu/dave

[27] D. P. Feldman and J. P. Crutchfield, "iscovering non-critical organization: statistical mechanical, information theoretic, and computational views of patterns in one-dimensional spin system," no. 98-04-026, 1998.

[28] C. R. Shalizi and K. L. Klinkner, "Blind construction of optimal nonlinear recursive predictors for discrete sequences," in *Uncertainty in Artificial Intelligence: Proceedings of the Twentieth Conference (UAI 2004)*, M. Chickering and J. Y. Halpern, Eds. Arlington, Virginia: AUAI Press, 2004, pp. 504–511. [Online]. Available: http://arxiv.org/abs/cs.LG/0406011

ASTON UNIVERSITY
LIBRARY & INFORMATION SERVICES

[29] A. Ray, "Symbolic dynamic analysis of complex systems for anomaly detection," *Journal of Signal Processing.*, vol. 84, pp. 1115–1130., 2004.

APPENDICES

APPENDIX A

PYTHON CODES FOR ANALYSIS

```
import fileinput
import numpy as np
import matplotlib.pyplot as plt


  # A function to convert degrees to radian
  # ========================================================
def ConvFromDegreeToRadian(x):
    return ((x * 3.14285714286) / 180)


  # A function to read the entries of a single file from
  # zero to nPoints
  # ========================================================
def GetListFromFile(fn, nPoints):
mainList = []
count = 0
```

```
# load each line from file
print('get each line from file...')
for eachLine in fileinput.input(fn):
temp = [0,0]
getLine = eachLine.rstrip("\n")
getListOfEachLine = getLine.split()
getFloatList = [float(i) for i in getListOfEachLine[0:2]]
x = getFloatList[0]; y = getFloatList[1];
temp[0] = x; temp[1] = y;
tempRad = map(ConvFromDegreeToRadian, temp)
mainList.append(tempRad)
count += 1
if count == nPoints:
    print (count)
    break
fileinput.close()
return mainList


  # A function that creates a list from a given file
  # with two index in the file to read from
  # ========================================================
def GetListFromFileByInd(fn,ind1, ind2):
    mainList = []
    # load each line from file
    print('get each line from file...')
    for eachLine in fileinput.input(fn):
        temp = [0,0]
        getLine = eachLine.rstrip("\n")
        getListOfEachLine = getLine.split()
```

```
        getFloatList = [float(i) for i in getListOfEachLine[0:4]]
        x = getFloatList[ind1]; y = getFloatList[ind2];
        temp[0] = x; temp[1] = y;
        tempRad = map(ConvFromDegreeToRadian, temp)
        mainList.append(tempRad)
    fileinput.close()
    return mainList


  # A function to that assign symbols to respictive partitions
  # given the dihedral angles of the residues
  # ========================================================
def GetStates(proY,alaX, alaY):
    proState = 0
    alaState = 0
    if (proY > -2.0 and proY < 0.4):
        proState = 2              # B1
    else:
        proState = 1              # A1
    # ALA-3
    if (alaX > -0.3):
        alaState = 3          # C2
    else:
        if (alaY > -2.2 and alaY < 0.2):
            alaState = 2          # B2
        else:
            alaState = 1          # A2
    return proState, alaState
```

```
    # A function to that generates all the sequence of symbols
    # for certain time stem t as nStep
    # ======================================================
def GenerateSymbolsFromProAlaFile(lstPro,lstAla,outFn, nStep):
    lenProOrAla = len(lstPro)

    # Symbols and counts
    count = 0
    symbols = ""
    psiPhiSymbols = ""
    tempStep = nStep
    iterator = nStep

    for i in range(lenProOrAla):
        # Get the Proline coordinates
        proX = (lstPro[i])[0]
        proY = (lstPro[i])[1]

        # Get the Alanine coordinates
        alaX = (lstAla[i])[0]
        alaY = (lstAla[i])[1]
        # States
        stateProAla = 0
        count += 1
        if count == iterator:
            proState, alaState = GetStates(proY,alaX, alaY)
            #print alaState ,  proState
            #break
            # A1B2 + B1B2
```

```
        if ((proState == 1 and alaState == 2) or
        (proState == 2
        and alaState == 2)):
            stateProAla = 1
        # A1A2
        elif ( proState == 1 and alaState == 1 ):
            stateProAla = 2
        # B1A2
        elif ( proState == 2 and alaState == 1 ):
            stateProAla = 3
        # A1C2 (B1C2 was not populated)
        else:
            stateProAla = 4
        # concat symbols
        symbols=symbols+str(stateProAla)
        psiPhiSymbols += str(proX)  +" "+ str(proY)  +
        " "+ str(alaX)
        +" "+ str(alaY)  +" "+ str(stateProAla) + "\n"
        # concat symbols
        iterator += tempStep
# Output strings
fPsiPhiSymbols = str(outFn)+"_psi_phi_n"+str(tempStep)+
str(".sym.dat")
flName = str(outFn)+"_n"+str(tempStep)+str(".sym.dat")
writeToFile(flName,symbols, "creating nstep symbol
data file..")
writeToFile(fPsiPhiSymbols,psiPhiSymbols,"creating
nstep phi, psi and
symbol data file..")
```

```
  # Writing to File Function
  # ========================================================
def writeToFile(fn,strVal,strComment):
    outFile = open(str(fn),'w')
    # write to file
    print (strComment)
    outFile.write(strVal)
    # close file
    outFile.close()

  # A function to plot the Ramamchandran given certain
  # time stem t as nPoint
  # ========================================================
def PlotList(vpalProFile,vpalAlaFile,nPoints):
    lstPro = GetListFromFile(vpalProFile,nPoints)
    lstAla = GetListFromFile(vpalAlaFile,nPoints)
    # Transpose the list of
    transPro = zip(*lstPro)
    transAla = zip(*lstAla)
    # Create a figure
    fig = plt.figure()
    # Create a sub plot for Pro-2
    pltPro = fig.add_subplot(121)
    pltPro.scatter(transPro[0], transPro[1],color='maroon',s=1,
    edgecolor='none')
    pltPro.set_ylabel('Phi')
    pltPro.set_xlabel('Psi')
    pltPro.set_title('Ramachandran plot of Proline')
```

```
    pltPro.axis([-3.20,3.20,-3.20,3.20])
    plt.grid(True)
    # Create a sub plot for Pro-2
    pltAla = fig.add_subplot(122)
    pltAla.scatter(transAla[0], transAla[1],color='blue',s=1,
    edgecolor='none')
    pltAla.set_xlabel('Psi')
    pltAla.set_title('Ramachandran plot of Alanine')
    pltAla.axis([-3.20,3.20,-3.20,3.20])
    plt.grid(True)

    plt.savefig('pro_ala_plot_'+str(nPoints)+'.png', dpi=300)
    plt.show()

  # Plot a function to plot all the symbols given certain
  # time stem t as nPoint
  # ======================================================
def PlotAllPsiPhibasedOnSymbols(fn,nPoints):
    count = 0
    # load each line from file
    colors = ['blue', 'green', 'magenta', 'cyan']

    print ('open and retrieve file data...')
    # Create a figure
    f, ((pltPro1, pltAla1), (pltPro2, pltAla2),(pltPro3, pltAla3),
     (pltPro4, pltAla4)) = plt.subplots(4, 2, sharex='col',
     sharey='row')

    for eachLine in fileinput.input(fn):
```

```
getLine = eachLine.rstrip("\n")
getListOfEachLine = getLine.split()
getFloatList = [float(i) for i in getListOfEachLine[0:4]]

proX = getFloatList[0]; proY = getFloatList[1];
alaX = getFloatList[2]; alaY = getFloatList[3];
symbol = int(getListOfEachLine[4])

if symbol == 1:
    pltPro1.scatter(proX, proY,color=colors[0],s=1,
    edgecolor='none')
    pltAla1.scatter(alaX, alaY,color=colors[0],s=1,
    edgecolor='none')
if symbol == 2:
    pltPro2.scatter(proX, proY,color=colors[1],s=1,
    edgecolor='none')
    pltAla2.scatter(alaX, alaY,color=colors[1],s=1,
    edgecolor='none')
if symbol == 3:
    pltPro3.scatter(proX, proY,color=colors[2],s=1,
    edgecolor='none')
    pltAla3.scatter(alaX, alaY,color=colors[2],s=1,
    edgecolor='none')
if symbol == 4:
    pltPro4.scatter(proX, proY,color=colors[3],s=1,
    edgecolor='none')
    pltAla4.scatter(alaX, alaY,color=colors[3],s=1,
    edgecolor='none')
count += 1
```

```
        print (count)
    print ('finish and close the file...')
    fileinput.close()
    pltPro1.set_ylabel('Phi')
    pltPro2.set_ylabel('Phi')
    pltPro3.set_ylabel('Phi')
    pltPro4.set_ylabel('Phi')
    pltPro4.set_xlabel('Psi')
    pltAla4.set_xlabel('Psi')
    plt.axis([-3.20,3.20,-3.20,3.20])
    plt.savefig('psi_phi_sym_plot_'+str(nPoints)+'.png', dpi=300)
    plt.show()

  # This function is use to create the matrix given the file number
def GenerateMatrixes(nBegin):
os.system("cat step_"+str(nBegin)+"m.sym.s_1.dat | gawk -f
 s_1-matrix_txt.awk > step_"+str(nBegin)+"m.sym.s_1.matrix.dat")

  # This function creates the plateau from different symbols files
  # starting at nBegin to nEnd by nStep by using the CSSR program
def PlateusAndMatrixes(nBegin,nEnd,nStep):

while (nBegin <= nEnd):
print 'file number: '+str(nBegin)
s = 0.1
while(1):
l = 1
while(l <= 10):
os.system("./CSSR.exe vpal_alphabet.dat
```

```
 step_"+str(nBegin)+"m.sym.dat "+str(l)+" -s  "+str(s)+" |
   gawk \'NF>10{print "+str(s)+", "+str(l)+", $12}\'>>
    step_"+str(nBegin)+"m.sym.s_l.dat")
l += 1
if s < 0.001:
break;
s = s*0.9
GenerateMatrixes(nBegin)
nBegin += nStep

  # This function is use to create the matrix of several files
  # from nBegin to nEnd at nStep
  # ========================================================
def Matrixes(nBegin,nEnd,nStep):
while (nBegin <= nEnd):
print 'file number: '+str(nBegin)
GenerateMatrixes(nBegin)
nBegin += nStep

# Generate the Plateaus and Their respective matrixes
# ========================================================
# Sample Run:
#PlateusAndMatrixes(1,5,1);
#Matrixes(40,1400,20)
#Matrixes(150,500,10)
#Matrixes(4,5,1)
```

APPENDIX B: MODIFICATIONS DONE IN GROMACS

The following are the modification done on GROMACS package version 4.5.5 in other to write the velocities and positions of some atoms into a generated file, "dn.dat"

File Name: /gromacs-4.5.5/include/types/inputrec.h

==========

1) Line: 287

Modifying the structure, t_inputrec, to enter this input parameter:

```
int intdn; /* Input parameter for creating and
writing to file dn.dat */
```

File Name: /gromacs-4.5.5/src/kernel/readir.c

==========

2) Line: 1163

Modifying the function get_ir() and enter the following line:

```
ITYPE ("la_trr", ir->intdn,  0);
```

Finally, a parameter should be entered in the .mdp file, like this:

```
la_trr 0 or 1
0 if no "dn.dat" file should be created,  but 1 to create "dn.dat"
file
File Name: /gromacs-4.5.5/include/mdrun.h
================================================
1) Line: 686
preprocessors decleration:

void la_write_traj( FILE *trrFile, int cmdArg, rvec *x, rvec *v);

2) Line: 136 - 137
Modify the gmx_mdoutf_t struct and enter these lines
  FILE *la_Trajectory;
  int cmdArg;

File Name: /gromacs-4.5.5/src/mdlib/stat.c
================================================
1) Line:  546 - 555
void la_write_traj( FILE *trrFile, int cmdArg, rvec *x, rvec
*v, const char *strSection)
{
  if ( cmdArg == 1 )
  {
    if(trrFile){
      fprintf(trrFile, "(1) %.5f %.5f %.5f %.5f %.5f %.5f
       (2) %.5f %.5f %.5f %.5f %.5f %.5f\n", x[0][0], x[0][1],
       x[0][2], v[0][0], v[0][1], v[0][2], x[1][0], x[1][1],
       x[1][2], v[1][0], v[1][1], v[1][2] );
    }
```

```
  }
}
```

2) Line: 686

Modification of the write_traj() function:

```
  la_write_traj(of->la_Trajectory,of->cmdArg,vecX,vecV);
```

File Name: /gromacs-4.5.5/src/kernel/md.c

==========

1) Line: 273 to 280

```
    if(ir->intdn == 1)
    {
        outf->cmdArg = 1;

        // Create a File
        outf->la_Trajectory = fopen ( "dn.dat", "a+" );
    }
```

2) Line: 1847 to 1850

```
   if(outf->la_Trajectory)
   {
        fclose(outf->la_Trajectory);// CLOSE DN CREATED FILE
    }
```