

FOR REFERENCE

NOT TO BE TAKEN FROM THIS ROOM

MOLECULAR DYNAMICS SIMULATION OF POLYMERIC SYSTEMS

by

BERTAN BADUR

B.S in Chemical Engineering
BOGAZIÇI UNIVERSITY 1986

M.S. in Industrial Engineering
BOGAZIÇI UNIVERSITY 1989

Bogazici University Library



39001100131054

14

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science
in
Chemical Engineering

BOGAZIÇI UNIVERSITY

1992

ACKNOWLEDGMENTS

I would like to express my deep gratitude to Doç. Dr. İvet Bahar for introducing me this topic and for her invaluable guidance as the supervisor of this thesis and for her encouragement , continuous inspiration and understanding throughout the course of this thesis.

I wish to thank Prof. Dr. Burak Erman and Yrd. Doç. Dr. Uğur Akman for their helps and serving on my thesis committee. I would like to thank to Türkan Haliloğlu for her interest and help during the course of this work. I also would like to thank Nurcan Kum for her contributions during the completion of the work.

ABSTRACT

In this thesis, the dynamic and static properties of a polymeric chain in a dilute solution was investigated with the method of molecular dynamics (MD) simulation.

We investigated both the equilibrium properties of polymer chains in dilute solution, such as the end-to-end distance, the radius of gyration and the radial distribution functions related to non-bonded pair correlations, and the dynamic properties of the chain such as the center of mass velocity, the orientational autocorrelation functions and the diffusion coefficient. A single oligomeric chain of 12 beads in a solvent is considered. The quality of the solvent is modified by proper modification of the polymer-solvent interaction energy parameter and the behavior of the chain in various solvents is investigated.

ÖZET

Bu çalışmada bir polimer çözeltisinin dinamik ve statik özellikleri moleküler dinamik yöntemi kullanılarak incelenmiştir.

Seyreltik bir polimer çözeltisinin ortalama konum vektörü ve ikili dağılım fonksiyonları gibi statik özellikleri , yayılım katsayısı ve hız otokorelasyon fonksiyonu gibi dinamik özellikleri incelenmiştir. 12 unite den oluşan bir zincirin davranışları polimer çözücü etkilşim parametresinin değiştirildiği farklı çözeltilerde araştırılmıştır.

TABLE OF CONTENTS

	<u>PAGE</u>
ACKNOWLEDGEMENTS	III
ABSTRACT	IV
ÖZET	V
TABLE OF CONTENTS	VI
LIST OF FIGURES.....	VIII
LIST OF TABLES	IX
LIST OF SYMBOLS	X
I. INTRODUCTION	1
1.1 Motivation	2
1.3 Literature Survey	4
II. THEORY	6
2.1 Molecular Dynamics	6
2.1.1 The Verlet Algorithm	7
2.1.2 Interaction Potentials	8
2.1.3 Periodic Boundary Conditions	12
2.1.4 The Initial Configuration	13
2.1.5 The Initial Velocities	15
2.2 Static and Dynamic Properties	16
2.2.1 Mean-square End-to-end Separation and Radius of Gyration	16
2.2.2 The Pair Distribution Function	17

2.2.3 The Time - Correlation Functions	17
2.2.4 Diffusion Coefficient	18
III. DESCRIPTION OF THE SIMULATION	19
3.1 Introduction	9
3.2 Description of the Simulation Program	21
3.2.1 The Basic Algorithm	22
3.2.2 Inputs	24
3.2.3 Initial Positions and Velocities	27
3.2.4 Generation of Polymer chain	29
3.2.5 The Initial Step	29
3.2.6 Programming Modified Verlet Algorithm	30
3.2.7 Calculation of Forces	32
3.2.8 Calculation of Static and Dynamic Properties	33
IV. RESULTS AND DISCUSSION	36
4.1 Results	36
4.1.1 Chain Configuration	38
4.1.2 The Pair Distribution Functions	39
4.1.3 Autocorrelation Functions and Diffusion Coefficients ...	41
4.2 Remarks and Further Developments	43
APPENDIX A. THE SIMULATION PROGRAM	44
APPENDIX B. THE DATA FILE	76
APPENDIX C. THE SIMULATION OUTPUTS	77
REFERENCES	89

LIST OF TABLES

	<u>PAGES</u>
TABLE 3.1 Reduced Variables in terms of Ordinary Variables	20
TABLE 3.2 Summary of the Algorithm of the Program	23
TABLE 4.1 Energy Parameters , Chain Configuration Results and Diffusion Coefficients.....	36
TABLE 4.2 Values of Fixed Parameters in terms of Reduced and Real Units	37
TABLE 4.3 Potential and Kinetic Energies, and Temperature in Reduced Units.....	38

LIST OF FIGURES

	<u>PAGES</u>
FIGURE 2.1 Magnitude of the Interaction Potential (Solid Line) and Shifted Force Potential (Dashed Line) for (a) the Lennard-Jones Potential and (b) its shifted Modification	10
FIGURE 2.2 Unit Cell of the Modified b.c.c. Lattice	14
FIGURE 2.3 Generating Vectors of the Tetrahedral Lattice and Example of a Random Walk	14
FIGURE 4.1 Solvent-Solvent Pair Distribution Functions	39
FIGURE 4.2 Polymer-Solvent Pair Distribution Functions	40
FIGURE 4.3 Polymer-Polymer Pair Distribution Functions	41
FIGURE 4.4 Center of Mass Velocity Autocorrelation Functions	42

LIST OF SYMBOLS

m_i	: Mass of particle i
r_i	: Cartesian position vector of particle i
f_i	: Total force vector Particle i
a_i	: Acceleration vector of particle i
v_{ij}	: Lennard-Jones 6-12 potential between particle i and j
V^s	: Shifted potential
V^{SF}	: Shifted force potential
σ	: Distance parameter
ϵ	: Energy parameter
ϵ_{ss}	: Energy parameter (solvent-solvent)
ϵ_{ps}	: Energy parameter (polymer-solvent)
ϵ_{pp}	: Energy parameter (polymer-polymer)
U	: Total potential energy
W	: Internal virial
P	: Instantaneous pressure
ρ	: Density
k_B	: Boltzmann constant
T	: Absolute temperature
V	: Volume of the simulation box
v_{ix}	: x component of velocity of particle i
$P(v_{ix})$	: Normalized probability density function for v_{ix}
$\langle r^2 \rangle$	: Mean-square end-to-end distance
$\langle s^2 \rangle$	: Radius of gyration
N_t	: Number of time steps that the simulation has run over
n	: Number of beads in the chain
$r_{k(i)}$	: Position vector of the kth bead at the ith time step
r_{cm}	: Position vector of the center of mass of the chain
$\langle g_{ki}(r) \rangle$	: Pair distribution function for the pair of particle (k,i) of type polymer-polymer, polymer-solvent, solvent-solvent
$C(a;t)$	: Time correlation function of a vectorial property a
D_p	: Diffusion coefficient for the translation of polymer chain
$C(v_{cm})$	: Autocorrelation of center of mass velocity

ρ^*	: Reduced density
T^*	: Reduced temperature
E^*	: Reduced energy
P^*	: Reduced pressure
τ^*	: Reduced time
f	: Reduced force
C	: Number of lattice cells along each side of the simulation box
N	: Total number of particles in the simulation box
$\langle n_{ki}(r) \rangle$	: Average number of particles of type k (solvent or polymer) contained in the interval Δr centered at a distance r from a particle of type i (solvent or polymer)

CHAPTER I

INTRODUCTION

1.1. Motivation

In recent years there has been a considerable interest in the study of polymeric systems with simulation techniques. The development of high speed computers made it possible to study more complicated systems.

In this thesis, the dynamic and static properties of a polymeric chain in a dilute solution were investigated with the method of molecular dynamics (MD) simulation.

There are basically two major methods to study polymeric systems: the Monte Carlo (MC) technique which is commonly employed to investigate static properties of macromolecules, and the Molecular Dynamic (MD) method which is essentially used to study the dynamics of chain molecules. Brownian dynamics which may be viewed as a branch of molecular dynamics method is also used as an efficient method to investigate chain dynamics [Allen 1986].

The MD technique simulates the dynamic behavior of particles, by the numerical solution of Newton's equation of motion, given the initial positions and velocities of atoms and

molecules and a potential energy function. Thus, when applied to polymer solutions, this method includes the solvent-polymer, polymer-polymer and solvent-solvent interactions.

In this thesis, we investigated both the equilibrium properties of polymer chains in dilute solution, such as the end-to-end distance, the radius of gyration and the radial distribution functions related to non-bonded pair correlations, and the dynamic properties of the chain such as the center of mass velocity, the orientational autocorrelation functions and the diffusion coefficient. A single oligomeric chain of 12 beads in a solvent is considered. The quality of the solvent is modified by proper modification of the polymer-solvent interaction energy parameter and the behavior of the chain in various solvents is investigated.

1.2 Basic Description of Molecular Dynamics

Molecular dynamics simulations is the technique used to obtain dynamic properties of many particle systems. In this method the classical equations of motion are solved for a set of molecules.

In the simulation of polymer-solvent systems with the MD technique a certain number of solvent molecules (atoms) and of polymer chains consisting of a certain number of polymer units

are originally assigned coordinates in a specified space called the simulation box. In many practical applications this is a cubic box with sides of 5-15 units in terms of the diameter of simulated particles. Each particle is also assigned an initial velocity, at the start up.

The inter- and intramolecular interactions between non-bonded particles are conveniently described in terms of potential energy functions. The forces between two particles can be obtained by differentiating the corresponding potential energy function with respect to the displacement of either particle. Given the initial position and velocity of each particle and the force acting on it, the position and velocities as a function of time are obtained from the solution of the classical equations of motion.

Desired static and dynamic properties are calculated from positions and velocities of particles by taking samples at certain time intervals during the course of simulation. In the simulation of bulk liquids with a finite number of particles, to avoid surface effects, periodic boundary conditions (section (2.2)) are applied.

1.3 Literature Survey

Among the two simulation methods mentioned above, the simpler Monte Carlo method generates samples of various conformations usually placed in lattices [Kremer, 1988]. These methods are very efficient in establishing or verifying scaling laws [de Gennes, 1988] of equilibrium properties, since they can be used for very long chains [Madras, 1988] or for highly concentrated systems [Kramer, 1983]. However the dynamics of the chain can not be reproduced in a realistic way since the MC algorithms are only able to simulate a priori defined stochastic processes.

Brownian dynamics offers a more realistic approach [Pear, 1979] in which a stochastic interaction between polymer and solvent is introduced. In this method the description of frictional forces is done through equations based on the continuous hydrodynamic theory.

MD simulations on the other hand can take care of the interaction of polymer molecules with both solvent and themselves. These type of studies were initiated by [Bishop et al., 1979] and [Rappaport, 1979]. Subsequently Bruns, [Bruns, 1981], Bansal [Bansal, 1983] and Bishop and collaborators [Bishop et al., 1983] observed a great influence of the form and parameters of the interaction potential on the final results. Oh

and collaborators [Oh et al., 1983] obtained results for chains of different lengths and with different values of the polymer-solvent interaction parameters. Static properties have also been investigated by Khalatur [Khalatur et al., 1986]. A very recent study of the influence of the solvent quality have been carried out [Smit et al., 1988]. Luque et al. [Luque, 1989] investigated the effect of chain length and solvent quality.

CHAPTER II

THEORY

2.1. Molecular Dynamics

Molecular dynamics is the technique used to solve the classical equations of motion for a system of N particles interacting via an overall potential U . The equation of motion for the i th particle is

$$m_i \ddot{\mathbf{r}}_i = \mathbf{f}_i \quad (2.1.1)$$

where m_i is the mass of particle i , $\mathbf{r}_i$ is the cartesian position vector of particle i with respect to a laboratory-fixed frame and $\mathbf{f}_i$ is the total force on particle i , which is given by the negative gradient of U with respect to $\mathbf{r}_i$, as

$$\mathbf{f}_i = -\nabla_{\mathbf{r}_i} U \quad (2.1.2)$$

The solution of the set of N equations of motion yields the trajectories (i.e. the time evolution of the position coordinates) of the N particles. Those particles represent in the present study, either the solvent molecules or each of the

structural units (united backbone atoms/groups) of the polymer chain.

A standard method for the solution of ordinary differential equations is the finite difference approach. In this method; given the positions, velocities and if necessary accelerations of atoms or molecules at time t , the positions, velocities etc. at a later time $t+\Delta t$ are obtained to a sufficient degree of accuracy.

2.1.1 The Verlet Algorithm

The most widely used method of integrating the equation of motion is that initially adopted by Verlet [Verlet, 1967] and attributed to Starnes [Gear, 1971]. This method is a direct solution of the second order equation (2.1.1). The method is based on the determination of positions vectors $\mathbf{r}_i(t+\Delta t)$ for particle i , from knowledge of previous position vectors $\mathbf{r}_i(t)$ and $\mathbf{r}_i(t-\Delta t)$, and from the accelerations $\mathbf{a}_i(t)$. The recurrence equation for advancing the positions of particle i reads

$$\mathbf{r}_i(t+\Delta t) = 2\mathbf{r}_i(t) - \mathbf{r}_i(t-\Delta t) + (\Delta t)^2 \mathbf{a}_i(t) \quad (2.1.3)$$

The equation for calculating the velocity $\mathbf{v}_i(t)$ is given by

$$\mathbf{v}_i(t) = [\mathbf{r}_i(t+\Delta t) - \mathbf{r}_i(t-\Delta t)] / 2(\Delta t) \quad (2.1.4)$$

The Verlet algorithm has some modified versions in literature. This algorithm and its modifications are discussed by Allen and Tildesley [Allen, 1986] in great detail. One of the modified algorithms [Allen, 1986, pp 84] is given by

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \Delta t \mathbf{v}_i(t) + \frac{2}{3}(\Delta t)^2 \mathbf{a}_i(t) - \frac{1}{6}(\Delta t)^2 \mathbf{a}_i(t - \Delta t) \quad (2.1.5)$$

$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \frac{1}{3}\Delta t \mathbf{a}_i(t+\Delta t) + \frac{5}{6}\Delta t \mathbf{a}_i(t) - \frac{1}{6}\Delta t \mathbf{a}_i(t - \Delta t) \quad (2.1.6)$$

It is clear from the above equations that for the evaluation of the new position vectors $\mathbf{r}_i(t+\Delta t)$ and velocities $\mathbf{v}_i(t+\Delta t)$ at a given step, it is necessary to store $\mathbf{r}_i(t)$, $\mathbf{v}_i(t)$, $\mathbf{a}_i(t)$ and $\mathbf{a}_i(t-\Delta t)$. In spite of the complexity of these formulas and the need to store the old accelerations, the above algorithm is preferred over the classical Verlet algorithm due to its more accurate calculation of velocities.

2.1.2 Interaction Potentials

One of the simplest idealized pair potentials commonly used in computer simulations is the Lennard-Jones 6-12 potential V_{LJ} which is given by

$$V_{LJ}(r_{ij}) = 4 \epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^6 \right] \quad (2.1.7)$$

where r_{ij} is the separation $|\mathbf{r}_j - \mathbf{r}_i|$ between the pair of non bonded interacting particles i and j . ϵ and σ are the energy and distance parameters, respectively, corresponding to that particular pair of particles. σ is taken to be equal to the sum of the van der Waals radii of particles i and j .

In molecular dynamic simulations, for computational efficiency, the Lennard-Jones potential is cut off at a distance r_c which is usually a multiple of σ . The truncation of the potential at a cutoff introduces some difficulties in the MD method. The function contains a discontinuity at the cutoff distance r_c ; whenever a pair of particles crosses that limit, the total energy will not be conserved. This can be avoided by shifting the potential function by an amount $V_{LJ}(r_c)$ as

$$\begin{aligned} V^S(r_{ij}) &= V_{LJ}(r_{ij}) - V_{LJ}(r_c) \quad \text{for } r_{ij} < r_c \\ V^S(r_{ij}) &= 0 \quad \text{for } r_{ij} \geq r_c \end{aligned} \quad (2.1.8)$$

Here V^S represent the shifted potential. There is still a problem with the shifted potential expression since its derivative, the force is discontinuous at r_c . To avoid this difficulty, the so-called 'shifted force potential' $V^{SF}(r_{ij})$ is adopted [Allen 1986, pp 145], as

$$V^{SF}(r_{ij}) = V^S(r_{ij}) - (r_{ij} - r_c) \left(\frac{dV_{LJ}(r_{ij})}{dr_{ij}} \right)_{r_{ij}=r_c} \quad (2.1.9)$$

In eq (2.1.9) the linear term is added to the potential so that its first derivative equates to zero at the cutoff distance. The Lennard-Jones and shifted force potentials are shown in Figure 2.1 as a function of r_{ij} .

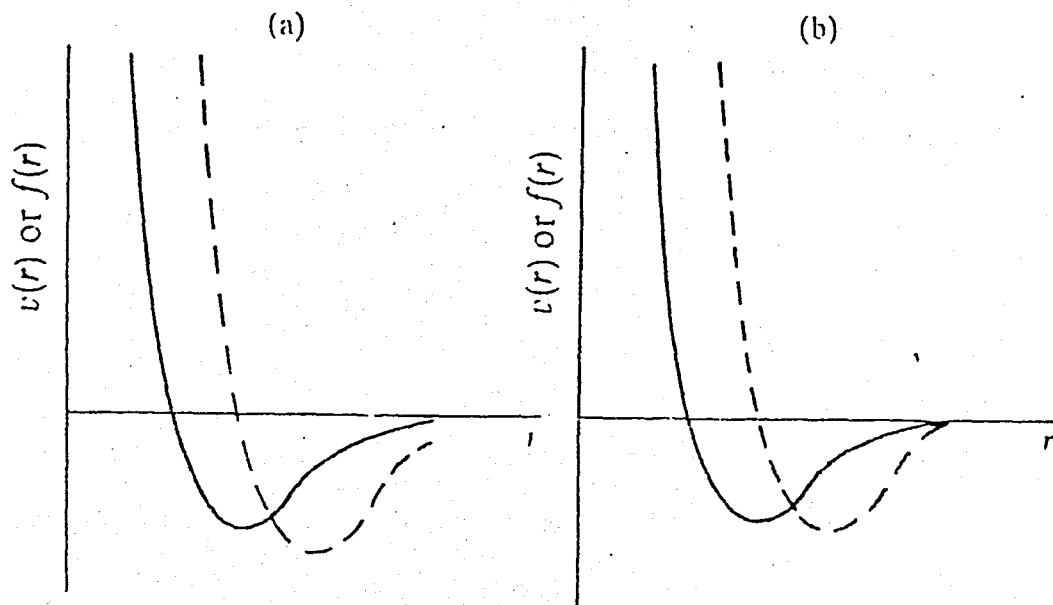


Figure 2.1 Magnitude of the Interaction Potential (Solid Line) and Shifted Force Potential (Dashed Line) for (a) the Lennard-Jones Potential and (b) its shifted Modification

One of the potential energy functions used in literature [Luque, 1989] between bonded polymer units k and $k+1$, is the empirical potential

$$V(r_{k,k+1}) = -0.5 k_0 r_0^2 \ln [1 - (r_{k,k+1}/r_0)^2] \quad (2.1.10)$$

for $0 < r_{k,k+1} < r_0$, and $V(r_{k,k+1}) = 0$ for $r_{k,k+1} > r_0$.

Here $r_0 = 1.95\sigma$ and $k_0 = 20\epsilon/\sigma^2$. The separation $r_{k,k+1}$ is nothing else than the magnitude l_k of the bond vector $\mathbf{l}_k$, connecting atoms k and $k+1$ on a given chain and therefore $r_{k,k+1}$ will be replaced by l_k in the following.

The overall potential U in the system of N particles is given by the summation

$$U = \sum_j \sum_i v^{SF}(r_{ij}) + \sum_k V(l_k) \quad (2.1.11)$$

Here the first summation is performed over all pairs i and j of non-bonded atoms/units. The second summation is carried out over all bonds in the polymer chain(s). Using the above notation, the internal virial W becomes [Allen 1986, pp 47]

$$W = -1/3 \left\{ \sum_i \sum_j \mathbf{r}_{ij} \cdot \nabla_{\mathbf{r}_{ij}} v^{SF}(r_{ij}) + \sum_k \mathbf{l}_k \cdot \nabla_{\mathbf{l}_k} V(l_k) \right\} \quad (2.1.12)$$

This quantity is used in the evaluation of the instantaneous pressure P , according to

$$P = \rho k_B T + W/V \quad (2.1.13)$$

Here ρ is the density of the system, k_B is the Boltzmann constant, T is the absolute temperature and V is the volume of the simulation box.

2.1.3 Periodic Boundary Conditions

Molecular dynamics simulations are usually performed for a limited number of molecules, $N < 5000$. The size of the system is determined by the available size of the computer and by the speed of execution of the program. In simulations of small systems such as liquid drops or a microcrystal, the cohesive forces between molecules may be sufficient to hold the system together unaided during the course of simulation. In the simulation of bulk liquids on the other hand, a major obstacle is the large fraction of molecules which lie on the surface of any small sample. These molecules will experience quite different forces from molecules in the bulk.

The problem of surface effects can be overcome by implementing periodic boundary conditions [Allen 1986, pp 24]. A cubic box is replicated throughout space to form an infinite lattice. In the course of the simulation, as a molecule moves in the original box, its periodic image in each of the neighboring boxes moves in exactly the same way. Thus, as a molecule leaves the central box, one of its images will enter through the opposite face. There are no walls at the boundary of the central box and no surface molecules.

2.1.4 The Initial Configuration

Before starting the simulation the particles forming the system must have an initial configuration. In MD simulations it is convenient to start from a lattice. Almost any lattice is suitable. Historically, the face centered cubic (fcc) structure with its $4M^3$ ($M = 2, 3, 4, \dots$) lattice parts has been the starting configuration for many simulations [Allen 1986, pp168].

In the present work, to conform closely to realistic polymer geometry, the original polymer chain is generated from a modified body-centered cubic lattice. Accordingly, there is an atom at the center of each cubic cell and four possible sites for the location of the next atom. Those sites are the opposite corners of the upper and lower faces of the cell, as shown in Figure 2.2. Two consecutive bonds belonging to a given chain are assigned directional vectors from the eight vectors

group I

$$\mathbf{a} = (1, 1, 1)$$

$$\mathbf{b} = (1, -1, -1)$$

$$\mathbf{c} = (-1, 1, -1)$$

$$\mathbf{d} = (-1, -1, 1)$$

group II

$$\mathbf{a}' = (-1, -1, -1)$$

$$\mathbf{b}' = (-1, 1, 1)$$

$$\mathbf{c}' = (1, -1, 1)$$

$$\mathbf{d}' = (1, 1, -1)$$

The following conditions are obeyed during chain generation: The directional vectors corresponding to successive bonds must belong

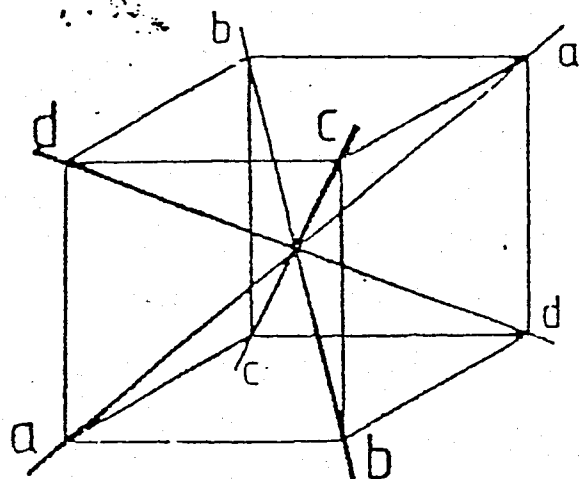


Figure 2.2 Unit Cell of the Modified b.c.c. Lattice

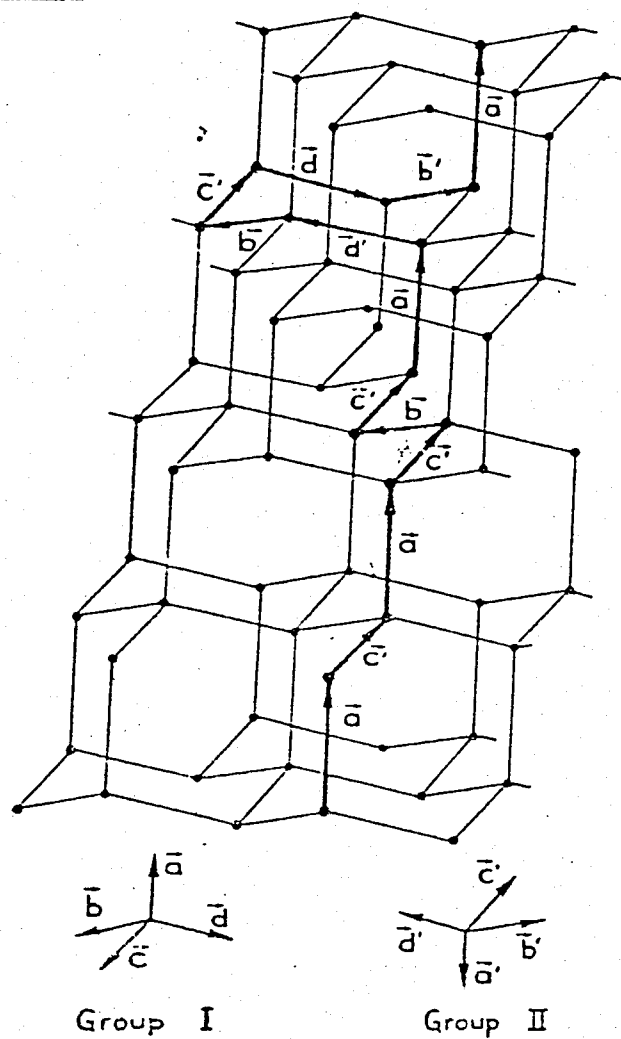


Figure 2.3 Generating Vectors of the Tetrahedral Lattice and Example of a Random Walk

to different groups and can not be of opposite sense (no back steps). An example of a walk fulfilling these conditions leads to a tetrahedral structure as shown Figure 2.3. It may be represented by the vector series $a \ c' \ a \ c' \ b \ c' \ a \ d' \ b \ c' \ b \ d' \ b' \ a$.

2.1.5 The Initial Velocities

For an MD simulation the initial velocities of all the atoms or molecules must be specified. It is usual to choose random velocities with magnitudes conforming to the Maxwell-Boltzmann distribution, for the temperature of simulations. Accordingly, the components v_{ix} , v_{iy} and v_{iz} of the velocity $\mathbf{v}_i$ of each atom i are chosen from the Gaussian distribution

$$P(v_{ix}) = (m_i/2\pi k_B T)^{1/2} \exp(-m_i v_{ix}^2/2k_B T) \quad (2.1.14)$$

Here $P(v_{ix})$ is the normalized probability density function for v_{ix} . The randomly generated initial velocities are then corrected so as to provide zero overall momentum for the system [Allen 1986, pp 170]. For that purpose, as a first step the average momentum vector is evaluated for the whole system and then, subtracted from the individual momentum vectors corresponding to each atom/group. In the particular case of particles having all the same mass m_i , this operation is

equivalent to readjusting the velocities by taking their difference with respect to the mean velocity.

2.2 Static and Dynamic Properties

In this section the static and dynamic properties of polymeric systems which are investigated by MD simulation are presented.

2.2.1 Mean-square End-to-end Separation and Radius of Gyration

The configurational properties of a polymer chain is characterized by the mean square end-to-end distance $\langle r^2 \rangle$ between the terminal atoms and the mean square radius of gyration $\langle s^2 \rangle$. These averages are found from

$$\langle r^2 \rangle = \frac{1}{N_t} \sum_{i=1}^{N_t} |(\mathbf{r}_n(i) - \mathbf{r}_1(i))|^2 \quad (2.2.1)$$

and

$$\langle s^2 \rangle = \frac{1}{N_t} \frac{1}{n} \sum_{i=1}^{N_t} \sum_{k=1}^n |(\mathbf{r}_k(i) - \mathbf{r}_{cm}(i))|^2 \quad (2.2.2)$$

where N_t is the number of time steps that the simulation has run over, n is number of beads or units in the chain, $\mathbf{r}_k(i)$ is the position vector of the k th bead at the i th time step. $\mathbf{r}_{cm}(i)$ is

the position vector of the center of mass of the chain which is evaluated at a given step i from

$$\mathbf{r}_{\text{cm}}(i) = \frac{1}{N_b} \sum_{k=1}^n \mathbf{r}_k(i) \quad (2.2.3)$$

2.2.2 The Pair Distribution Function

The pair distribution function $g(r)$ is the average number of atoms at a distance r from a given atom compared with the number at the same distance in an ideal gas at the same density. The pair distribution function $g_{ki}(r)$ for the pairs of particle (k,i) of the type polymer-solvent, polymer-polymer or solvent-solvent is found by

$$g_{ki}(r) = \rho^{-1} \langle n_{ki}(r) \rangle / 4\pi r^2 (\Delta r) \quad (2.2.4)$$

where $\langle n_{ki}(r) \rangle$ is the average number of particles of type k (solvent or polymer) contained in the interval Δr centered at a distance r from a particle of type i (solvent or polymer), [Allen, pp 183].

2.2.3. The Time-Correlation Functions

The time correlation function $C(\mathbf{a};t)$ of a vectorial property $\mathbf{a}$ is calculated by using the formula

$$C(a;t) = \frac{\langle a(t_0) \cdot a(t_0+t) \rangle - \langle a(t_0) \cdot a(\infty) \rangle}{\langle a(t_0) \cdot a(t_0) \rangle - \langle a(t_0) \cdot a(\infty) \rangle} \quad (2.2.5)$$

where $\langle a(t_0) \cdot a(t_0+t) \rangle$ is the average of the scalar product of $a(t_0)$ and $a(t_0+t)$, over different t_0 values for a given t . Since $\langle a(t_0) \cdot a(t_0+t) \rangle$ approaches zero for a free chain, and the term $\langle a(t_0) \cdot a(t_0) \rangle$ is independent of time and reduces to $\langle a^2 \rangle$, eq 2.2.5 may be rewritten as

$$C(a;t) = \frac{1}{\langle a^2 \rangle} \int [a(t_0) \cdot a(t_0+t)] dt_0 \quad (2.2.6)$$

2.2.4 Diffusion Coefficient

The diffusion coefficient for the translational motion of the polymer chain can be evaluated by integrating the center of mass velocity autocorrelation function as

$$D_p = \frac{1}{3} \int C(\mathbf{v}_{cm};t) dt \quad (2.2.6)$$

Here $C(\mathbf{v}_{cm};t)$ is the autocorrelation function of center of mass velocity of the polymer.

CHAPTER 3

DESCRIPTION OF THE SIMULATION

3.1 Introduction

In this thesis, the dynamic and static properties of a polymer-solvent system consisting of a chain of 12 atoms immersed in $N = 312$ molecules of solvent are investigated with MD simulation technique. The particles are placed in a cubic box of side equal to 7.4σ , where σ is the diameter of a particle. The interaction between all kinds of particle pairs is described by a Lennard-Jones shifted force potential as mentioned in Chapter 2. The adjacent polymer units are connected by a modified harmonic potential.

In this study, reduced units are used. Lengths are measured in units of σ , energies in units of ϵ . The relevant quantities in terms of reduced units are shown in Table 3.1. All particles -both polymer and solvent- are of the same mass m and diameter σ but in each of the three runs different values are used for the interaction energy parameters between polymer-polymer (ϵ_{pp}), polymer-solvent (ϵ_{ps}) and solvent-solvent (ϵ_{ss}).

TABLE 3.1Reduced variables in terms of Ordinary Variables

Density	$\rho^* = \rho\sigma^3$
Temperature	$\tau^* = k_B\tau/\epsilon$
Energy	$E^* = E/\epsilon$
Pressure	$P^* = P\sigma^3/\epsilon$
Time	$t^* = (\epsilon/m\sigma^2)^{1/2}t$
Force	$f^* = f\sigma/\epsilon$

Before the simulation starts all of the 324 particles are placed in a cubic box in accordance with a modified body-centered lattice, as presented in Section 2.1.4. A particle is selected randomly as the first bead of the polymer chain. The next bead is selected so as to give a tetrahedral bond angle between neighboring units. The method of choosing a tetrahedral chain from a lattice is described in section 2.1.4. Initial velocities are generated from a Gaussian distribution whose variance is the reduced temperature.

The equations of motion given by eqs 2.2.2 and 2.2.3 are integrated numerically by a modified Verlet algorithm [section 2.1.2]. The advantage of this algorithm is that the old accelerations are stored, which makes the calculation of

velocities more accurate than the original Verlet algorithm. The time step used in the integration of equations is $\delta t = 0.01 t^*$ where t^* is the reduced time.

The mean square end-to-end distance, the radius of gyration, the pair distribution functions of solvent-solvent, polymer-solvent and polymer-polymer units, the correlation function of center of mass velocity of polymer and the polymer diffusion coefficient are calculated for each run. The program MD.F used in simulations is presented in Appendix A.

3.2 Description of The Simulation Program

The simulation program was written with the programming language Microsoft FORTRAN 77 and run in both a Macintosh SE/30 personal computer and in a Silicon Graphics Personal Iris (SGI) 4D/35 workstation. It takes 106496 bytes of memory as a source file and 382124 bytes in binary code. In general, MD simulations take a substantial amount of CPU time. A typical run of 7000 steps takes about 80 hours on Macintosh SE/30, and 1.5 hours in SGI. Therefore the algorithms should be designed so that the programs can be restarted with minimum difficulty. The start facility enables the simulation to be broken up into manageable chunks of computing time. In the event of an unexpected machine failure the program can be started with a minimum loss of time.

3.2.1 The Basic Algorithm

The program first reads the basic parameters of simulation such as the mode of run, the number of steps and the size of the time step, using a small data file called SIM.DAT. This file is given in Appendix B. Then based on the fundamental parameters, basic quantities or frequently used expressions are calculated. The particles are placed in a modified body-centered cubic lattice. Initial velocities are generated and the polymer chain is selected so as to provide an originally tetrahedral valence bond angle between adjacent polymer beads.

In the beginning of the simulations, since the old accelerations are not defined, the classical Verlet algorithm is used instead of the modified Verlet. Then, starting from the second step, the present and old accelerations are calculated using the modified Verlet algorithm. 2000 iterations are carried out before the system comes to equilibration. At each step the modified Verlet algorithm is implemented. At specified intervals certain static and dynamic properties are calculated and recorded such as the end-to-end distance, the radius of gyration, the kinetic and potential energies.

After the system reaches equilibration, 5000 useful iterations are carried out. The calculations are similar to those before equilibration. At the end of the simulation, averages and standard deviations of relevant quantities are computed. Before the program stops, it prints the last step data: positions, velocities, acceleration on an output file (TYPE1.dat) so that the simulation can be continued by reading the instantaneous data of the last step from that file.

The summary of the algorithm of the program is shown in Table 3.2. Each step of the algorithm is explained in detail in the following sections.

TABLE 3.2

Summary of the Algorithm of the Program

1. Read data
2. Initial calculations
3. CALL BCC {initial positions}
4. CALL CONVEL {initial velocities}
5. Generation of polymer chain
6. Calculation of initial forces using classical Verlet algorithm

```

7.DO STEP = 1, NSTEP {main loop}
8.CALL MOVEAP {first part of modified Verlet algorithm}
9.Assigning and storing forces in previous step as the old
forces
10.CALL FORCEP {calculation of forces to evaluate  $a(t+\delta t)$ }
11.CALL MOVEBP {second part of modified Verlet algorithm}
12.IF MOD(STEP, IAVA) THEN
    Calculation of relevant quantities such as the
    instantaneous kinetic energy, the mean square of end-to-
    end distance, the radius of gyration, the center of
    mass velocities, etc.)
    ENDIF
13.END of main loop
14.Calculation of averages, standard deviations
15.Calculation of velocity autocorrelation function, diffusion
coefficient, pair distribution functions
16.Printing the results

```

3.2.2 Inputs

The basic parameters are read from the data file SIM.DAT. Explanation of input variables are as follows:

NTPR: is an integer variable from 0 to 4, each value of which corresponding to a different operation mode:

- The mode 0: The simulation starts from the beginning: reads the data, does both the equilibration and dynamics parts of MD simulations and calculates the averages.

- The mode 1: The simulation starts from the beginning and performs a specified number of steps (eg. 1000 steps) before equilibration and writes the last steps data and accumulated variables on a type file.

- The mode 2: The simulation starts by reading the relevant data from the intermediate type.1 file, performs a specified number of steps before equilibration and writes the last step accumulated variables on a type file.

- The mode 3: The programs reads the data from type file, completes the 2000 steps of equilibration, starts dynamics, performs a specified number of iterations and writes the relevant data on type file.

- The mode 4: The program reads the data from type file, performs dynamics, ends simulation writing data on type file.

N: Number of particles

NC: Number of polymer chains

NA: Number of polymer beads in each chain

BOX: Length of side of the cubic box in units of σ , diameter of particles.

RCUT: The cutoff distance r_c used in calculating shifted force potentials.

NSTEPA(1): An integer variable indicating the number of steps (2000) in the equilibration process.

NSTEPA(2): An integer variable indicating the number of steps in the dynamics part (5000).

IAVA: An integer variable specifying the interval at which samples are to be taken to calculate relevant quantities such as the end-to-end distance, the radius of gyration, the kinetic and potential energies. (10)

ITCOR: An integer variable specifying the interval at which the center of mass velocities are to be calculated. (10)

TEMPER: Reduced temperature.

SIGPP, SIGPS, SIGSS : σ_{ps} , σ_{pp} , σ_{ss} parameters in potential energy functions. Since the diameter of both polymer units and solvent particles are the same $\sigma_{ps} = \sigma_{pp} = \sigma_{ss} = \sigma$ and σ is taken in reduced units as $\sigma = 1$.

PSPP, EPSPS, EPSS: ϵ_{pp} , ϵ_{ps} and ϵ_{ss}

IDUM: A integer to set the seed of the random number generator subroutine.

DT: Time step in reduced units (0.01)

IPRINT: Optional print interval (20). The program prints the potential, kinetic and total energies, the square of end-to-end distances and radii of gyration, the pressure and temperature on a type file to facilitate the manipulation of simulation.

DELTA: Δr in computing pair distribution functions. (0.1 σ)

DELTAT: Size of pair distribution arrays. Number of separations r , for computing pair distribution functions. Maximum number of intervals is 50.

TCOR: The increment of time correlation function.

3.2.3 Initial Positions and Velocities

Initial positions of all particles are assigned in the subroutine BCC. The subroutine takes number of particles as an input parameter and writes the assigned x, y and z coordinates onto the arrays RX, RY, RZ which are in a common block. COMMON BLOCK 1. The allowed number of particles is given by

$$N = \frac{3}{2} C^3 \quad (3.1.1)$$

where C is number of lattices cells along each side of the simulation box. C must be an even number 2,4,6., to ensure an integer number of atoms. On each lattice site there is one atom at the center and four atoms at the corners. One eighth of each corner atom belongs to each cubic cell so that the number of atoms in each cell becomes one and a half.

Initial velocities are generated in the subroutine COMVEL which takes the number of particles and reduced

temperature as input parameters. The generated x,y,z components of velocities are stored in arrays VX , VY , VZ that are in COMMON BLOCK 1.

The COMVEL routine calls for function GAUSS to generate a normally distributed random variable whose mean is zero and variance is unity. The variance is equated to the reduced temperature by rescaling of the velocities. Each generated random variable is assigned to a x,y or z component velocity of a particle by the DO loop statement below:

```
DO 100 I = 1, N
  VX(I) = RTEMP * GAUSS ( IDUM )
  VY(I) = RTEMP * GAUSS ( IDUM )
  VZ(I) = RTEMP * GAUSS ( IDUM )
100  CONTINUE
```

Here RTEMP is the square root of the reduced temperature and GAUSS(IDUM) is a function giving a normally distributed random variable whose mean is zero and standard deviation unity,

The overall momentum in each direction is calculated and subtracted from the velocities to ensure zero momentum along each coordinate.

3.2.4 Generation of Polymer Chain

A particle is selected randomly among the total of 324 beads. Then a direction is chosen as explained in section 2.1.4. If there is a particle at that location which has not previously been selected as a part of the chain, this particle is identified as the next member of the polymer chain. If there is no such particle in any direction, the chain is canceled out. The procedure starts from beginning by selecting a new initial seed randomly and continued as above until all the chain is completed.

3.2.5 The Initial Steps

The modified Verlet algorithm requires both present $\mathbf{a}(t)$ and old $\mathbf{a}(t-\delta t)$ values of accelerations. Initially at $t=0$ only the present accelerations (forces) can be calculated from initial positions. The old accelerations are unknown, therefore the modified Verlet algorithm can not be used at the first iteration. The first iterations are performed by the classical Verlet algorithm of Section 2.1.1. The forces calculated initially at $t=0$ are stored as the old forces so that they can function subsequently to evaluate the old accelerations (forces) in the first step ($t = \Delta t$) enabling the modified Verlet algorithm to be applicable.

3.2.6 Programming Modified Verlet Algorithm

The modified Verlet algorithm mentioned in section 2.1.1 is programmed in two stages as shown below.

```

CALL MOVEAP

      DO 2010 I = 1 , NS
      FOX(I) = FX(I)
      FOY(I) = FY(I)
      FOZ(I) = FZ(I)
2010  CONTINUE

      DO 2020 I = 1 , NC
      DO 2030 J = 1 , NA
      FOCX(I,J) = FCX(I,J)
      FOCY(I,J) = FCY(I,J)
      FOCZ(I,J) = FCZ(I,J)
2030  CONTINUE
2020  CONTINUE

CALL FORCEP ( V, VC, W, STEP)

CALL MOVEBP ( DT )

```

In the MOVEAP routine positions of solvent and polymer atoms are incremented using the following program segments.

```

DO 100 I = 1, NS

RX(I) = RX(I) + DT * VX(I) + DTSQ23 * FX(I) - DTSQ6 * FOX(I)
RY(I) = RY(I) + DT * VY(I) + DTSQ23 * FY(I) - DTSQ6 * FOY(I)
RZ(I) = RZ(I) + DT * VZ(I) + DTSQ23 * FZ(I) - DTSQ6 * FOZ(I)
VX(I) = VX(I) + DT56 * FX(I) - DT6 * FOX(I)
VY(I) = VY(I) + DT56 * FY(I) - DT6 * FOY(I)
VZ(I) = VZ(I) + DT56 * FZ(I) - DT6 * FOZ(I)

100  CONTINUE

DO 110 I = 1, NC

DO 120 J = 1 , NA

RCX(I,J) = RCX(I,J) + DT * VCX(I,J)

```

```

      + DTSQ23 * FCX(I,J) - DTSQ6 * FOCX(I,J)
RCY(I,J) = RCY(I,J) + DT * VCY(I,J)
      + DTSQ23 * FCY(I,J) - DTSQ6 * FOCY(I,J)
RCZ(I,J) = RCZ(I,J) + DT * VCZ(I,J)
      + DTSQ23 * FCZ(I,J) - DTSQ6 * FOCZ(I,J)
VCX(I,J) = VCX(I,J) + DT56 * FCX(I,J) - DT6 * FOCX(I,J)
VCY(I,J) = VCY(I,J) + DT56 * FCY(I,J) - DT6 * FOCY(I,J)
VCZ(I,J) = VCZ(I,J) + DT56 * FCZ(I,J) - DT6 * FOCZ(I,J)

120    CONTINUE

110    CONTINUE

```

The velocities of all particles are first incremented to mid step values. After incrementing velocities, the forces calculated in the previous step are assigned to old forces arrays FOX,FOY,FOZ for solvent and FOCX,FOCY,FOCZ for polymer particles. Then the FORCEP routine is called to calculate forces. The MOVEB routine shown below increments the velocities using the forces (accelerations) obtained in the present step

```

      DT3 = DT / 3.0

      DO 200 I = 1, NS

      VX(I) = VX(I) + DT3 * FX(I)
      VY(I) = VY(I) + DT3 * FY(I)
      VZ(I) = VZ(I) + DT3 * FZ(I)

200    CONTINUE

      DO 210 I = 1, NC

      DO 220 J = 1, NA

      VCX(I,J) = VCX(I,J) + DT3 * FCX(I,J)
      VCY(I,J) = VCY(I,J) + DT3 * FCY(I,J)
      VCZ(I,J) = VCZ(I,J) + DT3 * FCZ(I,J)

220    CONTINUE

210    CONTINUE

```

3.2.7 Calculation of Forces

The forces on both solvent and polymer units are calculated by the FORCEP subroutine. The routine gets the positions solvent and polymer units from arrays RX, RY, RZ and RCX, RCY, RCZ in COMMON BLOCK 1 and COMMON BLOCK 2, respectively and assigns the calculated forces to arrays FX, FY, FZ into COMMON BLOCK 1 for solvent units and FCX, FCY, FCZ in COMMON BLOCK 2 for polymer units. In addition to forces the routine calculates the virial total potential and shifted potential energies described in Section 2.1.2.

The FORCEP routine consists of four main loops: The first loop calculates the forces on solvent atoms arising from solvent-solvent interactions due to Lennard-Jones potentials.

The distance between two particles are calculated by considering minimum image correlations. [Allen 1986 pp].

```

RXIJ = RXIJ - ANINT ( RXIJ * BOXINV ) * BOX
RYIJ = RYIJ - ANINT ( RYIJ * BOXINV ) * BOX
RZIJ = RZIJ - ANINT ( RZIJ * BOXINV ) * BOX

```

where the function ANINT returns the integer which is nearest to the argument.

The second loop computes forces on solvent molecules caused by polymer units. The third loop considers polymer - polymer interactions due to Lennard-Jones potentials. In the last loop the harmonic potentials of eq (2.1.9), associated with the connectivity of the polymer chain are considered.

3.2.8 Calculation of Static and Dynamic Properties

The kinetic energy, the square of the end-to-end distance, the radius of gyration and the center of mass velocities are calculated at specified intervals. At the end of iterations the averages and standard deviations are calculated and printed on the output file. The diffusion coefficient is calculated from the autocorrelation function of center of mass velocities. The polymer-polymer, solvent-solvent and polymer-solvent pair correlation functions are normalized and printed in tabular form.

a. Calculation of Diffusion Coefficients

The autocorrelation function of polymer center of mass velocities is computed as shown below.

```

DUFCOF = 0.0
DO 3200 I = 0,TCOR
ITMAX = NTCOR - I
ATAO = 0.
DO 3199 J = 1,ITMAX
DO 3198 JI = 1,NC

```

```

      >
      ATAO = ATAO + (ATX(J,JI)*ATX(J+I,JI)
:           + ATY(J,JI)*ATY(J+I,JI)
:           + ATZ(J,JI)*ATZ(J+I,JI))
:           /SQRT((ATX(J,JI)**2+ATY(J,JI)**2
:           + ATZ(J,JI)**2) *
:           (ATX(J+I,JI)**2+ATY(J+I,JI)**2
:           + ATZ(J+I,JI)**2) )
3198 CONTINUE
3199 CONTINUE
      ACF(I) = ATAO / REAL(ITMAX*NC)
      IF ((I .EQ. 0).OR.(I .EQ. TCOR))THEN
        DUFCOF = DUFCOF +ACF(I)*0.5
      ELSE
        DUFCOF = DUFCOF +ACF(I)
      ENDIF
3200 CONTINUE
      DUFCOF = DUFCOF*(ITCOR-1)/3.

```

Here DUFCOF is diffusion coefficient, NTCOR is size of the center of mass velocity arrays. ATX(I,J), ATY(I,J), ATZ(I,J) are x,y,z components of center of mass velocities of sample i and chain j. ACF is the autocorrelation function for the velocity of the center of mass. The diffusion coefficient is calculated by numerically integrating the autocorrelation function as described in section 2.2.3.

b. Calculation of Pair Distribution Functions

The solvent-solvent, solvent-polymer and polymer-polymer pair distribution functions, referred to as HISS, HISP and HIPP respectively, are computed by using the following code (shown for HISS)

```
RIJSQ = RXIJ ** 2 + RYIJ ** 2 + RZIJ ** 2
```

```
RIJ = SQRT(RIJSQ)
```

```
IBIN = INT(RIJ/DELR) +1
```

```

IF (IBIN .LE. MAXBIN ) THEN
  HISS(IBIN) = HISS(IBIN) + 2.
ENDIF
ENDIF

```

Here RXIJ,RYIJ,RZIJ are the distance between particle i and j .The ith element of the array HISS is incremented by two if the minimum image distance is between r and $r+\Delta r$ for the solvent atom pair i and j.

At the end of simulation the arrays HISS HIPS and HIPPP are normalized by the following code.

```

CONST = 4. * PI * DENS/3.
DO 3300 I=1,MAXBIN

  RLOW = REAL(I- 1)*DELR
  RUP = RLOW + DELR
  RIDEAL = CONST *(RUP**3 - RLOW**3)
  HHIPP(I) = HIPPP(I)/REAL(RIDEAL*NORM*N)
  HHIPS(I) = HIPS(I)/REAL(RIDEAL*NORM*N)
  HHISS(I) = HISS(I)/REAL(RIDEAL*NORM*N)
3300 * CONTINUE

```

CHAPTER IV

RESULTS AND DISCUSSION

4.1. Results

In this study three runs were taken with different polymer-polymer, polymer-solvent and solvent-solvent energy parameter of Lennard-Jones potentials. All other variables were kept constant. The values of energy parameters for solvent-solvent interaction ϵ_{ss} , polymer-solvent interaction ϵ_{ps} and polymer-polymer interaction ϵ_{pp} are given in Table 4.1. The values of the remaining fixed input variables are listed in Table 4.2. The computer output of run B is given in Appendix C, for illustrative purposes.

TABLE 4.1

Run	ϵ_{ss}	ϵ_{ps}	ϵ_{pp}	$\langle r^2 \rangle / \sigma^2$	$\langle s^2 \rangle / \sigma^2$	$\langle r^2 \rangle / \langle s^2 \rangle$	$D \times 10^{-2}$
A	0.25	0.25	0.25	15.557	2.212	7.03	16.97
B	0.40	0.10	0.40	6.579	1.336	4.93	14.71
C	0.10	0.50	0.10	15.990	2.301	6.94	9.49

TABLE 4.2**Values of Fixed Parameters in terms of Reduced and Real Units**

	Reduced	Real
Mass of an atom(unit)	1	2×10^{-23} g
Diameter of particles	1	2.7 Å
Length of cubic box	7.4σ	19.98 Å
Total number of steps	5000	5000
Time step size	$0.01\tau^*$	1.8×10^{-14} sec
Density	0.8	0.8 g/cm ³
Temperature	0.75	378 K
k_0 in eq(2.1.10)	$20\epsilon/\sigma^2$	68 kcal/Å ²
r_0 in eq(2.1.10)	1.95σ	5.265 Å

4.1.1. Chain Configuration

The mean-square end-to-end distance $\langle r^2 \rangle$ and mean-square radius of gyration $\langle s^2 \rangle$ resulting from the three runs are shown in the columns 4 and 5 of Table 4.1. The ratio $\langle r^2 \rangle / \langle s^2 \rangle$ is also

calculated and listed in the sixth column of Table 4.1. A large value for this ratio corresponds to the case of a chain with a relatively expanded configuration. The chain is more contracted in run B relative to other runs due to higher polymer-polymer attraction. In Table 4.3 the average kinetic energies, potential energies, and temperature corresponding to each run are listed, with their standard deviations.

TABLE 4.3

(Potential and Kinetic Energies, and Temperature in Reduced Units)

Run	Kinetic Energy	Potential E.	Temperature
A	0.893± 0.021	-0.428 ±0.000	0.595 ±0.014
B	0.995± 0.026	-0.226 ±0.055	0.663 ± 0.055
C	0.856 ±0.020	-0.567 ±0.022	0.571 ± 0.013

4.1.2 The Pair Distribution Functions

The pair distribution functions $g_{ss}(r)$, $g_{ps}(r)$ and $g_{pp}(r)$ are shown in Figures 4.1, 4.2 and 4.3 for the three runs A, B and C. The shapes of $g_{ss}(r)$ and $g_{ps}(r)$ curves are very similar. The $g_{ss}(r)$ curves have higher maxima because of the higher concentration of solvent molecules. The shape of $g_{pp}(r)$ curves are quite different from others due to the low concentration of polymer units. These results are quite similar to those obtained by Luque et al.

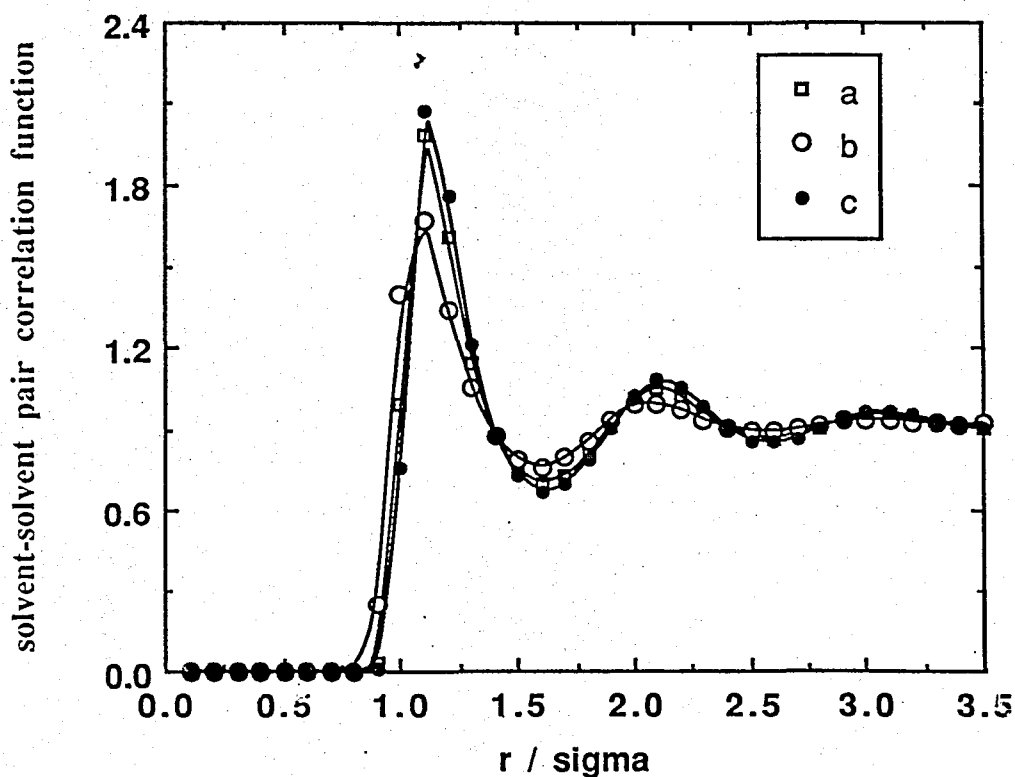


Figure 4.1. Solvent-solvent pair distribution function

[Luque. et al., 1989] at the same reduced density and chain lengths. When the polymer-polymer interaction parameter increases so as to lead to higher attractive potential, the $g_{pp}(r)$ curve starts from a very high value relative to other cases, as illustrated in run B.

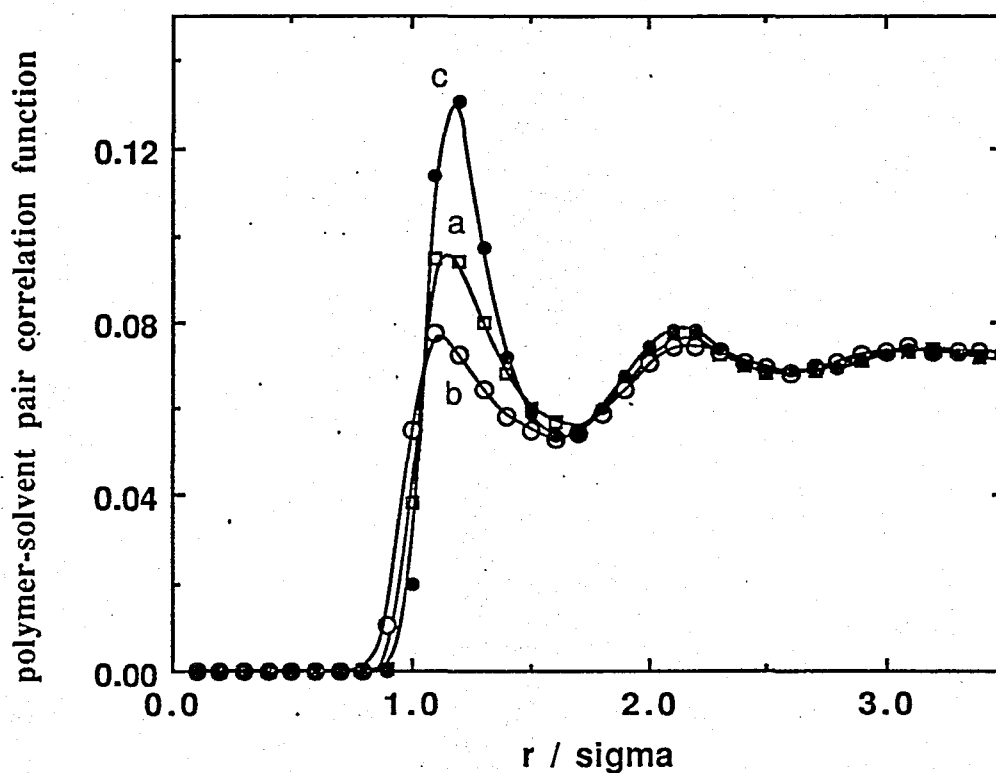


Figure 4.2. Polymer-solvent pair distribution function

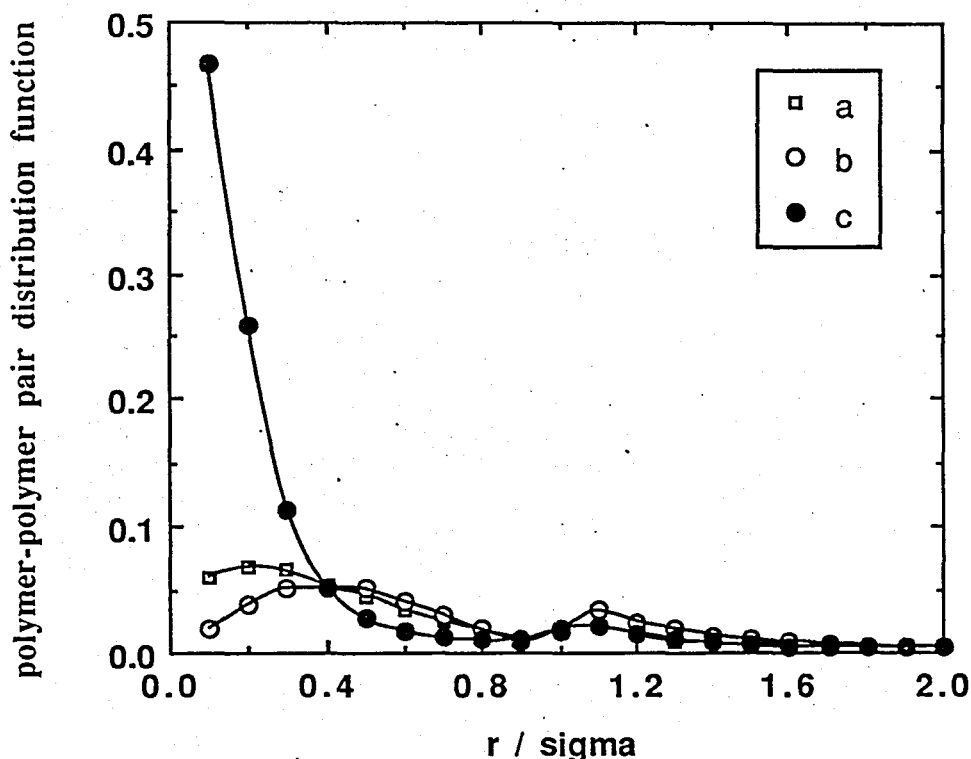


Figure 4.3. Polymer-polymer pair distribution function

4.1.3 Autocorrelation Functions and Diffusion Coefficients

The time dependence of the autocorrelation function of polymer center of mass velocities is shown in Figure 4.4, for the three runs. The corresponding diffusion coefficients are presented in the sixth column of Table 4.1. It is observed that the diffusion of the polymer in the solution with the highest polymer-solvent interaction is the slowest. This is apparent both from the fast decay of the corresponding velocity correlation function (run C) and from the low value of the resulting diffusion coefficient.

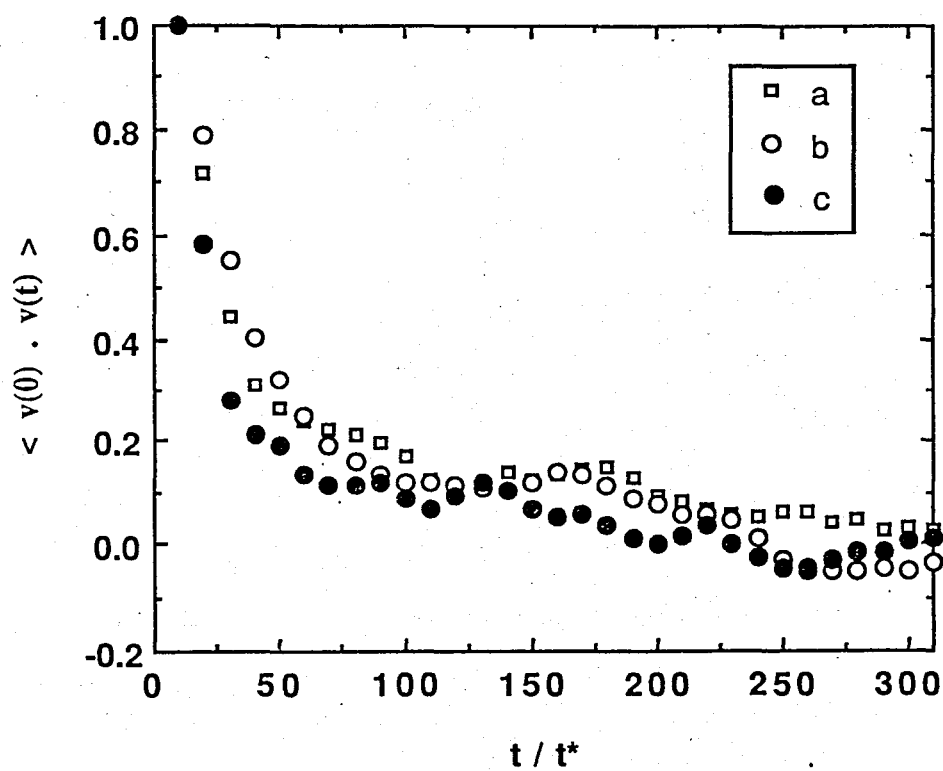


Figure 4.4. Center of mass velocity autocorrelation function

4.1. Remarks and Further Developments

The polymer-solvent system studied in this thesis may be investigated further by varying other parameters than the ϵ values such as chain length, the temperature, the density and the sizes of solvent and/or polymer units. Concentrated solutions containing more than one chain in a simulation box may also be examined within the same mathematical formalism. For a system containing more than one polymer chain, it is sufficient to connect a large number of units to generate polymer chains of desired length. Thus, the influence of increasing concentration of polymer chains may be readily examined.

The simulation was performed for a freely jointed chain, in which no intramolecular potential other than the bond stretching was included. Intramolecular potentials associated with bond angle bending and bond rotations/torsional motions may be included in the model by proper modification of the force subroutine. This approach will preserve the approximately tetrahedral structure and the realistic rotational isomeric states. Thus, a more realistic description of the polymer chain may be represented by this refinement.

APPENDIX A

THE SIMULATION PROGRAM

PROGRAM MDSPS

```

COMMON / BLOCK1 / RX, RY, RZ, VX, VY, VZ,
:      FOX, FOY, FOZ, FX, FY, FZ
COMMON / BLOCK2 / RCX, RCY, RCZ, VCX, VCY, VCZ,
:      FOCX, FOCY, FOCZ, FCX, FCY, FCZ

COMMON / BLOCK3 / N, NS, NC, NA
COMMON / BLOCK4 / BOX, BOXINV, RCUT, RCUTSQ, VRCSS, WRCSS
COMMON / BLOCK6 / SIGPPSQ, SIGSSSQ, SIGPSSQ, EPSPP24,
:      EPSSS24, EPSPS24, WRCPS, WRCPP, VRCPS,
:      VRCPP, EPSPS4, EPSPP4, ROHSQ, RKOH,
:      HIPP, HIPS, HISS, TECL, FICL,
:      MAXBIN, MAXTET, MAXFIF, DELR, DELT, DELF,
:      RKT, RKF, TET0
:      , IAVA
COMMON / BLOCK5 / DT, DT2, DT56, DT6, DTSQ2, DTSQ23, DTSQ6.
COMMON / BLOCK7 / GAMA, A0, A1, A2, A3, A4, A5

```

```

INTEGER    N, NS, NC, NA, IACH, NEXTA
INTEGER    I, J, JA
INTEGER    STEP, NSTEP
INTEGER    NSTEPA(2), NSTEPB(2)
INTEGER    IAVA, ITCOR, IPRINT, TCOR, NTCOR
INTEGER    MAXBIN, MAXTET, MAXFIF

REAL       RX(800), RY(800), RZ(800)
REAL       VX(800), VY(800), VZ(800)
REAL       FX(800), FY(800), FZ(800)
REAL       FOX(800), FOY(800), FOZ(800)
REAL       RCX(1,30), RCY(1,30), RCZ(1,30)
REAL       VCX(1,30), VCY(1,30), VCZ(1,30)
REAL       FCX(1,30), FCY(1,30), FCZ(1,30)
REAL       FOCX(1,30), FOCY(1,30), FOCZ(1,30)
REAL       ATX(300,3), ATY(300,3), ATZ(300,3), ACF(0:300)
REAL       HIPP(50), HIPS(50), HISS(50), TECL(20), FICL(20)
REAL       HHIPP(50), HHIPS(50), HHISS(50), TTECL(20), FFICL(20)
REAL       DIRX(2,4), DIRY(2,4), DIRZ(2,4)
REAL       SIGSS, SIGPS, SIGPP, SIGSSSQ, SIGPSSQ, SIGPPSQ
REAL       EPSSS, EPSPS, EPSPP

```

```

REAL      EPSSS4, EPSPS4, EPSPP4
REAL      EPSSS24, EPSPS24, EPSPP24
REAL      RKOH, ROH, ROHSQ
REAL      SCR2, SCR6, SCR12
REAL      VCRSS, WCRSS, VCRPS, WCRPS, VCRPP, WCRPP

```

```

REAL      ACV, ACK, ACE, ACEC, ACP, ACT
REAL      AVV, AVK, AVE, AVEC, AVP, AVT
REAL      ACVSQ, ACKSQ, ACESQ, ACECSQ, ACPSQ, ACTSQ
REAL      BCVSQ, BCKSQ, BCESQ, BCECSQ, BCPSQ, BCTSQ
REAL      RMX, RMY, RMZ, RCSQ
REAL      FLV, FLK, FLE, FLEC, FLP, FLT
REAL      DT, DENS, TEMPER, RCUT, BOX, PRES, NORM, TEMP
REAL      K, V, VC, W, E, EC, FREE, FREEN, H, VOL
REAL      KN, VN, EN, ECN, HN, NORMN
REAL      SR3, SR9, VLRC, WLRC, PI

```

```

PARAMETER (PI = 3.141592654)

```

```

PARAMETER (FREE = 3.0)

```

```

C

```

```

*****

```

```

**

```

```

C ** MAIN PROGRAM BEGINS **

```

```

C

```

```

*****

```

```

**

```

```

      open (unit=1,file ='datasim.dat',STATUS = 'OLD')

```

```

      read(1,*) NTPR, N,NC,NA,BOX,RCUT,NSTEPS(1),NSTEPS(2),

```

```

: IAVA,ITCOR,TEMPER,

```

```

: SIGPP,SIGPS,SIGSS,EPSP,EPSPS,EPSSS,IDUM ,DT ,IPRINT

```

```

      READ(1,*) DELR,DELTAT,MAXTET,MAXFIF,MAXBIN,

```

```

: TCOR,RKT,RKF,TETO

```

```

c NTPR :mode of run

```

```

c NC   :number of chains

```

```

c NA   :number of polymer units in each chain

```

```

c BOX  :box legth

```

```

c RCUT :cut off distance

```

```

c IDUM :aseed to generate random variables

```

```

c DT   :time step

```

```

c NSTEPS(1) : number of steps before equilibration

```

```

c NSTEPS(2) : number of steps after equilibration

```

```

c IAVA :number of steps such that averages are taken in between

```

```

c ITCOR : number of steps such that center of mass velocities

```

```

c are calculated in between

```

```

c SIGPP,SIGPS,SIGSS : sigma values for polymer-polymer,

```

```

c polymer-solvent and solvent-solvent

```

```

c EPSP,EPSPS,EPSSS : energy parameters of pol-pol ...

```

```

c MAXBIN :size of the pair distribution function arrays

```

```

c DELR :delta r interval in computing pair distribution functions

```

```

c NTCOR :number of points that the autocorrelation function is

```

```

c computed

```

```
Open (unit=2,file ='dene211.out',status='new')
```

```
NSTEPB(1) = 1
```

```
NSTEPB(2) = 1
```

```
c      N=TOTAL NUMBER OF ATOMS IN THE BOX
```

```
VOL = BOX ** 3
```

```
DENS = REAL ( N ) / VOL
```

```
NS = N
```

```
C      ** CALCULATION OF PARAMETERS **
```

```
C      ** CALCULATION OF TIME PARAMETERS **P
```

```
c      DT is time step
```

```
DT2 = DT / 2.0
```

```
DT56 = 5.0 * DT / 6.0
```

```
DTSQ23 = 2.0 * DT * DT / 3.0
```

```
DTSQ6 = DT * DT / 6.0
```

```
DT6 = DT / 6.0
```

```
DTSQ2 = DT * DT / 2.0
```

```
c      rcut is the critical separation between two atoms such that
```

```
c      beyond rcut no interaction takes place.
```

```
RCUTSQ = RCUT ** 2
```

```
BOXINV = 1.0 / BOX
```

```
c      calculations related to sigms
```

```
SIGPPSQ = SIGPP ** 2
```

```
SIGSSSQ = SIGSS ** 2
```

```
SIGPSSQ = SIGPS ** 2
```

```
c      EPSSS,EPSPS,EPSP are energy parameters of solvent-solvent,
```

```
c      solvent-polymer and polymer - polymer respectively
```

```
EPSSS4 = EPSSS * 4.0
```

```
EPSP4 = EPSP * 4.0
```

```
EPSPS4 = EPSPS * 4.0
```

```
EPSSS24 = EPSSS * 24.0
```

```
EPSP24 = EPSP * 24.0
```

```
EPSPS24 = EPSPS * 24.0
```

```
c      RKOH is the proportionality constant in the harmonic potential
```

```
c      equation
```

```
RKOH = 20. * EPSP / SIGPPSQ
```

```
ROH = 1.95 * SIGPP
```

```
ROHSQ = ROH ** 2
```

```
c      calculations of shifted potential terms
```

```
c      WRCSS,WRCPS,WRCPP,VRCSS,VRCPS,VRCPP are used in the
```

```
c      calculation of shifted force potentials
```

```
SRC2 = 1.0 / RCUTSQ
```

```
SRC6 = SRC2 * SRC2 * SRC2
```

```
SRC12 = SRC6 * SRC6
```

```
VRCSS = SRC12 - SRC6
```

```
WRCSS = VRCSS + SRC12
```

```
WRCSS = WRCSS / RCUT
```

```

SRC2 = SIGPSSQ / RCUTSQ
SRC6 = SRC2 * SRC2 * SRC2
SRC12 = SRC6 * SRC6
VRCPS = SRC12 - SRC6
WRCPS = VRCPS + SRC12
WRCPS = WRCPS / RCUT

```

```

SRC2 = SIGPPSQ / RCUTSQ
SRC6 = SRC2 * SRC2 * SRC2
SRC12 = SRC6 * SRC6
VRCPP = SRC12 - SRC6
WRCPP = VRCPP + SRC12
WRCPP = WRCPP / RCUT

```

```

C  ** CALCULATE LONG-RANGE CORRECTIONS **
C  ** NB: SPECIFIC TO LENNARD-JONES  **

```

```

      SR3 = ( 1.0 / RCUT ) ** 3.0
      SR9 = SR3 ** 3.0
      VLRC = ( 8.0 / 9.0 ) * PI * DENS * REAL ( N ) *
:      ( SR9 - 3.0 * SR3 )
      WLRC = ( 16.0 / 9.0 ) * PI * DENS * REAL ( N ) *
:      ( 2.0 * SR9 - 3.0 * SR3 )

```

```

      ITPR = 0
      IF ( NTPR .EQ. 3 ) ITPR = 1
      IF ( NTPR .GT. 1 ) GOTO 4000

```

```

C  ** INITIAL POSITIONS **
CALL BCC(N,BOX)

```

```

C  ** INITIAL VELOCITIES
CALL COMVEL(N,TEMPER ,IDUM)

```

```

C  ** CENERATION OF POLIMER CHAINS **
      NC1 = INT((N*2 / 3)**(1.0/3.0))
      CELL2 = BOX / REAL ( NC1*2 )
      DELTAS = 0.001

```

```

c  DIRX,DIRY,DIRZ are the coordinates ofdirectional vectors
      DO 1501 I = 1,2
        DO 1502 J = 1,4
          READ(1,*)DIRX(I,J),DIRY(I,J),DIRZ(I,J)
          DIRX(I,J) =DIRX(I,J) * CELL2
          DIRY(I,J) =DIRY(I,J) * CELL2
          DIRZ(I,J) =DIRZ(I,J) * CELL2
1502 CONTINUE
1501 CONTINUE
      CLOSE(1)
      DO 1500 I = 1 , NC

```

```

      CALL RAN2(IDUM,RANVV)
1518 IACH = INT( RANVV*NS )

```

```

      IF (IACH.EQ.0) IACH = 1
      CALL COUNT (IACH,I,1)
      RCIX = RX(IACH)
      RCIY = RY(IACH)
      RCIZ = RZ(IACH)

c   the routine COUNT includes the selected atom as the new member
c   of the polymer chain
c   the DELETE routine deletes the selected polymer atom among
c   solvent atoms
      CALL DELETE (NS, IACH)

      NS = NS - 1
      ID1 = 1
      DO 1510 J = 2, NA
        ICOUNT = 1
        CALL RAN2(IDUM,RANVV)
        ID = INT(4*RANVV)+1
1519      IFOUND = 0
        RCIJX = RCIX + DIRX(ID1,ID)
        RCIJY = RCIY + DIRY(ID1,ID)
        RCIJZ = RCIZ + DIRZ(ID1,ID)

        DO 1520 JA = 1, NS
          IF( ( RX(JA).GT. RCIJX -DELTAS).AND.
            : (RX(JA).LT. RCIJX +DELTAS).AND.
            : (RY(JA).GT. RCIJY -DELTAS).AND.
L      (RY(JA).LT. RCIJY +DELTAS).AND.
            : (RZ(JA).GT. RCIJZ -DELTAS).AND.
            : (RZ(JA).LT. RCIJZ +DELTAS)) THEN
            IFOUND = 1
            NEXTA = JA
            ENDIF
          IF (IFOUND.EQ.1) GOTO 1521
1520      CONTINUE
          ID = ID+1
          ICOUNT = ICOUNT + 1
          IF(ID.EQ.5) ID = 1
          IF (ICOUNT.LE.4) GOTO 1519
          GOTO 1529

1521      CALL COUNT (NEXTA,I,J)

      RCIX = RX(NEXTA)
      RCIY = RY(NEXTA)
      RCIZ = RZ(NEXTA)
      CALL DELETE (NS, NEXTA)
      NS = NS - 1

      ID1 = ID1 + 1
      IF (ID1.EQ.3) ID1 = 1
      GOTO 1510

```

```

1529   NS = N - J + 1
      DO 1530 JI = 1, J - 1
        NS = NS + 1
        RX(NS) = RCX(I, JI)
        RY(NS) = RCY(I, JI)
        RZ(NS) = RCZ(I, JI)
        VX(NS) = VCX(I, JI)
        VY(NS) = VCY(I, JI)
        VZ(NS) = VCZ(I, JI)

```

```

1530       continue
          GOTO 1518

```

```

1510   CONTINUE

```

```

1500   CONTINUE

```

```

      WRITE(2, 10001)
10001  FORMAT(//1X, 'TIMESTEP   ..ENERGY..',
:        ' CUTENERGY. ..KINETIC. ..POTENT..',
:        ' ..TEMPER.. ..PRESSURE. '/')

```

```

C    ** READS DATA FROM TYPE FILE **

```

```

4000   IF (NTPR .GT. 1) THEN

```

```

      Open (UNIT=3, file ='type1.dat', STATUS = 'OLD')
      NS = N - NA * NC
      DO 4001 I = 1, NS
4001    READ(3, *) RX(I), RY(I), RZ(I)
      do 4002 i = 1, NC
        do 4003 j = 1, NA
          READ(3, *) RCX(i, j), RCY(i, j), RCZ(i, j)
4003        continue
4002      continue
      DO 4027 I = 1, NS
4027    READ(3, *) VX(I), VY(I), VZ(I)
      do 4028 i = 1, NC
        do 4029 j = 1, NA
          READ(3, *) VCX(i, j), VCY(i, j), VCZ(i, j)
4029        continue
4028      continue

      DO 4021 I = 1, NS
4021    READ(3, *) FX(I), FY(I), FZ(I)
      do 4022 i = 1, NC
        do 4023 j = 1, NA
          READ(3, *) FCX(i, j), FCY(i, j), FCZ(i, j)
4023        continue
4022      continue
      DO 4024 I = 1, NS

```

```

4024     READ(3,*) FOX(I),FOY(I),FOZ(I)
      do 4025 I = 1,NC
        do 4026 J = 1,NA
          READ(3,*)FOCX(I,J),FOCY(I,J),FOCZ(I,J)
4026     continue
4025     continue

      READ(3,*) ACV,ACK,ACE,ACEC,ACT,ACP
      READ(3,*) ACVSQ,ACKSQ,ACESQ,ACECSQ,ACTSQ,ACPSQ

      READ(3,*) ACRCSQ,ACRGSQ,ACCOS,NTCOR

      do 4031 i = 1,NTCOR
        do 4032 j = 1,NC
          READ(3,*)ATX(i,j),ATY(i,j),ATZ(i,j)
4032     continue
4031     continue
      DO 4041 I = 1 , MAXBIN
4041     READ(3,*) HISS(I),HIPS(I),HIPP(I)

      READ(3,*)NSTEPB(ITPR+1)
      NSTEPB (ITPR + 1)= NSTEPB(ITPR+1) +1

      CLOSE(3)
      GOTO 1999
    ENDIF

C    ** CALCULATION OF INITIAL FORCES **
      STEP = -1
      CALL FORCEP(V,VC,W,STEP)V
      DO 6011 I = 1 , NS
C    W,V ,VC are virial,total potential andtotal shifted force
c    potentials
c      FOX,FOY and FOZ refer to the old forces at the previous step

      FOX(I) = FX(I)
      FOY(I) = FY(I)
      FOZ(I) = FZ(I)

6011   CONTINUE
      DO 6012 I = 1,NC
        DO 6013 J = 1,NA
          FOCX(I,J) = FCX(I,J)
          FOCY(I,J) = FCY(I,J)
          FOCZ(I,J) = FCZ(I,J)
6013   CONTINUE
6012   CONTINUE

      DO 6010 I = 1 , N

      RX(I) = RX(I) + DT * VX(I) + DTSQ2 * FX(I)
      RY(I) = RY(I) + DT * VY(I) + DTSQ2 * FY(I)
      RZ(I) = RZ(I) + DT * VZ(I) + DTSQ2 * FZ(I)

```

```

VX(I) = VX(I) + DT2 * FX(I)
VY(I) = VY(I) + DT2 * FY(I)
VZ(I) = VZ(I) + DT2 * FZ(I)

```

```
6010  CONTINUE
```

```
DO 6021 I = 1,NC
```

```
DO 6022 J = 1,NA
```

```
RCX(I,J) = RCX(I,J) + DT * VCX(I,J) + DTSQ2 * FCX(I,J)
```

```
RCY(I,J) = RCY(I,J) + DT * VCY(I,J) + DTSQ2 * FCY(I,J)
```

```
RCZ(I,J) = RCZ(I,J) + DT * VCZ(I,J) + DTSQ2 * FCZ(I,J)
```

```
VCX(I,J) = VCX(I,J) + DT2 * FCX(I,J)
```

```
VCY(I,J) = VCY(I,J) + DT2 * FCY(I,J)
```

```
VCZ(I,J) = VCZ(I,J) + DT2 * FCZ(I,J)
```

```
6022  CONTINUE
```

```
6021  CONTINUE
```

```
CALL FORCEP(V,VC,W,STEP)
```

```
DO 6132 I = 1 , NS
```

```
VX(I) = VX(I) + DT2 * FX(I)
```

```
VY(I) = VY(I) + DT2 * FY(I)
```

```
VZ(I) = VZ(I) + DT2 * FZ(I)
```

```
6132  CONTINUE
```

```
DO 6032 I = 1 , NC
```

```
DO 6033 J = 1 , NA
```

```
VCX(I,J) = VCX(I,J) + DT2 * FCX(I,J)
```

```
VCY(I,J) = VCY(I,J) + DT2 * FCY(I,J)
```

```
VCZ(I,J) = VCZ(I,J) + DT2 * FCZ(I,J)
```

```
6033  CONTINUE
```

```
6032  CONTINUE
```

```
C  ** ZERO ACCUMULATORS **
```

```
c  the variables beginning with AC is used to sum the values of
c  relevant quantities
```

```
c  ACV :potential energy,ACK : kinetic energy,ACE :total energy
```

```
c  ACEC : shifted potential +kinetic energy,ACT :temperature
```

```
c  ACP : pressure
```

```
c  NTCOR calculates number of times the center of mas velocity
```

```
c  is calculated
```

```
1998  ACV = 0.0
```

```

    ACK = 0.0
    ACE = 0.0
    ACEC = 0.0
    ACP = 0.0
    ACT = 0.0
    NTCOR = 0
c variables ending with SQ represent squares of accumulation variables
    ACVSQ = 0.0
    ACKSQ = 0.0
    ACESQ = 0.0
    ACECSQ = 0.0
    ACPSQ = 0.0
    ACTSQ = 0.0
    ACRCSQ = 0.0
    ACRGSQ = 0.0
c variables beginning with FL are used in calculating sum of
c squares of relevant variables
    FLV = 0.0
    FLK = 0.0
    FLE = 0.0
    FLEC = 0.0
    FLP = 0.0
    FLT = 0.0

c arrays HISS, HIPS, HIPP are used in the calculation of pair distribution
c functions to sum pairs of atoms
    DO 2041 I = 1, MAXBIN
        HISS(I) = 0.0
        HIPS(I) = 0.0
2041     HIPP(I) = 0.0

```

```

1999     ITPR = ITPR + 1

```

```

C
*****
**
C ** MAIN LOOP AFTER EQUILIBRATION BEGINS **
C
*****
**
    DO 2000 STEP = NSTEPB(ITPR), NSTEPA(ITPR)

C     ** IMPLEMENT ALGORITHM **
        WRITE(*,*) STEP
        CALL MOVEAP

c     assignment of old forces to forces

```

```

DO 2010 I = 1 , NS
  FOX(I) = FX(I)
  FOY(I) = FY(I)
  FOZ(I) = FZ(I)
2010  CONTINUE
DO 2020 I = 1 , NC
  DO 2030 J = 1 , NA
    FOCX(I,J) = FCX(I,J)
    FOCY(I,J) = FCY(I,J)
    FOCZ(I,J) = FCZ(I,J)
2030  CONTINUE
2020  CONTINUE

CALL FORCEP ( V, VC, W ,STEP)

CALL MOVEBP ( DT )

IF( MOD(STEP , IAVA ) .EQ. 0 ) THEN
c  calculation of kinetic energy
  CALL KINETP ( K )
c  calculations of avarage quantities
  V = V + VLRC
  W = W + WLRC
  E = K + V
  EC = K + VC
  EN = E / REAL ( N )
  VN = V / REAL ( N )
  KN = K / REAL ( N )
  ECN = EC / REAL ( N )
  TEMP = KN * 2.0 / FREE
  PRES = DENS * TEMP + W / VOL
C  ** INCREMENT ACCUMULATORS **

  ACE = ACE + EN
  ACEC = ACEC + ECN
  ACK = ACK + KN
  ACV = ACV + VN
  ACP = ACP + PRES

  ACESQ = ACESQ + EN ** 2
  ACECSQ = ACECSQ + ECN ** 2
  ACKSQ = ACKSQ + KN ** 2
  ACVSQ = ACVSQ + VN ** 2
  ACPSQ = ACPSQ + PRES ** 2
c  calculations of mean squared end to end distance and radius of gyra
c  gyrations
  ACRGST = 0.0
  ACGYRT = 0.0
DO 2500 I = 1 , NC

  RXIJ = RCX(I,NA) - RCX(I,1)
  RYIJ = RCY(I,NA) - RCY(I,1)
  RZIJ = RCZ(I,NA) - RCZ(I,1)

```

```

      RCSQ = RXIJ **2 + RYIJ **2 + RZIJ **2
      ACRCSQ = ACRCSQ + RCSQ
      ACR CST = ACR CST + RCSQ
      RGYR = 0.0
      RMX = 0.0
      RMY = 0.0
      RMZ = 0.0
      DO 2510 J = 1 , NA
      RMX = RMX + RCX(I,J)
      RMY = RMY + RCY(I,J)
      RMZ = RMZ + RCZ(I,J)
2510    CONTINUE
      RMX = RMX / REAL(NA)
      RMY = RMY / REAL(NA)
      RMZ = RMZ / REAL(NA)
      DO 2511 J = 1,NA
      RXIJ = RCX(I,J) - RMX
      RYIJ = RCY(I,J) - RMY
      RZIJ = RCZ(I,J) - RMZ
      RCSQ = RXIJ **2 + RYIJ **2 + RZIJ **2
      RGYR = RGYR + RCSQ
2511    CONTINUE
      RGYR = RGYR / REAL(NA)
      ACRGSQ = ACRGSQ + RGYR
      ACGYRT = ACGYRT + RGYR
C      WRITE(*,*) RGYR
2500    CONTINUE
      ACR CST = ACR CST / REAL(NA)
      ACGYRT = ACGYRT / REAL(NA)

      endif
c calculation of center of mass velocities of polymer
      IF ( MOD(STEP,ITCOR) .EQ. 0 ) THEN
        NTCOR = NTCOR + 1

        DO 2700 I = 1 ,NC
          CMVX = 0.0
          CMVY = 0.0
          CMVZ = 0.0
          DO 2710 J = 1 , NA
            CMVX = CMVX + VCX(I,J)
            CMVY = CMVY + VCY(I,J)
            CMVZ = CMVZ + VCZ(I,J)
2710          CONTINUE
            ATX(NTCOR,I) = CMVX / REAL(NA)
            ATY(NTCOR,I) = CMVY / REAL(NA)
            ATZ(NTCOR,I) = CMVZ / REAL(NA)
2700          CONTINUE
        ENDIF
C      ** OPTIONALLY PRINT INFORMATION **

      IF ( MOD ( STEP, IPRINT ) .EQ. 0 ) THEN

```

```

WRITE(2, '(1X, I5, 8(2X, F7.3))')
:   STEP, EN, ECN, KN, VN, TEMP, PRES, ACRCST, ACGYRT

ENDIF

2000 CONTINUE

C
*****
* *
C ** MAIN LOOP ENDS **
C
*****
* *

c   WRITE(*, '(1X, "**** END OF DYNAMICS **** "/)')

C ** WRITE OUT FINAL AVERAGES **
c   variables beginning with AV represents the average of the
c   relevant variables
c   NORM is the variable representing number of times
c   averages are taken
      NORM = REAL ( NSTEPA(ITPR) / IAVA )
      NORMN = REAL(NORM * NC)
      AVE = ACE / NORM
      AVEC = ACEC / NORM
      AVK = ACK / NORM
      AVV = ACV / NORM
      AVP = ACP / NORM
      AVRCSQ = ACRCSQ / NORMN
      AVRGYR = ACRGSQ / NORMN
c   variables beginning with B represent variances
      BCESQ = ( ACESQ / NORM ) - AVE ** 2
      BCECSQ = ( ACECSQ / NORM ) - AVEC ** 2
      BCKSQ = ( ACKSQ / NORM ) - AVK ** 2
      BCVSQ = ( ACVSQ / NORM ) - AVV ** 2
      BCPSQ = ( ACPSQ / NORM ) - AVP ** 2

      IF ( BCESQ .GT. 0.0 ) FLE = SQRT ( BCESQ )
      IF ( BCECSQ .GT. 0.0 ) FLEC = SQRT ( BCECSQ )
      IF ( BCKSQ .GT. 0.0 ) FLK = SQRT ( BCKSQ )
      IF ( BCVSQ .GT. 0.0 ) FLV = SQRT ( BCVSQ )
      IF ( BCPSQ .GT. 0.0 ) FLP = SQRT ( BCPSQ )

      AVT = AVK * 2.0 / FREE
      FLT = FLK * 2.0 / FREE

      WRITE(2, '( " AVERAGES", 8(2X, F7.3))')
:   AVE, AVEC, AVK, AVV, AVT, AVP, AVRCSQ, AVRGYR
      WRITE(2, '( " FLUCTS ", 6(2X, F7.3))')
:   FLE, FLEC, FLK, FLV, FLT, FLP
c   calculation of autocorrelation function from center of mass

```

```

c velocities
  WRITE(2,*) 'AUTOCORRELATION FUNCTION'
c DUFCONF is diffusion coefficient
  DUFCONF = 0.0
  DO 3200 I = 0,TCOR
    ITMAX = NTCOR - I
    ATAO = 0.
    DO 3199 J = 1,ITMAX
      DO 3198 JI = 1,NC

        ATAO = ATAO + (ATX(J,JI)*ATX(J+I,JI)
          :           + ATY(J,JI)*ATY(J+I,JI)
          :           + ATZ(J,JI)*ATZ(J+I,JI))
          :           /SQRT((ATX(J,JI)**2+ATY(J,JI)**2
          :           + ATZ(J,JI)**2) *
          :           (ATX(J+I,JI)**2+ATY(J+I,JI)**2
          :           + ATZ(J+I,JI)**2) )
3198      CONTINUE
3199    CONTINUE
c ACF is the array of autocorrelation function
    ACF(I) = ATAO / REAL(ITMAX*NC)
    IF ((I.EQ. 0).OR.(I.EQ. TCOR))THEN
      DUFCONF = DUFCONF + ACF(I)*0.5
    ELSE
      DUFCONF = DUFCONF + ACF(I)
    ENDIF
    WRITE(2, '(5X,F10.4)') ACF(I)
3200  CONTINUE
    DUFCONF = DUFCONF*(ITCOR)/3.
    WRITE(2,*) 'DIFFUSION COEFFICIENT = ', DUFCONF
    WRITE(2,*) 'PAIR CORRELATION FUNCTIONS'
    WRITE(2,*) '...SOLSOL... ...POLSOL... ...POLPOL...'
c calculation of pair distribution functions

    CONST = 4. * PI * DENS/3.
    DO 3300 I=1,MAXBIN
c the arrays HHISS,HHIPS,HHIPP are arrays of pair distribution
c functions of solvent-solvent,polymer-solvent
c and polymed-polymer
      RLOW = REAL(I- 1)*DELR
      RUP = RLOW + DELR
      RIDEAL = CONST *(RUP**3 - RLOW**3)
      HHIPP(I) = HIPP(I)/REAL(RIDEAL*NORM*N)
      HHIPS(I) = HIPS(I)/REAL(RIDEAL*NORM*N)
      HHISS(I) = HISS(I)/REAL(RIDEAL*NORM*N)
      WRITE(2, '(5X,3(2X,F10.5)') HHISS(I), HHIPS(I), HHIPP(I)
3300  CONTINUE

    IF ((NTPR.EQ. 0).AND.(ITPR.EQ. 1))GOTO 1998
    IF ((NTPR.EQ. 4).AND.(ITPR.EQ. 1))GOTO 1998

    CLOSE(2)

```

```

C      *****
C  ** WRITING TO TAPE FILE
C      *****
      Open (UNIT=2,file ='type1.dat',status='new')
      DO 5001 I = 1 , NS
5001      WRITE(2,*) RX(I),RY(I),RZ(I)
      do 5002 i =1,NC
          do 5003 j = 1,NA
              write(2,*)RCX(i,j),RCY(i,j),RCZ(i,j)
5003          continue
5002      continue
      DO 5027 I = 1 , NS
5027      WRITE(2,*) VX(I),VY(I),VZ(I)
      do 5028 i =1,NC
          do 5029 j = 1,NA
              write(2,*)VCX(i,j),VCY(i,j),VCZ(i,j)
5029          continue
5028      continue

      DO 5021 I = 1 , NS
5021      WRITE(2,*) FX(I),FY(I),FZ(I)
      do 5022 i =1,NC
          do 5023 j = 1,NA
              write(2,*)FCX(i,j),FCY(i,j),FCZ(i,j)
5023          continue
5022      continue
      DO 5024 I = 1 , NS
5024      WRITE(2,*) FOX(I),FOY(I),FOZ(I)
      do 5025 i =1,NC
          do 5026 j = 1,NA
              write(2,*)FOCX(i,j),FOCY(i,j),FOCZ(i,j)
5026          continue
5025      continue

      WRITE(2,*) ACV,ACK,ACE,ACEC,ACT,ACP
      WRITE(2,*) ACVSQ,ACKSQ,ACESQ,ACECSQ,ACTSQ,ACPSQ
      WRITE(2,*) ACRCSQ,ACRGSQ,ACCOS,NTCOR

      do 5031 i =1,NTCOR
          do 5032 j = 1,NC
              write(2,*)ATX(i,j),ATY(i,j),ATZ(i,j)
5032          continue
5031      continue
      DO 5041 I = 1 , MAXBIN
5041      WRITE(2,*) HISS(I),HIPS(I),HIPP(I)

      WRITE(2,*)NSTEPA(ITPR)

      CLOSE(2)
      PAUSE
      STOP

```

END

C ** SETS UP THE ALPHA BCC LATTICE FOR N LINEAR MOLECULES. **
 C ** SETS UP THE ALPHA BCC LATTICE FOR N LINEAR MOLECULES. **
 SUBROUTINE BCC(N,BOX)

COMMON / BLOCK1 / RX, RY, RZ, VX, VY, VZ,
 : FOX, FOY, FOZ, FX, FY, FZ

REAL RX(800), RY(800), RZ(800)
 REAL VX(800), VY(800), VZ(800)
 REAL FX(800), FY(800), FZ(800)
 REAL FOX(800), FOY(800), FOZ(800)

C ** CALCULATE THE SIDE OF THE UNIT CELL **

c NC is number of lattices in each side

NC = INT((N*2 / 3)**(1.0/3.0)/2)
 CELL = 1.0 / REAL (NC)
 CELL2 = 0.5 * CELL
 BOX2 = 0.5 * BOX

C ** BUILD THE UNIT CELL **

C ** SUBLATTICE A **

RX(1) = 0.0
 RY(1) = 0.0
 RZ(1) = 0.0

C ** SUBLATTICE B **

RX(2) = CELL2
 RY(2) = CELL2
 RZ(2) = 0.0

C ** SUBLATTICE C **

RX(3) = 0.0
 RY(3) = CELL2
 RZ(3) = CELL2

C ** SUBLATTICE D **

RX(4) = CELL2
 RY(4) = 0.0
 RZ(4) = CELL2

C ** CONSTRUCT THE LATTICE FROM THE UNIT CELL **

M = 0

DO 99 IZ = 1, NC

DO 98 IY = 1, NC

DO 97 IX = 1, NC

DO 96 IREF = 1, 4

RX(IREF+M) = RX(IREF) + CELL * REAL (IX - 1)
 RY(IREF+M) = RY(IREF) + CELL * REAL (IY - 1)
 RZ(IREF+M) = RZ(IREF) + CELL * REAL (IZ - 1)

96 CONTINUE

M = M + 4

97 CONTINUE

98 CONTINUE

99 CONTINUE

NC = 2*NC
 CELL = 1.0 / REAL (NC)
 CELL2 = 0.5 * CELL
 IREF = N - NC**3 + 1

RX(IREF) = CELL2
 RY(IREF) = CELL2
 RZ(IREF) = CELL2

C ** CONSTRUCT THE LATTICE FROM THE UNIT CELL **

M = 0

DO 199 IZ = 1, NC

DO 198 IY = 1, NC

DO 197 IX = 1, NC

RX(IREF+M) = RX(IREF) + CELL * REAL (IX - 1)
 RY(IREF+M) = RY(IREF) + CELL * REAL (IY - 1)
 RZ(IREF+M) = RZ(IREF) + CELL * REAL (IZ - 1)

M = M + 1

197 CONTINUE

198 CONTINUE

199 CONTINUE

C ** SHIFT CENTRE OF BOX TO THE ORIGIN **

DO 100 I = 1, N

RX(I) = RX(I) - 0.5

RY(I) = RY(I) - 0.5

RZ(I) = RZ(I) - 0.5

RX(I) = RX(I) * BOX

RY(I) = RY(I) * BOX

RZ(I) = RZ(I) * BOX

100 CONTINUE

RETURN

END

SUBROUTINE COMVEL(N,TEMP ,IDUM)

COMMON / BLOCK1 / RX, RY, RZ, VX, VY, VZ,
FOX, FOY, FOZ, FX, FY, FZ

C

* *

REAL RX(800), RY(800), RZ(800)

REAL VX(800), VY(800), VZ(800)

REAL FX(800), FY(800), FZ(800)

REAL FOX(800), FOY(800), FOZ(800)

C RTEMP is square root of reduced temperature

RTEMP = SQRT (TEMP)

DO 100 I = 1, N

VX(I) = RTEMP * GAUSS (IDUM)

VY(I) = RTEMP * GAUSS (IDUM)

VZ(I) = RTEMP * GAUSS (IDUM)

100 CONTINUE

C ** REMOVE NET MOMENTUM **

SUMX = 0.0

SUMY = 0.0

SUMZ = 0.0

DO 200 I = 1, N

SUMX = SUMX + VX(I)

```
SUMY = SUMY + VY(I)
SUMZ = SUMZ + VZ(I)
```

```
200 CONTINUE
```

```
SUMX = SUMX / REAL ( N )
SUMY = SUMY / REAL ( N )
SUMZ = SUMZ / REAL ( N )
```

```
DO 300 I = 1, N
```

```
VX(I) = VX(I) - SUMX
VY(I) = VY(I) - SUMY
VZ(I) = VZ(I) - SUMZ
```

```
300 CONTINUE
```

```
RETURN
END
```

```
REAL FUNCTION GAUSS ( IDUM )
```

```
C
```

```
*****
**
```

```
REAL      A1, A3, A5, A7, A9
PARAMETER ( A1 = 3.949846138, A3 = 0.252408784 )
PARAMETER ( A5 = 0.076542912, A7 = 0.008355968 )
PARAMETER ( A9 = 0.029899776 )
```

```
C
```

```
*****
**
```

```
SUM = 0.0
```

```
DO 10 I = 1, 12
```

```
CALL RAN2(IDUM,RANV)
SUM = SUM + RANV
```

```
10 CONTINUE
```

```
R = ( SUM - 6.0 ) / 4.0
R2 = R * R
```

```
GAUSS = ( ((( A9 * R2 + A7 ) * R2 + A5 ) * R2 + A3 ) * R2 + A1 )
      * R
```

RETURN
END

```

subroutine RAN2(idum,ran)
parameter (m=714025,ia=1366,ic=150889,rm=1./m)
dimension ir(97)
data iff/0/
if(idum.lt.0.or.iff.eq.0) then
  iff=1
  idum=mod(ic-idum,m)
  do 11 j=1,97
    idum=mod(ia*idum+ic,m)
    ir(j)=idum
11  continue
  idum=mod(ia*idum+ic,m)
  iy=idum
endif
  j=1+(97*iy)/m
  if(j.gt.97.or.j.lt.1)pause
  iy=ir(j)
  ran=iy*rm
  idum=mod(ia*idum+ic,m)
  ir(j)=idum
  return
end

```

SUBROUTINE DELETE (NS, ND)

c this routine deletes the selected polymer from solvent atoms
COMMON / BLOCK1 / RX, RY, RZ, VX, VY, VZ,

```

:   FOX, FOY, FOZ, FX, FY, FZ
REAL    RX(800), RY(800), RZ(800)
REAL    VX(800), VY(800), VZ(800)
REAL    FX(800), FY(800), FZ(800)
REAL    FOX(800), FOY(800), FOZ(800)

```

DO 100 I = ND, NS - 1

```

  RX(I) = RX(I+1)
  RY(I) = RY(I+1)
  RZ(I) = RZ(I+1)
  VX(I) = VX(I+1)
  VY(I) = VY(I+1)
  VZ(I) = VZ(I+1)

```

100 CONTINUE

RETURN
END

SUBROUTINE COUNT (NATIM, I,J)

c assigns the position and velocities of the selected polymer atom
COMMON / BLOCK1 / RX, RY, RZ, VX, VY, VZ,

```

      FOX, FOY, FOZ, FX, FY, FZ
COMMON / BLOCK2 / RCX, RCY, RCZ, VCX, VCY, VCZ,
      FOCX, FOCY, FOCZ, FCX, FCY, FCZ

```

```

REAL      RX(800), RY(800), RZ(800)
REAL      VX(800), VY(800), VZ(800)
REAL      FX(800), FY(800), FZ(800)
REAL      FOX(800), FOY(800), FOZ(800)
REAL      RCX(1,30), RCY(1,30), RCZ(1,30)
REAL      VCX(1,30), VCY(1,30), VCZ(1,30)
REAL      FCX(1,30), FCY(1,30), FCZ(1,30)
REAL      FOCX(1,30), FOCY(1,30), FOCZ(1,30)

```

```

RCX(I,J) = RX(NATIM)
RCY(I,J) = RY(NATIM)
RCZ(I,J) = RZ(NATIM)
VCX(I,J) = VX(NATIM)
VCY(I,J) = VY(NATIM)
VCZ(I,J) = VZ(NATIM)

```

```

RETURN
END

```

SUBROUTINE MOVEAP

```

COMMON / BLOCK1 / RX, RY, RZ, VX, VY, VZ,
      FOX, FOY, FOZ, FX, FY, FZ

COMMON / BLOCK2 / RCX, RCY, RCZ, VCX, VCY, VCZ,
      FOCX, FOCY, FOCZ, FCX, FCY, FCZ
COMMON / BLOCK3 / N , NS, NC, NA
COMMON / BLOCK5 / DT,DT2,DT56,DT6,DTSQ2,DTSQ23,DTSQ6

```

```

C
*****

```

```

**

```

```

C  ** FIRST PART OF VELOCITY VERLET ALGORITHM

```

```

**

```

```

C  **

```

```

REAL      RX(800), RY(800), RZ(800)
REAL      VX(800), VY(800), VZ(800)
REAL      FX(800), FY(800), FZ(800)
REAL      FOX(800), FOY(800), FOZ(800)
REAL      RCX(1,30), RCY(1,30), RCZ(1,30)
REAL      VCX(1,30), VCY(1,30), VCZ(1,30)
REAL      FCX(1,30), FCY(1,30), FCZ(1,30)
REAL      FOCX(1,30), FOCY(1,30), FOCZ(1,30)

```

```

C

```

```

*****
**

```

DO 100 I = 1, NS

```

RX(I) = RX(I) + DT * VX(I) + DTSQ23 * FX(I) - DTSQ6 * FOX(I)
RY(I) = RY(I) + DT * VY(I) + DTSQ23 * FY(I) - DTSQ6 * FOY(I)
RZ(I) = RZ(I) + DT * VZ(I) + DTSQ23 * FZ(I) - DTSQ6 * FOZ(I)
VX(I) = VX(I) + DT56 * FX(I) - DT6 * FOX(I)
VY(I) = VY(I) + DT56 * FY(I) - DT6 * FOY(I)
VZ(I) = VZ(I) + DT56 * FZ(I) - DT6 * FOZ(I)

```

100 CONTINUE

DO 110 I = 1, NC

DO 120 J = 1, NA

```

RCX(I,J) = RCX(I,J) + DT * VCX(I,J)
:      + DTSQ23 * FCX(I,J) - DTSQ6 * FOCX(I,J)
RCY(I,J) = RCY(I,J) + DT * VCY(I,J)
:      + DTSQ23 * FCY(I,J) - DTSQ6 * FOCY(I,J)
RCZ(I,J) = RCZ(I,J) + DT * VCZ(I,J)
:      + DTSQ23 * FCZ(I,J) - DTSQ6 * FOCZ(I,J)
VCX(I,J) = VCX(I,J) + DT56 * FCX(I,J) - DT6 * FOCX(I,J)
VCY(I,J) = VCY(I,J) + DT56 * FCY(I,J) - DT6 * FOCY(I,J)
VCZ(I,J) = VCZ(I,J) + DT56 * FCZ(I,J) - DT6 * FOCZ(I,J)

```

120 CONTINUE

110 CONTINUE

RETURN
END

SUBROUTINE MOVEBP (DT)

C ** SECOND PART OF VELOCITY VERLET ALGORITHM

**

```

COMMON / BLOCK1 / RX, RY, RZ, VX, VY, VZ,
:      FOX, FOY, FOZ, FX, FY, FZ
COMMON / BLOCK2 / RCX, RCY, RCZ, VCX, VCY, VCZ,
:      FOCX, FOCY, FOCZ, FCX, FCY, FCZ
COMMON / BLOCK3 / N, NS, NC, NA

```

C

**

```

REAL    RX(800), RY(800), RZ(800)
REAL    VX(800), VY(800), VZ(800)
REAL    FX(800), FY(800), FZ(800)
REAL    FOX(800), FOY(800), FOZ(800)
REAL    RCX(1,30), RCY(1,30), RCZ(1,30)

```

```

REAL      VCX(1,30), VCY(1,30), VCZ(1,30)
REAL      FCX(1,30), FCY(1,30), FCZ(1,30)
REAL      FOCX(1,30), FOCY(1,30), FOCZ(1,30)

```

```

INTEGER    I
REAL      DT2

```

C

```

*****
**

```

```

      DT3 = DT / 3.0

```

```

      DO 200 I = 1, NS

```

```

          VX(I) = VX(I) + DT3 * FX(I)
          VY(I) = VY(I) + DT3 * FY(I)
          VZ(I) = VZ(I) + DT3 * FZ(I)

```

```

200    CONTINUE

```

```

      DO 210 I = 1, NC

```

```

          DO 220 J = 1, NA

```

```

              VCX(I,J) = VCX(I,J) + DT3 * FCX(I,J)
              VCY(I,J) = VCY(I,J) + DT3 * FCY(I,J)
              VCZ(I,J) = VCZ(I,J) + DT3 * FCZ(I,J)

```

```

220    CONTINUE

```

```

210    CONTINUE

```

```

      RETURN
      END

```

```

      SUBROUTINE KINETP ( K )

```

c calculates kinetic energy

```

      COMMON / BLOCK1 / RX, RY, RZ, VX, VY, VZ,
:          FOX, FOY, FOZ, FX, FY, FZ
      COMMON / BLOCK3 / N, NS, NC, NA

```

```

      COMMON / BLOCK2 / RCX, RCY, RCZ, VCX, VCY, VCZ,
:          FOCX, FOCY, FOCZ, FCX, FCY, FCZ

```

C ** ROUTINE TO COMPUTE KINETIC ENERGY IN NOSE METHOD

**

```

REAL      K
REAL      RX(800), RY(800), RZ(800)
REAL      VX(800), VY(800), VZ(800)
REAL      FX(800), FY(800), FZ(800)
REAL      FOX(800), FOY(800), FOZ(800)
REAL      RCX(1,30), RCY(1,30), RCZ(1,30)
REAL      VCX(1,30), VCY(1,30), VCZ(1,30)
REAL      FCX(1,30), FCY(1,30), FCZ(1,30)
REAL      FOCX(1,30), FOCY(1,30), FOCZ(1,30)

```

C

```

*****
**

```

```

K = 0.0

```

```

DO 100 I = 1, NS

```

```

    K = K + VX(I) ** 2 + VY(I) ** 2 + VZ(I) ** 2

```

```

100 CONTINUE

```

```

DO 110 I = 1, NC

```

```

    DO 120 J = 1, NA

```

```

        K = K + VCX(I,J) ** 2 + VCY(I,J) ** 2 + VCZ(I,J) ** 2

```

```

120 CONTINUE

```

```

110 CONTINUE

```

```

K = 0.5 * K

```

```

RETURN
END

```

```

SUBROUTINE FORCEP ( V, VC, W, STEP)

```

```

COMMON / BLOCK1 / RX, RY, RZ, VX, VY, VZ,
:             FOX, FOY, FOZ, FX, FY, FZ

```

```

COMMON / BLOCK2 / RCX, RCY, RCZ, VCX, VCY, VCZ,
:             FOCX, FOCY, FOCZ, FCX, FCY, FCZ

```

```

COMMON / BLOCK3 / N, NS, NC, NA

```

```

COMMON / BLOCK4 / BOX, BOXINV, RCUT, RCUTSQ, VRCSS, WRCSS

```

```

COMMON / BLOCK6 / SIGPPSQ, SIGSSSQ, SIGPSSQ, EPSPP24,
:             EPSSS24, EPSPS24, WRCPS, WRCPP, VRCPS,
:             VRCPP, EPSPS4, EPSPP4, ROHSQ, RKOH,
:             HIPP, HIPS, HISS, TECL, FICL,
:             MAXBIN, MAXTET, MAXFIF, DELR, DELT, DELF,
:             RKT, RKF, TET0,
:             , IAVA

```

```

REAL      RX(800), RY(800), RZ(800)

```

```

REAL      VX(800), VY(800), VZ(800)
REAL      FX(800), FY(800), FZ(800)
REAL      FOX(800), FOY(800), FOZ(800)
REAL      RCX(1,30), RCY(1,30), RCZ(1,30)
REAL      VCX(1,30), VCY(1,30), VCZ(1,30)
REAL      FCX(1,30), FCY(1,30), FCZ(1,30)
REAL      FOCX(1,30), FOCY(1,30), FOCZ(1,30)
REAL      FTX(800), FTY(800), FTZ(800)
REAL      FCTX(1,30), FCTY(1,30), FCTZ(1,30)
REAL      HIP(50), HIPS(50), HISS(50), TECL(20), FICL(20)

```

```

REAL      BOXINV, RCUTSQ, SIGSQ
REAL      RXI, RYI, RZI, FXI, FYI, FZI
REAL      RXIJ, RYIJ, RZIJ, RIJSQ, FXIJ, FYIJ, FZIJ
REAL      SR2, SR6, SR12, VIJ, WIJ, FIJ
INTEGER   STEP
PARAMETER (PI = 3.141592654)

```

C

```

*****
**

```

C ** ZERO FORCES, POTENTIAL, VIRIAL **

```

DO 100 I = 1, NS

```

```

    FX(I) = 0.0
    FY(I) = 0.0
    FZ(I) = 0.0

```

```

100  CONTINUE

```

```

DO 110 I = 1, NC
  DO 120 J = 1, NA

```

```

    FCX(I,J) = 0.0
    FCY(I,J) = 0.0
    FCZ(I,J) = 0.0

```

```

120  CONTINUE
110  CONTINUE

```

```

NCUT = 0
V = 0.0
W = 0.0
VC = 0.0

```

C ** OUTER LOOP BEGINS **

```

DO 130 I = 1, NS - 1

```

```

    RXI = RX(I)
    RYI = RY(I)
    RZI = RZ(I)
    FXI = FX(I)

```

```
FYI = FY(I)
FZI = FZ(I)
```

```
C    ** INNER LOOP BEGINS **
```

```
DO 140 J = I + 1, NS
```

```
  RXIJ = RXI - RX(J)
  RYIJ = RYI - RY(J)
  RZIJ = RZI - RZ(J)
  RXIJ = RXIJ - ANINT ( RXIJ * BOXINV ) * BOX
  RYIJ = RYIJ - ANINT ( RYIJ * BOXINV ) * BOX
  RZIJ = RZIJ - ANINT ( RZIJ * BOXINV ) * BOX
  RIJSQ = RXIJ ** 2 + RYIJ ** 2 + RZIJ ** 2
```

```
c    calculation of solvent - solvent pair distribution function
```

```
  IF (MOD(STEP,IAVA) .EQ. 0 ) THEN
```

```
    RIJ = SQRT(RIJSQ)
```

```
    IBIN = INT(RIJ/DELR) + 1
```

```
    IF (IBIN .LE. MAXBIN ) THEN
```

```
      HISS(IBIN) = HISS(IBIN) + 2.
```

```
    ENDIF
```

```
  ENDIF
```

```
  IF ( RIJSQ .LT. RCUTSQ ) THEN
```

```
c    calculation of force between atoms i and j
```

```
    RIJ = SQRT(RIJSQ)
```

```
    RCIJ = RIJ - RCUT
```

```
    SR2 = SIGSSSQ / RIJSQ
```

```
    SR6 = SR2 * SR2 * SR2
```

```
    SR12 = SR6 ** 2
```

```
    VIJ = SR12 - SR6
```

```
    V = V + VIJ
```

```
    VC = V + WRCSS * RCIJ*6.
```

```
    WIJ = VIJ + SR12
```

```
    W = W + WIJ
```

```
    FIJ = WIJ / RIJSQ
```

```
    FIJ = FIJ - WRCSS / RIJ
```

```
    W = W - WRCSS*RIJ
```

```
    FXIJ = FIJ * RXIJ
```

```
    FYIJ = FIJ * RYIJ
```

```
    FZIJ = FIJ * RZIJ
```

```
    FXI = FXI + FXIJ
```

```
    FYI = FYI + FYIJ
```

```
    FZI = FZI + FZIJ
```

```
    FX(J) = FX(J) - FXIJ
```

```
    FY(J) = FY(J) - FYIJ
```

```
    FZ(J) = FZ(J) - FZIJ
```

```
    NCUT = NCUT + 1
```

```
  ENDIF
```

```
140    CONTINUE
```

C ** INNER LOOP ENDS **

FX(I) = FXI
FY(I) = FYI
FZ(I) = FZI

130 CONTINUE

C ** OUTER LOOP ENDS **

C ** CALCULATE SHIFTED POTENTIAL **

VC = VC - REAL (NCUT) * VRCSS

C ** MULTIPLY RESULTS BY ENERGY FACTORS **

DO 300 I = 1, NS

FX(I) = FX(I) * EPSSS24
FY(I) = FY(I) * EPSSS24
FZ(I) = FZ(I) * EPSSS24

300 CONTINUE

V = V * 4.0*EPSSS
VC = VC * 4.0*EPSSS
W = W * 24.0*EPSSS / 3.0

DO 200 I = 1, NS

c FTX,FTCX,... are arrays to accumulate forces

FTX(I) = 0.0
FTY(I) = 0.0
FTZ(I) = 0.0

200 CONTINUE

DO 210 I = 1, NC
DO 220 J = 1, NA

FCTX(I,J) = 0.0
FCTY(I,J) = 0.0
FCTZ(I,J) = 0.0

220 CONTINUE

210 CONTINUE

c WT,VT,VCT are variables to accumulate cirial ,potential and

c shifred potential

NCUTT = 0
VT = 0.0
WT = 0.0
VCT = 0.0

C ** OUTER LOOP BEGINS **

```
DO 230 I = 1, NS
```

```
  RXI = RX(I)
  RYI = RY(I)
  RZI = RZ(I)
  FXI = FTX(I)
  FYI = FTY(I)
  FZI = FTZ(I)
```

```
C  ** INNER LOOP BEGINS **
```

```
DO 231 J = 1, NC
```

```
DO 232 JA = 1, NA
```

```
  RXIJ = RXI - RCX(J,JA)
  RYIJ = RYI - RCY(J,JA)
  RZIJ = RZI - RCZ(J,JA)
  RXIJ = RXIJ - ANINT ( RXIJ * BOXINV ) * BOX
  RYIJ = RYIJ - ANINT ( RYIJ * BOXINV ) * BOX
  RZIJ = RZIJ - ANINT ( RZIJ * BOXINV ) * BOX
  RIJSQ = RXIJ ** 2 + RYIJ ** 2 + RZIJ ** 2
```

```
c  calculation of solvent - polymer pair distribution function
```

```
  IF (MOD(STEP,IAVA) .EQ. 0) THEN
```

```
    RIJ = SQRT(RIJSQ)
    IBIN = INT(RIJ/DELR) + 1
    IF (IBIN .LE. MAXBIN) THEN
      HIPS(IBIN) = HIPS(IBIN) + 2.
    ENDIF
```

```
  ENDF
```

```
c  calculation of force between atoms i and j
```

```
  IF ( RIJSQ .LT. RCUTSQ ) THEN
    RIJ = SQRT(RIJSQ)
    RCIJ = RIJ - RCUT
    SR2 = SIGPSSQ / RIJSQ
    SR6 = SR2 * SR2 * SR2
    SR12 = SR6 ** 2
    VIJ = SR12 - SR6
    VT = VT + VIJ
    VCT = VT + WRCPS * RCIJ*6.
    WIJ = VIJ + SR12
    WT = WT + WIJ
    FIJ = WIJ / RIJSQ
    FIJ = FIJ - WRCPS / RIJ
    WT = WT - WRCPS*RIJ
```

```
  FXIJ = FIJ * RXIJ
  FYIJ = FIJ * RYIJ
  FZIJ = FIJ * RZIJ
  FXI = FXI + FXIJ
```

```

    FYI = FYI + FYIJ
    FZI = FZI + FZIJ
    FCTX(J,JA) = FCTX(J,JA) - FXIJ
    FCTY(J,JA) = FCTY(J,JA) - FYIJ
    FCTZ(J,JA) = FCTZ(J,JA) - FZIJ
    NCUTT = NCUTT + 1

```

```

ENDIF

```

```

232    CONTINUE

```

```

231    CONTINUE

```

```

C    ** INNER LOOP ENDS **

```

```

    FTX(I) = FXI
    FTY(I) = FYI
    FTZ(I) = FZI

```

```

230    CONTINUE

```

```

C    ** OUTER LOOP ENDS **

```

```

C    ** CALCULATE SHIFTED POTENTIAL **

```

```

    VCT = VCT - REAL ( NCUTT ) * VRCPS

```

```

DO 240 I = 1, NS

```

```

    FX(I) = FTX(I) * EPSPS24 + FX(I)
    FY(I) = FTY(I) * EPSPS24 + FY(I)
    FZ(I) = FTZ(I) * EPSPS24 + FZ(I)

```

```

240    CONTINUE

```

```

DO 241 I = 1, NC

```

```

    DO 242 J = 1, NA

```

```

    FCX(I,J) = FCTX(I,J) * EPSPS24 + FCX(I,J)
    FCY(I,J) = FCTY(I,J) * EPSPS24 + FCY(I,J)
    FCZ(I,J) = FCTZ(I,J) * EPSPS24 + FCZ(I,J)

```

```

242    CONTINUE

```

```

241    CONTINUE

```

```

    V = VT * EPSPS4 + V
    VC = VCT * EPSPS4 + VC
    W = WT * EPSPS24 / 3.0 + W

```

```

C    ** POLIMER - POLIMER INTEEACTIONS

```

```

NCUTT = 0
VT = 0.0
WT = 0.0
VCT = 0.0

```

```

DO 310 I = 1, NC
  DO 320 J = 1, NA

```

```

    FCTX(I,J) = 0.0
    FCTY(I,J) = 0.0
    FCTZ(I,J) = 0.0

```

```

320   CONTINUE
310   CONTINUE

```

```

C   ** OUTER LOOP BEGINS **

```

```

  DO 330 I = 1, NC
    DO 331 J = 1, NA

```

```

      RXI = RCX(I,J)
      RYI = RCY(I,J)
      RZI = RCZ(I,J)
      FXI = FCTX(I,J)
      FYI = FCTY(I,J)
      FZI = FCTZ(I,J)

```

```

C   ** INNER LOOP BEGINS **

```

```

  DO 341 IA = I, NC
    IF ( IA .EQ. 1 ) THEN
      JI = J + 2
    ELSE JI = 1
    ENDIF
    IF ( JI .LE. NA ) THEN
      DO 342 JA = JI, NA

```

```

        RXIJ = RXI - RCX(IA,JA)
        RYIJ = RYI - RCY(IA,JA)
        RZIJ = RZI - RCZ(IA,JA)
        RXIJ = RXIJ - ANINT ( RXIJ * BOXINV ) * BOX
        RYIJ = RYIJ - ANINT ( RYIJ * BOXINV ) * BOX
        RZIJ = RZIJ - ANINT ( RZIJ * BOXINV ) * BOX
        RIJSQ = RXIJ ** 2 + RYIJ ** 2 + RZIJ ** 2

```

```

c   calculation of polymer - polymer pair distribution function

```

```

    IF (MOD(STEP,IAVA) .EQ. 0 ) THEN
      RIJ = SQRT(RIJSQ)
      IBIN = INT(RIJ/DELR) + 1
      IF (IBIN .LE. MAXBIN ) THEN
        HIPP(IBIN) = HIPP(IBIN) + 2.
      ENDIF
    ENDIF

```

```

c   calculation of force between atoms i and j

```

```

      IF ( RIJSQ .LT. RCUTSQ ) THEN

        RIJ = SQRT(RIJSQ)
        RCIJ = RIJ - RCUT
        SR2 = SIGPPSQ / RIJSQ
        SR6 = SR2 * SR2 * SR2
        SR12 = SR6 ** 2
        VIJ = SR12 - SR6
        VT = VT + VIJ
        VCT = VT + WRCPP * RCIJ*6.
        WIJ = VIJ + SR12
        WT = WT + WIJ
        FIJ = WIJ / RIJSQ
        FIJ = FIJ - WRCPP / RIJ
        WT = WT - WRCPP*RIJ

        FXIJ = FIJ * RXIJ
        FYIJ = FIJ * RYIJ
        FZIJ = FIJ * RZIJ
        FXI = FXI + FXIJ
        FYI = FYI + FYIJ
        FZI = FZI + FZIJ
        FCTX(IA,JA) = FCTX(IA,JA) - FXIJ
        FCTY(IA,JA) = FCTY(IA,JA) - FYIJ
        FCTZ(IA,JA) = FCTZ(IA,JA) - FZIJ
        NCUTT = NCUTT + 1

      ENDIF

342    CONTINUE
      ENDIF
341    CONTINUE

C    ** INNER LOOP ENDS **

      FCTX(I,J) = FXI
      FCTY(I,J) = FYI
      FCTZ(I,J) = FZI

331    CONTINUE
330    CONTINUE

C    ** OUTER LOOP ENDS **

C    ** CALCULATE SHIFTED POTENTIAL **

      VCT = VCT - REAL ( NCUTT ) * WRCPP

C    ** MULTIPLY RESULTS BY ENERGY FACTORS **

      DO 351 I = 1 , NC
        DO 352 J = 1 , NA

```

```

FCX(I,J) = FCTX(I,J) * EPSPP24 + FCX(I,J)
FCY(I,J) = FCTY(I,J) * EPSPP24 + FCY(I,J)
FCZ(I,J) = FCTZ(I,J) * EPSPP24 + FCZ(I,J)

```

```
352   CONTINUE
```

```
351   CONTINUE
```

```

V = VT * EPSPP4 + V
VC = VCT * EPSPP4 + VC
W = WT * EPSPP24 / 3.0 + W

```

```
c  polymer - polymer haemonic potentials
```

```
NCUTT = 0
```

```
VT = 0.0
```

```
WT = 0.0
```

```
DO 410 I = 1, NC
```

```
DO 420 J = 1, NA
```

```
FCTX(I,J) = 0.0
```

```
FCTY(I,J) = 0.0
```

```
FCTZ(I,J) = 0.0
```

```
420   CONTINUE
```

```
410   CONTINUE
```

```
DO 401 I = 1, NC
```

```
DO 402 J = 1, NA - 1
```

```
RXIJ = RCX(I,J) - RCX(I,J+1)
```

```
RYIJ = RCY(I,J) - RCY(I,J+1)
```

```
RZIJ = RCZ(I,J) - RCZ(I,J+1)
```

```
RXIJ = RXIJ - ANINT ( RXIJ * BOXINV ) * BOX
```

```
RYIJ = RYIJ - ANINT ( RYIJ * BOXINV ) * BOX
```

```
RZIJ = RZIJ - ANINT ( RZIJ * BOXINV ) * BOX
```

```
RIJSQ = RXIJ ** 2 + RYIJ ** 2 + RZIJ ** 2
```

```
c  calculation of polymer - polymer pair distribution function
```

```
IF (MOD(STEP, IAVA) .EQ. 0) THEN
```

```
RIJ = SQRT(RIJSQ)
```

```
IBIN = INT(RIJ/DELR) + 1
```

```
IF (IBIN .LE. MAXBIN) THEN
```

```
HIPP(IBIN) = HIPP(IBIN) + 2.
```

```
ENDIF
```

```
ENDIF
```

```
IF ( RIJSQ .LT. ROHSQ ) THEN
```

```
c  forces due to harmonic potentials
```

```
RPIJSQ = RIJSQ / ROHSQ
```

```
VIJ = LOG(1. - RPIJSQ)
```

```
VCT = VCT + VIJ
```

```
FIJ = 1.0 / (1 - RPIJSQ)
```

```
WT = WT + FIJ * SQRT(RIJSQ)
```

```

FXIJ = FIJ * RXIJ
FYIJ = FIJ * RYIJ
FZIJ = FIJ * RZIJ
FCTX(I,J) = FCTX(I,J) - FXIJ
FCTY(I,J) = FCTY(I,J) - FYIJ
FCTZ(I,J) = FCTZ(I,J) - FZIJ
FCTX(I,J+1) = FCTX(I,J+1) + FXIJ
FCTY(I,J+1) = FCTY(I,J+1) + FYIJ
FCTZ(I,J+1) = FCTZ(I,J+1) + FZIJ
NCUTT = NCUTT + 1

```

```

ENDIF

```

```

402   CONTINUE
401   CONTINUE

```

```

C  ** OUTER LOOP ENDS **

```

```

C  ** CALCULATE POTENTIAL **

```

```

VT = VCT

```

```

C  ** MULTIPLY RESULTS BY ENERGY FACTORS **

```

```

DO 451 I = 1 , NC

```

```

DO 452 J = 1 , NA

```

```

FCX(I,J) = FCTX(I,J) * RKOI + FCX(I,J)

```

```

FCY(I,J) = FCTY(I,J) * RKOI + FCY(I,J)

```

```

FCZ(I,J) = FCTZ(I,J) * RKOI + FCZ(I,J)

```

```

c      write(2,*)j,fctx(i,j),fcty(i,j),fctz(i,j)

```

```

452   CONTINUE

```

```

451   CONTINUE

```

```

V = V - 0.5 * RKOI * ROHSQ * VT

```

```

VC = VC - 0.5 * RKOI * ROHSQ * VT

```

```

W = W + RKOI * WT

```

```

RETURN

```

```

END

```

APPENDIX B

THE DATA FILE

0 324 1 12 7.4 2.5 2000 5000 10 50 0.75 1. 1. 1. 0.4 0.1 0.4 1.54 0.01
20
0.1 10 12 12 50 20 5.0 0.2 1.23015
1 1 1
-1 -1 1
1 -1 -1
-1 1 -1
-1 -1 -1
1 1 -1
-1 1 1
1 -1 1

APPENDIX C

THE SIMULATION OUTPUTS

EQUILIBRATION

TIMESTEP	ENERGY	CUTENERGY	KINETIC	POTENT	TEMPER	PRESSURE	<r2>	<s2>
20	0.449	0.880	0.732	-0.284	0.488	0.084	41.639	5.766
40	0.572	1.004	0.885	-0.312	0.590	0.097	42.066	5.416
60	0.633	1.064	0.906	-0.273	0.604	0.093	41.052	4.831
80	0.656	1.088	0.977	-0.320	0.651	0.091	36.600	4.064
100	0.674	1.106	0.965	-0.291	0.643	0.075	31.312	3.455
120	0.629	1.061	0.953	-0.324	0.635	0.073	27.909	3.041
140	0.711	1.142	1.003	-0.292	0.669	0.089	26.114	2.855
160	0.564	0.995	0.953	-0.389	0.635	0.065	22.144	2.602
180	0.705	1.137	0.995	-0.289	0.663	0.035	23.505	2.639
200	0.734	1.166	1.036	-0.302	0.691	0.093	23.396	2.813
220	0.719	1.150	1.022	-0.303	0.681	0.060	20.554	2.702
240	0.694	1.125	0.988	-0.294	0.658	0.065	19.253	2.529
260	0.640	1.072	0.974	-0.334	0.649	0.147	17.767	2.255
280	0.636	1.068	0.958	-0.322	0.638	0.137	16.151	2.049
300	0.680	1.111	0.988	-0.308	0.659	0.079	17.619	2.160
320	0.699	1.131	0.971	-0.272	0.647	0.092	18.165	2.307
340	0.717	1.148	1.036	-0.319	0.691	0.093	17.006	2.171
360	0.721	1.153	0.963	-0.242	0.642	0.054	18.003	2.122
380	0.653	1.085	0.994	-0.341	0.663	0.090	18.127	1.988
400	0.672	1.104	0.960	-0.288	0.640	0.073	18.959	2.102
420	0.680	1.112	1.012	-0.332	0.675	0.085	20.817	2.226
440	0.720	1.152	1.029	-0.309	0.686	0.101	20.376	2.189
460	0.702	1.134	0.999	-0.297	0.666	0.121	20.423	2.179
480	0.796	1.228	1.023	-0.227	0.682	0.062	20.552	2.220
500	0.689	1.120	0.961	-0.272	0.641	0.095	20.217	2.150
520	0.713	1.144	0.986	-0.274	0.658	0.068	18.640	2.089
540	0.705	1.137	1.016	-0.311	0.677	0.107	18.566	1.972
560	0.708	1.140	0.988	-0.280	0.659	0.091	17.633	1.946
580	0.578	1.009	0.916	-0.338	0.611	0.052	16.430	1.877
600	0.726	1.158	1.004	-0.278	0.669	0.073	17.982	1.967
620	0.764	1.196	1.019	-0.254	0.679	0.110	18.745	2.151
640	0.661	1.093	0.974	-0.313	0.650	0.131	19.070	2.120
660	0.754	1.186	1.009	-0.255	0.672	0.098	20.042	2.183
680	0.718	1.150	1.005	-0.286	0.670	0.050	21.634	2.334
700	0.650	1.081	0.912	-0.263	0.608	0.048	20.004	2.221
720	0.645	1.077	0.958	-0.313	0.639	0.087	18.673	2.039
740	0.647	1.079	0.937	-0.291	0.625	0.061	15.958	2.130
760	0.704	1.136	0.947	-0.242	0.631	0.055	15.512	2.330
780	0.696	1.128	1.021	-0.325	0.681	0.073	18.280	2.387
800	0.710	1.142	0.971	-0.260	0.647	0.099	21.726	2.484
820	0.681	1.113	0.999	-0.318	0.666	0.069	20.720	2.406
840	0.655	1.086	0.994	-0.339	0.662	0.083	17.533	2.437
860	0.735	1.167	0.995	-0.260	0.664	0.062	17.418	2.498
880	0.607	1.038	0.994	-0.387	0.663	0.100	18.645	2.499
900	0.734	1.166	1.001	-0.266	0.667	0.058	19.061	2.594
920	0.640	1.071	0.953	-0.313	0.635	0.041	18.767	2.575
940	0.659	1.091	0.953	-0.294	0.635	0.028	20.159	2.636
960	0.611	1.042	0.985	-0.374	0.656	0.102	22.409	2.618
980	0.704	1.135	0.987	-0.283	0.658	0.042	22.659	2.701
1000	0.699	1.130	1.012	-0.313	0.675	0.037	21.874	2.682

1020	0.629	1.060	0.968	-0.340	0.645	0.084	21.838	2.699
1040	0.641	1.073	0.923	-0.282	0.615	0.110	24.540	2.827
1060	0.712	1.144	1.023	-0.311	0.682	0.112	24.251	2.654
1080	0.676	1.107	1.016	-0.340	0.677	0.089	22.874	2.507
1100	0.685	1.117	1.026	-0.341	0.684	0.101	18.658	2.267
1120	0.745	1.176	1.039	-0.295	0.693	0.071	18.056	2.017
1140	0.698	1.130	0.985	-0.287	0.657	0.092	18.592	2.014
1160	0.705	1.137	1.010	-0.305	0.673	0.058	18.088	2.110
1180	0.670	1.101	0.992	-0.322	0.661	0.112	18.507	2.050
1200	0.669	1.101	0.992	-0.324	0.662	0.084	18.581	2.134
1220	0.663	1.095	0.976	-0.313	0.651	0.070	21.450	2.280
1240	0.686	1.117	0.992	-0.306	0.661	0.073	21.937	2.392
1260	0.677	1.109	0.992	-0.315	0.661	0.012	21.475	2.385
1280	0.680	1.111	0.985	-0.305	0.657	0.034	21.227	2.425
1300	0.701	1.133	1.020	-0.319	0.680	0.082	21.775	2.583
1320	0.672	1.104	0.973	-0.301	0.649	0.034	22.387	2.731
1340	0.653	1.084	0.961	-0.309	0.641	0.054	23.976	2.795
1360	0.677	1.108	0.993	-0.316	0.662	0.074	25.728	2.854
1380	0.656	1.088	0.988	-0.332	0.659	0.036	23.378	2.686
1400	0.708	1.140	1.002	-0.294	0.668	0.033	21.302	2.433
1420	0.665	1.097	0.962	-0.297	0.641	0.064	19.246	2.145
1440	0.664	1.096	0.980	-0.316	0.653	0.111	16.575	2.102
1460	0.767	1.199	1.042	-0.275	0.695	0.064	16.899	2.074
1480	0.653	1.085	0.982	-0.329	0.655	0.070	18.224	1.907
1500	0.696	1.128	0.973	-0.277	0.649	0.056	18.478	1.893
1520	0.743	1.174	0.998	-0.255	0.665	0.043	17.311	1.908
1540	0.731	1.162	1.023	-0.292	0.682	0.098	15.695	1.892
1560	0.714	1.146	1.029	-0.315	0.686	0.093	17.339	2.076
1580	0.668	1.099	0.955	-0.287	0.637	0.045	18.176	2.190
1600	0.713	1.145	1.007	-0.293	0.671	0.105	16.226	1.955
1620	0.783	1.215	1.049	-0.266	0.700	0.109	15.244	1.821
1640	0.705	1.137	0.987	-0.282	0.658	0.062	12.773	1.747
1660	0.777	1.208	1.016	-0.239	0.677	0.093	12.462	1.785
1680	0.785	1.217	0.995	-0.210	0.663	0.055	13.195	1.682
1700	0.727	1.159	0.983	-0.257	0.655	0.084	11.929	1.463
1720	0.781	1.213	1.015	-0.234	0.676	0.075	10.031	1.390
1740	0.768	1.200	0.964	-0.196	0.642	0.075	8.582	1.399
1760	0.760	1.192	1.016	-0.257	0.678	0.098	6.966	1.289
1780	0.712	1.144	0.972	-0.260	0.648	0.083	7.406	1.276
1800	0.693	1.125	0.921	-0.228	0.614	0.033	6.905	1.261
1820	0.705	1.138	0.977	-0.272	0.652	0.067	5.686	1.106
1840	0.789	1.221	0.986	-0.197	0.658	0.080	4.227	1.115
1860	0.756	1.188	0.977	-0.221	0.651	0.109	4.355	1.083
1880	0.725	1.157	0.966	-0.242	0.644	0.041	4.930	1.149
1900	0.773	1.205	0.995	-0.223	0.664	0.069	4.915	1.286
1920	0.783	1.215	1.011	-0.228	0.674	0.099	2.881	1.217
1940	0.761	1.193	1.000	-0.239	0.666	0.097	2.434	1.131
1960	0.813	1.245	1.020	-0.207	0.680	0.078	3.120	1.169
1980	0.815	1.246	1.065	-0.251	0.710	0.100	2.774	1.218
2000	0.764	1.196	0.971	-0.207	0.647	0.064	2.433	1.261
AVERAGES	0.697	1.128	0.986	-0.289	0.657	0.073	18.330	2.257
FLUCTS	0.059	0.059	0.042	0.041	0.028	0.028		

AUTOCORRELATION FUNCTION

1.0000
 0.7938
 0.5360
 0.3914
 0.3083
 0.2281
 0.1555
 0.1036
 0.0667

0.0530
 0.0638
 0.0587
 0.0417
 0.0554
 0.1024
 0.1293
 0.1324
 0.1601
 0.1955
 0.1846
 0.1455
 0.1148
 0.1018
 0.0930
 0.0861
 0.0768
 0.0687
 0.0694
 0.0676
 0.0538
 0.0241

DIFFUSION COEFFICIENT = 17.16598

PAIR CORRELATION FUNCTIONS

solvent-solvent pair distribution fn

0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 6.4125918E-03
 0.7520965
 2.080937
 1.751353
 1.202036
 0.8792240
 0.7131238
 0.6716412
 0.7024384
 0.8009313
 0.9113868
 1.027866
 1.085721
 1.049287
 0.9745327
 0.8976204
 0.8648713
 0.8531926
 0.8783323
 0.9131591
 0.9416978
 0.9600811
 0.9668137
 0.9525756
 0.9310992
 0.9105913
 0.9045542

0.9096985
0.9192122
0.8917876
0.8266290
0.7570810
0.6904511
0.6252586
0.5636789
0.5096845
0.4585129
0.4085971
0.3604485
0.3125862
0.2674808
0.2252953

solvent-polymer pair distribution fn

0.0000000E+00
0.0000000E+00
0.0000000E+00
0.0000000E+00
0.0000000E+00
0.0000000E+00
0.0000000E+00
0.0000000E+00
0.0000000E+00
1.0786744E-02
6.3828059E-02
8.7755546E-02
8.2242645E-02
7.0776209E-02
6.4289086E-02
5.8914606E-02
5.4525793E-02
5.8450997E-02
6.1630152E-02
6.7406446E-02
7.3319748E-02
7.4710257E-02
7.4481122E-02
7.2625540E-02
6.9163233E-02
6.7409448E-02
6.8730965E-02
7.0640102E-02
7.1632028E-02
7.3632948E-02
7.4111946E-02
7.2739683E-02
7.3219083E-02
7.2211549E-02
7.1629785E-02
7.0835881E-02
7.1961530E-02
7.2354116E-02
6.9451250E-02
6.4879157E-02
5.9460472E-02
5.3760871E-02
4.8863176E-02
4.4327419E-02
3.9673384E-02
3.5620905E-02
3.1710289E-02

2.7939025E-02
 2.4487866E-02
 2.0995630E-02
 1.7711317E-02

polymer-polymer pair distribution fn

4.6077240E-02
 3.9494779E-02
 3.8801886E-02
 4.5579113E-02
 4.6983685E-02
 4.5975965E-02
 3.4975152E-02
 1.8485423E-02
 7.9414211E-03
 1.4962357E-02
 2.6950307E-02
 1.5041838E-02
 8.8617625E-03
 7.9182051E-03
 6.5574264E-03
 6.4163026E-03
 5.4706163E-03
 4.3219323E-03
 3.5892678E-03
 3.4244957E-03
 3.0182272E-03
 2.4915629E-03
 2.0809090E-03
 1.8408677E-03
 1.6987958E-03
 1.5162285E-03
 1.3296136E-03
 1.0031814E-03
 9.6427696E-04
 8.6825114E-04
 7.4951921E-04
 6.2220590E-04
 5.3798070E-04
 4.8444734E-04
 4.5161130E-04
 3.9971856E-04
 3.5045011E-04
 2.6211317E-04
 2.8597561E-04
 2.4214938E-04
 1.7415915E-04
 1.9262047E-04
 1.4284905E-04
 1.4285016E-04
 9.7723263E-05
 8.0121448E-05
 5.6824174E-05
 3.4035507E-05
 3.3952379E-05
 1.5043588E-05

DYNAMICS

TIMESTEP	ENERGY	CUTENERGY	KINETIC	POTENT	TEMPER	PRESSURE	<r2>	<s2>
20	0.775	1.207	1.013	-0.238	0.676	0.074	2.974	1.326
40	0.784	1.216	0.989	-0.205	0.660	0.049	2.488	1.422

60	0.666	1.098	0.974	-0.308	0.649	0.062	3.172	1.387
80	0.787	1.219	0.989	-0.202	0.659	0.068	4.147	1.439
100	0.781	1.212	0.967	-0.187	0.645	0.137	4.195	1.374
120	0.825	1.257	0.990	-0.165	0.660	0.059	3.891	1.328
140	0.773	1.205	1.038	-0.265	0.692	0.112	2.916	1.285
160	0.734	1.165	0.953	-0.220	0.636	0.030	2.788	1.298
180	0.799	1.231	1.024	-0.225	0.683	0.079	3.191	1.229
200	0.792	1.224	0.998	-0.206	0.666	0.085	4.463	1.197
220	0.786	1.218	0.994	-0.208	0.663	0.070	3.141	1.160
240	0.806	1.238	1.008	-0.203	0.672	0.036	3.611	1.164
260	0.749	1.181	0.985	-0.236	0.656	0.098	4.172	1.170
280	0.773	1.205	0.980	-0.207	0.653	0.064	3.655	1.231
300	0.795	1.227	0.975	-0.179	0.650	0.009	3.400	1.192
320	0.779	1.211	1.014	-0.235	0.676	0.082	2.267	1.038
340	0.793	1.225	0.972	-0.179	0.648	0.085	1.051	1.024
360	0.834	1.266	1.017	-0.183	0.678	0.067	1.418	1.071
380	0.764	1.196	1.025	-0.260	0.683	0.082	2.067	1.056
400	0.747	1.179	0.961	-0.214	0.641	0.042	1.799	1.018
420	0.821	1.253	0.962	-0.141	0.641	0.050	1.183	1.043
440	0.808	1.239	1.004	-0.197	0.670	0.062	1.624	1.038
460	0.746	1.178	0.975	-0.229	0.650	0.070	1.554	0.992
480	0.876	1.308	1.043	-0.167	0.695	0.056	1.220	0.996
500	0.688	1.120	1.017	-0.329	0.678	0.136	1.046	1.019
520	0.852	1.284	1.029	-0.177	0.686	0.134	1.272	1.076
540	0.764	1.196	0.982	-0.218	0.654	0.073	2.218	1.104
560	0.777	1.209	1.008	-0.231	0.672	0.050	2.483	1.040
580	0.801	1.233	1.044	-0.243	0.696	0.164	2.495	0.923
600	0.810	1.242	0.981	-0.171	0.654	0.031	2.620	1.015
620	0.736	1.168	0.977	-0.242	0.651	0.049	2.849	1.018
640	0.727	1.158	0.956	-0.229	0.637	0.086	2.763	1.020
660	0.736	1.168	0.995	-0.259	0.663	0.083	2.879	1.028
680	0.740	1.172	0.969	-0.229	0.646	0.128	3.224	1.067
700	0.736	1.168	1.029	-0.293	0.686	0.078	3.814	1.068
720	0.803	1.235	0.977	-0.174	0.651	0.071	4.741	1.203
740	0.789	1.221	1.009	-0.220	0.673	0.114	4.812	1.162
760	0.818	1.250	0.992	-0.174	0.661	0.020	3.957	1.206
780	0.815	1.247	1.030	-0.215	0.687	0.048	3.388	1.047
800	0.795	1.227	0.974	-0.178	0.649	0.048	3.275	1.023
820	0.791	1.223	0.998	-0.208	0.666	0.096	2.740	0.990
840	0.880	1.312	1.033	-0.153	0.688	0.071	3.192	0.924
860	0.859	1.291	1.006	-0.148	0.671	0.093	3.341	0.956
880	0.839	1.271	1.016	-0.177	0.677	0.084	3.366	0.977
900	0.823	1.255	1.025	-0.202	0.683	0.099	4.777	1.052
920	0.787	1.219	0.990	-0.203	0.660	0.091	4.924	1.124
940	0.688	1.120	0.942	-0.254	0.628	0.081	4.136	1.016
960	0.807	1.239	0.960	-0.152	0.640	0.050	3.610	1.046
980	0.763	1.195	0.995	-0.232	0.663	0.076	3.777	1.020
1000	0.842	1.273	0.997	-0.155	0.665	0.075	4.600	1.123
1020	0.749	1.181	1.061	-0.311	0.707	0.148	4.703	1.049
1040	0.797	1.229	1.003	-0.206	0.669	0.040	4.306	1.049
1060	0.776	1.208	0.984	-0.207	0.656	0.055	3.093	0.958
1080	0.734	1.166	0.987	-0.253	0.658	0.090	3.315	0.973
1100	0.734	1.166	0.983	-0.249	0.656	0.011	4.054	1.115
1120	0.736	1.167	0.983	-0.247	0.655	0.046	4.233	1.163
1140	0.774	1.206	1.000	-0.226	0.667	0.044	3.826	1.098
1160	0.705	1.137	0.968	-0.262	0.645	0.039	4.194	1.102
1180	0.682	1.113	0.983	-0.301	0.655	0.116	5.228	1.202
1200	0.734	1.166	0.946	-0.212	0.630	0.029	6.111	1.306
1220	0.684	1.116	0.991	-0.307	0.661	0.067	6.309	1.192
1240	0.738	1.170	0.977	-0.240	0.652	0.096	6.711	1.259
1260	0.675	1.107	0.976	-0.300	0.650	0.092	5.845	1.265
1280	0.778	1.210	1.029	-0.251	0.686	0.082	4.350	1.310
1300	0.714	1.146	1.023	-0.310	0.682	0.120	3.715	1.302

1320	0.697	1.128	0.957	-0.261	0.638	0.085	5.654	1.398
1340	0.723	1.154	0.974	-0.251	0.649	0.057	6.132	1.282
1360	0.743	1.175	1.014	-0.271	0.676	0.133	6.218	1.252
1380	0.745	1.177	1.005	-0.260	0.670	0.072	6.193	1.276
1400	0.745	1.177	1.014	-0.268	0.676	0.089	6.382	1.198
1420	0.771	1.203	0.969	-0.198	0.646	0.040	5.894	1.226
1440	0.771	1.203	0.973	-0.202	0.649	0.087	5.234	1.129
1460	0.811	1.242	1.026	-0.215	0.684	0.105	7.296	1.203
1480	0.772	1.204	0.975	-0.203	0.650	0.086	8.811	1.325
1500	0.745	1.177	0.976	-0.232	0.651	0.082	7.777	1.403
1520	0.820	1.251	1.022	-0.203	0.682	0.139	7.627	1.553
1540	0.736	1.168	0.980	-0.244	0.653	0.045	8.993	1.643
1560	0.752	1.184	0.988	-0.236	0.659	0.077	11.684	1.819
1580	0.736	1.168	1.015	-0.279	0.677	0.116	12.530	1.976
1600	0.762	1.194	0.992	-0.230	0.662	0.060	10.494	1.972
1620	0.740	1.171	0.990	-0.251	0.660	0.081	10.006	1.886
1640	0.794	1.226	1.028	-0.234	0.685	0.093	10.712	1.812
1660	0.785	1.217	1.011	-0.226	0.674	0.042	12.808	1.767
1680	0.713	1.144	0.985	-0.272	0.656	0.071	13.557	1.726
1700	0.669	1.101	0.945	-0.275	0.630	0.083	12.643	1.643
1720	0.767	1.198	1.023	-0.257	0.682	0.051	12.348	1.696
1740	0.784	1.216	1.040	-0.256	0.694	0.070	13.615	1.766
1760	0.770	1.202	1.004	-0.234	0.669	0.089	12.315	1.882
1780	0.795	1.227	1.037	-0.241	0.691	0.033	11.824	1.863
1800	0.743	1.175	1.022	-0.279	0.681	0.058	12.087	1.791
1820	0.768	1.199	0.989	-0.221	0.659	0.039	12.104	1.787
1840	0.736	1.168	1.016	-0.279	0.677	0.088	11.026	1.777
1860	0.702	1.134	0.993	-0.291	0.662	0.103	10.875	1.857
1880	0.671	1.102	0.960	-0.290	0.640	0.061	11.600	1.992
1900	0.685	1.117	0.959	-0.274	0.639	0.004	11.828	2.041
1920	0.689	1.120	0.984	-0.295	0.656	0.058	11.053	1.945
1940	0.709	1.140	0.984	-0.275	0.656	0.072	10.453	1.951
1960	0.705	1.137	0.974	-0.269	0.649	0.103	10.291	2.025
1980	0.683	1.114	0.971	-0.289	0.648	0.056	8.962	1.954
2000	0.738	1.170	1.007	-0.270	0.672	0.066	9.535	1.850
2020	0.718	1.150	0.992	-0.273	0.661	0.064	11.500	1.837
2040	0.757	1.188	1.064	-0.308	0.710	0.111	13.174	1.912
2060	0.774	1.206	1.018	-0.244	0.679	0.109	13.991	2.075
2080	0.704	1.135	1.022	-0.318	0.681	0.086	14.774	2.131
2100	0.711	1.142	0.979	-0.269	0.653	0.066	15.392	2.379
2120	0.721	1.153	1.012	-0.291	0.675	0.076	16.117	2.468
2140	0.686	1.118	1.031	-0.345	0.687	0.070	16.689	2.375
2160	0.680	1.112	1.005	-0.324	0.670	0.102	17.251	2.385
2180	0.690	1.122	1.019	-0.329	0.679	0.092	17.084	2.347
2200	0.621	1.052	0.967	-0.347	0.645	0.034	17.316	2.270
2220	0.717	1.149	1.022	-0.305	0.681	0.076	15.910	2.255
2240	0.707	1.139	1.000	-0.293	0.667	0.075	19.237	2.287
2260	0.690	1.121	1.011	-0.321	0.674	0.140	21.435	2.310
2280	0.659	1.091	0.994	-0.335	0.663	0.093	22.030	2.346
2300	0.634	1.066	0.959	-0.325	0.639	0.072	20.939	2.310
2320	0.667	1.099	1.020	-0.353	0.680	0.064	20.454	2.284
2340	0.692	1.124	0.974	-0.282	0.649	0.051	21.486	2.363
2360	0.730	1.162	0.983	-0.253	0.655	0.033	21.791	2.422
2380	0.609	1.041	0.943	-0.334	0.629	0.053	20.592	2.323
2400	0.769	1.201	1.022	-0.253	0.681	0.070	20.860	2.132
2420	0.707	1.139	1.015	-0.308	0.677	0.090	16.716	1.870
2440	0.617	1.048	0.940	-0.323	0.627	0.069	14.649	1.805
2460	0.697	1.128	0.989	-0.293	0.659	0.101	12.564	1.777
2480	0.694	1.126	0.934	-0.240	0.623	0.120	13.430	1.757
2500	0.815	1.246	1.033	-0.219	0.689	0.102	17.012	1.806
2520	0.707	1.139	0.986	-0.278	0.657	0.104	16.508	1.844
2540	0.771	1.203	0.991	-0.220	0.660	0.075	14.017	1.840
2560	0.727	1.158	0.973	-0.246	0.649	0.032	15.451	2.032

2580	0.734	1.166	1.015	-0.281	0.677	0.078	14.706	1.893
2600	0.765	1.197	1.029	-0.264	0.686	0.079	12.943	1.815
2620	0.732	1.164	0.971	-0.239	0.647	0.102	13.367	1.799
2640	0.708	1.140	0.976	-0.268	0.651	0.070	13.172	1.728
2660	0.750	1.182	0.990	-0.239	0.660	0.033	11.763	1.807
2680	0.801	1.233	1.041	-0.241	0.694	0.062	13.341	1.907
2700	0.742	1.173	1.007	-0.265	0.671	0.038	13.116	1.974
2720	0.790	1.221	1.042	-0.252	0.694	0.062	12.715	2.072
2740	0.704	1.136	0.991	-0.287	0.661	0.057	12.111	2.237
2760	0.618	1.050	0.968	-0.350	0.645	0.115	12.483	2.230
2780	0.630	1.061	0.967	-0.337	0.645	0.044	12.170	2.206
2800	0.698	1.130	0.971	-0.273	0.648	0.060	13.995	2.141
2820	0.688	1.120	0.982	-0.294	0.655	0.097	13.464	2.037
2840	0.768	1.200	1.024	-0.256	0.683	0.074	12.188	1.886
2860	0.723	1.154	1.002	-0.279	0.668	0.091	10.164	1.790
2880	0.716	1.148	0.976	-0.260	0.651	0.108	9.972	1.731
2900	0.735	1.167	1.008	-0.273	0.672	0.104	9.565	1.771
2920	0.778	1.210	1.055	-0.277	0.703	0.068	11.537	1.914
2940	0.741	1.173	1.008	-0.267	0.672	0.169	10.426	1.803
2960	0.746	1.177	1.008	-0.262	0.672	0.079	8.772	1.647
2980	0.729	1.161	0.973	-0.243	0.649	0.098	7.038	1.491
3000	0.780	1.212	1.027	-0.247	0.685	0.095	5.707	1.434
3020	0.777	1.208	1.025	-0.249	0.683	0.052	6.753	1.603
3040	0.696	1.128	0.969	-0.273	0.646	0.061	7.402	1.678
3060	0.730	1.162	0.974	-0.243	0.649	0.090	6.012	1.499
3080	0.657	1.089	0.992	-0.334	0.661	0.123	6.513	1.223
3100	0.781	1.213	0.985	-0.204	0.657	0.132	5.932	1.318
3120	0.634	1.066	1.020	-0.386	0.680	0.134	5.586	1.340
3140	0.702	1.134	0.962	-0.260	0.642	0.044	6.036	1.424
3160	0.758	1.190	0.982	-0.223	0.654	0.068	6.912	1.300
3180	0.717	1.149	0.999	-0.282	0.666	0.118	6.726	1.211
3200	0.791	1.223	1.012	-0.220	0.675	0.047	7.026	1.171
3220	0.797	1.229	0.993	-0.196	0.662	0.052	6.995	1.142
3240	0.773	1.205	1.001	-0.228	0.667	0.103	6.213	1.154
3260	0.797	1.229	1.012	-0.215	0.675	0.051	6.814	1.131
3280	0.801	1.233	0.973	-0.172	0.649	0.007	7.112	1.096
3300	0.714	1.146	0.978	-0.264	0.652	0.112	5.393	0.875
3320	0.913	1.345	1.029	-0.116	0.686	0.115	4.892	0.928
3340	0.854	1.286	1.053	-0.198	0.702	0.141	4.456	0.836
3360	0.926	1.358	1.022	-0.096	0.681	0.062	4.895	0.916
3380	0.795	1.227	0.977	-0.183	0.652	0.115	5.639	0.876
3400	0.901	1.333	1.025	-0.125	0.683	0.064	4.479	0.842
3420	0.840	1.272	0.985	-0.145	0.657	0.087	3.434	0.935
3440	0.776	1.208	0.966	-0.190	0.644	0.053	2.699	0.816
3460	0.879	1.311	1.022	-0.144	0.682	0.084	1.871	0.895
3480	0.821	1.253	0.999	-0.178	0.666	0.047	1.179	0.881
3500	0.853	1.285	1.025	-0.172	0.683	0.099	1.118	0.896
3520	0.856	1.288	0.972	-0.116	0.648	0.056	1.278	0.949
3540	0.755	1.187	0.969	-0.214	0.646	0.080	1.271	0.891
3560	0.724	1.156	0.948	-0.224	0.632	0.060	1.670	0.984
3580	0.735	1.167	0.959	-0.224	0.639	0.055	2.306	1.078
3600	0.776	1.208	1.017	-0.241	0.678	0.125	2.079	1.047
3620	0.716	1.148	0.968	-0.252	0.646	0.074	2.017	1.044
3640	0.775	1.207	1.016	-0.241	0.677	0.062	2.408	1.073
3660	0.773	1.205	0.981	-0.207	0.654	0.058	2.559	1.147
3680	0.688	1.120	0.963	-0.274	0.642	0.059	2.300	1.024
3700	0.841	1.273	0.986	-0.145	0.657	0.113	2.491	1.121
3720	0.759	1.191	0.992	-0.232	0.661	0.063	2.976	1.021
3740	0.771	1.203	0.973	-0.202	0.649	0.016	4.062	1.075
3760	0.761	1.193	0.988	-0.228	0.659	0.037	4.101	1.101
3780	0.752	1.184	1.001	-0.249	0.667	0.115	3.472	0.971
3800	0.814	1.246	1.005	-0.192	0.670	0.074	4.476	1.058
3820	0.861	1.293	1.056	-0.196	0.704	0.080	4.809	1.009

3840	0.809	1.241	1.001	-0.193	0.667	0.089	4.675	0.975
3860	0.854	1.286	1.023	-0.169	0.682	0.054	4.024	1.046
3880	0.810	1.242	1.021	-0.210	0.680	0.095	3.999	1.006
3900	0.866	1.298	1.029	-0.162	0.686	0.131	3.490	1.065
3920	0.833	1.265	0.995	-0.162	0.663	0.031	2.607	1.078
3940	0.762	1.194	0.972	-0.210	0.648	0.077	1.689	1.007
3960	0.690	1.122	0.938	-0.248	0.626	0.070	1.916	0.922
3980	0.855	1.287	0.998	-0.143	0.666	0.034	3.259	0.991
4000	0.808	1.240	0.984	-0.176	0.656	0.072	3.868	1.122
4020	0.828	1.260	1.034	-0.206	0.689	0.089	3.065	1.090
4040	0.811	1.243	1.025	-0.214	0.683	0.111	3.109	1.071
4060	0.818	1.250	1.001	-0.183	0.668	0.078	2.741	1.036
4080	0.838	1.270	1.014	-0.176	0.676	0.073	2.323	1.115
4100	0.799	1.230	0.964	-0.165	0.643	0.032	1.806	1.088
4120	0.811	1.242	0.992	-0.182	0.662	0.085	1.223	0.910
4140	0.791	1.223	0.983	-0.192	0.655	0.062	1.361	0.974
4160	0.836	1.268	1.009	-0.173	0.672	0.065	1.597	1.011
4180	0.832	1.264	1.002	-0.170	0.668	0.040	1.652	0.900
4200	0.822	1.254	0.985	-0.163	0.657	0.093	1.077	0.948
4220	0.858	1.290	1.023	-0.165	0.682	0.103	1.114	0.917
4240	0.855	1.287	1.040	-0.185	0.693	0.130	1.405	0.900
4260	0.867	1.299	0.990	-0.123	0.660	0.066	1.854	0.908
4280	0.742	1.174	0.948	-0.205	0.632	0.081	2.241	0.955
4300	0.774	1.206	0.988	-0.214	0.659	0.037	2.587	1.071
4320	0.652	1.084	0.999	-0.348	0.666	0.166	2.092	0.964
4340	0.770	1.202	0.983	-0.213	0.655	0.022	1.827	1.018
4360	0.834	1.266	1.040	-0.206	0.694	0.089	1.372	1.013
4380	0.805	1.237	0.987	-0.183	0.658	0.036	1.316	0.983
4400	0.692	1.124	0.957	-0.265	0.638	0.115	1.830	1.026
4420	0.793	1.225	0.985	-0.192	0.657	0.077	1.384	1.037
4440	0.731	1.163	0.995	-0.263	0.663	0.118	1.016	0.995
4460	0.793	1.225	1.059	-0.266	0.706	0.087	1.384	1.089
4480	0.748	1.180	0.980	-0.233	0.654	0.088	1.123	1.033
4500	0.857	1.289	1.000	-0.142	0.666	0.106	1.355	0.947
4520	0.879	1.311	1.005	-0.127	0.670	0.093	2.167	0.995
4540	0.889	1.321	1.054	-0.165	0.703	0.092	2.375	0.959
4560	0.781	1.213	1.019	-0.238	0.679	0.073	2.456	0.882
4580	0.869	1.301	1.049	-0.180	0.699	0.128	2.002	0.923
4600	0.873	1.305	1.016	-0.144	0.678	0.089	1.673	0.842
4620	0.831	1.263	0.984	-0.152	0.656	0.069	0.904	0.862
4640	0.781	1.213	0.969	-0.188	0.646	0.115	1.866	0.843
4660	0.886	1.318	1.038	-0.152	0.692	0.124	2.127	0.852
4680	0.844	1.276	0.983	-0.139	0.655	0.047	2.776	0.955
4700	0.728	1.161	0.979	-0.251	0.653	0.088	3.238	0.845
4720	0.769	1.201	0.961	-0.192	0.641	0.049	4.190	0.963
4740	0.805	1.237	1.021	-0.216	0.681	0.103	4.831	1.005
4760	0.785	1.217	0.969	-0.183	0.646	0.049	5.416	0.902
4780	0.888	1.320	1.000	-0.112	0.667	0.063	5.881	0.884
4800	0.943	1.375	1.022	-0.079	0.681	0.117	4.608	0.845
4820	0.683	1.115	0.959	-0.275	0.639	0.127	3.793	0.745
4840	0.858	1.290	0.999	-0.142	0.666	0.116	4.371	0.840
4860	0.902	1.334	1.010	-0.108	0.673	0.046	5.039	0.826
4880	0.858	1.290	0.957	-0.099	0.638	0.035	4.835	0.823
4900	0.813	1.245	0.948	-0.135	0.632	0.002	5.328	0.875
4920	0.906	1.338	1.038	-0.132	0.692	0.062	5.213	0.818
4940	0.955	1.387	1.066	-0.111	0.711	0.070	5.646	0.873
4960	0.707	1.139	0.941	-0.234	0.627	0.065	4.313	0.879
4980	0.829	1.261	0.983	-0.154	0.655	0.051	4.457	0.902
5000	0.766	1.198	0.937	-0.171	0.624	0.027	3.505	0.874
AVERAGES	0.769	1.201	0.995	-0.226	0.663	0.076	6.579	1.336
FLUCTS	-0.063	0.063	0.026	0.055	0.017	0.031		

CENTER OF MASS VELOCITY AUTOCORRELATION FUNCTION

1.0000
 0.7905
 0.5505
 0.4047
 0.3187
 0.2478
 0.1899
 0.1609
 0.1322
 0.1188
 0.1214
 0.1160
 0.1093
 0.1018
 0.1215
 0.1376
 0.1370
 0.1161
 0.0894
 0.0758
 0.0585
 0.0562
 0.0463
 0.0121
 -0.0318
 -0.0465
 -0.0488
 -0.0516
 -0.0477
 -0.0532
 -0.0364

DIFFUSION COEFFICIENT = 14.71796

PAIR CORRELATION FUNCTIONS

solvent-solvent pair distribution fn

0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 5.5887219E-03
 0.7568097
 2.078826
 1.756469
 1.211142
 0.8792306
 0.7275443
 0.6752405
 0.6996478
 0.7884408
 0.9079878
 1.022390
 1.082177
 1.056707
 0.9836840
 0.9080873
 0.8617991
 0.8548449

0.8702828
 0.9072432
 0.9416365
 0.9630473
 0.9677449
 0.9532503
 0.9322386
 0.9151314
 0.9050425
 0.9052690
 0.9152068
 0.8895902
 0.8258279
 0.7640446
 0.6936306
 0.6278193
 0.5649068
 0.5068772
 0.4539788
 0.4042353
 0.3585492
 0.3131343
 0.2691536
 0.2263308

solvent-polymer pair distribution fn

0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 0.0000000E+00
 2.1811711E-05
 1.0786743E-02
 5.5088684E-02
 7.7209286E-02
 7.2813645E-02
 6.4307712E-02
 5.8284722E-02
 5.5263482E-02
 5.3145394E-02
 5.4742254E-02
 5.8882568E-02
 6.4294547E-02
 7.0515536E-02
 7.4469082E-02
 7.4132964E-02
 7.3291682E-02
 7.0949607E-02
 6.9462322E-02
 6.8142422E-02
 6.9202892E-02
 7.0766941E-02
 7.2657324E-02
 7.3036157E-02
 7.4111938E-02
 7.3118165E-02
 7.3287509E-02
 7.3269241E-02
 7.2579868E-02
 7.2062433E-02
 7.1500123E-02

6.9777139E-02
6.5076441E-02
5.9696317E-02
5.4571364E-02
4.9249485E-02
4.4501562E-02
4.0203229E-02
3.6336299E-02
3.1954218E-02
2.8180245E-02
2.4566010E-02
2.1023838E-02
1.7613033E-02

polymer-polymer pair distribution fn

1.8430898E-02
3.8441584E-02
5.1800519E-02
5.2104648E-02
5.1243946E-02
4.1236591E-02
3.0737499E-02
1.8256402E-02
8.8841990E-03
1.7982034E-02
3.4812666E-02
2.4466202E-02
1.8462334E-02
1.3565400E-02
1.0363363E-02
8.5533662E-03
6.8941042E-03
5.1241195E-03
4.4614593E-03
3.7475612E-03
3.3441661E-03
2.7666315E-03
2.1063897E-03
1.5994973E-03
1.2362321E-03
8.5588970E-04
6.9454819E-04
5.3611316E-04
4.2806333E-04
3.6706554E-04
3.3414690E-04
2.5754989E-04
2.3031390E-04
1.9487373E-04
1.7754776E-04
1.3941404E-04
1.2726872E-04
9.5234442E-05
6.3826439E-05
3.9373881E-05
3.5206365E-05
1.7835227E-05
1.4284905E-05
7.7918266E-06
1.8613954E-06
4.7479384E-06
4.5459342E-06
0.0000000E+00

REFERENCES

1. Allen M. P. , D. J. Tildesley , 'Computer Simulation of Liquides' , Clarendon Press Oxford (1986).
2. Bishop M., M.H.Kalos ,and H.L. Friche , J. Chem. Phys. 70,1299(1979).
3. Bishop M., M.H.Kalos ,and H.L. Friche , J. Chem. Phys. 79,3500(1983)
4. Bruns W. abd R. Bansal , J.Chem. Phys. 74,2064 (1981).
5. Bruns W. abd R. Bansal , J.Chem. Phys. 75,5149 (1981).
6. Ermak D.L. and J.A. McCammon, J. Chem. Phys. 69,1352(1978).
- 7 . Fixman M., IUPAC 32nd International Symposium on Macromolecules Reprints (Society of Polymer Science, Kyoto,Japan.1988)
8. Gear C. W. 'Numerical Initial Value Problems In Ordinary Differential Equations' , Prentice-Hall Englewood Cliffs , NJ. (1971).
9. de Gennes P.G. Scaling Concepts in Polymer Theory (Cornell University, Ithaca, NY, 1979)
10. Kang P. S. , Y.L. Lee and T. Ree ,Bull. Korean Chm. Soc. 4.223,(1983)
11. Khalatur P. G. , Yu. G.Papulov and A. S. Pavlov, Mol. Phys. 58,887(1986).
12. Kremer K. , Macromolecules 1, 1632 (1983).
13. Krener K. , and Binder K. Comput Phys Rep 7,259(1988).
14. Luque J., J. Santamaria and J. J. Freire , J.Chem. Phys. 91,584(1989).
15. Madras N. and A. D. Sokal , J. Stat. Phys. 50,109 (1988).
16. Oh J., Y. S. Lee and T. Ree ,Bull. Korean Chm. Soc. 4.87,(1983).

17. Rapaport D. C. , J. Chem. Phys. 71,3299(1979).
18. Smit B. , A. van der Put , C. J. Peters , J. deSwaan Arons , and J. P. J. Michels , J. Chem. Phys. 88,3372(1988)
19. Valeur B. , J. P. Jarry and F. Geny J. Polymer Science 13,667(1975).
20. Verlet L. Phys. Rev. 159,98(1967).