Molecular Dynamics Simulation of Polymer Chain

Zelalem Nigussa Urgessa (zola@aims.ac.za)
African Institute for Mathematical Sciences (AIMS)

Supervised by Dr. Kristian Müller-Nedebock (PhD)
University of Stellenbosch

June 8, 2007

Abstract

This essay details Molecular Dynamics simulation of a real polymer chain (in the presence of excluded volume effect) and provides comparison with the theoretical result. The chain was modeled as a freely flexible bead-rod chain interacting through 6-12 Lennard-Jones and Finitely Extensible Non-Linear Elastic (FENE) potentials. The Lennard-Jones potential models the interaction of any two monomers separated by a distance less than 2.5σ and FENE potential is between adjacent monomers. The physical quantities measured by the simulation are mean square end-to-end distance $\langle R^2 \rangle$ and the radius of gyration $\langle R_g^2 \rangle$

The simulation was carried out for a polymer consisting of 5 to 50 monomers in steps of 5, and the result for $\langle R^2 \rangle$ and $\langle R_g^2 \rangle$ versus N is plotted. The graph shows that the mean square end-to-end distance is proportional to $N^{2\nu}$, where ν is found to be 0.588 and the radius of gyration is found to be proportional to $\frac{1}{6}\langle R^2 \rangle$. These values are in good agreement with the theoretical result. To obtain a best fit between the simulation result and the theoretical values one has to consider a higher number of monomers.

KEY WORDS: *Molecular Dynamics, Verlet Algorithm, End-to-End Distance and Radius of Gyration*

Contents

1. Physics of Polymers

1.1 Introduction

Polymers are long chain macromolecules, made of repeating monomeric units, with a large molecular weight. The name polymer means many parts, and is derived from the Greek word: polus- (many), meros- (parts). The smallest part of a polymer is called a monomer [1]. A monomer is a molecule, which combines with other molecules of the same or different type to form a polymer. Polymers composed of few monomer units are known as Oligomer. Polymers currently constitute one of the most commonly used class of material. The diversity of this material is seen in its application in the food, plastic, defence, and pharmaceutical. Polymers can be divided into two groups, synthetic polymers and biopolymers. Synthetic polymers are known as man-made polymers because of our ability to control the synthesis and growth of the macromolecule. The weight and size vary from polymer to polymer. Although the fundamental property of bulk polymers is the degree of polymerisation, the physical structure of the chain is also an important factor that determines the macroscopic properties [2]. For example end-to-end distance $\vec{R}$ of a given polymer is one of the physical properties that determines how long a given polymer is.

This essay is organised as follows: In Chapter one the theoretical bases for understanding of the polymer end-to-end distance and its probability distribution for ideal and real polymer chain are presented. Chapter two presents a brief description of molecular dynamics (MD) simulation and techniques used in MD simulation. In the third Chapter the simulation procedures is discussed. Finally the fourth Chapter presents the result followed by discussion and conclusion.

1.2 Physical Properties of Polymer Chain

1.2.1 Polymer End-to-End Distance

Polymer chains consist of atoms that are connected by covalent bonds. These covalent bonds are usually capable of rotating, creating numerous possible conformations. It is, therefore, difficult to define the shape of a polymer chain. It is for this reason that the concept of end-to-end distance, $\vec{R}$, was introduced. The end-to-end distance is the distance between the two terminal atoms of the polymer chain. The length of $\vec{R}$ depends on the solution in which the polymer is placed [2].

The statistical approach to polymer physics is based on an analogy between a polymer and either a brownian motion, or some other type of random walk. The simplest possible polymer model is presented by the ideal chain, corresponds to homogeneous random walk. There are several different ways of modelling a polymer. The most widely used method is the bead-rod model [3] shown in Fig 1.1.

In this type of model we imagine that the monomers are connected to each another in a single chain by a chemical bond, hence their mutual separation is almost constant. However, each

reapting units (segment bond unit) can rotate freely with respect to the other. To find the end-to-end distance of this ideal polymer chain let us consider there to be N monomers connected to one other by a constant bond length b. Since our ideal polymer chain has many internal degree of freedom, the distribution of each monomer is isotropic in space. Hence a configuration of a macromolecule consisting of N identical repeating units is represented by a randon walk of N steps, each of a length b [2]. Therefore, we can assume that there is no correlation between the direction that different bonds take, and that all direction have the same probability. This means that each step of bond formation between any two monomers is statistically independent. In reality this is not true. There is in fact an interaction between each pair of monomers due to dipole-dipole interaction when they come close and weak Van der Waals interaction when the are apart. Moreover, there is interaction due to the hydrogen bond between each pair of adjacent monomers. We will consider this effect in the case of a real polymers.

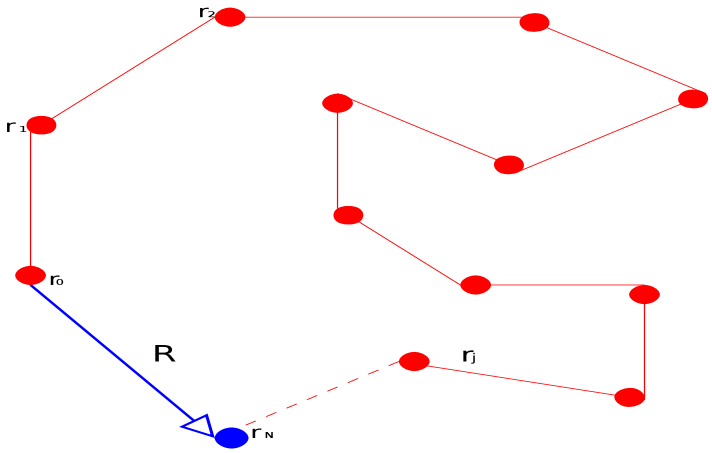

Figure 1.1: The bead-rod model of polymer chain

To find the polymer end-to-end distance for the above model, we denote the coordinates of element j by $\vec{r}_j$, assuming that the chain is composed of $N+1$ elements joined successively and the elements are numbered $0, 1, 2, ..., N$ from one end to the other.

$$\vec{R} = \vec{r}_b - \vec{r}_a = \sum_{j=0}^{N} \vec{r}_j, \qquad j = 0, 1, 2, \ldots, N, \tag{1.1}$$

where $\vec{r}_b$ and $\vec{r}_a$ are the final and initial position vectors of the chain respectively. Because each $\vec{r}_j$ is uncorrelated we must have

$$\langle \vec{R} \rangle = \sum_{j=0}^{N} \langle \vec{r}_j \rangle = 0. \tag{1.2}$$

To understand what is this means imagine a single particle undertaking a random walk. For simplicity assume this is a walk in one dimension (along a straight line). Each consecutive step may be either forward or back, we cannot predict which, though we can say the particle equally

likely to step forward as to step back. (A drunk man comes to mind!) From a given starting position, what distance is the particle likely to travel after many steps? This can be determined simply by adding together the steps, taking into account the fact that steps backward subtract from the total, while steps forward add to the total. Since both forward and backward steps are equally probable, we come to the surprising conclusion that the probable distance travelled sums up to zero!

In the case of polymer chain, this means that the probability of the end-to-end vector being R is the same as for being $-R$, so that the two contributions cancel out. If however, instead of adding the distance of each step we added the square of the distance, we realise that we will always be adding positive quantities to the total. In this case the sum will be some positive number, which grows larger with every step. This obviously gives a better idea about the distance (squared in this case) that a particle moves. If we assume each step happens at regular time intervals, we can easily see how the square distance grows with time. This can be seen as follows

$$\langle \vec{R}^2 \rangle = \sum_{n=0}^{N} \sum_{m=0}^{N} \langle \vec{r}_n \cdot \vec{r}_m \rangle = \sum_{m=0}^{N} \langle \vec{r}_n^2 \rangle \delta_{m,n} = N\ b^2. \tag{1.3}$$

This shows that the average size of a given ideal polymer chain of N monomers is proportional to $\sqrt{N}$ [4].

To find a detailed description of the end-to-end distance on the number of segments N let us use the concept of probability distribution. For simplicity, let us first consider an ideal polymer chain. In the case of this model, the polymer is consided as a freely connected bead-rod chain where two or more monomers can occupy the same physical position.

1.2.2 Probability Distribution of End-to-End Distance for Ideal Polymer Chain

The freely-joined chain is the simplest model of a polymer. In this model, fixed length polymer segments are linearly connected, and all bond and torsion angles are equiprobable. Therefore, the distribution of a given ideal polymer chain can be considered as a randon walk which follow a Gaussian distribution. To find the expression for the probability distribution for such system, it is obvious to use the concept of Markov conditional probability, which is defined as follows:

A discrete random walk of $N \in \mathbb{Z}^+$ step in a space X is a sequence of vectors $\vec{r}_0, \vec{r}_1, \vec{r}_2 \ldots, \vec{r}_N$ where $\vec{r}_i \in X, \forall_i \in 0, 1, 2, \ldots, N$. Each step $\vec{r}_j - \vec{r}_{j+1}$ is chosen at random and independent of the preceding step according to some distribution [5].

Using this definition, we can discretize the chain as a set of vectors $\vec{r}_0, \vec{r}_1, \vec{r}_2, \ldots, \vec{r}_{N-1}, \vec{r}_N$ at the corresponding segments $0, 1, 2, \ldots, N$. Therefore, the conditional probability distribution $P(N, R|N_0, R_0)$ can be written as [6]

$$P(R, N|R_0, N_0) = \int P(R, N|R_0, N_0) P(R_0, N_0) dR. \tag{1.4}$$

In words this is to say that knowing the the probability $P(N_0, R_0)$ when N_0 monomers is placed we can find the probability the probability distribution $P(R, N)$.

Using the above definition and the concept of path integral [7], Equation 1.4 may be written as

$$P(R, N|R_0, N_0) = \int \dots \int P(R_N, N|R_{N-1}, N-1) \dots P(R_2, N_2|R_1, N_1) P(R_1, N_1|R_0, N_0) dR_{N-1} \dots dR_1. \tag{1.5}$$

As discussed above each step of the random walk is statistically independent, so that we can write the total probability distribution as a product of each discrete steps as follows; To this end let us write the above equation in terms of $\vec{r}_j$. From equation (1.1) we have $\vec{R} = \vec{r}_b - \vec{r}_a = \sum_{j=0}^{N} \vec{r}_j$. Therefore,

$$P_N(\vec{r}_b - \vec{r}_a) = \prod_{n=0}^{N} \left[\int d^3(\vec{r}_{n+1} - \vec{r}_n) \frac{1}{4\pi b^2} \delta\left(|\vec{r}_{n+1} - \vec{r}_n| - b\right) \right] \delta^3 \left((\vec{r}_b - \vec{r}_a) - \sum_{n=0}^{N} (\vec{r}_{n+1} - \vec{r}_n) \right). \tag{1.6}$$

Let $|\vec{r}_{n+1} - \vec{r}_n| = \Delta\vec{r}_n$

$$P_N(\vec{r}_b - \vec{r}_a) = \prod_{n=0}^{N} \left[\int d^3(\Delta\vec{r}_n) \frac{1}{4\pi b^2} \delta\left(|\Delta\vec{r}_n| - b\right) \right] \delta^3 \left((\vec{r}_b - \vec{r}_a) - \sum_{n=0}^{N} (\Delta\vec{r}_n) \right). \tag{1.7}$$

In this expression, the last delta function ensures that the vectors $\Delta\vec{r}_n$ of the chain element add up correctly to the distance vector $(\vec{r}_b - \vec{r}_a)$. The δ-function under the product enforces the fixed length of the chain element b. The normalisation factor $\frac{1}{4\pi b^2}$ arises from the fact that distributions are isotropic and spherically symmetric. This means that:

$$P_1(\Delta\vec{r}) = \frac{1}{4\pi b^2} \delta(|\Delta\vec{r}| - b) \qquad \Leftrightarrow \qquad \int d^3\vec{r}_b P_1(\vec{r}_b - \vec{r}_a) = 1.$$

The same normalisation holds for all N:

$$\int d^3\vec{r}_b P_N(\vec{r}_b - \vec{r}_a) = 1.$$

The second δ-function in Equation (1.6) can be decomposed into Fourier components as follows:

$$\delta^3 \left((\vec{r}_b - \vec{r}_a) - \sum_{n=0}^{N} (\Delta\vec{r}_n) \right) = \int \frac{d^3k}{(2\pi)^3} e^{ik(\vec{r}_b - \vec{r}_a) - ik\sum_{n=0}^{N} \Delta\vec{r}_n}.$$

Subsituting this expression in Equation (1.6) we can have

$$P_N(\vec{r}_b - \vec{r}_a) = \int \frac{d^3k}{(2\pi)^3} e^{ik(\vec{r}_b - \vec{r}_a)} \prod_{n=0}^{N} \left[\int d^3(\Delta\vec{r}_n) \frac{1}{4\pi b^2} \delta(|\Delta\vec{r}_n| - b) e^{-ik\Delta\vec{r}_n} \right]. \tag{1.8}$$

Let us define:

$$\widetilde{P}_N(k) = \prod_{n=0}^{N} \left[\int d^3(\Delta\vec{r}_n) \frac{1}{4\pi b^2} \delta(|\Delta\vec{r}_n| - b) e^{-ik\Delta\vec{r}_n} \right].$$

So that Equation (1.8) becomes;

$$P_N(\vec{r}_b - \vec{r}_a) = \int \frac{d^3k}{(2\pi)^3} e^{ik(\vec{r}_b - \vec{r}_a)} \widetilde{P}_N(k) \tag{1.9}$$

To find an appropriate expression for $P_N(\vec{r}_b - \vec{r}_a)$, we have to find an expression for $\widetilde{P}_N(k)$. If we expand the above expression for $\widetilde{P}_N(k)$ we can see that the Fourier transform $\widetilde{P}_N(k)$ factors into a product of N Fourier-transformed one-link probabilities, so we can easily see that:

$$\widetilde{P}_N(k) = [\widetilde{P}_1(k)]^N. \tag{1.10}$$

$P_1(k)$ can be calculated as follows

$$\widetilde{P}_1(k) = \int d^3(\Delta\vec{r}_n) \frac{1}{4\pi b^2} \delta(|\Delta\vec{r}_n| - b) e^{-ik\Delta\vec{r}_n}.$$

Using three dimensional integration for $0 \leq \Delta\vec{r}_n \leq \infty, \quad -1 \leq \cos\theta \leq 1, \quad 0 \leq \phi \leq 2\pi$, and $d^3(\Delta\vec{r}_n) = (\Delta\vec{r}_n)^2 d(\Delta\vec{r}_n) d(\cos\theta) d\phi$ where θ is the angle between k and $\Delta\vec{r}_n$,

$$\widetilde{P}_1(k) = \int_0^\infty (\Delta\vec{r}_n)^2 d(\Delta\vec{r}_n) \int_{-1}^{1} d(\cos\theta) \int_0^{2\pi} d\phi \frac{1}{4\pi b^2} \delta(|\Delta\vec{r}_n| - b) e^{-ik\Delta\vec{r}_n \cos\theta}.$$

Using the definition of delta function $\Delta\vec{r}_n = b$, and integral over ϕ gives 2π, thus

$$\widetilde{P}_1(k) = 2\pi b^2 \int_{-1}^{1} d(\cos\theta) \frac{1}{4\pi b^2} e^{-ik\Delta\vec{r}_n \cos\theta}.$$

If we make a substitution $u = \cos\theta$;

$$\widetilde{P}_1(k) = 2\pi b^2 \frac{1}{-ikb} \frac{1}{4\pi b^2} \left[e^{-ikbu} \right]_{-1}^{1} = \frac{\sin(kb)}{kb}. \tag{1.11}$$

Using equation 1.11 and 1.10, equation 1.9 becomes;

$$P_N(R) = \int \frac{d^3k}{(2\pi)^3} e^{ikR} \left[\frac{\sin(kb)}{kb} \right]^N, \tag{1.12}$$

where we have subsituted $\vec{r}_b - \vec{r}_a$ as R

But for any isotropic Fourier integral $f(k)$ that depend only on the magnitude of k but not on direction of k, we can write a general result for isotropic Fourier integrals in three dimensions of a form shown below as[8]:

$$\int_{-\infty}^{\infty} f(k)e^{ikR}d^3k = \frac{4\pi}{R}\int_0^{\infty} k\sin(Rk)f(k)dk.$$

In our case also the distribution function only depends on the length of k. Therefore,

$$P_N(R) = \frac{4\pi}{R}\int_0^{\infty}\frac{dk}{(2\pi)^3}k\sin(Rk)\left[\frac{\sin(kb)}{kb}\right]^N = \frac{1}{2\pi^2 R}\int_0^{\infty} dk\ k\sin(Rk)\left[\frac{\sin(kb)}{kb}\right]^N. \quad (1.13)$$

This equation is an exact integral of a probability distribution for ideal polymer chain [7].

Asymptotic Solution

The asymptotic behaviour of the probability distribution for $N \gg 1$ can be found from the Fourier transform of Equation 1.11 by expanding $\sin(kb)$ for kb $\ll 1$ in Taylor series as:

$$\widetilde{P}_1(k) = \frac{\sin(kb)}{kb} = \frac{1}{kb}\sum_{n=0}^{\infty}(-1)^n\frac{(kb)^{2n+1}}{(2n+1)!}$$
$$\approx 1 - \frac{1}{6}(kb)^2 + O(k^4b^4) \approx e^{\frac{-1}{6}k^2b^2}.$$

Therefore,

$$[\widetilde{P}_1(k)]^N \approx e^{\frac{-N}{6}k^2b^2}.$$

If we subsitute this value in equation 1.12 we find

$$P_N(R) = \int\frac{d^3k}{(2\pi)^3}e^{ikR}e^{\frac{-N}{6}k^2b^2} = \frac{1}{(2\pi)^3}\int d^3\ ke^{\frac{-N}{6}k^2b^2+ikR} \quad (1.14)$$

This is an integral of standard Gaussian form

$$\int_{-\infty}^{\infty} e^{-mx^2+nx}d^3x = \left(\frac{\pi}{m}\right)^{\frac{3}{2}}e^{-\frac{n^2}{4m}}.$$

By comprision, we can find $m = \frac{-N}{6}b^2$ and $n = iR$ in our integral.

Therefore,

$$P_N(R) = \frac{1}{(2\pi)^3}\left(\frac{6\pi}{Nb^2}\right)^{\frac{3}{2}}e^{-\frac{6R^2}{4Nb^2}} = \left(\frac{3}{2\pi Nb^2}\right)^{\frac{3}{2}}e^{\frac{-3R^2}{2Nb^2}}. \quad (1.15)$$

Form the theory of random walks, this probability distribution is a well known result. It is a Gaussian probability distribution for a random walk in three dimension [2]. The factor 3 in the

exponents and a multiplying factor is for the consideration of a three-dimensional case. Hence for any d-dimensional ideal polymer chain the probability distribution can be generalised as;

$$P_N(R) = D\left(\frac{d}{2\pi N b^2}\right)^{\frac{d}{2}} e^{\frac{-dR^2}{2Nb^2}}, \tag{1.16}$$

where D represents the $d-1$ dimensional surface are of a sphere.

Exact Solution

The exact solution was obtained in various methods by different peoples; of these we will consider that of Yamakawa [3]. Our equation for the probability distribution of a polymer chain (Equation 1.13) may be written in the form,

$$P_N(R) = \frac{1}{2\pi^2 R}\left(\frac{-i}{2b^2}\right)\int_{-\infty}^{\infty} \xi e^{\frac{i\xi R}{b}}\left[\frac{\sin(\xi)}{\xi}\right]^N d\xi \qquad \text{where} \quad \xi = kb. \tag{1.17}$$

Using a power series expansion for $\sin^N \xi$ we can write

$$\sin^N \xi = \frac{1}{(2i)^N}\sum_{p=0}^{N}(-1)^p \frac{N!}{p!(N-p)!} e^{Ni\xi - 2pi\xi}. \tag{1.18}$$

With this substitution, Equation (1.17) can be written as,

$$P_N(R) = \frac{1}{2^{N+2} i^{N-1}\pi^2 b^2 R}\sum_{p=0}^{N}(-1)^p \frac{N!}{p!(N-P)!}\int_{-\infty}^{\infty} \frac{e^{i(N-2p+\frac{R}{b})\xi}}{\xi^{N-1}} d\xi. \tag{1.19}$$

To solve this integral, we use the concept of the contour integral. Let us consider the integral

$$I = \int_{-\infty}^{\infty} \frac{e^{iax}}{(x-i\varepsilon)^{N-1}} dx \quad \text{where} \quad \varepsilon > 0$$

When $a > 0$, the contour of the integration consists of a large semicircle of radius r in the upper complex plane and the real axis between $-r$ and r. The same is true for the lower half of the plane if $a \leq 0$.

For $a > 0$, the Cauchy integral theorem guarantees that

$$\int_{-\infty}^{\infty} \frac{e^{ia\xi}}{(\xi)^{N-1}} d\xi = \lim_{(r,\varepsilon)\to(\infty,0)} I = 0 \quad (a > 0)$$

On the other hand, when $a \leq 0$ application of Goursat's theorem leads to

$$\int_{-\infty}^{\infty} \frac{e^{ia\xi}}{(\xi)^{N-1}} d\xi = -\lim_{(r,\varepsilon)\to(\infty,0)} I = -\lim_{\varepsilon\to 0} \frac{2\pi i}{(N-2)!} \left[\frac{d^{N-2}e^{ia\xi}}{d\xi^{N-2}}\right]_{\xi=-i\varepsilon} = \frac{2\pi i^{N-1}}{(N-2)!} a^{N-2} \quad (a \leq 0).$$

Therefore, for $a = N - 2p + \frac{R}{b} \leq 0$ and putting $N - p = k$ the exact probability distribution will be given by

$$P_N(R) = \frac{1}{2^{N+2}\pi b^2 R(N-2)!} \sum_{p=0}^{k\leq(N-\frac{R}{b})/2} (-1)^k \frac{N!}{k!(N-k)!} \left(N - 2k - \frac{R}{b}\right)^{N-2}. \tag{1.20}$$

We derived this expression just for generality, and we did not use this to compare with the numerical values.

1.2.3 Probability Distribution of End-to-End Distance for Real Polymer Chain

In the model of the previous section, we assumed that the orientation of each bond is random and completely independent of the orientation of the previous bond. This means that the polymer is able to fold back on to itself at certain location, which is physically impossible since any two physical bodies can not occupy the same position at the same time. In other words any two monomers in space interact when they come closer to each other. This interaction between segments is called excluded volume interaction.

In general there are two types of interaction in a given polymer chain[2]. Short-range and long-range interaction. The short-range interaction is the interaction between two segments that extend only up to a finite distance along the chain. This interaction is found to decay exponentially for large N and there is no change in fundamental result in $\langle R^2 \rangle$. Therefore, for longer chains we can neglect the short-range interaction[2]. The second type of interaction is the so called *excluded volume effect*. In brief this interaction denies the possibility of two segments sharing the same physical location.

Polymers are generally found in three-dimensional space, but for generality let us consider a polymer chain that is distributed in d-dimensional space. To find the probability distribution in the presence of this excluded volume effect, let us reconsider our polymer chain. Moreover, for simplicity to find the expression for the probability distribution let us use the asymptotic solution as a starting point. Consider the quantity $X(R)dR$, which is the total number of excluded volume chains, with the N-th step lying at a distance between R and $R + dR$ from the origin. Since all possible paths have the same weight, $X(R)$ is the same as the distribution function $P(R)$ [2]. That means, in the absence of this volume effect $X(R)$ is the same as the probability distribution we found in Equation (1.16):

$$X_0(R) = P_N(R) = D \left(\frac{d}{2\pi N b^2}\right)^{\frac{d}{2}} e^{\frac{-dR^2}{2Nb^2}}. \tag{1.21}$$

However, in the presence of the excluded volume effect where there is no overlapping, resulting in an increase in chain length, this distribution will change. To approximate an expression for the probability distribution of such a case let us assume that the total volume of the space in which the polymer is distributed is R^d and the volume of each segment (single bead-rod) is γ. If $X(R)$ is the probability that no overlapping has occurred when we place N polymer chain in a total volume of R^d, the total number of excluded volume chains between R and $R + dR$ will be

$$P(R) = X(R)X_0(R).$$

For a given R^d and γ the probability that any two segments overlap is just the ratio $\frac{\gamma}{R^d}$. Therefore, the probability of not over lapping is simply given by $1 - \frac{\gamma}{R^d}$. For a given N segments there are $\frac{N(N-1)}{2}$ possible combinations, where the factor $\frac{1}{2}$ is to avoid double counting. Hence the probability that there is no overlap occurs in any of these combinations is given by

$$X(R) = \left(1 - \frac{\gamma}{R^d}\right)^{\frac{N(N-1)}{2}}.$$

This can be rewritten as

$$X(R) = e^{\left[\frac{N(N-1)}{2}\ln(1-\frac{\gamma}{R^d})\right]}.$$

For $N \gg 1, and \;\; R^d \gg \gamma$; therefore, $(N-1) = N$ and $\ln(1 - \frac{\gamma}{R^d}) \approx -\frac{\gamma}{R^d}$, then $X(R)$ becomes

$$X(R) = e^{\frac{-N^2\gamma}{2R^d}}. \tag{1.22}$$

Therefore

$$P(R) = X(R)X_0(R) = D\left(\frac{d}{2\pi Nb^2}\right)^{\frac{d}{2}} e^{\left(\frac{-dR^2}{2Nb^2} - \frac{N^2\gamma}{2R^d}\right)}. \tag{1.23}$$

This is an expression of the probability distribution of a polymer chain in d-dimensions in the presence of the excluded volume effect.

Since our aim to to find the scaling, the probability distribution $P(R)$ can be taken as the number of accessible states $\Omega(R)$ [4], [2]. Therefore, entropy $S_N(R)$ of the system by definition can be given as

$$S_N(R) = -K\ln(P(R)) = S_0 + \frac{dKR^2}{2Nb^2} + \frac{KN^2\gamma}{2R^d}, \tag{1.24}$$

where S_0 is the initial entropy of the system.

From the expression for the entropy we can find the Mean Free Energy $F(R)$ using the relation

$$F(R) = \overline{E} - TS(R). \tag{1.25}$$

But the mean internal energy $\overline{E}$ of the given polymer can be taken as constant [4]. Therefore, the mean free energy becomes

$$F_N(R) = -TK\ln(P(R)) = F_0 + \frac{dKR^2}{2Nb^2} + \frac{KN^2\gamma}{2R^d}. \tag{1.26}$$

As a matter of fact any system exits at its minimum free energy. To minimise this energy we have to differentiate $F_N(R)$ with respect to R and equate to zero. This is shown below.

$$\frac{dKTR}{Nb^2} - \frac{dKTN^2\gamma}{2R^{d+1}} = 0.$$

Therefore,

$$R \propto N^{\nu} \qquad \text{where} \quad \nu = \frac{3}{d+2} \tag{1.27}$$

Using this result for R we can find the mean square end-to-end distance by squaring both sided as

$$\langle R^2\rangle \propto N^{2\nu} \qquad \text{where} \quad \nu = \frac{3}{d+2} \tag{1.28}$$

In three-dimension

$$\langle R^2\rangle \sim N^{2\times\frac{3}{5}} \tag{1.29}$$

As found in the section 1.2.1, for one dimensional polymer where $\nu = \frac{3}{1+2} = 1, \langle R^2\rangle \propto N$ and in two dimension, $\nu = \frac{3}{2+2} = \frac{3}{4}$, $\langle R^2\rangle \propto N^{\frac{3}{2}}$. This shows that equation 1.28 is a remarkable result[4]. Moreover, this equation shows that for higher dimension ν found to decrease showing that the end-to-end distance becomes shorter. For instance, for $d = 4$ we see that $\nu = \frac{1}{2}$ and $\langle R^2\rangle \propto N$. In fact this is a reality. As the dimension increase there is a possibility for the polymer to fold in different direction that causes the end-to-end distance becomes shorter. This can be generalized as in the table 1.1.

d	ν	$\langle R^2\rangle$
1	1	N^2
2	$\frac{3}{4}$	$N^{\frac{3}{2}}$
3	$\frac{3}{5}$	$N^{\frac{6}{5}}$
4	$\frac{1}{2}$	N
5	$\frac{3}{8}$	$N^{\frac{3}{4}}$

Table 1.1: Proportionality of $\langle R^2\rangle$ to the number of monomers for different dimensions

In the case of three dimension, we see that the characteristic size of excluded volume chains is proportional to $N^{\frac{3}{5}}$, and not $N^{\frac{1}{2}}$. The above is a very rough theory of excluded volume effect. The statistical properties of excluded volume chains have been extensively investigated in numerical simulation and for large N it has been found that the size obeys the following formula:

$$\langle R^2 \rangle \sim N^{2\nu}, \tag{1.30}$$

where the exponent ν is approximately 0.588, very close to $\frac{3}{5}$ [2]. This critical exponent is to be computed numerically.

1.2.4 Radius of Gyration R_g

The mean-square end-to-end distance of a linear polymer is not an experimentally observed quantity [9]. Of more experimental relevancy is the so-called radius of *Gyration* about the centre of mass. It is an alternative measure of the size of a polymer chain which can be measured by light scattering experiments [2]. The radius of gyration is defined as the radial distance from a given axis at which the mass of a body could be concentrated without altering the rotational inertia of the body about that axis. For a polymer chain made of N monomers, the square of radius of gyration is defined as:

$$\langle \vec{R}_g^2 \rangle = \frac{1}{N} \sum_{j=1}^{N} \langle (\vec{R}_i - \vec{R}_{cm})^2 \rangle, \tag{1.31}$$

where $\vec{R}_{cm}$ is the position of centre of mass, given by

$$\vec{R}_{cm} = \frac{1}{N} \sum_{j=1}^{N} \vec{R}_i.$$

If we expand the term in the parentheses of equation 1.31

$$\langle \vec{R}_g^2 \rangle = \frac{1}{N} \sum_{j=1}^{N} \langle (\vec{R}_i^2 - 2\vec{R}_i \cdot \vec{R}_{cm} + \vec{R}_{cm}^2) \rangle = \frac{1}{N} \left\{ \sum_{j=1}^{N} \langle \vec{R}_i^2 \rangle - 2 \sum_{j=1}^{N} \langle \vec{R}_i \cdot \vec{R}_{cm} \rangle + N \langle \vec{R}_{cm}^2 \rangle \right\}.$$

Subsituting for $\vec{R}_{cm}$

$$\langle \vec{R}_g^2 \rangle = \frac{1}{N} \left\{ \sum_{j=1}^{N} \langle \vec{R}_i^2 \rangle - \frac{2}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} \langle \vec{R}_i \cdot \vec{R}_j \rangle + \frac{N}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \langle \vec{R}_i \cdot \vec{R}_j \rangle \right\}.$$

After simplification we can find

$$\langle \vec{R}_g^2 \rangle = \frac{1}{N} \left\{ \sum_{j=1}^{N} \langle \vec{R}_i^2 \rangle - \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} \langle \vec{R}_i \cdot \vec{R}_j \rangle \right\}. \tag{1.32}$$

Using a substitution

$$|\vec{R}_i - \vec{R}_j|^2 = |\vec{R}_i|^2 + |\vec{R}_j|^2 - 2|\vec{R}_i \cdot \vec{R}_j| \quad \Rightarrow \quad |\vec{R}_i \cdot \vec{R}_j| = \frac{1}{2} \left\{ |\vec{R}_i|^2 + |\vec{R}_j|^2 - |\vec{R}_i - \vec{R}_j|^2 \right\}$$

equation 1.32 becomes

$$\langle \vec{R}_g^2 \rangle = \frac{1}{N} \left\{ \sum_{j=1}^{N} \langle \vec{R}_i^2 \rangle - \frac{1}{2N} \sum_{i=1}^{N} \sum_{j=1}^{N} \langle \left\{ \vec{R}_i^2 + \vec{R}_j^2 - |\vec{R}_i - \vec{R}_j|^2 \right\} \rangle \right\}. \tag{1.33}$$

This can be expanded as

$$\langle \vec{R}_g^2 \rangle = \frac{1}{N} \left\{ \sum_{j=1}^{N} \langle \vec{R}_i^2 \rangle - \frac{1}{2N} \sum_{i=1}^{N} \sum_{j=1}^{N} \langle \vec{R}_i^2 \rangle - \frac{1}{2N} \sum_{i=1}^{N} \sum_{j=1}^{N} \langle \vec{R}_j^2 \rangle + \frac{1}{2N} \sum_{i=1}^{N} \sum_{j=1}^{N} \langle |\vec{R}_i - \vec{R}_j|^2 \rangle \right\}.$$

But we know that

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \langle \vec{R}_i^2 \rangle = \sum_{i=1}^{N} \sum_{j=1}^{N} \langle \vec{R}_j^2 \rangle = N \sum_{i=1}^{N} \langle \vec{R}_i^2 \rangle$$

Therefore,

$$\langle \vec{R}_g^2 \rangle = \frac{1}{2N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \langle |\vec{R}_i - \vec{R}_j|^2 \rangle. \tag{1.34}$$

Since the summation over j is regarded as vacant if $j = 0$, we can write the above summation as

$$\langle \vec{R}_g^2 \rangle = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{i-1} \langle |\vec{R}_i - \vec{R}_j|^2 \rangle$$

Here $\vec{R}_i - \vec{R}_j$ is the relative position of the segment at position i and j. Therefore, with the separation distance between each successive segment as b we can write

$$\langle |\vec{R}_i - \vec{R}_j|^2 \rangle = (i - j) b^2 \qquad (i > j)$$

Hence

$$\langle \vec{R}_g^2 \rangle = \frac{b^2}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{i-1} (i - j) \tag{1.35}$$

To find the relation between the polymer end-to-end distance and the radius of gyration let us use the following facts

$$\sum_{j=1}^{i-1} (i - j) = i(i-1) - \sum_{j=1}^{i-1} j \quad \text{and} \quad \sum_{j=1}^{N} j = \frac{1}{2} N(N+1)$$

This shows that

$$\sum_{j=1}^{i-1} (i - j) = i(i-1) - \frac{1}{2}(i-1)(i-1+1) = \frac{1}{2} i(i-1)$$

With this substitution equation 1.35 becomes

$$\langle \vec{R}_g^2 \rangle = \frac{b^2}{2N^2} \sum_{i=1}^{N} i(i-1) = \frac{b^2}{2N^2} \left\{ \sum_{i=1}^{N} i^2 - \sum_{i=1}^{N} i \right\}.$$

Using the same trick

$$\sum_{i=1}^{N} i^2 = \frac{1}{6} N(N+1)(2N+1).$$

The final expression for radius of gyration becomes

$$\langle \vec{R}_g^2 \rangle = \frac{b^2}{2N^2} \left\{ \frac{1}{6} N(N+1)(2N+1) - \frac{1}{2} N(N+1) \right\} = \frac{Nb^2}{6} (1 - \frac{1}{N^2}). \tag{1.36}$$

From equation 1.3 we know that $\vec{R}^2 = Nb^2$. Therefore;

$$\langle \vec{R}_g^2 \rangle = \frac{\langle \vec{R}^2 \rangle}{6} \left(1 - \frac{1}{N^2} \right). \tag{1.37}$$

For large N

$$\langle \vec{R}_g^2 \rangle \approx \frac{\langle \vec{R}^2 \rangle}{6}. \tag{1.38}$$

Now let us see the numerical simulation aspect of the essay which we have used to find the exponent ν and the relation between $\langle \vec{R}_g^2 \rangle$ and $\langle \vec{R}^2 \rangle$.

2. Molecular Dynamics

2.1 Molecular Dynamics and its Aim

Biological systems exhibit a complexity and diversity far richer than the simple solid or liquid systems traditionally studied in physics or chemistry. Indeed, the powerful quantitative methods offered by these two disciplines and which are used to analyse the behaviour of prototypical simple systems are often difficult to extend to the domain of biological systems [10]. Advances in computer technology and breakthroughs in simulation techniques have been steadily reducing the gap between quantitative models and actual biological behaviour. We carry out computer simulations in the hope of understanding the properties of assemblies of molecules in terms of their structure and the microscopic interactions between them. This serves as a complement to conventional experimentations, enabling us to make new discoveries which might otherwise not have been possible using conventional methods [11].

The two main families of simulation techniques are molecular dynamics (MD) and Monte Carlo (MC); additionally, there is a whole range of hybrid techniques which combine features from both of these [12]. Monte Carlo methods, which are usually based on random walks and thus sequences of events, contain an apparent dynamics. It is, however, crucial to recognise that the sequence of events does not correspond to a sequence in real time, as opposed to the role of time as an explicit variable in molecular dynamics methods. The sequence of events is generally independent of the physical phenomena which we wish to describe by a Monte Carlo method and depends on the chosen algorithm. The creation of smart sequences is in fact, an important design goal in the construction of efficient Monte Carlo algorithms. The absence of time in Monte Carlo algorithms implies the fundamental inability to describe non-equilibrium processes.

Molecular dynamics is a standard computational technique used in condensed matter physics, materials science, chemistry, and other fields, consisting of following the temporal evolution of a system of N particles, interacting with each other by means of certain view. In classical molecular dynamics, the evolution is based on the Newton's law, and the forces are obtained as gradients of a certain potential which is function of all the particle coordinates [13].

In this simulation technique the time evolution of a set of interacting atoms is followed by integrating their equations of motion. In its simplest form, it is the numerical integration of Newton's equations of motion for a system of N interacting particles [12]. Mathematically;

$$m_i \frac{d^2 \vec{r}_i}{dt^2} = \vec{F}_i, \qquad i = 0, 1, 2, \ldots, N, \tag{2.1}$$

where m_i is the mass of the i-th particle.

And the force F_i acting on the i-th particle is usually described by some potential V as :

$$\vec{F}_i = -\vec{\nabla}_i V(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N), \qquad i = 0, 1, 2, \ldots, N \tag{2.2}$$

where $\vec{\nabla}_i = \frac{\partial}{\partial \vec{r}_i}$.

Therefore, the essential ingredient containing the physics is constituted by the potentials. Thus to have a physically acceptable result of the simulation we need to know the nature of the potentials that keeps the atoms or molecules in a given material together.

As discused in Chapter 1, in order to model the polymers, we utilise a bead-rod model approximation which replaces the real physical polymers with beads. Each bead represents one or more chemical units, and the rods represent the bonds between neighbouring units along a chain. As a matter of fact each bead interacts.

In physics, an interaction specifically refers to the action of one physical object upon another resulting in a potential energy; the physical objects under consideration may range from point particles to quantum fields. The interaction energy usually depends on the relative position of the objects. In the bead-rod model of a polymer chain there are two types of interactions: the interaction between the bonding beads, and that between non-bonding beads. The interactions between non-bonded beads are modeled by the Lennard-Jones potential, whereas the interactions between the bonded beads along the chain are modeled via the Finitely Extensible Non-Linear Elastic (FENE) potential [10]. These two types of model potentials are given below.

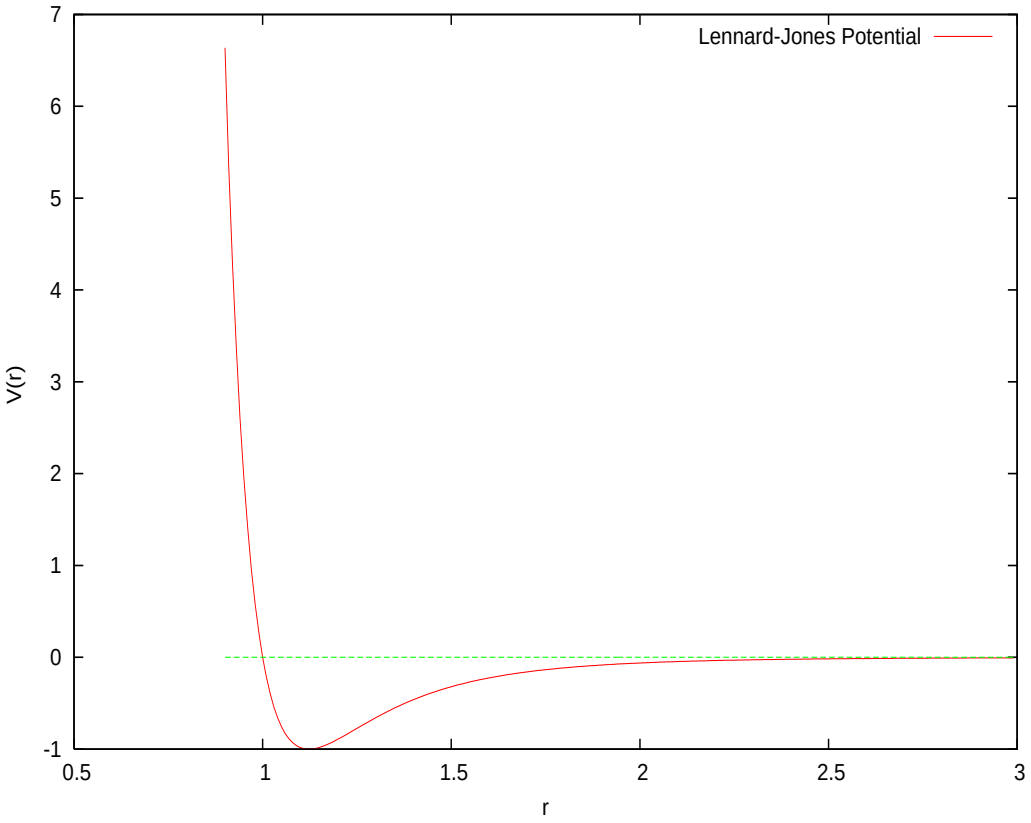

Figure 2.1: 6-12 Lennard-Jones potential: with $\sigma = \epsilon = 1$

The 6-12 Lennard-Jones potential for two particles at position i and j separated by a distance $\vec{r}_{ij}$ is given by :

$$U(\vec{r}_{ij}) = \begin{cases} 4\epsilon \left(\left(\frac{\sigma}{\vec{r}_{ij}} \right)^{12} - \left(\frac{\sigma}{\vec{r}_{ij}} \right)^{6} \right) & \text{if } \vec{r}_{ij} \leq r_c \\ 0 & \text{if } \vec{r}_{ij} > r_c \end{cases}$$

where ϵ and σ are constants that depend on the nature of the interaction and r_c the maximum interaction distance. This potential has an attractive tail at large $\vec{r}_{ij}$, it reaches a minimum around 1.122σ. It is strongly repulsive at a shorter distance, passing through 0 at $\vec{r}_{ij} = \sigma$ and increasing steeply as $\vec{r}_{ij}$ is decreased further as shown in Fig 2.1.

The term $\frac{1}{\vec{r}_{ij}^{12}}$ dominates at short distances, modelling the repulsion between atoms when they are brought very close to each other. Its physical origin is related to the Pauli principle: when the electron clouds surrounding the atoms start to overlap, the energy of the system increases abruptly. On the other hand $\frac{1}{\vec{r}_{ij}^{6}}$ dominates at larger distances, and is the origin of the attractive portion of the graph. It is the result of the van der Waals force, which originates from dipole-dipole interaction.

The second type of model potential, FENE potential is given by

$$U(\vec{r}_{ij}) = \begin{cases} \frac{-1}{2} k r_0^2 \ln \left(1 - \left[\frac{\vec{r}_{ij}}{r_0} \right]^2 \right) & \text{if } \vec{r}_{ij} < r_0 \\ 0 & \text{if } \vec{r}_{ij} \geq r_0 \end{cases}$$

where r_0 is an upper bound on the bond distance and, k is essentially a spring constant [10].

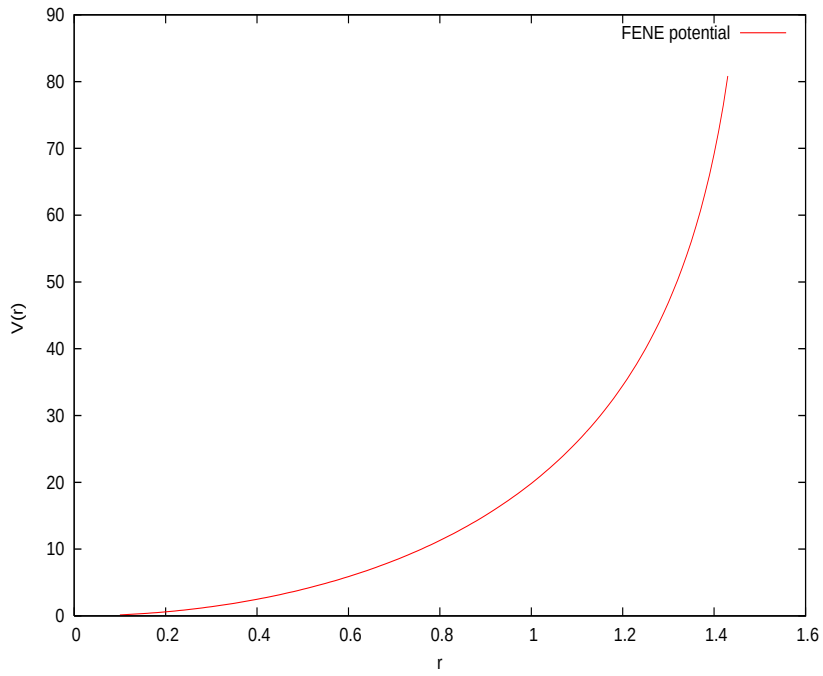

Figure 2.2: FENE potential: with $k = 30\frac{\epsilon}{\sigma^2}$, $r_0 = 1.5\sigma$, and $\sigma = \epsilon = 1$

This potential is as a result of the bond between the carbon-carbon bond in the backbone of the polymer. It can be modeled as of that of Hook's law but has non-linear form. The graph of this potential is shown as in Fig 2.2.

As this figure shows it is an attractive potential modelling interaction between the carbon bonds for a distance $\vec{r}_{ij}$ less than the cut-off distance r_0 and restricts the bond length to be less r_0.

When these potentials sum up, they model the maximum and minimum separation between given two monomers in a polymer chain. The graph of this combined potential is shown in Fig 2.3 .

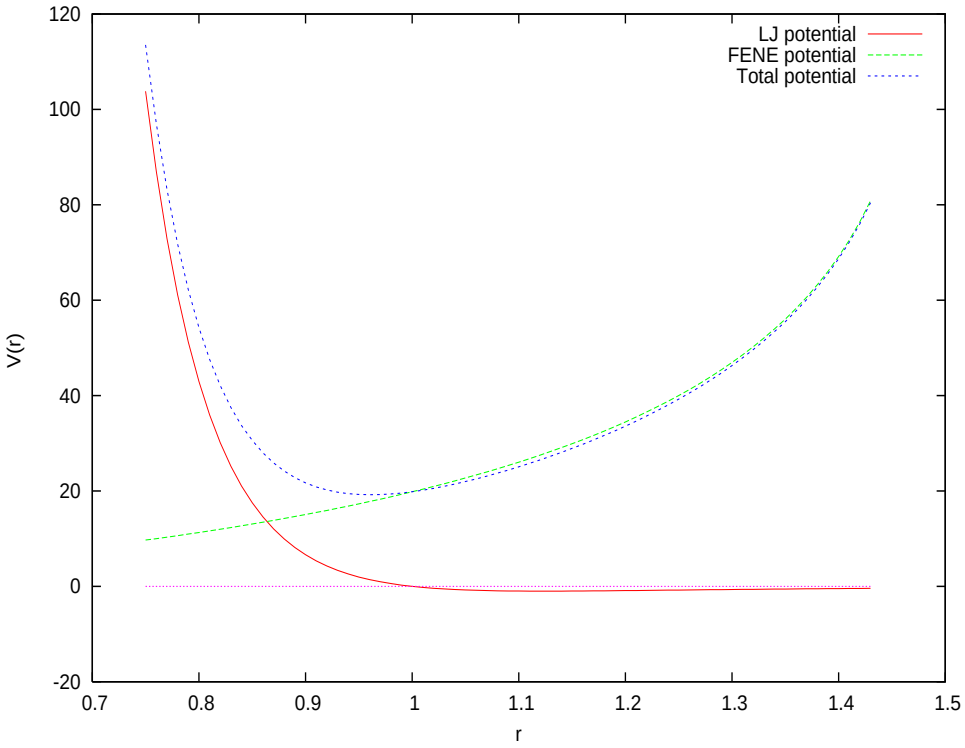

Figure 2.3: FENE and Lennard-Jones Potential

As can be seen from Fig 2.3 , these two potentials restricts the bond length between bonding monomers to be $0.75 \times \sigma \leq \vec{r}_{ij} < 1.45 \times \sigma$. The Lennard-Jones potential restricts the lower limit and that of FENE restricts the upper limit. Here the lower limit is controlled during configuration of the system. It is to mean that the Lennard-Jones potential for two particles separated at a distance less than $0.75 \times \sigma$ is infinite so that we do not consider it.

Once the potential energy is known, it is easy to find the force on each particle. And from the knowledge of the forces on each particle, it is straight forward to find the acceleration of each particle in the system. Integration of the equations of motion then yields a trajectory. We can use the trajectory to describe the positions, velocities of the particles as they vary over time from which the average values of particle properties can be determined. The method is deterministic; once the positions and velocities of each atom are known, the state of the system can be predicted at any time in the future or in the past.

To find the trajectory of the system economically, we have to develop a systematic time integration algorithm. Molecular dynamics simulations can be time consuming and computationally expensive [13]. However, computers are becoming faster and cheaper. Simulations of dissolved proteins are calculated up to the nanosecond time scale; however, simulations into the millisecond regime have been reported [11].

2.2 Time Integration Algorithm

The engine of a molecular dynamics program is its time integration algorithm, which is required to integrate the equation of motion of the interacting particles and follow their trajectory.

Time integration algorithms are based on the finite difference method, where time is discretised on a finite element Δt, over consecutive time steps. Knowing the position and some of their time derivatives at time t, the integration scheme gives the same quantity at a later time t + Δt. By iterating the procedure, the time evolution of the system can be followed over long intervals.

Of course, these schemes are approximate and there are errors associated with them. The two most common errors that might arise during discretisation of he time step are truncation and round-off errors. Truncation errors are related to the accuracy of the finite difference method with respect to the true solution. Finite difference method are usually based on a Taylor expansion truncated at some term. These errors do not depend on the impelmentation: they are intrinsic to the algorithm. The round-off error is simply related to the finite number of digits used in the computer arithmetics. Both errors can be reduced by decreasing Δt, and by considering higher orders of the expansions of the physical quantity under consideration. Most of the numerical algorithms used in molecular dynamics simulations use Taylor series expansions of different order.

A Large quantity of numerical algorithms have been developed for integrating these equations of motion. These includes: Verlet algorithm, Leap-frog algorithm, Velocity Verlet and Beemans algorithm. The two most popular integration methods for MD calculations are the Verlet algorithm and Predictor-corrector algorithm [12]. In choosing which algorithm to use, one should consider the following criteria: The algorithm should conserve energy and momentum, it should be computationally efficient, and it should permit a long time step for integration. Since we have used the Verlet and Velocity Verlet algorithms, let us examine how these algorithms are developed.

2.3 The Verlet Algorithm

Since we wish to compute the motion of many particles in molecular dynamics over a very large number of time steps, the well known Euler-type method, with its large numerical errors can not produce a good result [14]. It is, therefore, necessary to use a slightly more complicated scheme for solving the differential equation arising from Newton's second law. One of these better methods is the Verlet algorithm. It is the most commonly used time integration algorithm in MD simulation [12]. The basic idea is to write the position $\vec{r}(t)$ both forward and backward

in time using the Taylor expansion method:

$$\vec{r}(t+\Delta t) = \vec{r}(t) + \Delta t\frac{d\vec{r}(t)}{dt} + \frac{(\Delta t)^2}{2!}\frac{d^2\vec{r}(t)}{dt^2} + \frac{(\Delta t)^3}{3!}\frac{d^3\vec{r}(t)}{dt^3} + O((\Delta t)^4) \tag{2.3}$$

$$\vec{r}(t-\Delta t) = \vec{r}(t) - \Delta t\frac{d\vec{r}(t)}{dt} + \frac{(\Delta t)^2}{2!}\frac{d^2\vec{r}(t)}{dt^2} - \frac{(\Delta t)^3}{3!}\frac{d^3\vec{r}(t)}{dt^3} + O((\Delta t)^4) \tag{2.4}$$

If we sum the terms, we find the position of the particle

$$\vec{r}(t+\Delta t) = 2\vec{r}(t) - \vec{r}(t-\Delta t) + \vec{a}(t)\frac{(\Delta t)^2}{2!} + O((\Delta t)^4) \tag{2.5}$$

where $\frac{d^2\vec{r}}{dt^2} = \vec{a}(t)$ is the acceleration. By definition, velocity is simply given by

$$\vec{v}(t) = \frac{1}{2\Delta t}\left(\vec{r}(t+\Delta t) - \vec{r}(t+\Delta t)\right) + O((\Delta t)^2) \tag{2.6}$$

What do these equations mean? We first assume we are at some point in the middle of a simulation, say at a time step numbered n. At the start of this time step, we know the positions of the atoms $\vec{r}(t_0)$, and their corresponding positions at the previous time step $\vec{r}(t_0 - \Delta t)$. The first thing we must do is calculate the forces $\vec{F}(t_0)$ acting on the atoms. We can then input this information into the first equation, together with a suitable choice of time interval (Δt), and calculate the the atom positions $\vec{r}(t_0 + \Delta t)$ at the $(n+1)th$ time step. This equation is accurate to an order given by the fourth power of the time interval. once we know the positions $\vec{r}(t+\Delta t)$, we can calculate the velocity ($\vec{v}_n(t)$) of each atom at time step n using equation 2.6, which is accurate to the second power of the time interval. When this is done we are ready to continue with the next time step. Providing the time interval (Δt) is small enough, this procedure is good enough to reveal all the interesting properties of the system!

Despite its simplicity, the drawback of this method is its dependence on two initial conditions, $\vec{r}(t_0)$ and $\vec{r}(t_0 - \Delta t)$, and the lack of a way to calculate the velocity with an error less than $(\Delta t)^2$. Moreover, the atom positions are given for the $(n+1)$th time step, while the velocities are given for the previous time step.

An even better implementation of the same basic algorithm is the so-called Velocity Verlet algorithm, where position, velocity and acceleration at a time $t + \Delta t$ are obtained from the same quantity and at the same time t in the following way

$$\vec{r}(t+\triangle t) = \vec{r}(t) + \Delta t\vec{v}(t) + \frac{(\Delta t)^2}{2!}\vec{a}(t) + O((\Delta t)^3)$$

$$\vec{v}(t+\frac{\Delta t}{2}) = \vec{v}(t) + \frac{\Delta t}{2!}\vec{a}(t)$$

$$\vec{a}(t+\Delta t) = -\frac{1}{m}\vec{\nabla}V(\vec{r}(t+\Delta t))$$

$$\vec{v}(t+\Delta t) = \vec{v}(t+\frac{\Delta t}{2}) + \frac{\Delta t}{2!}\vec{a}(t+\Delta t)$$

In simple form this can be written as

$$\vec{r}(t+\Delta t) = \vec{r}(t) + \vec{v}(t)\Delta t + \frac{1}{2m}\vec{F}(t)(\Delta t)^2 + O((\Delta t)^3)$$

$$\vec{v}(t+\Delta t) = \vec{v}(t) + +\frac{\Delta t}{2m}\left(\vec{F}(t+\Delta t) + \vec{F}(t)\right) + O((\Delta t)^2)$$

From this set of equation we can easily see that once the potential is known, and the initial configuration and velocities given at time t_0, we can find the position and momentum of the system of particles after some time $t+\Delta t$. The force at times t and $t+\Delta t$ must also be known. This can be further found using the position and potential at times t and $t+\Delta t$. This is discused in section 3.2. In this manner we can find the position and velocity of the system step by step by iterating through the algorithm. The problem is for how long and how large we have to iterate the simulation?

2.4 How Long? How Large?

Molecular dynamics evolves a finite-sized molecular configuration forward in time, in a step-by-step fashion. There are limits on the typical time and length scales that can be investigated, and the consequences involved must be considered in analysing the results. The best method of fixing the time step is to plot the total energy as a function of time, and to check that the total energy is in fact conserved. If the graph of the total energy versus time is more or less constant for a specific time step, we can conclude that the time scale is appropriate. Otherwise we have to reduce the time step to fulfil the energy conservation condition and the stability of the system.

Simulation runs are typically short: typically $t \sim 10^3$ to 10^6 MD steps[12], corresponding to perhaps a few nanoseconds of real time, and in special cases extending to the microsecond regime. This means that we need to test whether or not a simulation has reached equilibrium before we can trust the averages it calculates. Moreover, there is a clear need to subject the simulation averages to a statistical analysis, to make a realistic estimate of the errors. How long should we run this for? This depends on the system and the physical properties of interest.

In the case of polymer end-to-end distance and radius of gyration, we do not need to know whether the system comes to equilibrium or not. There is no equilibrium as long as the polymer is imposed to move in space. If our objective is only to measure these quantities, we simply let the polymer move for a desired number of times and we can then measure the mean values of the quantities. Of course, there is still remains the a problem of how to choose the number of times we run the simulation.

To produce a good result it is obviously necessary to run the simulation a greater number of times say 10^3 to 10^6. The number of iterations that produces our data, here in this essay, are discused and justified in chapter three and four.

3. Simulation Procedure

3.1 Initial Configuration

The output of any simulation basically depends on the input data given to the system. Therefore, due consideration has to be given to the way the start-up of the simulation is arranged.

In molecular dynamics there are two ways of starting a simulation. The first is to start from scratch by giving a set of initial positions and velocities. In this simulation technique, the positions of the particles are usually defined on a lattice, assuming a certain crystal structure, and the velocities may be taken to be either zero or they can be assigned randomly. Some randomisation must then be introduced in the starting sample. If this were not the case, then all atoms would be symmetrically equivalent, and the equation of motion cannot do anything except evolve them all in precisely the same way. In the case of a perfect lattice, there would be no net force on the atoms from symmetry reasons, and therefore atoms would sit idle indefinitely. Initial randomisation is usually the only place where chance enters a molecular dynamics simulation. The subsequent time evolution is completely deterministic, expect that there are errors due to the rounding off and potential truncation.

Another possibility is to take the initial position and velocity to be the final position and velocity of a previous MD run.

In this essay, the simulation was started from scratch. The first monomer is placed at the centre of a cube whose dimension is proportional to the square root of the number of monomers. Then the second monomer positions itself on one of the six faces of the cube randomly. Once the second monomer takes its position, it acts as reference for the third monomer. In this manner the rest of the monomers are placed and there is no overlapping. For overlapping not to occur, when a given monomer take its position, the position occupied by this monomer is registered as a list. In this way the position of all the monomers is listed. This all procedures can be seen in the appendix.

In general, during the placement of the monomers, the effect of the excluded volume effect is taken into account; there is a minimum separation of the adjacent pair of monomers so that the position of the polymer as a whole is in the range of potential interaction discussed in 2.1.

After all the monomers have been placed, a small random displacement is added to the lattice position. The amplitude of these displacements is very small compared to the distance between each monomer. The maximum separation of adjacent monomers is attenuated by the potential energy used in the simulation. This will be discused in section 3.2.

As for the velocities, each monomer is randomly given an initial velocity in three dimensions. To avoid the resulting small total linear momentum that corresponds to the translational motion of the whole system, the velocity of centre of the mass is subtracted from the velocities of the monomers, component by component. This insures that the system is at a zero total momentum state. Moreover, each component of the velocities are set in the direction of the force (for the same component) so that there is no rotational motion. The techniques used to set these

expressions are shown in the code (available in the appendix).

3.2 Potential Truncation and Algorithm

In molecular dynamics, atoms interact with each other. These interactions originate forces which act upon atoms, and atoms move under the action of these instantaneous forces. As the atoms move, their relative positions change, and the force also varies.

Calculation of the atomic forces is usually the most computationally intensive operation. It is normally assumed that the forces between atoms are pair forces; that is, they act exclusively between pairs of atoms. (Higher order forces, involving three- or four body forces are also sometimes considered - especially in complex molecular structures). If there are N atoms in the system, there will be at most $\frac{N(N-1)}{2}$ unique atom pairs, each with an associated force to compute. The time it takes to perform a molecular dynamics simulation on a computer is thus (approximately) proportional to N^2 [15]. Usually however, a cut-off is applied at a certain interatomic separation, beyond which it is assumed the force is zero. This allows for more efficiency in computing the forces, since the computation of all atom pairs is no longer necessary. Therefore, to get a result that is both reliable and economic from the simulation we need to truncate the potential.

In general, an efficient molecular dynamics code must:

- deal efficiently with the locality of interactions, that is, avoid running over all pairs just to find out that the majority of them were too far from each other to be interacting.

- deal with the fact that the topology changes as times goes on, so that the lists of physical quantities which are under investigation have to be kept updated dynamically.

In our simulation, the truncation is accomplished by introducing neighbour lists (available in the appendix), and by performing some kind of filtering on the particle pairs. The model potential only acts between the particle pairs that are in the list.

In Chapter two we saw that forces are usually obtained as the gradient of a potential energy which depends on the position of the particle. The realism of the simulation therefore depends on the ability of the potential chosen to reproduce the behaviour of the material under the conditions in which the simulation is run. That is, it mimics the behaviour of the real system only to the extent that interatomic forces are similar to those that real atoms would experience when arranged in the same configuration.

As described in chapter one, a polymer chain consists of molecules connected by a covalent bond. The interaction between these monomers are modeled by FENE and the 6-12 Lennard-Jones potential, discused in chapter two. The 6-12 Lennard-Jones interaction acts between any two monomers separated by a distance less than $r_c = 2.5\sigma$.The choice of this potential cutoff reduces the time of the simulation. The FENE potential acts only between adjoining monomers,

and has the effect of limiting the bond length to r_0. The parameters in the FENE potential are taken to be $r_0 = 1.5\sigma$ and k =30$\frac{\epsilon}{\sigma^2}$. In the simulation the constants σ, ϵ and mass of the particle m are taken to be unity.

Once the potentials and their range of action are determined, it is easy to compute each component of the force. We know that for known potential $V(\vec{r})$, the force is given by:

$$\vec{F}(\vec{r}) = -\vec{\nabla}V(\vec{r}) \tag{3.1}$$

Therefore, force due to these potentials will be

$$|\vec{F}_{LJ}(\vec{r}_{ij})| = \begin{cases} 24\epsilon\left(\frac{2\sigma^{12}}{\vec{r}_{ij}^{12}} - \frac{\sigma^6}{\vec{r}_{ij}^{7}}\right) & \text{if } \vec{r}_{ij} \leq r_c \\ 0 & \text{if } \vec{r}_{ij} > r_c \end{cases}$$

$$|\vec{F}_{FENE}(\vec{r}_{ij})| = \begin{cases} -\vec{r}_{ij}kr_0^2\left(\frac{1}{r_0^2 - \vec{r}_{ij}^2}\right) & \text{if } \vec{r}_{ij} < r_0 \\ 0 & \text{if } \vec{r}_{ij} \geq r_0 \end{cases}$$

Each component of the force is found so that we are able to find each component of acceleration for each particle at ant time (this can be seen in the appendix).

To find the force, position and velocity of each particle after some time Δt the following Verlet and Verlet velocity algorithms are used.

$$\vec{r}(t + \Delta t) = \vec{r}(t) + \Delta t\vec{v}(t) + \frac{(\Delta t)^2}{2}\frac{\vec{F}(t)}{m} \tag{3.2}$$

Therefore, each component (x, y, z) of the position is easily computed. Then the relative separation of each particle is also computed. To fine the instantaneous velocity we have used

$$\vec{v}(t + \Delta t) = \vec{v}(t) + \Delta t\frac{\vec{F}(t) + \vec{F}(t + \Delta t)}{2m} \tag{3.3}$$

As can be seen from equation 3.3, to find the instantaneous velocities of each molecule we need to find $\vec{F}(t)$ and $\vec{F}(t + \Delta t)$, where each force is calculated from the concept of the potential energy. That is why finding the velocity and acceleration is said to be the most time consuming part of the simulation process.

To find the average values of the physical quantities this algorithm is iterated one hundred thousand number of times. We have also iterated the simulation one million times (for 40 monomers) and it has produced almost the same result. But it takes abut five days.

3.3 Conservation of Energy

During the simulation, to check the conservation of energy, the instantaneous potential and kinetic energies are calculated and saved as a list.

The average potential energy V is obtained by averaging the instantaneous values, which is obtained straightforward at the same time as the force computation is made. The knowledge of V is required to verify energy conservation. This is an important check to do in any MD simulation, as stated before.

The instantaneous kinetic energy is given by

$$KE = \frac{1}{2}\sum_{j=1}^{N} m_j[\vec{v}_j]^2 \tag{3.4}$$

and is therefore extremely easy to compute. Its average is checked at each iteration.

Along with this, total energy $E = KE + V$ is checked. To see what is happening to energies with time during the run, we check the energy flows back and forth between kinetic and potential, causing $KE(t)$ and $V(t)$ to fluctuate while their sum remains fixed, see Fig 4.1.

As can be seen from Figure 4.1, there is a very small fluctuation in the total energy. These fluctuations are usually caused by errors in the time integration and truncation of the Taylor series expansion and, and can be reduced by reducing the time step. The time steps taken for each figure is 0.005.

3.4 Polymer End-to-End Distance and Radius of Gyration

As stated in chapter one, the objective of this essay is to compute the critical exponent ν that shows the proportionality of the polymer end-to-end distance to the number of monomers and to find radius of gyration.

Using the above algorithm, at each instant of time, the position and velocity of each particle is computed and the mean square end-to-end distance can be readily found by taking the vector difference as:

$$\langle R^2 \rangle = \langle (\vec{x}_N - \vec{x}_0)^2 + (\vec{y}_N - \vec{y}_0)^2 + (\vec{z}_N - \vec{z}_0)^2 \rangle, \tag{3.5}$$

where 0, and N are the indices of the first and last coordinate of the monomer.

As discussed in chapter one, the radius of Gyration is given by

$$\langle R_g^2 \rangle = \frac{1}{N}\sum_{j=1}^{N} \langle (\vec{R}_i - \vec{R}_{cm})^2 \rangle. \tag{3.6}$$

How this things are computed using the code is as follows. The initial configuration is set as in 3.1. Once the initial position and the potentials for a given N monomers are known from the initial configuration, the positions and velocity after some time Δt is calculated using equation 3.2 and 3.3. At the same time $\langle R^2 \rangle$ and $\langle R_g^2 \rangle$ are computed using equation 3.5 and 3.6, respectively. In this manner using the Verlet algorithm discussed in 2.3 the code iterates for the desired number

of times. Then the number of monomers is increased step by step and the result of $\langle R^2 \rangle$ and $\langle R_g^2 \rangle$ is computed for all N. Finally the program plots the mean result as a function of N.

4. Result and discussion

4.1 The Energy Conservation

As described in Chapter two and three, the best way of checking the validity of the result of molecular dynamics simulations is to check the conservation of energy. In our simulation techniques this quantity is checked at every iteration step and the following graph is produced by the code attached on the appendix. Moreover, using the concept of equipartition theorem the temperature of the system is plotted on the same graph as the energies so that we can also plot the temperature of the system. From the equipartition principle, mean kinetic energy of a given N molecules in three dynamics is given by

$$KE = \frac{3}{2} N k_B T \tag{4.1}$$

where N is the number of molecules, and k_B is the Boltzmann constant which we consided as unity for simplicity. Hence

$$T k_B = \frac{2KE}{3N} \tag{4.2}$$

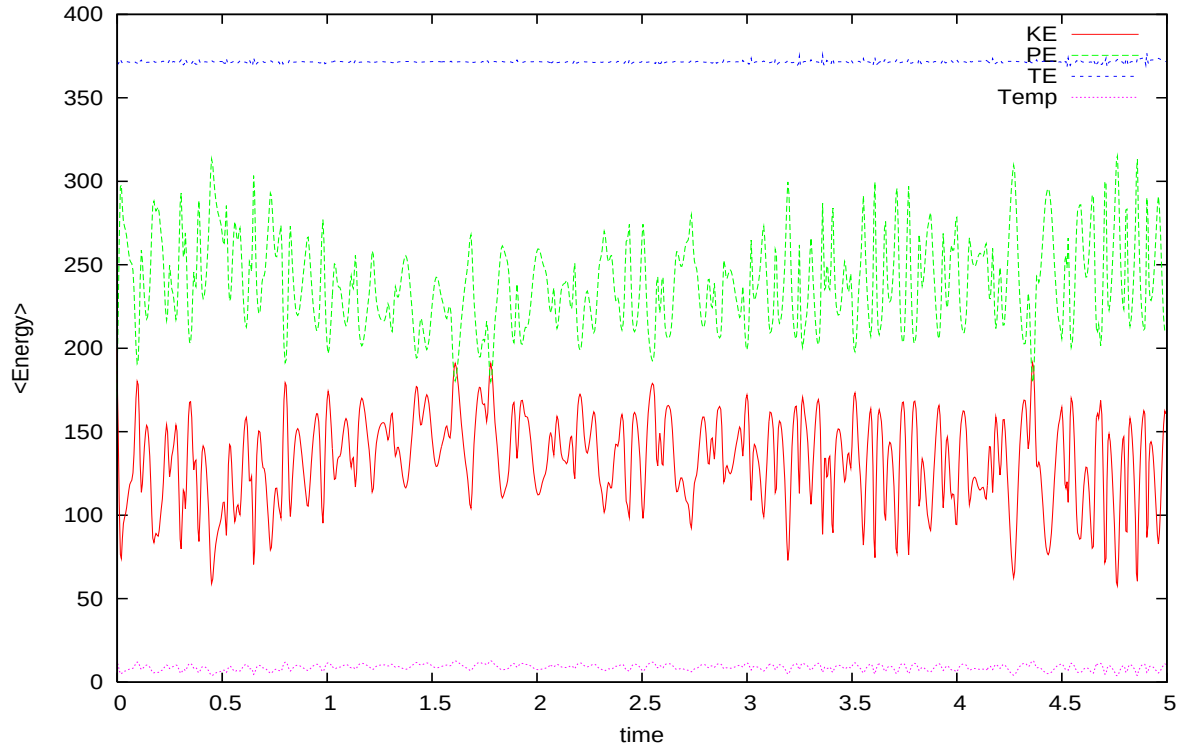

Figure 4.1: Energy switches back and forth between potential and kinetic energy; were KE is the kinetic energy, PE is potential energy, TE is total energy and Temp is temperature.

As can be seen from this Figure , the total energy is more or less constant. The small fluctuation in total energy is due to the errors in positions and velocities in their Taylor series expansion. The graph clearly shows that energy is transfered between potential and kinetic energies. This shows that the particles are interacting.

The temperature curve shows that, the temperature is almost constant through out the processes. This insures that the system is stable. During running it is observed that any increment in initial kinetic energy per particle greater that eight causes the instability of the system. This means that above this kinetic energy (temperature) since particles start to move at larger speed, the system becomes hotter and hotter that causes the to bond break and the system becomes unstable.

4.2 The End-to-End Distance and Radius of Gyration

As stated before, the main objective of this essay is to find the critical exponent ν that we have found in chapter one. As indicated in chapter one, in three-dimension (equation 1.29 and 1.30), polymers mean square end-to-end distance $\langle R^2 \rangle$ is proportional to $N^{2\nu}$. i.e.

$$\langle R^2 \rangle \sim N^{2\times\frac{3}{5}} \tag{4.3}$$

To simulate this exponent the the mean square end-to-end distance versus N for different values of N was computed and the result is plotted as in the Fig 4.2.

The procedure is as follows: first a polymer composed of 5 to 50 monomers, in step of 5, is subjected to an interaction 100,000 times their potential. During each iteration the end-to-end distance and radius of gyration is computed (seen on appendix). Then the mean value of mean square end-to-end distance and the radius of gyration are easily computed. Thus, the result of mean values are plotted as a function of the number of monomers.

We have confermend that the mean square end-to-end distance which is found to be proportional to $N^{2\times\frac{3}{5}}$ theoretically is proportional to $N^{2\times\nu}$. Our simulation result also assures this idea. As can be seen from the figure 4.2 the plot best fits at $\nu = 0.588$. From this we can conclude that the end-to-end distance is proportional to $N^{0.588}$. A best fit can be found if we can consider larger number of monomers. Due to very time taking computation we could not go beyond 50 monomers. To produce this result, the program takes about 72 hours if the graph of energy versus time is to be plotted while the simulation is running and abut 48 hours if we are running only to get the graph of $\langle R^2 \rangle$ and$\langle R_g^2 \rangle$.

The radius of gyration, as given in equation 1.37 and 1.37 , is given by

$$\langle R_g^2 \rangle = \frac{\langle R^2 \rangle}{6}(1 - \frac{1}{N^2}) \tag{4.4}$$

and for large N this can be approximated as

$$\langle R_g^2 \rangle \approx \frac{\langle R^2 \rangle}{6}. \tag{4.5}$$

In this essay, since we have only consided smaller number of monomers, equation 4.4 is used as a fitting curve for the radius of gyration. The plot shows that it quit similar to the simulation result. From this we can conclude that for large N, $\langle R_g^2 \rangle \approx \frac{\langle R^2 \rangle}{6}$. For the case the end-to-end distance we have to consider larger number of polymer for a best fit since we are using some assumption in calculating the theoretical value.

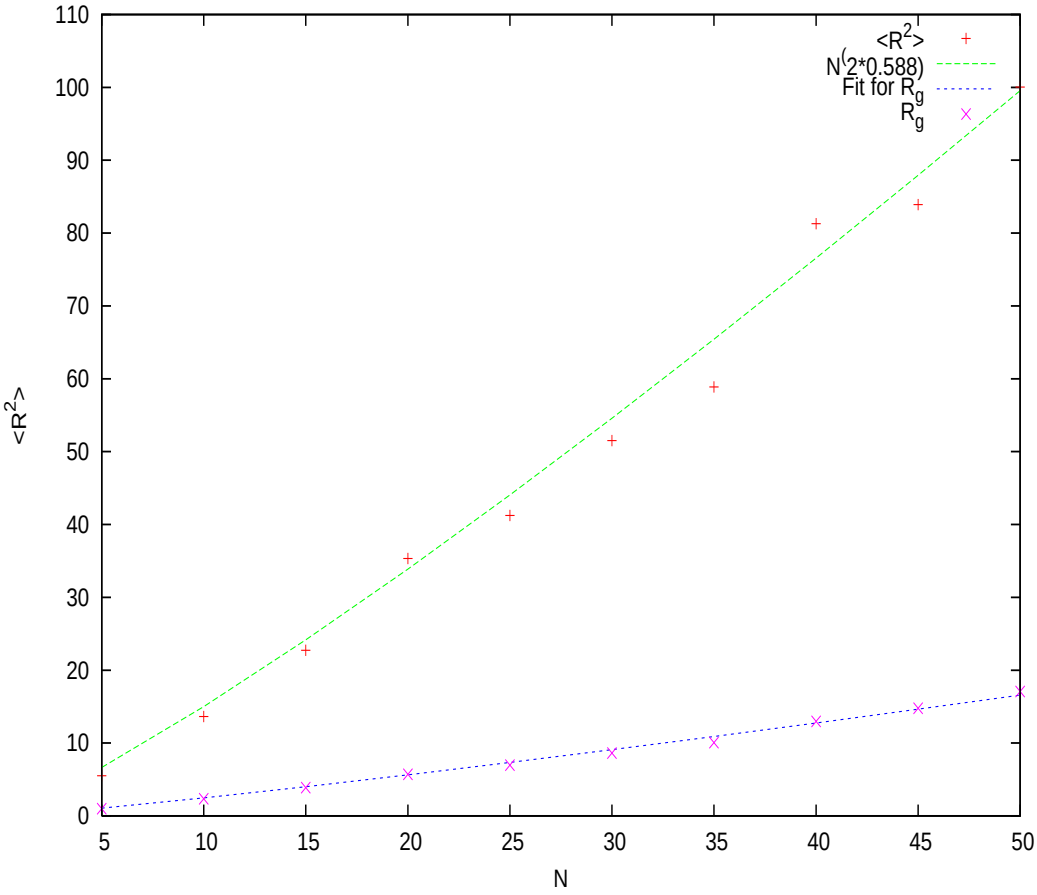

Figure 4.2: Mean square end-to-end distance and radius of gyration (for 100,000 iteration), where $\langle R^2 \rangle$ is the simulation result for mean square end-to-end distance, R_g is the simulation result for square of radius of gyration, $N(2 * 0.588)$ is the theoretical fit (equation 1.30 for $\nu = 0.588$) for mean square end-to-end distance, and fit for R_g is the theoretical fit (equation 4.4) for square of radius of gyration.

[]

We hope that the program on the appendix works provided that the PC is installed with python 2.4.

5. Conclusion

Classically a simple polymer chain can be understood as a sequence of masses joined by some potential. For chains that cannot be stretched infinitely far, and that have mutually excluded beads, it is very difficult to calculate chain properties analytically. In these cases, one usually has to resort to computation. Molecular dynamics simulations enable the study of chain properties and are widely used in physics.

In this essay, we studied the background of this simulation method and analysed the properties of various models of polymer chains. The physical properties analysed using this simulation result were polymers end-to-end distance and its radius of gyration.

In the simulation, the 6-12 Lennard-Jones and Finitely Extensible Non-Linear Elastic (FENE) potentials are used as model potentials between non-bonding and bonding monomers, respectively. The Verlet velocity algorithm was used for the time integration algorithm. At each step, the conservation of total energy was monitored to check the stability of the system.

Using a time step of 0.005 the simulation result shows that the polymer mean square end-to-end distance is proportional to $N^{2\nu}$, where $\nu = 0.588$ and the radius of gyration for larger N is found to be proportional to $\frac{1}{6}\langle R^2 \rangle$. To compare these simulation results with the theoretical values, the data was fitted with values of $\langle R^2 \rangle$ and $\langle R_g^2 \rangle$ that had been calculated theoretically.

In general, we have shown that molecular dynamics is a powerful technique to simulate non-equilibrium dynamics of polymers and other soft matter. By controlling the time step and the initial configuration a deterministic result can be found of for any system involving dynamics.

Appendix A. Code

```
##This program run on python2.4 soft ware
#Zelalem Nigussa Urgessa (zolahan2005@yahoo.com)
#NB: "here" when U see this command, it is to say that
#the line is long and I made to be on "new line".
#to run the program U must remove "here" command
#and bring back the next line to be on one line
# to the line "here" is written.
from __future__ import division
from scipy import *
from random import *
from math import *
from visual import *
import MA
import Gnuplot
gp = Gnuplot.Gnuplot(persist=1)
RR=[]
RR_gy=[]
Rfit=[]
RfitRgy=[]
for N_monomers in xrange(10,51,5): #the number of monomers to be used
    sigma = 1 #a constant
    epslom = 1 #a constant
    k = 30 * epslom / sigma**2 #a constant
    r0 = 1.5 * sigma #a constant
    rc = 2.5 * sigma #a constant
    ini_min = 0.95 * sigma #minimum initial separation
    ini_max = 1.0 * sigma #maximum initial separation
    lattice_sz = int(2*sqrt(N_monomers)) #the size of the lattice
    lattice_sp = (ini_min + ini_max)/2 #distance between each monomers
    lattice_rd = (ini_max - ini_min)/4 #amplitude of displacement for randomization
    dxns = [[0,0,1], #the faces to be selected once the first monomer is placed
            [0,1,0],
            [1,0,0],
            [0,0,-1],
            [0,-1,0],
            [-1,0,0]]
          #the place where the first monomer is placed
    center = (lattice_sz//2) * ones((3,), Int)
    list_occ_pos = [list(center)] #the occupied list
    def dxnmap(i): # This calls the faces which are taken
        return dxns[i]
         #to choice (which is a part in module) from unoccupied (not yet defined)
```

```
def next_step(unocc):
    return dxnmap(choice(unocc))
     # To know the neighbour positions in cluding the present position
def neighbours(pos):
    return pos + array(dxns)

def unocc_indices(theArray, occ_pos):   # this identify the unoccupied indeces
    i = 0
    sz = len(theArray)    # given lists form which to check for choice
    unocc = []
    while i < sz:
        if not list(theArray[i]) in occ_pos:
            unocc.append(i)
        i = i + 1
    return unocc   #this tells where is occupied and where is not
      #for a time being the occupied position is the center
current_pos = array(list_occ_pos[0])


      #this is loop that start puting the other monomers
for n in range(1,N_monomers):
      #This gives current position + array of the six faces
    all_neighbours = neighbours(current_pos)
    unocc = unocc_indices(all_neighbours, list_occ_pos)
      # this check an indices i in all_neigh above and
      #append for Unoccupied if not in it.
      # add to the current position
    current_pos=current_pos + array(next_step(unocc))
      # to get all the occupied possitions
    list_occ_pos.append(list(current_pos))


Position_beforPert=array(list_occ_pos)*lattice_sp
      #this is to define position befor randemization,
      #the multiplication is to make in terms of sigima
newposition = []
      #this step makes the distrubition some what random by displacing each
      #particles by r at some random angle theta and phi in spherical coordinate
r = stats.rand.uniform(0, lattice_rd, N_monomers)
      #for more on how stats.random works, see on help window
theta = stats.rand.uniform(0,pi, N_monomers)
phi = stats.rand.uniform(0,2*pi, N_monomers)
     #component wise
x = r*cos(phi)*sin(theta)
y = r*sin(phi)*sin(theta)
```

```
z = r*cos(theta)


displacements = transpose(concatenate(([x],[y],[z]))) #reshaping
     # position after perturbation
Position_afterPert = Position_beforPert + displacements
     #this step is for visualization on the screen while the program is ranning
     #To place particles on the center of screen for running
CM = average(Position_afterPert)
max_R = max(Position_afterPert)
min_R = min(Position_afterPert)
scene2 = display(title = 'The Polymer', width = 600, height ="here"
           400, center = CM, background=(0,0,1))
      #the size and background of the screen
scene2.range = 2*(max(max_R))*array([1,1,1]) #dimension of the screen
     #this step create beads as balls and connect to each other
balls = []
for i in arange(0,N_monomers): #it is just the number of monomers
    balls.append(sphere(pos = list(Position_afterPert[i]), "here"
             radius = 0.1*sigma, color = (1,0,0) ))
     #to connect each balls with line
c = curve( x = Position_afterPert[:,0], y = Position_afterPert[:,1], "here"
               z = Position_afterPert[:,2])


    #this step is simply a code to see
def visualize(Position_afterPert):
    for i in arange(0,N_monomers):
        balls[i].pos =   Position_afterPert[i]
    c.x = Position_afterPert[:,0]
    c.y = Position_afterPert[:,1]
    c.z = Position_afterPert[:,2]
    return

    # all steps above is configuration
    # after this we start the simulation
r = Position_afterPert #initial configuration of the system
    #component wise
x = r[:,0]
y = r[:,1]
z = r[:,2]
      #initial end to tnd distance, but has no-thing to do. We can remove
      #After this we have to decided the range of interaction
def PhysicalQ(x, y, z, vx, vy, vz, only_compute_force = False):
      #relative position of all particles
```

```
  #all, dx,dy and dy are N_monomer by N_monomer arrayies
dx = array([x]) - transpose([x])
dy = array([y]) - transpose([y])
dz = array([z]) - transpose([z])
dr = sqrt(dx**2 + dy**2 + dz**2)
  # This step solves the Magnitude of Forces:
  #    Lennard-Jones:
  #the LJ potential is zero if the distance
  #between any two monomer is >2.5*sigma
  # again distance between a monomer and it self
  #is zero so that the diagonal of dr is zero
  #eye(N_monomers) returns all the diagonals 1 and 0 else where,
  #in Mask zero is for value and 1 is not valid
is_not_LJ_interacting = (dr > 2.5*sigma) + eye(N_monomers)
  #is_not_LJ_interacting returns 1 for dr > 2.5*sigma and 0 for
  #dr < 2.5*sigma also 1 for diagonal elements
  # 0's are the place where the matrix has got values and 1s are where
  #the elements are not valied
dr_LJ_MA = MA.masked_array(dr, is_not_LJ_interacting)
  #Mask (MA.masked_array) is a module that returns an array with
  # 1 for invalid values 0 for valied values
  #in dr the values of dr >2.5*sigma are masked here since they are invaild.
  #that means the module gives them a number 1
  #and these values do not involve in any calculation
F_LJ = 24 * epslom * ( 2*sigma**12/dr_LJ_MA**13 -  sigma**6/dr_LJ_MA**7 )
  #this force is a force between any two monomers
  #that are separated by dr < 2.5*sigma
  #    FENE:
  #linalg.basic.toeplitz returns an array with 1's adjcent to the
  #diagonal (this is in owr case,
  #we can make as we like) and 0's else where.
  #logical_not is negation
is_not_bond = logical_not(linalg.basic.toeplitz(array([0, 1] + "here"
              [0]*(N_monomers-2)), ))
   #this "is_not_bond returns" a matrix with 1 for
   #adjacent elements and 1 eslse where
   #therefore, the valid elements are those with 0 element
dr_FENE_MA = MA.masked_array(dr, is_not_bond)
   #as before, those elements with 1's is masked
F_FENE = - dr_FENE_MA * k * r0**2 / (r0**2 - dr_FENE_MA**2)
    #    Lennard-Jones + FENE:
Net_force = F_LJ.filled(0) + F_FENE.filled(0)
        # Components of Forces:
        #to reconstruct the forces and to find their components
        #this is to remove the force due to it self
```

```
    self_distance = eye(N_monomers)
            #this asures there is no zero division
    dr_valid = MA.masked_array(dr, self_distance)
    Fx_net = dx * Net_force / dr_valid
    Fy_net = dy * Net_force / dr_valid
    Fz_net = dz * Net_force / dr_valid

            # Resultant Forces on monomers
            #reconstruction
    Tot_Fx = array(MA.sum(Fx_net))
    Tot_Fy = array(MA.sum(Fy_net))
    Tot_Fz = array(MA.sum(Fz_net))
             #start up the summation of the quantities
    Tot_PE = 0.
    Tot_KE = 0.
    Tot_E = 0.
            #to calculate the quantities, if the above condition is fulfilled


    if not only_compute_force:
           # Potential Energy
           #LJ
        U_LJ = 4 * epslom * (sigma**12/dr_LJ_MA**12 -  sigma**6/dr_LJ_MA**6)
           #FENE
        U_FENE = -0.5 * k * r0**2 * log(1 - dr_FENE_MA.filled(0)**2 / r0**2)
        U_FENE = MA.masked_array(U_FENE, is_not_bond)#to reconstruct
            # half is to remove double counting
        Tot_PE = sum(ravel(U_LJ.filled(0))) / 2 + "here"
                    sum(MA.ravel(U_FENE.filled(0)))/2
            #the velocities are not yet given
        v2 = vx**2 + vy**2 + vz**2
        Tot_KE = 0.5 * sum(v2)
        Tot_E = Tot_KE + Tot_PE
            #this all are returned in termes of the defined quantities
    return Tot_Fx, Tot_Fy, Tot_Fz, Tot_PE, Tot_KE, Tot_E, "here"
                 dr[0, N_monomers - 1]**2


          #velocities
m = 1
gp = Gnuplot.Gnuplot(persist=1)
tau = sqrt(epslom / (m * sigma**2))
dt = 0.005 * tau #time step taken
g= Gnuplot.Gnuplot(persist=1)
        # Choose the kinetic Energy:
```

```
KE = 10 * N_monomers # KE per particle
        # Eliminate bulk rotation:
        ## i.e., let velocities of all monomers be along the line of action of th
        ## net forces acting on them:
vx = 0 ; vy = 0 ; vz = 0
Tot_Fx, Tot_Fy, Tot_Fz, Tot_PE, Tot_KE, Tot_E, E2E_d_sq = "here"
                PhysicalQ(x, y, z, vx, vy, vz, only_compute_force = True)
Force = sqrt(Tot_Fx**2 + Tot_Fy**2 + Tot_Fz**2)
       #this is to make force and velocity along the same direction
ini_speed = stats.rand.uniform(1, 1, N_monomers)#randomly given as 1
vx = -ini_speed * Tot_Fx / Force #to make in teh same direction as force
vy = -ini_speed * Tot_Fy / Force
vz = -ini_speed * Tot_Fz / Force
      # Eliminate bulk translation:
      ## i.e., compute velocities relative to centre-of-mass frame
      #or to remove center of mass motion
vx_cm = average(vx)
vy_cm = average(vy)
vz_cm = average(vz)
vx = vx - vx_cm
vy = vy - vy_cm
vz = vz - vz_cm
     # Rescale velocities to required Kinetic Energy (KE):
     ## (f is the scale factor)
currKE = sum(vx**2 + vy**2 + vz**2)
f = sqrt(2 * KE / (m * currKE))
vx = f * vx
vy = f * vy
vz = f * vz

Tot_Fx, Tot_Fy, Tot_Fz, Tot_PE, Tot_KE, Tot_E, E2E_d_sq = "here"
             PhysicalQ(x, y, z, vx, vy, vz)
      #just to see the center of mass velocity
print "Centre of mass velocity =", (vx_cm, vy_cm, vz_cm)
print


visualize(Position_afterPert)
Tot_Fx, Tot_Fy, Tot_Fz, Tot_PE, Tot_KE, Tot_E, E2E_d_sq "here"
           = PhysicalQ(x, y, z, vx, vy, vz)
      #now we can append all the values
lst_KE = []
lst_PE = []
lst_E = []
lst_t = []
```

```
     #to save each element
lst_PE.append(Tot_PE)
lst_KE.append(Tot_KE)
lst_E.append(Tot_E)
lst_t.append(0)
End_to_endDis=[]
Radius_gyration=[]
gp.xlabel('Time')
gp.ylabel('Energy')


Number_of_times=100000 #Iteration
for N_t_steps in range(Number_of_times):
    x1 = x + vx * dt + 0.5 * Tot_Fx / m * dt**2 #Verlet algorthm
    y1 = y + vy * dt + 0.5 * Tot_Fy / m * dt**2
    z1 = z + vz * dt + 0.5 * Tot_Fz / m * dt**2
       #to find radius of center of mass
    Xcm=sum(x1)/N_monomers
    Ycm=sum(y1)/N_monomers
    Zcm=sum(z1)/N_monomers
       # to find radius of gyration
    dxx=(array(x1)-Xcm)**2
    dyy=(array(y1)-Ycm)**2
    dzz=(array(z1)-Zcm)**2
    R_gy=(sum(dxx) +sum(dyy) +sum(dzz))/N_monomers
       #Rsq2=(x1[0]-x1[N_monomers-1])**2 +(y1[0]-y1[N_monomers-1])**2 "here"
                    +(z1[0]-z1[N_monomers-1])**2
    Tot_Fx1, Tot_Fy1, Tot_Fz1, Tot_PE, Tot_KE, Tot_E, E2E_d_sq = "here"
                   PhysicalQ(x1, y1, z1, vx, vy, vz)
    vx1 = vx + (Tot_Fx + Tot_Fx1) / m  * dt / 2#Verlet velocity algorithm
    vy1 = vy + (Tot_Fy + Tot_Fy1) / m  * dt / 2
    vz1 = vz + (Tot_Fz + Tot_Fz1) / m  * dt / 2
    Position_afterPert = transpose(concatenate(([x1], [y1], [z1]), 0)) #new posit
    visualize(Position_afterPert)
    Tot_Fx, Tot_Fy, Tot_Fz, Tot_PE, Tot_KE, Tot_E, E2E_d_sq = "here"
                  PhysicalQ(x1, y1, z1, vx1, vy1, vz1)
    lst_PE.append(Tot_PE)
    lst_KE.append(Tot_KE)
    lst_E.append(Tot_E)
    lst_t.append(N_t_steps * dt)
    End_to_endDis.append(E2E_d_sq)
    Radius_gyration.append(R_gy)
            #End_to_endDis.append(Rsq2)
            ##this step is just simply energy conservation,
            #but it takes time to see forthe whole process
```

```
                #to make the program some what faster you can comment this step
        if N_t_steps > 250:
            lst_PE.remove(lst_PE[0])
            lst_KE.remove(lst_KE[0])
            lst_E.remove(lst_E[0])
            lst_t.remove(lst_t[0])
            Temp.remove(Temp[0])
        plotKE = Gnuplot.PlotItems.Data(lst_t, lst_KE, title = 'KE', with = "lines")
        plotPE = Gnuplot.PlotItems.Data(lst_t, lst_PE, title = 'PE', with = "lines")
        plotTE = Gnuplot.PlotItems.Data(lst_t, lst_E,  title = 'TE', with = "lines")
        gp.set_range(option = 'xrange', value = (lst_t[0],lst_t[-1]))
        gp.plot(plotKE, plotPE, plotTE)
                #you can comment up to this line
                #this is to renew the position and the velocities
        vx = vx1
        vy = vy1
        vz = vz1
        x = x1
        y = y1
        z = z1
           #this step is to find the mean and the fiting curves
    Rmean=sum(End_to_endDis)/len(End_to_endDis)
    R_gymean=sum(Radius_gyration)/len(Radius_gyration)
    RR_gy.append([N_monomers,R_gymean])
    RR.append([N_monomers,Rmean] )
           #Fitting curves
    Rseq=N_monomers**(2*0.588)
    Rfit.append([N_monomers,Rseq])
    fit_Rgy=Rseq*(1-1/N_monomers**2)/6
    RfitRgy.append([N_monomers,fit_Rgy])
           #to plot
plotRmean= Gnuplot.PlotItems.Data(RR,  title = '<R^{2}>', with = "points")
plotfit= Gnuplot.PlotItems.Data(Rfit,  title = 'N^(2*0.588)', with = "lines")
plotfit2= Gnuplot.PlotItems.Data(RfitRgy,  title = 'Fit for R_g', with = "lines")
plotfit3= Gnuplot.PlotItems.Data(RR_gy,  title = 'R_g', with = "points")
g.xlabel('N')
g.ylabel('<R^{2}>')
         #To save as Eps
#gp.hardcopy(filename="Endtoend1.eps",terminal="postscript",eps=True,color=True)
         #to save as png
g("set terminal png")
g("set out 'RandRg4z.png'")
g.plot(plotRmean,plotfit,plotfit2,plotfit3 )
```

Bibliography

[1] Michael Rubinstein . Aalph H. Colby. *Polymer Physics*. Oxford University Press, 2003.

[2] Masao Doi. *Introduction to Polymer physics*. Clarendon Press, Oxford, 1996.

[3] Stuart Alan Rice, editor. *Professor Emeritus Hiromi Yamakawa: Modern Theory of Polymer Solutions*. Kyoto University, Japan. electronc edition, $http://www.molsci.polym.kyoto-u.ac.jp/archives/redbook.pdf$, 2001.

[4] Yaneer Bar-Yam. *Dyanmics of Complex system*. Westview Press, USA, 1997.

[5] Tanniemola Liverpool and Kristian K. Muller-Nedebock. Aims course study guide, random walk, polymer and biophysics, 2005. Unpublished manuscript.

[6] Federick Reif. *Fundamentals of Statistical and Thermal Physics*. University of California, Berkeley, 1965.

[7] FW Wiegel. *Introduction to path integral methods in Physics and Polymer science*. World scientific publishing Co Pte Ltd USA, 1985.

[8] Barton. *A Elements of Greens Functions and Propagation*. Oxford Science Press, 1989.

[9] Barry D. Hughes. *Random walk and random environment Volume I*. Oxford Science Publications, 1998.

[10] Kurt Kremer. Dynamics of entangled linear polymer melts: A molecular-dyanmics simulation. *J.Chem.Phy*, 92:5057–5070, 1990.

[11] M.B. Zimmt K.A. Peterson and M.D. Fayer. Short polymer chain stastistics and the relation to end to end electronic excitation. *American Chemical Society*, 88:1145–1154, 1988.

[12] Furio Ercolessi. A molecular dynamics primer, 1997. Unpublished manuscript.

[13] J.M. Thijssen. *Computational Physics*. Cambridge University Press, 1999.

[14] Nicholas J. Giordano. *Computational Physics*. Prentice Hall Upper Saddle River New Jersey, 1997.

[15] Darryl D. Humphreys, Richard A. Friesner, and Bruce J. Berne. A multiple-time-step molecular dynamics algorithm for macromolecules. *J.Chem.Phy*, 98:6885–6892, 1994.