1-1-1990

Monte Carlo simulations of flexible polymeric chains in solution/

Scott K. Starry
University of Massachusetts Amherst

MONTE CARLO SIMULATIONS OF FLEXIBLE POLYMERIC CHAINS IN SOLUTION

A Dissertation Presented

by

SCOTT K. STARRY

Submitted to the Graduate School of the
University of Massachusetts in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May, 1990

Polymer Science and Engineering

MONTE CARLO SIMULATIONS OF FLEXIBLE POLYMERIC CHAINS IN SOLUTION

A Dissertation Presented

by

SCOTT K. STARRY

Approved as to style and content by:

M. Muthukumar, Chairman of Committee

David A. Hoagland, Member

Peter A. Monson, Member

Edwin L. Thomas, Member

William J. MacKnight, Department Head
Polymer Science and Engineering Department

ACKNOWLEDGEMENTS

I would like to thank my committee members for their patience, criticism, and dedication. In particular, I thank Dr. David Hoagland for reminding me to compare carefully with experimental data, Dr. Peter Monson for ensuring that I said what I meant, Dr. Edwin L. Thomas for keeping me focused on doing science when the situation was bleak, and Dr. M. Muthukumar for providing the encouragement needed to finish this document.

The simulations presented here could not have been performed without the facilities at the Cornell National Supercomputer Facility in Ithaca, New York and the John von Neumann Computing Center in Princeton, New Jersey. To the consultants and administrators of both I extend my appreciation for your willingness to provide "workarounds" and solve problems. The decommissioning of the facilities at JvNCC is regretable.

Many people have influenced my life while in graduate school. Among those who helped me along the way, I would like to thank Dumont Jones, for teaching me statistical mechanics and becoming a good friend; Jyh-Shyong Ho, for teaching me vectorization techniques and offering a simpler way to live; Varadarajan Ramaswami, for keeping me sane while my X-ray experiments failed and sharing my interests; Herr Dr. Harald Modler, for providing scientific and cultural insight; Mahesh Kotnis, for being concerned and thoughtful; and Kleanthes G. Koniaris, for making science fun again. Jean-Philippe Autran, Cristina Urdaneta, and Karen Winey deserve particular mention for their valued friendship and scientific aptitude.

Most importantly, I would like to thank my family for their support, both moral and financial. To my parents, I appreciate your understanding for so many broken appointments while pursuing my degrees, and your willingness to try to understand my outlook towards the world. To my grandmother, I offer my respect for demonstrating how to live life well and my deepest gratitude for always opening your heart to me.

ABSTRACT

MONTE CARLO SIMULATIONS

OF FLEXIBLE POLYMERIC CHAINS

IN SOLUTION

MAY, 1990

SCOTT K. STARRY, B.A.A.S. (PHILOSOPHY), UNIVERSITY OF DELAWARE

B.Ch.E., UNIVERSITY OF DELAWARE

M.S., UNIVERSITY OF MASSACHUSETTS

Ph.D., UNIVERSITY OF MASSACHUSETTS

Directed by: Professor M. Muthukumar

Dilute and semidilute concentrations of polymeric solutions are often used to elucidate fundamental characteristics of the molecules and specific interactions with a solvent. However, the theories that describe these regions of concentration have not been able to predict the values and scaling relationships observed in such experiments. This thesis uses both lattice based and continuous space Monte Carlo computer simulations to investigate these regions of concentration in three ways: (1) Evaluate the lattice and continuous space models for their ability to predict experimental scaling relationships; (2) attempt to provide values from the equation of state that may be compared with experimental data, and (3) calculate the scattering function for an amorphous system to investigate local organization and to provide comparisons with solution X-ray and neutron scattering data.

Results from the lattice simulations were found to follow the trends predicted by mean field lattice theories at dilute and semidilute concentrations. This means that the approximations made in obtaining numerical results from the theories are acceptable. However, as mean field approximations are known to be invalid at these concentrations, lattice models in general are inapplicable to dilute and semidilute concentrations.

The continuous space models investigated here approximate the trends from experimental data in dilute and semidilute concentrations better than lattice models, but still do not provide values from the equation of state that agree with experiment. The investigation of higher concentrations, greater than about 5% bead volume, was performed in an isobaric-isothermal ensemble since the pair correlation function of the canonical ensemble is no longer sufficient at such concentrations. While the development of simulations using the isobaric-isothermal ensemble is in its nascent state, these simulations appear to offer a powerful apparatus by which to investigate the thermodynamic properties of a system.

A new method was developed by which to calculate the scattering function $S(q)$ efficiently from a large system of amorphous chains. Hence, the results for models representing polymeric chains in solution may be compared with experimentally determined data. The agreement of results from these simulations with X-ray and neutron scattering data is encouraging.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Studies of dilute and semidilute solutions are important to the understanding of intramolecular and intermolecular interactions. Effects such as collapse transitions, screening effects, and global organization of the fluid all have their bases in solutions of low to intermediate concentration. Because of this, a number of analytical theories and numerous experiments have been performed to elucidate the nature of the microscopic influences that affect the properties observed in dilute and semidilute solutions.

Computer simulations have the unique abilities of providing direct control of the microscopic influences on a system and of permitting direct observation of the states of organization within a fluid. As such, computer simulations serve as a bridge between theoretical predictions and experimental observations, testing the validity of theoretical formulations and directly addressing the molecular origins of observed phenomena. (*cf.* Fig. 1.1.) In addition, as simulations are derived from the Hamiltonian for the proposed model, the numerical results may be compared with the values obtained from experiments to determine the soundness of the model itself. Providing tests for the applicability of a model, the validity of assumptions, and the consistency of observations make simulations a powerful tool.

To investigate the thermodynamic and structural properties of dilute and semidilute solutions, Monte Carlo simulations are performed here on lattices and in continuous space using semiflexible chains in high temperature, athermal systems. Thermodynamic quantities are investigated through statistical mechanical predictions for the second virial coefficient, osmotic pressure, and compressibility. Structural information is provided through bead-bead and chain-chain radial

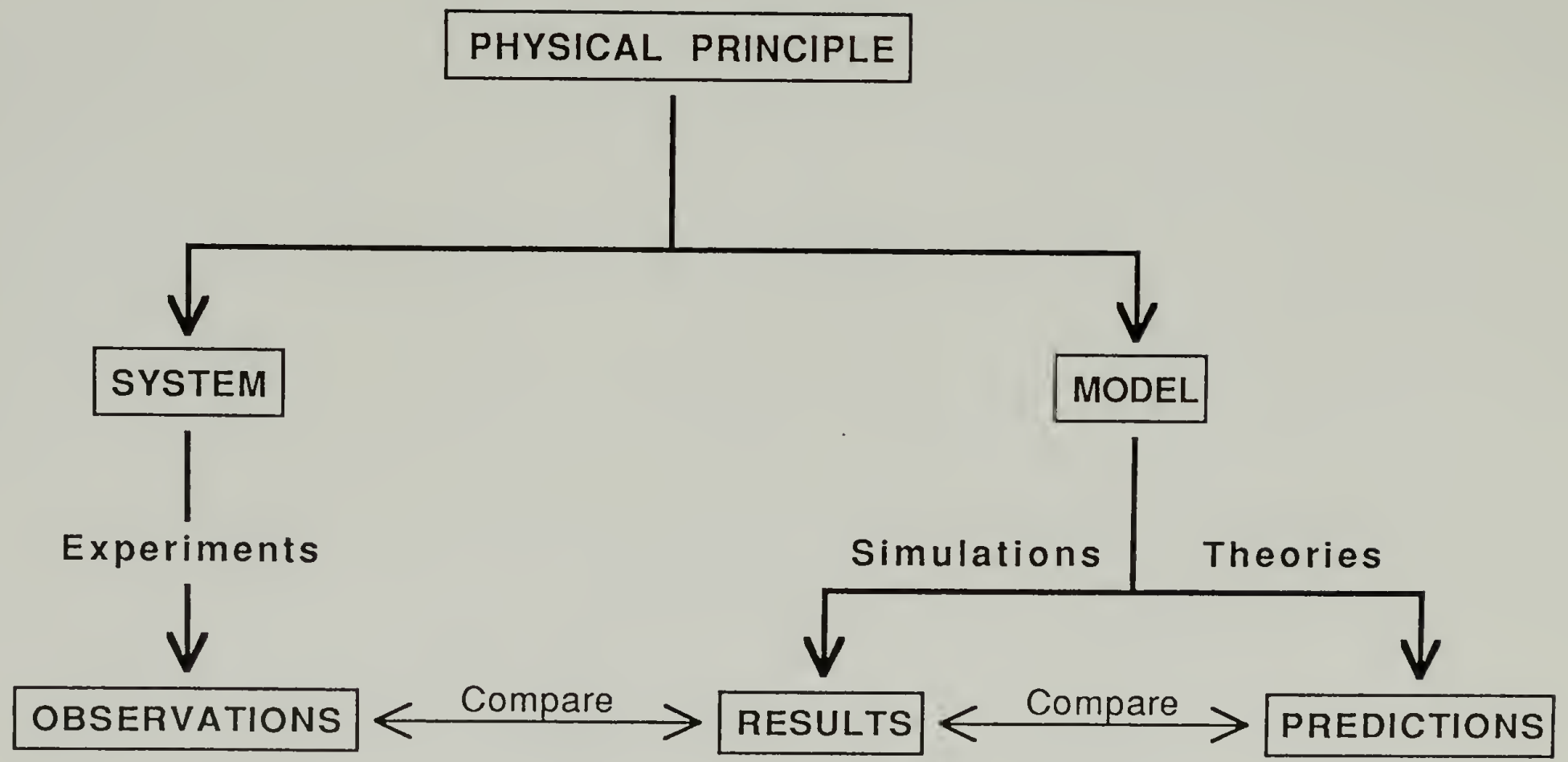

Figure 1.1
Diagram of the methodological differences in investigating a physical principle. Note that the numerical results from simulations may be compared directly with experimental data as well as testing the predictions from theoretical models.

distribution functions, geometric quantities (*e.g.*, the mean squared radius of gyration for a chain), and the scattering function. Predictions for scaling relationships from theories and correlations with power laws from experiments are both presented to evaluate the validity of the former and to determine the applicability of simulations to provide the latter.

This chapter will provide a short background of the relevant literature, for both lattice and continuous space theories as well as for simulations, that pertains to the simulations performed here. In closing this chapter, the insights gleaned from the simulations performed here and the status of the field in general are presented.

1.1 Developments of Lattice Theories.

Some of the earliest theoretical work done on polymers was that of Meyer [1939] in which he suggested that the entropy of mixing polymers with small solvent molecules could be calculated by constructing a lattice on which the monomeric units for the polymer would occupy the same type of site as a solvent molecule, thereby introducing the assumption that a monomeric unit was of the same size as the solvent molecule. The entropy for the system is calculated from determining the number of configurations available to the system, and the enthalpy is determined by introducing van der Waals interaction energies between unconnected, nearest neighbors [Freed and Bawendi, 1989]. Flory [1941] and Huggins [1941] both expanded upon Meyer's scheme for counting by using mean field approximations. In Flory's version, the sites available for occupancy were chosen on a probabilistic basis [Flory, 1953, Chp. 12]. This meant that sites were not correlated to one another as the placement of the initial "monomer" at a site was chosen randomly. The difference between Flory's version, which has become the *de facto* standard, and Huggins' version is that the latter included the site coordination number of the lattice. This meant that connectivity appeared both

through the placement of a chain on the lattice and from an effective interaction energy between adjacent, unconnected segments. While Huggins' version actually agrees with simulation results better, Flory's version (now called "Flory-Huggins theory") with only one fitting parameter χ was more readily accepted [Freed and Bawendi, 1989].

There are conceptual problems with this simple lattice model, however. First, there is the original assumption from Meyer that a single lattice site size may be used to characterize both the monomeric units and the solvent molecules. Flory [1953, Chp. 12] considered using the same lattice for both species, especially for all concentrations, to be a fundamental flaw in the theory. Second, there is the mean field assumption itself in which the counting sequence presumes that the monomeric units are evenly distributed about the lattice. In a dilute solution where chains are isolated from one another with large regions of unoccupied sites, or conversely, many sites occupied by solvent molecules, the mean field assumption will be poor [Flory, 1953, Chp. 12].

There are experimental differences from the theoretical predictions as well. The parameter χ is predicted to be dependent upon the inverse of the temperature and to be independent of composition; experimental data have been fit to χ providing "χ"s that are dependent upon concentration, pressure, and molecular weight [Wolf, 1975]. This means that a simple, one parameter theory (such as Flory's original, mean field approach) is inadequate to describe the physics involved in these systems.

These difficulties led people to study the problem using three different methods. Flory [1970] developed his equation of state theory for the statistical thermodynamics of polymeric systems (*i.e.*, he attacked the first theoretical problem) while others (*e.g.*, Koningsveld and Kleitjens [1971]) improved the methods of counting the available configurations on a lattice (*i.e.*, they attacked the second theoretical problem). Experimenters kept trying, wrongly, to develop correlations to place a phenomenological significance on the χ parameter. A third, and then new, group decided to simulate the insertion method of Flory, thereby addressing

the second theoretical problem, by performing the lattice calculation on a computer. In this way, people sought to determine (1) if a single sized lattice was as serious a problem as Flory believed and (2) if Flory's counting scheme of randomly selecting sites for monomer insertion was valid. The former point was often lost in the search.

1.2 Simulations on a Lattice.

Freed and Bawendi [1989], who are keenly interested in determining exactly why the lattice approximations perform so poorly in many cases, provide a complete overview of developments using lattice techniques. However, two points from their paper require comment. First, Freed and Bawendi are correct in stating that, due to the construct of the lattice, the most common investigations are either of dilute solutions or of melts. The reasons for this are that dilute solutions are computationally tractable, while melts are represented well by the mean field approximation. Second, Freed and Bawendi's statement that "lattice model Monte Carlo simulations ... [are able to] describe all of the general equilibrium phenomena that are observed for polymers in dilute and semidilute solutions" is of limited validity as the simulations mentioned are only for single chains in the original literature.

What lattice simulations do is to provide results that agree well with some lattice theories. (This also indicates the counting mechanism is not the most important flaw in lattice simulations.) The lack of agreement among results from lattice simulations with experimentally observed trends, particularly for thermodynamic quantities, indicates that the bases of various lattice *models* are flawed, and this lack of agreement to experiment has gone unnoticed in the excellent agreement that lattice simulations have with lattice theories. (See §1.5.)

Most of the investigations for dilute and semidilute solutions on lattices deal with insertion probabilities, essentially to test the simulation results against the

predictions of some lattice theory, usually Flory's or Huggins'. One of the earliest studies is from Bellemans and de Vos [1973] in which they find their results to agree extremely well for a number of chain lengths with Huggins' predictions. Okamoto [1976] as well as Okamoto and Bellemans [1979] show poor agreement with Flory's predictions, but good agreement with Huggins' predictions. However, while the values differ from those predicted by Flory, the trend for the insertion probabilities is quite similar. Kolinski [1984] tried a face-centered cubic lattice to see if this had an effect on the combinatorial entropy calculated; it did in that the values obtained were significantly different from Flory's predictions, but again the trends remained the same. Finally, Dickman and Hall [1986b], using a new approximation for the equation of state, and Dickman [1987], using a new lattice technique, found their results to follow the predictions of Huggins well, though Freed and Bawendi's own calculations fit Dickman and Hall's results even better. Simply stated, Huggins' method of counting seems to be more appropriate than Flory's on a lattice.

Two points were missed in all of these studies as attention was focused on the concurrence of lattice simulations with lattice theory. First, no comparisons were made to experimental data. Flory's predictions for the osmotic pressures of dilute solutions were too low, but Huggins' predictions were lower. Thus, the trend of these simulations was opposite to what was required to agree with experimental data and indicated that lattice models themselves were flawed. Second, Okamoto [1976] ignored his own continuous space simulation data that did give the appropriate trend for the osmotic pressure by stating that the osmotic pressures determined from the lattice simulations were "... similar to the results of the study on the systems in a three-dimensional space." He made this statement even though his data for the highest concentration he studied (40.%) differed by a factor of 1.7 between the two methods.

1.3 Developments of Continuous Space Theories.

de Gennes reviews conceptually the most important topics in solution theory in Chapter 3 of his book [1979] so only semidilute scaling relationships will be presented here. des Cloizeaux [1975] utilized a technique introduced by de Gennes [1972] for studying single chains in solution and extended this treatment to semidilute concentrations to derive the dependence of the osmotic pressure upon the molecular weight of the polymers and on the concentration of the solution. From this work came des Clozieaux's law

$$\pi \propto \phi^{9/4}, \tag{1-1}$$

where π is the osmotic pressure, and ϕ is the monomeric concentration for the solution. This "law" seems to be confirmed by experiments performed by Cotton, *et al.* [1973], and presented in Candau, Strazielle, and Benoit [1976]. (See Figure 1.2.)

From neutron scattering experiments, Daoud, *et al.* [1975] found that the squared radius of gyration of a polymer R_g^2 scaled with concentration ϕ as

$$R_g^2 \propto \phi^{-1/4}. \tag{1-2}$$

(See Figure 1.3.) This relation is developed in Daoud's paper (for the squared end-to-end distance R_e^2) both using screening length arguments as presented by Edwards [1965; 1966] and using the field approach of des Cloizeaux [1975]. This was redone conceptually by de Gennes in his book [1979] with the famous "blob" argument.

Considering enumeration techniques, the work of Croxton [1979] is important as he presents and develops the screened convolution approximation and notes that the hard sphere Percus-Yevick equation adapted for polymeric solutions should provide an upper bound for the compressibility of a "pearl necklace" model. Honnell and Hall [1989] reworked a model of Dickman and Hall [1986c]

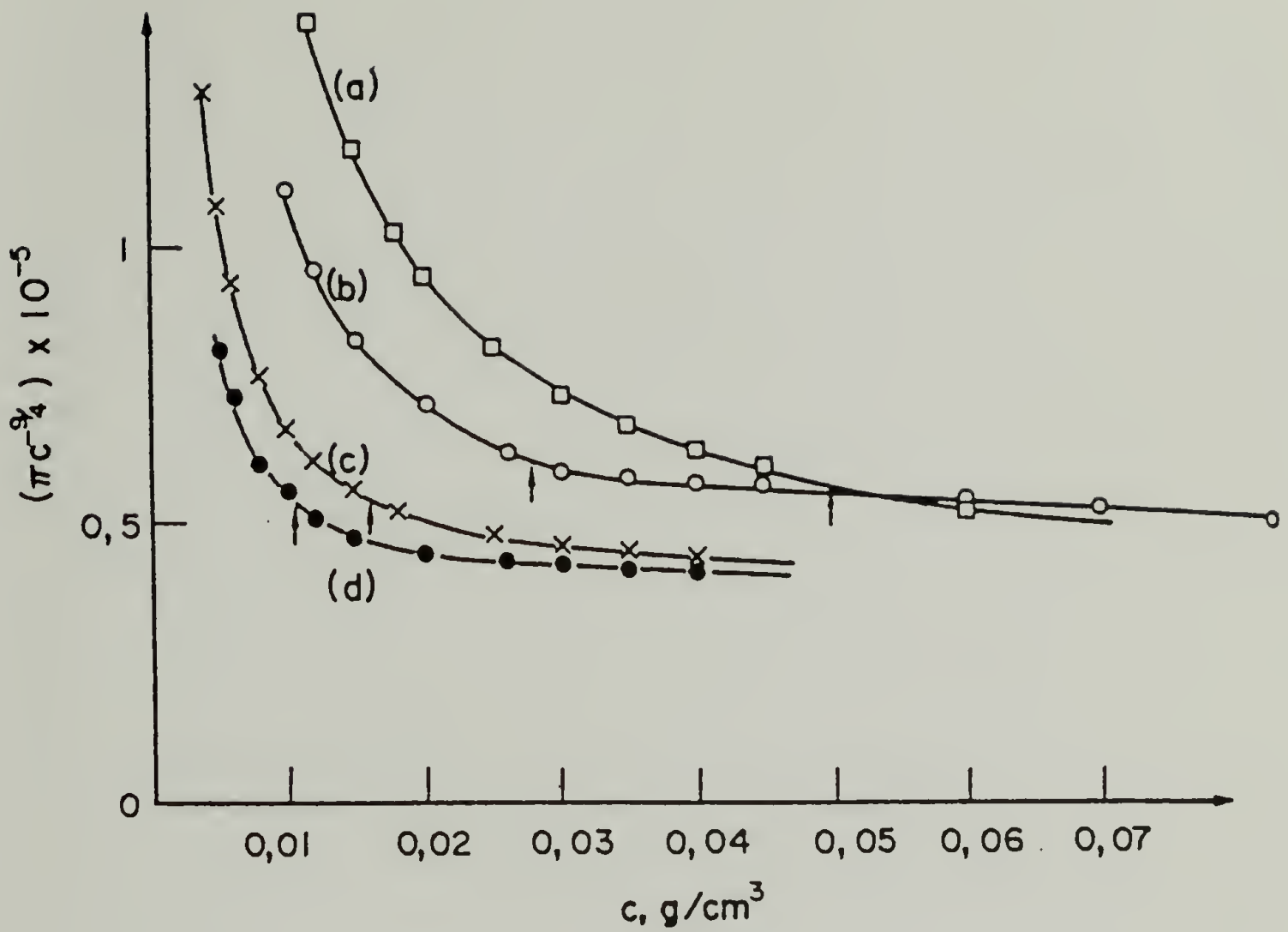

Figure 1.2

Dependence of the quantities $(\pi c^{-9/4})$ and (πc^{-2}) for linear polystyrene as a function of concentration in a good solvent. The osmotic pressure π has units of g/cm^2 while the concentration has units of g/cm^3. (a) Solvent: Toluene; $M_n = 72,000$; $T = 37°$ C. (b) Solvent: Benzene; $M_n = 155,000$; $T = 30°$ C. (c) Solvent: Toluene; $M_n = 312,000$; $T = 30°$ C. (d) Solvent: Toluene; $M_n = 540,000$; $T = 30°$ C. From Candau, Strazielle, and Benoit [1973], Figure 1.

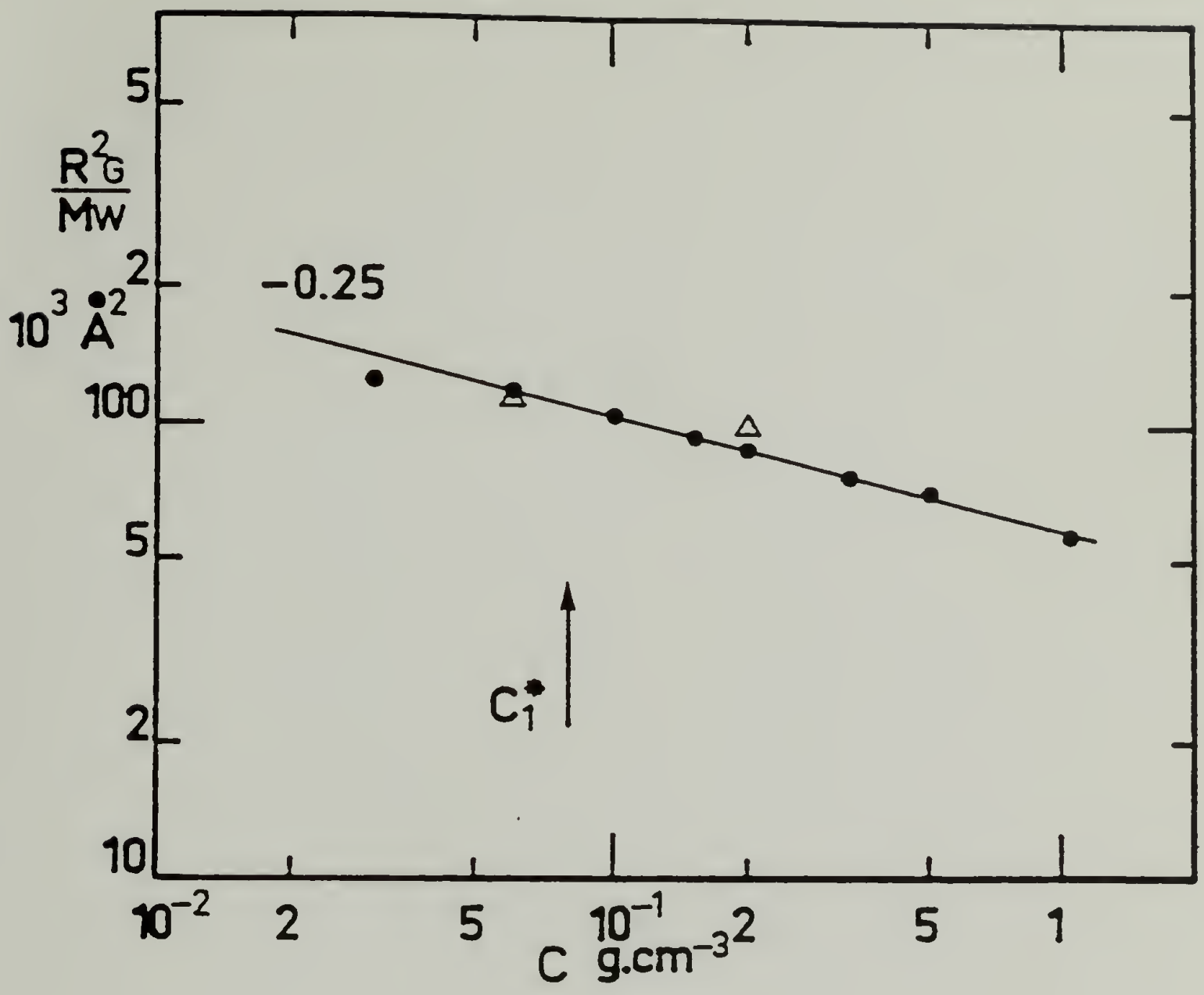

Figure 1.3

Log-log plot of $R_g^2 M_w^{-1}$ as a function of concentration for linear polystyrene in carbon disulfide. (•) $M_w = 114,000$; (△) $M_w = 500,000$. From Daoud, *et al.* [1975], Figure 7.

for tangent hard spheres and obtained an approximation to the equation of state that provided a significant improvement over Dickman and Hall's lattice results. Wertheim [1987] presented his second-order thermodynamic perturbation theory (TPT2) that was developed to investigate mechanisms of polymerization and that provides numerical results similar to those obtained by Honnell and Hall.

A different approach using extrapolation formulas was developed by Muthukumar and Edwards [1982] and was extended by Muthukumar [1986] using monomer density fluctuations and three-body interactions. In this later paper, Muthukumar is able to provide an equation for the free energy that is applicable over the whole range of concentrations in solution, though the difficulty of defining three-body interaction energies remains.

1.4 Simulations in Continuous Space.

Due to practical considerations, primarily the speed of computation available, the early continuous space (or "off-lattice") studies only considered the properties for a single chain; recently, researchers have tried to investigate the melt state, primarily through numerical means. As such, the literature for dilute and semidilute solutions remains inadequate with much of the work having been performed in the 1970's.

Mentioned previously was Okamoto's work [1976] in which he presented some results for the osmotic pressure of a continuous space system, but in which he ignored the differences from his lattice results. Nonetheless, this seems to be the first paper in which continuous space results for the osmotic pressure are reported. The trend for the pressure is in the correct direction with the values reported being significantly greater than the estimation provided by Flory's theory.

Curro [1976; 1980] performed simulations to verify an equation of state that he had developed specifically for continuous space simulations using the rotational isomeric model. He obtained information on the mean squared end-to-end

distance $\langle R_e^2 \rangle$ and the compressibility Z for his systems. He compared his results to the predictions from a virial expansion and to Flory-Huggins theory. Curro's simulation results for the compressibility are larger than the values predicted by Flory-Huggins theory, which means that these rotational isomeric calculations are a better approximation. However, the compressibilities obtained appear to be too small when compared to the values from his reference interaction site model (RISM) [Schweizer and Curro, 1988b] and when compared to the compressibilities obtained in this thesis.

Curro also brought forth, for the first time in simulations, the difficulty of defining the concentration at which the semidilute concentration is reached. His approach was to plot the mean squared end-to-end distance *versus* the bead concentration with the latter on a logarithmic abscissa. The point at which the curve suddenly turned downward ($\langle R_e^2 \rangle$ decreasing with increasing concentration) was considered to indicate the onset of the semidilute region. The possibility exists that this method simply displays an artifact of presentation. The usual linear or log-log plots of this information do not show any deviation at the same concentration from what would be predicted by dilute solution theories. While the compressibility does begin to increase rapidly at approximately the same concentration, without an analytical development for such a presentation, caution must be exercised.

Structure factors had been limited to single chain studies used to elucidate the conformation of a freely jointed chain under the influence of intramolecular forces [Baumgärtner, 1980] and to determine the dependence of the structure factor on the wavenumber [Baumgärtner and Binder, 1979]. Vacatello, *et al.* [1980], calculated the intensity of scattering from a system of chains that were constructed of 30 segments that had the segment length and the valance angle fixed (the latter at 112°) with the torsion angles and interchain distances being variable. Here, the radial distribution function was carefully determined and then integrated to provide the scattering intensity. This method required that a separate numerical integration be performed for each wavenumber and would

be expected to contain a large error, though the latter was not reported. The scattering results developed for this thesis are believed to be the first in which the scattering function is *directly* calculated from the definition of the scattering factor for a system of chains.

1.5 Summary of the Thesis.

First, a lattice model was tested to determine its applicability to replicating the experimentally observed trends in dilute and semidilute solutions and providing the scaling relationships that had been derived analytically. We found that, in general, lattice simulations follow the trends predicted by mean field lattice theories throughout the dilute and semidilute regions, even though the mean field approximations are not appropriate at these concentrations. Hence, lattice simulations and lattice models should only be applied to the melt or bulk state where the mean field approximation is valid.

Second, the scattering function for a system of chains may now be calculated efficiently from the function's definition. This new method is believed to be applicable in any type of ensemble used for Monte Carlo simulations. This ability is a significant step towards performing calculations on a model believed to represent an amorphous, or weakly organized, polymeric sample, and this calculation permits comparison among model structures and experimentally determined data. The agreement obtained using both X-ray and neutron scattering results is encouraging.

Third, difficulties were discovered in relating relating the simulation results to experimental data for predictive purposes. There does not seem to be a one-to-one correspondence between experimental concentrations, *e.g.*, specific mass, and concentrations in the simulations, *e.g.*, bead volume fraction. For example, a single sample's overlap concentration corresponded to three different bead volume

fractions for relating scattering factors, osmotic pressures, and mean squared end-to-end distances determined from the simulations to the experimental data. This lack of a direct correspondence between the experimental concentrations and the simulations' volume fractions complicates the interpretation of the results and indicates that the size relationships within the model chosen greatly affect the correlation to experiment.

Fourth, the continuous space simulations performed here approximate the experimental trends observed and tend towards some of the scaling relationships predicted from theory. Two different statistical mechanical ensembles were studied. Canonical ensemble simulations, which are computationally simple, provide thermodynamic data for concentrations of less than 5% bead volume for a "pearl necklace" model. The concentration is limited by the increasing importance of three-body interactions as the bead volume increases such that the pair correlation function, on which many of the canonical ensemble's relations are based, is no longer an adequate description of the interactions involved. Isobaric-isothermal ensemble simulations provide information at concentrations higher than 5% bead volume. While the development of simulations using this ensemble is in its nascent state, the isobaric-isothermal simulations appear to offer a powerful apparatus to investigate the thermodynamic properties of a system.

CHAPTER 2

SIMULATIONS OF SELF-AVOIDING CHAINS ON A SIMPLE CUBIC LATTICE

Meyer's idea of using lattices to simplify the analysis of polyatomic systems [1940] was used by both Huggins [1941] and Flory [1941] to model polymeric systems. These later modifications become the *de facto* standards of polymeric theory, primarily due to the simplicity of the approaches. In the 1970s, a number of people (*e.g.*, Baumgärtner [1987 (review); with Binder, 1979], Bellemens [with de Vos, 1973; with Janssens, 1974], Okamoto [1976; with Bellemens, 1979]) began to use computer simulations to check the significant, and lesser, predictions of the lattice models by performing computer "experiments" on lattices. Much of this work encompasses insertion probabilities, the method by which Flory [1953, Chp. 12] calculated his entropies and enthalpies of interaction.

Along with these fundamental thermodynamic properties, other quantities, such as the free energy and osmotic pressure [Bellemans and de Vos, 1973; Okamoto, 1976] could be derived from the results obtained using insertion probabilities. The difficulties, however, were that such relations required numerically integrating the insertion probabilities, thereby introducing significant error and that simulations determining insertion probabilities are not very efficient [Bellemans and Janssens, 1974]. This latter difficulty often limited the number of samples included for the statistics, and almost always limited the simulations to the study of a single chain [Baumgärtner, 1987]. Thus, attempts at defining the ultimate properties for a chain of infinite length, and its asymptotic behavior, dominated the types of simulations performed.

As computational speed and algorithmic expertise grew in the early 1980s, lattice simulations gained greatly in efficiency, and a number of people became involved in more complex types of simulations in which, for example, coil-globular collapse transitions [Webman, Lebowitz, and Kalos, 1981], order-disorder phase

transitions [Baumgartner and Yoon, 1983], and melt-state entanglement studies [Kolinski, Skolnick, and Yaris, 1986a–d] were all attempted. At the same time, while the simulations still served the purpose of testing models, much more emphasis was being placed on using the simulations as an independent method to describe or predict macroscopic behavior of real, polymeric systems.

One of these attempts, in this case to extract all of the thermodynamic information from a canonical ensemble, was proposed by Dickman [1987]. With Hall, Dickman had worked extensively on equations of state [Dickman and Hall, 1986b and 1986c] and in the process had devised a method by which the osmotic pressure could be obtained from a lattice simulation in a reasonable number of samples. This is the method by which we sought to begin our investigation into the aspects of semiflexible chains as they influence packing, orientation, and measurable thermodynamic parameters.

What we propose to show, however, is that lattice models in general do *not* appear to be able to predict the appropriate semidilute scaling relationships that are observed experimentally. Lattice simulations appear to have an inherent tendency towards following mean field predictions at all concentrations and are therefore inappropriate for investigating thermodynamic quantities at semidilute concentrations.

2.1 Basis for the Lattice Simulations.

The algorithm, closely based on one proposed by Dickman [1987] for the equation of state, is as follows: Consider a canonical ensemble, in which the number of particles in the system N, the volume V, and the temperature T are all constant. The volume is constructed of a simple cubic lattice of size $V = H^3$, that is, a lattice comprised of H layers, with the center of the lattice set at the Cartesian point $(x, y, z) = (0, 0, 0)$ to utilize symmetry in the model. (Note that this restricts the values of H to odd integers.) The reference lattice has the

dimensions $\lceil -H/2 \rceil \leq x_i \leq \lfloor +H/2 \rfloor$, where x_i represents any of the coordinates x, y, or z. While the boundaries in the x- and y-directions are periodic to simulate a large, if not infinite, system, walls are introduced at positions $z = \lceil -H/2 \rceil - 1 = \lfloor -H/2 \rfloor$ and $z = \lfloor +H/2 \rfloor + 1 = \lceil +H/2 \rceil$.

The walls are introduced because walls represent an infinite potential through which the beads cannot pass. This causes the packing fraction to tend toward zero and uncorrelate (or "decouple") the beads' motion from one another [Percus, 1976]. When particles are not correlated in continuous space, the ideal gas relation holds exactly at the wall, and one is able to obtain the osmotic pressure *via* the relation $(P/\mathrm{k}_B T) = \rho_w = \pi^*$ [Percus, 1976; Chandler, 1987, §3.6], where ρ_w is the density of the beads at the wall, π^* is the modified osmotic pressure, P is the thermodynamic pressure for the system, k_B is Boltzmann's constant, and T is the temperature of the system. Hence, in a *continuous space* simulation, by measuring the density of beads at a wall, one may obtain the osmotic pressure.

Obtaining the pressure in a system from the particles' density at a wall has been used in previous studies [Liu, Kalos, and Chester, 1974; Percus, 1976; Snook and Henderson, 1978; Sullivan, Levesque, and Weis, 1980] by counting the number of "contacts," that is, the number of approaches within some arbitrarily defined distance ϵ to a wall. Using this method for polymeric systems has posed a more difficult problem due to the connectivity of the chains, which makes defining the distance ϵ more difficult, and the painfully slow approach to a statistically significant result for polymeric models in general [Baumgärtner, 1987]. Even the simple hard sphere fluids need several million samples to provide useable values for determining the density at the wall [Snook and Henderson, 1978; Sullivan, Levesque, and Weis, 1980]; the time to perform such a study on a polymeric system would be prohibitive.

However, on a lattice many of the problems caused by a polymeric system can be mitigated. First, movement on a lattice is more readily accomplished than in continuous space so that the need for large numbers of samples is less restrictive. Second, the meaning of a "contact" is easily defined as the presence

of a bead on the layer of the lattice adjacent to the layer containing a wall; this eliminates arbitrarily defining a distance ϵ and the complications introduced by the connectivity of the chain. Unfortunately, using the ideal gas equation of state for determining the osmotic pressure is appropriate only for molecular potentials in isotropic, continuous space fluids, and not for lattices [Dickman, 1987].

By noting the form of the osmotic pressure π^* for the canonical ensemble

$$\pi^* \equiv \frac{P}{\mathrm{k}_B T} = \left(\frac{\partial \ln Q}{\partial V} \right)_{N,T} \tag{2–1}$$

[McQuarrie, 1976, Chp. 2], Dickman modified the definition of the partition function for a canonical ensemble Q so that it could be used on a lattice (Eqn. 2–2) and then generalized the partition function by including a repulsive energy λ (*cf.* Eqn. 2–3) to simulate the change in volume indicated in Equation 2–1. Thus,

$$Q(N,V,T) = Q(n,N_c,H,L,T) = (N_c!)^{-1} \sum_{i=1}^{nN_c} e^{-\beta U(r_i)} \tag{2–2}$$

is converted to

$$Q'(n,N_c,H,L,T,\lambda) = (N_c!)^{-1} \sum_{i=1}^{nN_c} e^{-\beta U(r_i)} \lambda^{N_w} \tag{2–3}$$

where

H	$\equiv$	The height of the lattice;
k_B	$\equiv$	Boltzmann's constant;
L	$\equiv$	The length of the cubic lattice in the other two directions $= H$;
n	$\equiv$	The number of beads per chain;
N	$\equiv$	The number of beads in the system $= nN_c$;
N_c	$\equiv$	The number of chains in the system;
N_w	$\equiv$	The number of beads on the layer next to the wall;
P	$\equiv$	The thermodynamic pressure;
$Q(N,V,T)$	$\equiv$	The canonical ensemble's partition function;

T	$\equiv$	The temperature of the system;
$U(r_i)$	$\equiv$	The potential energy of the system;
V	$\equiv$	The volume of the system $= H^3 = HL^2$;
r_i	$\equiv$	The composite x_i, y_i, z_i coordinates for the bead i in the system;
β	$\equiv$	The inverse of the thermal energy for the system $= (\mathrm{k}_B T)^{-1}$;
λ	$\equiv$	The repulsive potential from the wall at $z = \lfloor H/2 \rfloor + 1$.

So that, when $\lambda = 1$, $Q' = Q$, and, when $\lambda = 0$, $Q' = 0$.

As λ is raised to the power N_w, this repulsive potential only affects the beads on the top layer of the lattice. This construct is considered to be equivalent to changing the volume of the system. Equation 2–1 can then be modified through the following relations:

$$\begin{aligned}
\pi^* &= \frac{\partial}{\partial V} \left[\ln Q(n, N_c, H, L, T)\right] \\
&= L^{-2} \frac{\partial}{\partial H} \left[\ln Q(n, N_c, H, L, T)\right] \\
&= L^{-2} \left[\ln Q(n, N_c, H, L, T) - \ln Q(n, N_c, H-1, L, T)\right] \\
&= L^{-2} \left[\ln Q'(n, N_c, H, L, T, 1) - \ln Q'(n, N_c, H, L, T, 0)\right].
\end{aligned} \tag{2–4}$$

By the fundamental theorem of calculus, Equation 2–4 becomes

$$\pi^* = L^{-2} \int_0^1 \left(\frac{N_w}{\lambda}\right) d\lambda = \int_0^1 \left(\frac{\rho_w(\lambda)}{\lambda}\right) d\lambda, \tag{2–5}$$

which provides the compressibility Z for the system as

$$Z \equiv \frac{n\pi^*}{\phi_B} = \frac{n}{\phi_B} \int_0^1 \left(\frac{\rho_w(\lambda)}{\lambda}\right) d\lambda. \tag{2–6}$$

Two notes should be made about the form of the partition function given by Equation 2–3. The first discusses the normalization factor $(N_c!)^{-1}$, and the second describes the form of the potential energy for the system $U(r_i)$.

As the osmotic pressure is a colligative property, the objects that are influencing the pressure must be clearly delineated. As is well known from experiments [Billmeyer, 1984, Chp. 8], the osmotic pressure for a polymeric system depends on the concentration of *polymers* present, not the number of monomeric repeat units. Since computer simulations often have the concentration expressed in the units of "site fraction," "bead density," or, misleadingly, "monomer concentration" (*cf.* §2.2), the correct normalization factor must be ensured. The factor $(N_c!)^{-1}$ denotes that the osmotic pressure depends on the concentration of chains present.

Properly, the normalization factor in the partition function is used to account for degeneracies in a system so that a given state is not weighted too heavily due to the counting procedure used. The $(N_c!)^{-1}$ term of the canonical partition function recognizes that the conformations of the chains in a system are what describe that system's energy. The usual normalization factor, $(N!)^{-1}$, implies that the beads' positions inherently define the state of the system. This is not true here due to the connectivity of the chains, which causes the beads to be distinguishable. For example, two different states can exist in which the chains have different conformations, but where the actual sites occupied are the same. Since the chains themselves are indistinguishable, the term $(N_c!)^{-1}$ is the correct one to use.

The potential energy term $U(r_i)$ of the canonical partition function is the energy of interaction for each bead within a chain (*cf.* Eqns. 2–2 and 2–3) and may be represented as $U(r_1, r_2, r_3, \ldots, r_{nN_c})$ to emphasize that it is dependent upon the position of each of the n beads for the N_c chains present. In its form for this simulation there are three types of interactions: (1) an excluded volume term for the sites such that no two "beads" can occupy the same lattice site, (2) hard, impenetrable walls are placed at $z = \lfloor -H/2 \rfloor$ and $z = \lceil +H/2 \rceil$, and (3) a "bend energy" τ that permits the study of semiflexible chains on the lattice. (Dickman's simulations [1987] only have the first two types of interaction.) By including this last interaction, the more contorted a chain's shape is, the more

energy it contains; conversely, a completely straight chain would be in its lowest energy state, with respect to the bend energy of that chain. The potential energy on the lattice for these simulations is then represented by

$$\beta U(r_i) = \left(\begin{cases} 0, & r_i \neq r_j; \\ \infty, & r_i = r_j. \end{cases} \right) + \left(\begin{cases} \infty, & z_i < \lceil -H/2 \rceil; \\ 0, & \lceil -H/2 \rceil \leq z_i \leq \lfloor +H/2 \rfloor; \\ \infty, & \lfloor +H/2 \rfloor < z_i. \end{cases} \right) + (N_b \tau), \tag{2-7}$$

where τ is the dimensionless energy *per* bend in a chain and N_b is the total number of bends in the system.[1] (One should note that, while no attractive potentials and only hard core repulsive potentials have been included here, modifications to the potential energy would be easy to include.)

2.2 Correspondence of the Simulations' "Chains" to Real Polymers.[2]

The correspondence between the chains utilized in these simulations, or simulations generally, and the polymers whose properties we desire to investigate is often misunderstood. These simulations are designed to investigate global aspects of a chain's behavior in which the detailed chemical nature of a macromolecule is relatively unimportant. One goal is to determine the least complicated way by which this investigation may be accomplished. Unfortunately, the terminology used in the literature often confuses the issue. This section should clarify what is actually being represented.

---

[1] The dimensionless bend energy τ is related to the temperature of the system by the equation $\tau = k_B T/\epsilon_b$, where ϵ_b is the parameterized energy for a bend in a chain. Thus, a system with a large value of τ, which will primarily have highly contorted chains, may be considered either to have a high temperature or to contain very flexible chains.

[2] In deference to standard usage in Polymer Science as well as to several hundred years of physics and mechanics, the terms "conformation" and "configuration" will be used in the following way:

"Conformation" represents the arrangement or shape of a single chain. "Configuration" represents the arrangement of a system of chains. The terms "chain's conformation" and "system's configuration," while redundant, will be used in certain cases for clarity.

The "chains" in these simulations consist of "beads" and "links" in the manner of the "pearl necklace" or "bead-rod" model [Baumgärtner, 1987]. The "links" are rigid, infinitely thin connectors (actually, direction vectors) between bead centers and provide the connectivity of the chain. On a lattice, the links lie along the lattice's segments; in continuous space, the links are freely jointed about a bead's center. The "beads" serve to represent the excluded volume of a polymer. On a lattice, the beads consist of linked, occupied sites; in continuous space, the beads are hard spheres. Essentially any desired aspect ratio for a polymer can be modeled in free space by varying the ratio of the length of the link to the diameter of a bead, with the restriction that $l/D \geq 1$ where l is the length of a link and D is the diameter of a bead.

A different set of terminology than that used in the previous paragraph is often applied to simulations. In many cases this alternate set of terms is misleading. "Chains" are often referred to as "polymers," "beads" as "monomers," and "links" as "bonds."[3] These alternate terms developed from analogies that in most cases do not exist. For example, while chains are meant to model polymers, chains normally consist of tens of beads, a bead being the basic repeat unit,[4] not the several hundred to thousands that would be needed to replicate the organization of a polymer. Similarly, beads, being the basic repeat unit, are conceptually related to the idea of monomeric building blocks, but beads often represent hundreds of chemical, monomeric units so that the one-to-one correspondence implied by the term "monomer" is misleading. As well be shown in §3.5.3 using results from neutron scattering experiments, a rather short chain can be used to model a polymer's behavior. For example, a chain of 11 beads (having 10 links) can represent the conformation of an atactic polystyrene molecule of molecular weight 100,000

---

[3] "Links" are also commonly referred to as "rods," "segments," or "Kuhn steps." These are convenient synonyms though the last is sloppy language. Each link has a *length* of one Kuhn step, but is in itself not a Kuhn step.

[4] Actually, a bead-link pair should be considered as the basic repeat unit.

(degree of polymerization $\cong$ 1000) rather well. This means that each bead-link pair represents on the order of 100 styrenic monomers. Finally, considering links to be chemical bonds between atoms leads one into the same distortions of size as claiming that beads represent chemical monomers.

However, there are cases in which beads, or occupied lattice sites, *are* meant to represent monomers, links represent chemical bonds, and chains represent oligomers [Boyd, 1989; Vacatello, *et al.*, 1980]. For example, Boyd in a recent paper carried out the simulation of tetracosane [$CH_3(CH_2)_{22}CH_3$] as 24 repulsive centers, using his own potential energy function. Also, the included angles between links and the length of the links were fixed to correspond to the experimental values, 109.5° and 0.154 nm, respectively. Thus, each center was intended to represent a methylene unit, and Boyd ignored end effects. He then compared his simulation data directly to that of tetracosane. While he overstates his case in calling tetracosane "polymethylene" (*i.e.*, polyethylene), this type of direct comparison between the results from computer simulations and experimental data is surprisingly rare.

The results presented in this thesis, while compared to experimental data for polymers, would not be directly comparable to oligomers having the same number of monomers as the chains have beads. For instance, a chain having 11 beads should not be compared to hendecane [$CH_3(CH_2)_9CH_3$]. The most important reason for not comparing the two directly is geometric: The lattice simulations keep the links at 90° angles, and the continuous space simulations have links that are freely jointed about the bead centers. Also, in both types of simulation no potentials are included that would favor *trans* or *gauche* states. In fact, the lowest energy form for the chains is a rigid rod, not a planar zig-zag. Again, all that is meant to be represented in these simulations is the behavior derived from global considerations that affect the conformations of the chains.

So, in general, do *not* equate beads with monomers, links with bonds, or chains with polymers. The global properties of a macromolecule can often be simulated without including detailed, chemical information. Consider the contorted

shapes shown in Figure 2.2; only enough junction points need to be present to represent a polymer on some large length scale (*e.g.*, 10–100 nm). This correspondence also permits the results of a given set of simulations to be applied to a large class of macromolecules without having to simulate each class separately.

2.3 Method for the Lattice Simulations.

There are four primary concerns for the method of the simulation: How the system of chains is initialized; how the chains' conformations are generated; how a system's configuration is accepted, and how the statistics for the desired quantities are obtained. Each of these topics will be presented in separate sections.[5]

Each site was only permitted to have one bead present so that the chains moved as a self-avoiding walk. This mechanism is the lattice version of the "pearl necklace" model [Baumgärtner, 1987], but is extended to a system of chains rather than just a single chain on the lattice.

The lattice sites not taken by the chains were left unoccupied. This is different from the usual lattice model in which unlinked sites are considered to contain solvent [Flory, 1953, Chp. 12; Okamoto and Bellemans, 1979]. As the only interaction present is that of excluded volume, the chains are considered to be in an athermal system surrounded by a very good solvent.

2.3.1 Initialization of the System.

A simple cubic lattice was used, enumerated such that the Cartesian point $(x, y, z) = (0, 0, 0)$ was at the center of the lattice; this arrangement requires that an odd number of lattice sites are present *per* edge length to retain symmetry and

---

[5] A representative code used in the lattice simulations is included in Appendix A.

that the reference lattice meets the requirement that $\lceil -H/2 \rceil \leq x_i \leq \lfloor +H/2 \rfloor$ in integral steps where x_i represents any of the coordinates x, y, or z, and H is the number of layers in the lattice and equals the number of sites along a length of edge. Thus, for $H = 11$, the range would be $-5 \leq x_i \leq 5$ in integral steps for each coordinate, x, y, or z. Periodic boundary conditions are present for the x-direction and y-direction; however, the z direction is bounded by "hard walls" at $z = \lceil -H/2 \rceil - 1 = \lfloor -H/2 \rfloor$ and $z = \lfloor +H/2 \rfloor + 1 = \lceil +H/2 \rceil$ as required by Dickman's method [1987].

In the original version of the initialization subroutine, `INITCH`, to place the first bead of a chain, a site within the reference simple cubic lattice was chosen randomly.[6] This site was then tested to see if it was occupied. (With lattices, a simple storage of a one or a zero in an array is sufficient to denote occupancy simplifying the counting of occupied sites, *e.g.*, for the density in a layer.) If the site was found to be vacant, the position was stored, and the site marked as occupied. If the site was already occupied, then a new site within the reference lattice was chosen to place the initial "bead."

A direction for the second "bead" was selected among the six possible directions on the lattice, and this vector was added to the position of the initial site. If the site chosen for the second bead was occupied, a new direction vector was selected. If the site was vacant, the bead was placed, and a direction would be chosen to place the third bead. This process continued until all n beads for a given chain were placed. Then, the site for the initial bead of the next chain would be chosen, and the process repeated until all N_c chains had been placed.

The process for determining whether a site was occupied for beads other than the first required two tests. First, the z-coordinate was checked to be sure that it met the condition, $\lceil -H/2 \rceil \leq z \leq \lfloor +H/2 \rfloor$; if this condition was violated, which meant that the bead's attempted position was within one of the hard walls, a

---

[6] All "random" choices are made in these simulations using the pseudo-random number generator subroutine `DRAN`. See Appendix D for the code.

new direction vector was chosen. Second, because the x- and y-coordinates were periodic and no restriction was placed on the chains to remain within the reference lattice, the coordinates of any bead (excluding the initial bead) were mapped back onto the reference lattice before applying a simple test for occupation.[7]

This original algorithm had to be modified as the concentration increased. Initially, crude modifications, such as resetting the initial seed value of random number generator after an arbitrary number of failures to place a bead, were sufficient to initialize a system. However, as the bead density increased past a value of approximately 0.2, rather substantial adjustments to this algorithm were made based on some ideas from Kolinski, Skolnick, and Yaris [1986a]. For example, the initial beads for each chain were placed on a lattice $\lceil -H/4 \rceil \leq x_i \leq \lfloor +H/4 \rfloor$ with the center point $(0,0,0)$ being the same as the reference lattice's. The coordinates were then multiplied by a factor of two to move the initial points away from one another. In this manner the initial points for each chain were guaranteed not to be adjacent, thereby reducing the possibility of one chain's growth impeding the growth of another chain. The second bead for every chain was then placed. Finally, the rate of growth of the chains was limited so that no chain could propagate faster than any other.[8]

The important characteristic of the system to note, which occurs using either method of initialization, is that the initial conformation of a chain is likely to be highly contorted. This implies that the initial system is in a high energy state, or equivalently, is a system that has a high temperature. To study low temperature systems, a careful process of equilibration must be applied before sampling.

---

[7] The reason for periodic boundary conditions is to avoid edge effects and retain a constant density in simulations where small numbers of particles are meant to represent large, thermodynamic systems. For a good discussion, see Allen and Tildesley [1987], Chapter 1.

[8] The "crude" modifications are demonstrated within the subroutine `INITCH` in Appendix A. The more sophisticated techniques are documented within the subroutine `STRAD` in Appendix B.

2.3.2 Generation of the Chains' Conformations.

Using reptation in computer simulations to move chains predates de Gennes' model [1971] and has proven itself to be an efficient method by which to manipulate chains, both on a lattice [Kolinski, Skolnick, and Yaris, 1986c; Webman, Lebowitz, and Kalos, 1980] and in continuous space [Boyd, 1989]. A number of ways have been proposed by which to perform this motion with the earliest proposals coming from Kron [1965; Kron and Ptitsyn, 1967]. We have chosen a modified version of the mechanism proposed by Wall and Mandel [1975].

Wall and Mandel used a method in which each chain has a "head" and a "tail" designated. A new position for the "head" bead is determined by adding an arbitrary direction vector (one of six in the case of the simple cubic lattice) to the position of the head bead. If the chosen site is vacant, the position is accepted, and the rest of the chain is "slithered" along its contour with the former position of the "tail" bead being lost. If the site had been occupied, the chain remained in its former position, and the designation of "head" bead and "tail" bead was switched. In contrast, we randomly chose which end bead to move without permanently affixing a label of "head" or "tail" to an end bead. As Wall and Mandel mention in their paper, this is an equivalent method.[9]

The full procedure for chain movement is as follows: The chain to be moved was chosen at random from all of the chains present. The end of that chain to be moved was then randomly chosen. A (randomly chosen) direction vector was selected from the six possible choices, and this vector was added to the former position of the designated end bead to propose a new position. This new position was then tested for occupancy.

If the chosen site was unoccupied, then the "head" bead (using Wall and Mandel's terminology) was moved to the new site, and the rest of the chain was

---

[9] In fact, this reference [Wall and Mandel, 1975] includes almost all of the variations possible using reptation as the mechanism of movement.

slithered along its contour. For example, take a chain consisting of five beads with bead 5 considered to be the "head" bead. Let 1-2-3-4-5 represent the sites of the original chain. The bead at site 5 attempts to move to a new site 5′. Presuming this attempt is successful, all of the beads "move" along the contour such that $4 \rightarrow 4' = 5$, $3 \rightarrow 3' = 4$, $2 \rightarrow 2' = 3$, and $1 \rightarrow 1' = 2$ with the original position of the bead at site 1 being lost.[10]

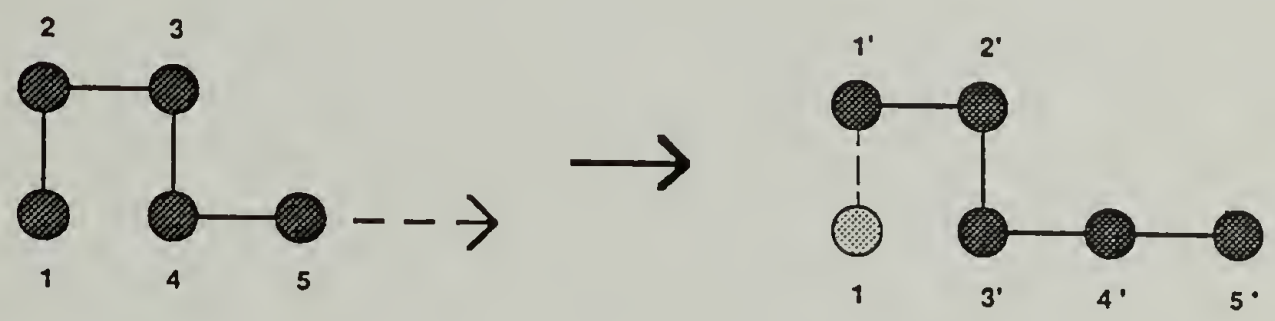

Figure 2.1
A Step by Reptation for a Five Bead Chain on a Lattice.

If the chosen site was occupied, then the following three possibilities exist:

1. The chosen site was occupied by the penultimate bead attached to the "head" bead. Because the chain cannot fold back on itself, as this is a self-avoiding walk, the algorithm required that a new direction vector be chosen [Kron and Ptitsyn, 1967; Rosenbluth and Rosenbluth, 1955]. Hence, while there are six nearest neighbor sites on a simple cubic lattice, only five of these were valid choices at any time.

2. The chosen site was occupied by the ultimate bead at the other end of the chain, *i.e.*, the "tail" bead. In this case, the move was accepted because, when the chain slithered along its contour, the tail bead would move from this site [Wall and Mandel, 1975].

---

[10] This manipulation involves a large amount of array elements being shifted and led to the concept of "snipping" an end bead and placing it at the other terminus of a chain for serial machines [Kron, 1965]. On vector machines, such manipulations are simple and contribute negligibly to the overall execution time of the program. At present, choosing a method is a simply matter of personal taste when all beads are equivalent.

3. The chosen site was occupied by a bead on the chain being moved that was neither the penultimate bead nor the alternate ultimate bead, or the site was occupied by a bead from another chain. In any of these cases, the attempted move failed, and the chain remained in its previous conformation.

Thus, the mechanism for movement is similar to Wall and Mandel's [1975], but discards the concept of alternately labeling the ultimate beads as "head" or "tail" to reverse the direction of movement depending on the success of the previous attempted move. For the present mechanism, a random selection of which end bead on the designated chain is to be moved is implemented. While this mechanism prescribes if a particular conformation of a chain is accepted, in our work (and as opposed to other studies [*e.g.*, Okamoto and Bellemans, 1979; Wall and Mandel, 1975]) this mechanism does *not* determine if a particular configuration of a system is accepted. This latter point is covered in the next section.

2.3.3 Acceptance of the Systems' Configurations.

As in most other Monte Carlo simulations, a simple Metropolis test [Metropolis, *et al.*, 1953] was applied to the system's energy to determine if a new configuration would be accepted. While the contributions to this energy will be delineated below, two important distinctions to the usual mechanism must be discussed: cooperation among the chains, and correlation among the samples.

2.3.3.1 The Metropolis Sampling Method.

To understand the statistical difficulties involved, we begin with an explanation from Metropolis, *et al.* [1953] on their method of implementing the Boltzmann factor to determine how to weight a configuration in determining a canonical average. This technique is the most common sampling method used in Monte Carlo simulations.

For any classical, three-dimensional system containing N interacting particles, each particle requires three coordinates to describe its position. Thus, the $3N$ coordinates completely describe the positions of all the particles in the system. Similarly, to each coordinate there corresponds a conjugate momentum so that, by including the $3N$ momenta, the mechanical state of the system is specified completely [McQuarrie, 1976, Chp. 7].

Denoting the $3N$ momenta as $p_1, p_2, p_3, \ldots, p_{3N}$ and the $3N$ coordinates as $q_1, q_2, q_3, \ldots, q_{3N}$, then to calculate the equilibrium value for some quantity of interest F the canonical ensemble average is defined by the relation

$$\langle F \rangle \equiv \left[\frac{\int F \exp(-E/\mathrm{k}_B T) d^{3N}p \; d^{3N}q}{\int \exp(-E/\mathrm{k}_B T) d^{3N}p \; d^{3N}q} \right], \qquad (2\text{–}8)$$

where $(d^{3N}p \; d^{3N}q)$ is a volume element in the $6N$-dimensional phase space, recalling that N is the number of particles in the system [Metropolis, *et al.*, 1953]. While the forces between particles are independent of their velocity, such that the integration over the momenta p can be removed, one still has a $3N$-dimensional integral to be solved.

The direct approach to solving Equation 2–8 is to sample configurations randomly and then weight them by the appropriate Boltzmann factor, $\exp(-E/\mathrm{k}_B T)$. The difficulty with using this direct approach is that there is a high probability, especially in close-packed systems [Metropolis, *et al.*, 1953] or systems of high particle density, that one would chose a configuration for which the Boltzmann factor is quite small. This means that the chosen configuration would have little contribution to the final equilibrium value of the quantity F. Hence, the direct approach would waste time and effort. Metropolis, *et al.* made the seminal change of modifying the scheme so that the averaging is done by choosing the configurations with a probability of $\exp(-E/\mathrm{k}_B T)$ and weighting these configurations equally.

Denoting the difference between the present total energy of a system and the previous total energy of a system as ΔE, if $\Delta E \leq 0$, the move is automatically

accepted. If $\Delta E > 0$, then the move is allowed contingent on the probability, $\xi \equiv \exp(-\Delta E/k_B T)$, which is bounded by 0 and 1. Let $\xi' \equiv \exp(-\Delta E'/k_B T)$ such that $0 < \xi' < 1$. The acceptance of the move is determined by choosing a number ξ' randomly from a uniform distribution between 0 and 1 whereby, if $\xi' \leq \xi$ the new move is accepted; otherwise, the particles remain in their former positions. This test implies that the energy of the system ΔE was less than an arbitrarily chosen energy change $\Delta E'$ (represented through ξ') so that the present change in the system's energy is considered to be admissible, and the present configuration is accepted. Whether the present configuration is accepted or rejected, the resultant configuration is considered to be a new configuration for the purpose of determining the equilibrium value of F, which is now calculated as an arithmetic average,

$$\langle F \rangle = (\frac{1}{M}) \sum_{j=1}^{M} F_j \ , \qquad (2-9$$

where F_j is the value of the property F after the j^{th} move and M is the total number of moves.

2.3.3.2 Cooperation Among the Chains.

The usual procedure to avoid bias from sampling in simulations is to apply the Metropolis test after any particle has attempted a move. This was the mechanism used in the original paper [Metropolis, *et al.*, 1953] where, if an attempted move of a particle were forbidden, the former configuration of the system was counted again into the average for some desired property F. (See Eqns. 2–7 and 2–8). For polymeric systems employing reptation, Wall and Mandel [1975] implemented the Metropolis test after any chain had attempted a move. Wall and Mandel's method is equivalent to the method used in the original paper because in reptation only one bead is moving to a new location; all other beads just follow to previously occupied positions. Hence, effectively only one bead has moved to a new site.

We desired to include cooperation among the chains such that degenerate energy states for different configurations are permitted even if a higher energy state must be passed through to arrive at the second configuration. (This concept is illustrated in the example below.) This condition was implemented because of concern with low temperature simulations in which chains can become trapped in one system configuration, unable to explore other possibilities without this ability to ignore higher energy, intermediate states.

The mechanism for including cooperation does not bias the results any more than Wall and Mandel's original reptation mechanism [1975]. In fact, as more possibilities exist to explore phase space between samples, it may actually bias the results less. In any case, no matter how much movement occurs between samples, if a configuration fails the Metropolis test, then the system is reset to the *previously sampled* configuration, not just the previous configuration. By this approach, unbiased results may be obtained while providing a mechanism by which the system may move freely and cooperatively.

The idea behind cooperation is shown in Figure 2.2 where (a) shows the initial configuration for a system of three chains, (b) shows a configuration after a chain has moved from position 1 to $1'$, (c) shows a configuration after a chain has moved from position 2 to $2'$, and (d) shows a configuration after a chain has moved from position 3 to $3'$. One will note that configurations (a) and (d) have the same energy (*i.e.*, are degenerate), if one is just counting the number of bends in the system. However, the system moved from (a) to (b) through a higher energy state. While the Metropolis sampling technique allows for the possibility of higher energy states, when the these systems' temperatures were low enough ($\tau < 1$), any higher energy state was rejected [Starry, 1987] so that the present mechanism of reptation could not sample phase space efficiently.[11] The cooperative mechanism was put into place to temper this limitation.

---

[11] What this means, of course, is that reptation is not suited to studies at low system temperatures.

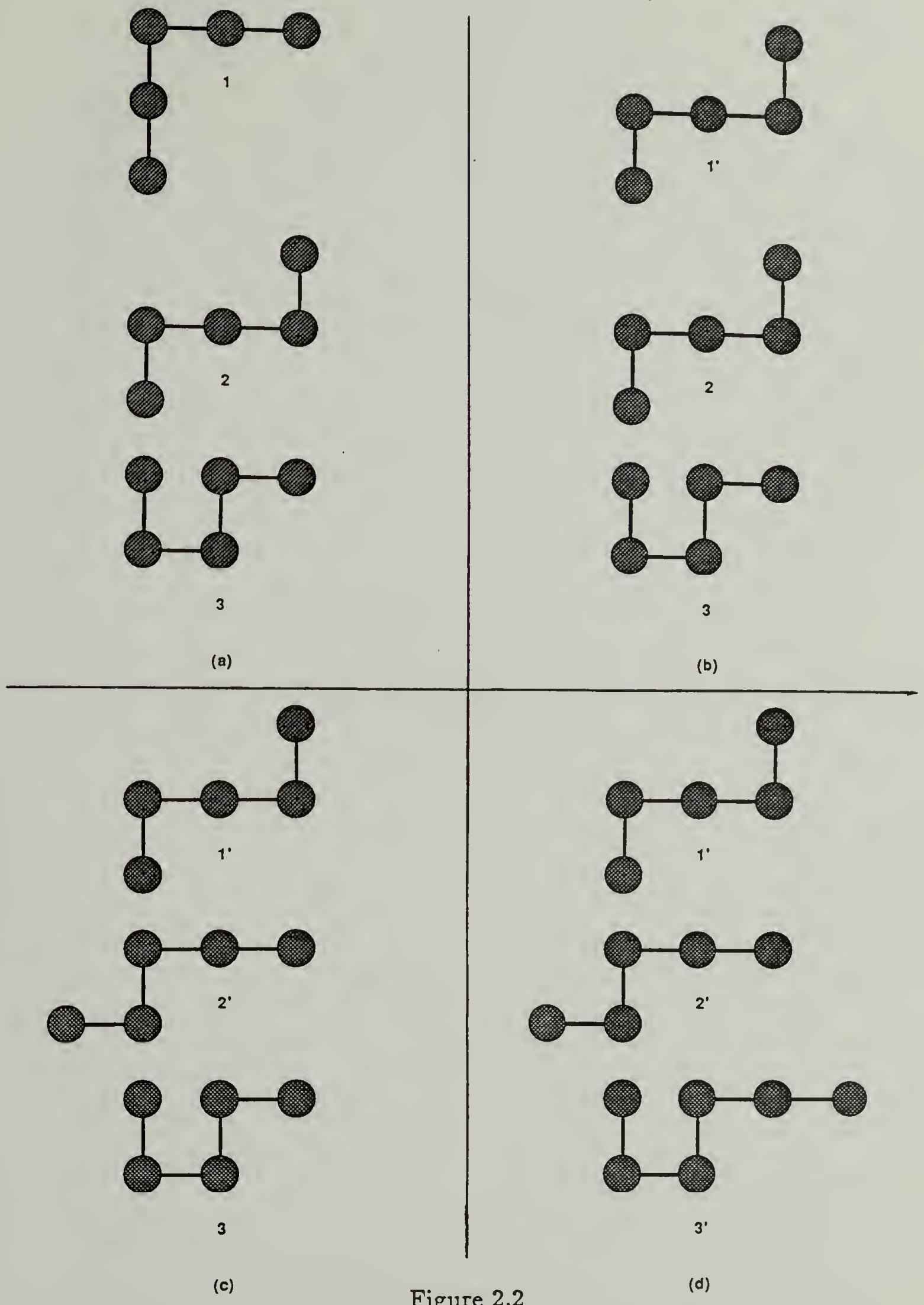

Figure 2.2

An example of cooperative motion for chains. Note that configurations (b) and (c) have higher energies, *i.e.*, more bends, than the degenerate initial and final configurations.

A second impetus for this mechanism was that, in real, temporally based systems, all of the chains are moving simultaneously. Thus, moving all of the chains before a Metropolis sampling test was performed seemed to be reasonable. This concept is misleading and should be considered *void*. There is *no* temporal aspect to these Monte Carlo simulations.[12] The interpretation of cooperation as chains moving simultaneously in time, while conceptually tempting, should not be accepted.

There should be no complications in the statistical bias for the cooperative mechanism used here. The only difficulty noted by Wall and Mandel [1975] is that, by using reptation, regions of phase space that have "double culs-de-sac" cannot be explored. (See Figure 2.3 below.) Wall and Mandel did not perceive this exclusion to affect the results greatly as the amount of phase space that would have such artifacts was considered to be small; investigations within this group [Dadmun, 1988] confirm this assumption.

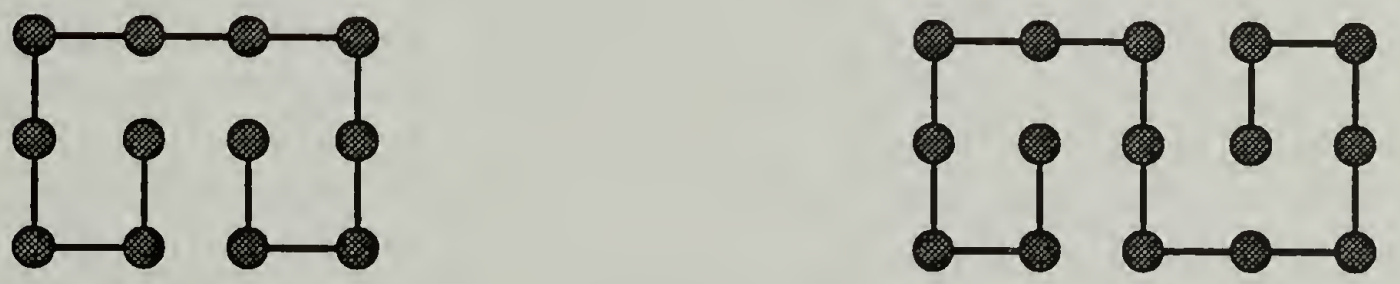

Figure 2.3
Examples of Double Culs-de-sac (in Two Dimensions).

In the absence of a formal proof of the bias introduced by cooperative reptation, skepticism may remain, but the agreement of these results when using different numbers of steps before applying the Metropolis test, by using other mechanisms of movement, and after comparing with experimental data belies any

[12] Even the ergodic condition does not introduce a temporal aspect because ergodicity only implies that an average over an *infinite* amount of time will be equal to the ensemble average. For an especially clear discussion of ergodicity, and its implications, see Hobson [1971], pp. 4–7.

significant bias having been introduced by this mechanism. However, such is the case for reptation only. Other mechanisms for multiple movement of chains that would be required for efficient computing at low system temperatures, *e.g.*, parallel kink jump, are not obviously unbiased. A comprehensive theory of the statistical bias introduced by alternate mechanisms is needed. (See §4.2.1.)

2.3.3.3 Correlation Among the Configurations.

The aspect of correlation when sampling a system's configuration has more to do with practical difficulties than with conceptual ones. The difficulty here is that reptation, along with some other forms of motion, does not move the chains a considerable distance—only one Kuhn step *per* move. Due to this minute change, even if a new conformation for a chain has been accepted, the new conformation is quite like the former conformation. Similarly, even when a whole set of chains has been moved, the system's configuration is much like its former configuration. The mechanism used to circumvent this limitation of reptational motion merits a short discussion.

If the desired quantities (*e.g.*, the squared radius of gyration R_g^2) are sampled just after allowing each chain in the system to attempt a move, then the configuration obtained would be closely correlated to the previously sampled configuration. Since the concept of Monte Carlo sampling is to wander relatively randomly throughout phase space, the samples taken should be effectively not be correlated.

In these simulations, a chain was considered to have moved significantly if the chain had slithered a linear distance equal to its contour length. In such a situation the conformation of the chain effectively would not be correlated to its previously sampled conformation. Since the initial systems contained chains of 10 Kuhn steps in length, each chain in the system was permitted 10 attempts to move (on average) between Metropolis tests. The actual mechanism comprised

selecting a chain at random $10 \cdot N_c$ times, where N_c is the number of chains in the system, and attempting to move that chain. (The chains were chosen randomly to avoid any bias in the ordering.) Thus, between any given set of Metropolis samples some chains would have attempted to move more than 10 times, and some chains less; but, over several Metropolis samplings, each chain would have attempted to move an average of 10 times between samples.

In addition to the desire to obtain samples that were not correlated, there is the additional, practical concern of trying to obtain accurate averages. By sampling from a certain point in phase space and moving slowly away from that point while traveling through phase space, the average value that corresponds to the desired "equilibrium" value require more samples as the average at any time is heavily weighted by the initial configurations of the system; thus, the simulation must run longer to ensure that the appropriate average has been obtained. (On the other hand, if many configurations are created, but not sampled, then a good deal of computer time has been spent without gaining any data. Balancing these considerations requires experimentation, making comparisons, and performing controls.) In all cases a large number of samples (several thousand) need to be obtained for a valid average to be calculated.

The influence moving the chains several times before sampling should not influence the statistical bias, which was discussed in §2.3.3.2 above. As the reptation mechanism is used exclusively, the results presented here should be statistically *unbiased* by the mechanisms used to provide cooperation and preclude correlation. In general, however, the influence on the statistical bias from other mechanisms of movement must be addressed. (See §4.2.1.)

2.3.4 Sampling of the Desired Quantities.

The primary quantities of interest in these simulations, as can be seen from Equations 2–4 to 2–6, were the modified osmotic pressure π^*, the bulk density ϕ_B,

and the compressibility Z. To obtain the osmotic pressure, one had to monitor the bead density on the top layer of the lattice ρ_w. In addition, to test Daoud's relation ($R_e^2 \sim \rho^{-1/4}$) [Daoud, *et al.*, 1975] and determine the Flory exponent ν for the systems, the mean squared end-to-end distance $\langle R_e^2 \rangle$ was calculated. The Flory exponent was determined from the relation

$$\nu = \frac{1}{2}[\ \ln\langle R_e^2 \rangle\ /\ \ln(n-1)]\ . \qquad (2\text{–}10)$$

The mean squared radius of gyration $\langle R_g^2 \rangle$ was also tabulated, both as a measure of the dependence of the average size of a chain upon concentration and as a measure by which to check the system for "finite size effects." (See §2.4.2 and §4.2.1.)

To monitor the program itself, measurements of two types were included, statistics on how easily new configurations were generated and statistics for physical quantities. For the former, the number of failures for generating a new configuration fell into three categories: failures for attempting to move into the walls (located in the z-direction), excluded volume failures (*i.e.*, trying to move to an occupied site), and energy failures in which the new configuration was rejected through an application of the Metropolis energy test (*cf.* §2.3.3.1). Presuming that the system is moving well (as determined from the first two categories), this last monitor on the energy failures is the most important for evaluating the validity of the averages that are calculated. This is because the configuration after the Metropolis test is the one added into all of the running sums whether the present configuration is a new or the previously sampled configuration. Thus, if the fraction of attempted new configurations that are accepted is too low, then a significant portion of phase space has not been sampled, and only a few, isolated configurations have been considered.[13] (This was the manner in which we dis-

[13] The fraction of accepted configurations that is considered to be desirable varies. Some authors believe that the number of successes and failures should balance so that about 50% of the configurations should be new. Others claim that a fraction of 20 to 30% actually samples phase space more efficiently. In any case numbers on the order of 1% or less definitely indicate problems, and even in the 10% range care needs to be exercised; *i.e.*, trends in other properties must be closely observed. (See Allen and Tildesley, 1987, pp. 121–123.)

covered the difficulty of using this algorithm for low temperature studies, defined here as $\tau < 1$.)

The statistics of physical quantities were monitored as a function of the repulsive energy λ. Geometrical quantities (such as $\overline{R_e^2}$ and $\overline{R_g^2}$)[14], bulk densities (ϕ_B), and the Hamiltonian energies for the system were all averaged over the span of one λ to determine if these gave the appropriate trends. For example, as the repulsive energy on the wall increased, the chains would be increasingly confined to the inner layers of the lattice (in the z-direction) so that the bulk density would increase while the mean squared end-to-end distance and mean squared radius of gyration would decrease. Observing changes in these quantities as λ increased provided a way to determine if the quantities were modified in a predictable, and correct, manner.

The ensemble average for the the mean squared end-to-end distance was calculated as

$$\langle R_e^2 \rangle \equiv \langle ({}^n\mathbf{r} - {}^1\mathbf{r})^2 \rangle \qquad (2\text{–}11)$$
$$= \frac{\sum_{l=1}^{N_\lambda} \sum_{k=1}^{N_s} \sum_{j=1}^{N_c} ({}^n\mathbf{r}_{j,k,l} - {}^1\mathbf{r}_{j,k,l})^2}{N_c \, N_s \, N_\lambda},$$

while the mean squared radius of gyration was calculated as

$$\langle R_g^2 \rangle \equiv \frac{1}{n} \sum_{i=1}^{n} \langle ({}^i\mathbf{r} - {}^{\mathrm{cm}}\mathbf{r})^2 \rangle \qquad (2\text{–}12)$$
$$= \frac{\sum_{l=1}^{N_\lambda} \sum_{k=1}^{N_s} \sum_{j=1}^{N_c} \sum_{i=1}^{n} ({}^i\mathbf{r}_{j,k,l} - {}^{\mathrm{cm}}\mathbf{r}_{j,k,l})^2}{n \, N_c \, N_s \, N_\lambda}.$$

where

N_s $\equiv$ Number of samples taken;

N_λ $\equiv$ Number of λ values utilized in the simulation;

---

[14] The distinction between $\langle R_e^2 \rangle$ and $\overline{R_e^2}$ is that the former represents the mean squared average obtained for the whole canonical ensemble simulation while the latter represents a mean squared average over a contiguous subset of values.

$^{cm}\mathbf{r}_j, k, l$ ≡ Position vector for the center of mass of chain j in sample k having a repulsive energy l;

$^{i}\mathbf{r}_j, k, l$ ≡ Position vector for bead i of chain j in sample k having a repulsive energy l.

Note that the canonical mean squared end-to-end distance $\langle R_e^2 \rangle$ and mean squared radius of gyration $\langle R_g^2 \rangle$ were averaged over all values of the repulsive wall energy λ even though for each λ a different average value of $\overline{R_e^2}$ and $\overline{R_g^2}$ was obtained. This method was used to be consistent with the calculation of the values for the osmotic pressure and compressibility, both of which must be obtained from quantities garnered from all values of λ. Using the canonical averages does not obscure any information or trends as the values of $\overline{R_e^2}$ for a given concentration ϕ_B only differed by less than 1% and those for $\overline{R_g^2}$ differed by less than 0.5%.

A similar concern, of averaging over all λs to obtain an ensemble average, occurred for the bulk density ϕ_B. The density for any given sample was obtained by counting the number of occupied sites on the middle layers ($\lceil -H/4 \rceil \leq z \leq \lfloor +H/4 \rfloor$) and dividing by the number of sites *per* layer times the number of middle layers, that is, $H^2 \times \lfloor H/2 \rfloor$. Thus, one may represent the bulk density by the equation

$$\langle \phi_B \rangle \equiv \sum_{j=\lceil -H/4 \rceil}^{\lfloor +H/4 \rfloor} \sum_{i=1}^{H^2} \langle {}^{ij}\vartheta \rangle \qquad (2\text{–}13)$$

$$= \frac{\sum_{l=1}^{N_\lambda} \sum_{k=1}^{N_s} \sum_{j=\lceil -H/4 \rceil}^{\lfloor +H/4 \rfloor} \sum_{i=1}^{H^2} {}^{ij}\vartheta_{k,l}}{\lfloor H/2 \rfloor \; H^2 \; N_s \; N_\lambda},$$

where

$${}^{ij}\vartheta = \begin{cases} 0, & \text{if site } i \text{ of layer } j \text{ is vacant;} \\ 1, & \text{if site } i \text{ of layer } j \text{ is occupied.} \end{cases} \qquad (2\text{–}14)$$

The reason for the bulk density ϕ_B to be used, rather than the average density $\overline{\phi} \equiv n \; N_c / H^3$, is that, for simulations in which "hard" walls are present, the average density neglects the influence of the walls. Thus, the general consensus has been that using bulk densities (that is, densities of the fluid calculated far from

the walls) provides a better basis for representing the data [Snook and Henderson, 1978]. In Dickman's method [1987] where a repulsive potential is designed to force beads off of the topmost layer in the lattice ($z = \lfloor H/2 \rfloor$), the average density would also not properly represent the accessible number of lattice sites; hence the bulk density is a better choice.

The density on topmost layer of the lattice $\rho_w(\lambda)$ was calculated very similarly to the bulk density, but only one layer needed to be considered. One should also note that, in accordance with Equations 2–5 and 2–6, ρ_w is a function of the repulsive energy λ and was stored as such to perform the necessary integration. The representation for the the bulk density is given by the equation

$$\rho_w(\lambda) = \begin{cases} \dfrac{\sum_{k=1}^{N_s} \sum_{i=1}^{H^2} i\lfloor H/2 \rfloor \vartheta_k}{H^2 N_s}, & 0 < \lambda < 1; \\ \dfrac{\sum_{k=1}^{N_s} \sum_{i=1}^{H^2} i(\lceil -H/2 \rceil + 1) \vartheta_k}{H^2 N_s}, & \lambda = 1. \end{cases} \qquad (2\text{–}15)$$

(The reason for the exception of sampling the density on the penultimate bottom layer when $\lambda = 1$ is to be consistent with Dickman's method of monitoring the density on the bottom layer, where no repulsive potential is applied, during the sampling of all other λs to provide $\rho_w(1)$ [Dickman, 1987]).

Two thermodynamic quantities remain to be determined: the modified osmotic pressure π^*, and the compressibility for the simulations Z, which is compared to the compressibility from Flory-Huggins theory Z_{FH}. The osmotic pressure was obtained by using a Simpson's rule integration of the discrete $\rho_w(\lambda)$ values according to Equation 2–5 with the added information that $\rho_w(0) = 0$ since, as noted after Equation 2–3, at $\lambda = 0$, $Q' = 0$; hence, no configuration may have beads on the top layer at $\lambda = 0$.[15] The compressibility for the simulation

[15] The use of a Simpson's rule integration requires that an odd number of λs must be used. Since one end point is already defined at $\lambda = 0$, N_λ is an even number. This number ranged from ten to sixteen to keep the variance of Z within acceptable limits. The λs were not distributed evenly, but were bunched at the ends of the range (near 0 and 1) where the slope on the function $\rho_w(\lambda)/\lambda$ was steep.

was simply obtained using Equation 2–6 once the modified osmotic pressure and bulk density had been determined for the system. Finally, the compressibility from Flory-Huggins theory was provided by the equation [Muthukumar, 1985]

$$Z_{FH} = (\frac{n}{\phi_B})[3\ \ln(1-\alpha\phi_B) - \ln(1-\phi_B)], \qquad (2\text{–}16)$$

where $\alpha = (\frac{1}{3})(1 - 1/n)$. At this point one is now ready to determine the values for all of these quantities and to analyze the results.

2.4 Results from the Lattice Simulations.

2.4.1 Discussion of the Results.

While the codes have been written generally so that they are applicable to chains of various flexibility, the results shown here by this author and from others are only for quite flexible chains. There are three reasons for this. First, this work was begun on "high" temperature systems so that comparisons with known results could be made. Second, we found that the lattice simulations could not be performed for "low" temperature systems as there were not enough degrees of freedom available to allow the chains to evolve to an ordered state; for $\tau = (k_B T/\epsilon_b) < 1$, the system froze, and no movement of the chains occurred. Third, and most important, the results obtained made it clear that lattice simulations *could not provide* the correct thermodynamic trends and scaling relationships that are observed in experiments. This last point is a serious, and overlooked, consideration for lattice simulations generally.

Three principal systems were studied on the simple cubic lattice over a wide range of concentrations. The first system, chosen for simplicity and for comparison with previous work [Dadmun, 1987; Bellemans and de Vos, 1973] had chains of 10 segments (11 beads; $n = 11$) in length on a lattice having 11^3 sites. The second

system, which tested the finite size effects of the lattice's dimensions, had chains of 10 segments in length on a lattice having 17^3 sites. The third system, which tested the effect of chain length, had chains of 50 segments (51 beads; $n = 51$) in length on a lattice with 51^3 sites.

Since the size of the system did have an effect on the results obtained for the compressibility, three other system sizes were investigated to determine if a pattern could be discerned. These secondary systems all used chains of 10 segments in length (for speed of computation) and had lattices of (i) 15^3 sites, (ii) 21^3 sites, and (iii) 35^3 sites. As will be shown below, while some significant changes were noted in the results obtained here for the compressibility Z, these effects do *not* affect the conclusion that computationally accessible results from lattice simulations fail to provide the experimentally realized, semidilute scaling forms for the mean squared end-to-end distance and the osmotic pressure. [16]

In Figure 2.4 the results for the compressibility *per* bead Z/n are presented. The dotted and dashed lines represent the predictions of Flory-Huggins theory for the respective chain lengths used, *i.e.*, $n = 11$ (10 segments) and $n = 51$ (50 segments) (*cf.* Eqn. 2–16). Note that the majority of the systems studied do not deviate from the trends indicated by the Flory-Huggins predictions. Figure 2.5 shows a similar graph of our data and that from Bellemans and de Vos [1973] for systems containing chains of 10 segments in length ($n = 11$). The agreement displayed in both figures between the simulations' data and the predictions as well as between the results of different researchers is satisfying until one realizes that the compressibilities obtained should *not* be following the same trend as a mean field prediction at low to moderate concentrations, *i.e.*, the dilute and semidilute regions for solutions. Only in the bulk state or at high concentrations should a mean field result be appropriate, and at such concentrations the applicability

---

[16] The largest system studied here consisted of 1040 chains of 51 beads each on a 51^3 site lattice. A single execution of the program required about 42 CPU hours on the IBM 3090VF processors at the Cornell National Supercomputer Facility.

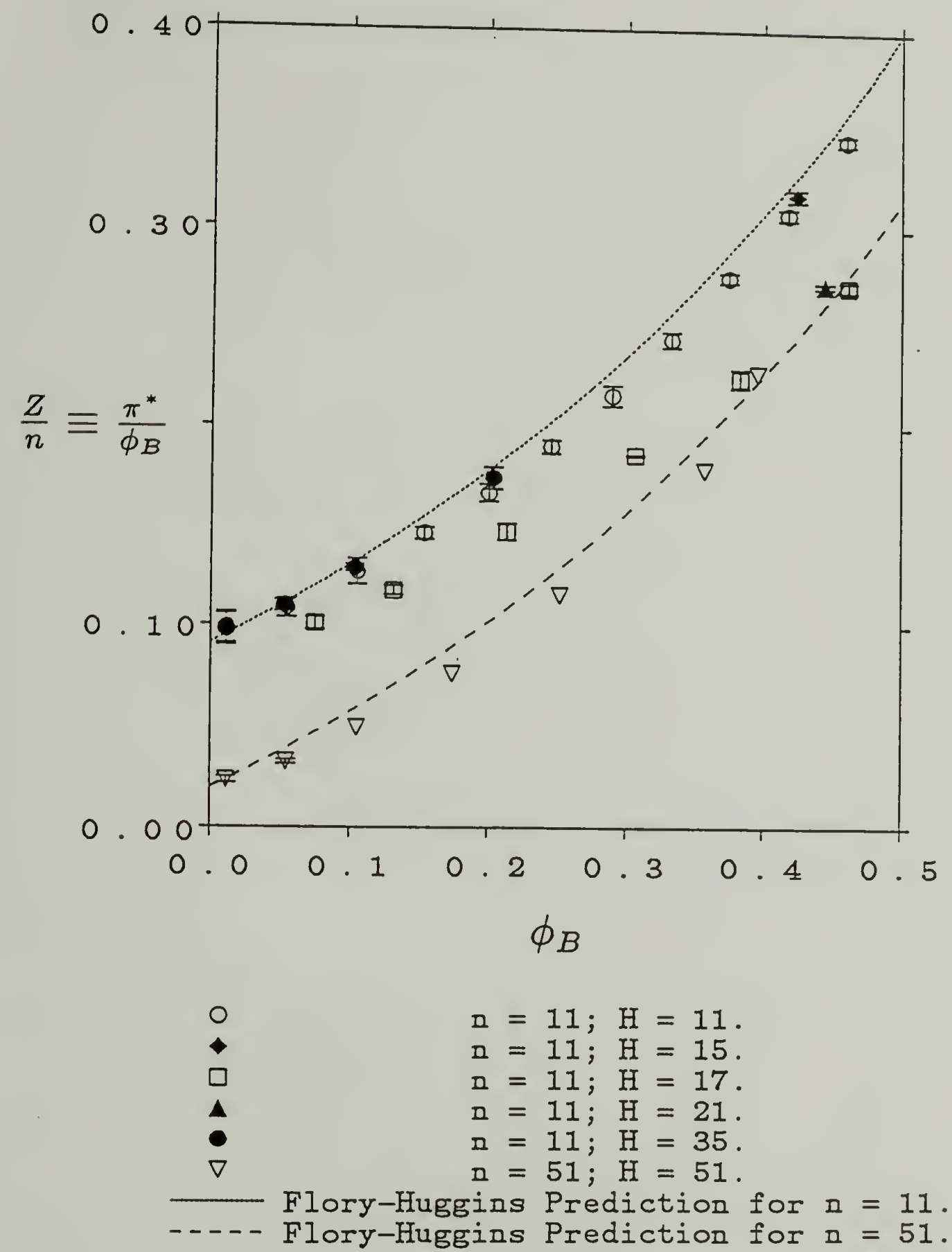

Figure 2.4

Compressibility *per* bead as a function of bead volume fraction for flexible chains on a simple cubic lattice.

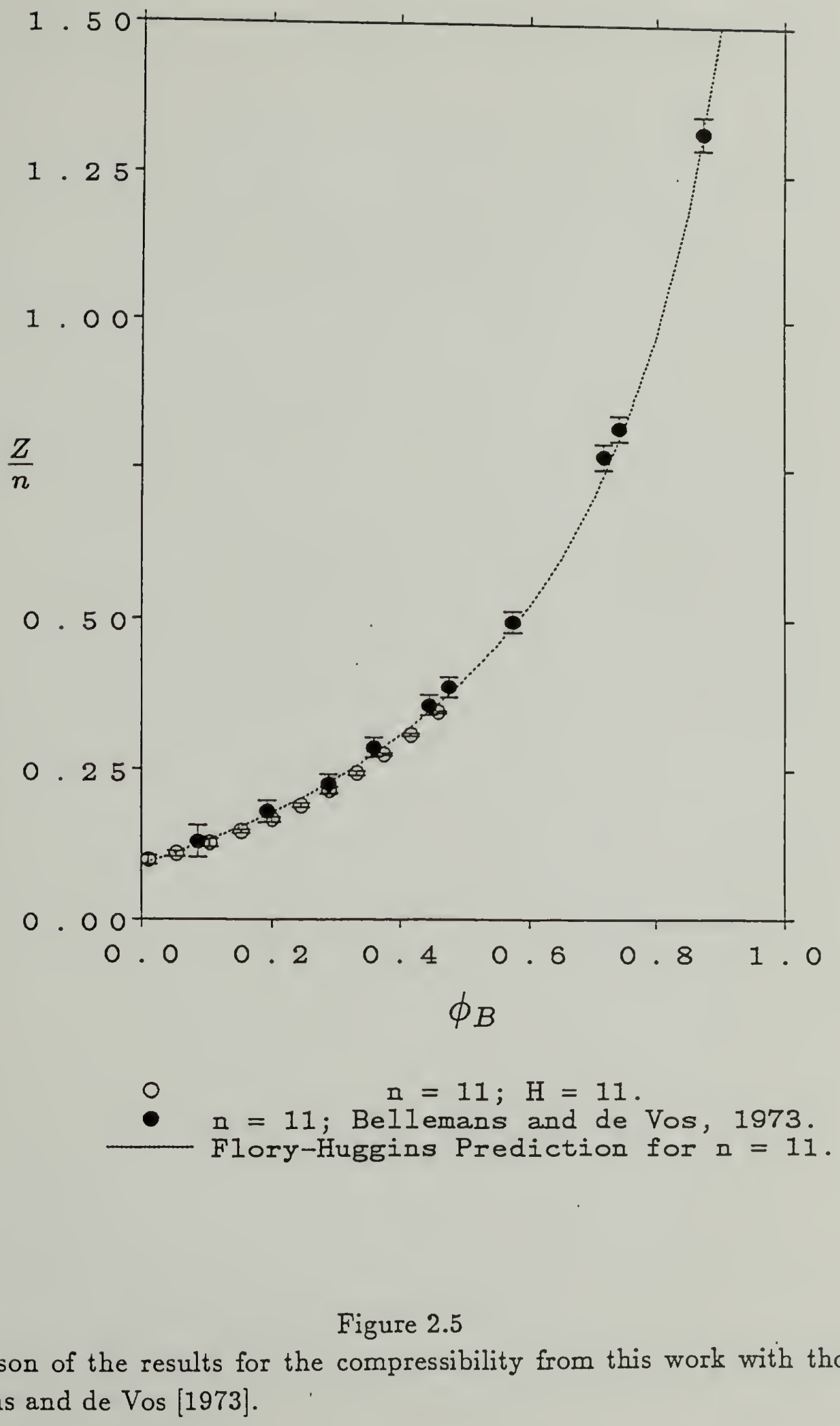

Figure 2.5

Comparison of the results for the compressibility from this work with those of Bellemans and de Vos [1973].

has been shown rather often, both in simulations [Baumgärtner, 1987; Bellemans and Janssens, 1974; Kolinski, Skolnick, and Yaris, 1986a, 1986c, 1987c] and in experiments [Baumgärtner, 1987; Cotton, *et al.*, 1974].

Figures 2.6 and 2.7 show the scaling relationship of des Cloizeaux's law [des Cloizeaux, 1975; de Gennes, 1979, pp. 76–80] that relates the osmotic pressure to the volume fraction of the solution as

$$\pi^* \sim \phi_B^{9/4}, \tag{2–17}$$

with the solid line representing this relation. The data from these simulations (Figure 2.6) neither approach nor deviate clearly from the expected trend. The results appear to curve continuously with no significant linear region on the log-log plot; the same is true for the results of Bellemans and de Vos [1973] shown in Figure 2.7. On the other hand, there is a region $0.35 \leq \phi_B \leq 0.60$ where des Cloizeaux's law is within the error displayed by the data of Bellemans and de Vos. However, that their simulations can actually represent the semidilute scaling forms is belied by the data for the mean squared end-to-end radii. (See below.) This may indicate that using the scaling form for the osmotic pressure is not a rigorous enough test of a simulation's data. One should also note that the system having 17^3 sites in Figure 2.6 deviates from the common trend; this again introduces the specter of finite size effects, which will be discussed shortly.

Figures 2.8 and 2.9 show the scaling relationship derived by Daoud, *et al.* [1975; de Gennes, 1979, pp. 76–80] that should hold in semidilute concentrations,

$$R_e^2 \sim \phi_B^{-1/4}. \tag{2–18}$$

Here, the lack of agreement for all systems with the experimentally determined relationship is clear. Only at the highest concentrations, $\phi_B \geq 0.4$, in Figure 2.8 does the system of 51^3 sites possibly approach the solid line denoting Daoud's relation, and not convincingly.

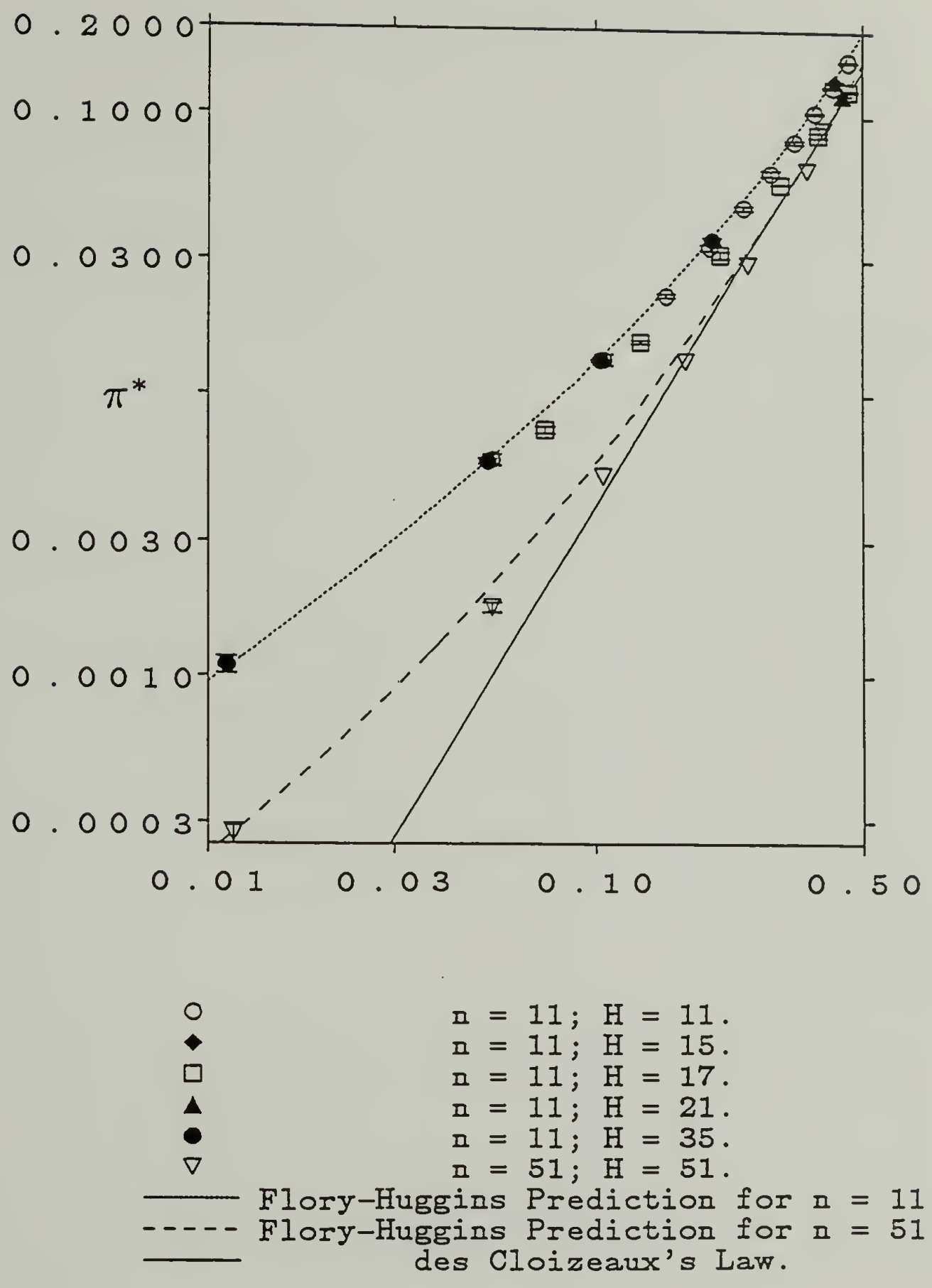

Figure 2.6

Modified osmotic pressure as a function of bead volume fraction for flexible chains on a simple cubic lattice.

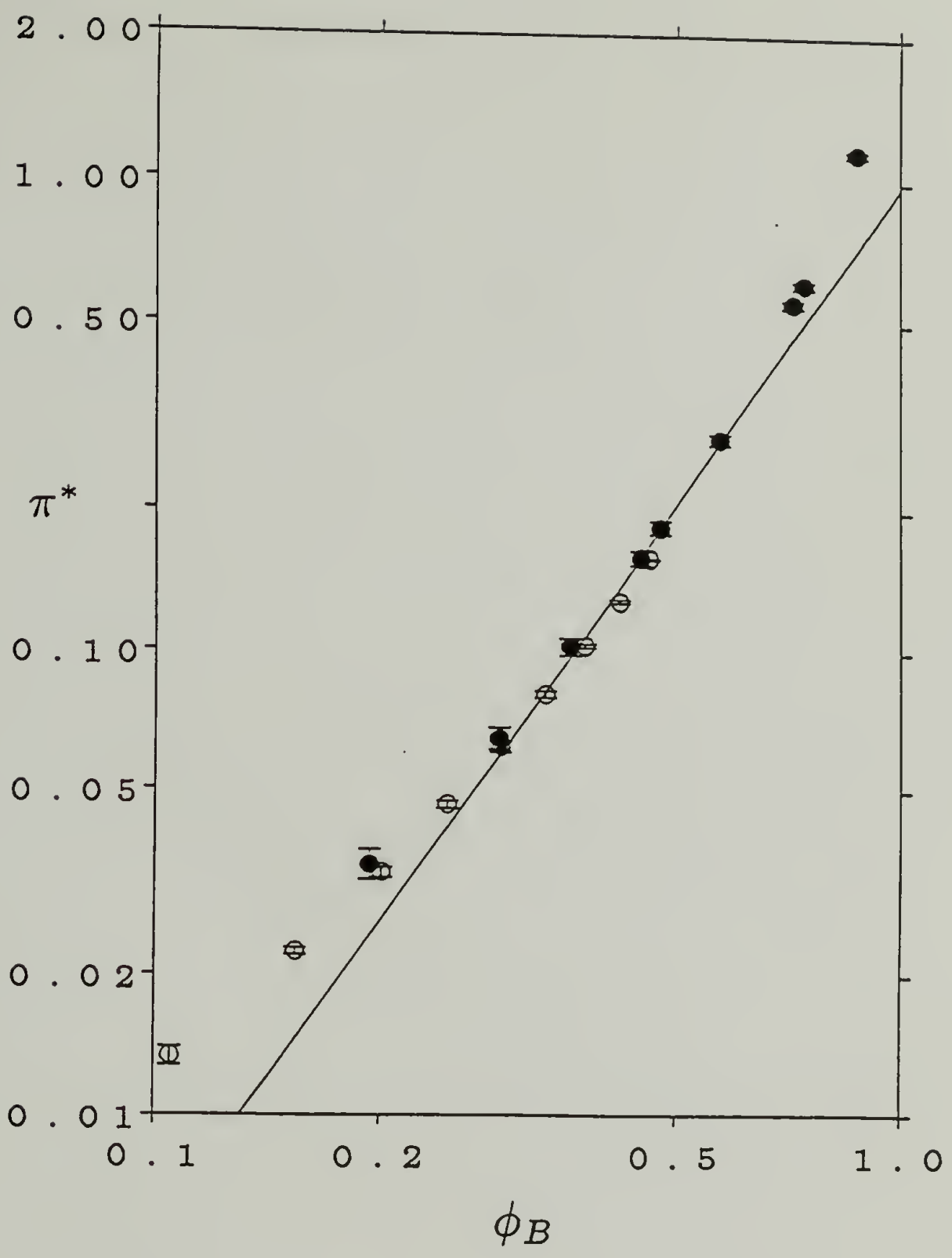

Figure 2.7

Comparison of the results for the modified osmotic pressure from this work with those of Bellemans and de Vos [1973].

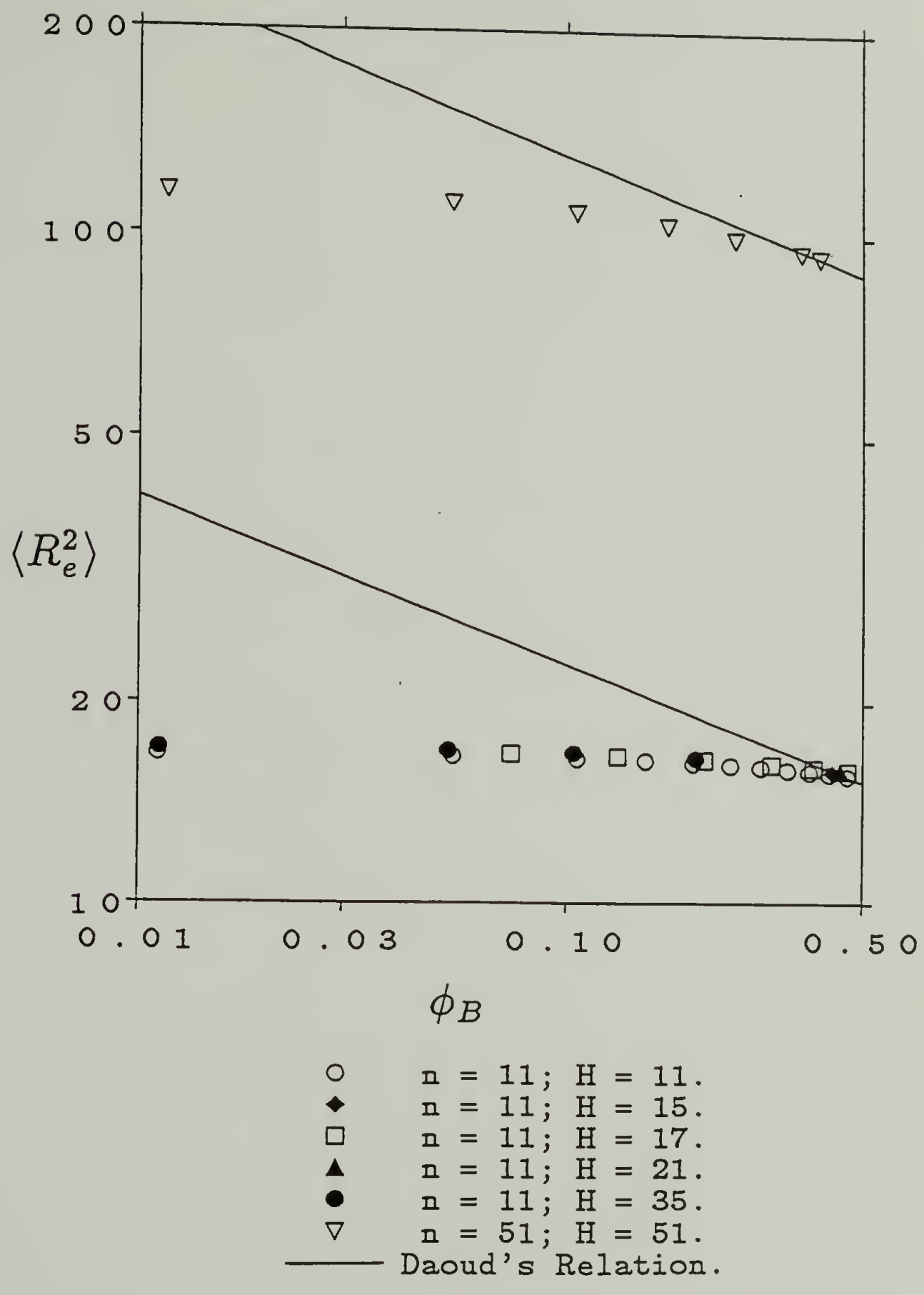

Figure 2.8

Mean squared end-to-end distance as a function of bead volume fraction for flexible chains on a simple cubic lattice.

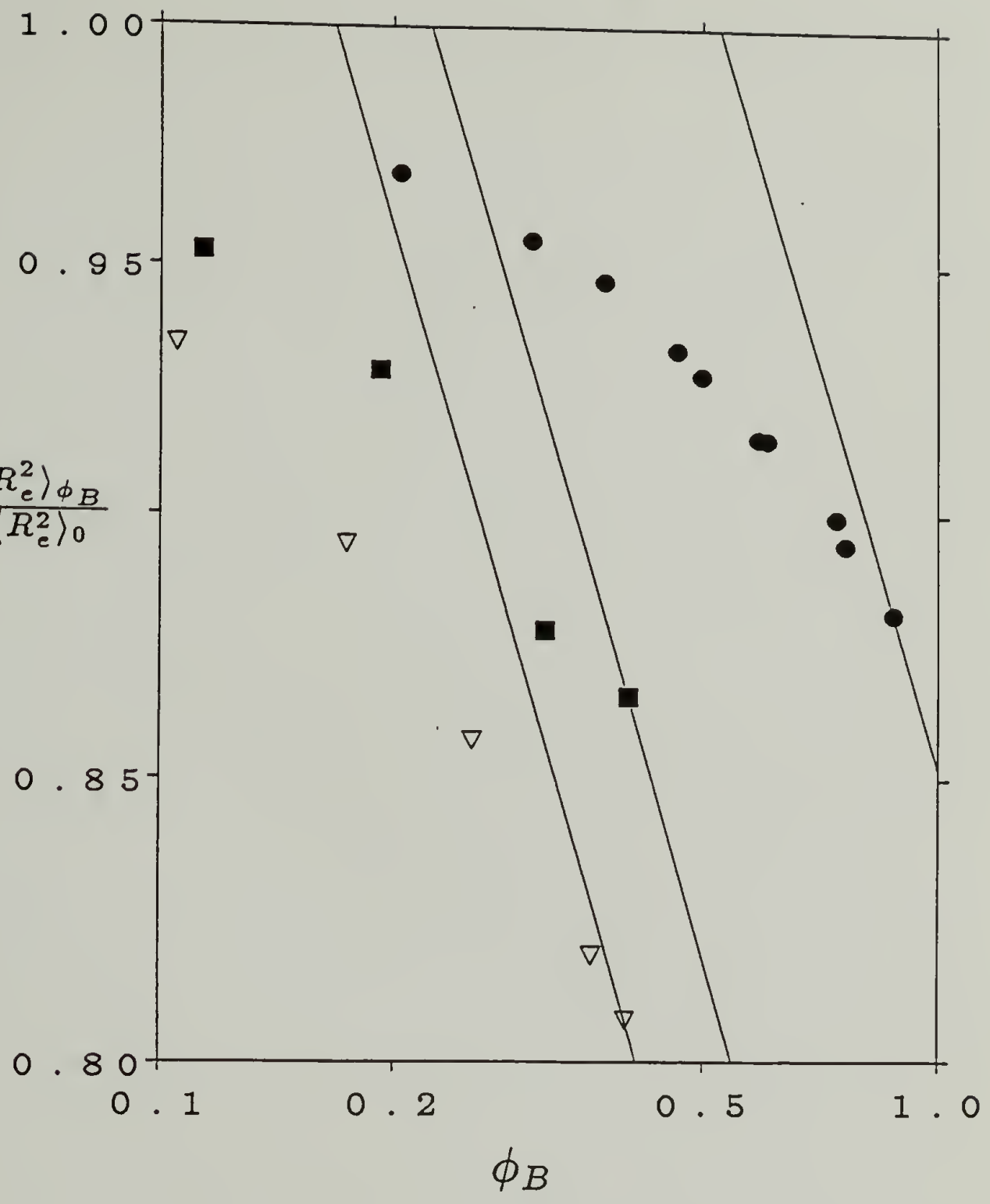

Figure 2.9

Comparison of the results for the mean squared end-to-end distance with those of Bellemans and de Vos [1973].

Figure 2.9, which comprises data from Bellemans and de Vos [1973] plotted with data from these simulations for chains of the same length, shows no agreement whatsoever with Daoud's relation. What Figure 2.9 does indicate, however, is that, as the chain length is increased, a closer approach to the scaling form predicted by Daoud, *et al.* occurs. Thus, the simulations may be flawed by not yet investigating chains that are long enough to represent this scaling form. But, the agreement of shorter chains with the scattering data, and the fact that the concentrations investigated here for the chains having $n = 51$ are an order of magnitude greater that the expected onset of the semidilute region (see below) means that the results are internally inconsistent. This again points to severe difficulties with the lattice simulations, and emphasizes the need for a complete consideration of the influence of relative sizes on the results from simulations. (See §4.2.1)

The approach at the higher concentrations studied to the predicted scaling forms for the semidilute region requires one to answer the question, "At what point do the bulk concentrations ϕ_B in these simulations enter into the semidilute region?" To answer this, we present the "blob" argument as developed by de Gennes [1979, pp. 80–85].

The basis for the development is that, below an "overlap" concentration c^*, polymeric chains can be considered to be in a dilute solution in which each chain is effectively contained in its own "blob" and is isolated from the influences of the other chains. (See Figure 2.10.) When the concentration has reached $c = c^*$, then the blobs have just begun to touch, and the semidilute concentration region is entered. Above c^* the chains begin to intertwine with one another and, as the concentration of chains is increased, eventually become entangled enough to place one in the concentrated region.

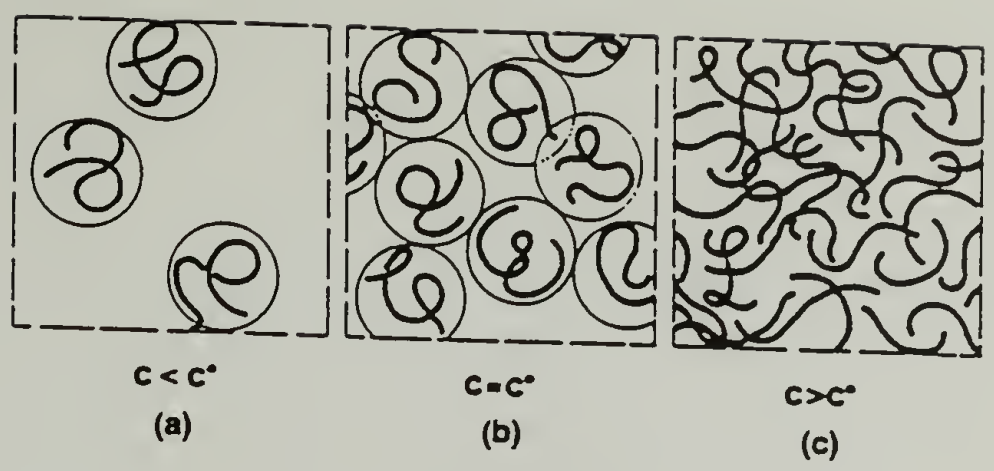

Figure 2.10
Crossover between Dilute and Semidilute Solutions.
(a) Dilute Solution; (b) Onset of Overlap; (c) Semidilute Solution.
Taken from de Gennes [1979, p. 77].

The transition between the dilute and semidilute concentrations is not abrupt and is really just a crossover between the regions. de Gennes states that the overlap concentration c^* is expected to be comparable with the local concentration inside a single coil. In an athermal solvent, like our simulations, this implies that

$$c^* \cong \frac{n_s}{R_F^3} = a^{-3} n_s^{1-3\nu} = a^{-3} n_s^{-4/5}. \qquad (2\text{–}19)$$

where ν is the usual Flory exponent, a is the Kuhn length, and n_s is the number of segments in a chain. Thus, as the chain length ($n_s = n - 1$) is increased, the overlap concentration decreases rapidly. This was, in fact, the reason that the system having $n = 51$ ($n_s = 50$) was investigated.

For these simulations, $a \equiv$ a Kuhn step $= 1$ so that the overlap concentration is simply related to the chain length as $c^* \cong (n-1)^{-4/5}$. A chain of $n = 11$ beads would then have $c^* \cong 0.2$, and a chain of $n = 51$ beads would have $c^* \cong 0.04$. Thus, even if one may be concerned that for the chains of 11 beads (10 segments) not enough data exists at high enough concentrations to show semidilute scaling behavior, all but one datum for the system of 51 beads *per* chain are above this system's estimate for c^* with only a hint of possible scaling behavior at a concentration an order of magnitude greater than c^*. Combining this lack of scaling behavior with the observation that, for *all* of the systems studied, the dilute

solution behavior followed the mean field trend, one finds that lattice simulations are poor at predicting and/or investigating the thermodynamic properties of dilute and semidilute solutions.

Simply stated, mean field theories are known to fail to predict the trends and scaling relationships observed in experiments for the dilute and semidilute regions. As such, if lattice simulations are following the trends predicted by mean field theories for these lower concentrations, then the lattice simulations will fail to predict the experimental trends at these concentrations.

2.4.2 Finite Size Effects.

Two systems studied, chains of 10 segments in length in a box of 17^3 sites and chains of 10 segments in length in a box of 21^3 sites, deviated from the mean field trends shown for the compressibility in Figure 2.4. Such a deviation is often caused by "finite size effects," in which the size of the system chosen for efficiency or convenience influences the physics obtained.

Table 2.1 shows the comparison of system sizes with geometrical data (in this case the mean squared end-to-end distance $\langle R_e^2 \rangle$ and the mean squared radius of gyration $\langle R_g^2 \rangle$). If the conformations available to the chains were restricted by the size of the periodic box, then, as the size of the box was increased, the ratio $\langle R_e^2 \rangle / \langle R_g^2 \rangle$ should continually decrease until the chains had reached their completely expanded shape. This is because the size of the box affects the end-to-end distance less than the radius of gyration. However, for these simulations this ratio remains effectively constant for all system sizes. Hence, there is no trend indicating that the size of the periodic box chosen is influencing the geometry of the system, thereby leading to erroneous results.

Using the value of 6.0 for the ratio $\langle R_e^2 \rangle / \langle R_g^2 \rangle$, one may calculate the radius of gyration for the smallest system (*i.e.*, 11^3 sites) as $\langle R_g^2 \rangle = 2.6$; this provides a height to radius of gyration ratio of 6.8, which is considered to be sufficient to

Table 2.1

Ratio of the height of the lattice H to the size of the molecules at 40.0% bead volume.

System n, H	$\langle R_e^2 \rangle$	$\langle R_g^2 \rangle$	$\frac{\langle R_e^2 \rangle}{\langle R_g^2 \rangle}$	$\frac{H}{R_e}$	$\frac{H}{R_g}$
11, 11*	15.7	[2.6]	[6.0]	2.78	[6.8]
11, 15[1]	15.8	2.65	5.95	3.78	9.22
11, 17	16.0	2.67	5.98	4.26	10.4
11, 21[2]	15.9	2.66	5.97	5.27	12.9
11, 25[3]	16.0	2.67	5.98	6.26	15.3
11, 35	15.9	2.67	5.98	8.77	21.4
51, 51	92.4	15.3	6.06	5.31	13.1

* The numbers in brackets are estimated. Refer to the text.

[1] at 42.3% bead volume.

[2] at 44.3% bead volume.

[3] at 41.7% bead volume.

avoid finite size effects in flexible chain systems. (See, for example, Bellemans and de Vos [1973] or Webman, Lebowitz, and Kalos [1980] in which the system sizes are much smaller with respect to the size of the molecule with no apparent influence on the results.) The absence of finite size effects is also supported by the radial distribution functions discussed in §3.5.1 where one sees that the local density is equal to the average density within 3 R_g of the observation point. With the smallest system being twice this size and the rest of the systems studied having height to $\langle R_g^2 \rangle$ ratios from two to five times larger, finite size effects (as commonly understood) *are not* the cause of the different trends observed for the 17^3 and 21^3 site systems. (See §4.2.1.)

2.4.3 Summary for the Lattice Simulations.

We claim that lattices are, in general, inappropriate to determine the thermodynamic properties of dilute and semidilute solutions. While the present data has only been obtained on a simple cubic lattice, similar agreement between the trends from simulation results and those given by mean field lattice theories have been found: Dickman [1987], Dickman and Hall [1986b, 1986c], and Okamoto [1976] on quadrilateral (square) lattices; Bellemans and de Vos [1973] on simple cubic lattices; Kolinski [1984] on face centered cubic lattices; as well as Okamoto and Bellemans [1979] and Kolinski, Skolnick, and Yaris [1986a–d, 1987a–c] on tetrahedral (diamond) lattices. As a group, these simulations include the most used types of lattices for simulations, and all follow the mean field trend over the whole range of concentrations studied.[17] As such, they also *all* fail to predict the semidilute scaling relations that are found in experimental systems.

---

[17] The study by Kolinski [1984] deviates quite strongly from the *values* predicted by Flory-Huggins theory, but the *trend* that defines the scaling relations is the same as for Flory-Huggins theory.

Admittedly, this argument does not *prove* that lattices *must* fail to predict the appropriate scaling forms, but the presentation does show that the most commonly used lattices, with present computational abilities, are inappropriate. One reason for this failure is probably an insufficient number of degrees of freedom available on a lattice. If this is the primary reason, then possibly something complicated, such as Skolnick's dodecahedral lattice [Vas, 1989], could eliminate this problem. (Note that extremely long chains on an infinitely large lattice should replicate continuous space.) Unfortunately, Skolnick has only looked at the bulk state using this lattice.

As a corollary, one should note that this inapplicability of lattice simulations to the lower concentration ranges implies nothing about the value of lattice simulations predicting properties of the bulk state. As mean field theory does work well at high concentrations, there is little reason to believe that lattice simulations, which apparently adhere to predictions of mean field theory throughout the concentration range, should fail to be appropriate for predictions of bulk state behavior. In fact, Kolinski, Skolnick, and Yaris [*cf.* 1986a, 1986c, 1987c] have done well correlating experimental data to results from lattice simulations of the bulk state.

Hence, the results from these simulations, and comparisons with others' work, indicate that lattice formulations, both in theory and in simulations, are unable to provide correct scaling and thermodynamic results in dilute and semidilute concentrations. This failure of lattice formulations has previously been either unnoticed or unreported. Finally, as our goal is still to predict and investigate the thermodynamic behavior of dilute and semidilute solutions, for both flexible and virgate molecules, we modified our simulations to be performed in three-dimensional, continuous space.

CHAPTER 3

SIMULATIONS OF SELF-AVOIDING CHAINS IN CONTINUOUS SPACE

With the failure of finding the appropriate semi-dilute scaling law relationships using lattice based simulations, the code was reconstructed to perform the desired simulations in continuous space, also called "free space" or "off-lattice" simulations. The belief was that the lack of enough degrees of freedom were influencing the results observed on a lattice, and certainly that this lack of freedom caused the lattice simulations to "freeze," *i.e.*, explore no new configurations, when the reduced energy parameter $\tau = k_B T/\epsilon_b < 1$. (See §2.4.1.)

However, moving to continuous space caused a serious problem: Dickman's algorithm [Dickman, 1987] could no longer be used. All of the problems associated with determining contact distances and adjusting for connectivity that were removed by Dickman's algorithm on a lattice (*cf.* §2.1) now returned. Previous simulations had shown that the convergence to a solution for the osmotic pressure by counting wall contacts was extremely slow even for simple spherical fluids with several million samples being needed [Baumgärtner, 1987; Snook and Henderson, 1978; Sullivan, Levesque, and Weis, 1980]. Such a system, even if convergence could be guaranteed for a polymeric system, would be prohibitively expensive in computer time, requiring several years (read "graduate lifetimes") to obtain results for even simple, dilute systems.

As such, we developed the code in two directions simultaneously: (1) We redesigned the canonical ensemble simulations to provide radial distribution functions; these may be related to the second virial coefficient, thereby providing thermodynamic information on dilute systems. (2) We developed an isobaric-isothermal ensemble simulation to provide the desired thermodynamic quantities directly for all concentrations. What will be shown below is that these two methods complement one another. While the isobaric-isothermal ensemble can be

applied over the whole range of concentration, the convergence of this ensemble's averages is slow at low concentrations, less than approximately 5% bead volume. This is the region of concentration in which the canonical ensemble's evaluation of the second virial coefficient becomes applicable. In addition, the canonical ensemble was used to present a novel solution to a previously unsolved problem: the *direct* calculation of the molecular scattering factor from a system of many chains.

This chapter is organized in the following manner: Descriptions of the mechanism used to move the chains will rely heavily on the discussion presented in Chapter 2. Philosophical considerations, *e.g.*, Metropolis sampling, will simply reference the previous section. Differences from Dickman's method will be pointed out in the context of a description of the canonical ensemble; the isobaric-isothermal ensemble needs no reference to Dickman's algorithm. The formulas for the radial distribution functions, molecular scattering factor, and compressibilities are simple and will be discussed only briefly. The primary emphases will be placed on the utility of these simulations and on the framework that they offer to investigate the thermodynamic and structural properties of solutions by using Monte Carlo simulations.

3.1 Basis for the Continuous Space, Canonical Ensemble Simulations.

The three primary quantities of interest to obtain from these simulations are the beads' radial distribution functions, the center of masses' radial distribution function, and the chains' scattering function. The first of these comes in a number of forms, but formulas for the center beads' and end beads' radial distribution functions will suffice; these distribution functions are mainly used for gaining a sense of the environment in which a bead at a particular position on a chain finds itself. Comparisons among various positions along the chain provide insight into the relative amount of freedom available to a bead at a given position and its comparison to other positions. These are *not* the same as the two body radial

distribution functions that appear throughout the development of the statistical mechanics of simple fluids. For example, these functions *cannot* be integrated to provide the molecular structure factor; the reason for this will be explained. The second quantity, the center of masses' radial distribution function, is used both to gain insight into the environment of the fluid as a whole, primarily providing qualitative information concerning how much the chains are interacting, and most importantly to permit the determination of the second virial coefficient for a system from which the osmotic pressure and the compressibility may be obtained. The third quantity, the chains' scattering function, is an end in itself, which, by varying the value of the wavevector investigated, can supply information in the range of both X-ray scattering and neutron scattering for solutions.

3.1.1 Beads' Radial Distribution Functions.

There are several variants for beads' radial distribution functions that may be investigated, and several types were determined. The only ones presented here are those found to be most useful in gaining a sense of the environment for the chains. These are the "complete" radial distribution function, which gives the averaged environment around any point in space; the center beads' radial distribution function, which provides the averaged distribution of the beads on a single chain while observing from the center bead of that chain; and the end beads' radial distribution function, which provides the averaged distribution of the beads on a single chain while observing from an end bead on that chain. These latter two distributions are plotted together for comparison.

The formal development for the radial distribution function with its dependence upon probabilities and density distributions may be found in McQuarrie [1976, pp. 257–259]. For the present purposes one needs only to recognize that the term $\overline{\phi} g(r) 4\pi r^2 \, dr$ represents the number of particles between r and $r + dr$ from some reference point, where $\overline{\phi}$ is the mean density for the system, $g(r)$ is the

radial distribution function, and $r = |\mathbf{r}_p - \mathbf{r}_r|$ is the distance between the position of some particle $\mathbf{r}_p$ and the position of the reference point $\mathbf{r}_r$. Stated another way, $\overline{\phi} g(r) d\mathbf{r}$ is the probability of finding a second particle within a spherical shell of thickness $d\mathbf{r}$ given that the reference point for the observation is at the origin. In either case, the relation that follows from this is

$$\int_0^\infty \overline{\phi} g(r) 4\pi r^2 \, dr = N - 1, \tag{3-1}$$

where N is the number of particles in the system. The term $N - 1$ arises from the fact that some particle is taken as a reference and is therefore not counted. Obviously, in an experimental system where N is on the order of 10^{23}, $N - 1 \approx N$.

An advantage of simulations is that, since one is able to observe the positions of the particles exactly, one may investigate the contribution to a quantity from different particles. This concept will be more important in the presentation of the scattering results, but may be shown here as an example. The complete radial distribution function can be split into the contribution from beads on the same chain as the observer $g_s(r)$ and the contribution from beads on other chains $g_o(r)$. Such a division leads to the following relation:

$$\begin{aligned}
\int_0^\infty \overline{\phi} g(r) 4\pi r^2 \, dr &= \int_0^\infty \overline{\phi} [g_s(r) + g_o(r)] 4\pi r^2 \, dr \\
&= [4\pi\overline{\phi} \int_0^\infty g_s(r) r^2 \, dr] + [4\pi\overline{\phi} \int_0^\infty g_o(r) r^2 \, dr] \\
&= [(n-1)] + [n(N_c - 1)] \\
&= nN_c - 1 = N - 1,
\end{aligned} \tag{3-2}$$

where n is the number of beads *per* chain and N_c is the number of chains in the system. Thus, as the concentration is varied, for example, one would be able to compare how much of the radial distribution is being contributed by other chains as they entwine with the reference chain, that is, the chain containing the reference bead.

Along similar lines are the center beads' radial distribution function $g_c(r)$, in which the center bead on a chain serves as the reference site, and the end beads'

radial distribution function $g_e(r)$, in which both end beads of a chain are used as reference sites. In both of these cases, only the positions for beads on the same chain as the reference beads' were compared. Also, the values for $g_c(r)$ and $g_e(r)$ were obtained by averaging the intrachain positions from all of the chains over all of the Metropolis samples taken.

The important point to remember is that these two radial distribution functions only serve as qualitative indicators of the environment in which a chain is found, though these functions are useful indicators. As will become clear below, these radial distribution functions *cannot* be used as the basis for Fourier transforms by which to obtain structure factors or similar quantities.

3.1.2 Center of Masses' Radial Distribution Function.

The development of the center of masses' radial distribution function $g_{cm}(r)$ is completely analogous to the beads' radial distribution functions. The change is that the reference points and the "particles" that one observes are just mathematical constructs rather than physical objects. The importance of this distribution function, however, is that it actually meets the requirements of a pair correlation function for certain applications as now the chain has been reduced to a point in space, very much like particles in a simple fluid. Hence, this radial distribution function may be used to determine the second virial coefficient for a solution of chains since the coefficient is a colligative quantity. On the other hand, this distribution may not be used to determine a structure factor as scattering does not arise from the placement of the center of mass of a molecule, but from the scatterers within that molecule.

The relation of the center of masses' radial distribution function to the second virial coefficient A_2 is given by

$$A_2 = 2\pi \int_0^\infty [1 - g_{cm}(r)] r^2 \, dr = 2\pi R_g^3 \int_0^\infty [1 - g_{cm}(r)] (\frac{r}{R_g})^2 d(\frac{r}{R_g}), \qquad (3\text{–}3)$$

where $R_g \equiv \sqrt{\langle R_g^2 \rangle}$ is the root-mean-squared radius of gyration for the chains. The introduction of R_g causes the integral to become dimensionless and to be on the order of unity. This places all of the system dependent length scales into a quantity separate from the radial distribution function, which is itself independent of the size of the system. This change is important for simulations because, defining the chain density $\rho_c \equiv N_c/V$, the quantity $A_2\rho_c$ is independent of the size for the simulated system, but the second virial coefficient A_2 is not.

Finally, from this radial distribution function, one may obtain an approximation for the compressibility Z of a system through the relation

$$Z \equiv \frac{P}{\mathrm{k}_B T\ \rho_c} = \frac{P\ V}{\mathrm{k}_B T\ N_c} \simeq 1 + A_2^{\circ}\rho_c, \qquad (3\text{–}4)$$

where A_2° is the true second virial coefficient determined by extrapolating the values obtained from Equation 3–3 to $\rho_c = 0$. As will be shown later, this approximation is only valid to a maximum of 5% bead volume, which restricts the usefulness of Equation 3–4 considerably.

3.1.3 Chains' Scattering Function.

This is the most important quantity determined from the canonical ensemble simulations, as is shown in the results section (§3.5), but the basis for the chains' scattering function is simple. The conversion of this simple idea into a useable simulation is not straightforward, however, and is fully developed in Appendix E.

The scattering function, $S(|\mathbf{q}|)$, is given by the relation

$$S(|\mathbf{q}|) \equiv \frac{1}{V} I_{\mathrm{i}}(|\mathbf{q}|) = \frac{1}{n^2\ V} \langle f^2 \sum_{j=1}^{N} \sum_{k=1}^{N} \exp[i\mathbf{q} \cdot \mathbf{r}_{jk}] \rangle, \qquad (3\text{–}5)$$

where $I_{\mathrm{i}}(|\mathbf{q}|)$ is the ideal intensity of scattering, n is the number of beads *per* chain, V is the volume of the system, f is the atomic form factor and is constant here since all of the scatterers are of the same type, $\mathbf{q}$ is the wavevector, $i = \sqrt{-1}$, and

$\mathbf{r}_{jk} = \mathbf{R}_j - \mathbf{R}_k$ is the difference vector between scatterers j and k [*cf.* Alexander, 1969]. Note that the normalization is done by the factor n^2, not N^2. This factor is used to be consistent with single chain scattering calculations; the actual factor used for normalization is of little consequence.

3.2 Basis for the Continuous Space, Isobaric-Isothermal Ensemble Simulations.

The construction of an isobaric-isothermal ensemble is not more difficult than a canonical ensemble mathematically, but can be more difficult to simulate. An isobaric-isothermal ensemble has the number of particles in the system N, the temperature T, and the pressure P as constants. The volume V is varied while monitoring the energy of the system to determine if the pressure input to the simulation leads to an equilibrium volume in accordance with the relation

$$V = -\mathrm{k}_B T \left(\frac{\partial \ln \Delta}{\partial P}\right)_{N,T} , \qquad (3\text{–}6)$$

where $\Delta = \Delta(N, T, P)$ is the partition function for the isobaric-isothermal ensemble. Stated another way, in addition to the energy fluctuations from the varied conformations of the chains, PV work is now also done on the system, thereby complicating the simulation.

The essential quantity to be obtained from this ensemble is the compressibility Z over a wide range of concentrations. With $N = nN_c$, T, and P all constant, the determination of the equilibrium volume V immediately provides one with Z through the equation

$$Z \equiv \frac{P\ V}{\mathrm{k}_B T\ N_c}. \qquad (3\text{–}7)$$

Obtaining values for the compressibility returns us to the original purpose of determining thermodynamic data and scaling forms from simulations of dilute, semi-dilute, and concentrated solutions; this is the purpose for which lattice simulations failed to give the appropriate experimental relations.

3.3 Method for the Continuous Space, Canonical Ensemble Simulations.

Consider a canonical ensemble, in which the number of particles in the system N, the volume V, and the temperature T are all constant. The volume is constructed of sides of length $L = 1$ so that the system has a unit volume, *i.e.*, $V = 1$. The center of the box formed by these sides resides at the Cartesian point $(x, y, z) = (0, 0, 0)$ so that the symmetry of the system may be used. Thus, the reference volume has dimensions $-\frac{1}{2} \leq x_i \leq \frac{1}{2}$, where x_i represents any of the coordinates x, y, or z, *and* the volume is periodic in all directions. That is, there are no "hard walls" in the z-direction, which were required in Dickman's method (*cf.* §2.1).

Into this volume are introduced chains that consist of hard spheres connected by rigid links. (For a discussion of the relation of these "chains" to true macromolecules, see §2.2.) This is commonly referred to as the "pearl necklace" model. The centers of the spheres serve as sites about which the rigid rods are able to rotate freely. The spheres provide the constraint of excluded volume, and thereby have the chains obey self-avoiding walk statistics. The length of the links is set so that the excluded volume of the beads prevents the possibility of the links crossing through one another. This constraint fixes the maximum link length to be $\sqrt{2}D$, where D is the diameter of a bead, and restricts the aspect ratios available to be between $(n-1)$ and $\sqrt{2}(n-1)$, where n is the number of beads *per* chain.

The limitation that the links should not cross during a move is unnecessary. As this is a Monte Carlo simulation in which the objective is to sample as much of phase space as possible, allowing the links to pass through one another (*i.e.*, using "phantom links") just permits access to different states of phase space more easily. Tests were made to determine if having phantom links affected the results. Seeing no difference in the trends observed [Starry, 1989], these simulations retained the length of a link as $\sqrt{2}D$ to facilitate a possible conversion of this code for Molecular Dynamics simulations, where phantom links are not permitted.

3.3.1 Initialization of the Ensemble.

The initialization of the system was similar to the last method mentioned for the lattice simulations (*cf.* §2.3.1). All of the initial beads for the chains were placed in a box that had sides of length $L = 0.5$. The centers of the beads could be placed anywhere within the boundaries of $[-\frac{1}{4} + \frac{1}{2}D] \leq x_i \leq [\frac{1}{4} - \frac{1}{2}D]$, where x_i represents any of the Cartesian coordinates x, y, or z, and none of the beads could overlap. After all of the initial beads were placed, their coordinates were multiplied by a factor of two so that the initial box was of the size of the reference box, each side having length $L = 1$.

A second bead was added to each chain by randomly choosing a direction in three-dimensional space. This was done by choosing a polar angle φ in the range $0 \leq \varphi < 2\pi$, using a uniform random number generator,[1] and an azimuthal angle θ *via* the constraint $-1 \leq \cos\theta < 1$, using a uniform random number generator, with $\sin\theta = \sqrt{1 - \cos^2\theta}$. This method of selecting a random direction avoids the common mistake of just choosing the polar angle φ and the azimuthal angle θ uniformly from within their respective ranges. This latter, misguided procedure heavily weights the choices toward the polar regions; this can be shown from considerations of the solid angle being accessed.

After selecting a new position for the second bead of each chain, a check was made for excluded volume failures. As the probability of such failures was low, because there is a great deal of freedom in three-dimensional space, if a failure did occur, a new direction was chosen to place the second bead. No further manipulations were deemed to be necessary.

When all chains consisted of two beads, the procedure was just to add all of the remaining beads to one chain before propagating any other chain, with the excluded volume test being performed at each addition, of course. This simple

---

[1] The uniform random number generator used for the canonical ensemble, continuous space simulations was `RANF()`, a supplied function call on the CYBER 205 and ETA^{10}.

method was adequate for initializing the systems studied with the canonical ensemble. However, this method begins to have trouble at about 20% bead volume, and different schemes then need to be applied. (See §3.4.)

At the end of the initialization, the direction vectors connecting the beads ("link vectors"), the angles between the links ("link angles"), and the center of masses for the chains were calculated. These are the quantities that need to be monitored and altered as the chains are moved.

3.3.2 Generation of the Chains' Conformations.

Reptation was used as the mechanism for movement, just as in the lattice simulations (*cf.* §2.3.2), because these are "high temperature" simulations; that is, the chains are quite contorted on average throughout the simulation, thereby having a high energy content. The difference between the lattice simulations and here is that, instead of simply having five choices for the lattice, one has an effectively infinite number of choices for the position of the next reptation step. This direction was chosen randomly by selecting a polar angle γ, such that $-\pi \leq \gamma < \pi$, using a uniform random number generator; and by selecting an azimuthal angle θ, such that $-1 \leq \cos\theta < 1$, where $\cos\theta$ is chosen using a uniform random number generator. By definition, $\sin\theta = \sqrt{1 - \cos^2\theta}$. This method of choice avoids the common error of simply choosing a polar angle γ and azimuthal angle θ randomly, a method that favors positions near the poles in the solid angle.

The full procedure for chain movement is as follows: The chain to be moved was chosen at random. The end to be moved was selected randomly. The angle γ and quantity $\cos\theta$ were chosen randomly; this defines the new position for the ultimate bead as the distance from its previous position must be equal to the link length. The new position is tested for violations of excluded volume. If no violations occur, the new position is accepted, and the chain is "slithered" along its length. (See §2.3.2.) Otherwise, the former conformation of the chain is retained.

3.3.3 Acceptance of the Systems' Configurations.

The tests used, and the assumptions involved, for accepting a new configuration of the chains are the same as for the lattice simulations. See Section 2.3.3 for a detailed discussion of the method.

The one significant difference is in how the "bend energy" for the chains is calculated, a quantity critical to the Metropolis test (*cf.* §2.3.3.1). In the lattice simulations, one simply counted the number of bends that existed in all of the chains and divided this number by the dimensionless bend energy $\tau = k_B T/\epsilon_b$ to determine the system's reduced energy. Since the freely jointed chains may have a link angle between $\cos^{-1}(\frac{\sqrt{2}}{4})$ and π, the bend energy ϵ_b was set equal to a function of the form $(\cos\alpha + 1)/2$. The dimensionless system energy then becomes

$$\frac{E}{k_B T} = \frac{1}{\tau} \sum_{j=1}^{N_a} \left(\frac{\cos\alpha_j + 1}{2}\right), \qquad (3-8)$$

where N_a is the number of link angles in the system. The key point is that, as the link angle approaches π, the bend energy decreases. Hence, a fully extended chain is in its lowest energy state, consistent with the lattice simulations.[2]

3.4 Method for the Continuous Space, Isobaric-Isothermal Ensemble Simulations.

Consider an isobaric-isothermal ensemble in which the number of particles N, the temperature T, and the pressure P of the system are all constant. The volume of the system V is allowed to vary so that the size of the system can reach an equilibrium volume for the value of P that was introduced initially. The actual procedure for the user is to monitor if the system's volume tends to be expanding or contracting over a period of time, and then to reset the initial pressure for

---

[2] The lower limit on the link angle $\alpha = \cos^{-1}(\frac{\sqrt{2}}{4}) \approx 69.3°$ arises from the fact that the link lengths are fixed at $\sqrt{2}D$. Any smaller angle would cause an excluded volume failure.

the system in the appropriate direction to try to reverse the trend on the next execution of the program. For example, if for an initial N, T, and $P = P_1$ the system volume $V = V_1$ is increasing over a long period (*i.e.*, several hundred to several thousand Monte Carlo steps), the pressure P_1 is too small for the rest of the parameters chosen. A new pressure $P_2 > P_1$ would be input to the program, and the system would be rerun. If the volume of the system V_2 now tends to decrease over a long period, then the desired pressure has been bracketed, and one may bifurcate the range of pressures and simulate until time, patience, or money have expired to determine the appropriate values for P and V for computing the compressibility. (See Eqn. 3–7.) If the volume still tends to increase, then a new pressure $P_3 > P_2 > P_1$ must be chosen.[3]

The procedure described here is tedious and "user intensive." At present the automatic manipulation of these systems has not been implemented as the system is extremely sensitive to the values of the parameters chosen. More experience needs to be gained in studying the sensitivity of the system's volume to the various parameters before an algorithm may be included to perform these adjustments. (See §4.2.2.)

The chains that constitute the system are the same as those for the canonical ensemble: the "pearl necklace" model in which hard spheres are connected by freely jointed, rigid links, thereby having chains that obey self-avoiding walk statistics. (For a discussion of the relation of these chains to true macromolecules, see §2.2.) Again, the length of the links are fixed equal to $\sqrt{2}D$, where D is the diameter of a bead, so that the excluded volume test for the beads also prevents the crossing of links. (See §3.3 for the implications of this limitation.)

---

[3] Experience has shown that this method is more effective, if unfortunately user intensive, than waiting for the simulation to stabilize on its own.

3.4.1 Initialization of the Ensemble.

The method for initializing the system is the same as that described in §3.3.1 with one modification. Instead of placing the initial beads randomly within the reduced reference box, this space is divided into "cells" into which only one bead may be placed. The distance between the placement of the beads is controlled by a user adjustable parameter `ALPHA`. The technique is outlined in the code of `ST_REP` found in Appendix C.

The reason for this modification is that, at concentrations above approximately 20% bead volume, a random placement of initial beads would cause some chains to be too close to one another such that the system would not create an initial configuration. If very dense systems are to be studied, *e.g.*, > 50% bead volume, then techniques such as Kolinski, Skolnick, and Yaris' simultaneous reptation and propagation methods [1986a] need to be used.

3.4.2 Generation of the Chains' Conformations.

The generation of the conformations for the chains is the same as that described in §3.3.2 with one addition (which, technically, does not affect the *conformation* of a chain). In any movement step, a chain was rigidly translated in space before being reptated. This was done to provide a more efficient exploration of phase space and less correlation among samples. (See §2.3.3.3.) The amount of translation applied was small, on the order of a Kuhn length, so that this motion would rarely cause an excluded volume failure.

3.4.3 Acceptance of the Systems' Configurations.

The acceptance of the configurations is the same as that described in §3.3.3. The equation used to determine a system's energy has been modified from Equa-

tion 3–8 to include the contribution from the PV work *via* the term PV_t/k_BT, where V_t is the present (temporary) volume for the system. This results in the following relation:

$$\frac{E}{k_BT} = \frac{1}{\tau}\sum_{j=1}^{N_a}\left(\frac{\cos\alpha_j + 1}{2}\right) + \frac{P\,V_t}{k_BT}. \qquad (3\text{–}9)$$

As before, a fully extended chain is in its lowest energy state, consistent with the lattice simulations, though now a system of fully extended chains may not be in the system's lowest energy state.

3.5 Results and Discussion.

Conceptually, there are four different types of data being realized from the continuous space simulations: environmental conditions of the system (*via* beads' radial distribution functions), second virial coefficients (*via* the center of masses' radial distribution function), scattering results (primarily chain scattering functions *via* direct calculation), and compressibilities (*via* isobaric-isothermal ensemble simulations). There is much information to be gleaned from each of these types. As such, a number of graphs have been included for support or clarity in addition to the information that is distinctly desired.

3.5.1 Beads' Radial Distribution Functions.

The first results presented are of a qualitative nature: the bead correlation functions that allow one to observe the local environment of the chains, and particularly bead positions within a chain, *e.g.*, center beads *versus* end beads. Figures 3.1, 3.2, and 3.3 show the most informative of the many types of these functions that can be made: composite, or "complete," pair correlation functions. These graphs show the average, generic environment that any bead would experience

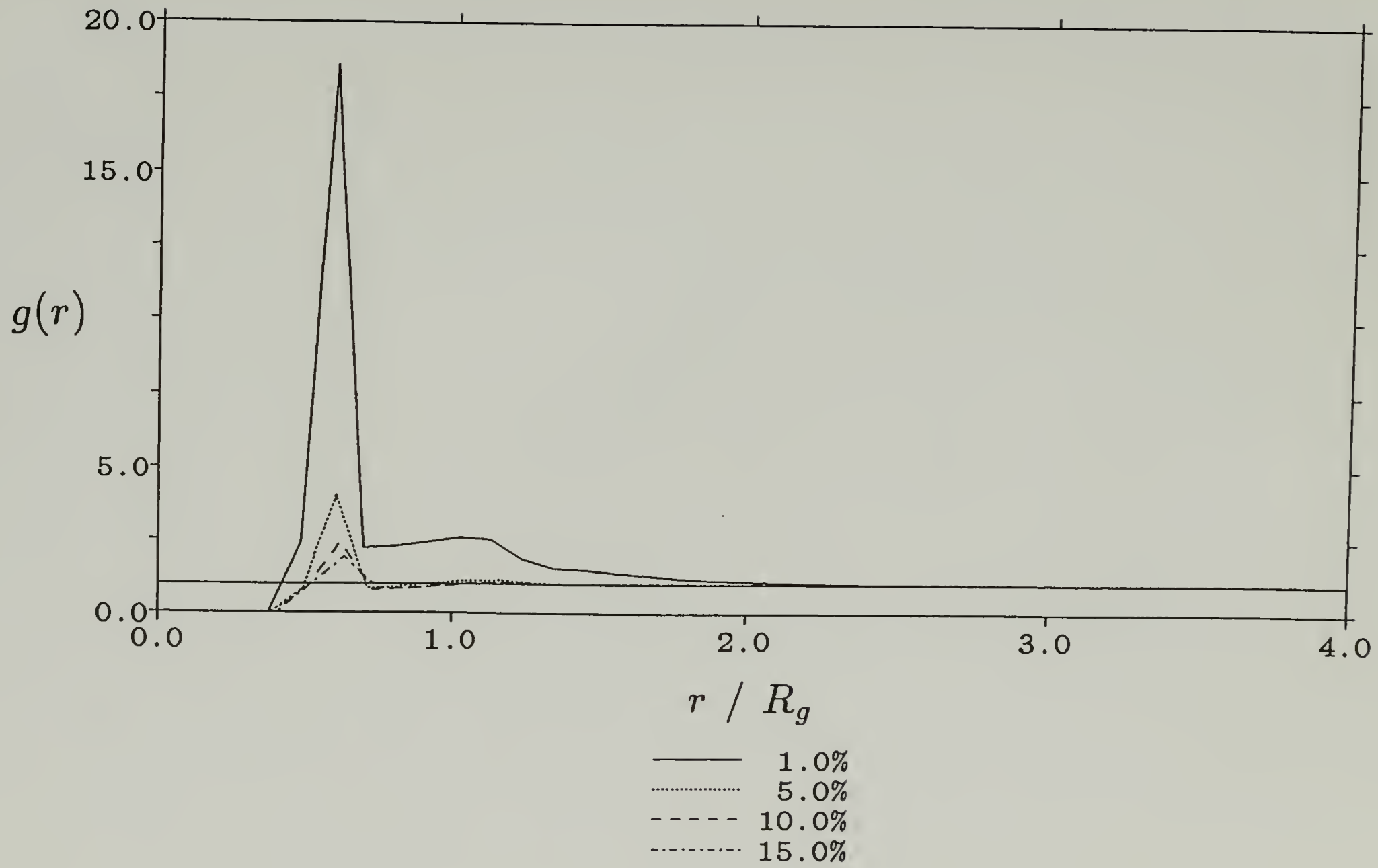

Figure 3.1
Complete bead-bead radial distribution function for various bead volume fractions.

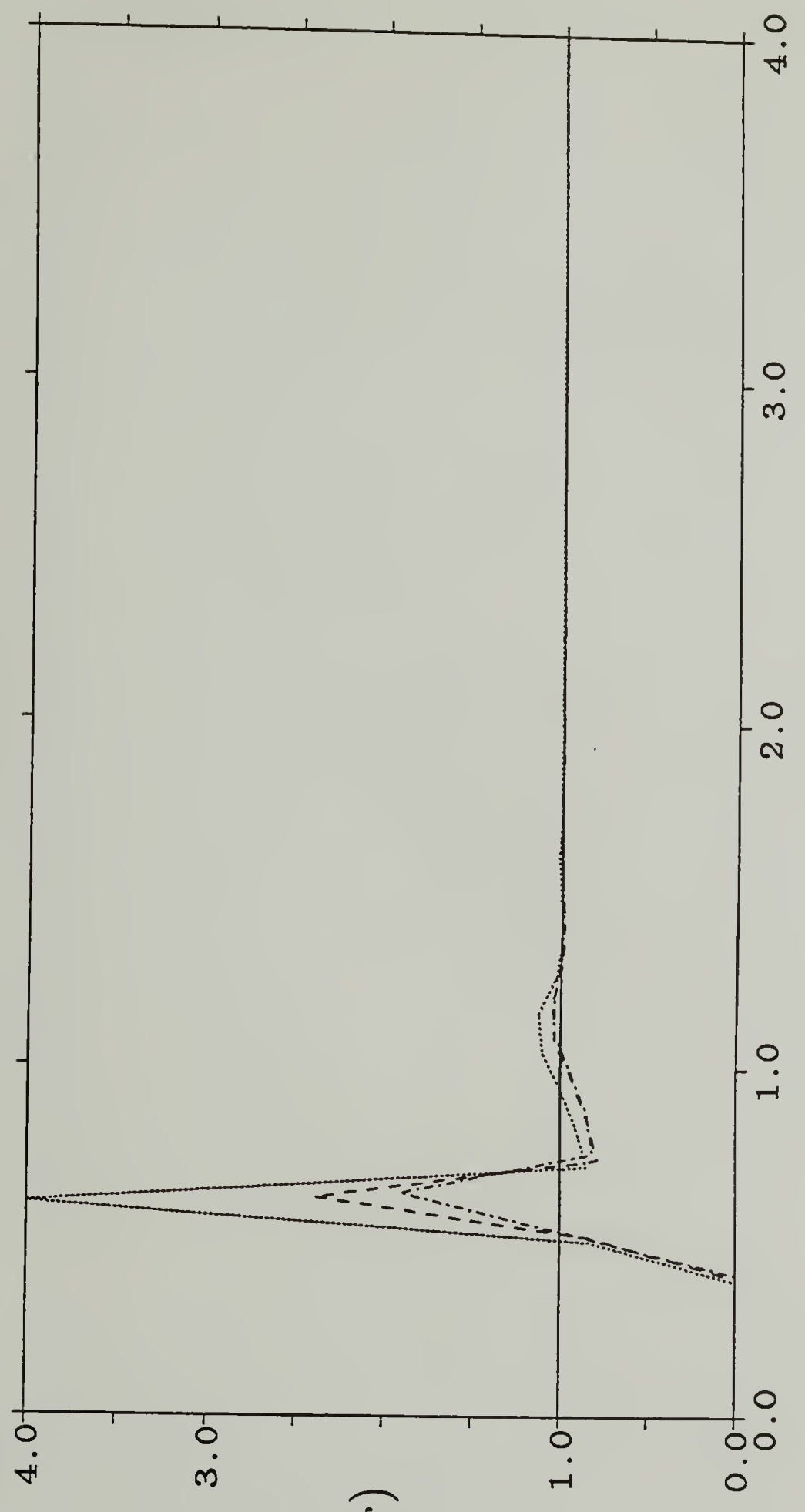

Figure 3.2

Complete bead-bead radial distribution function for various bead volume fractions. 1.0% bead volume results have been removed for clarity.

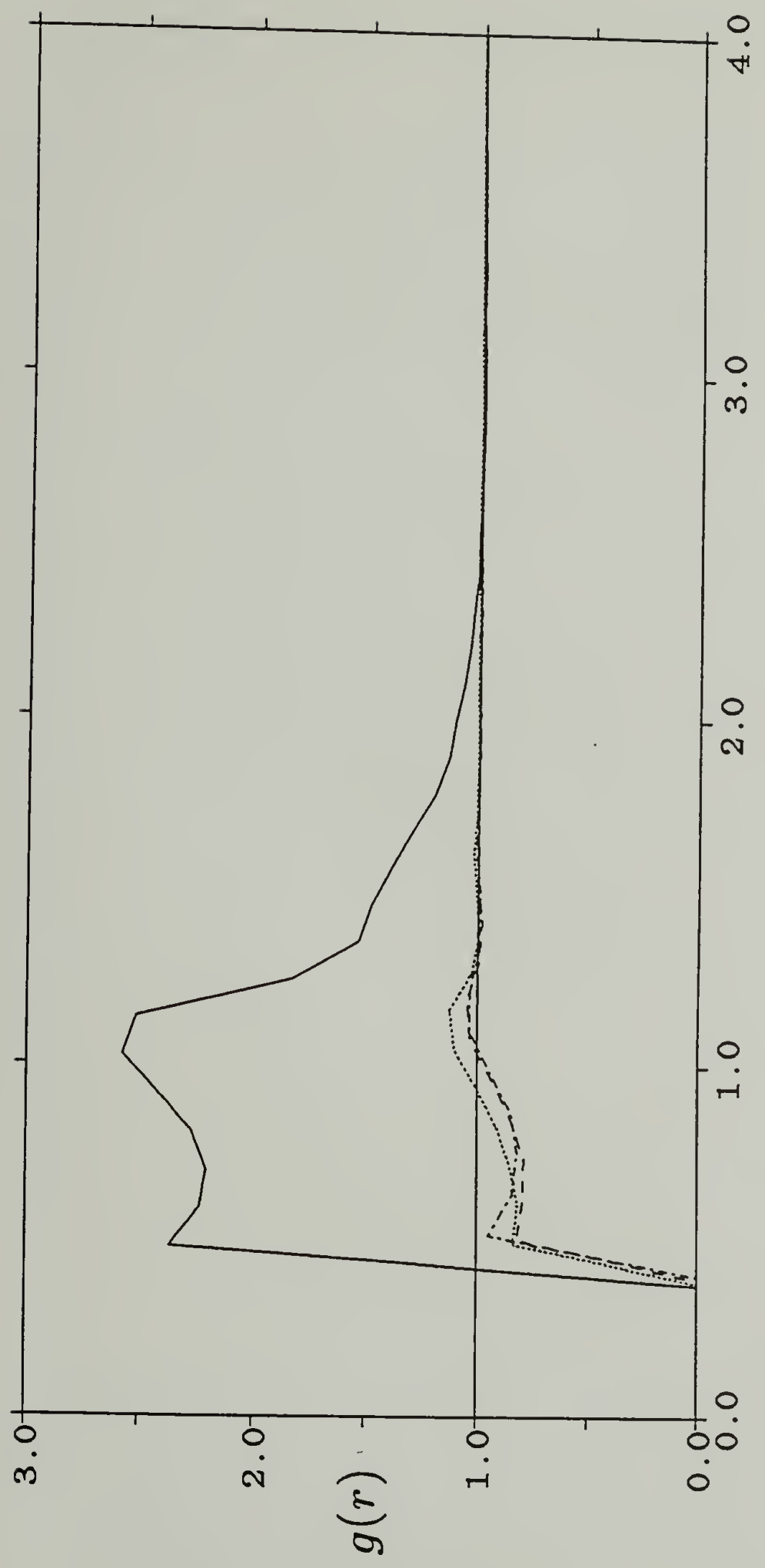

Figure 3.3

Complete bead-bead radial distribution function for various bead volume fractions. Contribution from the connected, nearest neighbors has been removed.

over a large number of configurations and provide a simple, quickly interpreted representation of that environment.

While there is little quantitative information to be gained from these plots, a few observations should be made. First, Figure 3.1 shows that the first peak at about 0.6 r/R_g shifts to higher values of r as the concentration increases from 1.0% bead volume to 15.0%. (Figure 3.2 is the same data without the 1.0% bead volume to more clearly display the data from the other concentrations.) The reason for this shift may be interpreted in either of two ways: the root-mean-squared radius of gyration R_g is becoming smaller due to crowding of the chains, or, as the chains interpenetrate at the higher concentrations (better demonstrated in §3.5.2), the chains straighten causing the most probable position to move farther from the observing bead. Second, the radial distribution functions have become equal to 1, which means that the local density is equal to the bulk density, before $r/R_g = 3$. This lends credence to the claim that, in the lattice simulations, finite size effects were not influencing the results. Referring back to Table 2.1, the ratio of the box length to R_g was estimated to be 6.8 for the smallest case (*i.e.*, $n = 11$ in a box with 11^3 sites). If the bulk density is reached within one-half of this distance, the ratio of the box size to the chain length is not contributing to the results obtained. Third, the trends shown by the more concentrated systems in Figure 3.2 in which the radial distribution function oscillates over and under the $g(r) = 1$ line are consistent with the radial distribution functions determined by Longman, Wignall, and Sheldon [1976] on molten polyethylene.

Figure 3.3 is presented to assist in qualitatively determining how much of the primary peak is ascribable to attached neighbors. Because of the constraint of connectivity within a chain, any end bead must be connected to one neighbor and any central bead to two neighbors; hence, the environment of any bead is greatly influenced by its attached neighbors. At a distance equal to the length of the connecting link, the radial distribution function will have an immense contribution from the connected neighbors. That is why the 1.0% bead volume's primary peak so much greater, relatively, to the rest of its radial distribution function than for

the other concentrations: more of the 1.0% sample's primary peak is derived from connected neighbors as there is little interpenetration of the chains in that system. This necessary artifact of the model obscures the information as to how much unconnected beads contribute to the number density in that particular radial shell. Figure 3.3 removes the contribution of the connected neighbors from the curves shown in Figure 3.1 so that one may see the relative contribution of unconnected beads at that length. (One will note the much smaller difference in amplitude between the 1.0% radial distribution values and the other concentrations in Figure 3.3.)

Another utility of beads' radial distribution function is to gain a feel for the difference in the environment experienced by beads at different points on the chain. Figures 3.4 through 3.7 represent the difference in the environment between what is experienced by a bead at the center of a chain and what is experienced by one at an end of a chain. Recall that these are high temperature simulations where the chains are very flexible; as such, there is only a small difference in the environment for a chain in a 1.0% solution (Figure 3.4) and a 15.0% solution (Figure 3.6) though the greater interpenetration of the chains at 15.0% again causes the primary peak to shift to longer distances (*i.e.*, larger r/R_g) and increases the relative contribution of the second peak. More difference in the environments of the two concentrations would be apparent for stiffer chains.

The key point to remember from these graphs is that they are qualitative in nature. These radial distribution functions should *not* be confused with the usual pair correlation function that is so important to developments of simple solution theories in statistical mechanics. The latter pair correlation function can be used to determine pressures, chemical potentials, and even structure factors. (See, for example, McQuarrie [1976, Chps. 12 and 13].) To properly determine the normal pair correlation function in these simulations, a separate monitor for every possible pair of beads in the system is required. Thus, in a system of two chains having five beads each, *i.e.*, ten total scatterers, and applying all possible symmetry relations, there would be 12 distinct types of pairs to monitor while,

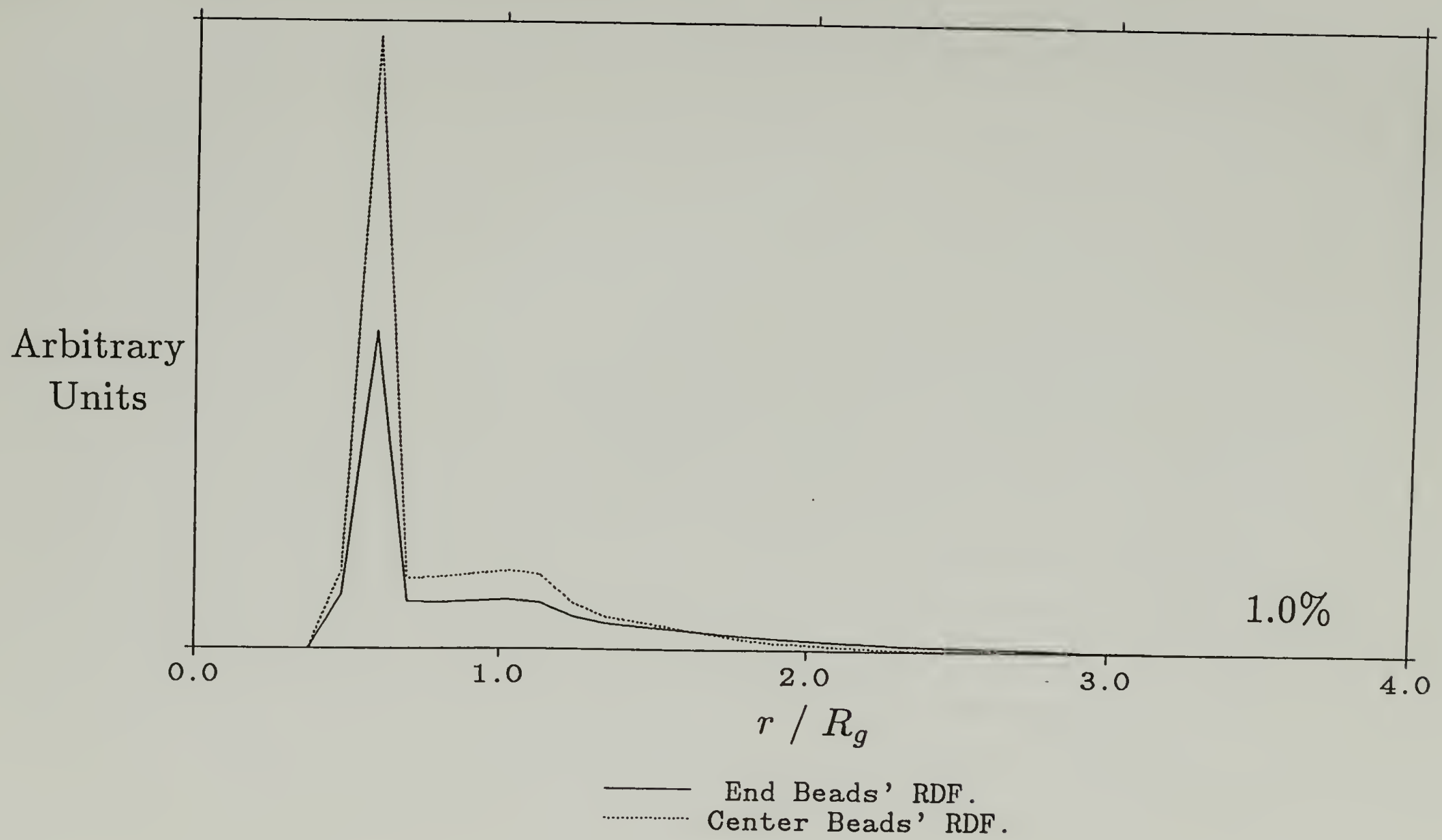

Figure 3.4
Comparison of the intrachain environment as experienced by an end bead and the center bead of the same chain at 1.0% bead volume.

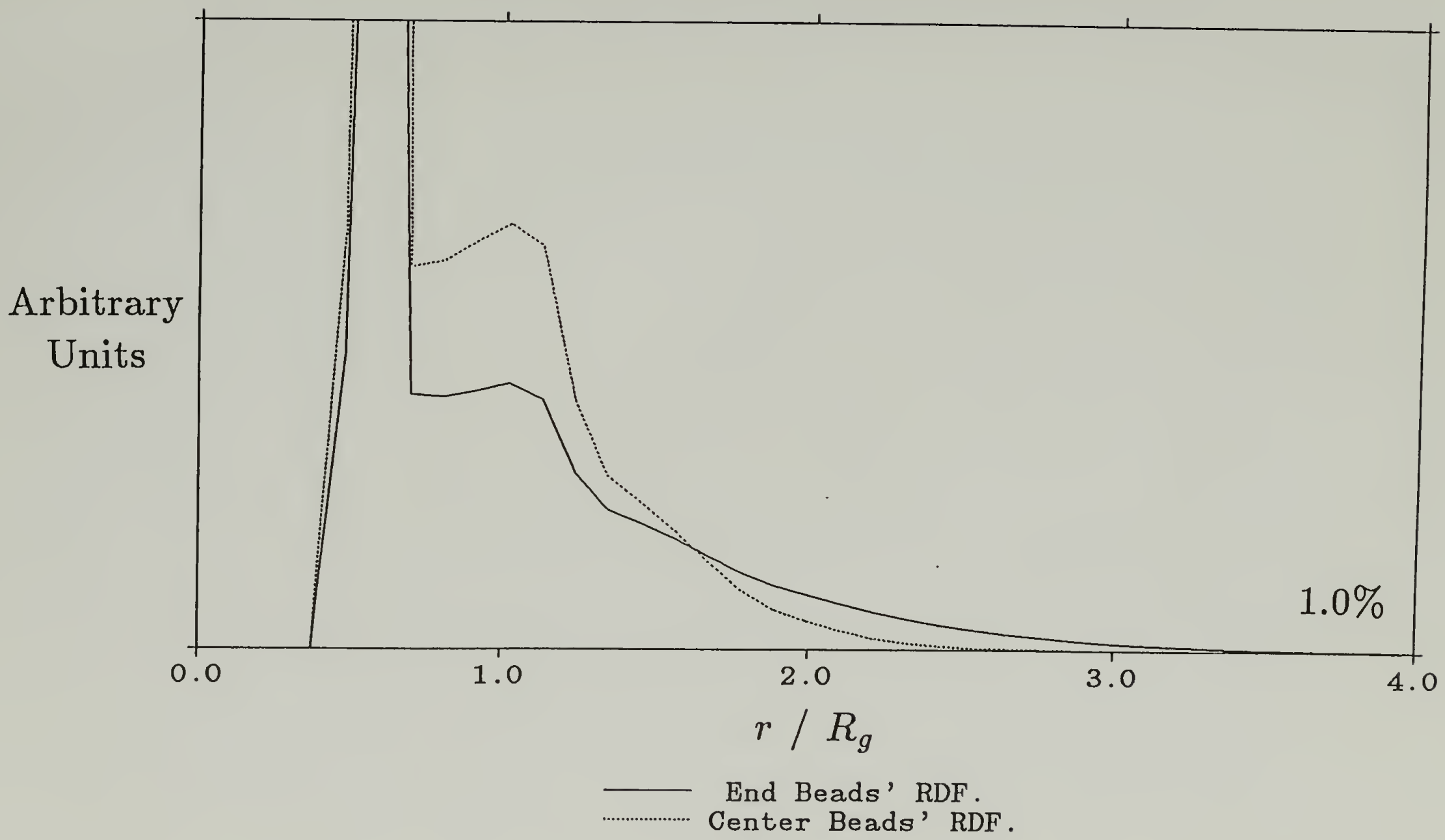

Figure 3.5
Comparison of the intrachain environment as experienced by an end bead and the center bead of the same chain at 1.0% bead volume; expanded scale.

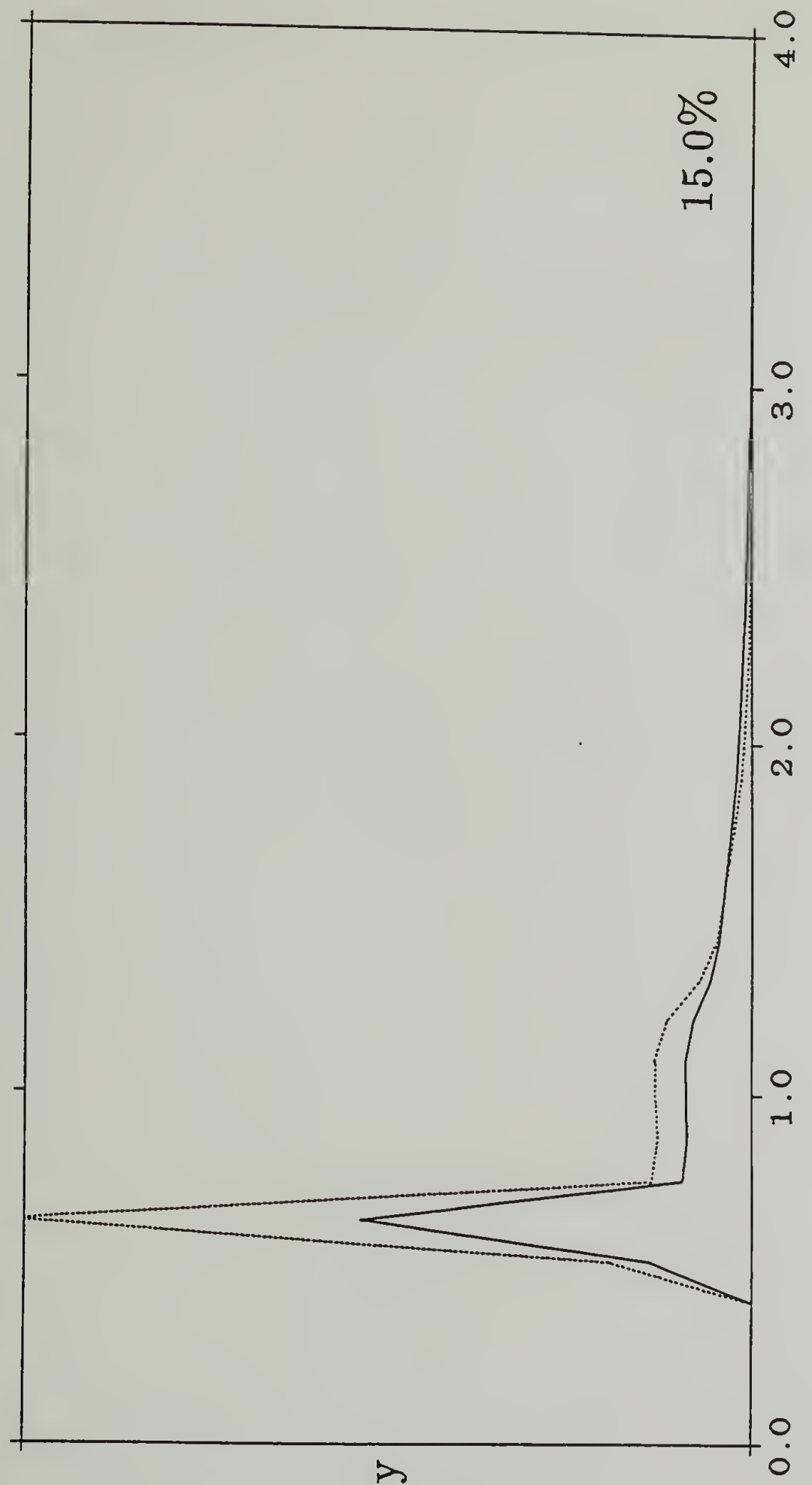

Figure 3.6

Comparison of the intrachain environment as experienced by an end bead and the center bead of the same chain at 15.0% bead volume.

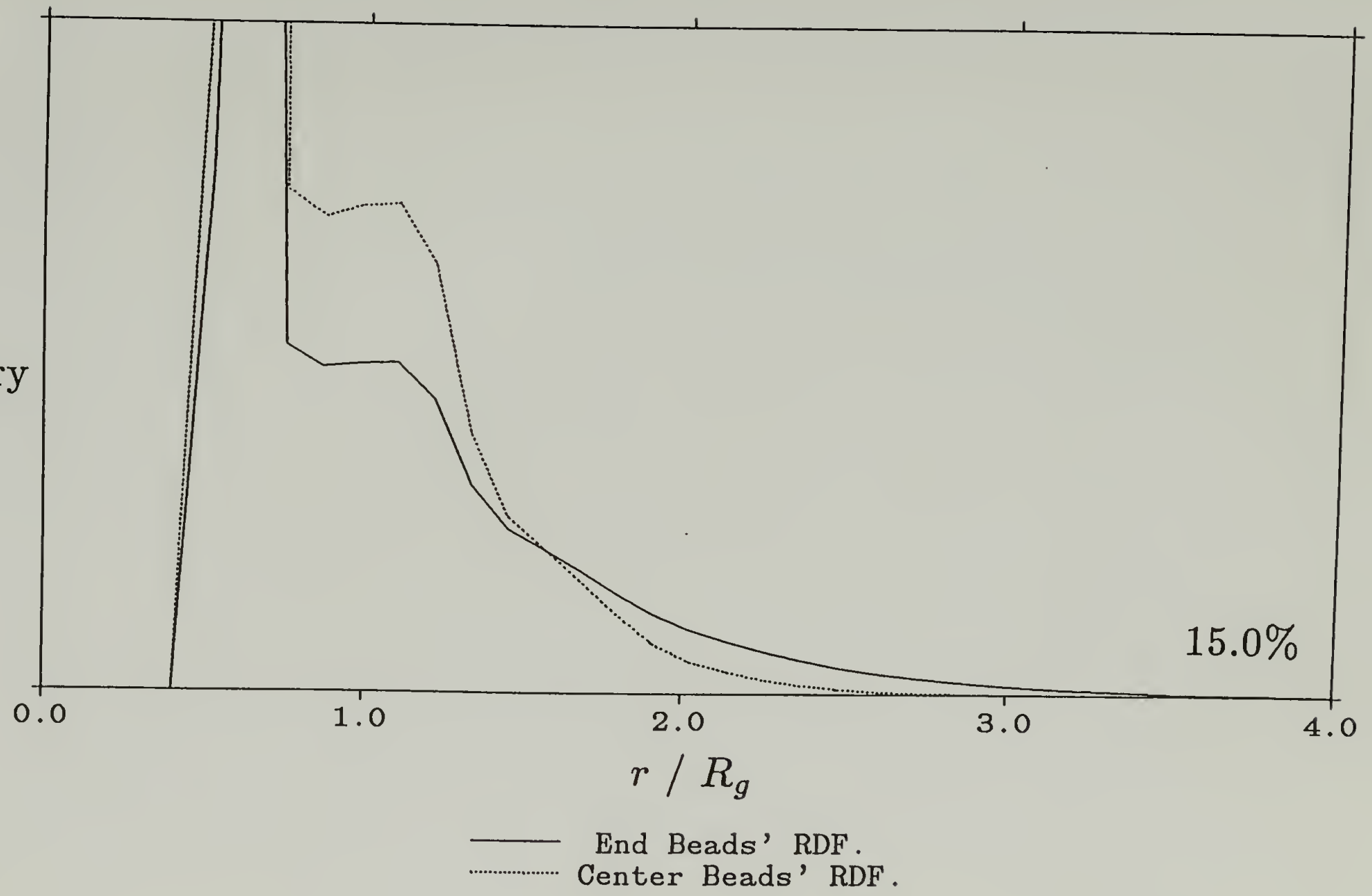

Figure 3.7
Comparison of the intrachain environment as experienced by an end bead and the center bead of the same chain at 15.0% bead volume; expanded scale.

for two chains having 11 beads each, 45 separate types of pairs exist. Not only is this bookkeeping computationally expensive, but one must then combine and numerically integrate the final functions to obtain pressures, chemical potentials, or structure factors such that the error in the measurement can quickly become large. Our direct method of calculation for pressures and the chains' scattering function avoids these difficulties. Simply stated, the radial distribution functions of the beads presented here really just provide insight into the local environment of the beads without providing any further thermodynamic information.

3.5.2 Center of Masses' Radial Distribution Function.

The second grouping of figures deals with the radial distribution for the center of masses of the chains. This provides two pieces of information. First, one again obtains environmental information as in the case of the beads' radial distribution functions. This information is easily interpreted as an indicator to how interpenetrated the chains are as a function of concentration. Second, and more importantly, this function may be integrated to obtain the second virial coefficient as presented in Equation 3–3. This is of particular importance since we still desire to determine thermodynamic properties of our systems, and the isobaric-isothermal ensemble does not work well at low concentrations (*cf.* §3.5.4).

Figure 3.8 shows how the environment changes with concentration. This figure shows that for higher concentrations, *e.g.*, 15% bead volume, there is a significant contribution to the center of mass radial distribution function $g_{cm}(r)$ at distances close to zero. In fact, the center of masses are closer than the diameter of a bead, a distance that the bead centers presented in Figures 3.1 to 3.7 could, of course, not approach. This implies that the chains are becoming interpenetrated as their mathematical centers are extremely close.

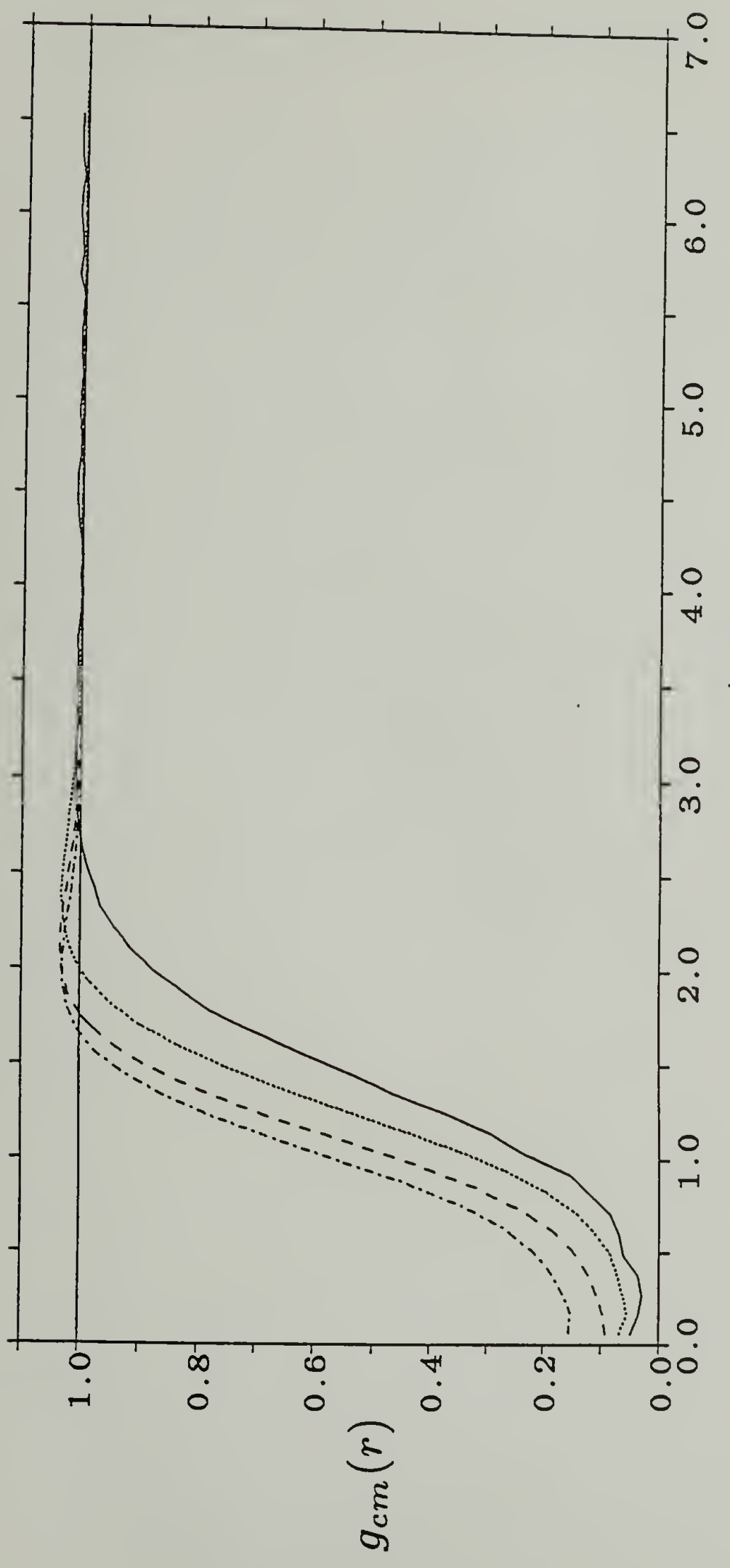

Figure 3.8

Center of masses' radial distribution function for various bead volume fractions.

Figures 3.9 through 3.12 are included to show the statistical error involved in these calculations.[4] Note that these are much smaller than the usual 10 to 20% errors associated with Monte Carlo simulations. While not all quantities can be determined with this precision, thoughtfully designed algorithms combined with the faster processors available can permit Monte Carlo calculations to be much more precise than is commonly accepted from simulations performed several years ago.

Figure 3.9 shows the 1.0% bead volume system, which has the most statistical error as would be expected since the fluctuations in the system are not damped by the presence of other chains; the chains in the 1.0% bead volume system can move about space quite freely with little possibility of an excluded volume failure. One may note that, according to the error bars, the 1.0% system's center of mass radial distribution function has not quite converged to unity at large distances as it should. As Figure 3.9 is the result of ten simulations, a prohibitively large number of simulations would be required to reduce the error significantly.

The second virial coefficients determined from these simulations are presented in Table 3.1. While their significance will be discussed further in §3.5.4, one will quickly realize why the simple canonical ensemble in continuous space is inadequate to provide the desired thermodynamic information at all concentrations. These simulations only provide reliable information to about 5% bead volume. No significant change is seen for the compressibility Z at higher concentrations. The difficulty is that, above about 5% in this system, higher order contributions become important as would be represented by third, fourth, fifth, *etc.* virial coefficients; the contributions represented by the second virial coefficient have become essentially constant.

This limitation on the range of concentration over which simple "two particle" interactions describe the system also restricts the use of the pressure equation

---

[4] All estimates of error throughout this document are reported at the 96.5% confidence level (2σ).

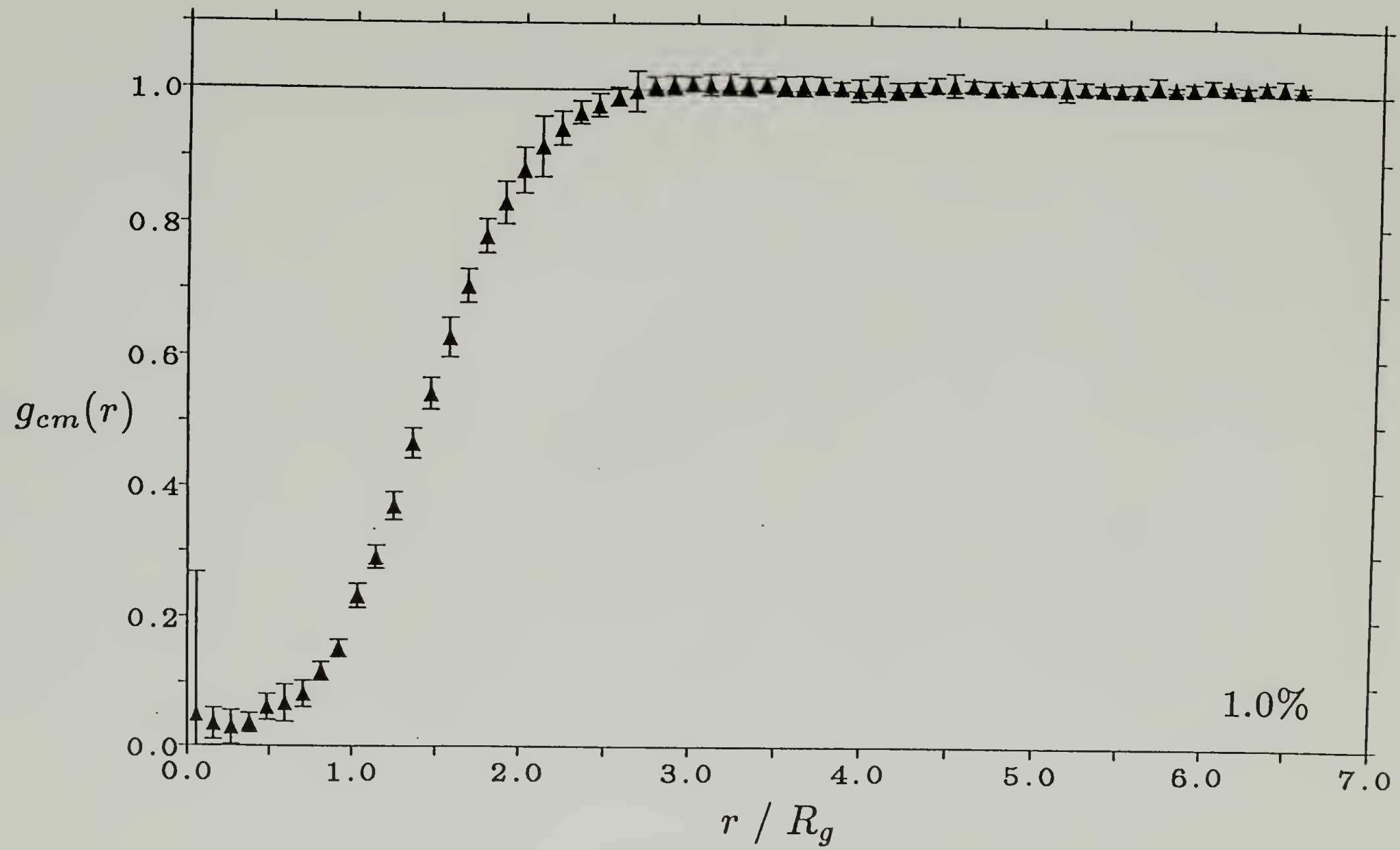

Figure 3.9
Center of masses' radial distribution function for a 1.0% bead volume solution.
Statistical error displayed at the 96.5% confidence level.

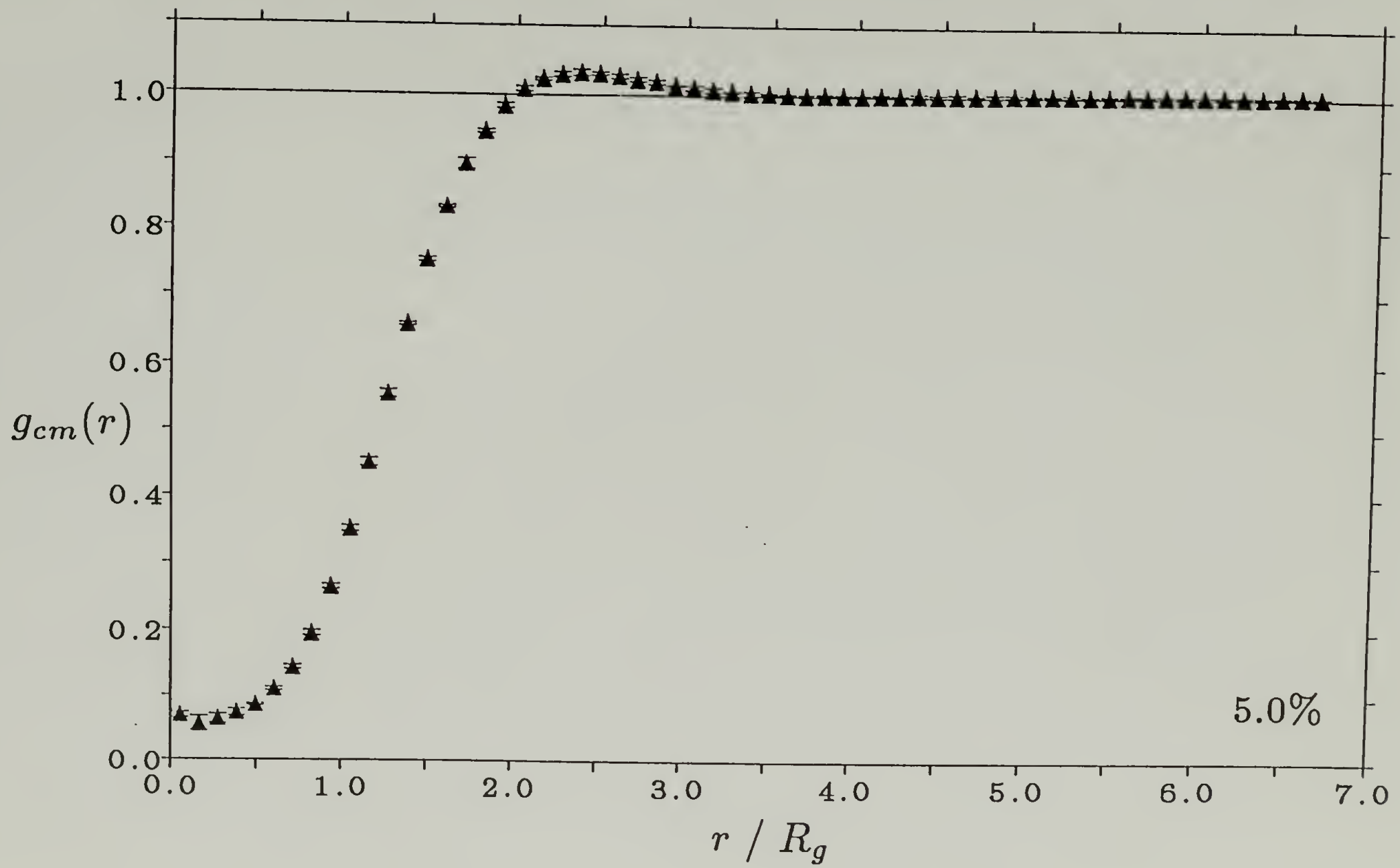

Figure 3.10
Center of masses' radial distribution function for a 5.0% bead volume solution.
Statistical error displayed at the 96.5% confidence level.

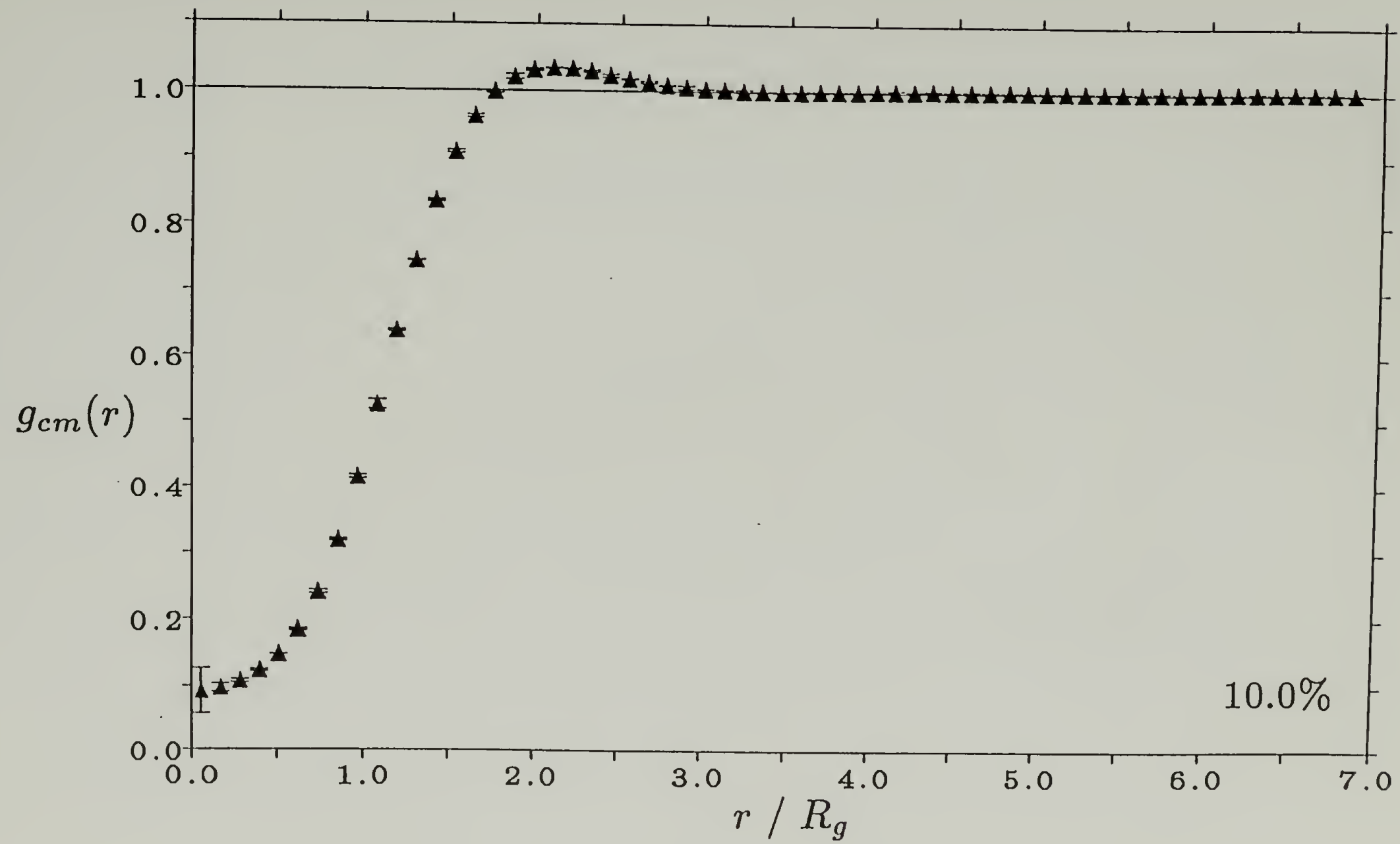

Figure 3.11
Center of masses' radial distribution function for a 10.0% bead volume solution.
Statistical error displayed at the 96.5% confidence level.

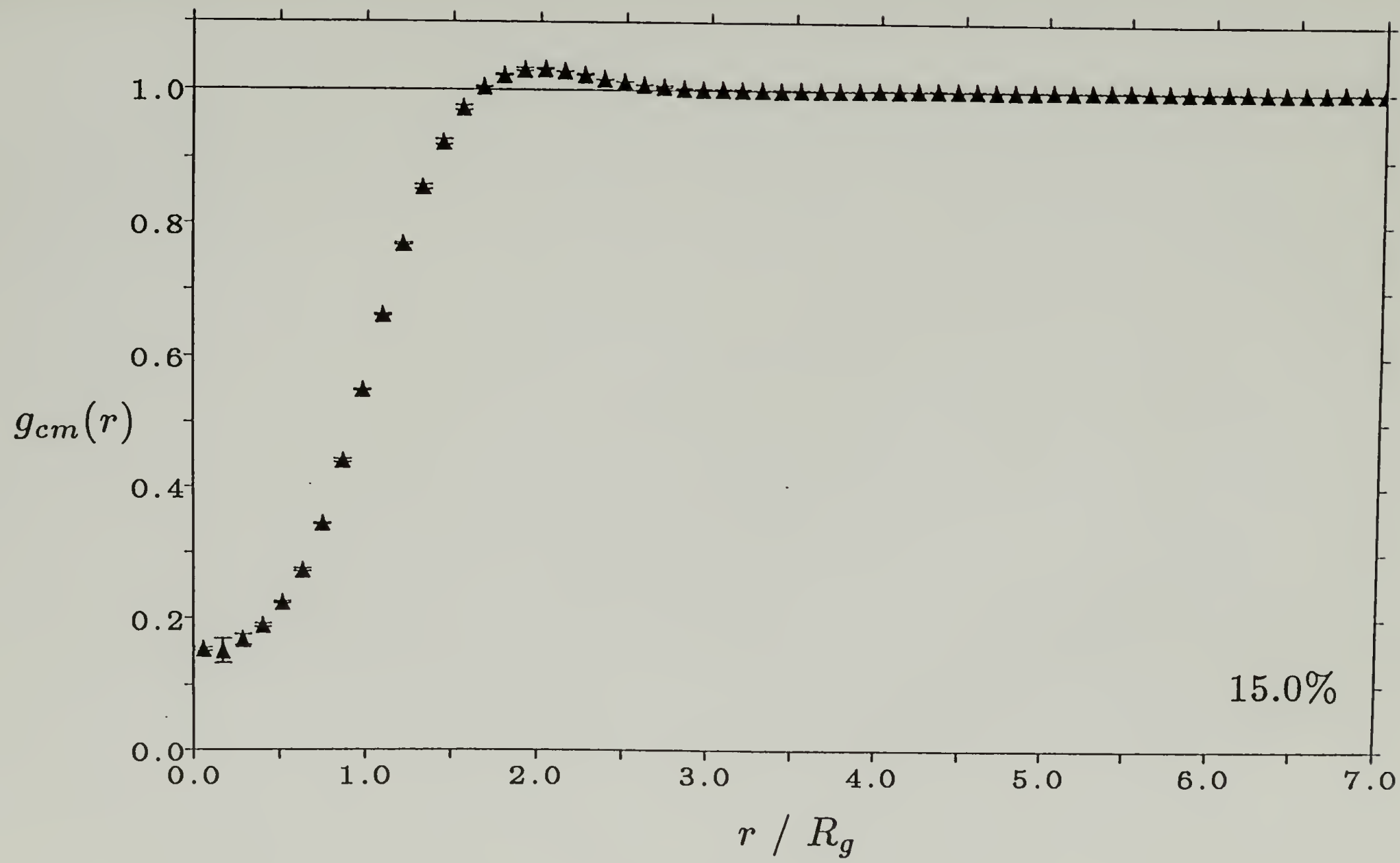

Figure 3.12
Center of masses' radial distribution function for a 15.0% bead volume solution.
Statistical error displayed at the 96.5% confidence level.

Table 3.1
Values for A_2 and the compressibility as determined from Equations 3–3 and 3–4 with the second virial coefficient A_2° determined by extrapolation.

Bead Volume (percent)	A_2 x 10^3 (volume / no. of chains)		Z (unitless)	
	(max.)	(min.)	(max.)	(min.)
1.0	1.8	(4.5) (-0.9)	1.09	(1.22) (0.96)
2.5	1.3	(2.0) (0.5)	1.15	(1.24) (1.06)
5.0	0.83	(1.18) (0.47)	1.20	(1.29) (1.12)
7.5	0.60	(0.86) (0.35)	1.22	(1.31) (1.12)
10.0	0.47	(0.67) (0.26)	1.23	(1.33) (1.13)
12.5	0.38	(0.56) (0.21)	1.23	(1.34) (1.13)
15.0	0.33	(0.45) (0.22)	1.24	(1.33) (1.16)

From graphical extrapolation of the A_2 values to $\rho_c = 0$,
$A_2^0 = 2.2 \times 10^3$ (vol. / no. of chains)^2

or relations for the chemical potential for simple fluids [McQuarrie, 1976, Chp. 13] because these relations also rely on simple interactions being the only ones contributing significantly. However, these simulations should be able to provide good estimates for dilute concentrations as the errors obtained here are relatively small.

3.5.3 Chains' Scattering Function.

These are the most novel results presented in this thesis as well as being important. To the best of our knowledge these are the first calculations in which the scattering function $S(q)$ has been calculated directly from a system of chains. While structure factors and scattering factors have been determined in computer simulations, the only direct determinations (using Equation 3–5) were for single chains (*e.g.*, Baumgärtner and Binder [1979], Baumgärtner [1980]). Curro and Schweizer determine an *intramolecular* structure factor [1987a, 1987b, 1988a–d] *via* a recently developed equation of state [Curro and Schweizer, 1987a–b; Schweizer and Curro, 1988a–d], but their calculations are based on presumptions for the melt state that cannot be guaranteed in more dilute solutions. The direct application of a Fourier transform to the two body, bead radial distribution function is inapplicable because individual radial distribution functions must be retained for every distinct pair of scatterers in the system, and then each radial distribution must be separately integrated with the necessary phase information being lost before the resultant intensities can be summed; even if these criteria were met, the error that would have propagated from this method would most likely make this expensive calculation of little worth. (Note that one cannot integrate the center of mass radial distribution function as was done for the second virial coefficient in §3.5.2 since scattering is not a colligative property and depends on the position of the scatterer.)

Since three different types of scattering functions will be presented, a comment on the nomenclature is in order. The term $S(q)$ will represent the intrachain

and interchain contributions to the scattering. This quantity is closely related to the intensity that one would measure in an X-ray or neutron scattering experiment, for example. The term $S_s(q)$ represents only the intrachain contribution to the overall scattering; the notation is derived from the "same" chain being involved in the scattering. The third term $S_o(q)$ represents only the interchain contribution to the scattering; the notation here derives from "other" chains contributing to the scattering. As one would expect, $S(q) = S_s(q) + S_o(q)$ with proper normalization for the terms.

An important observation to be made is that, as these are simulations in which the actual positions of the chains are known exactly within a Monte Carlo step, we are able to separate the contributions to the scattered intensity, just as we could distinguish the various environments for the beads (*cf.* §3.5.1), to determine quantitatively the magnitude and form of these contributions. Thus, we can investigate the structure of a single chain at any concentration, rather than just observing the average conformations of isolated chains in dilute solutions. Conversely, we can determine the larger scale organization of a fluid at any concentration without having this information obscured in dilute solution by the contributions from isolated chains.

Figure 3.13 serves the purpose of comparing the value of $S_s(q)$ for the 1.0% bead volume system, which again represents the intrachain scattering, to the standard, isolated Gaussian coil. The 1.0% system is presented as the effect on the shape of the "pearl necklace" (or "bead-link") model is affected little by the presence of other chains at this concentration. The two important trends to observe are both related to the fact that the Gaussian coil may interpenetrate itself while the "pearl necklace" model must observe excluded volume constraints. First, the magnitude of $S_s(q)$ is less than that of the structure factor for the Gaussian coil outside the core of the polymer, *i.e.*, the volume taken by $\frac{4}{3}\pi R_g^3$; as the chains occupy space, their ends cannot cross through other beads on the chain in the exploration of phase space. Second, within the core region, there is no structure to be found at high wavenumbers (short distances) for the Gaussian coil

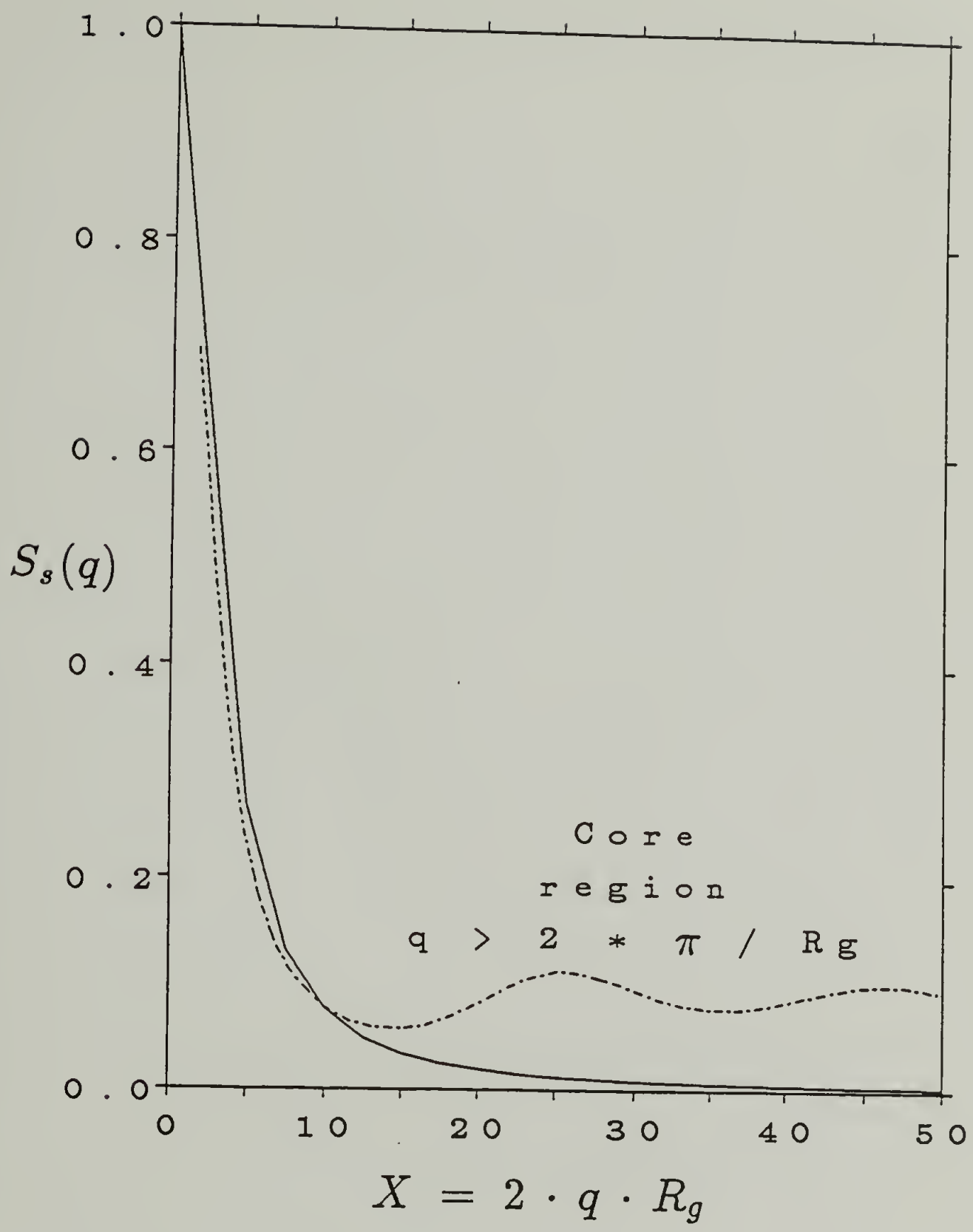

Figure 3.13

Comparison of the scattering from a single Gaussian chain to that from a chain in a 1.0% bead volume solution.

for which $q \overset{\lim}{\to} \infty S_s(q) = 0$. The simulation, however, is able to indicate what the average structure within the "core region" of a chain would be. This ability of the simulation should allow one to compare models against neutron scattering data. (Notably, the scattering function here is largely influenced by the "bead-link" model used, but then the model can be changed.)

Figures 3.14, 3.15, and 3.16 display the various structure factors that were obtained for the 1.0% bead volume system. First, note that the statistical errors on the values obtained are quite small,[5] and are negligible for the full scattering function $S(q)$. As the errors are even less for higher concentrations, for clarity error bars will not be included on any other scattering plots. Second, note the ability to separate the contributions to the full scattering $S(q)$ in Figure 3.14 into its composite intrachain $S_s(q)$ (Fig. 3.15) and interchain $S_o(q)$ (Fig. 3.16) parts. This separation allows one to confirm the expected trend that, for this dilute system, the scattering contributed by other chains within the core region of the chain (*i.e.*, $X = 2qR_g$) is effectively nil. One should *not* be concerned that the values for $S_o(q)$ in Figure 3.16 are negative. Only the full scattering function $S(q)$ is an inherently positive quantity; the useful division into intrachain and interchain contributions is mathematically arbitrary.

One other point should be made concerning the presentation of the data. As these simulations were performed in a periodic box, only certain wavenumbers can be investigated. (This point is developed fully in Appendix E.) Figures 3.14 to 3.16 show the distinct points that were obtained; the remainder of the figures concerning scattering connect these points to better display the trends. This requirement of investigating only certain wavenumber is not very limiting, however, as can be seen from the number of points displayed. In fact, for $q > 55$ some wavenumbers were not included as the graphs became too crowded.

---

[5] Recall that all estimates of error throughout this document are reported at the 96.5% confidence level (2σ).

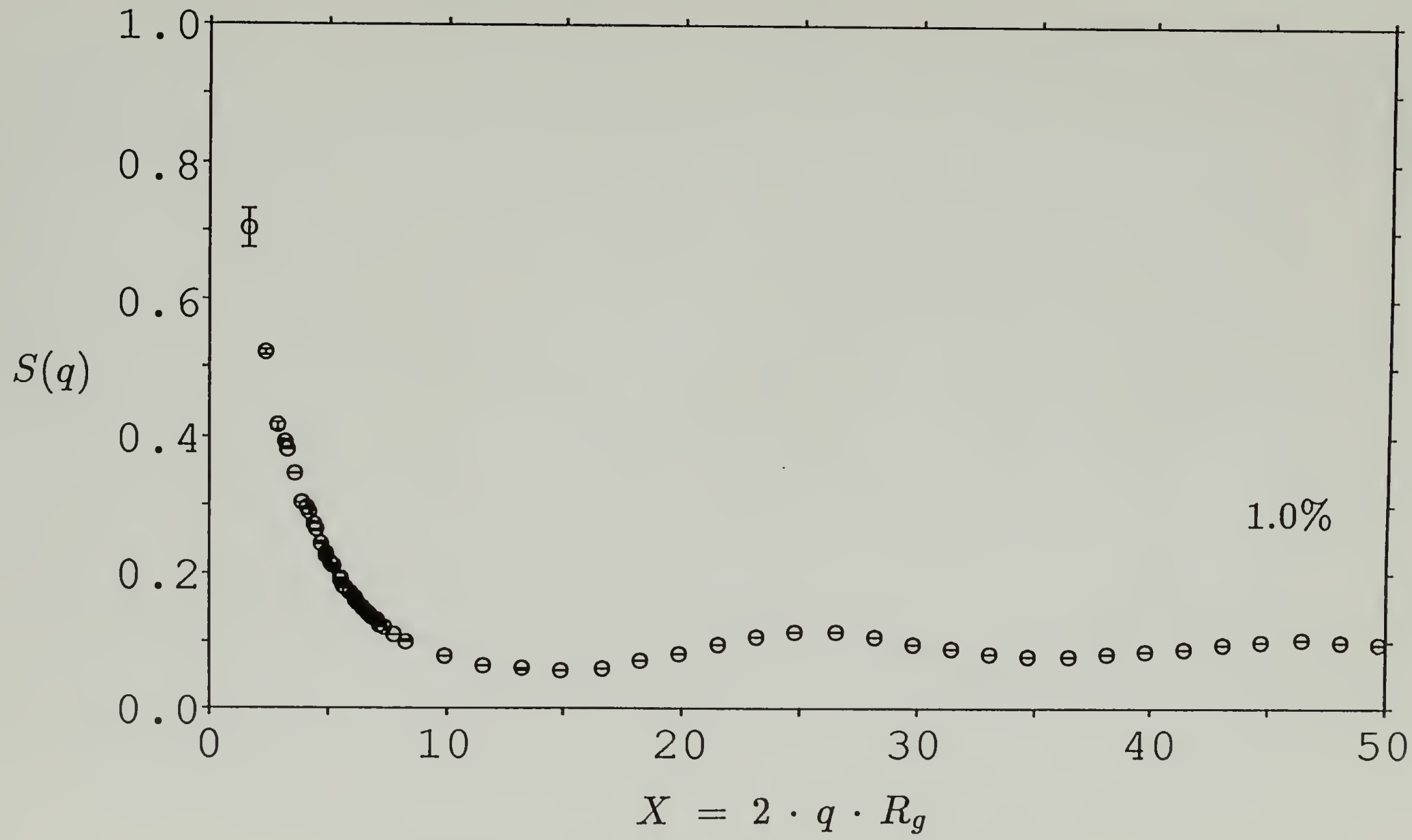

Figure 3.14
Scattering from all chains for a 1.0% bead volume solution.

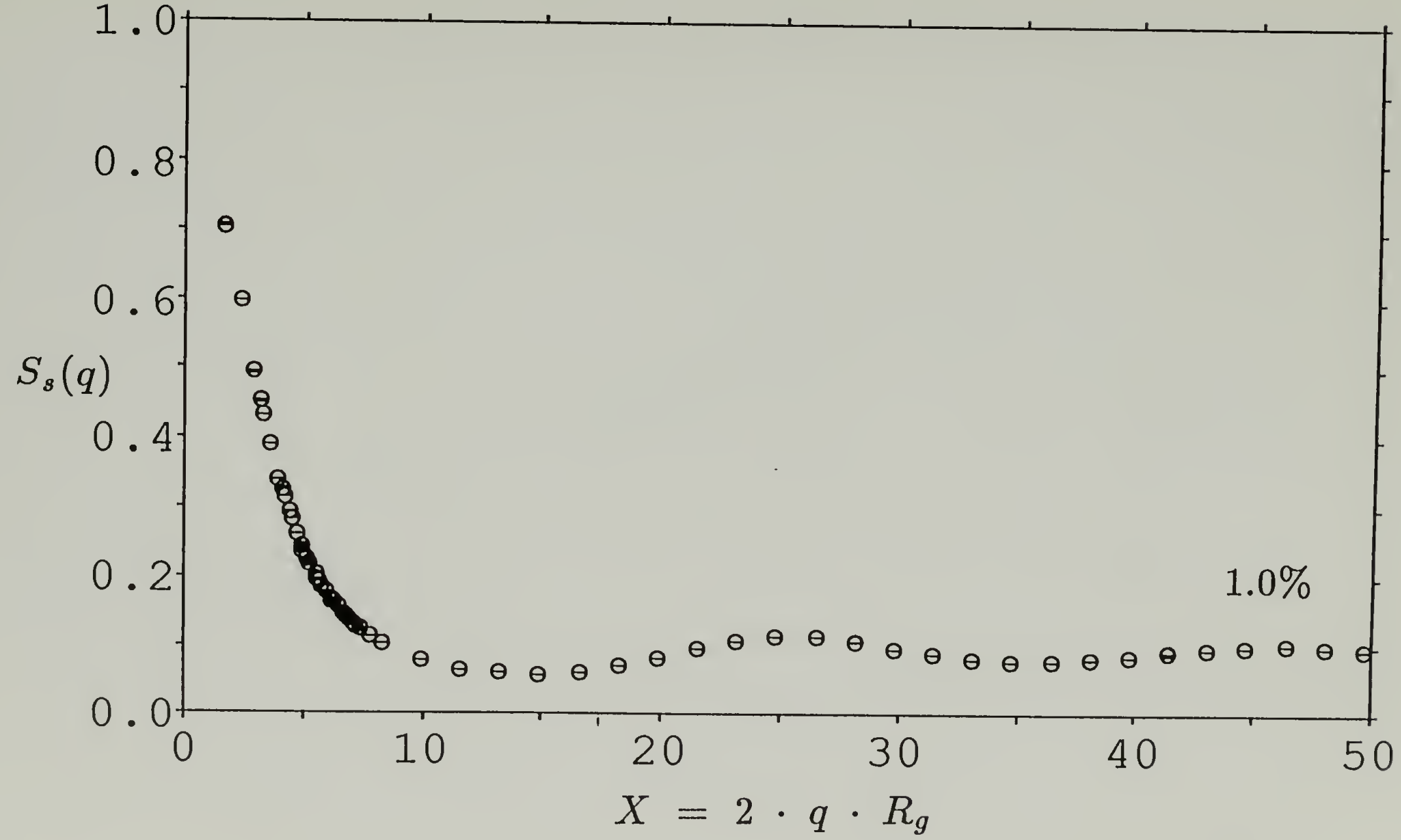

Figure 3.15
Intrachain scattering for a 1.0% bead volume solution.

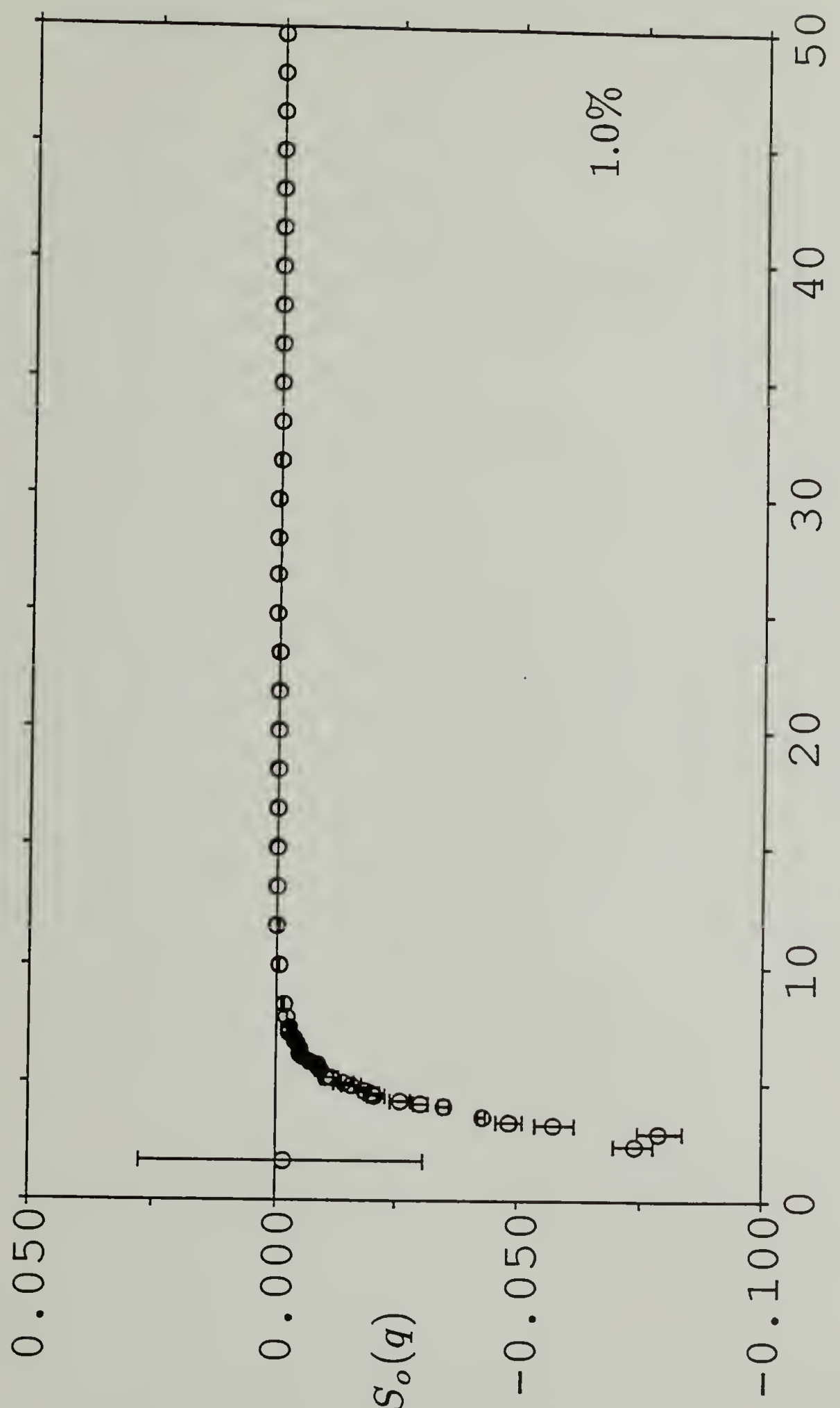

Figure 3.16
Interchain scattering for a 1.0% bead volume solution.

Figures 3.17 and 3.18 display the data for the full scattering function $S(q)$. The data have been split into two parts for conceptual clarity with Figure 3.17 representing the scattering outside the core region of a chain, and Figure 3.18 representing the scattering inside the core region. (This organization is carried through the rest of the scattering graphs.)

Figure 3.17 displays the scattering function for wavenumbers q that would be experimentally attainable by X-ray scattering. Note that as the concentration increases, the peaks shown become greater in amplitude and move to lower wavenumbers. The former means that the solutions show organized packing even at dilute concentrations, as identified by the second virial coefficient (*cf.* §3.5.2), of about 5.0% bead volume. The latter shift in the peaks is strictly because of the the root-mean-squared radius of gyration decreasing with concentration. This is demonstrated by Figure 3.19 in which the scattering function $S(q)$ is plotted as a function of the wavenumber q only. Here, all of the maximum intensities fall at the same wavenumber.

Our 1.0% results agree well in form with those presented by Vacatello, *et al.* [1980] for their model calculation of X-ray scattering for an isolated molecule. More importantly, using the 15.0% bead volume sample, the shape of the curve begins to emulate the scattering curves of real substances, two good examples (due to their simple molecular structure and amorphous nature) being SiO_2 [Warren, 1937] and carbon black [Klug and Alexander, 1974, Chp. 11].

Figure 3.18 displays the scattering function for wavenumbers q that can be investigated *via* neutron scattering. The trend towards higher amplitudes of the scattering function as the concentration is increased indicates the effect of interpenetration on the sample. This is confirmed by comparing this graph to Figure 3.22 in which the intensity does not change for the intrachain scattering as a function of concentration.

Figure 3.20 is the same information as in Figure 3.18, but with the influence of the root-mean-squared radius of gyration removed. In this case one observes that

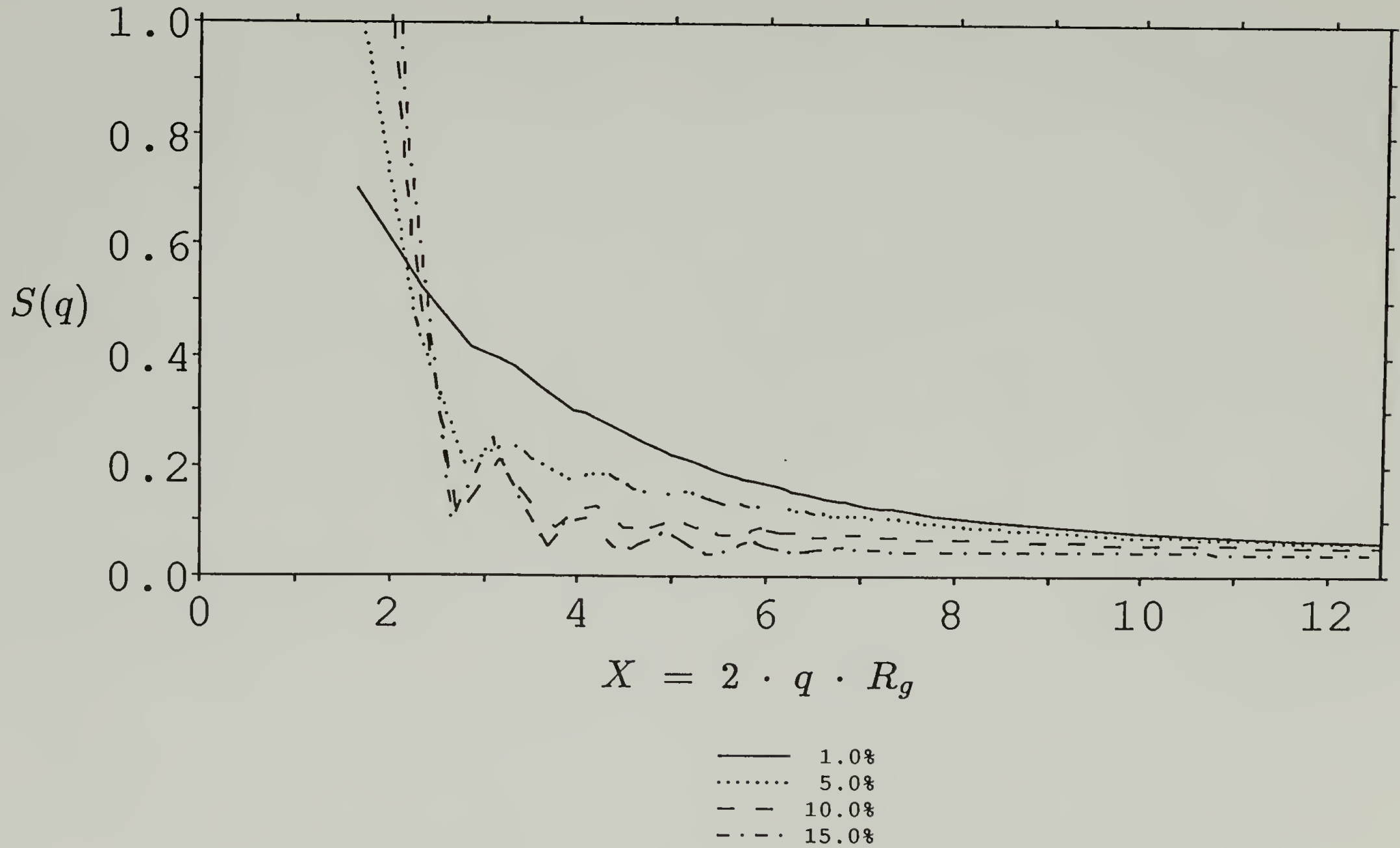

Figure 3.17
Scattering from all chains for various bead volume fractions; outer region.

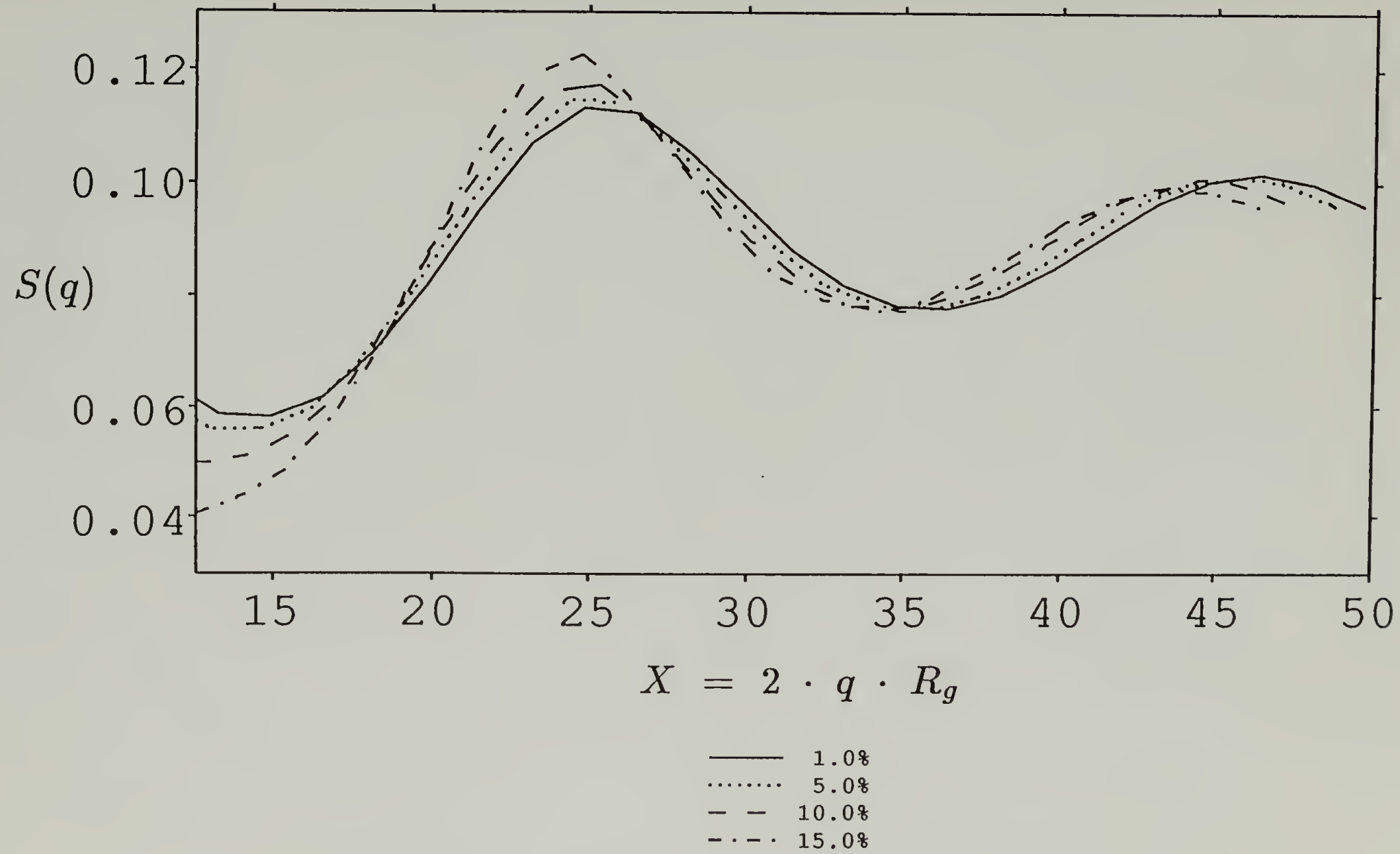

Figure 3.18
Scattering from all chains for various bead volume fractions; core region.

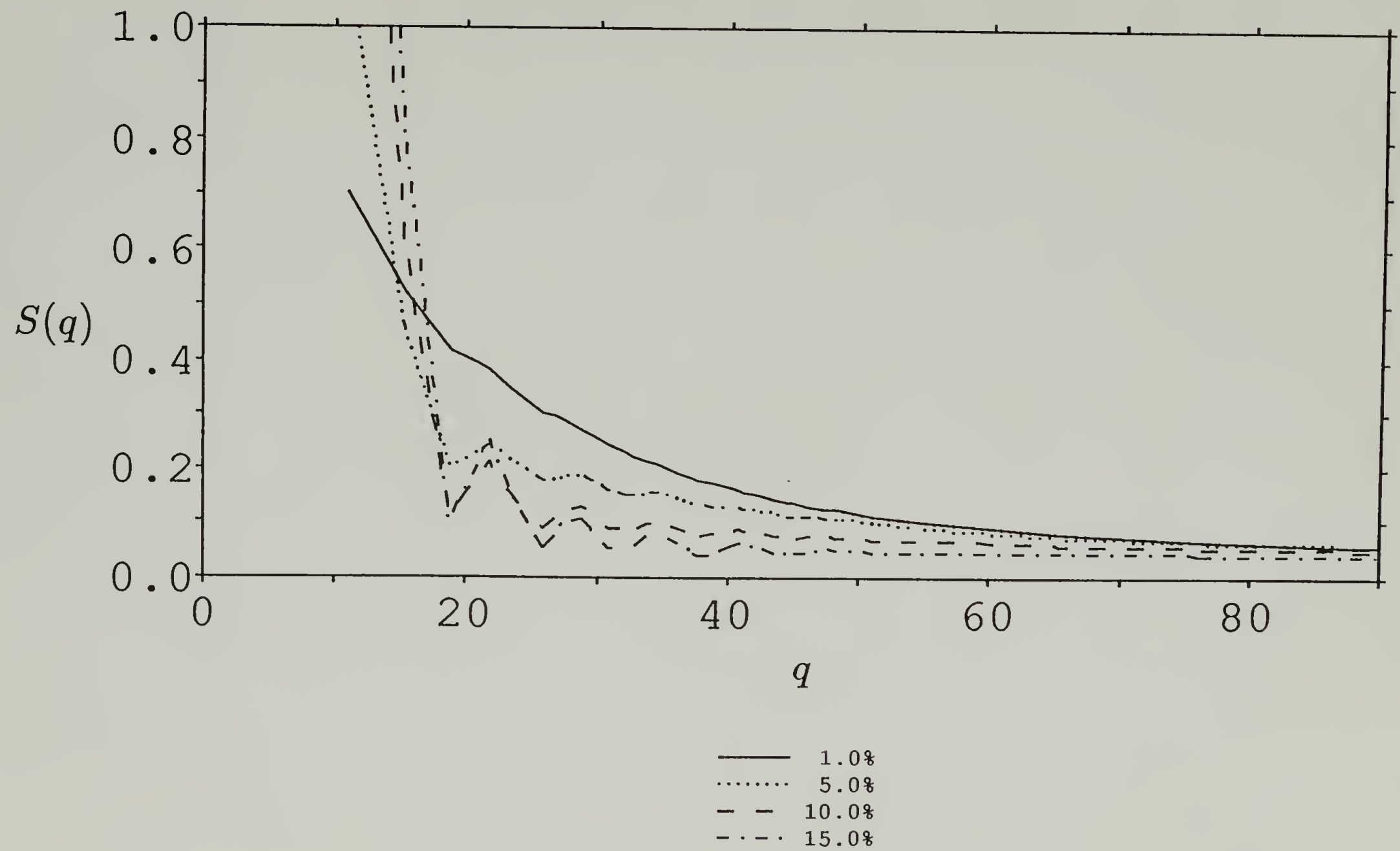

Figure 3.19
Scattering from all chains for various bead volume fractions; outer region.

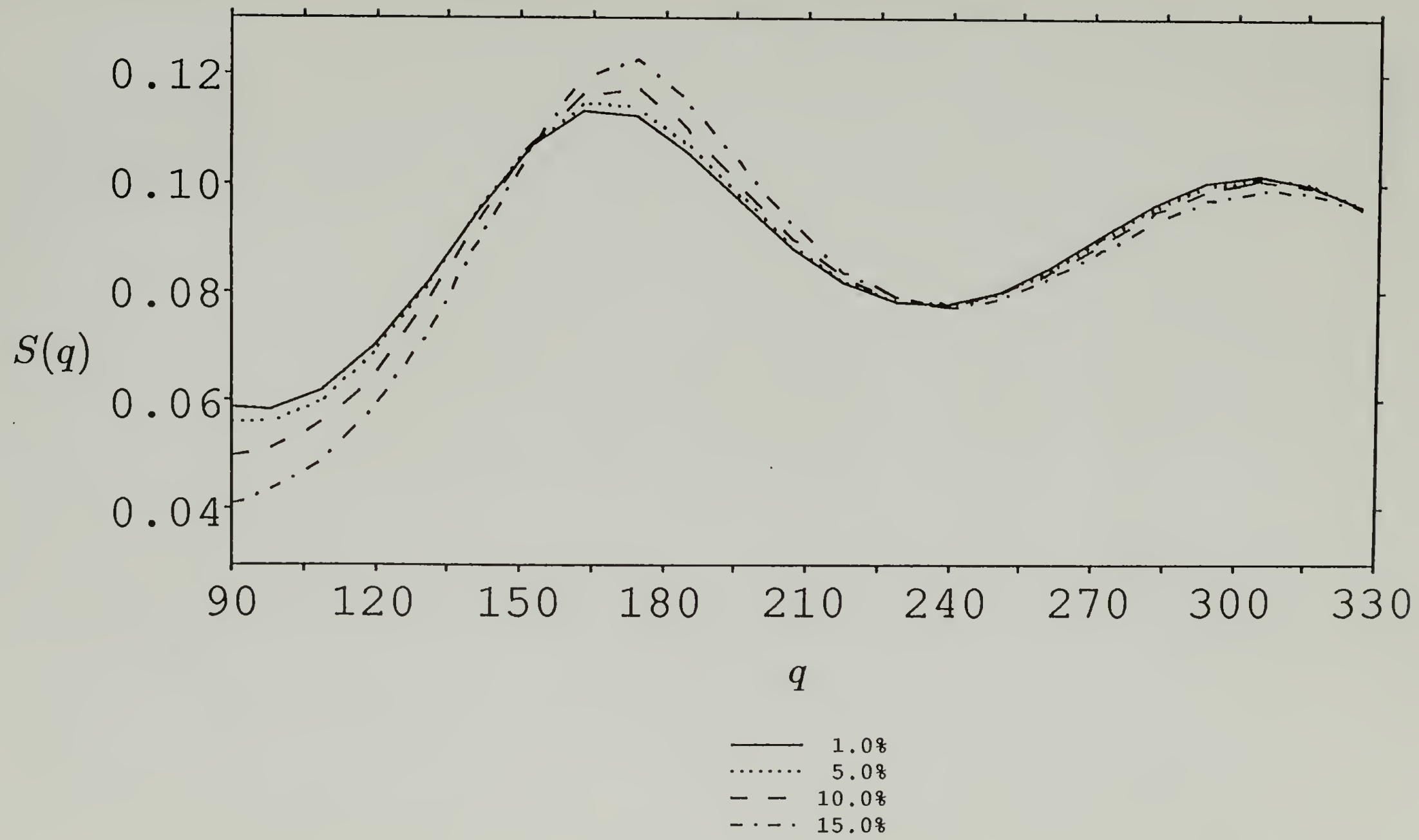

Figure 3.20
Scattering from all chains for various bead volume fractions; core region.

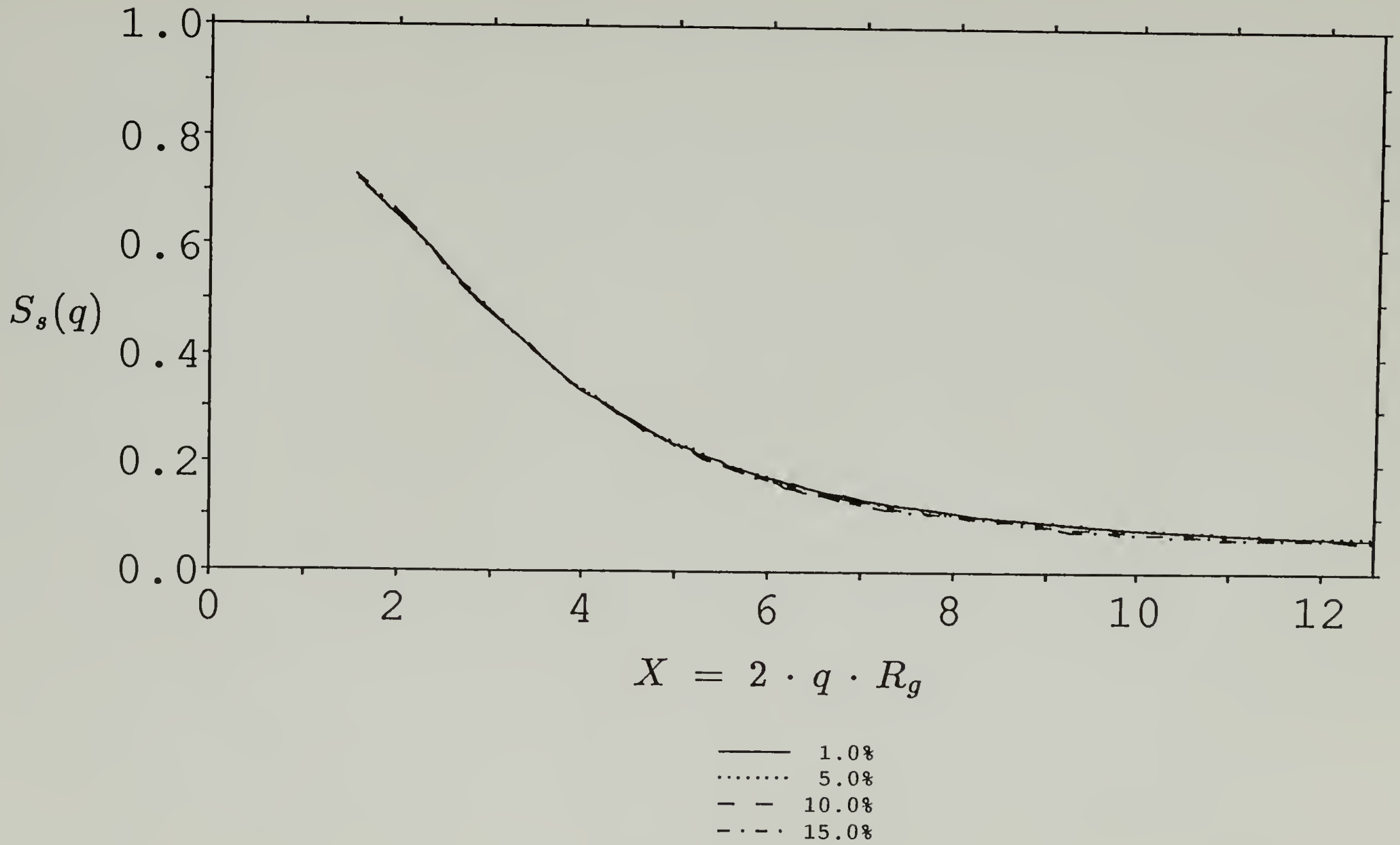

Figure 3.21
Intrachain scattering for various bead volume fractions; outer region.

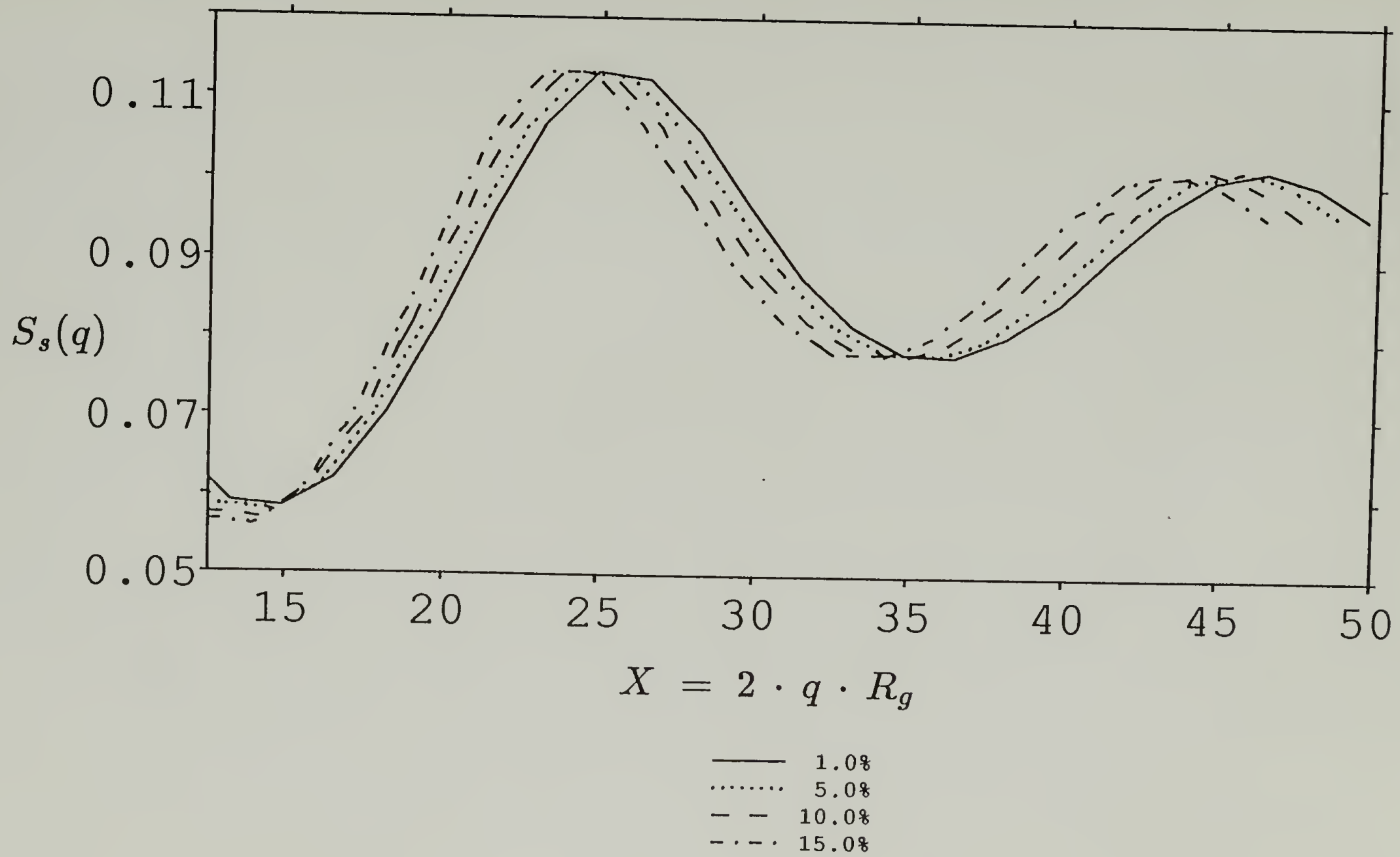

Figure 3.22
Intrachain scattering for various bead volume fractions; core region.

the regions from which scattering occurs are actually becoming smaller (moving to higher wavenumbers), which is reasonable as the chains intertwine.

The increase in amplitude of the scattering function as concentration increases is also shown in the data calculated by Schweizer and Curro [1988d] from their integral equation of state at much higher concentrations, 30% to 50% bead volume. A more important comparison is made with the data of Daoud, *et al.* [1975]. This data is for semidilute concentrations of deuterated polystyrene in carbon disulfide. Comparing their curve of a 1.1×10^6 weight averaged molecular weight deuterated polystyrene at a concentration of 2.5×10^{-2} g/cm^3 (Figure 11 in their paper) to our data for a 1.0% bead volume percent sample, the agreement between the trends is remarkable with even the deviation from linearity occurring at about the same value of X. (See Fig. 3.25.) A key point to realize, however, is that data of Daoud, *et al.*do not describe the core region that Figure 3.18 represents. There is at present no *solution* data in the literature for this region of scattering.

Figures 3.23 and 3.24 complete the series by showing the contribution of interchain scattering $S_o(q)$ to the full scattering function $S(q)$. Of interest is the nonlinear increase in the amplitude of $S_o(q)$ in the core region of the chain (Fig. 3.24) as a function of concentration.

3.5.4 Compressibilities and Scaling Relations.

Finally, we arrive at the point to which the original intent of this study was aimed: to determine thermodynamic information from computer simulations and to confirm the ability of these simulations to provide the appropriate semidilute scaling forms. Figure 3.26 is an important graph that displays the compressibility *per* bead as determined by the canonical ensemble simulations (open circles), by the isobaric-isothermal ensemble (filled circles), by the second virial coefficient (Eqn. 3–4), by Honnell and Hall's formulation [1989], by Wertheim's second-order

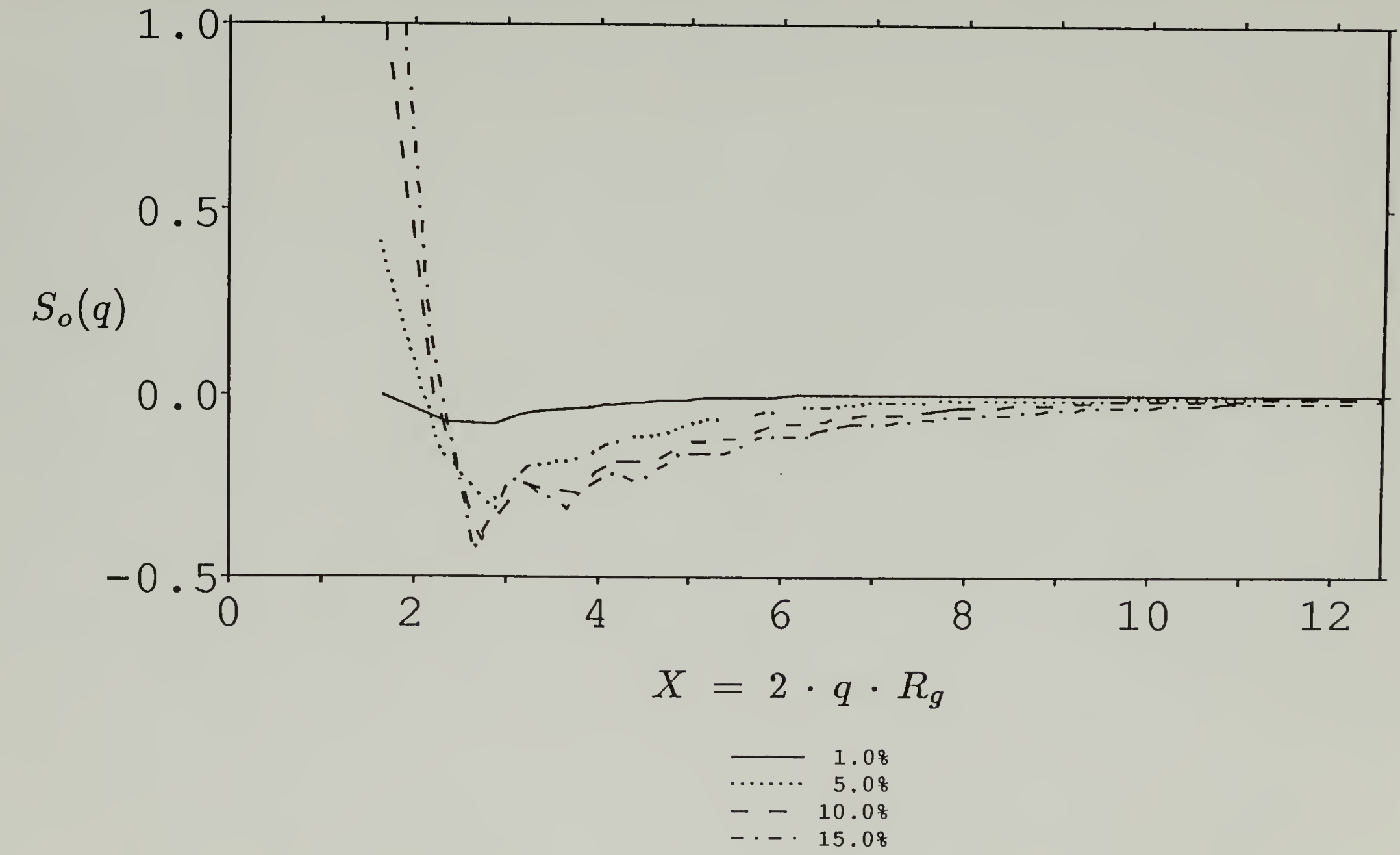

Figure 3.23
Interchain scattering for various bead volume fractions; outer region.

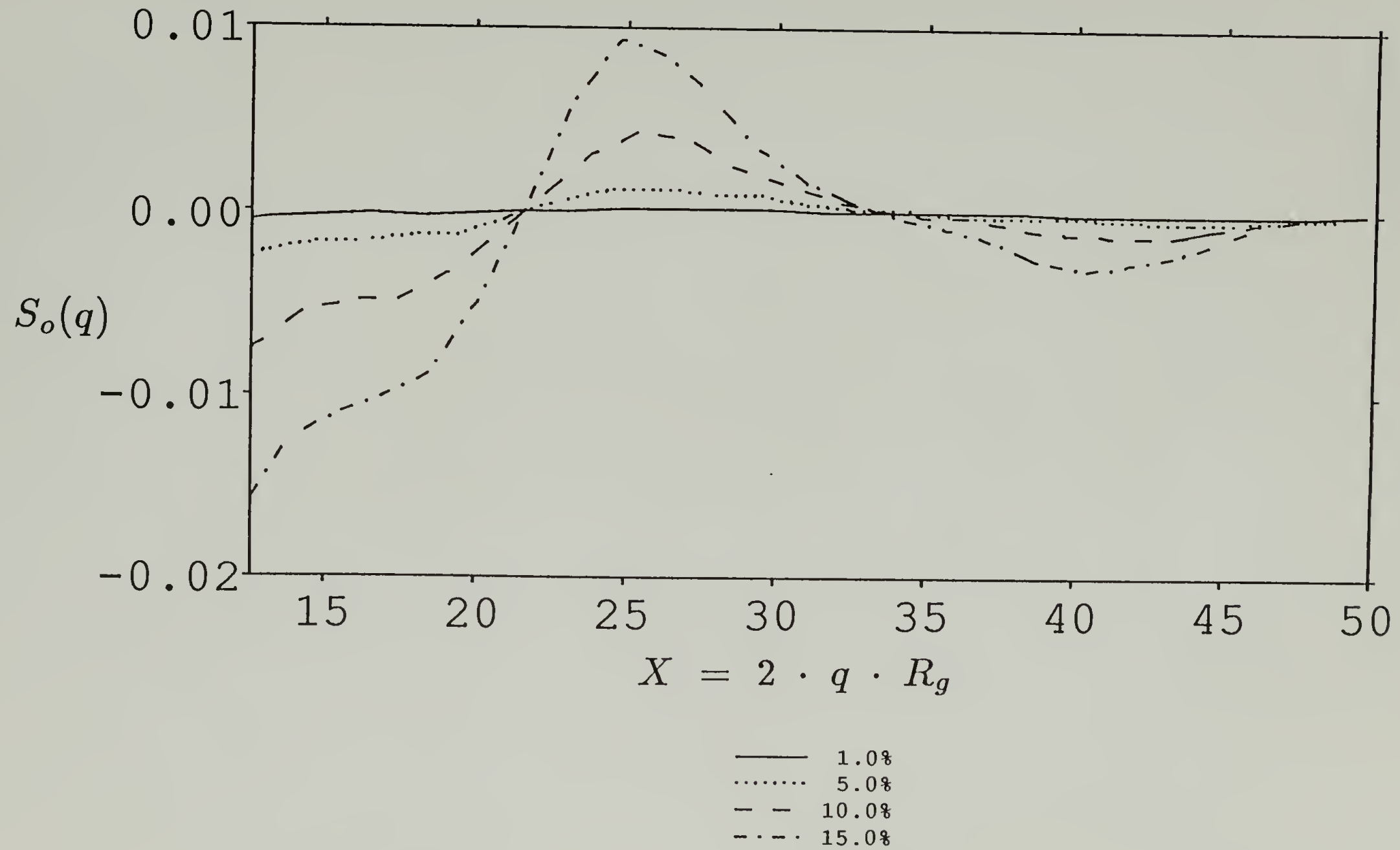

Figure 3.24
Interchain scattering for various bead volume fractions; core region.

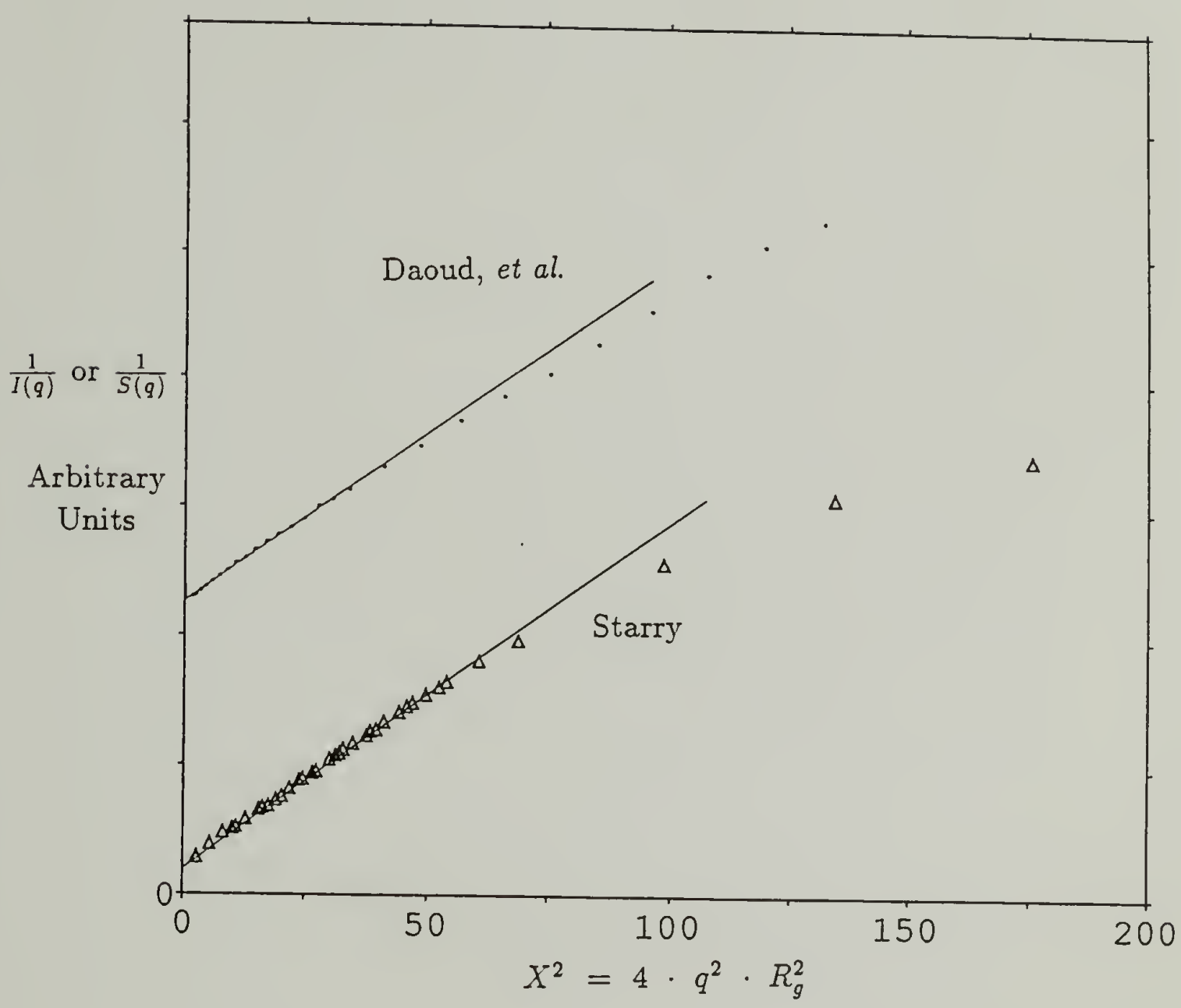

Figure 3.25

Comparison of Daoud's neutron scattering data to the results for a 1.0% bead volume solution.

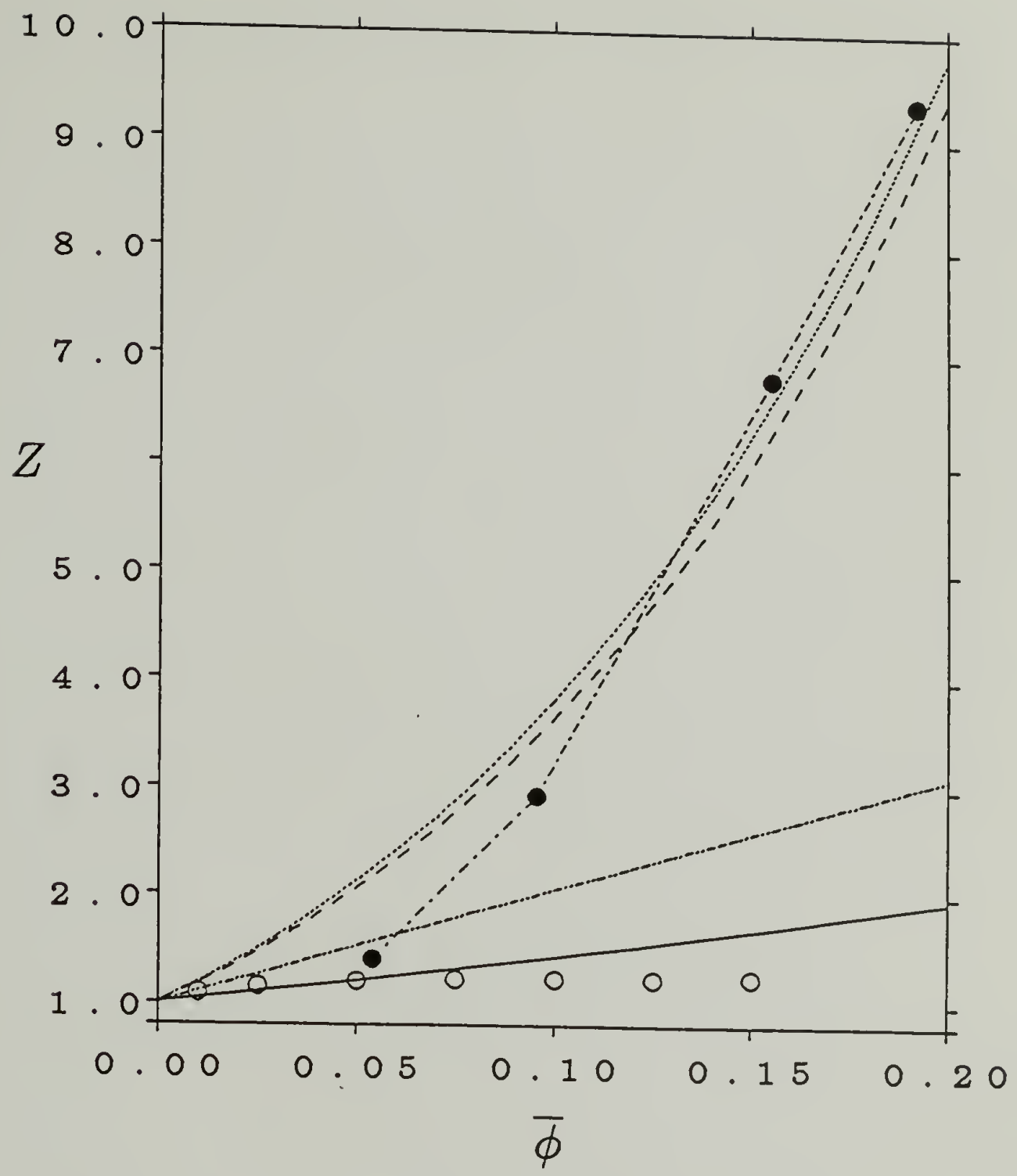

Figure 3.26

Compressibility as a function of bead volume fraction for various models and these simulations.

thermodynamic perturbation theory [1987], and by the Flory-Huggins mean field approach.

The predictions of the canonical ensemble become unreasonable above 5% bead volume. This is due to the inadequacy of calculating only two body interactions as was discussed in §3.5.2. The values shown from the isobaric-isothermal ensemble no longer follow the mean field trend, which is an encouraging result. This means that the fault found with the lattice simulations does seem to be alleviated by moving to continuous space, thereby justifying this approach. Note also how the compressibility determined using the second virial coefficient (A_2°, Eqn. 3–4) from the canonical ensemble simulations merges with the results from the isobaric-isothermal ensemble calculations at approximately 5% bead volume. This consistency between the two methods provides support that our approach is correct.

To compare our values with the literature, we have plotted the predictions of Honnell and Hall's approximation to the equation of state [1989] and Wertheim's second-order thermodynamic perturbation theory (TPT2) [1987]. At higher concentrations, above about 12% bead volume, both theories agree reasonably well with our results with their results being slightly less. However, at concentrations from 0 to about 10% bead volume, these two theories predict values of the compressibility up to two times greater than are obtained from our simulations. As our isobaric-isothermal results tend toward the curve determined from the approximation derived from the virial expansion (to second order in concentration), we believe that our simulations are displaying the correct trend.

Another comparison can be made with the recent theory of Schweizer and Curro [1988b] in which they are primarily concerned with the melt state, but have calculated a compressibility for a 16 segment chain ($n = 17$) at 15.% bead volume from their integral equation of state. Our value for the compressibility *per* bead is 3.8 times greater than theirs. Unfortunately, there is presently no method by which to determine if either value is correct. Equilibration problems exist for both

methods so that either or both methods could be well away from the appropriate value. Importantly, as well, a number of assumptions that are necessary for the integral equation of state method to work well do not apply at these concentrations, and pressures are known to be "significantly underestimated" by their method [Schweizer and Curro, 1988a].

For the moment then, our values appear to be consistent and of the correct magnitude. More simulations need to be done to determine the precision of the values obtained, and solution experiments need to be performed (extending the work of Cotton, *et al.* [1973] and Candau, Strazielle, and Benoit [1976]) by which the accuracy of the simulations may be evaluated.

Figure 3.27 shows the reduced osmotic pressure as a function of bead volume. The values are tending towards the scaling relation predicted by des Cloizeaux's law [des Cloizeaux, 1975] at a lower concentration than was observed in the lattice simulations, but the trend is not yet convincing. Again, more simulations need to be done to show that these pressures are actually the appropriate values to be obtained.

Figure 3.28 shows the trend for the mean squared end-to-end radii as a function of concentration to test Daoud's relation [Daoud, *et al.*, 1975]. The data are plotted in a format suggested by Curro [1976]. Curro claims that the break shown in the free space curves indicates that one has entered the semidilute region. This is not upheld by agreement with Daoud's relation, but the concentration at which the break occurs is consistent with the concentration at which the scattering function presented in §3.5.3 agrees with data of Daoud, *et al.* This may indicate that the scaling form determined by Daoud is incorrect, or that something fundamental is still missing from these continuous space simulations.

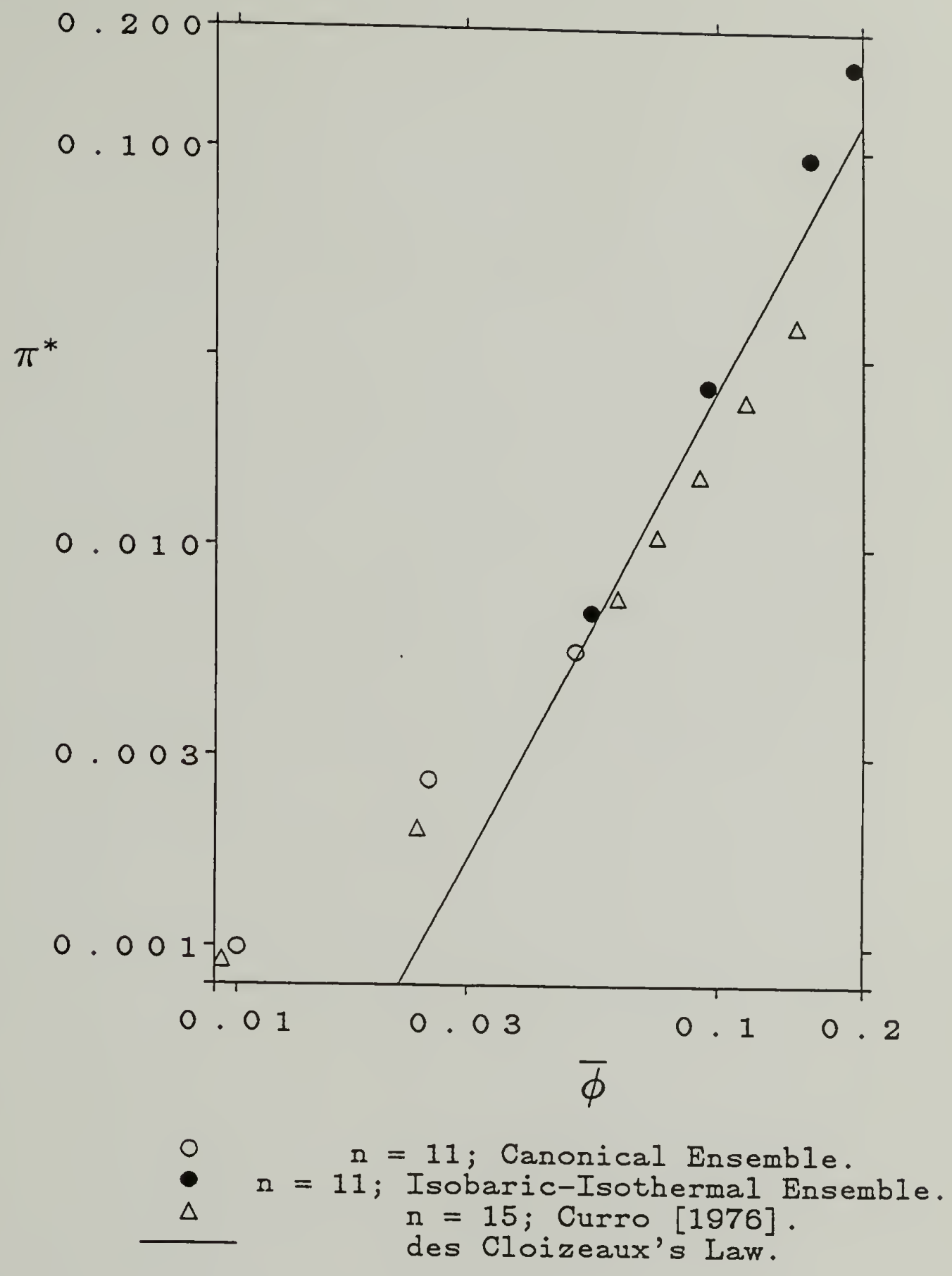

Figure 3.27

Modified osmotic pressure as a function of bead volume fraction for these simulations and Curro's work [1976] as compared to des Cloizeaux's law.

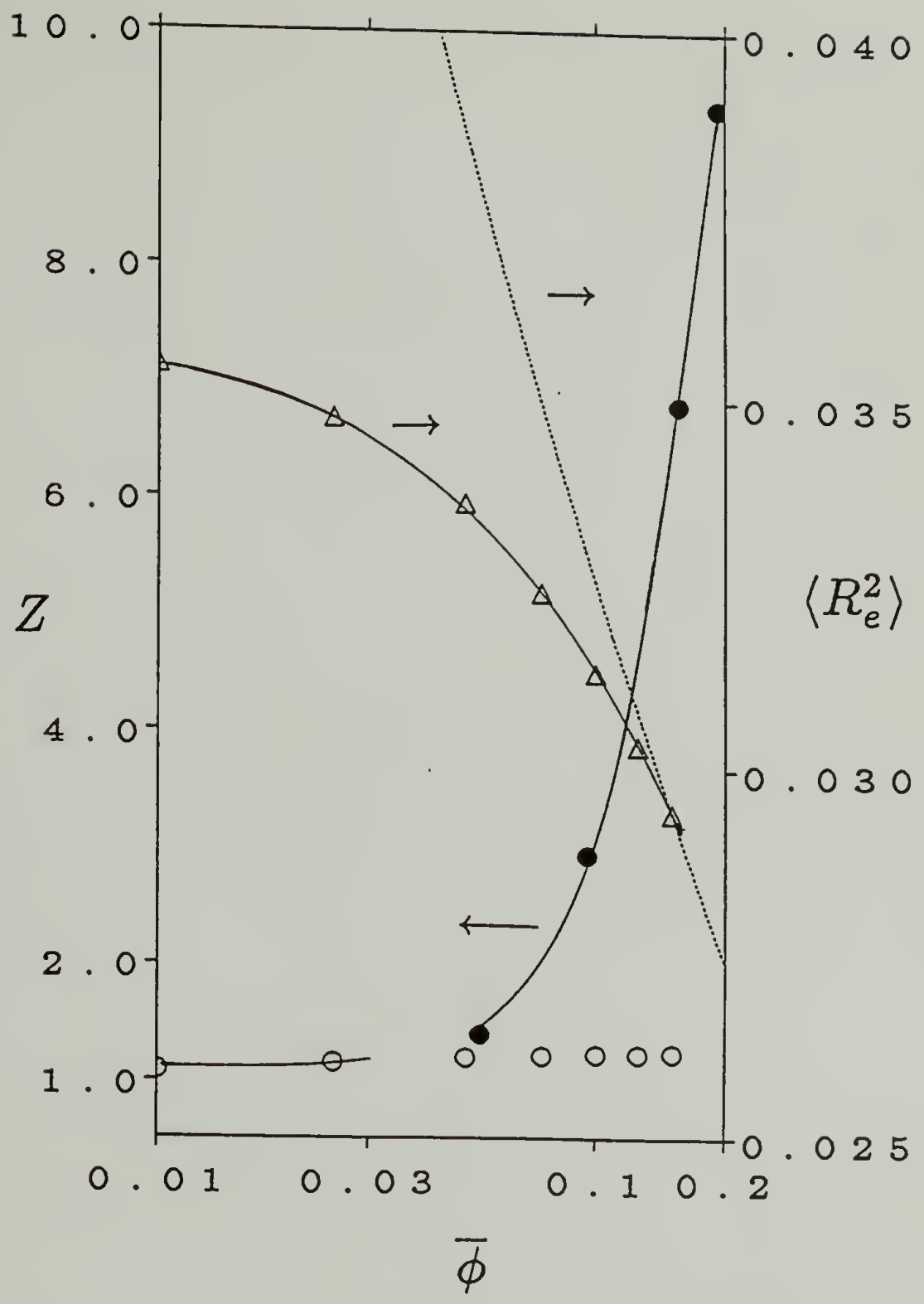

Figure 3.28

Comparison of the compressibility and the mean squared end-to-end distance for determining the onset of the overlap concentration as suggested by Curro [1976].

CHAPTER 4

CONCLUSIONS AND FUTURE WORK

4.1 Conclusions.

First, lattice simulations appear to be unable to predict thermodynamic values and scaling relationships in dilute and semidilute solutions. This important observation has gone unreported, most likely because people's attentions were oriented either towards testing lattice theories or towards testing bulk state properties. The former is an useful tool to investigate the validity of the theoretical assumptions, and the latter is an extremely important condition for polymeric processing.

Second, the novel capability of calculating the scattering function directly from a large system of chains using a Monte Carlo simulation now exists. This ability is a significant step towards performing calculations on a model that one believes applies to an amorphous or weakly organized, polymeric sample and permits comparison between model structures and experimentally observed data. The agreement that was obtained with both X-ray and neutron scattering results is extremely encouraging.

Third, a powerful apparatus to investigate the thermodynamic properties of a system using the continuous space, isobaric-isothermal ensemble has been developed. While this method of simulation is in its nascent form, the extensions available to investigate both inherent properties of a system, *e.g.*, the heat capacity, and organizational characteristics, *e.g.*, phase changes, make this an intriguing method to utilize.

4.2 Future Work: Development of the Methods Used.

The development of the methods used in these simulations need to move in two directions: fundamental studies, and mechanistic studies. The former is a recognition that some of the presumptions made in simulations are neither well-founded nor well-understood. The latter deals with the practical concerns of obtaining reliable results from computer simulations that can be compared with theoretical predictions and, especially, experimental data.

4.2.1 Foundations.

One problem that has existed for a long time and been avoided by most researchers is the aspect of statistical bias that can be introduced into one's simulations by the method through which the objects are moved. Metropolis, *et al.* [1953] founded the basis for sampling configurations of simple fluids, in particular, "hard disk" and "hard sphere" fluids, based on a comparison of the change in the system's energy to a randomly chosen change in energy and simultaneously demonstrated that their mechanism of generating new configurations was statistically unbiased. For polymeric systems, the "Metropolis test," which only treats accepting a new configuration based on the system's change in energy, was lifted and applied to chains moving by reptation [Kron, 1965; Kron and Ptitsyn, 1967], without proof that reptation generated system that were unbiased statistically. Wall and Mandel [1975] showed that reptation, while not completely unbiased, was very close to the ideal since the parts of phase space that could not be explored should be of negligible importance; results from this group [Dadmun, 1988] confirm this assumption.

The difficulty is that reptation is only a viable method for moving chains that are contorted. For chains that are stiff, reptation is an inefficient method by which to sample phase space. Rigid body translations and rotations must be included, and one needs to resort to mechanisms such as alternating kink-jumps to sample

the available phase space efficiently. Also, for parallel kink-jump, or any other method in which all of the chains undergo movement simultaneously, determining how well phase space is being sampled has yet to be investigated at all. The proof that kink-jump, parallel kink-jump, and other methods of movement are either unbiased, or predictably biased so that one may compensate for the error, is necessary before firm belief can be placed in the results that are obtained by such mechanisms.

The second specter that has arisen in these simulations is the complication of finite size effects in the simulations. This is also not a new problem, but one that has been avoided by most researchers. Certainly one may perform checks, as was done in this work, but, just as here, spurious results may be obtained that are poorly understood.

An exhaustive study of finite size effects would be difficult to implement. A number of characteristic size ratios would have to be tested: the volume of a chain to the volume of the system, the volume occupied by all of the chains to the volume of the system (*i.e.*, the concentration), the length of a chain to the size of the volume (*i.e.*, "molecular weight" effects), and internal sizes of the model (*e.g.*, bead diameter to link length for a "pearl necklace" model) [Ho and Baumgärtner, 1989]. However, without such a study for at least a commonly used model, the accuracy of the simulation results may be questioned.

4.2.2 Mechanisms.

The mechanisms for simulation also require further development. That the lattice simulations fail to provide the appropriate thermodynamic relationships, and that continuous space simulations do better, may indicate that the severe restrictions on the degrees of freedom available in most lattice simulations are the difficulty. If this hypothesis is true, then running simulations on an complex lattice where many sites are available for a move, rather than the 3 for tetrahedral lattices or the 5 for simple cubic systems, may allow one to retain the speed of

computation of a lattice while providing similar results to a free space simulation. Unfortunately, the results for a dodecahedral lattice developed by Jeffrey Skolnick (permitting 11 possible moves) are only for melts [Vas, 1989] and have yet to appear in the literature. Also, our expectation is that an order of magnitude increase in the number of possible sites *per* move is required, not just the factor of 2 or 3 increase offered by Skolnick's dodecahedral lattice.

The continuous space simulations require development as well. The isobaric-isothermal ensemble has difficulties with converging to a unique solution, and therefore requires an inordinate amount of operator intervention and luck. Boyd has seen this in his work [1989] as well, and these difficulties with convergence appear to have caused him to discontinue his experiments before a large enough number of samples were taken to be statistically significant. Effort needs to be placed into the isobaric-isothermal ensemble simulations to discern the patterns of volume fluctuations so that stability may be reached more quickly and these simulations may become more tractable.

4.3 Future Work: Extensions of These Simulations.

First, as these codes were originally designed to be used to simulate low temperature solutions in which the semiflexible polymeric chains become rigid, the codes should be used for these types of systems. This would allow us to compare not only with theories for stiff, semiflexible chains [Gujrati and Goldstein, 1981], but to compare with experiments on liquid crystalline systems.

Second, as the systems of stiff chains order, one may look at phase transitions similar to the isotropic-nematic phase transitions found in many liquid crystalline systems. To the present, such simulations have been done on lattices for simplicity [Baumgärtner, 1986; 1987]; however, Baumgärtner [1985] has shown that moving rigid objects on lattice causes severe difficulties and that mean field presumptions are extremely poor. Utilizing the continuous space simulations not only frees one from the lattice, but also permits more realistic packing arrangements, *e.g.*,

nematic phases in which the individual chains deviate slightly in orientation from the molecular director.

Third, molecular weight effects need to be studied. Schweitzer and Curro [1988a–d] have spent a great deal of effort investigating how the predictions of their integral equation of state for the melt state are affected by this variable. Noting that thermodynamic quantities, such as the osmotic pressure, are greatly affected by the molecular weight, its influence on the results in dilute and semidilute solutions needs to be investigated.

Fourth, as the 1.0% scattering function calculated here agrees extremely well with the neutron scattering data from Daoud, *et al.*, more simulations should be performed for extremely dilute systems, on the order 0.1 to 1% bead volume. This would provide enough different concentrations to calibrate the results and to determine if the agreement between the simulations was just fortuitous or has real predictive value. The latter is, of course, what is desired.

Fifth, these codes were developed to be quite general. This means that the core of these codes could be utilized within molecular dynamics simulations to move one into the study of the effects of interchain potentials and fields upon thermodynamic relationships while still having effectively generic chains with an easily varied stiffness.

Last, more comparisons between simulation results and experimental data need to be made. The primary information available for semidilute solutions was obtained in the mid-1970's (*e.g.*, Candau, Strazielle, and Benoit [1976], Cotton, *et al.* [1973], Daoud, *et al.* [1975]) using osmotic pressure and neutron scattering experiments. Polystyrene was the only polymer investigated and only three solvents, benzene, toluene and carbon disulfide, were used to study the linear polymers. More data from other types of systems need to be amassed before the applicability of computer simulations and of theoretical predictions may be properly assessed.

APPENDIX A

LATTICE BASED, CANONICAL ENSEMBLE CODE

Representative Code
for the Lattice Based, Canonical Ensemble Simulations
As Executed on the IBM 3090VF Processors
at the Cornell National Supercomputer Facility, Ithaca, New York

---

```
C*************************************************************************
C                          Modified as of 5/03/88
C*************************************************************************
      IMPLICIT NONE
C
C     ***    All the values that need to be changed for different    ***
C     ***       simulations are in the next three  PARAMETER         ***
C     ***                          statements.                       ***
C     ***    (The initial seed value, ISEED, for  DRAN  is in the    ***
C     ***                    last  DATA  statement.)                 ***
C
C  Scalar constants.
C
      DOUBLE PRECISION  TAU
      PARAMETER     (TAU=11.0D+00)
      INTEGER       CHAINS, DM, DM2, DM3, DMD2, DMD4, EQUILS,
     &                LMBPTS, MONOMR, MONXCH, NSAMP, TIMLIM
      PARAMETER     (CHAINS=25, DM=17, DM2=DM**2, DM3=DM**3,
     &                DMD2=DM/2, DMD4=DM/4, EQUILS=1000,
     &                LMBPTS=16, MONOMR=11, MONXCH=MONOMR*CHAINS,
     &                NSAMP=20000, TIMLIM=10*CHAINS)
      INTEGER       DMXNSP, LMBPT1, MCLIM
      PARAMETER     (DMXNSP=DM*NSAMP, LMBPT1=LMBPTS+1,
     &                MCLIM=TIMLIM/CHAINS)
C
C     ***       Values below this point should NOT be modified,    ***
C     ***                 except for the DATA value of ISEED.      ***
C
C
C  Scalar variables.
C
      DOUBLE PRECISION  ALPHA, BOLTZ, DELHT, DUMSM1, DUMSM2,
     &                  HAMINI, HAMLTN, HAMNEW, LAMBDA,
     &                  PHIB, PRESS, RGY2, RGYAVG,
     &                  RHO, RNNAVG, RNN2,
     &                  SRFNEW, STIFF, SUCCES, XCM, YCM, ZCM,
     &                  Z, ZTHEOR
      INTEGER       ACCEPT, ARRSFT, ATMPTS, BNDINI, BNDNEW,
     &                BOXCNT, CONFGS, DIRECT, DIRTAG, ENDMV,
     &                ENGCNT, HIGHCT, I, INAUGH, ISEED, J, J1, K,
     &                LAYER, LMDA, LOWCT, MCSTEP,
     &                SAMPL, SRFINI, SYSCHG,
```

```
     &                    UNUM, VOLCNT, XDIR, YDIR, ZDIR,
     &                    XTMP, YTMP, ZTMP, XTMPA, YTMPA, ZTMPA
          REAL           RLAMB
C
CC  Array variables.
C
          DOUBLE PRECISION  DENSTY(LMBPTS),
     &                      HAMAVS(LMBPTS), HAMS2(LMBPTS),
     &                      HAMVAR(LMBPTS), LMDAVL(0:LMBPTS),
     &                      RGYLAM(LMBPTS), RGYRA2(NSAMP,LMBPTS),
     &                      RHOLMD(0:LMBPTS), RNNLAM(LMBPTS),
     &                      RNTON2(NSAMP,LMBPTS)
          INTEGER        BNDTOT(NSAMP)
          INTEGER        DIR(MONOMR-1,CHAINS), DIR1(MONXCH-CHAINS)
          EQUIVALENCE    (DIR(1,1), DIR1(1))
          INTEGER        DIRTMP(MONOMR-1,CHAINS), DIR1TP(MONXCH-CHAINS)
          EQUIVALENCE    (DIRTMP(1,1), DIR1TP(1))
          INTEGER        LAT(-DMD2:DMD2, -DMD2:DMD2, -DMD2:DMD2), LAT1(DM3)
          EQUIVALENCE    (LAT(-DMD2, -DMD2, -DMD2), LAT1(1))
          INTEGER        LATTMP(-DMD2:DMD2, -DMD2:DMD2, -DMD2:DMD2),
     &                      LAT1TP(DM3)
          EQUIVALENCE    (LATTMP(-DMD2, -DMD2, -DMD2), LAT1TP(1))
          INTEGER        LAYDEN(-DMD2:DMD2), LAYDNT(NSAMP, -DMD2:DMD2)
          INTEGER        TPD(MONOMR-1,CHAINS), TPX(MONOMR,CHAINS),
     &                      TPY(MONOMR,CHAINS), TPZ(MONOMR,CHAINS)
          INTEGER        XPOS(MONOMR,CHAINS), YPOS(MONOMR,CHAINS),
     &                      ZPOS(MONOMR,CHAINS)
          INTEGER        XPOS1(MONXCH), YPOS1(MONXCH), ZPOS1(MONXCH)
          EQUIVALENCE    (XPOS(1,1), XPOS1(1)), (YPOS(1,1), YPOS1(1)),
     &                      (ZPOS(1,1), ZPOS1(1))
          INTEGER        XPOSTP(MONOMR,CHAINS), YPOSTP(MONOMR,CHAINS),
     &                      ZPOSTP(MONOMR,CHAINS)
          INTEGER        XPS1TP(MONXCH), YPS1TP(MONXCH), ZPS1TP(MONXCH)
          EQUIVALENCE    (XPOSTP(1,1), XPS1TP(1)),
     &                     (YPOSTP(1,1), YPS1TP(1)),
     &                     (ZPOSTP(1,1), ZPS1TP(1))
C
CC  Function definitions.
C
          DOUBLE PRECISION  DRAN
          EXTERNAL          DRAN, ITIMER
C
CC  Initialization via  DATA  statements.
C
          DATA           HAMAVS, HAMS2/LMBPTS*0.0D+00, LMBPTS*0.0D+00/
          DATA             LMDAVL/0.00D+00,
     &                            0.01D+00, 0.02D+00, 0.05D+00,
     &                            0.10D+00, 0.20D+00, 0.30D+00,
     &                            0.40D+00, 0.50D+00, 0.60D+00,
     &                            0.70D+00, 0.80D+00, 0.90D+00,
     &                            0.93D+00, 0.96D+00, 0.98D+00, 1.00D+00/
          DATA             RHOLMD/LMBPT1*0.0D+00/, SUCCES/0.0D+00/
          DATA           ACCEPT, ATMPTS, BOXCNT, ENGCNT/0, 0, 0, 0/
          DATA             LAYDEN, LAYDNT/DM*0, DMXNSP*0/
```

```
      DATA          VOLCNT/0/
C
C  The initial seed for the random number generator, DRAN.
C
      DATA          ISEED/157663/
C
C  Definitions of important constants and variables.
C
C       BNDINI      = Total number of bends in the initial system.  Used
C                     to calculate the total system energy.
C       BNDNEW      = Total number of bends in the new system.  Used to
C                     calculate the total system energy.
C       BNDTOT(TS)  = Total number of bends in the system for a MC-step
C                     TS.  Used to calculate the total system energy.
C       BULKDN      = A running sum of the number of monomers that exist
C                     in the middle Z layers of the cubic lattice.
C                     Used to calculate the bulk density.
C       CHAINS      = The number of chains on the lattice.
C       DRAN        = The random number generator.  (See the subroutine
C                     for details.)
C       DIR(I,J)    = An array that holds the directions chosen, using
C                     DIRECT, for monomer I of chain J.  As the direc-
C                     tions are associated with the previous monomer,
C                     the dimensions are set to  (MONOMR-1, CHAINS).
C       DIRECT      = The value from the random number generator, DRAN,
C                     used to choose the new direction in which to
C                     place/move a monomer.
C       DM          = Dimension of one side of the cubic lattice, LAT.
C       DM3         = The number of sites in the cubic lattice = DM**3.
C       I           = A counter that always represents the monomer being
C                     considered.
C       ISEED       = The seed for the random number generator function,
C                     DRAN.
C       J           = A counter that always represents the chain being
C                     considered.
C       LAT(X,Y,Z)  = The lattice matrix onto which the positions of all
C                     of the monomers of all of the chains are mapped.
C       LAYDEN(X)   = The number of monomers on a given layer X for the
C                     initial conformation.
C       LAYDNT(S,X)= The number of monomers on a given layer X for the
C                     conformation of the S sample.
C       LAYER       = One of the allowed X values within the cubic
C                     lattice, LAT.
C       MONOMR      = The number of monomers in each chain.  (All chains
C                     are of the same length.)
C       NSAMP       = The number of Monte Carlo steps over which values
C                     are averaged.
C       XDIR        = A temporary variable used to store the X coordinate
C                     change derived from  DIRECT.
C       YDIR        = A temporary variable used to store the Y coordinate
C                     change derived from  DIRECT.
C       ZDIR        = A temporary variable used to store the Z coordinate
C                     change derived from  DIRECT.
```

```
C       XPOS(I,J)   = The value of the X coordinate of a monomer's site
C                       in periodic phase space.
C       YPOS(I,J)   = The value of the Y coordinate of a monomer's site
C                       in periodic phase space.
C       ZPOS(I,J)   = The value of the Z coordinate of a monomer's site
C                       in periodic phase space.  I is the number of the
C                       monomer within chain J.
C       XTMP        = A temporary variable used to refer to X layers.
C       YTMP        = A temporary variable used to refer to Y layers.
C       ZTMP        = A temporary variable used to refer to ZPOS(I,J)
C                       values.
C       ZTMPA       = A temporary variable used to rescale the  ZPOS(I,J)
C                       value into the lattice matrix to use the periodic
C                       boundary condition.
C
C    The dimensions of the matrix must be odd for the dimensions of the
C  LAT  matrix to be correct.  The tests are all arranged so that all
C  positions (X, Y, and Z) are symmetric around zero.
C
      CALL ITIMER(1)
      IF (MOD(DM,2) .EQ. 0) THEN
        STOP 'DM must be an odd integer.'
      ENDIF
C
C    As the integration to obtain the compressibility  Z  uses Simpson's
C  rule, the number of integration points  LMBPTS  must be an even value
C  for the integration to be valid.
C
      IF (MOD(LMBPTS,2) .NE. 0) THEN
        STOP 'LMBPTS must be even for the integration to be valid.'
      ENDIF
C
C     Test for possible integer overflows.
C  Maximum value is 2^31 - 1 = 2147483647.
C
      IF (DM2*NSAMP .GT. 2147483000) THEN
        STOP 'The integer limitations of  LAYDEN  have been exceeded.'
      ENDIF
      INAUGH = ISEED
C
C  ***  Generate an initial configuration.  ***
C
      UNUM = 50
      LAMBDA = LMDAVL(1)
      CALL INITCH (TAU,
     &             CHAINS, DM, DM2, DM3, DMD2, DMD4, MONOMR,
     &             HAMINI, LAMBDA,
     &             BNDINI, INAUGH, ISEED, SRFINI, UNUM,
     &             DIR, LAT, LAT1, LAYDEN, XPOS, YPOS, ZPOS)
C
      CALL EQUIL (TAU,
     &            CHAINS, DM, DM2, DM3, DMD2, MONOMR, MONXCH,
     &            HAMINI, LAMBDA,
     &            BNDINI, ISEED,
```

```
     &                DIR1, DIRTMP, DIR1TP, LAT1, LATTMP, LAT1TP,
     &                 TPD, TPX, TPY, TPZ, XPOS1, YPOS1, ZPOS1,
     &                 XPOSTP, YPOSTP, ZPOSTP,
     &                 XPS1TP, YPS1TP, ZPS1TP)
C
        DO 30000 LMDA = 1, LMBPTS
          LAMBDA = LMDAVL(LMDA)
          PRINT*, 'LAMBDA = ', LAMBDA
C
C  ***  Set the number of Monte Carlo steps to be performed.  ***
C
          DO 20000 CONFGS = 1, NSAMP+EQUILS
            ACCEPT = 0
C
C    MCLIM  is set so that, on average, each chain should move ten
C  times before a sample is taken.
C
            DO 10000 MCSTEP = 1, MCLIM
C
C    Initialize the temporary arrays.
C
              DO 11000 K = 1, MONXCH-CHAINS
                DIR1TP(K) = DIR1(K)
11000         CONTINUE
              DO 12000 K = 1, MONXCH
                XPS1TP(K) = XPOS1(K)
                YPS1TP(K) = YPOS1(K)
                ZPS1TP(K) = ZPOS1(K)
12000         CONTINUE
              DO 13000 K = 1, DM3
                LAT1TP(K) = LAT1(K)
13000         CONTINUE
C
C    Within  DO 14000, all changes to, and information from, Direction,
C  Position, and Lattice arrays must be done using the temporary arrays.
C
              DO 14000 J1 = 1, CHAINS
                ATMPTS = ATMPTS + 1
C
C  Choose a chain to move.
C
                J = DRAN(ISEED)*CHAINS + 1
C
C    Choose which end of the chain to move.  ENDMV  can either equal 1
C  or 2; if  ENDMV = 1, the first monomer is moved.  (ENDMV is reset to
C  1 for  symmetry.)  If  ENDMV = 2, the last monomer is moved, and
C  ENDMV is set to  MONOMR.
C
```

```
                ENDMV = DRAN(ISEED)*2 + 1
                IF (ENDMV .EQ. 1) THEN
                    ENDMV = 1
                    LOWCT = 2
                    HIGHCT = MONOMR
                    DIRTAG = 1
                    ARRSFT = -1
                  ELSE
                    ENDMV = MONOMR
                    LOWCT = 1
                    HIGHCT = MONOMR-1
                    DIRTAG = MONOMR-1
                    ARRSFT = 1
                ENDIF
C
C    The direction in which the end selected to move is chosen.  See
C  the test returning to this statement (GO TO 14100) concerning the
C  one disallowed move.
C
14100           DIRECT = (DRAN(ISEED) * 6.0) + 1
                IF (DIRECT .EQ. 1) THEN
                    XDIR =  1
                    YDIR =  0
                    ZDIR =  0
                    DIRECT = 1
                  ELSEIF (DIRECT .EQ. 2) THEN
                    XDIR = -1
                    YDIR =  0
                    ZDIR =  0
                    DIRECT = 1
                  ELSEIF (DIRECT .EQ. 3) THEN
                    XDIR =  0
                    YDIR =  1
                    ZDIR =  0
                    DIRECT = 2
                  ELSEIF (DIRECT .EQ. 4) THEN
                    XDIR =  0
                    YDIR = -1
                    ZDIR =  0
                    DIRECT = 2
                  ELSEIF (DIRECT .EQ. 5) THEN
                    XDIR =  0
                    YDIR =  0
                    ZDIR =  1
                    DIRECT = 3
                  ELSE
                    XDIR =  0
                    YDIR =  0
                    ZDIR = -1
                    DIRECT = 3
                ENDIF
C
C    Set the coordinates for the moved monomer.
C
```

```
                XTMP = XPOSTP(ENDMV,J) + XDIR
                YTMP = YPOSTP(ENDMV,J) + YDIR
                ZTMP = ZPOSTP(ENDMV,J) + ZDIR
C
C     This next test avoids the problem stated in the initialization of
C   IF loop 20.  As the chain would never move back on itself, the test
C   disallows an end to move to a position taken by its neighbor on that
C   chain.  The argument still stands as to whether this biases the
C   system;  I, of course, believe that it does not as such a move is
C   inherently prohibited.  (See Wall and Mandel, 1975.)
C
                IF (        (XTMP .EQ. XPOSTP(ENDMV-ARRSFT, J))
     &                .AND.  (YTMP .EQ. YPOSTP(ENDMV-ARRSFT, J))
     &                .AND.  (ZTMP .EQ. ZPOSTP(ENDMV-ARRSFT, J)) )
     &              GO TO 14100
C
C     See if the new site is "out-of-the-box."  That is, that the Z
C   coordinate has ventured past the limits of [-DMD2 <= Z <= DMD2].
C   One need not test for X or Y as periodic boundary conditions are
C   used.  The point of this test is to save some calculation time.
C
C   NOTE 1:  The test is different here than in single chain movement.
C            Here, one does not use one Monte Carlo step if a chain
C            moves out of the box.  In this case, the move is
C            disallowed and another chain is chosen to move.
C
C   NOTE 2:  The  GO TO 14000  jump here prevents vectorization of the
C            DO 14000 loop.  This is not critical as there are inner
C            loops which would be better to vectorize.  The IBM can
C            vectorize outer loops, however, depending upon the
C            circumstances.
C
C
                    IF ( (ZTMP .LT. -DMD2)   .OR.   (ZTMP .GT. DMD2) ) THEN
                      BOXCNT = BOXCNT + 1
                      GO TO 14000
                    ENDIF
C
C       ***  Excluded Volume Test for the chain being moved.  ***
C
C     The new lattice position for the end of the monomer that is moved
C   is calculated.  If this position is taken, then there are two
C   possibilities:  the chain has chosen a position occupied by its other
C   end, which by reptation is a valid move, or the site is occupied by
C   some other monomer, and the move is invalid.
C
                XTMPA = MOD( (DM + MOD(XTMP+DMD2, DM)), DM )
     &                       - DMD2
                YTMPA = MOD( (DM + MOD(YTMP+DMD2, DM)), DM )
     &                       - DMD2
                ZTMPA = ZTMP
                IF (LATTMP(XTMPA,YTMPA,ZTMPA) .EQ. 1) THEN
```

```
                    IF (ENDMV .EQ. 1) THEN
                        I = MONOMR
                      ELSE
                        I = 1
                    ENDIF
C
CC    XPOSTP(I,J), YPOSTP(I,J), and ZPOSTP(I,J)  represent the position
C   of the, as yet, unmoved end (tail) of the chain  J.
C
                    IF (       (XTMP .EQ. XPOSTP(I,J))
     &                  .AND. (YTMP .EQ. YPOSTP(I,J))
     &                  .AND. (ZTMP .EQ. ZPOSTP(I,J))   ) THEN
                        CONTINUE
                      ELSE
                        VOLCNT = VOLCNT + 1
                        GO TO 14000
                    ENDIF
                  ENDIF
C
CC    Zero the old tail position for the chain being moved.
C
                  IF (ENDMV .EQ. 1) THEN
                      XTMPA = MOD( (DM + MOD(XPOSTP(MONOMR,J)+DMD2, DM)),
     &                             DM ) - DMD2
                      YTMPA = MOD( (DM + MOD(YPOSTP(MONOMR,J)+DMD2, DM)),
     &                             DM ) - DMD2
                      ZTMPA = ZPOSTP(MONOMR,J)
                      LATTMP(XTMPA,YTMPA,ZTMPA) = 0
                    ELSE
                      XTMPA = MOD( (DM + MOD(XPOSTP(1,J)+DMD2, DM)), DM )
     &                         - DMD2
                      YTMPA = MOD( (DM + MOD(YPOSTP(1,J)+DMD2, DM)), DM )
     &                         - DMD2
                      ZTMPA = ZPOSTP(1,J)
                      LATTMP(XTMPA,YTMPA,ZTMPA) = 0
                  ENDIF
C
CC    Store the new positions and direction, and translate the others
C   from the previous configuration into temporary storage.  By using
C   the variables LOWCT, HIGHCT, and ARRSFT, the same code can be used
C   for the movement of either end of the chain.
C
CC    Insert the new positions into the Direction and Position
C   Transfer arrays (TPD, TPX, TPY, TPZ).
C
                  TPD(DIRTAG,J) = DIRECT
                  TPX(ENDMV,J) = XTMP
                  TPY(ENDMV,J) = YTMP
                  TPZ(ENDMV,J) = ZTMP
                  DO 14200 I = LOWCT, HIGHCT-1
                    TPD(I,J) = DIRTMP(I+ARRSFT, J)
14200             CONTINUE
```

```
                DO 14300 I = LOWCT, HIGHCT
                  TPX(I,J) = XPOSTP(I+ARRSFT, J)
                  TPY(I,J) = YPOSTP(I+ARRSFT, J)
                  TPZ(I,J) = ZPOSTP(I+ARRSFT, J)
14300           CONTINUE
C
C    Shift the new values back into the temporary Direction and Position
C  arrays (DIRTMP, XPOSTP, YPOSTP, ZPOSTP).
C
                DO 14400 I = 1, MONOMR-1
                  DIRTMP(I,J) = TPD(I,J)
14400           CONTINUE
                DO 14500 I = 1, MONOMR
                  XPOSTP(I,J) = TPX(I,J)
                  YPOSTP(I,J) = TPY(I,J)
                  ZPOSTP(I,J) = TPZ(I,J)
14500           CONTINUE
C
C    Flag the new head position for the chain being moved.
C
                IF (ENDMV .EQ. 1) THEN
                    XTMPA = MOD( (DM + MOD(XPOSTP(1,J)+DMD2, DM)), DM )
     &                         - DMD2
                    YTMPA = MOD( (DM + MOD(YPOSTP(1,J)+DMD2, DM)), DM )
     &                       - DMD2
                    ZTMPA = ZPOSTP(1,J)
                    LATTMP(XTMPA,YTMPA,ZTMPA) = 1
                  ELSE
                    XTMPA = MOD( (DM + MOD(XPOSTP(MONOMR,J)+DMD2, DM)),
     &                           DM ) - DMD2
                    YTMPA = MOD( (DM + MOD(YPOSTP(MONOMR,J)+DMD2, DM)),
     &                           DM ) - DMD2
                    ZTMPA = ZPOSTP(MONOMR,J)
                    LATTMP(XTMPA,YTMPA,ZTMPA) = 1
                ENDIF
C
C            ***  All the chains have been moved.  ***
C
14000          CONTINUE
C
C  ***  The Excluded Volume Test.  ***
C
C
C  ***  The Energy Test.  ***
C
C    NOTE:  The method of testing the energy of the system must incor-
C           porate two effects:  the energy due to bends, and the
C           repulsive energy.  The former is determined by counting
C           the number of bends in all of the chains in the system
C           (loop 15000).  The repulsive energy is determined by
C           counting the number of monomers presently on the top
C           layer (SRFNEW), and multiplying this by the temperature
C           factor (TAU) and the weighting of the repulsive energy
C           ( ln(LAMBDA) ).  The sum of these two energies is called,
```

```
C           here, the pseudo-Hamiltonian as the quantity differs
C           from the Hamiltonian only by the temperature factor.
C
C             Finally, the values of the old (HAMINI) and new (HAMNEW)
C           pseudo-Hamiltonians are compared.  If the new value is less
C           than the old value, the new configuration is immediately
C           accepted.  If the new value is greater, the Boltzmann
C           factor (BOLTZ) is calculated using the difference between
C           the pseudo-Hamiltonian energies.  (The Boltzmann factor
C           must be between 0 and 1, here, so that a direct test
C           against a random number returned from  DRAN  is possible.)
C
C       ==> This test of the energies is the essence of Metropolis
C       ==> sampling.
C
                BNDNEW = 0
                DO 15000 J = 1, CHAINS
                  DO 15100 I = 1, MONOMR-2
                    IF (DIRTMP(I,J) .NE. DIRTMP(I+1,J))
     &                BNDNEW = BNDNEW+1
15100             CONTINUE
15000           CONTINUE
                IF (INT(LAMBDA + 1.0D-6) .EQ. 1) THEN
                    DUMSM2 = 0.0D+00
                  ELSE
                    DUMSM2 = 0.0D+00
                    DO 16000 K = 1, DM2
                      DUMSM2 = DUMSM2 + LAT1TP(K + (DM-1)*DM2)
16000               CONTINUE
                ENDIF
                SRFNEW = DUMSM2
                HAMNEW = DBLE(BNDNEW) - (SRFNEW * TAU * DLOG(LAMBDA))
                IF (HAMNEW .GT. HAMINI) THEN
C
C    The test is done on  DELHT  to avoid floating underflow problems
C  with the  DEXP  function.
C
                  DELHT = (HAMNEW - HAMINI) / TAU
                  IF (DELHT .GE. 170.0D+00) THEN
                      ENGCNT = ENGCNT + 1
                      GO TO 10000
                    ELSE
                      BOLTZ = DEXP( - DELHT )
                      IF (BOLTZ .LT. DRAN(ISEED)) THEN
                        ENGCNT = ENGCNT + 1
                        GO TO 10000
                      ENDIF
                  ENDIF
                ENDIF
C
C    ***  The new conformations are acceptable.  Transfer all   ***
C    ***   of the temporary values into the permanent arrays.   ***
C
```

```
                ACCEPT = ACCEPT + 1
                BNDINI = BNDNEW
                HAMINI = HAMNEW
                DO 17000 K = 1, MONXCH-CHAINS
                  DIR1(K) = DIR1TP(K)
17000           CONTINUE
                DO 18000 K = 1, MONXCH
                  XPOS1(K) = XPS1TP(K)
                  YPOS1(K) = YPS1TP(K)
                  ZPOS1(K) = ZPS1TP(K)
18000           CONTINUE
                DO 19000 K = 1, DM3
                  LAT1(K) = LAT1TP(K)
19000           CONTINUE
C
C    ***          This is the end of the  MCSTEP  loop.          ***
C
10000       CONTINUE
C
C  ***  The Measurement Section.  ***
C
C    The system is given  EQUILS  steps to come to equilibrium, or at
C  least to shed the initial character of the system, before statistics
C  are accumulated.  The number of samples is  NSAMP, and  SAMPL
C  serves as the counter.
C
            IF (CONFGS .GT. EQUILS) THEN
              SUCCES = SUCCES + ACCEPT
              SAMPL = CONFGS - EQUILS
              BNDTOT(SAMPL) = BNDINI
              HAMAVS(LMDA) = HAMAVS(LMDA) + HAMINI
              HAMS2(LMDA) = HAMS2(LMDA) + HAMINI**2
              RNN2 = 0.0D+00
              RGY2 = 0.0D+00
              DO 21000 J = 1, CHAINS
                RNN2 = RNN2 + DBLE( (XPOS(MONOMR,J) - XPOS(1,J))**2
     &                            + (YPOS(MONOMR,J) - YPOS(1,J))**2
     &                            + (ZPOS(MONOMR,J) - ZPOS(1,J))**2 )
                XCM = 0.0D+00
                YCM = 0.0D+00
                ZCM = 0.0D+00
                DO 21100 K = 1, MONOMR
                  XCM = XCM + DBLE(XPOS(K,J))
                  YCM = YCM + DBLE(YPOS(K,J))
                  ZCM = ZCM + DBLE(ZPOS(K,J))
21100           CONTINUE
                XCM = XCM / DBLE(MONOMR)
                YCM = YCM / DBLE(MONOMR)
                ZCM = ZCM / DBLE(MONOMR)
```

```
                DO 21200 K = 1, MONOMR
                  RGY2 = RGY2 + (DBLE(XPOS(K,J)) - XCM)**2
     &                        + (DBLE(YPOS(K,J)) - YCM)**2
     &                        + (DBLE(ZPOS(K,J)) - ZCM)**2
21200             CONTINUE
21000           CONTINUE
                RNTON2(SAMPL,LMDA) = RNN2 / DBLE(CHAINS)
                RGYRA2(SAMPL,LMDA) = RGY2 / DBLE(CHAINS * MONOMR)
C
C    This loop (DO 22000) is specifically designed to be vectorized by
C  vector reduction, a method used throughout this program.  Vector
C  reduction cannot be done on integers, only on  REAL  quantities;
C  that is, Reals or Double Precision values.
C
C    LAT1 allows the summation on the X-Y planes to occur as
C  LAT(X,Y,Z)  has X changing most rapidly, then Y, then finally Z.
C  As such, every DM2 steps through LAT1 a new layer is reached.  LAYER
C  serves to increment this change, and LAYDNT holds the number of
C  monomers present on each X-Y plane.
C
              DO 22000 LAYER = -DMD2, DMD2
                DUMSM2 = 0.0D+00
                DO 22100 K = 1, DM2
                  DUMSM2 = DUMSM2 + LAT1(K + (LAYER+DMD2)*DM2)
22100           CONTINUE
                LAYDNT(SAMPL,LAYER) = IDNINT(DUMSM2)
22000         CONTINUE
              ENDIF
C
C         ***  This is the end of the  CONFGS  loop.  ***
C
20000     CONTINUE
C
C    The average pseudo-Hamiltonian for the present value of LAMBDA
C  (HAMAVS), the average pseudo-Hamiltonian squared (HAMS2), and the
C  average end-to-end distance of the polymer chains (RNTON) over
C  NSAMP  tries are calculated.
C
          HAMAVS(LMDA) = HAMAVS(LMDA) / DBLE(NSAMP)
          HAMS2(LMDA) = HAMS2(LMDA) / DBLE(NSAMP)
          DUMSM1 = 0.0D+00
          DO 31000 K = 1, NSAMP
            DUMSM1 =  DUMSM1 + RNTON2(K,LMDA)
31000     CONTINUE
          RNNLAM(LMDA) = DUMSM1 / DBLE(NSAMP)
          DUMSM1 = 0.0D+00
          DO 31500 K = 1, NSAMP
            DUMSM1 =  DUMSM1 + RGYRA2(K,LMDA)
31500     CONTINUE
          RGYLAM(LMDA) = DUMSM1 / DBLE(NSAMP)
C
C    The average density of monomers on the top layer (RHO) for NSAMP
C  tries is calculated so that the pressure for the system may be
C  (eventually) determined.
```

```
C
C  NOTE:  The IF test is used because at  LAMBDA = 1.0, the number of
C         monomers at the top most layer will be random.  As such,
C         R. Dickman suggests using one layer removed from the bottom
C         to obtain a reasonable value.  See the paper.
C
          IF (INT(LAMBDA + 1.0D-06) .EQ. 1) THEN
              DUMSM1 = 0.0D+00
              DO 32000 K = 1, NSAMP
                DUMSM1 = DUMSM1 + LAYDNT(K, -DMD2+1)
32000         CONTINUE
            ELSE
              DUMSM1 = 0.0D+00
              DO 33000 K = 1, NSAMP
                DUMSM1 = DUMSM1 + LAYDNT(K, DMD2)
33000         CONTINUE
          ENDIF
          RHO = DUMSM1 / DBLE(NSAMP) / DBLE(DM2)
          RHOLMD(LMDA) = RHO / LAMBDA
C
C    This is the calculation of the density, DENSTY, for the given
C  LAMBDA  value being executed.  The value is taken from the number of
C  monomers that reside in the middle layers of the box at each sample.
C
C  NOTE:  The number of sites in the middle of the lattice is just
C         DMD2*DM2; that is, half the total number of layers (via
C         integer arithmetic) times the number of sites on each layer.
C
          DENSTY(LMDA) = 0.0D+00
          DO 34000 LAYER = -DMD4, DMD4
            DUMSM1 = 0.0D+00
            DO 34100 K = 1, NSAMP
              DUMSM1 = DUMSM1 + LAYDNT(K,LAYER)
34100       CONTINUE
            DENSTY(LMDA) = DENSTY(LMDA) + DUMSM1
34000     CONTINUE
          DENSTY(LMDA) = DENSTY(LMDA) / DBLE(NSAMP) / DBLE(DMD2*DM2)
C
C              ***  End of the  LAMBDA  loop.  ***
C
30000   CONTINUE
C
C    Averages for the pseudo-Hamiltonian (HAMLTN), bulk density (PHIB),
C  and end-to-end distances (RNNAVG) for the simulation are calculated.
C
        HAMLTN = 0.0D+00
        PHIB   = 0.0D+00
        RNNAVG = 0.0D+00
        RGYAVG = 0.0D+00
```

```
      DO 4000 K = 1, LMBPTS
        HAMLTN = HAMLTN + HAMAVS(K)
        PHIB   = PHIB + DENSTY(K)
        RNNAVG = RNNAVG + RNNLAM(K)
        RGYAVG = RGYAVG + RGYLAM(K)
        HAMVAR(K) = HAMS2(K) - HAMAVS(K)**2
4000  CONTINUE
      HAMLTN = HAMLTN / DBLE(LMBPTS)
      PHIB   = PHIB / DBLE(LMBPTS)
      RNNAVG = RNNAVG / DBLE(LMBPTS)
      RGYAVG = RGYAVG / DBLE(LMBPTS)
      STIFF  = DLOG(RNNAVG) / DLOG(DBLE(MONOMR - 1))
C
C     The next calculation gives the fraction of attempts to change
C  the PERMANENT configuration of the system that were successful.
C  This value (SUCCES) only depends on the results of the energy test.
C
C  NOTE 1:  The denominator is obtained from the maximum values for
C           the counters of the loops involved:
C
C                        LMBPTS  * NSAMP * MCLIM.
C
C  NOTE 2:  To calculate all of the changes that occur within the
C           temporary lattice, add the BOXCNT and VOLCNT values,
C           and use (ATMPTS - sum)/ATMPTS.
C
      SUCCES = SUCCES / DBLE(LMBPTS*NSAMP) / DBLE(MCLIM)
C
C     The pressure for the system is calculated using the Simpson's
C  Rule.  This integration uses the knowledge that the pressure
C  must be zero at  LAMBDA = 0  so that one end point is taken as
C  zero and the other as  RHOLMD(LMBPTS).
C
      PRESS = 0.0D+00
      DO 5000 K = 1, LMBPTS-1, 2
        PRESS = PRESS
     &          + ((LMDAVL(K+1) - LMDAVL(K)) / 3.0D+00)
     &            * (RHOLMD(K-1) + 4.0D+00*RHOLMD(K) + RHOLMD(K+1))
5000  CONTINUE
C
C     Finally, the compressibility (Z) for the system is calculated
C  along with the compressibility that would be obtained from the
C  three-dimensional Flory-Huggins theory (ZTHEOR).
C
      Z     = PRESS * DBLE(MONOMR) / PHIB
      ALPHA = (1.0D+00 - 1.0D+00/DBLE(MONOMR)) / 3.0D+00
      ZTHEOR = (DBLE(MONOMR)/PHIB)
     &         * ( 3.0D+00 * DLOG(1.0D+00 - ALPHA*PHIB)
     &             -  DLOG(1.0D+00 - PHIB) )
C
C          ******************  DONE  ******************
C
```

```
C*****************************************************************
C                      THE SECOND PRINTING SECTION.
C*****************************************************************
C
C       OPEN (UNIT=50)
        WRITE(50,8000)
8000    FORMAT (/, /, ' ***  This is the information for the final ',
     &          'configuration of the system.  ***', /)
        WRITE(50,8510)
8510    FORMAT(' ---  Results  ---',/)
        SYSCHG = NSAMP * SUCCES
        WRITE (50,8520)
8520    FORMAT(6X, 'PHIB', 8X, 'SYSCHG', 6X, 'RNNAVG', 10X, 'RGYAVG',
     &         10X, 'STIFF', 13X, 'Z', 13X, 'ZTHEOR')
        WRITE (50, 8530) PHIB, SYSCHG, RNNAVG, RGYAVG, STIFF, Z, ZTHEOR
8530    FORMAT(1X, 2PD13.5, 3X, I6, 3X, 2PD13.5, 3X, 2PD13.5,
     &          3( 3X, 1PD13.6), /)
        WRITE (50, 8540)
8540    FORMAT(5X, 'HAMLTN', 10X, 'PRESS')
        WRITE (50, 8550) HAMLTN, PRESS
8550    FORMAT(1X, 1PD13.6, 3X, 1PD13.6, /)
        WRITE(50,8600)
8600    FORMAT(' ---  System Parameters  ---', /)
        WRITE (50,8610)
8610    FORMAT (2X, 'CHAINS', 4X, 'MONOMR', 6X, 'DM', 10X, 'TAU',
     &          13X, 'INAUGH')
        WRITE (50,8620) CHAINS, MONOMR, DM, TAU, INAUGH
8620    FORMAT (1X, I5, 6X, I4, 6X, I4, 5X, 1PD13.6, 5X, I10, /)
        WRITE(50,8700)
8700    FORMAT(' ---  Statistics For This Execution  ---', /)
        WRITE (50, 8710)
8710    FORMAT (5X, 'SUCCES', 9X, 'NSAMP', 6X, 'EQUILS', 5X, 'TIMLIM',
     &          6X, 'ATMPTS', 7X, 'ENGCNT', 7X, 'VOLCNT', 7X, 'BOXCNT')
        WRITE (50, 8720) SUCCES, NSAMP, EQUILS, TIMLIM, ATMPTS,
     &                   ENGCNT, VOLCNT, BOXCNT
8720    FORMAT (1X, 2PD13.6, 5X, I6, 5X, I6, 5X, I6, 4( 3X, I10), /)
        WRITE(50,8800)
8800    FORMAT(' ---  Checks and Raw Information  ---', /)
        WRITE(50,8810)
8810    FORMAT(' LAMBDA', 7X, 'DENSTY', 10X, 'HAMAVS', 10X,
     &          'HAMVAR', 10X, 'RNNLAM', 10X, 'RGYLAM', 10X,
     &          'RHOLMD', 7X, 'Avg Num on H', /)
        DO 8820 K = 1, LMBPTS
          RLAMB = SNGL(LMDAVL(K))
          RHO = RHOLMD(K) * LMDAVL(K) * DBLE(DM2)
          WRITE(50,8830) RLAMB, DENSTY(K), HAMAVS(K),
     &       HAMVAR(K), RNNLAM(K), RGYLAM(K), RHOLMD(K), RHO
8820    CONTINUE
8830    FORMAT(2X, F4.2, 1X, 7(3X, 1PD13.6))
        WRITE(50,8900)
8900    FORMAT(/)
C
C
        CLOSE (UNIT=50)
```

```
C
C*************************************************************************
C                          END SECOND PRINTING SECTION
C*************************************************************************
C
        CALL ITIMER(1)
        STOP
        END
C
C
C
        SUBROUTINE INITCH (TAU,
     &                     CHAINS, DM, DM2, DM3, DMD2, DMD4, MONOMR,
     &                     HAMINI, LAMBDA,
     &                     BNDINI, INAUGH, ISEED, SRFINI, UNUM,
     &                     DIR, LAT, LAT1, LAYDEN, XPOS, YPOS, ZPOS)
        IMPLICIT NONE
C
C  Parameters passed.
C
        DOUBLE PRECISION  TAU
        INTEGER   CHAINS, DM, DM2, DM3, DMD2, DMD4, MONOMR
C
C  Scalar variables passed.
C
        DOUBLE PRECISION  HAMINI, LAMBDA
        INTEGER   BNDINI, INAUGH, ISEED, SRFINI, UNUM
C
C  Array variables passed.
C
        INTEGER   DIR(MONOMR-1,CHAINS),
     &            LAT(-DMD2:DMD2, -DMD2:DMD2, -DMD2:DMD2),
     &            LAT1(DM3), LAYDEN(-DMD2:DMD2),
     &            XPOS(MONOMR,CHAINS), YPOS(MONOMR,CHAINS),
     &            ZPOS(MONOMR,CHAINS)
C
C  Local scalar variables.
C
        DOUBLE PRECISION DUMSM2
        INTEGER  BULKDN, CHNATM, DIRECT, I, INITRY, J, K, LAYER, MONATM,
     &           XDIR, YDIR, ZDIR, XTMP, YTMP, ZTMP, XTMPA, YTMPA
C
C  Local array variables.
C
C       INTEGER  LAYDEN(-DMD2:DMD2)
C
C  NOTE:  In FORTRAN 77 there is no such entity as a local array
C         variable.  All arrays must be declared and dimensioned in
C         both the main program and the subroutine, *and* all arrays
C         must be passed by name in the subroutine call or in a COMMON
C         statement.
C
C  Function declarations.
C
```

```
      DOUBLE PRECISION  DRAN
      EXTERNAL          DRAN
      DATA              CHNATM, INITRY, MONATM/0, 0, 0/
C
   50 INAUGH = ISEED
      INITRY = INITRY + 1
      IF (INITRY .GT. 1000) THEN
        STOP 'Initialization failed.'
      ENDIF
      CHNATM = 0
      DO 100 K = 1, DM3
        LAT1(K) = 0
  100 CONTINUE
C
C  ***  Generate the initial conformations of the chains.  ***
C
      DO 200 J = 1, CHAINS
C
C  Test to avoid an infinite loop in placing the initial monomer.
C
  210   CHNATM = CHNATM + 1
        IF (CHNATM .GT. 5*CHAINS) THEN
          ISEED = IABS(MOD(ISEED*25173 + 13849, 1073741824))
          GO TO 50
        ENDIF
C
C    Choose the initial site at which to begin the chain.
C
        XTMP = INT(DRAN(ISEED) * DM) - DMD2
        YTMP = INT(DRAN(ISEED) * DM) - DMD2
        ZTMP = INT(DRAN(ISEED) * DM) - DMD2
C
C    Test to see if the site is already occupied.  If so, obtain a new
C  initial position.
C
C  NOTE:  The lattice checking method that is used here is only
C         reasonable in terms of time up to approximately 10^6 sites,
C         according to Jyh-Shyong Ho.  Here, that translates into a
C         cubic box having an edge of 100 sites.
C
        IF (LAT(XTMP,YTMP,ZTMP) .EQ. 1) GO TO 210
C
C    Mark the site as occupied, and store the positions
C  permanently.
C
        LAT(XTMP,YTMP,ZTMP) = 1
        XPOS(1,J) = XTMP
        YPOS(1,J) = YTMP
        ZPOS(1,J) = ZTMP
C
C    Choose the direction to lay down the next monomer (2 To MONOMR)
C  using a random number.  The value of  DIRECT  is between one and six
C  as there are six possible sites.  One site is of course occupied by
C  the previous monomer so only a maximum of five sites are available.
```

```
C  The idea is that it is easier to ignore one out of six than to
C  program around the problem.
C
           MONATM = 0
           DO 220 I = 2, MONOMR
C
CC    Test to be sure that the calculation has not been trapped into an
C  infinite loop.
C
  221        MONATM = MONATM + 1
             IF (MONATM .GT. 5*MONOMR) THEN
               ISEED = IABS(MOD(ISEED*25173 + 13849, 1073741824))
               GO TO 50
             ENDIF
C
CC    Randomly select a direction for the next monomer.  Store the
C  direction chosen in  DIR(I-1,J) after the move has been tested.
C
             DIRECT = (DRAN(ISEED) * 6.0) + 1
             IF (DIRECT .EQ. 1) THEN
                 XDIR =  1
                 YDIR =  0
                 ZDIR =  0
                 DIRECT = 1
               ELSEIF (DIRECT .EQ. 2) THEN
                 XDIR = -1
                 YDIR =  0
                 ZDIR =  0
                 DIRECT = 1
               ELSEIF (DIRECT .EQ. 3) THEN
                 XDIR =  0
                 YDIR =  1
                 ZDIR =  0
                 DIRECT = 2
               ELSEIF (DIRECT .EQ. 4) THEN
                 XDIR =  0
                 YDIR = -1
                 ZDIR =  0
                 DIRECT = 2
               ELSEIF (DIRECT .EQ. 5) THEN
                 XDIR =  0
                 YDIR =  0
                 ZDIR =  1
                 DIRECT = 3
               ELSE
                 XDIR =  0
                 YDIR =  0
                 ZDIR = -1
                 DIRECT = 3
             ENDIF
C
CC    Set the coordinates for the next monomer.
C
```

```
                XTMP = XPOS(I-1,J) + XDIR
                YTMP = YPOS(I-1,J) + YDIR
                ZTMP = ZPOS(I-1,J) + ZDIR
C
CCC
C     Perform various tests to determine if the site selected is valid.
C   If not, obtain a new direction (SN 221) and try again.
C
C     1.  See if the new site is "out-of-the-box."  That is, that the Z
C   coordinate has ventured past the limits of [-DMD2 <= Z <= DMD2].
C   One need not test for X or Y as periodic boundary conditions are
C   used.
C
                IF ( (ZTMP .LT. -DMD2)  .OR.  (ZTMP .GT. DMD2) ) GO TO 221
C
C     2.  Determine if any other monomers occupy the site.
C   This requires mapping the  XPOS  and  YPOS  values back into
C   the original box (i.e, the periodic boundaries) and then testing
C   against the sites occupied in the  LAT  array.
C
C   NOTE:  The  MOD  function is the modulus function, usually done in
C          integer arithmetic, and equivalent to
C                        MOD(A,B) = A - (A/B)*B
C          and provides the remainder from the division of A/B.
C
                XTMPA = MOD( (DM + MOD(XTMP+DMD2, DM)), DM ) - DMD2
                YTMPA = MOD( (DM + MOD(YTMP+DMD2, DM)), DM ) - DMD2
                IF (LAT(XTMPA,YTMPA,ZTMP) .EQ. 1) GO TO 221
C
C     All tests have been passed; store the values into permanent
C   arrays.
C
C     NOTE:  The direction for the bond is associated with the previous
C            monomer.  This is done to make vectorization more efficient
C            by starting with the first element in the array.
C
                LAT(XTMPA,YTMPA,ZTMP) = 1
                XPOS(I,J) = XTMP
                YPOS(I,J) = YTMP
                ZPOS(I,J) = ZTMP
                DIR(I-1,J) = DIRECT
  220         CONTINUE
  200     CONTINUE
C
C     Count the number of bends in the all of the chains to calculate
C   the bend energy's contribution to the system's total energy.
C
C   NOTE:  The counter on I runs from 1 (direction associated with the
C          first monomer) to  MONOMR-2 (direction associated with the
C          monomer two from the end of the chain).  The latter limit is
C          used as the  IF  statement tests against  I+1 (the direction
C          associated with the penultimate monomer, which is the last
C          monomer of interest here).
C
```

```
        BNDINI = 0
        DO 300 J = 1, CHAINS
          DO 310 I = 1, MONOMR-2
C
C  NOTE:  The test is I versus I+1 in the array as this keeps the
C         computer moving forward in the array.  Backwards references
C         are to be avoided.
C
            IF (DIR(I,J) .NE. DIR(I+1,J)) BNDINI = BNDINI+1
  310     CONTINUE
  300   CONTINUE
C
C    Count the number of monomers on each layer to get the bulk density
C  and the pressure of the system.
C
C    This loop (DO 400) is specifically designed to be vectorized
C  using vector reduction.  LAT1  allows the summation on the X-Y
C  planes to occur as  LAT(X,Y,Z)  has X changing most rapidly, then
C  Y, then finally Z.  As such, every  DM2  steps through  LAT1  a new
C  is reached.
C
              DO 400 LAYER = -DMD2, DMD2
                DUMSM2 = 0.0D+00
                DO 410 K = 1, DM2
                  DUMSM2 = DUMSM2 + LAT1(K + (LAYER+DMD2)*DM2)
  410           CONTINUE
                LAYDEN(LAYER) = IDNINT(DUMSM2)
  400         CONTINUE
C
C    Calculate the number of monomers on the top layer (SRFINI) and
C  the pseudo-Hamiltonian energy for the initial system.
C
              SRFINI = LAYDEN(DMD2)
              HAMINI = BNDINI - (SRFINI * TAU * DLOG(LAMBDA))
C
C    Calculate the total number of monomers on the middle layers of the
C  cubic lattice.
C
        BULKDN = 0
        DO 500 LAYER = -DMD4, DMD4
          BULKDN = BULKDN + LAYDEN(LAYER)
 500    CONTINUE
C
C
C*****************************************************************
C                        FIRST PRINTING SECTION
C*****************************************************************
C
        OPEN (UNIT=UNUM)
        WRITE(UNUM,8000)
8000    FORMAT (/, '                                  MULTI-CHAIN MOVE', /,
     &             '                                COOP FORTRAN (5/03/88)', /)
        WRITE(UNUM, 9000)
9000    FORMAT (' For the initial configuration:', /)
```

```
      WRITE(UNUM,9010) BNDINI
9010  FORMAT(' Number of bends in the system (BNDINI) = ', I6)
      WRITE(UNUM,9020) SRFINI
9020  FORMAT(' Number of monomers on the surface (SRFINI) = ', I6)
      WRITE(UNUM,9030) HAMINI
9030  FORMAT(' Value for the Pseudo-Hamiltonian (HAMINI) = ',
     &       1PD12.5)
      WRITE(UNUM,9040) BULKDN
9040  FORMAT(' Number of monomers in the middle layers (BULKDN) = ',
     &       I6, /)
      DO 9050 LAYER = -DMD2, DMD2
        WRITE(UNUM,9060) LAYER, LAYDEN(LAYER)
9050  CONTINUE
9060  FORMAT(' ', 'Density on LAYER', I4, ' = ', I6)
C
C  This section, which has been disabled, prints the positions of all
C  the monomers in the initial configuration.
C
C     WRITE(UNUM,9070)
C9070  FORMAT(/, ' CHAINS', 3X, 'MONOMER', 3X, 'DIR(I,J)', 6X,
C    &         'XTMP', 5X, 'YTMP', 7X, '  ZTMP')
C     DO 9080 J=1,CHAINS
C       DO 9085 I=1,MONOMR
C         IF (I .EQ. MONOMR) THEN
C           WRITE(UNUM,9090) J,I, XPOS(I,J), YPOS(I,J), ZPOS(I,J)
C         ELSE
C           WRITE(UNUM,9100) J,I, DIR(I,J), XPOS(I,J), YPOS(I,J),
C    &        ZPOS(I,J)
C         ENDIF
C9085   CONTINUE
C9080   CONTINUE
C9090   FORMAT(' ', 2X, I2, 7X, I2, 19X, I4, 5X, I4, 5X, I8)
C9100   FORMAT(' ', 2X, I2, 7X, I2, 9X, I2, 8X, I4, 5X, I4, 5X, I8)
C
C     CLOSE (UNIT=UNUM)
C
C*****************************************************************
C               END FIRST PRINTING SECTION
C*****************************************************************
C
      PRINT*, 'INITRY, CHNATM, MONATM', INITRY, CHNATM, MONATM
      RETURN
      END
C
C
C
      SUBROUTINE EQUIL (TAU,
     &                  CHAINS, DM, DM2, DM3, DMD2, MONOMR, MONXCH,
     &                  HAMINI, LAMBDA,
     &                  BNDINI, ISEED,
     &                  DIR1, DIRTMP, DIR1TP, LAT1, LATTMP, LAT1TP,
     &                   TPD, TPX, TPY, TPZ, XPOS1, YPOS1, ZPOS1,
     &                   XPOSTP, YPOSTP, ZPOSTP,
     &                   XPS1TP, YPS1TP, ZPS1TP)
```

```
C
      IMPLICIT NONE
C
CCC   Parameters passed.
C
      DOUBLE PRECISION  TAU
      INTEGER           CHAINS, DM, DM2, DM3, DMD2, MONOMR, MONXCH
C
CCC   Scalar variables passed.
C
      DOUBLE PRECISION  HAMINI, LAMBDA
      INTEGER           BNDINI, ISEED
C
CCC   Array variables passed.
C
      INTEGER           DIR1(MONXCH-CHAINS),
     &                   DIRTMP(MONOMR-1,CHAINS),
     &                   DIR1TP(MONXCH-CHAINS)
      INTEGER           LAT1(DM3),
     &                   LATTMP(-DMD2:DMD2, -DMD2:DMD2, -DMD2:DMD2),
     &                   LAT1TP(DM3)
      INTEGER           TPD(MONOMR-1,CHAINS), TPX(MONOMR,CHAINS),
     &                   TPY(MONOMR,CHAINS), TPZ(MONOMR,CHAINS)
      INTEGER           XPOS1(MONXCH), YPOS1(MONXCH), ZPOS1(MONXCH),
     &                   XPOSTP(MONOMR,CHAINS), YPOSTP(MONOMR,CHAINS),
     &                    ZPOSTP(MONOMR,CHAINS),
     &                   XPS1TP(MONXCH), YPS1TP(MONXCH),
     &                    ZPS1TP(MONXCH)
C
C     Local scalar variables.
C
      DOUBLE PRECISION  BOLTZ, DELHT, DUMSM2, HAMNEW,
     &                   SRFNEW, SUCCES
      INTEGER           ACCEPT, ARRSFT, BNDNEW,
     &                   CONFGS, DIRECT, DIRTAG, ENDMV, EQUILS,
     &                   HIGHCT, I, J, J1, K,
     &                   LOWCT, MCLIM, MCSTEP,
     &                   XDIR, YDIR, ZDIR,
     &                   XTMP, YTMP, ZTMP, XTMPA, YTMPA, ZTMPA
C
CC    Function definitions.
C
      DOUBLE PRECISION  DRAN
      EXTERNAL          DRAN
C
C  ***  Set the number of Monte Carlo steps to be performed.  ***
C
C   Note:  MCLIM  is set so that, on average, each chain should move ten
C          times before a sample is taken.
C
```

```
          EQUILS = 5000
          MCLIM  = 10
          ACCEPT = 0
          DO 20000 CONFGS = 1, EQUILS
C
            DO 10000 MCSTEP = 1, MCLIM
C
C     Initialize the temporary arrays.
C
              DO 11000 K = 1, MONXCH-CHAINS
                DIR1TP(K) = DIR1(K)
11000         CONTINUE
              DO 12000 K = 1, MONXCH
                XPS1TP(K) = XPOS1(K)
                YPS1TP(K) = YPOS1(K)
                ZPS1TP(K) = ZPOS1(K)
12000         CONTINUE
              DO 13000 K = 1, DM3
                LAT1TP(K) = LAT1(K)
13000         CONTINUE
C
C     Within  DO 14000, all changes to, and information from, Direction,
C   Position, and Lattice arrays must be done using the temporary arrays.
C
              DO 14000 J1 = 1, CHAINS
C
C   Choose a chain to move.
C
                J = DRAN(ISEED)*CHAINS + 1
C
C     Choose which end of the chain to move.  ENDMV  can either equal 1
C   or 2; if  ENDMV = 1, the first monomer is moved.  (ENDMV is reset to
C   1 for  symmetry.)  If  ENDMV = 2, the last monomer is moved, and
C   ENDMV is set to  MONOMR.
C
                ENDMV = DRAN(ISEED)*2 + 1
                IF (ENDMV .EQ. 1) THEN
                    ENDMV = 1
                    LOWCT = 2
                    HIGHCT = MONOMR
                    DIRTAG = 1
                    ARRSFT = -1
                  ELSE
                    ENDMV = MONOMR
                    LOWCT = 1
                    HIGHCT = MONOMR-1
                    DIRTAG = MONOMR-1
                    ARRSFT = 1
                ENDIF
C
C     The direction in which the end selected to move is chosen.  See the
C   test returning to this statement (GO TO 14100) concerning the one
C   disallowed move.
C
```

```
14100          DIRECT = (DRAN(ISEED) * 6.0) + 1
               IF (DIRECT .EQ. 1) THEN
                   XDIR =  1
                   YDIR =  0
                   ZDIR =  0
                   DIRECT = 1
                 ELSEIF (DIRECT .EQ. 2) THEN
                   XDIR = -1
                   YDIR =  0
                   ZDIR =  0
                   DIRECT = 1
                 ELSEIF (DIRECT .EQ. 3) THEN
                   XDIR =  0
                   YDIR =  1
                   ZDIR =  0
                   DIRECT = 2
                 ELSEIF (DIRECT .EQ. 4) THEN
                   XDIR =  0
                   YDIR = -1
                   ZDIR =  0
                   DIRECT = 2
                 ELSEIF (DIRECT .EQ. 5) THEN
                   XDIR =  0
                   YDIR =  0
                   ZDIR =  1
                   DIRECT = 3
                 ELSE
                   XDIR =  0
                   YDIR =  0
                   ZDIR = -1
                   DIRECT = 3
               ENDIF
C
C    Set the coordinates for the moved monomer.
C
               XTMP = XPOSTP(ENDMV,J) + XDIR
               YTMP = YPOSTP(ENDMV,J) + YDIR
               ZTMP = ZPOSTP(ENDMV,J) + ZDIR
C
C    This next test avoids the problem stated in the initialization of
C  IF loop 20.  As the chain would never move back on itself, the test
C  disallows an end to move to a position taken by its neighbor on that
C  chain.  The argument still stands as to whether this biases the
C  system;  I, of course, believe that it does not as such a move is
C  inherently prohibited.
C
               IF (       (XTMP .EQ. XPOSTP(ENDMV-ARRSFT, J))
     &            .AND.   (YTMP .EQ. YPOSTP(ENDMV-ARRSFT, J))
     &            .AND.   (ZTMP .EQ. ZPOSTP(ENDMV-ARRSFT, J)) )
     &          GO TO 14100
C
C    See if the new site is "out-of-the-box."  That is, that the Z
C  coordinate has ventured past the limits of [-DMD2 <= Z <= DMD2].
```

```
C  One need not test for X or Y as periodic boundary conditions are
C  used.  The point of this test is to save some calculation time.
C
C  NOTE 1:  The test is different here than in single chain movement.
C           Here, one does not use one Monte Carlo step if a chain
C           moves out of the box.  In this case, the move is
C           disallowed and another chain is chosen to move.
C
C  NOTE 2:  The  GO TO 14000  jump here prevents vectorization of the
C           DO 14000 loop.  This is not critical as there are inner
C           loops which would be better to vectorize.  The IBM can
C           vectorize outer loops, however, depending upon the
C           circumstances.
C
C
                IF ( (ZTMP .LT. -DMD2)  .OR.  (ZTMP .GT. DMD2) )
     &            GO TO 14000
C
C       ***  Excluded Volume Test for the chain being moved.  ***
C
C    The new lattice position for the end of the monomer that is moved
C  is calculated.  If this position is taken, then there are two
C  possibilities:  the chain has chosen a position occupied by its other
C  end, which by reptation is a valid move, or the site is occupied by
C  some other monomer, and the move is invalid.
C
                XTMPA = MOD( (DM + MOD(XTMP+DMD2, DM)), DM )
     &                    - DMD2
                YTMPA = MOD( (DM + MOD(YTMP+DMD2, DM)), DM )
     &                    - DMD2
                ZTMPA = ZTMP
                IF (LATTMP(XTMPA,YTMPA,ZTMPA) .EQ. 1) THEN
                  IF (ENDMV .EQ. 1) THEN
                      I = MONOMR
                    ELSE
                      I = 1
                  ENDIF
C
C    XPOSTP(I,J), YPOSTP(I,J), and ZPOSTP(I,J)  represent the position
C  of the, as yet, unmoved end (tail) of the chain  J.
C
                  IF (       (XTMP .EQ. XPOSTP(I,J))
     &                .AND. (YTMP .EQ. YPOSTP(I,J))
     &                .AND. (ZTMP .EQ. ZPOSTP(I,J))  ) THEN
                      CONTINUE
                    ELSE
                      GO TO 14000
                  ENDIF
                ENDIF
C
C    Zero the old tail position for the chain being moved.
C
```

```
                IF (ENDMV .EQ. 1) THEN
                    XTMPA = MOD( (DM + MOD(XPOSTP(MONOMR,J)+DMD2, DM)),
     &                           DM ) - DMD2
                    YTMPA = MOD( (DM + MOD(YPOSTP(MONOMR,J)+DMD2, DM)),
     &                           DM ) - DMD2
                    ZTMPA = ZPOSTP(MONOMR,J)
                    LATTMP(XTMPA,YTMPA,ZTMPA) = 0
                  ELSE
                    XTMPA = MOD( (DM + MOD(XPOSTP(1,J)+DMD2, DM)), DM )
     &                      - DMD2
                    YTMPA = MOD( (DM + MOD(YPOSTP(1,J)+DMD2, DM)), DM )
     &                      - DMD2
                    ZTMPA = ZPOSTP(1,J)
                    LATTMP(XTMPA,YTMPA,ZTMPA) = 0
                ENDIF
C
C     Store the new positions and direction, and translate the others
C   from the previous configuration into temporary storage.  By using
C   the variables LOWCT, HIGHCT, and ARRSFT, the same code can be used
C   for the movement of either end of the chain.
C
C     Insert the new positions into the Direction and Position
C   Transfer arrays (TPD, TPX, TPY, TPZ).
C
                TPD(DIRTAG,J) = DIRECT
                TPX(ENDMV,J) = XTMP
                TPY(ENDMV,J) = YTMP
                TPZ(ENDMV,J) = ZTMP
                DO 14200 I = LOWCT, HIGHCT-1
                  TPD(I,J) = DIRTMP(I+ARRSFT, J)
14200           CONTINUE
                DO 14300 I = LOWCT, HIGHCT
                  TPX(I,J) = XPOSTP(I+ARRSFT, J)
                  TPY(I,J) = YPOSTP(I+ARRSFT, J)
                  TPZ(I,J) = ZPOSTP(I+ARRSFT, J)
14300           CONTINUE
C
C     Shift the new values back into the temporary Direction and Position
C   arrays (DIRTMP, XPOSTP, YPOSTP, ZPOSTP).
C
                DO 14400 I = 1, MONOMR-1
                  DIRTMP(I,J) = TPD(I,J)
14400           CONTINUE
                DO 14500 I = 1, MONOMR
                  XPOSTP(I,J) = TPX(I,J)
                  YPOSTP(I,J) = TPY(I,J)
                  ZPOSTP(I,J) = TPZ(I,J)
14500           CONTINUE
C
C     Flag the new head position for the chain being moved.
C
```

```
                IF (ENDMV .EQ. 1) THEN
                    XTMPA = MOD( (DM + MOD(XPOSTP(1,J)+DMD2, DM)), DM )
     &                       - DMD2
                    YTMPA = MOD( (DM + MOD(YPOSTP(1,J)+DMD2, DM)), DM )
     &                       - DMD2
                    ZTMPA = ZPOSTP(1,J)
                    LATTMP(XTMPA,YTMPA,ZTMPA) = 1
                  ELSE
                    XTMPA = MOD( (DM + MOD(XPOSTP(MONOMR,J)+DMD2, DM)),
     &                           DM ) - DMD2
                    YTMPA = MOD( (DM + MOD(YPOSTP(MONOMR,J)+DMD2, DM)),
     &                           DM ) - DMD2
                    ZTMPA = ZPOSTP(MONOMR,J)
                    LATTMP(XTMPA,YTMPA,ZTMPA) = 1
                ENDIF
C
C               ***  All the chains have been moved.  ***
C
14000         CONTINUE
C
C  ***  The Energy Test.  ***
C
C     NOTE:  The method of testing the energy of the system must incor-
C            porate two effects:  the energy due to bends, and the
C            repulsive energy.  The former is determined by counting
C            the number of bends in all of the chains in the system
C            (loop 15000).  The repulsive energy is determined by
C            counting the number of monomers presently on the top
C            layer (SRFNEW), and multiplying this by the temperature
C            factor (TAU) and the weighting of the repulsive energy
C            ( ln(LAMBDA) ).  The sum of these two energies is called,
C            here, the pseudo-Hamiltonian as the quantity differs
C            from the Hamiltonian only by the temperature factor.
C
C              Finally, the values of the old (HAMINI) and new (HAMNEW)
C            pseudo-Hamiltonians are compared.  If the new value is less
C            than the old value, the new configuration is immediately
C            accepted.  If the new value is greater, the Boltzmann
C            factor (BOLTZ) is calculated using the difference between
C            the pseudo-Hamiltonian energies.  (The Boltzmann factor
C            must be between 0 and 1 here, so that a direct test
C            against a random number returned from  DRAN  is possible.)
C
C         ==>  This test of the energies is the essence of Metropolis
C         ==>  sampling.
C
              BNDNEW = 0
              DO 15000 J = 1, CHAINS
                DO 15100 I = 1, MONOMR-2
                  IF (DIRTMP(I,J) .NE. DIRTMP(I+1,J))
     &              BNDNEW = BNDNEW+1
15100           CONTINUE
15000         CONTINUE
```

```
                DUMSM2 = 0.0D+00
                DO 16000 K = 1, DM2
                  DUMSM2 = DUMSM2 + LAT1TP(K + (DM-1)*DM2)
16000           CONTINUE
                SRFNEW = DUMSM2
                HAMNEW = DBLE(BNDNEW) - (SRFNEW * TAU * DLOG(LAMBDA))
                IF (HAMNEW .GT. HAMINI) THEN
C
C     The test is done on  DELHT  to avoid floating underflow problems
C   with the  DEXP  function.
C
                  DELHT = (HAMNEW - HAMINI) / TAU
                  IF (DELHT .GE. 170.0D+00) THEN
                      GO TO 10000
                    ELSE
                      BOLTZ = DEXP( - DELHT )
                      IF (BOLTZ .LT. DRAN(ISEED)) THEN
                        GO TO 10000
                      ENDIF
                  ENDIF
                ENDIF
C
C     ***  The new conformations are acceptable.  Transfer all   ***
C     ***   of the temporary values into the permanent arrays.   ***
C
                ACCEPT = ACCEPT + 1
                BNDINI = BNDNEW
                HAMINI = HAMNEW
                DO 17000 K = 1, MONXCH-CHAINS
                  DIR1(K) = DIR1TP(K)
17000           CONTINUE
                DO 18000 K = 1, MONXCH
                  XPOS1(K) = XPS1TP(K)
                  YPOS1(K) = YPS1TP(K)
                  ZPOS1(K) = ZPS1TP(K)
18000           CONTINUE
                DO 19000 K = 1, DM3
                  LAT1(K) = LAT1TP(K)
19000           CONTINUE
C
C     ***          This is the end of the  MCSTEP  loop.          ***
C
10000       CONTINUE
C
C           ***  This is the end of the  CONFGS  loop.  ***
C
20000     CONTINUE
          SUCCES = ACCEPT / DBLE(EQUILS*MCLIM)
          PRINT*, 'SUCCES in EQUIL = ', SUCCES
C
          RETURN
          END
C
C
```

```
      DOUBLE PRECISION FUNCTION DRAN(IY)
      INTEGER IY
C
C     DRAN is a uniform random number generator based on theory and
C  suggestions given in D. E. Knuth (1969), Vol. 2.  The integer IY
C  should be initialized to an arbitrary integer prior to the first
C  call to DRAN.  The calling program should not alter the value of IY
C  between subsequent calls to DRAN.  Values of DRAN will be returned
C  in the interval (0,1).
C
      INTEGER  IA, IC, ITWO, M2, M, MIC
      DOUBLE PRECISION  HALFM, S
      DOUBLE PRECISION  DATAN, DSQRT
      SAVE     IA, IC, M2, MIC, S
      DATA  M2/0/, ITWO/2/
      IF (M2 .NE. 0) GO TO 20
C
C  If first entry, compute machine integer word length.
C
      M = 1
   10 M2 = M
      M = ITWO * M2
      IF (M .GT. M2) GO TO 10
      HALFM = M2
C
C  Compute multiplier and increment for linear congruential method.
C
      IA = 8*IDINT(HALFM * DATAN(1.D0) / 8.D0) + 5
      IC = 2*IDINT(HALFM * (0.5D0 - DSQRT(3.D0)/6.D0)) + 1
      MIC = (M2 - IC) + M2
C
C  S is the scale factor for converting to floating point.
C
      S = 0.5 / HALFM
C
C  Compute next random number.
C
   20 IY = IY * IA
C
C  The following statement is for computers which do not allow
C  integer overflow upon addition.
C
      IF (IY .GT. MIC) IY = (IY - M2) - M2
C
      IY = IY + IC
C
C  The following statement is for computers where the word length
C  for addition is greater than for multiplication.
C
      IF (IY/2 .GT. M2) IY = (IY - M2) - M2
```

```
C
C  The following statement is for computers where integer
C  overflow affects the sign bit.
C
      IF (IY .LT. 0) IY = (IY + M2) + M2
C
      DRAN = DBLE(IY) * S
      RETURN
      END
```

APPENDIX B

CONTINUOUS SPACE, CANONICAL ENSEMBLE CODE

Representative Code
for the Continuous Space, Canonical Ensemble Simulations
As Executed on the Control Data Corporation ETA^10 Processors
at the John von Neumann Computing Center, Princeton, New Jersey

Includes the Radial Distribution Functions' Subroutine RADIAL
and the Scattering Function Subroutine SCATTR.

```
      PROGRAM RADDIS
C     Last modified by Scott K. Starry.
C
C     Parameters.
C     Change the parameters to the appropriate values before using.
C
      INTEGER      BEADS, CHAINS, EQSTEP, MCCNFG, MCLIM, MCUN, NSAMP,
     &              UNOBSR
      PARAMETER    (BEADS=11, CHAINS=735, EQSTEP=500,
     &              MCLIM=10, NSAMP=5000, UNOBSR=1000,
     &              MCUN=MCLIM*UNOBSR, MCCNFG=MCLIM*NSAMP+MCUN)
      REAL         DIAM, DIAM2, DIAMD2, DIAMD8, TAU
      PARAMETER    (DIAM=3.284512136E-02, DIAM2=DIAM**2,
     &              DIAMD2=DIAM/2.0, DIAMD8=DIAM/8.0,
     &              TAU=11.0)
C
C-----------------------------------------------------------------
C  NOTE:  LOCTN > 3*BDXCH ==> SORTPR = INT(LN(3*BDXCH)/LN(2)) + 1
C           The equation for  SORTPR  is necessary to keep the time
C           of execution to a minimum.
C
C         NRNEI  is the number of elements for  NB().  This
C                is checked in Subroutine NEIGH to see if NRNEI is
C                large enough.
C-----------------------------------------------------------------
C
      INTEGER      BDXCH, BDP1, BDX2, BDX3, BDX2P1, BM1XCH,
     &              CENBD, CRAYNM, LOCTN, LOCD2, NRNEI, SORTPR
      PARAMETER    (BDXCH=BEADS*CHAINS, BDP1=BDXCH+1, BDX2=2*BDXCH,
     &              BDX3=3*BDXCH, BDX2P1=2*BDXCH+1,
     &              BM1XCH=(BEADS-1)*CHAINS,
     &              CENBD=BEADS/2+1, CRAYNM=200,
     &              NRNEI=250000, SORTPR=15,
     &              LOCTN=2**SORTPR, LOCD2=LOCTN/2)
```

```
C
C-----------------------------------------------------------------------
C  NOTE:  BONDL < SQRT(2)*DIAM  by just a bit.
C         SQRT(2) = 1.4142 1356 2373 0950 4880 1688 72
C         RANGE  is a function of density, placement, and correctness;
C                be careful.
C-----------------------------------------------------------------------
C
      INTEGER       QMAX, QSIZE, QSTEP, SHELLS, SHELPR
      REAL          BONDL, BONDL2, BOXSZ, PI, PIX2,
     &               RANGE, RBON, RBON2, RMAX, RPIX2, RSLRD2,
     &               SHLRAD, SINCNV
      PARAMETER     (BONDL=1.414213562373000*DIAM, BONDL2=BONDL**2,
     &                BOXSZ=1.0, PI=3.14159265358979323846,
     &                PIX2=2.0*PI, QMAX=1.0/DIAM, QSTEP=1,
     &                QSIZE=(QMAX-6)/QSTEP+34,
     &                RANGE=3.0*BONDL, RPIX2=1.0/PIX2,
     &                SHLRAD=0.25*DIAM, SHELLS=0.87*BOXSZ/SHLRAD+1,
     &                SHELPR=0.50*BOXSZ/SHLRAD, SINCNV=5000.0/PI,
     &                RBON=1.0/BONDL, RBON2=1.0/BONDL2,
     &                RMAX=1.0E+307, RSLRD2 =(1.0/SHLRAD)**2)
C
C    Scalar variables.
C
      INTEGER       ACCEPT, ACCFLG, ARRSFT, ATMPTS, B, B2, BDCNT,
     &               C, C1, C1XB1, C2, C2M1XB, CM1XBD,
     &               CMCNT, CMCTMX, CNT, CONFGS,
     &               END, ENDMV, ENDAV, ENGCNT, FILNUM, FLAG,
     &               HIGHCT, HIAV, HIIL
      INTEGER       I, INAUGH, IRX, IRY, IRZ, ISEED, J, K,
     &               LOWCT, LOWAV, LOWIL, M, MCSTEP, NONEI,
     &               PENULT, PENAV, Q, SAMPL, SYSCHG,
     &               TAIL, TMPINT, TPWR, TPWRMX, VOLCNT
C
      REAL          RLAMB, DM
      REAL          ALPHA, ANGAVG, ANGDIF, ANGLSM,
     &               BDDEN, BDDIS2, BDNORM, BDVOL,
     &               BDX, BDY, BDZ,
     &               BDXM1, BDYM1, BDZM1, BDXP1, BDYP1, BDZP1,
     &               BNDAVS, BNDINI, BNDNEW, BNDS2, BOLTZ
      REAL          CHDEN, CMAVG, CMDIS, CMRDFN,
     &               CMX, CMY, CMZ, CMXM1, CMYM1, CMZM1,
     &               CMXP1, CMYP1, CMZP1, CNRNRM, COTHE, DELHT,
     &               DUMSM, DUMSM1, DUMSM2, DUMSM3,
     &               ENDNRM, GAMMA,
     &               HAMINI, HAMLTN, HAMNEW, HAMS2, HAMVAR, MIDNRM
      REAL          PRESS1, PRESS2, PRESS3,
     &               RGY2, RGYAVG, RJM, RNN2, RNNAVG,
     &               RX, RY, RZ, RXNEW, RYNEW, RZNEW,
     &               SHLVOL, SITHE, SRFBDS, STIFF, SUCCES,
     &               TAUTMP, TMPDP, TMPDP1
      REAL          XBOX, YBOX, ZBOX, XTEMP, YTEMP, ZTEMP,
     &               XTMP, YTMP, ZTMP
```

```
C
C     Array variables.
C
      INTEGER       BDPERC(CHAINS), BDOTR(0:SHELLS), CMSHEL(0:SHELLS),
     &               IL(LOCTN),
     &               JA1(BDXCH), JA2(BDXCH), JA3(BDXCH),
     &               LOCX(LOCTN)
      INTEGER       NB(NRNEI), ORDCHN(CHAINS), RNDCHN(CHAINS)
C
      REAL          ANG1(BM1XCH), ANG1PV(BM1XCH), ANG1TP(BM1XCH),
     &               ANGMV(CHAINS), ANGMV2(CHAINS, NSAMP),
     &               BDCEN(0:SHELLS), BDEND(0:SHELLS), BDMID(0:SHELLS),
     &               BDRDFC(0:SHELLS), BDRDFE(0:SHELLS),
     &               BDRDFF(0:SHELLS), BDRDFM(0:SHELLS)
      REAL          BDRDFO(0:SHELLS), BDRDFS(0:SHELLS),
     &               BDSLFG(0:SHELLS), BDV2(CHAINS),
     &               CM1(3*CHAINS), CM1PV(3*CHAINS), CM1TP(3*CHAINS),
     &               CMDIST(CHAINS), CMDIS2(CHAINS, NSAMP)
      REAL          CMRDFA(0:SHELLS),
     &               CX(LOCTN), CX1(BDX3), CX2(BDX3), CXSORT(LOCTN),
     &               FULPR(QSIZE), FULSCT(QSIZE),
     &               OTRPR(QSIZE), OTRSCT(QSIZE),
     &               POS1(3*BDXCH), POS1TP(3*BDXCH), PS1TP(3*BDXCH),
     &               RGYRA2(NSAMP), RJM2(BDXCH), RNTON2(NSAMP),
     &               SINTAB(0:9999), SLFPR(QSIZE), SLFSCT(QSIZE)
      REAL          TPA(BM1XCH),
     &               TPVX(BM1XCH), TPVY(BM1XCH), TPVZ(BM1XCH),
     &               TPX(BDXCH), TPY(BDXCH), TPZ(BDXCH),
     &               VEC1(3*BM1XCH), VEC1TP(3*BM1XCH),
     &               WAVEIN(33), WAVENM(QSIZE)
C
      INTEGER       IRG(BDX3), IRGY(BDXCH), IRGZ(BDXCH)
      EQUIVALENCE   (IRG(BDP1), IRGY(1)), (IRG(BDX2P1), IRGZ(1))
      INTEGER       KJB(BDX3), JBS(BDXCH), KBS(BDXCH), MBS(BDXCH)
      EQUIVALENCE   (KJB(1), JBS(1)), (KJB(BDP1), KBS(1)),
     &               (KJB(BDX2P1), MBS(1))
      INTEGER       KJE(BDX3), JES(BDXCH), KES(BDXCH), MES(BDXCH)
      EQUIVALENCE   (KJE(1), JES(1)), (KJE(BDP1), KES(1)),
     &               (KJE(BDX2P1), MES(1))
C
      REAL          X1(BDXCH), Y1(BDXCH), Z1(BDXCH)
      EQUIVALENCE   (X1(1), POS1(1)), (Y1(1), POS1(BDXCH+1)),
     &               (Z1(1), POS1(2*BDXCH+1))
      REAL          X1TP(BDXCH), Y1TP(BDXCH), Z1TP(BDXCH)
      EQUIVALENCE   (X1TP(1), POS1TP(1)), (Y1TP(1), POS1TP(BDXCH+1)),
     &               (Z1TP(1), POS1TP(2*BDXCH+1))
      REAL          XCM(CHAINS), YCM(CHAINS), ZCM(CHAINS)
      EQUIVALENCE   (XCM(1), CM1(1)), (YCM(1), CM1(CHAINS+1)),
     &               (ZCM(1), CM1(2*CHAINS+1))
      REAL          XCMTP(CHAINS), YCMTP(CHAINS), ZCMTP(CHAINS)
      EQUIVALENCE   (XCMTP(1), CM1TP(1)), (YCMTP(1), CM1TP(CHAINS+1)),
     &               (ZCMTP(1), CM1TP(2*CHAINS+1))
```

```
      REAL            XVEC1(BM1XCH), YVEC1(BM1XCH), ZVEC1(BM1XCH)
      EQUIVALENCE     (XVEC1(1), VEC1(1)), (YVEC1(1), VEC1(BM1XCH+1)),
     &                  (ZVEC1(1), VEC1(2*BM1XCH+1))
      REAL            XVC1TP(BM1XCH), YVC1TP(BM1XCH), ZVC1TP(BM1XCH)
      EQUIVALENCE     (XVC1TP(1), VEC1TP(1)),
     &                  (YVC1TP(1), VEC1TP(BM1XCH+1)),
     &                  (ZVC1TP(1), VEC1TP(2*BM1XCH+1))
C
CC    Declarations of functions.
C
      EXTERNAL        SCOPY, SECOND
      REAL            ELSPTL, ELSPT1, ELSPT2, SECOND, TIMSUM
C
CC    Initial ISEED.
C
      DATA            FILNUM/10/
      DATA            ISEED/717889529/
      DATA            ATMPTS, ENGCNT, NONEI, VOLCNT/0, 0, 0, 0/
      DATA            TIMSUM/0.0/
      DATA            OTRSCT, SLFSCT/QSIZE*0.0, QSIZE*0.0/
      DATA            WAVEIN/ 3.0,  6.0,  9.0, 11.0, 12.0,
     &                       14.0, 17.0, 18.0, 19.0, 21.0,
     &                       22.0, 24.0, 26.0, 27.0, 29.0,
     &                       30.0, 33.0, 34.0, 35.0, 36.0,
     &                       38.0, 41.0, 42.0, 43.0, 45.0,
     &                       48.0, 50.0, 51.0, 54.0, 57.0,
     &                       59.0, 66.0, 75.0/
C
C     Main program.
C
C     Check to be sure that the number of CHAINS and BEADS will
C   not crash the program.
C
        IF (CHAINS .LT. 2) THEN
          STOP ' There must be at least 2 chains in the system.'
        ENDIF
        IF (BEADS .LT. 5) THEN
          STOP ' There must be at least 5 beads per chain.'
        ENDIF
C
C     Make sure that the chains have an odd number of beads.
C
        IF (MOD(BEADS,2) .EQ. 0) THEN
          STOP ' The number of beads per chain must be an odd number.'
        ENDIF
C
C     Make sure that there is enough storage allocated for the
C   Nearest Neighbor Table's sorting procedure.
C
        IF (SORTPR .NE. (INT(LOG(FLOAT(BDX3)) / LOG(2.0)) + 1))
     &    STOP ' SORTPR has been incorrectly calculated.'
C
```

```
C-------------------------------------------------------------------------01B
C                            Initializations.
C-------------------------------------------------------------------------
C
C    Test to be sure that the length of the chains does not influence
C  the periodic boundary conditions.
C
      IF (((BEADS - 1)*BONDL + DIAM) .GE. 0.50) THEN
        STOP ' The chains are longer than the cut-off radius.'
      ENDIF
C
C    Save the initial value of the  ISEED  to be sure the same
C  simulation is not repeated.
C
      INAUGH = ISEED
      CALL RANSET(ISEED)
C
C    Compute the volume excluded by the beads,
C  BDDEN, and the volume excluded by the chains, CHDEN.
C  (The last depends on BONDL = SQRT(2)*DIAM.)
C
      BDVOL = PI * (DIAM**3) / 6.0
      BDDEN = (BDXCH * BDVOL) / BOXSZ
      CHDEN = (CHAINS * BDVOL
     &          * ( (SQRT(2.0) * (16.0 - 3.0*PI) / 8.0)
     &              *(BEADS - 1) + 1.0 )) / BOXSZ
      DO 1000 CNT = 0, 9999
        SINTAB(CNT) = SIN(FLOAT(CNT) / SINCNV)
1000  CONTINUE
C
C    Set the value for  TAUTMP  equal to a large value to start, and
C  then lower it to the desired value of  TAU  by an order of magnitude
C  in each iteration of Loop 10000.
C
C  NOTE:  The reduction of  TAUTMP  by an order of magnitude each
C         iteration may be too much to equilibrate the system
C         efficiently.
C
C  NOTE:  By the test below, at least three iterations of the loop
C         occur for any value of  TAU  even if it is large.  Since the
C         initialization is by self-avoiding walk, the initial tempera-
C         ture of the system is effectively infinite.
C
      TPWRMX = NINT(LOG(10.0 / TAU) / LOG(2.0))
      IF (TPWRMX .LT. 2) TPWRMX = 2
C
C    Initalize the system.
C
      TAUTMP = TAU * (2.0**(TPWRMX+1))
      ELSPT1 = SECOND()
```

```
      CALL STRAD (BDXCH, BEADS, BM1XCH, CHAINS,
     &                BONDL, DIAM, DIAM2, RBON2,
     &                ISEED,
     &                HAMINI, TAUTMP,
     &                BDPERC,
     &                ANG1, X1, Y1, Z1, XCM, YCM, ZCM,
     &                 XVEC1, YVEC1, ZVEC1)
      ELSPT2 = SECOND()
      PRINT*, 'Time for STREP (secs.)  ', ELSPT2-ELSPT1
C
C     Equilibrate the system to the desired value of  TAU.
C
      DO 10000 TPWR = TPWRMX, 0, -1
        HAMINI = 2.0 * HAMINI
        TAUTMP = TAU * (2.0**TPWR)
        ELSPT1 = SECOND()
        CALL EQRAD (BDP1, BDX2, BDX3, BDX2P1, BDXCH, BEADS, BM1XCH,
     &                  CHAINS, CRAYNM, EQSTEP,
     &                  LOCD2, LOCTN, MCLIM, NRNEI, SORTPR,
     &                 BONDL, BOXSZ, DIAM, DIAM2, RANGE, RBON2, RMAX,
     &                 ISEED,
     &                 HAMINI, TAUTMP,
     &                 IL, IRG, IRGY, IRGZ, JA1, JA2, JA3, JBS, JES,
     &                  KBS, KES, KJB, KJE, LOCX, MBS, MES, NB,
     &                  ORDCHN, RNDCHN,
     &                 ANG1, ANG1TP, CM1, CM1TP,
     &                  CX, CX1, CX2, CXSORT,
     &                 POS1, POS1TP,
     &                 TPA, TPVX, TPVY, TPVZ, TPX, TPY, TPZ,
     &                 VEC1, VEC1TP,
     &                 X1TP, Y1TP, Z1TP, XCMTP, YCMTP, ZCMTP,
     &                 XVC1TP, YVC1TP, ZVC1TP)
C
        ELSPT2 = SECOND()
        PRINT*, 'Time for EQRAD (secs.)  ', ELSPT2-ELSPT1
10000   CONTINUE
C
C-----------------------------------------------------------------01E
C
C-----------------------------------------------------------------02B
C                        Run the system.
C-----------------------------------------------------------------
C
      ELSPTL = SECOND()
      DO 21000 CNT = 0, SHELLS
        CMSHEL(CNT) = 0
        BDOTR(CNT) = 0
        BDCEN(CNT)  = 0.0
        BDEND(CNT)  = 0.0
        BDMID(CNT)  = 0.0
21000   CONTINUE
      DO 22000 Q = 1, 33
        WAVENM(Q) = PIX2 * SQRT(WAVEIN(Q))
22000   CONTINUE
```

```
          Q = 33
          DO 23000 CNT = 6, QMAX, QSTEP
            Q = Q + 1
            WAVENM(Q) = PIX2 * SQRT(3.0) * CNT
23000     CONTINUE
C
          CALL SCOPY(3*CHAINS, CM1, 1, CM1PV, 1)
          CALL SCOPY(BM1XCH, ANG1, 1, ANG1PV, 1)
          DO 40000 CONFGS = 1, NSAMP+UNOBSR
            ACCEPT = 0
            ACCFLG = 0
C           CALL NEIGH(BDP1, BDX2, BDX3, BDX2P1, BDXCH, CRAYNM,
C     &                 LOCD2, LOCTN, NRNEI, SORTPR,
C     &                BOXSZ, DIAM, RANGE, RMAX,
C     &                ISEED,
C     &                IL, IRG, IRGY, IRGZ, JA1, JA2, JA3, JBS, JES,
C     &                KBS, KES, KJB, KJE, LOCX, MBS, MES, NB,
C     &                CX, CX1, CX2, CXSORT, POS1)
C
            DO 30000 MCSTEP = 1, MCLIM
C
CC    Transfer the information to the working arrays.
C
              IF (ACCFLG .EQ. 0) THEN
                CALL SCOPY(BM1XCH, ANG1, 1, ANG1TP, 1)
                CALL SCOPY(3*CHAINS, CM1, 1, CM1TP, 1)
                CALL SCOPY(3*BDXCH, POS1, 1, POS1TP, 1)
                CALL SCOPY(3*BM1XCH, VEC1, 1, VEC1TP, 1)
              ENDIF
              ACCFLG = 0
              DO 31000 C1 = 1, CHAINS
                ATMPTS = ATMPTS + 1
                C = CHAINS*RANF() + 1
                CM1XBD = (C - 1) * BEADS
                C1XB1  = (C - 1) * (BEADS - 1)
C
CC    Begin the reptation of the chosen chain, C.
C
                ENDMV = 2*RANF() + 1
C
CC    For ENDMV=1, the vector  A  is  -VEC(2,C)  and the vector  R
C  is  -VEC(1,C).  For  ENDMV=BEADS, vector  A  is  VEC(BEADS-2,C)  and
C  vector  R  is  VEC(BEADS-1,C).
C
                IF (ENDMV .EQ. 1) THEN
                    ENDMV = 1
                    END = CM1XBD + 1
                    PENULT = END + 1
                    TAIL = CM1XBD + BEADS
                    LOWCT  = PENULT
                    HIGHCT = TAIL
                    ENDAV = C1XB1 + 1
                    PENAV = ENDAV + 1
                    LOWAV = PENAV
```

```
                  HIAV  = C1XB1 + (BEADS - 1)
                  ARRSFT = -1
C
                  RX = -XVC1TP(ENDAV)
                  RY = -YVC1TP(ENDAV)
                  RZ = -ZVC1TP(ENDAV)
                ELSE
                  ENDMV = BEADS
                  END = CM1XBD + BEADS
                  PENULT = END - 1
                  TAIL  = CM1XBD + 1
                  LOWCT  = TAIL
                  HIGHCT = PENULT
                  ENDAV = C1XB1 + (BEADS - 1)
                  PENAV = ENDAV
                  LOWAV = C1XB1 + 1
                  HIAV  = ENDAV - 1
                  ARRSFT = 1
C
                  RX = XVC1TP(PENAV)
                  RY = YVC1TP(PENAV)
                  RZ = ZVC1TP(PENAV)
              ENDIF
C
C     Compute the cosine and sine of the rotation angles GAMMA and
C   THETA.  GAMMA varies from -PI to PI.  THETA  varies from 0 to PI.
C
              GAMMA = PIX2 * (RANF() - 0.50)
              COTHE = 2.0*RANF() - 1.0
              SITHE = SQRT(1.0 - COTHE**2)
C
C     Calculate the new co-ordinates for the end bead.  This amounts
C   to applying a vector decomposition as if the vector formed by
C   the present positions of the penultimate and end beads lies along
C   the laboratory Z-axis.  This really has no bearing on the result
C   as long as the new vector has a length equal to  BONDL  which
C   is assured by this method.
C
              XTEMP = X1TP(END) + BONDL*SITHE*COS(GAMMA)
              YTEMP = Y1TP(END) + BONDL*SITHE*SIN(GAMMA)
              ZTEMP = Z1TP(END) + BONDL*COTHE
C
C     Check to be sure the new polymer position satisfies
C   the excluded volume condition.
C
C   NOTE:  ***     This is the brute force (OLD) method.     ***
C          ***  Incorporate the neighbor table method ASAP.  ***
C
C             FLAG = 1
C             DO 31100 C2 = 1, CHAINS
C               IF (C2 .NE. C) THEN
```

```
C                   DO 31110 B2 = 1, BEADS
C                     XTMP = ABS(XTEMP - X1TP((C2-1)*BEADS + B2))
C                     YTMP = ABS(YTEMP - Y1TP((C2-1)*BEADS + B2))
C                     ZTMP = ABS(ZTEMP - Z1TP((C2-1)*BEADS + B2))
C                     IRX  = XTMP + 0.50
C                     IRY  = YTMP + 0.50
C                     IRZ  = ZTMP + 0.50
C                     BDDIS2 =    (XTMP - IRX)**2
C     &                        + (YTMP - IRY)**2 + (ZTMP - IRZ)**2
C                     IF (BDDIS2 .LT. DIAM2) FLAG = FLAG + 2
C31110              CONTINUE
C                   ELSE
C                     DO 31120 B2 = 2, BEADS-1
C                       XTMP = ABS(XTEMP - X1TP(CM1XBD + B2))
C                       YTMP = ABS(YTEMP - Y1TP(CM1XBD + B2))
C                       ZTMP = ABS(ZTEMP - Z1TP(CM1XBD + B2))
C                       IRX  = XTMP + 0.50
C                       IRY  = YTMP + 0.50
C                       IRZ  = ZTMP + 0.50
C                       BDDIS2 =  (XTMP - IRX)**2
C     &                          + (YTMP - IRY)**2 + (ZTMP - IRZ)**2
C                       IF (BDDIS2 .LT. DIAM2) FLAG = FLAG + 2
C31120                CONTINUE
C                   ENDIF
C                   IF (FLAG .GT. 1) THEN
C                     VOLCNT = VOLCNT + 1
C                     GO TO 31000
C                   ENDIF
C31100            CONTINUE
C
C    Check to be sure the new polymer position satisfies
C  the excluded volume condition.
C
C  NOTE:  ***     This is the brute force (NEW) method.     ***
C         ***  Incorporate the neighbor table method ASAP.  ***
C
C    This is the traditional method optimized for the Convex C-210.
C
                FLAG = 1
                DO 31100 B2 = 2, BEADS-1
                  XTMP = ABS(XTEMP - X1TP(CM1XBD + B2))
                  YTMP = ABS(YTEMP - Y1TP(CM1XBD + B2))
                  ZTMP = ABS(ZTEMP - Z1TP(CM1XBD + B2))
                  IRX  = XTMP + 0.50
                  IRY  = YTMP + 0.50
                  IRZ  = ZTMP + 0.50
                  BDDIS2 =    (XTMP - IRX)**2
     &                       + (YTMP - IRY)**2  + (ZTMP - IRZ)**2
                  IF (BDDIS2 .LT. DIAM2) FLAG = FLAG + 2
31100           CONTINUE
                IF (FLAG .GT. 1) THEN
                  VOLCNT = VOLCNT + 1
                  GO TO 31000
                ENDIF
```

```
C
                DO 31200 B2 = 1, BEADS
                  DO 31210 C2 = 1, C-1
                    XTMP = ABS(XTEMP - X1TP((C2 - 1)*BEADS + B2))
                    YTMP = ABS(YTEMP - Y1TP((C2 - 1)*BEADS + B2))
                    ZTMP = ABS(ZTEMP - Z1TP((C2 - 1)*BEADS + B2))
                    IRX  = XTMP + 0.50
                    IRY  = YTMP + 0.50
                    IRZ  = ZTMP + 0.50
                    BDV2(C2) =    (XTMP - IRX)**2
     &                         + (YTMP - IRY)**2  + (ZTMP - IRZ)**2
31210               CONTINUE
                  DO 31215 C2 = 1, C-1
                    IF (BDV2(C2) .LT. DIAM2) FLAG = FLAG + 2
31215             CONTINUE
                  IF (FLAG .GT. 1) THEN
                    VOLCNT = VOLCNT + 1
                    GO TO 31000
                  ENDIF
                  DO 31220 C2 = C+1, CHAINS
                    XTMP = ABS(XTEMP - X1TP((C2 - 1)*BEADS + B2))
                    YTMP = ABS(YTEMP - Y1TP((C2 - 1)*BEADS + B2))
                    ZTMP = ABS(ZTEMP - Z1TP((C2 - 1)*BEADS + B2))
                    IRX  = XTMP + 0.50
                    IRY  = YTMP + 0.50
                    IRZ  = ZTMP + 0.50
                    BDV2(C2) =    (XTMP - IRX)**2
     &                         + (YTMP - IRY)**2  + (ZTMP - IRZ)**2
31220               CONTINUE
                  DO 31225 C2 = C+1, CHAINS
                    IF (BDV2(C2) .LT. DIAM2) FLAG = FLAG + 2
31225             CONTINUE
                  IF (FLAG .GT. 1) THEN
                    VOLCNT = VOLCNT + 1
                    GO TO 31000
                  ENDIF
31200           CONTINUE
C
C   This is the Neighbor Table Method optimized for the CONVEX C-210.
C
C                 FLAG = 1.0
C                 B2 = CM1XBD + ENDMV
C                 LOWIL = IL(B2)
C                 HIIL = IL(B2+1)
C
C    Test to be sure that the bead has neighbors.
C
C  NOTE:  This test should be unnecessary since chains are present.
C         Remove it once one is sure that the process is working.
C
C                 IF (LOWIL .EQ. HIIL) THEN
C                   NONEI = NONEI + 1
C                   GO TO 31300
C                 ENDIF
```

```
C               DO 31100 I = LOWIL, HIIL-1
C                 XTMP = XTEMP - X1TP(NB(I))
C                 YTMP = YTEMP - Y1TP(NB(I))
C                 ZTMP = ZTEMP - Z1TP(NB(I))
C                 XBOX = XTMP - DNINT(XTMP)
C                 YBOX = YTMP - DNINT(YTMP)
C                 ZBOX = ZTMP - DNINT(ZTMP)
C                 BDDIS2 = XBOX**2 + YBOX**2 + ZBOX**2
C                 IF (BDDIS2 .LE. DIAM2) FLAG = FLAG + 1.0
C31100           CONTINUE
C               IF (IDINT(FLAG) .NE. 1) THEN
C                 VOLCNT = VOLCNT + 1
C                 GO TO 31000
C               ENDIF
C
C   The excluded volume test is successful.  Set  XTP(B,C), YTP(B,C),
C  ZTP(B,C)  equal to  XTEMP, YTEMP, ZTEMP, respectively.
C
C    Calculate the new end vector and angle.
C
31300           RXNEW = XTEMP - X1TP(END)
                RYNEW = YTEMP - Y1TP(END)
                RZNEW = ZTEMP - Z1TP(END)
C
C    Compute the new vectors VEC(1,C)  or  VEC(BEADS-1,C),
C  as necessary, and the new center of mass for the chain.
C
                XCMTP(C) = XCMTP(C)
     &                    + (XTEMP - X1TP(TAIL)) / FLOAT(BEADS)
                YCMTP(C) = YCMTP(C)
     &                    + (YTEMP - Y1TP(TAIL)) / FLOAT(BEADS)
                ZCMTP(C) = ZCMTP(C)
     &                    + (ZTEMP - Z1TP(TAIL)) / FLOAT(BEADS)
              IF (ENDMV .EQ. 1) THEN
                TPVX(ENDAV) = -RXNEW
                TPVY(ENDAV) = -RYNEW
                TPVZ(ENDAV) = -RZNEW
              ELSE
                TPVX(PENAV) = RXNEW
                TPVY(PENAV) = RYNEW
                TPVZ(PENAV) = RZNEW
              ENDIF
              DO 31400 CNT = LOWAV, HIAV
                TPVX(CNT) = XVC1TP(CNT+ARRSFT)
                TPVY(CNT) = YVC1TP(CNT+ARRSFT)
                TPVZ(CNT) = ZVC1TP(CNT+ARRSFT)
31400         CONTINUE
Ccvd$ noeqvchk
              DO 31450 CNT = C1XB1+1, C1XB1+BEADS-1
                XVC1TP(CNT) = TPVX(CNT)
                YVC1TP(CNT) = TPVY(CNT)
                ZVC1TP(CNT) = TPVZ(CNT)
31450         CONTINUE
```

```
C
C  Calculate the angles for the new system.
C
               TPA(C1XB1+1) = 0.0
               TPA(PENAV) = - RBON2 * (RXNEW*RX + RYNEW*RY + RZNEW*RZ)
               DO 31500 CNT = LOWAV+1, HIAV
                 TPA(CNT) = ANG1TP(CNT+ARRSFT)
31500          CONTINUE
               DO 31550 CNT = C1XB1+1, C1XB1+BEADS-1
                 ANG1TP(CNT) = TPA(CNT)
31550          CONTINUE
C
C  Calculate the positions for the new system.
C
               TPX(END) = XTEMP
               TPY(END) = YTEMP
               TPZ(END) = ZTEMP
               DO 31600 CNT = LOWCT, HIGHCT
                 TPX(CNT) = X1TP(CNT+ARRSFT)
                 TPY(CNT) = Y1TP(CNT+ARRSFT)
                 TPZ(CNT) = Z1TP(CNT+ARRSFT)
31600          CONTINUE
Ccvd$ noeqvchk
               DO 31650 CNT = CM1XBD+1, CM1XBD+BEADS
                 X1TP(CNT) = TPX(CNT)
                 Y1TP(CNT) = TPY(CNT)
                 Z1TP(CNT) = TPZ(CNT)
31650          CONTINUE
C
C  ***   End  CHAINS:  All of the chains have been moved.  ***
C
31000        CONTINUE
C
C    Metropolis sampling.
C
C  NOTE:  Check the new  ESSLLIBS  for a routine to do this on the IBM.
C         If one adds 1.0 to the elements of ANG1TP() as they are
C         calculated, then one could use the DASUM function.  (The
C         computation of BNDNEW would also need to be adjusted appropri-
C         ately.)
C
             ANGLSM=0.0
             DO 33000 CNT = 1, BM1XCH
               ANGLSM = ANGLSM + ANG1TP(CNT)
33000        CONTINUE
             BNDNEW = (0.50 / TAU) * ((BEADS-2)*CHAINS + ANGLSM)
C
```

```
                HAMNEW = BNDNEW
                IF (HAMNEW .GT. HAMINI) THEN
                  DELHT = (HAMNEW - HAMINI)
                  IF (DELHT .GE. 170.0) THEN
                      ENGCNT = ENGCNT + 1
                      GO TO 30000
                    ELSE
                      BOLTZ = EXP(- DELHT)
                      IF (BOLTZ .LT. RANF()) THEN
                        ENGCNT = ENGCNT + 1
                        GO TO 30000
                      ENDIF
                  ENDIF
                ENDIF
C
CC    New positions are acceptable.  Place in permanent storage.
C
                ACCEPT = ACCEPT + 1
                ACCFLG = 1
                BNDINI = BNDNEW
                HAMINI = HAMNEW
C
CC    Place the new positions into the permanent arrays.
C
                CALL SCOPY(BM1XCH, ANG1TP, 1, ANG1, 1)
                CALL SCOPY(3*CHAINS, CM1TP, 1, CM1, 1)
                CALL SCOPY(3*BDXCH, POS1TP, 1, POS1, 1)
                CALL SCOPY(3*BM1XCH, VEC1TP, 1, VEC1, 1)
C
CC    ***          This is the end of the  MCSTEP  loop.          ***
C
30000      CONTINUE
C
C  ***  The Measurement Section.  ***
C
C     The system is given  UNOBSR  steps to come to equilibrium, or at
C  least to shed the initial character of the system, before statistics
C  are accumulated.  The number of samples is  NSAMP, and  SAMPL
C  serves as the counter.
C
              IF (CONFGS .GT. UNOBSR) THEN
                SUCCES = SUCCES + ACCEPT
                SAMPL = CONFGS - UNOBSR
                BNDAVS = BNDAVS + BNDINI
                BNDS2 = BNDS2 + BNDINI**2
                HAMLTN = HAMLTN + HAMINI
                HAMS2 = HAMS2 + HAMINI**2
```

```
                RNN2 = 0.0
                RGY2 = 0.0
                DO 41000 C = 1, CHAINS
                  CM1XBD = (C - 1)*BEADS
                  I = CM1XBD + BEADS
                  J = CM1XBD + 1
                  RNN2 = RNN2 + (X1(I) - X1(J))**2
     &                        + (Y1(I) - Y1(J))**2
     &                        + (Z1(I) - Z1(J))**2
41000             CONTINUE
                DO 42000 C = 1, CHAINS
                  CM1XBD = (C - 1)*BEADS
                  DO 42100 B = CM1XBD+1, CM1XBD+BEADS
                    RGY2 = RGY2 + (X1(B) - XCM(C))**2
     &                          + (Y1(B) - YCM(C))**2
     &                          + (Z1(B) - ZCM(C))**2
42100             CONTINUE
42000           CONTINUE
                RNTON2(SAMPL) = RNN2 / FLOAT(CHAINS)
                RGYRA2(SAMPL) = RGY2 / FLOAT(BDXCH)
C
C    Check the overall center of mass motion and internal motion
C  of the chains.
C
                DO 43000 CNT = 1, CHAINS
                  CMDIS =    (CM1PV(CNT) - CM1(CNT))**2
     &                   + (CM1PV(CNT + CHAINS)
     &                      - CM1(CNT + CHAINS))**2
     &                   + (CM1PV(CNT + 2*CHAINS)
     &                      - CM1(CNT + 2*CHAINS))**2
                  CMDIS2(CNT,SAMPL) = CMDIS
43000           CONTINUE
                CALL SCOPY(3*CHAINS, CM1, 1, CM1PV, 1)
C
                DO 44000 CNT = 1, CHAINS
                  ANGDIF = 0.0
                  C1XB1 = (CNT - 1) * (BEADS - 1)
                  DO 44100 B = C1XB1+2, C1XB1+(BEADS-1)
                    ANGDIF = ANGDIF
     &                     + ABS(ACOS(ANG1PV(B)) - ACOS(ANG1(B)))
44100             CONTINUE
                  ANGMV2(CNT,SAMPL) = ANGDIF
44000           CONTINUE
                CALL SCOPY(BM1XCH, ANG1, 1, ANG1PV, 1)
C
C    Collect data for the radial distribution functions.
C
                CALL RADIAL(BDXCH, BEADS, CENBD, CHAINS, SHELLS,
     &                      RSLRD2,
     &                      BDOTR, CMSHEL,
     &                      BDCEN, BDEND, BDMID, CM1, X1, Y1, Z1)
C
C    Collect data for the scattering function.
C
```

```
            ELSPT1 = SECOND()
            CALL SCATTR(BDXCH, BEADS, CHAINS, QSIZE,
     &                  PIX2, RPIX2, SINCNV,
     &                  OTRSCT, RJM2, SINTAB, SLFSCT, WAVENM,
     &                  POS1, PS1TP)
            TIMSUM = TIMSUM + SECOND() - ELSPT1
C
            IF (MOD(CONFGS,250) .EQ. 0) THEN
              OPEN(UNIT=FILNUM)
              DO 9501 Q = 1, QSIZE
                OTRPR(Q) = (FLOAT(CHAINS) * OTRSCT(Q)
     &                      / FLOAT(BDXCH)**2 / FLOAT(CONFGS-UNOBSR))
                SLFPR(Q) = (2.0*SLFSCT(Q)
     &                      / FLOAT(BDXCH)**2 / FLOAT(CONFGS-UNOBSR))
     &                     + 1.0/FLOAT(BDXCH)
                FULPR(Q) = OTRPR(Q) + SLFPR(Q)
 9501         CONTINUE
              WRITE(FILNUM,9510)
              DO 9502 Q = 1, QSIZE
                WRITE(FILNUM,9600) WAVENM(Q), SLFPR(Q),
     &                             OTRPR(Q), FULPR(Q)
 9502         CONTINUE
              CLOSE(UNIT=FILNUM)
              FILNUM = FILNUM + 1
            ENDIF
C
          ENDIF
C
C         *** This is the end of the CONFGS loop. ***
C
40000   CONTINUE
C
C    The average pseudo-Hamiltonian, (HAMLTN), the average pseudo-
C  Hamiltonian squared (HAMS2), and the average end-to-end distance
C  of the polymer chains (RNTON) over NSAMP tries are calculated.
C
        BNDAVS = BNDAVS / FLOAT(NSAMP)
        BNDS2 = BNDS2 / FLOAT(NSAMP)
        HAMLTN = HAMLTN / FLOAT(NSAMP)
        HAMS2 = HAMS2 / FLOAT(NSAMP)
        DUMSM1 = 0.0
        DUMSM2 = 0.0
        DO 51000 CNT = 1, NSAMP
          DUMSM1 = DUMSM1 + SQRT(RNTON2(CNT))
          DUMSM2 = DUMSM2 + SQRT(RGYRA2(CNT))
51000   CONTINUE
        RNNAVG = DUMSM1 / FLOAT(NSAMP)
        RGYAVG = DUMSM2 / FLOAT(NSAMP)
        DO 52000 C = 1, CHAINS
          DUMSM = 0.0
          DUMSM1 = 0.0
```

```
          DO 52100 CNT = 1, NSAMP
            DUMSM  = DUMSM  + SQRT(CMDIS2(C,CNT))
            DUMSM1 = DUMSM1 + ANGMV2(C,CNT)
52100     CONTINUE
          CMDIST(C) = DUMSM
          ANGMV(C)  = DUMSM1 / FLOAT(NSAMP*(BEADS - 2)) * (180.0 / PI)
52000   CONTINUE
        DUMSM1 = 0.0
        DUMSM2 = 0.0
        DO 53000 CNT = 1, CHAINS
          DUMSM1 = DUMSM1 + CMDIST(CNT)
          DUMSM2 = DUMSM2 + ANGMV(CNT)
53000   CONTINUE
        CMAVG = DUMSM1 / FLOAT(CHAINS)
        ANGAVG = DUMSM2 / FLOAT(CHAINS)
        PRINT*, 'NONEI  ', NONEI
        PRINT*, 'CMAVG, ANGAVG  ', CMAVG, ' ', ANGAVG
        PRINT*, ' '
        WRITE(6,*) 'CMDIST(CHAINS)'
        DO 53500 J = 1, CHAINS, 3
53500     WRITE(6,*) '  ', (CMDIST(I), ' ', I=J,J+2)
        PRINT*, ' '
        WRITE(6,*) 'ANGMV(CHAINS)'
        DO 53600 J = 1, CHAINS, 3
53600     WRITE(6,*) '  ', (ANGMV(I), ' ', I=J,J+2)
        PRINT*, ' '
        TMPDP = 1.0 - FLOAT(VOLCNT + ENGCNT*CHAINS)/ATMPTS
        PRINT*, 'General Success  ', TMPDP
        PRINT*, 'Time for SCATTR  ', TIMSUM
C
C    The averages and standard deviations for the Center of Masses' RDF.
C
        CMRDFN  = 0.0
        DO 54000 I = 1, CHAINS-1
          CMRDFN = CMRDFN + I
54000   CONTINUE
        BDNORM = FLOAT(BEADS * BEADS * (CHAINS - 1))
        CNRNRM = FLOAT(BEADS - 1)
        ENDNRM = FLOAT(2 * (BEADS - 1))
        MIDNRM = 0.0
        DO 54500 I = 1, BEADS-2
          MIDNRM = MIDNRM + (2*I - 1)
54500   CONTINUE
        MIDNRM = MIDNRM + (BEADS - 2)
C
```

```
      TMPDP = 4.0 * PI * SHLRAD**3
      DO 55000 CMCNT = 0, SHELPR
        SHLVOL = TMPDP * (CMCNT**2 + CMCNT + (1.0/3.0))
        CMRDFA(CMCNT) = FLOAT(CMSHEL(CMCNT)) / CMRDFN
     &                  / FLOAT(NSAMP) / SHLVOL
        BDRDFO(CMCNT) = FLOAT(BDOTR(CMCNT)) / BDNORM
     &                  / FLOAT(NSAMP) / SHLVOL
        BDSLFG(CMCNT) = (BDEND(CMCNT) + BDMID(CMCNT)) / BDNORM
     &                  / FLOAT(CHAINS) / FLOAT(NSAMP) / SHLVOL
55000 CONTINUE
      BDCNT = BONDL / SHLRAD
      BDCEN(BDCNT) = BDCEN(BDCNT) + 2*CHAINS*NSAMP
      BDEND(BDCNT) = BDEND(BDCNT) + 2*CHAINS*NSAMP
      BDMID(BDCNT) = BDMID(BDCNT) + 2*CHAINS*NSAMP*(BEADS - 2)
      DO 56000 CMCNT = 0, SHELPR
        SHLVOL = TMPDP * (CMCNT**2 + CMCNT + (1.0/3.0))
        BDRDFC(CMCNT) = BDCEN(CMCNT) / CNRNRM
     &                  / FLOAT(CHAINS) / FLOAT(NSAMP) / SHLVOL
        BDRDFE(CMCNT) = BDEND(CMCNT) / ENDNRM
     &                  / FLOAT(CHAINS) / FLOAT(NSAMP) / SHLVOL
        BDRDFM(CMCNT) = BDMID(CMCNT) / MIDNRM
     &                  / FLOAT(CHAINS) / FLOAT(NSAMP) / SHLVOL
        BDRDFS(CMCNT) = (BDEND(CMCNT) + BDMID(CMCNT)) / BDNORM
     &                  / FLOAT(CHAINS) / FLOAT(NSAMP) / SHLVOL
56000 CONTINUE
      DO 56500 CMCNT = 0, SHELPR
        BDRDFF(CMCNT) = BDRDFO(CMCNT) + BDRDFS(CMCNT)
56500 CONTINUE
      DO 57000 Q = 1, QSIZE
        OTRSCT(Q) = (FLOAT(CHAINS) * OTRSCT(Q)
     &              / FLOAT(BDXCH)**2 / FLOAT(NSAMP))
        SLFSCT(Q) = (2.0 * SLFSCT(Q)
     &               / FLOAT(BDXCH)**2 / FLOAT(NSAMP))
     &              + 1.0/FLOAT(BDXCH)
57000 CONTINUE
      DO 58000 Q = 1, QSIZE
        FULSCT(Q) = OTRSCT(Q) + SLFSCT(Q)
58000 CONTINUE
C
C    Averages for the pseudo-Hamiltonian (HAMLTN), bulk density (PHIB),
C  radius of gyration (RGYAVG), and end-to-end distances (RNNAVG) for
C  the simulation are calculated.
C
      HAMVAR = HAMS2 - HAMLTN**2
      STIFF  = LOG(RNNAVG**2 * RBON2) / LOG(FLOAT(BEADS - 1))
C
C    The next calculation gives the fraction of attempts to change
C  the PERMANENT configuration of the system that were successful.
C  This value (SUCCES) only depends on the results of the energy test.
C
C  NOTE:  The denominator is obtained from the maximum values for
C         the counters of the loops involved:
C
C                              NSAMP * MCLIM.
```

```
C
      SUCCES = SUCCES / FLOAT(NSAMP * MCLIM)
C
C           ****************  DONE  ****************
C-----------------------------------------------------------------03E
C
C*****************************************************************
C                       THE SECOND PRINTING SECTION.
C*****************************************************************
C
C     OPEN (UNIT=50)
      WRITE(50,8000)
8000  FORMAT (/, /, ' ***  This is the information for the final ',
     &          'configuration of the system.  ***', /)
      WRITE(50,8510)
8510  FORMAT(' ---  Results  ---',/)
      SYSCHG = NSAMP * SUCCES
      WRITE (50,8520)
8520  FORMAT(6X, 'BDDEN', 7X, 'SYSCHG', 6X, 'RNNAVG', 10X, 'RGYAVG',
     &       10X, 'STIFF')
      WRITE (50, 8530) BDDEN, SYSCHG, RNNAVG, RGYAVG, STIFF
8530  FORMAT(1X, 2PE13.5, 3X, I6, 3X, 2PE13.5, 3X, 2PE13.5,
     &       3X, 1PE13.6, /)
      WRITE (50, 8550)
8550  FORMAT(5X, 'HAMLTN', 10X, 'PRESS1', 10X, 'PRESS2',
     &       10X, 'PRESS3')
      WRITE (50, 8555) HAMLTN
8555  FORMAT(1X, 1PE13.6, /)
      WRITE(50,8600)
8600  FORMAT(' ---  System Parameters  ---', /)
      DM = BOXSZ / BONDL
      WRITE (50,8610)
8610  FORMAT (2X, 'CHAINS', 4X, 'BEADS', 6X, 'DM', 11X, 'TAU',
     &        14X, 'INAUGH')
      WRITE (50,8620) CHAINS, BEADS, DM, TAU, INAUGH
8620  FORMAT (1X, I5, 6X, I4, 6X, F6.2, 7X, 1PE13.6, 5X, I10, /)
      WRITE(50,8700)
8700  FORMAT(' ---  Statistics For This Execution  ---', /)
      WRITE (50, 8710)
8710  FORMAT (5X, 'SUCCES', 9X, 'NSAMP', 6X, 'UNOBSR', 5X, 'MCLIM',
     &        7X, 'EQSTEP')
      WRITE (50, 8720) SUCCES, NSAMP, UNOBSR, MCLIM, EQSTEP
8720  FORMAT (1X, 2PD13.6, 5X, I6, 5X, I6, 5X, I6, 5X, I6, /)
      WRITE (50, 8721)
8721  FORMAT (6X, 'ATMPTS', 7X, 'ENGCNT', 7X, 'VOLCNT')
      WRITE (50, 8725) ATMPTS, ENGCNT, VOLCNT
8725  FORMAT (1X, 3( 3X, I10), /)
      WRITE(50,8800)
8800  FORMAT(' ---  Checks and Raw Information  ---', /)
      WRITE(50,8810)
8810  FORMAT(' LAMBDA', 7X, 'CHDEN', 9X, 'HAMLTN', 10X,
     &       'HAMVAR', 10X, 'RNNAVG', 10X, 'RGYAVG', 10X,
     &       'PRESS1', 7X, 'Avg Num on H', /)
       RLAMB = 1.00
```

```
            WRITE(50,8830) RLAMB, CHDEN, HAMLTN,
     &         HAMVAR, RNNAVG, RGYAVG
8830      FORMAT(2X, F4.2, 1X, 5(3X, 1PE13.6))
          WRITE(50,8900)
8900      FORMAT(/)
C
C
          CLOSE (UNIT=50)
          OPEN (UNIT=55)
          OPEN (UNIT=56)
          OPEN (UNIT=57)
          OPEN (UNIT=58)
          OPEN (UNIT=59)
          OPEN (UNIT=60)
          OPEN (UNIT=62)
          OPEN (UNIT=65)
          WRITE(55,9000)
          WRITE(56,9010)
          WRITE(57,9020)
          WRITE(58,9030)
          WRITE(59,9040)
          WRITE(60,9050)
          WRITE(62,9060)
          WRITE(65,9070)
          DO 9100 CMCNT = 0, SHELPR
            TMPDP = CMCNT * SHLRAD
            WRITE(55,9200) CMCNT, TMPDP, CMRDFA(CMCNT)
            WRITE(56,9200) CMCNT, TMPDP, BDRDFO(CMCNT)
            WRITE(57,9200) CMCNT, TMPDP, BDRDFE(CMCNT)
            WRITE(58,9200) CMCNT, TMPDP, BDRDFM(CMCNT)
            WRITE(59,9200) CMCNT, TMPDP, BDRDFS(CMCNT)
            WRITE(60,9200) CMCNT, TMPDP, BDRDFC(CMCNT)
            WRITE(62,9200) CMCNT, TMPDP, BDSLFG(CMCNT)
            WRITE(65,9200) CMCNT, TMPDP, BDRDFF(CMCNT)
9100      CONTINUE
9000      FORMAT('  Label of Shell', 7x, 'Boundary', 9x,
     &          'RDF of Chains')
9010      FORMAT('  Label of Shell', 7x, 'Boundary', 9x,
     &          'RDF of Others')
9020      FORMAT('  Label of Shell', 7x, 'Boundary', 11x,
     &          'RDF of Ends')
9030      FORMAT('  Label of Shell', 7x, 'Boundary', 11x,
     &          'RDF of Mids')
9040      FORMAT('  Label of Shell', 7x, 'Boundary', 11x,
     &          'RDF of Self')
9050      FORMAT('  Label of Shell', 7x, 'Boundary', 10x,
     &          'RDF of Center')
9060      FORMAT('  Label of Shell', 7x, 'Boundary', 8x,
     &          'Self for Graph')
9070      FORMAT('  Label of shell', 7x, 'Boundary', 12x,
     &          ' Full RDF')
9200      FORMAT(5X, I6, 10X, 1PE12.5, 8X, 1PE12.5)
          CLOSE(UNIT=55)
          CLOSE(UNIT=56)
```

```
      CLOSE(UNIT=57)
      CLOSE(UNIT=58)
      CLOSE(UNIT=59)
      CLOSE(UNIT=60)
      CLOSE(UNIT=62)
C
      OPEN(UNIT=40)
      WRITE(40,9510)
      DO 9500 Q = 1, QSIZE
        WRITE(40,9600) WAVENM(Q), SLFSCT(Q), OTRSCT(Q), FULSCT(Q)
 9500 CONTINUE
 9510 FORMAT(2X, 'Wave Number', 4X, 'Self Scat. Fcn.',
     &          5X, 'Others Scat. Fcn.', 5X, 'Full Scat. Fcn.')
 9600 FORMAT(1X, 1PE12.5, 5X, 1PE12.5, 9X, 1PE12.5, 9X, 1PE12.5)
      CLOSE(UNIT=40)
C
C
C**********************************************************************
C                        END SECOND PRINTING SECTION
C**********************************************************************
C
      ELSPT2 = SECOND()
      PRINT*, 'Timing for the running of the system:  ', ELSPT2-ELSPTL
      STOP ' The program RADDIS is not yet complete.'
      END
C
C
C
      SUBROUTINE STRAD (BDXCH, BEADS, BM1XCH, CHAINS,
     &                  BONDL, DIAM, DIAM2, RBON2,
     &                  ISEED,
     &                  HAMINI, TAUTMP,
     &                  BDPERC,
     &                  ANG1, X1, Y1, Z1, XCM, YCM, ZCM,
     &                   XVEC1, YVEC1, ZVEC1)
C
C  Last modified 05/08/89.  Scott K. Starry
C
C    Program to generate the initial configuration of a set of
C  CHAINS  linear polymers, each consisting of BEADS particles.  The
C  method is a simple self-avoiding walk mechanism.
C
C    Parameters passed.
C
      INTEGER   BDXCH, BEADS, BM1XCH, CHAINS
      REAL      BONDL, DIAM, DIAM2, RBON2
C
C    Scalar variables passed.
C
      INTEGER   ISEED
      REAL      HAMINI, TAUTMP
```

```
C
C     Array variables passed.
C
          INTEGER   BDPERC(CHAINS)
          REAL      ANG1(BM1XCH),
     &              X1(BDXCH), Y1(BDXCH), Z1(BDXCH),
     &              XCM(CHAINS), YCM(CHAINS), ZCM(CHAINS),
     &              XVEC1(BM1XCH), YVEC1(BM1XCH), ZVEC1(BM1XCH)
C
C     Local scalar variables.
C
          INTEGER   B, B1, C, C1, C1M1XB, C1XB1, CM1XBD, CNT, I, J
          REAL      ANGLSM, BDVOL, BNDNEW, CHNVOL, CSTHET, DIST2, DUMSM,
     &              INIL, PHI, PI, RINIL, SNTHET, SRFBDS,
     &              XCOFM, YCOFM, ZCOFM, XTMP, YTMP, ZTMP
C
C     Compute the value of PI.
C
          PI = ACOS(-1.0)
C
C     Generate the initial beads of each polymer.  This is done in a
C   REDUCED cubic volume of sides  INIL=L/4, where  L  is the actual
C   length.  After the points are generated in this fashion, each
C   coordinate is multiplied by four to fill the actual volume.  This
C   procedure is designed to prevent the initial beads from being too
C   close together.
C
          INIL = (1.0 / 2.0)
          RINIL = 1.0 / INIL
C
C     Test to see if the beads can be placed with the present initial
C   size for the box and if the whole system can fit in the final box.
C   (The latter test presumes that the initial BOXSZ is 1.0 and depends
C   on  BONDL=SQRT(2)*DIAM.)  The value of 0.70 is near the hexagonal
C   close packed limit.
C
          BDVOL = (PI / 6.0) * DIAM**3
          CHNVOL = BDVOL *
     &           ( (SQRT(2.0) * (16.0 - 3.0*PI) / 8.0)
     &             *(BEADS - 1) + 1.0 )
          IF ( (BDVOL * CHAINS * RINIL**3) .GT. 0.70 )
     &    STOP ' The initial box is not large enough to place the beads.'
          IF ( (CHNVOL * CHAINS) .GT. 0.70 )
     &    STOP ' The box is not large enough for all of the chains.'
C
          DO 1000 C = 1, CHAINS
            I = (C - 1)*BEADS + 1
C
 1010       XTMP = INIL * ((RANF() - 0.50) - DIAM)
            YTMP = INIL * ((RANF() - 0.50) - DIAM)
            ZTMP = INIL * ((RANF() - 0.50) - DIAM)
C
C     Perform the "excluded volume test" at this stage by checking
C   to make sure that the last bead position generated is not within
```

```
C DIAM  of any other bead generated previously.
C
          DO 1100 C1 = 1, C-1
            J = (C1 - 1)*BEADS + 1
            DIST2 =  (XTMP - X1(J))**2 + (YTMP - Y1(J))**2
     &             + (ZTMP - Z1(J))**2
            IF (DIST2 .LE. DIAM2) GO TO 1010
 1100       CONTINUE
            X1(I) = XTMP
            Y1(I) = YTMP
            Z1(I) = ZTMP
 1000     CONTINUE
C
C    The initial beads of each polymer have now been generated
C  in the reduced volume.  The configuration is now expanded so that it
C  fills the actual volume.
C
          DO 2000 C = 1, CHAINS
            I = (C - 1)*BEADS + 1
            X1(I) = RINIL * X1(I)
            Y1(I) = RINIL * Y1(I)
            Z1(I) = RINIL * Z1(I)
 2000     CONTINUE
C
C    Next, the second bead of each polymer is generated.  The second
C  bead is added to each polymer before proceeding to the next step.
C
          DO 3000 C = 1, CHAINS
            I = (C - 1)*BEADS + 1
C
C    Randomly choose the orientation of the second bead relative
C  to the first.  The next two statements return  PHI  and the cosine
C  of THETA, CSTHET.  (THETA  is the azimuthal angle; PHI  is the
C  polar angle.)
C
 3010       PHI = 2.0 * PI * RANF()
            CSTHET = 2.0*RANF() - 1.0
            SNTHET = SQRT(1.0 - CSTHET**2)
C
C    Compute the coordinates for the second bead in chain C, and
C  store these results in  XTMP, YTMP, ZTMP.
C
            XTMP = X1(I) + BONDL*SNTHET*SIN(PHI)
            YTMP = Y1(I) + BONDL*SNTHET*COS(PHI)
            ZTMP = Z1(I) + BONDL*CSTHET
C
C    Perform the excluded volume test, comparing bead (2,C)
C  to the positions of all other beads generated so far.
C
C  NOTE:  No test needs to be performed between bead 1 and bead 2 of
C         chain C here as the second bead was just laid down with
C         respect to the first bead.
C
```

```
C  NOTE:  There should not be any trouble laying down the second
C         bead due to the expansion of the volume.  Thus, returning
C         to Label 3010 without a trap should be sufficient in the
C         unlikely event of a failure of the excluded volume test.
C
          DO 3100 C1 = 1, C-1
            C1M1XB = (C1 - 1)*BEADS
            DO 3110 B = 1, 2
              J = C1M1XB + B
              DIST2 =  (XTMP - X1(J) - ANINT(XTMP - X1(J)))**2
     &               + (YTMP - Y1(J) - ANINT(YTMP - Y1(J)))**2
     &               + (ZTMP - Z1(J) - ANINT(ZTMP - Z1(J)))**2
              IF (DIST2 .LE. DIAM2) GO TO 3010
 3110       CONTINUE
 3100     CONTINUE
C
C     If the excluded volume test is passed, then accept the
C  generated values of the second bead in polymer C.
C
          X1(I+1) = XTMP
          Y1(I+1) = YTMP
          Z1(I+1) = ZTMP
 3000   CONTINUE
C
C     At this point, all of the chains have two beads, and the values
C  of BDPERC(C) must all be set to 2.
C
          DO 4000 C = 1, CHAINS
            BDPERC(C) = 2
 4000     CONTINUE
C
C     Next, further beads are added to each chain.  This is done until
C  all of the beads for a single chain have been placed.  Then, the
C  beads for the next chain are laid down.  Since not all chains have
C  the same length, the array  BDPERC  is used to handle the excluded
C  volume tests.
C
          DO 5000 C = 1, CHAINS
            CM1XBD = (C - 1)*BEADS
            DO 5100 B = 3, BEADS
              I = CM1XBD + B
C
C     Attempt to propagate a new bead.
C
C     Generate the coordinates of the new bead.
C
 5010         PHI = 2.0 * PI * RANF()
              CSTHET = 2.0*RANF() - 1.0
              SNTHET = SQRT(1.0 - CSTHET**2)
C
              XTMP = X1(I-1) + BONDL*SNTHET*COS(PHI)
              YTMP = Y1(I-1) + BONDL*SNTHET*SIN(PHI)
              ZTMP = Z1(I-1) + BONDL*CSTHET
```

```
C
C    Perform the excluded volume test for the new bead.
C
C  NOTE:  In this test, even beads within the chain  C  must
C         be tested.
C
            DO 5110 C1 = 1, CHAINS
              C1M1XB = (C1 - 1)*BEADS
              DO 5111 B1 = 1, BDPERC(C1)
                J = C1M1XB + B1
                IF ( .NOT. ((C1 .EQ. C) .AND. (B1 .EQ. B)) ) THEN
                  DIST2 =   (XTMP - X1(J) - ANINT(XTMP - X1(J)))**2
     &                  + (YTMP - Y1(J) - ANINT(YTMP - Y1(J)))**2
     &                  + (ZTMP - Z1(J) - ANINT(ZTMP - Z1(J)))**2
C
C   If the excluded volume test fails, find a new position for
C  the bead being placed.
C
C  NOTE:  There is presently no trap here for chains that are caught
C         in a tight spot.  This is because Ho believes that in
C         three dimensions there is always space to place the next
C         monomer.  This must be tested at high concentrations
C         to see if the time for creating the initial configuration
C         is acceptable.
C
                  IF (DIST2 .LE. DIAM2) GO TO 5010
                ENDIF
 5111         CONTINUE
 5110       CONTINUE
C
C    If the excluded volume test is passed, store the new bead
C  coordinates in X, Y, and Z.
C
            X1(I) = XTMP
            Y1(I) = YTMP
            Z1(I) = ZTMP
            BDPERC(C) = BDPERC(C) + 1
C
 5100     CONTINUE
 5000   CONTINUE
C
C    Loops 6000 to 8000 calculate the bond vectors, centers of mass,
C  and bond angles for the chains.
C
Ccvd$ nodepchk
        DO 6000 C = 1, CHAINS
          C1XB1 = (C - 1)*(BEADS - 1)
          CM1XBD = (C - 1)*BEADS
          DO 6100 B = 1, BEADS-1
            XVEC1(C1XB1+B) = X1(CM1XBD+B+1) - X1(CM1XBD+B)
            YVEC1(C1XB1+B) = Y1(CM1XBD+B+1) - Y1(CM1XBD+B)
            ZVEC1(C1XB1+B) = Z1(CM1XBD+B+1) - Z1(CM1XBD+B)
 6100     CONTINUE
 6000   CONTINUE
```

```
C
        DO 7000 C = 1, CHAINS
          CM1XBD = (C - 1)*BEADS
          XCOFM = 0.0
          YCOFM = 0.0
          ZCOFM = 0.0
          DO 7100 B = CM1XBD+1, CM1XBD+BEADS
            XCOFM = XCOFM + X1(B)
            YCOFM = YCOFM + Y1(B)
            ZCOFM = ZCOFM + Z1(B)
 7100     CONTINUE
          XCM(C) = XCOFM / FLOAT(BEADS)
          YCM(C) = YCOFM / FLOAT(BEADS)
          ZCM(C) = ZCOFM / FLOAT(BEADS)
 7000   CONTINUE
C
Ccvd$ nodepchk
        DO 8000 C = 1, CHAINS
          C1XB1  = (C - 1)*(BEADS - 1)
          ANG1(C1XB1+1) = 0.0
          DO 8100 B = C1XB1+2, C1XB1+(BEADS-1)
            ANG1(B) = - RBON2 * (    XVEC1(B-1) * XVEC1(B)
     &                           +  YVEC1(B-1) * YVEC1(B)
     &                           +  ZVEC1(B-1) * ZVEC1(B) )
 8100     CONTINUE
 8000   CONTINUE
C
C    Calculate the energy of the bends in the system.
C
        ANGLSM = 0.0
        DO 10500 CNT = 1, BM1XCH
          ANGLSM = ANGLSM + ANG1(CNT)
10500   CONTINUE
        BNDNEW = (0.50 / TAUTMP) * ((BEADS-2)*CHAINS + ANGLSM)
C
        HAMINI = BNDNEW
        PRINT*, 'STREP has exited.'
        RETURN
        END
C
C
C
```

```
      SUBROUTINE EQRAD (BDP1, BDX2, BDX3, BDX2P1, BDXCH,
     &                  BEADS, BM1XCH, CHAINS, CRAYNM, EQSTEP,
     &                  LOCD2, LOCTN, MCLIM, NRNEI, SORTPR,
     &                 BONDL, BOXSZ, DIAM, DIAM2,
     &                  RANGE, RBON2, RMAX,
     &                 ISEED,
     &                 HAMINI, TAUTMP,
     &                 IL, IRG, IRGY, IRGZ, JA1, JA2, JA3,
     &                  JBS, JES, KBS, KES, KJB, KJE,
     &                  LOCX, MBS, MES, NB, ORDCHN, RNDCHN,
     &                 ANG1, ANG1TP, CM1, CM1TP,
     &                  CX, CX1, CX2, CXSORT,
     &                  POS1, POS1TP,
     &                  TPA, TPVX, TPVY, TPVZ, TPX, TPY, TPZ,
     &                  VEC1, VEC1TP,
     &                  X1TP, Y1TP, Z1TP, XCMTP, YCMTP, ZCMTP,
     &                  XVC1TP, YVC1TP, ZVC1TP)
C
C     Last modified 11/12/88.  Scott K. Starry
C
C
C
C    Parameters passed.
C
      INTEGER   BDP1, BDX2, BDX3, BDX2P1, BDXCH, BEADS, BM1XCH,
     &           CHAINS, CRAYNM, EQSTEP,
     &           LOCD2, LOCTN, MCLIM, NRNEI, SORTPR
      REAL      BONDL, BOXSZ, DIAM, DIAM2, RANGE, RBON2, RMAX
C
C    Scalar variables passed.
C
      INTEGER   ISEED
      REAL      HAMINI, TAUTMP
C
C    Array variables passed.
C
      INTEGER   IL(LOCTN), IRG(BDX3), IRGY(BDXCH), IRGZ(BDXCH),
     &           JA1(BDXCH), JA2(BDXCH), JA3(BDXCH),
     &           JBS(BDXCH), JES(BDXCH), KBS(BDXCH), KES(BDXCH)
      INTEGER   KJB(BDX3), KJE(BDX3), LOCX(LOCTN),
     &           MBS(BDXCH), MES(BDXCH), NB(NRNEI),
     &           ORDCHN(CHAINS), RNDCHN(CHAINS)
      REAL      ANG1(BM1XCH), ANG1TP(BM1XCH),
     &           CM1(3*CHAINS), CM1TP(3*CHAINS),
     &           CX(LOCTN), CX1(BDX3), CX2(BDX3), CXSORT(LOCTN),
     &           POS1(3*BDXCH), POS1TP(3*BDXCH)
      REAL      TPA(BM1XCH), TPVX(BM1XCH), TPVY(BM1XCH),
     &           TPVZ(BM1XCH), TPX(BDXCH), TPY(BDXCH),
     &           TPZ(BDXCH), VEC1(3*BM1XCH), VEC1TP(3*BM1XCH)
      REAL      X1TP(BDXCH), Y1TP(BDXCH), Z1TP(BDXCH),
     &           XCMTP(CHAINS), YCMTP(CHAINS), ZCMTP(CHAINS),
     &           XVC1TP(BM1XCH), YVC1TP(BM1XCH), ZVC1TP(BM1XCH)
C
C    Local scalar variables.
C
```

```
      INTEGER   ACCEPT, ACCFLG, ACCMOV, ARRSFT, ATMPTS,
     &           B2, C, C1, C1XB1, C2, C2M1XB, CM1XBD, CNT,
     &           END, ENDMV, ENDAV, ENGCNT, EQUIL, FLAG
      INTEGER   HIGHCT, HIAV, HIIL, I, IRX, IRY, IRZ,
     &           LOWCT, LOWAV, LOWIL, MCSTEP, NONEI,
     &           PENULT, PENAV, VOLCNT
      REAL      ANGLSM, BDDIS2, BNDNEW, BNDINI, BOLTZ, COTHE,
     &           DELHT, DUMSM, GAMMA, HAMNEW, MOVTOT, MOVED,
     &           PI, RX, RY, RZ, RXNEW, RYNEW, RZNEW
      REAL      SITHE, SRFBDS, SUCCES, SUCMOV,
     &           XBOX, YBOX, ZBOX,
     &           XTEMP, YTEMP, ZTEMP, XTMP, YTMP, ZTMP
C
      PI = ACOS(-1.0)
      ATMPTS = 0
      ENGCNT = 0
      VOLCNT = 0
      MOVTOT = 0.00
      SUCCES = 0.00
      SUCMOV = 0.00
C       CALL NEIGH(BDP1, BDX2, BDX3, BDX2P1, BDXCH, CRAYNM,
C    &              LOCD2, LOCTN, NRNEI, SORTPR,
C    &             BOXSZ, DIAM, RANGE, RMAX,
C    &             ISEED,
C    &             IL, IRG, IRGY, IRGZ, JA1, JA2, JA3, JBS, JES,
C    &             KBS, KES, KJB, KJE, LOCX, MBS, MES, NB,
C    &             CX, CX1, CX2, CXSORT, POS1)
      DO 40000 EQUIL = 1, EQSTEP
        ACCEPT = 0
        ACCFLG = 0
        ACCMOV = 0
        DO 30000 MCSTEP = 1, MCLIM
C
C    Transfer the information to the working arrays.
C
          IF (ACCFLG .EQ. 0) THEN
            CALL SCOPY(BM1XCH, ANG1, 1, ANG1TP, 1)
            CALL SCOPY(3*CHAINS, CM1, 1, CM1TP, 1)
            CALL SCOPY(3*BDXCH, POS1, 1, POS1TP, 1)
            CALL SCOPY(3*BM1XCH, VEC1, 1, VEC1TP, 1)
C         CALL NEIGH(BDP1, BDX2, BDX3, BDX2P1, BDXCH, CRAYNM,
C    &              LOCD2, LOCTN, NRNEI, SORTPR,
C    &             BOXSZ, DIAM, RANGE, RMAX,
C    &             ISEED,
C    &             IL, IRG, IRGY, IRGZ, JA1, JA2, JA3, JBS, JES,
C    &             KBS, KES, KJB, KJE, LOCX, MBS, MES, NB,
C    &             CX, CX1, CX2, CXSORT, POS1TP)
          ENDIF
          ACCFLG = 0
C
C    The loops 30100, 30200, 30210 are used to mix the order in
C  which the chains are chosen to be moved.  This method ensures
C  that a chain will only be attemped to be moved once in Loop
```

```
C  31000 before the neighbor table is updated and that the label
C  on the chain will not influence the results.
C
            DO 30100 C = 1, CHAINS
              ORDCHN(C) = C
30100       CONTINUE
            DO 30200 C1 = 1, CHAINS
              C = (CHAINS + 1 - C1)*RANF() + 1
              RNDCHN(C1) = ORDCHN(C)
              DO 30210 CNT = C, CHAINS-1
                ORDCHN(CNT) = ORDCHN(CNT+1)
30210         CONTINUE
30200       CONTINUE
            DO 31000 C1 = 1, CHAINS
              ATMPTS = ATMPTS + 1
              C = RNDCHN(C1)
              CM1XBD = (C - 1) * BEADS
              C1XB1  = (C - 1) * (BEADS - 1)
              ENDMV = 2*RANF() + 1
C
C    For ENDMV=1, the vector  A  is  -VEC(2,C)  and the vector  R
C  is  -VEC(1,C).  For  ENDMV=BEADS, vector  A  is  VEC(BEADS-2,C)  and
C  vector  R  is  VEC(BEADS-1,C).
C
              IF (ENDMV .EQ. 1) THEN
                  ENDMV = 1
                  END = CM1XBD + 1
                  PENULT = END + 1
                  LOWCT  = END + 1
                  HIGHCT = CM1XBD + BEADS
                  ENDAV = C1XB1 + 1
                  PENAV = ENDAV + 1
                  LOWAV = PENAV
                  HIAV  = C1XB1 + (BEADS - 1)
                  ARRSFT = -1
C
                  RX = -XVC1TP(ENDAV)
                  RY = -YVC1TP(ENDAV)
                  RZ = -ZVC1TP(ENDAV)
              ELSE
                  ENDMV = BEADS
                  END = CM1XBD + BEADS
                  PENULT = END - 1
                  LOWCT  = CM1XBD + 1
                  HIGHCT = END - 1
                  ENDAV = C1XB1 + (BEADS - 1)
                  PENAV = ENDAV
                  LOWAV = C1XB1 + 1
                  HIAV  = ENDAV - 1
                  ARRSFT = 1
C
```

```
                RX = XVC1TP(PENAV)
                RY = YVC1TP(PENAV)
                RZ = ZVC1TP(PENAV)
              ENDIF

C
C   Compute the cosine and sine of the rotation angles GAMMA and
C THETA.  GAMMA  varies from -PI to PI.  THETA  varies from 0 to PI.
C
              GAMMA = PI * (2.0*RANF() - 1.0)
              COTHE = 2.0*RANF() - 1.0
              SITHE = SQRT(1.0 - COTHE**2)
C
C   Calculate the new co-ordinates for the bead.  This amounts
C to applying a vector decomposition as if the vector formed by
C the present positions of the penultimate and end beads lies along
C the laboratory Z-axis.  This really has no bearing on the result
C as long as the new vector has a length equal to  BONDL  which is
C assured by these equations.
C
              XTEMP = X1TP(END) + BONDL*SITHE*COS(GAMMA)
              YTEMP = Y1TP(END) + BONDL*SITHE*SIN(GAMMA)
              ZTEMP = Z1TP(END) + BONDL*COTHE
C
C   Check to be sure the new polymer position satisfies
C the excluded volume condition.
C
C  NOTE:  ***        This is the brute force method.        ***
C         ***  Incorporate the neighbor table method ASAP.  ***
C
C             FLAG = 1
C             DO 31100 C2 = 1, CHAINS
C               C2M1XB = (C2 - 1)*BEADS
C               IF (C2 .NE. C) THEN
C                   DO 31110 B2 = 1, BEADS
C                     XTMP = ABS(XTEMP - X1TP(C2M1XB + B2))
C                     YTMP = ABS(YTEMP - Y1TP(C2M1XB + B2))
C                     ZTMP = ABS(ZTEMP - Z1TP(C2M1XB + B2))
C                     IRX  = XTMP + 0.50
C                     IRY  = YTMP + 0.50
C                     IRZ  = ZTMP + 0.50
C                     BDDIS2 =  (XTMP - IRX)**2
C    &                          + (YTMP - IRY)**2 + (ZTMP - IRZ)**2
C                     IF (BDDIS2 .LT. DIAM2) FLAG = FLAG + 2
C31110               CONTINUE
C                 ELSE
C                   DO 31120 B2 = 2, BEADS-1
C                     XTMP = ABS(XTEMP - X1TP(C2M1XB + B2))
C                     YTMP = ABS(YTEMP - Y1TP(C2M1XB + B2))
C                     ZTMP = ABS(ZTEMP - Z1TP(C2M1XB + B2))
C                     IRX  = XTMP + 0.50
C                     IRY  = YTMP + 0.50
C                     IRZ  = ZTMP + 0.50
```

```
C                       BDDIS2 =  (XTMP - IRX)**2
C     &                          + (YTMP - IRY)**2 + (ZTMP - IRZ)**2
C                       IF (BDDIS2 .LT. DIAM2) FLAG = FLAG + 2
C31120                 CONTINUE
C                    ENDIF
C                    IF (FLAG .GT. 1) THEN
C                      VOLCNT = VOLCNT + 1
C                      GO TO 31000
C                    ENDIF
C31100              CONTINUE
C
C     Check to be sure the new polymer position satisfies
C  the excluded volume condition.
C
C  NOTE:  ***          This is the brute force method.          ***
C         ***  Incorporate the neighbor table method ASAP.   ***
C
C     This is the traditional method optimized for the Convex C-210.
C
                  FLAG = 1
                  DO 31100 B2 = 2, BEADS-1
                    XTMP = ABS(XTEMP - X1TP(CM1XBD + B2))
                    YTMP = ABS(YTEMP - Y1TP(CM1XBD + B2))
                    ZTMP = ABS(ZTEMP - Z1TP(CM1XBD + B2))
                    IRX  = XTMP + 0.50
                    IRY  = YTMP + 0.50
                    IRZ  = ZTMP + 0.50
                    BDDIS2 =    (XTMP - IRX)**2
      &                       + (YTMP - IRY)**2  + (ZTMP - IRZ)**2
                    IF (BDDIS2 .LT. DIAM2) FLAG = FLAG + 2
31100               CONTINUE
                    IF (FLAG .GT. 1) THEN
                      VOLCNT = VOLCNT + 1
                      GO TO 31000
                    ENDIF
C
                    DO 31200 B2 = 1, BEADS
                      DO 31210 C2 = 1, C-1
                        XTMP = ABS(XTEMP - X1TP((C2 - 1)*BEADS + B2))
                        YTMP = ABS(YTEMP - Y1TP((C2 - 1)*BEADS + B2))
                        ZTMP = ABS(ZTEMP - Z1TP((C2 - 1)*BEADS + B2))
                        IRX  = XTMP + 0.50
                        IRY  = YTMP + 0.50
                        IRZ  = ZTMP + 0.50
                        BDDIS2 =    (XTMP - IRX)**2
      &                           + (YTMP - IRY)**2  + (ZTMP - IRZ)**2
                        IF (BDDIS2 .LT. DIAM2) FLAG = FLAG + 2
31210                 CONTINUE
                      IF (FLAG .GT. 1) THEN
                        VOLCNT = VOLCNT + 1
                        GO TO 31000
                      ENDIF
```

```
                  DO 31220 C2 = C+1, CHAINS
                    XTMP = ABS(XTEMP - X1TP((C2 - 1)*BEADS + B2))
                    YTMP = ABS(YTEMP - Y1TP((C2 - 1)*BEADS + B2))
                    ZTMP = ABS(ZTEMP - Z1TP((C2 - 1)*BEADS + B2))
                    IRX  = XTMP + 0.50
                    IRY  = YTMP + 0.50
                    IRZ  = ZTMP + 0.50
                    BDDIS2 =    (XTMP - IRX)**2
     &                       + (YTMP - IRY)**2  + (ZTMP - IRZ)**2
                    IF (BDDIS2 .LT. DIAM2) FLAG = FLAG + 2
31220               CONTINUE
                    IF (FLAG .GT. 1) THEN
                      VOLCNT = VOLCNT + 1
                      GO TO 31000
                    ENDIF
31200             CONTINUE
C
C    The following method is optimized for the CONVEX C-210.
C
C                 FLAG = 1.0
C                 LOWIL = IL(END)
C                 HIIL = IL(END+1)
C
C    Test to be sure that the bead has neighbors.
C
C  NOTE:  This test should be unnecessary since chains are present.
C         Revove it once one is sure that the process is working.
C
C                 IF (LOWIL .EQ. HIIL) THEN
C                   NONEI = NONEI + 1
C                   GO TO 31300
C                 ENDIF
C                 DO 31109 I = LOWIL, HIIL-1
C                   XTMP = XTEMP - X1TP(NB(I))
C                   YTMP = YTEMP - Y1TP(NB(I))
C                   ZTMP = ZTEMP - Z1TP(NB(I))
C                   XBOX = XTMP - ANINT(XTMP)
C                   YBOX = YTMP - ANINT(YTMP)
C                   ZBOX = ZTMP - ANINT(ZTMP)
C                   BDDIS2 = XBOX**2 + YBOX**2 + ZBOX**2
C                   IF (BDDIS2 .LE. DIAM2) then
C                     FLAG = FLAG + 1.0
C                   endif
C31109            CONTINUE
C 912    continue
C                 IF (IDINT(FLAG) .NE. 1) THEN
C                   VOLCNT = VOLCNT + 1
C                   GO TO 31000
C                 ENDIF
C
C   The excluded volume test is successful.  Set  XTP(B,C), YTP(B,C),
C  ZTP(B,C)  equal to  XTEMP, YTEMP, ZTEMP, respectively.
C
C    Calculate the new end vector and angle.
```

```
C
31300           RXNEW = XTEMP - X1TP(END)
                RYNEW = YTEMP - Y1TP(END)
                RZNEW = ZTEMP - Z1TP(END)
C                 .
C    Compute the new vectors VEC(1,C)  or  VEC(BEADS-1,C),
C  as necessary, and the new center of mass for the chain.
C
                IF (ENDMV .EQ. 1) THEN
                    I = END + (BEADS - 1)
                    XCMTP(C) = XCMTP(C)
     &                         + (XTEMP - X1TP(I)) / FLOAT(BEADS)
                    YCMTP(C) = YCMTP(C)
     &                         + (YTEMP - Y1TP(I)) / FLOAT(BEADS)
                    ZCMTP(C) = ZCMTP(C)
     &                         + (ZTEMP - Z1TP(I)) / FLOAT(BEADS)
C
                    TPVX(ENDAV) = -RXNEW
                    TPVY(ENDAV) = -RYNEW
                    TPVZ(ENDAV) = -RZNEW
                  ELSE
                    I = END - (BEADS - 1)
                    XCMTP(C) = XCMTP(C)
     &                         + (XTEMP - X1TP(I)) / FLOAT(BEADS)
                    YCMTP(C) = YCMTP(C)
     &                         + (YTEMP - Y1TP(I)) / FLOAT(BEADS)
                    ZCMTP(C) = ZCMTP(C)
     &                         + (ZTEMP - Z1TP(I)) / FLOAT(BEADS)
C
                    TPVX(PENAV) = RXNEW
                    TPVY(PENAV) = RYNEW
                    TPVZ(PENAV) = RZNEW
                ENDIF
                DO 31400 CNT = LOWAV, HIAV
                  TPVX(CNT) = XVC1TP(CNT+ARRSFT)
                  TPVY(CNT) = YVC1TP(CNT+ARRSFT)
                  TPVZ(CNT) = ZVC1TP(CNT+ARRSFT)
31400           CONTINUE
                DO 31450 CNT = C1XB1+1, C1XB1+BEADS-1
                  XVC1TP(CNT) = TPVX(CNT)
                  YVC1TP(CNT) = TPVY(CNT)
                  ZVC1TP(CNT) = TPVZ(CNT)
31450           CONTINUE
C
C  Calculate the angles for the new system.
C
                TPA(C1XB1+1) = 0.0
                TPA(PENAV) = - RBON2 * (RXNEW*RX + RYNEW*RY + RZNEW*RZ)
                DO 31500 CNT = LOWAV+1, HIAV
                  TPA(CNT) = ANG1TP(CNT+ARRSFT)
31500           CONTINUE
                DO 31550 CNT = C1XB1+1, C1XB1+BEADS-1
                  ANG1TP(CNT) = TPA(CNT)
31550           CONTINUE
```

```
C
C  Calculate the positions for the new system.
C
                TPX(END) = XTEMP
                TPY(END) = YTEMP
                TPZ(END) = ZTEMP
                DO 31600 CNT = LOWCT, HIGHCT
                  TPX(CNT) = X1TP(CNT+ARRSFT)
                  TPY(CNT) = Y1TP(CNT+ARRSFT)
                  TPZ(CNT) = Z1TP(CNT+ARRSFT)
31600           CONTINUE
                DO 31650 CNT = CM1XBD+1, CM1XBD+BEADS
                  X1TP(CNT) = TPX(CNT)
                  Y1TP(CNT) = TPY(CNT)
                  Z1TP(CNT) = TPZ(CNT)
31650           CONTINUE
C          CALL NEIGH(BDP1, BDX2, BDX3, BDX2P1, BDXCH, CRAYNM,
C     &                 LOCD2, LOCTN, NRNEI, SORTPR,
C     &                BOXSZ, DIAM, RANGE, RMAX,
C     &                ISEED,
C     &                IL, IRG, IRGY, IRGZ, JA1, JA2, JA3, JBS, JES,
C     &                KBS, KES, KJB, KJE, LOCX, MBS, MES, NB,
C     &                CX, CX1, CX2, CXSORT, POS1TP)
C
C  ***   End  CHAINS:  All of the chains have been moved.  ***
C
31000           CONTINUE
C
C    Metropolis sampling.
C
C  NOTE:  Check the new  ESSLLIBS  for a routine to do this on the IBM.
C
              ANGLSM=0.0
              DO 33000 CNT = 1, BM1XCH
                ANGLSM = ANGLSM + ANG1TP(CNT)
33000         CONTINUE
              BNDNEW = (0.50 / TAUTMP) * ((BEADS-2)*CHAINS + ANGLSM)
C
              HAMNEW = BNDNEW
              IF (HAMNEW .GT. HAMINI) THEN
                DELHT = (HAMNEW - HAMINI)
                IF (DELHT .GE. 170.0) THEN
                    ENGCNT = ENGCNT + 1
                    GO TO 30000
                  ELSE
                    BOLTZ = EXP(- DELHT)
                    IF (BOLTZ .LT. RANF()) THEN
                      ENGCNT = ENGCNT + 1
                      GO TO 30000
                    ENDIF
                ENDIF
              ENDIF
```

```
C
C     New positions are acceptable.  Place in permanent storage.
C
              ACCEPT = ACCEPT + 1
              MOVED = 0.0
              DO 34000 CNT = 1, CHAINS
                IF (CM1TP(CNT) .NE. CM1(CNT)) MOVED = MOVED + 1.0
34000         CONTINUE
              MOVTOT = MOVTOT + MOVED
              IF (MOVED .NE. 0.0) ACCMOV = ACCMOV + 1
              ACCFLG = 1
              BNDINI = BNDNEW
              HAMINI = HAMNEW
              CALL SCOPY(BM1XCH, ANG1TP, 1, ANG1, 1)
              CALL SCOPY(3*CHAINS, CM1TP, 1, CM1, 1)
              CALL SCOPY(3*BDXCH, POS1TP, 1, POS1, 1)
              CALL SCOPY(3*BM1XCH, VEC1TP, 1, VEC1, 1)
C
C           CALL NEIGH(BDP1, BDX2, BDX3, BDX2P1, BDXCH, CRAYNM,
C     &                 LOCD2, LOCTN, NRNEI, SORTPR,
C     &                BOXSZ, DIAM, RANGE, RMAX,
C     &                ISEED,
C     &                IL, IRG, IRGY, IRGZ, JA1, JA2, JA3, JBS, JES,
C     &                KBS, KES, KJB, KJE, LOCX, MBS, MES, NB,
C     &                CX, CX1, CX2, CXSORT, POS1TP)
C
C     ***          This is the end of the  MCSTEP  loop.          ***
C
30000         CONTINUE
              SUCCES = SUCCES + ACCEPT
              SUCMOV = SUCMOV + ACCMOV
C
C     ***          This is the end of the  EQUIL   loop.          ***
C
40000    CONTINUE
         PRINT*, 'VOL, ENG  ', VOLCNT, ENGCNT
         PRINT*, 'MOV, SUC  ', MOVTOT, SUCMOV
         IF (SUCMOV .NE. 0.0D0) MOVTOT = MOVTOT / SUCMOV
           PRINT*, 'Avg. number of chains moved per accepted config.  ',
     &           MOVTOT
         SUCCES = SUCCES / FLOAT(EQSTEP * MCLIM)
         SUCMOV = SUCMOV / FLOAT(EQSTEP * MCLIM)
         PRINT*, 'Success in the energy tests of REPEQ =  ', SUCCES
         PRINT*, 'Success in the movement tests of REPEQ =  ', SUCMOV
         PRINT*, 'NONEI  ', NONEI
         RETURN
         END
C
C
C
```

```
      SUBROUTINE RADIAL(BDXCH, BEADS, CENBD, CHAINS, SHELLS,
     &                  RSLRD2,
     &                  BDOTR, CMSHEL,
     &                  BDCEN, BDEND, BDMID, CM1, X1, Y1, Z1)
C
C     Parameters passed.
C
      INTEGER  BDXCH, BEADS, CENBD, CHAINS, SHELLS
      REAL     RSLRD2
C
C     Arrays passed.
C
      INTEGER  BDOTR(0:SHELLS), CMSHEL(0:SHELLS)
      REAL     BDCEN(0:SHELLS), BDEND(0:SHELLS), BDMID(0:SHELLS),
     &          CM1(3*CHAINS), X1(BDXCH), Y1(BDXCH), Z1(BDXCH)
C
C     Local scalar variables.
C
      INTEGER  B, BDCNT, C, CM1XBD, CMCNT, CNT, IRX, IRY, IRZ
      REAL     XTMP, YTMP, ZTMP
C
C     The counting for the Center of Masses' RDF.
C
C  NOTE:  The Center of Mass from which the distances are measured
C         should be included in this counting.
C
C  NOTE:  This calculation is only representative to 0.50*BOXSZ.
C         Information at longer distances requires more reflections
C         to be taken into account.
C
      DO 45000 C = 1, CHAINS-1
        DO 45100 CNT = C+1, CHAINS
          XTMP = ABS(CM1(C)              - CM1(CNT))
          YTMP = ABS(CM1(C +   CHAINS) - CM1(CNT +   CHAINS))
          ZTMP = ABS(CM1(C + 2*CHAINS) - CM1(CNT + 2*CHAINS))
          IRX = XTMP + 0.50
          IRY = YTMP + 0.50
          IRZ = ZTMP + 0.50
          CMCNT = SQRT( RSLRD2 * (   (XTMP - IRX)**2
     &                             + (YTMP - IRY)**2
     &                             + (ZTMP - IRZ)**2 ) )
          CMSHEL(CMCNT) = CMSHEL(CMCNT) + 1
45100     CONTINUE
45000   CONTINUE
C
C    The counting for the Beads' RDF.
C
C  NOTE:  The Bead from which the distances are measured should be
C         included in this counting.
C
C  NOTE:  This calculation is only representative to 0.50*BOXSZ.
C         Information at longer distances requires more reflections
C         to be taken into account.
C
```

```
C  Loop 46000 is for the final chain's correlation to all other
C  chains in the system.  This is referred to as "Others' RDF."
C
C  Loop 47000 is for the correlation of a chain to itself.  This
C  is referred to as "Self RDF."  All of the chains are involved
C  in obtaining this average.
C
C
        DO 46000 B = BDXCH-BEADS+1, BDXCH
          DO 46100 CNT = 1, BDXCH-BEADS
            XTMP = ABS(X1(B) - X1(CNT))
            YTMP = ABS(Y1(B) - Y1(CNT))
            ZTMP = ABS(Z1(B) - Z1(CNT))
            IRX = XTMP + 0.50
            IRY = YTMP + 0.50
            IRZ = ZTMP + 0.50
            BDCNT = SQRT( RSLRD2 * (    (XTMP - IRX)**2
     &                                + (YTMP - IRY)**2
     &                                + (ZTMP - IRZ)**2 ) )
            BDOTR(BDCNT) = BDOTR(BDCNT) + 1
46100     CONTINUE
46000   CONTINUE
C
        DO 47000 C = 1, CHAINS
          CM1XBD = (C - 1) * BEADS
C
C  The middle beads vs. the middle beads.
C
          DO 47100 B = 2, BEADS-3
            DO 47110 CNT = B+2, BEADS-1
              XTMP = X1(B + CM1XBD) - X1(CNT + CM1XBD)
              YTMP = Y1(B + CM1XBD) - Y1(CNT + CM1XBD)
              ZTMP = Z1(B + CM1XBD) - Z1(CNT + CM1XBD)
              BDCNT = SQRT(RSLRD2 * (XTMP**2 + YTMP**2 + ZTMP**2))
              BDMID(BDCNT) = BDMID(BDCNT) + 2.0
47110       CONTINUE
47100     CONTINUE
C
C  Bead 1 vs. the middle beads.
C
          DO 47200 B = 3, BEADS-1
            XTMP = X1(1 + CM1XBD) - X1(B + CM1XBD)
            YTMP = Y1(1 + CM1XBD) - Y1(B + CM1XBD)
            ZTMP = Z1(1 + CM1XBD) - Z1(B + CM1XBD)
            BDCNT = SQRT(RSLRD2 * (XTMP**2 + YTMP**2 + ZTMP**2))
            BDMID(BDCNT) = BDMID(BDCNT) + 1.0
            BDEND(BDCNT) = BDEND(BDCNT) + 1.0
47200     CONTINUE
C
C  Bead BEADS vs. the middle beads.
C
```

```
            DO 47300 B = 2, BEADS-2
              XTMP = X1(BEADS + CM1XBD) - X1(B + CM1XBD)
              YTMP = Y1(BEADS + CM1XBD) - Y1(B + CM1XBD)
              ZTMP = Z1(BEADS + CM1XBD) - Z1(B + CM1XBD)
              BDCNT = SQRT(RSLRD2 * (XTMP**2 + YTMP**2 + ZTMP**2))
              BDMID(BDCNT) = BDMID(BDCNT) + 1.0
              BDEND(BDCNT) = BDEND(BDCNT) + 1.0
47300       CONTINUE
C
C  Bead 1 vs. Bead BEADS.
C
            XTMP = X1(1 + CM1XBD) - X1(BEADS + CM1XBD)
            YTMP = Y1(1 + CM1XBD) - Y1(BEADS + CM1XBD)
            ZTMP = Z1(1 + CM1XBD) - Z1(BEADS + CM1XBD)
            BDCNT = SQRT(RSLRD2 * (XTMP**2 + YTMP**2 + ZTMP**2))
            BDEND(BDCNT) = BDEND(BDCNT) + 2.0
C
C     Bead CENBD vs. all other beads on the chain.
C
            DO 47400 B = 1, CENBD-2
              XTMP = X1(CENBD + CM1XBD) - X1(B + CM1XBD)
              YTMP = Y1(CENBD + CM1XBD) - Y1(B + CM1XBD)
              ZTMP = Z1(CENBD + CM1XBD) - Z1(B + CM1XBD)
              BDCNT = SQRT(RSLRD2 * (XTMP**2 + YTMP**2 + ZTMP**2))
              BDCEN(BDCNT) = BDCEN(BDCNT) + 1.0
47400       CONTINUE
              DO 47500 B = CENBD+2, BEADS
                XTMP = X1(CENBD + CM1XBD) - X1(B + CM1XBD)
                YTMP = Y1(CENBD + CM1XBD) - Y1(B + CM1XBD)
                ZTMP = Z1(CENBD + CM1XBD) - Z1(B + CM1XBD)
                BDCNT = SQRT(RSLRD2 * (XTMP**2 + YTMP**2 + ZTMP**2))
                BDCEN(BDCNT) = BDCEN(BDCNT) + 1.0
47500         CONTINUE
47000     CONTINUE
          RETURN
          END
C
C
C
          SUBROUTINE SCATTR (BDXCH, BEADS, CHAINS, QSIZE,
     &                       PIX2, RPIX2, SINCNV,
     &                       OTRSCT, RJM2, SINTAB, SLFSCT, WAVENM,
     &                        POS1, PS1TP)
C
C     Parameters passed.
C
          INTEGER  BDXCH, BEADS, CHAINS, QSIZE
          REAL     PIX2, RPIX2, SINCNV
C
C     Arrays passed.
C
          REAL     OTRSCT(QSIZE), RJM2(BDXCH), SINTAB(0:9999),
     &             SLFSCT(QSIZE), WAVENM(QSIZE),
     &             POS1(3*BDXCH), PS1TP(3*BDXCH)
```

```
C
C    Local scalar variables.
C
        INTEGER  C, C1, CM1XBD, CNT, IR, J, M, Q
        REAL     RJM
C
C    Shift all of the chains back into their window locations,
C  regardless of whether this splits the chains or not.
C
        DO 47500 CNT = 1, 3*BDXCH
          PS1TP(CNT) = POS1(CNT) - ANINT(POS1(CNT))
47500   CONTINUE
C
C    Calculation of the Scattering Function from the Difference Vector.
C
C  Loop 48000 is for the contribution to the Scattering Function
C  from beads on the same chain.
C
        DO 48000 C = 1, CHAINS
          DO 48100 J = 1, BEADS-1
            DO 48110 M = J+1, BEADS
              CM1XBD = (C - 1) * BEADS
              RJM = SQRT(  (PS1TP(CM1XBD + J)
     &                       - PS1TP(CM1XBD + M))**2
     &                   + (PS1TP(BDXCH + CM1XBD + J)
     &                       - PS1TP(BDXCH + CM1XBD + M))**2
     &                   + (PS1TP(2*BDXCH + CM1XBD + J)
     &                       - PS1TP(2*BDXCH + CM1XBD + M))**2 )
              DO 48111 Q = 1, QSIZE
                IR = RPIX2 * (RJM * WAVENM(Q))
                CNT = SINCNV * ((RJM * WAVENM(Q)) - PIX2*IR)
                SLFSCT(Q) = SLFSCT(Q)  + SINTAB(CNT)
     &                                     /(RJM * WAVENM(Q))
48111         CONTINUE
48110       CONTINUE
48100     CONTINUE
48000   CONTINUE
C
C    Loop 49000 deals with the contribution to the Scattering Function
C  from beads on other chains from the reference chain.
C
        CM1XBD = INT(CHAINS * RANF()) * BEADS
        DO 49000 J = CM1XBD+1, CM1XBD+BEADS
          DO 49100 M = 1, CM1XBD
            RJM2(M) = SQRT(  (PS1TP(J)           - PS1TP(M))**2
     &                   + (PS1TP(  BDXCH + J) - PS1TP(  BDXCH + M))**2
     &                   + (PS1TP(2*BDXCH + J) - PS1TP(2*BDXCH + M))**2 )
49100     CONTINUE
          DO 49200 M = CM1XBD+BEADS+1, BDXCH
            RJM2(M) = SQRT(  (PS1TP(J)           - PS1TP(M))**2
     &                   + (PS1TP(  BDXCH + J) - PS1TP(  BDXCH + M))**2
     &                   + (PS1TP(2*BDXCH + J) - PS1TP(2*BDXCH + M))**2 )
49200     CONTINUE
```

```
            DO 49300 Q = 1, QSIZE
              DO 49310 M = 1, CM1XBD
                IR = RPIX2 * (RJM2(M) * WAVENM(Q))
                CNT = SINCNV * ((RJM2(M) * WAVENM(Q)) - PIX2*IR)
                OTRSCT(Q) = OTRSCT(Q)  + SINTAB(CNT)
     &                                 /(RJM2(M) * WAVENM(Q))
49310           CONTINUE
                DO 49320 M = CM1XBD+BEADS+1, BDXCH
                IR = RPIX2 * (RJM2(M) * WAVENM(Q))
                CNT = SINCNV * ((RJM2(M) * WAVENM(Q)) - PIX2*IR)
                OTRSCT(Q) = OTRSCT(Q)  + SINTAB(CNT)
     &                                 /(RJM2(M) * WAVENM(Q))
49320           CONTINUE
49300         CONTINUE
49000       CONTINUE
            RETURN
            END
```

APPENDIX C

CONTINUOUS SPACE, ISOBARIC-ISOTHERMAL ENSEMBLE CODE

Representative Code
for the Continuous Space, Isobaric-Isothermal Ensemble Simulations
As Executed on the Alliant FX/40-4 Processors
at the University of Massachusetts, Amherst, Massachusetts

```
C
C     Parameters.
C     Change the parameters to the appropriate values before using.
C     The following variables can be independently controlled:  the
C     number of chains, the length of the chain, the volume occupied
C     by a chain, and the osmotic presure.  This allows control of
C     polymer number concentration, molecular weight, and volume
C     fraction.  The initial volume of the system is set equal to unity.
C
C     The osmotic pressure defined below (OSMOT) is dimensionless, and
C     is really the "initial" compressibility = (osmotic pressure)*
C     (initial volume)/(number of chains)/(kT).
C
        INTEGER       BEADS, CHAINS, EQSTEP, LENLIM, MCLIM,
     &                 NPRINT, NSAMP,
     &                 UNOBSR
C
        PARAMETER     (BEADS=11, CHAINS=100, EQSTEP=100,
     &                  LENLIM=1,
     &                  MCLIM=1, NPRINT=100, NSAMP=100000, UNOBSR=00)
C
        REAL*8        ALPHA, DIAM, DIAM2, DIAMD2, DIAMD8, OSMOT,
     &                 TAU,
     &                 TRANMX, VDIFMX
        PARAMETER     (ALPHA=3.00d0,
     &                  DIAM=6.60D-02,
     &                  DIAM2=DIAM**2,
     &                  DIAMD2=DIAM/2.0D0, DIAMD8=DIAM/8.0D0,
     &                  OSMOT=10.00d0, TAU=0.50D0,
     &                  TRANMX=0.0d0, VDIFMX=0.0005d0 )
C
C  Define PI and compute bead volume.
C
        REAL*8        BDVOL, PI
        PARAMETER     ( PI= 3.14159 26535 89793 23846 ,
     &                    BDVOL = (DIAM**3)*PI/6.D0 )
C
        INTEGER       BDXCH, BM1XCH
        PARAMETER     (BDXCH=BEADS*CHAINS, BM1XCH=(BEADS-1)*CHAINS)
C
C  NOTE:  BONDL < SQRT(2)*DIAM  by just a bit.
C         SQRT(2) = 1.4142 1356 2373 0950 4880 1688 72
```

```
C
      REAL*8        BONDL, BONDL2, RBON2
      PARAMETER     (BONDL=1.4142135623730000D0*DIAM, BONDL2=BONDL**2,
     &                RBON2=(1.0D0/BONDL2))
C
C     Scalar variables.
C
      INTEGER       ACCEPT, ACCFLG, ACCMOV, ARRSFT, ATMPTS,
     &               B, B2, BDCNT, BEADLW, BEADHI, BOXCNT,
     &               C, C1, C1XB1, C2, C2M1XB, CM1XB,
     &               CM1XBD, CNT, CONFGS,
     &               END, ENDMV, END_AV, ENGCNT,
     &               FAILDN, FAILUP, FLAG, GFLAG, HIGHCT, HI_AV
      INTEGER       I, INAUGH, IRX, IRY, IRZ, ISEED, J, K,
     &               LENSTP, LOWCT, LOW_AV, MCSTEP,
     &               PENULT, PEN_AV, PRINFG, RCOUNT, SAMPL, SELCNT,
     &               SELFG, SYSCHG,
     &               TPWR, TPWRMX, VFLGDN, VFLGUP,
     &               VLNULD, VLNULU, VOLCNG,
     &               VOLCNT, VOLDFG, VOLDN, VOLFLG, VOLNUL, VOLUP,
     &               VOLUFG
C
      REAL*4        ELSPTL
      REAL*4        RLAMB, DM
      REAL*4        TARR1(2)
      REAL*8        ANGLSM,
     &               BDDIS2, BDAVS, BDS2,
     &               BNDAVS, BNDINI, BNDNEW, BNDS2, BNDVAR, BOLTZ,
     &               CHGND, CHGNU, CHGRAD,
     &               CHGRAU, COTHE,
     &               CTPX, CTPY, CTPZ,
     &               CXTEMP, CYTEMP, CZTEMP,
     &               DELHT, DELX, DELY, DELZ,
     &               DENSTY, DENOM, DENOM2
      REAL*8        DIST, DISTAV,
     &               DISVAR, DSTAV, DSTSM2,
     &               DTSM2,
     &               ENRAT,
     &               FRACDN, FRACUP, GAMMA,
     &               HAMINI, HAMLTN, HAMNEW, HAMS2, HAMVAR,
     &               HMLTN, HMS2, LDIFF,
     &               LDIFMX, LENGTH, LENTMP, LEN2, LENTP2, MOVED
      REAL*8        PNTIN, PNTRL,
     &               RANDOM, RANTMP, RATIO,
     &               RGY2, RGYAVG, RLEN, RLENTP, RNN2, RNNAVG,
     &               RX, RY, RZ, RXNEW, RYNEW, RZNEW, SCCES,
     &               SCMOV, SELRAT,
     &               SITHE, SMVOL2, STIFF, SUCCES, SUCMOV,
     &               TAUTMP, TIMEXC,
     &               VOLAVS, VOLFAC,
     &               VOLRAT, VOLSM2, VOLTMP, VOLUME, VOLVAR,
     &               VLAVS, VLSM2
      REAL*8        XCMC, YCMC, ZCMC,
     &               XDIST, YDIST, ZDIST,
```

```
     &                 XTEMP, YTEMP, ZTEMP,
     &                 XTEST, YTEST, ZTEST,
     &                 XTMP, YTMP, ZTMP,
     &                 XTRANS, YTRANS, ZTRANS
C
CC   Array variables.
C
        INTEGER        BDPERC(CHAINS)
C
        REAL*8         ANG1(BM1XCH), ANG1TP(BM1XCH),
     &                  CM1(3*CHAINS),
     &                  CM1TP(3*CHAINS),
     &                  POS1(3*BDXCH), POS1TP(3*BDXCH),
     &                  RGYRA2(NSAMP), RNTON2(NSAMP)
        REAL*8         TPA(BM1XCH),
     &                  TPVX(BM1XCH), TPVY(BM1XCH), TPVZ(BM1XCH),
     &                  TPX(BDXCH), TPY(BDXCH), TPZ(BDXCH),
     &                  VEC1(3*BM1XCH), VEC1TP(3*BM1XCH),
     &                  X1C(BDXCH), Y1C(BDXCH), Z1C(BDXCH)
C
        REAL*8         X1(BDXCH), Y1(BDXCH), Z1(BDXCH)
        EQUIVALENCE    (X1(1), POS1(1)), (Y1(1), POS1(BDXCH+1)),
     &                   (Z1(1), POS1(2*BDXCH+1))
        REAL*8         X1TP(BDXCH), Y1TP(BDXCH), Z1TP(BDXCH)
        EQUIVALENCE    (X1TP(1), POS1TP(1)), (Y1TP(1), POS1TP(BDXCH+1)),
     &                   (Z1TP(1), POS1TP(2*BDXCH+1))
        REAL*8         XCM(CHAINS), YCM(CHAINS), ZCM(CHAINS)
        EQUIVALENCE    (XCM(1), CM1(1)), (YCM(1), CM1(CHAINS+1)),
     &                   (ZCM(1), CM1(2*CHAINS+1))
        REAL*8         XCMREF(CHAINS), YCMREF(CHAINS), ZCMREF(CHAINS)
        REAL*8         XCMTP(CHAINS), YCMTP(CHAINS), ZCMTP(CHAINS)
        EQUIVALENCE    (XCMTP(1), CM1TP(1)), (YCMTP(1), CM1TP(CHAINS+1)),
     &                   (ZCMTP(1), CM1TP(2*CHAINS+1))
        REAL*8         XVEC1(BM1XCH), YVEC1(BM1XCH), ZVEC1(BM1XCH)
        EQUIVALENCE    (XVEC1(1), VEC1(1)), (YVEC1(1), VEC1(BM1XCH+1)),
     &                   (ZVEC1(1), VEC1(2*BM1XCH+1))
        REAL*8         XVC1TP(BM1XCH), YVC1TP(BM1XCH), ZVC1TP(BM1XCH)
        EQUIVALENCE    (XVC1TP(1), VEC1TP(1)),
     &                   (YVC1TP(1), VEC1TP(BM1XCH+1)),
     &                   (ZVC1TP(1), VEC1TP(2*BM1XCH+1))
C
CC   Declarations of functions.
C
        EXTERNAL       DASUM, DRAN, DTIME
        REAL*8         DASUM, DRAN
        REAL*4         DTIME
C
CC   Initial ISEED.
C
          DATA           ISEED/4444444/
C
CC   Main program.
C
        ELSPTL = DTIME(TARR1)
```

```
C
C-----------------------------------------------------------------------
C                              Initializations.                        -01B
C-----------------------------------------------------------------------
C
C    Test to be sure that the length of the chains does not influence
C  the periodic boundary conditions.
C
        IF (((BEADS - 1)*BONDL + DIAM) .GE. 1.0D0) THEN
          STOP ' The chains are longer than the box.'
        ENDIF
C
C    Save the initial value of the  ISEED  to be sure the same
C  simulation is not repeated.
C
        INAUGH = ISEED
C
C    The subroutine EQISO.F equilibrates the system via the Metropolis
C  method at a constant volume of one.  This may be particularly
C  important at low temperatures in which the chains must straighten
C  out.  The system temperature TAUTMP is lowered in three stages.
C  TAU is the final system temperature, and the temperature of the
C  final call to EQISO.F
C
C    Set the value for  TAUTMP  equal to a large value to start, and
C  then lower it to the desired value of  TAU  by an order of magnitude
C  in each iteration of Loop 10000.
C
C  NOTE:  The reduction of  TAUTMP  by an order of magnitude each
C         iteration may be too much to equilibrate the system
C         efficiently.
C
C  NOTE:  By the test below, at least three iterations of the loop
C         occur for any value of  TAU  even if it is large.  Since the
C         initialization is by self-avoiding walk, the initial tempera-
C         ture of the system is effectively infinite.
C
        TPWRMX = IDINT(DLOG10(100.D0 / TAU))
        IF (TPWRMX .LT. 2) TPWRMX = 2
C
C    Initalize the system temperature and timing function.
C
        TAUTMP = TAU * (10.0D0**TPWRMX)
        ELSPTL = DTIME(TARR1)
C
C    Generate the initial configuration in the unit cube.
C
        CALL ST_REP (ALPHA, BDXCH, BEADS, BM1XCH, CHAINS,
     &                 BONDL, DIAM, DIAM2, PI,
     &                  RBON2,
     &                 ISEED,
     &                 HAMINI, OSMOT, TAUTMP,
     &                 BDPERC,
     &                 ANG1, X1, Y1, Z1, XCM, YCM, ZCM,
```

```
     &                  XCMTP, YCMTP, ZCMTP,
     &                  XVEC1, YVEC1, ZVEC1)
        ELSPTL = DTIME(TARR1)
        print*, 'Time for ST_REP (secs.)', ELSPTL
C
CC    Loop 10000: Equilibrate the system to the desired value of  TAU.
CC    This is done in the initial unit cube.
C
        DO 10000 TPWR = TPWRMX, 0, -1
          TAUTMP = TAU * (10.0D0**TPWR)
C
CC    Store the center-of-mass coordinates for comparison after
CC    the program leaves EQ_REP.
C
          IF (TPWR .EQ. TPWRMX) THEN
            CM1TP(1: 3*CHAINS) = CM1(1: 3*CHAINS)
          ENDIF
C
CC    Adjust the value of HAMINI to correspond to the
CC    new temperature.
C
          IF( TPWR .NE. TPWRMX ) THEN
            HAMINI =  10.d0 * HAMINI
          ENDIF
C
          CALL EQ_REP (BDXCH, BDVOL, BEADS, BM1XCH, CHAINS,
     &                  EQSTEP,
     &                  MCLIM,
     &                  BONDL, DIAM2, RBON2, PI,
     &                  ISEED,
     &                  HAMINI, OSMOT, TAUTMP, TRANMX,
     &                  ANG1, ANG1TP, CM1, CM1TP,
     &                  POS1, POS1TP,
     &                  TPA, TPVX, TPVY, TPVZ, TPX, TPY, TPZ,
     &                  VEC1, VEC1TP,
     &                  X1TP, Y1TP, Z1TP,
     &                  X1C, Y1C, Z1C, XCMTP, YCMTP, ZCMTP,
     &                  XVC1TP, YVC1TP, ZVC1TP)
C
          ELSPTL = DTIME(TARR1)
C
CC    Compute and write the average distance through which a
CC    chain (center-of-mass) moves during the course of EQISO.F
C
          XDIST = 0.D0
          YDIST = 0.D0
          ZDIST = 0.D0
          DO 5000 C = 1, CHAINS
            XDIST = XDIST + (XCM(C) - XCMREF(C))**2
            YDIST = YDIST + (YCM(C) - YCMREF(C))**2
            ZDIST = ZDIST + (ZCM(C) - ZCMREF(C))**2
5000      CONTINUE
          DIST = (XDIST + YDIST + ZDIST)/DBLE(CHAINS)
C
```

```
C
            print*, 'For last EQ_REP, chains moved (avg.)',DIST
            print*, 'Time and Temp for EQ_REP (secs.)', ELSPTL, TAUTMP
            print*, ' '
10000    CONTINUE
C
C
C-----------------------------------------------------------------------
C                                                                      01E
C-----------------------------------------------------------------------
C                                                                      02B
C                          Run the system.
C-----------------------------------------------------------------------
C
C       Initializations.
C
         BNDAVS = 0.D0
         BNDS2  = 0.D0
         DISTAV = 0.D0
         DSTSM2 = 0.D0
         FAILDN = 0
         FAILUP = 0
         GFLAG  = 0
         HAMLTN = 0.D0
         HAMS2  = 0.D0
         LENGTH = 1.D0
         RCOUNT = 0
         SELCNT = 0
         TIMEXC = 0.D0
         VOLAVS = 0.D0
         VOLCNG = 0
         VOLCNT = 0
         VOLDN  = 0
         VOLFLG = 0
         VOLSM2 = 0.D0
         VOLUME = 1.D0

C    Loop 50000:  Sampling loop.  With each pass through this loop,
C  samples are recorded for accumulating the final average quantities.
C
            DO 50000 CONFGS = 1, NSAMP+UNOBSR
C             print*, 'confgs,nsamp=', confgs, nsamp
              ACCEPT = 0
              ACCMOV = 0
C    Loop 40000:  Volume change and metropolis testing.  The metropolis
C  test is performed with every pass through this loop.  The recording
C  of configuration data ("sampling") is executed after LENLIM passes
C  through this loop (i.e.,after the program drops through the loop.)
C
              DO 40000 LENSTP = 1, LENLIM
C
C             Initialize the distance traveled by the chain CM.
C             These variables measure the distance traveled during
C             the course of one step (on average).
C
```

```
                XDIST = 0.D0
                YDIST = 0.D0
                ZDIST = 0.D0
                DIST = 0.D0
C
CC      Choose a new volume.
CC      It is always necessary to check for
CC      excluded volume failures.  In the case of failure, the
CC      program cycles back to the top of this loop, and chooses
CC      another volume.
C
                RANDOM = DRAN(ISEED)
                VOLTMP = VDIFMX*( 2.D0*RANDOM - 1.D0 ) + VOLUME
                LENTMP = ( VOLTMP )** (1.D0/ 3.D0)
                RLEN = 1.D0/LENGTH
                LEN2 = (LENGTH**2)
                RLENTP = 1.D0/LENTMP
                LENTP2 = LENTMP**2
C
CC    Transfer the information to the working arrays.
C
                IF (ACCFLG .EQ. 0) THEN
C
CC    These Fortran 8X statements are replaced by calls to DCOPY.F
CC    on the Convex.
C
                  ANG1TP(1:BM1XCH) = ANG1(1:BM1XCH)
                  CM1TP(1: 3*CHAINS) = CM1(1: 3*CHAINS)
                  POS1TP(1:3*BDXCH) = POS1(1:3*BDXCH)
                  VEC1TP(1:3*BM1XCH) = VEC1(1:3*BM1XCH)
                ENDIF
                ACCFLG = 0
                VOLUFG = 0
                VOLDFG = 0
C
CC      Loop 30000 is the "core" Monte Carlo Loop.  This moves the
CC    chains to a new configuration (with the volume fixed), and
CC    insures that the generated configurations meet the excluded
CC    volume test.  In each MC step, an attempt is made to move
CC    each chain once and only once.  (The attempt is successful
CC    if, after moving a single chain, the exc. vol. test is passed.)
CC    Loop 31000 moves a single chain.  There are MCLIM MC steps in
CC    loop 30000 (i.e., for each volume chosen).
C
                DO 30000 MCSTEP = 1, MCLIM
C
CC      In loop 31000, one chain is picked out and moved by reptation
CC    plus translation (if TRANMX .ne. 0).
C
```

```
                  DO 31000 C1 = 1, CHAINS
                    ATMPTS = ATMPTS + 1
                    C = CHAINS*DRAN(ISEED) + 1
                    CM1XBD = (C - 1) * BEADS
                    C1XB1  = (C - 1) * (BEADS - 1)
                    ENDMV = 2*DRAN(ISEED) + 1
C
C     Translate chain C through a randomly chosen vector.
C     X1C, Y1C, and Z1C are arrays for the temporary storage
C     of the positions of chain C, in its translated position.
C     These arrays are taken to be of dimension BEADS*CHAINS
C     for convenience.
C
                    XTRANS = TRANMX * ( 2.D0*DRAN(ISEED) - 1.D0 )
                    YTRANS = TRANMX * ( 2.D0*DRAN(ISEED) - 1.D0 )
                    ZTRANS = TRANMX * ( 2.D0*DRAN(ISEED) - 1.D0 )
                    DO 33000 B2 = 1, BEADS
                      I = CM1XBD + B2
                      X1C(I) = X1TP(I) + XTRANS
                      Y1C(I) = Y1TP(I) + YTRANS
                      Z1C(I) = Z1TP(I) + ZTRANS
33000               CONTINUE
                    XCMC = XCMTP(C) + XTRANS
                    YCMC = YCMTP(C) + YTRANS
                    ZCMC = ZCMTP(C) + ZTRANS
C
C     This IF construction sets some variables used in reptating chain C.
C  For ENDMV=1, the vector  A  is  -VEC(2,C)  and the vector  R
C  is  -VEC(1,C).  For  ENDMV=BEADS, vector  A  is  VEC(BEADS-2,C)  and
C  vector  R  is  VEC(BEADS-1,C).
C
                    IF (ENDMV .EQ. 1) THEN
                        ENDMV = 1
                        END = CM1XBD + 1
                        PENULT = END + 1
                        LOWCT  = END + 1
                        HIGHCT = CM1XBD + BEADS
                        END_AV = C1XB1 + 1
                        PEN_AV = END_AV + 1
                        LOW_AV = PEN_AV
                        HI_AV  = C1XB1 + (BEADS - 1)
                        ARRSFT = -1
                        BEADLW = 0
                        BEADHI = BEADS-1
C
                        RX = -XVC1TP(END_AV)
                        RY = -YVC1TP(END_AV)
                        RZ = -ZVC1TP(END_AV)
                      ELSE
                        ENDMV = BEADS
                        END = CM1XBD + BEADS
                        PENULT = END - 1
                        LOWCT  = CM1XBD + 1
                        HIGHCT = END - 1
```

```
                        END_AV = C1XB1 + (BEADS - 1)
                        PEN_AV = END_AV
                        LOW_AV = C1XB1 + 1
                        HI_AV  = END_AV - 1
                        ARRSFT = 1
                        BEADLW = 2
                        BEADHI = BEADS+1
C
                        RX = XVC1TP(PEN_AV)
                        RY = YVC1TP(PEN_AV)
                        RZ = ZVC1TP(PEN_AV)
                    ENDIF
C
CC    Compute the cosine and sine of the rotation angles GAMMA and
C   THETA.  GAMMA varies from -PI to PI.  THETA  varies from 0 to PI.
C
                    GAMMA = PI * (2.0D+00*DRAN(ISEED) - 1.0D+00)
                    COTHE = 2.0D0*DRAN(ISEED) - 1.0D0
                    SITHE = DSQRT(1.0D0 - COTHE**2)
C
CC    Calculate the new coordinates for the end bead.  (Reptation.)
C   This amounts to applying a vector decomposition as if the
C   vector formed by the present positions of the penultimate
C   and end beads lies along the laboratory Z-axis.  This really
C   has no bearing on the result as long as the new vector has
C   a length equal to  BONDL, which is assured by this method.
C
                    XTEMP = X1C(END) + BONDL*SITHE*DCOS(GAMMA)
                    YTEMP = Y1C(END) + BONDL*SITHE*DSIN(GAMMA)
                    ZTEMP = Z1C(END) + BONDL*COTHE
C
CC
CC        Perform the excluded volume test for the moved chain.
CC        NB:  The box size here is (LENGTH)**3
CC
C   NOTE:  ***         This is the brute force method.          ***
C          ***  Incorporate the neighbor table method ASAP.    ***
C
                    FLAG = 1
                    SELFG = 1
C
CC        First, check for failures due to chain C interacting
CC        with itself.  This happens often, and is relatively
CC        efficient to check.
C
                    C2 = C
                    DO 31101 B2 = 2, BEADS-1
C
CC        XTEMP, YTEMP, ZTEMP give the coordinates of the
C         beads being checked against.
C
```

```
                    I = (C2-1)*BEADS + B2
                    XTMP = ( DABS(XTEMP - X1C(I)) )*RLEN
                    YTMP = ( DABS(YTEMP - Y1C(I)) )*RLEN
                    ZTMP = ( DABS(ZTEMP - Z1C(I)) )*RLEN
                    IRX = XTMP + 0.5D+00
                    IRY = YTMP + 0.5D+00
                    IRZ = ZTMP + 0.5D+00
                    BDDIS2 = ( (XTMP - IRX)**2 + (YTMP - IRY)**2
     &                     + (ZTMP - IRZ)**2  )*LEN2
                    IF (BDDIS2 .LT. DIAM2) THEN
                      FLAG = 2
                      SELFG = 2
                    ENDIF
C
31101             CONTINUE
31115             IF(FLAG.GT.1) GOTO 32000
C
C       Two different versions of the excluded volume check (for
C       chain C with other chains) are included
C       depending on whether TRANMX = 0.d0.  If this parameter
C       is zero, then there is no translational movement of the
C       chain, and it is satisfactory to check against the end
C       bead of chain C only.  Otherwise, all of the beads in
C       C must be checked for excluded volume.
C
                  IF( TRANMX .EQ. 0.D0 ) THEN
C
C       Check against the end beads of chain C only.
C
                DO 31100 B2 = 1, BEADS
                  DO 31110 C2 = 1, C-1
                    I = (C2-1)*BEADS + B2
                    XTMP = ( DABS(XTEMP - X1TP(I)) )*RLEN
                    YTMP = ( DABS(YTEMP - Y1TP(I)) )*RLEN
                    ZTMP = ( DABS(ZTEMP - Z1TP(I)) )*RLEN
                    IRX = XTMP + 0.5D+00
                    IRY = YTMP + 0.5D+00
                    IRZ = ZTMP + 0.5D+00
                    BDDIS2 = ( (XTMP - IRX)**2 + (YTMP - IRY)**2
     &                     + (ZTMP - IRZ)**2  )*LEN2
                    IF (BDDIS2 .LT. DIAM2) FLAG = 2
31110               CONTINUE
                    IF( FLAG .GT. 1) GO TO 32000

                    DO 31111 C2 = C+1, CHAINS
                      I = (C2-1)*BEADS + B2
                      XTMP = ( DABS(XTEMP - X1TP(I)) )*RLEN
                      YTMP = ( DABS(YTEMP - Y1TP(I)) )*RLEN
                      ZTMP = ( DABS(ZTEMP - Z1TP(I)) )*RLEN
                      IRX = XTMP + 0.5D+00
                      IRY = YTMP + 0.5D+00
                      IRZ = ZTMP + 0.5D+00
```

```
                    BDDIS2 = ( (XTMP - IRX)**2 + (YTMP - IRY)**2
     &                     + (ZTMP - IRZ)**2  )*LEN2
                    IF (BDDIS2 .LT. DIAM2) FLAG = 2
31111               CONTINUE
                    IF( FLAG .GT. 1) GO TO 32000

31100             CONTINUE
                ELSE
C
C     All of the beads on chain C must be checked, as the entire
C  chain has been translated.
C
                  DO 31300 B = BEADLW, BEADHI
                    IF( B.EQ.0 .OR. B.EQ.(BEADS+1) )  THEN
                      XTEST = XTEMP
                      YTEST = YTEMP
                      ZTEST = ZTEMP
                    ELSE
                      J = (C-1)*BEADS + B
                      XTEST = X1C(J)
                      YTEST = Y1C(J)
                      ZTEST = Z1C(J)
                    ENDIF
                    DO 31200 B2 = 1, BEADS
                      DO 31210 C2 = 1, C-1
C
C       xtest,ytest,ztest give the coordinates of the
C       bead being checked against.
C
                        I = (C2-1)*BEADS + B2
                        XTMP = ( DABS(XTEST - X1TP(I)) )*RLEN
                        YTMP = ( DABS(YTEST - Y1TP(I)) )*RLEN
                        ZTMP = ( DABS(ZTEST - Z1TP(I)) )*RLEN
                        IRX = XTMP + 0.5D+00
                        IRY = YTMP + 0.5D+00
                        IRZ = ZTMP + 0.5D+00
                        BDDIS2 = ( (XTMP - IRX)**2 + (YTMP - IRY)**2
     &                         + (ZTMP - IRZ)**2  )*LEN2
                        IF (BDDIS2 .LT. DIAM2) FLAG = 2
31210                 CONTINUE
                      IF( FLAG .GT. 1) GO TO 32000

                      DO 31211 C2 = C+1, CHAINS
                        I = (C2-1)*BEADS + B2
                        XTMP = ( DABS(XTEST - X1TP(I)) )*RLEN
                        YTMP = ( DABS(YTEST - Y1TP(I)) )*RLEN
                        ZTMP = ( DABS(ZTEST - Z1TP(I)) )*RLEN
                        IRX = XTMP + 0.5D+00
                        IRY = YTMP + 0.5D+00
                        IRZ = ZTMP + 0.5D+00
                        BDDIS2 = ( (XTMP - IRX)**2 + (YTMP - IRY)**2
     &                         + (ZTMP - IRZ)**2  )*LEN2
                        IF (BDDIS2 .LT. DIAM2) FLAG = 2
31211                 CONTINUE
```

```
31200                     IF( FLAG .GT. 1) GO TO 32000
31300                   CONTINUE
                      CONTINUE
                    ENDIF
C
32000               IF (FLAG .NE. 1) THEN
                      VOLCNT = VOLCNT + 1
                      IF(SELFG .NE. 1) SELCNT = SELCNT + 1
                      GO TO 31000
                    ENDIF
C
C       End of the excluded volume test.
C
C    The next statements put the new chain C center-of-mass and
C    bead positions in POS1TP, and CM1TP.  The new bond vectors
C    are stored in VEC1TP, and the new bond angles in ANG1TP.  (See
C    the EQUIVALENCE statements at the top of this listing.)
C    Note that these arrays are all temporary.   The transfer to
C    a permanent array can be made only in the 40000 loop, if the
C    Metropolis test is passed.  This loop is just developing a new
C    system configuration.
C
C   ZTP(B,C)  equal to  XTEMP, YTEMP, ZTEMP, respectively.
C
C     Calculate the new end vector and angle.
C
                    RXNEW = XTEMP - X1C(END)
                    RYNEW = YTEMP - Y1C(END)
                    RZNEW = ZTEMP - Z1C(END)
C
C     Compute the new vectors VEC(1,C)  or  VEC(BEADS-1,C),
C   as necessary, and the new center of mass for the chain.
C
                    IF (ENDMV .EQ. 1) THEN
                        I = END + (BEADS - 1)
                        XCMTP(C) = XCMC
     &                           + (XTEMP - X1C(I)) / DBLE(BEADS)
                        YCMTP(C) = YCMC
     &                           + (YTEMP - Y1C(I)) / DBLE(BEADS)
                        ZCMTP(C) = ZCMC
     &                           + (ZTEMP - Z1C(I)) / DBLE(BEADS)
C
                        TPVX(END_AV) = -RXNEW
                        TPVY(END_AV) = -RYNEW
                        TPVZ(END_AV) = -RZNEW
                      ELSE
                        I = END - (BEADS - 1)
                        XCMTP(C) = XCMC
     &                           + (XTEMP - X1C(I)) / DBLE(BEADS)
                        YCMTP(C) = YCMC
     &                           + (YTEMP - Y1C(I)) / DBLE(BEADS)
                        ZCMTP(C) = ZCMC
     &                           + (ZTEMP - Z1C(I)) / DBLE(BEADS)
```

```
C
                    TPVX(PEN_AV) = RXNEW
                    TPVY(PEN_AV) = RYNEW
                    TPVZ(PEN_AV) = RZNEW
                  ENDIF
c
                  DO 31400 CNT = LOW_AV, HI_AV
                    TPVX(CNT) = XVC1TP(CNT+ARRSFT)
                    TPVY(CNT) = YVC1TP(CNT+ARRSFT)
                    TPVZ(CNT) = ZVC1TP(CNT+ARRSFT)
31400             CONTINUE
                  DO 31450 CNT = C1XB1+1, C1XB1+BEADS-1
                    XVC1TP(CNT) = TPVX(CNT)
                    YVC1TP(CNT) = TPVY(CNT)
                    ZVC1TP(CNT) = TPVZ(CNT)
31450             CONTINUE
C
C  Calculate the angles for the new system.
C
                  TPA(C1XB1+1) = 0.0D0
                  TPA(PEN_AV) = -(RXNEW*RX + RYNEW*RY + RZNEW*RZ)*RBON2
                  DO 30200 CNT = LOW_AV+1, HI_AV
                    TPA(CNT) = ANG1TP(CNT+ARRSFT)
30200             CONTINUE
                  DO 30250 CNT = C1XB1+1, C1XB1+BEADS-1
                    ANG1TP(CNT) = TPA(CNT)
30250             CONTINUE
C
C  Calculate the positions for the new system.
C
                    TPX(END) = XTEMP
                    TPY(END) = YTEMP
                    TPZ(END) = ZTEMP
                    DO 30300 CNT = LOWCT, HIGHCT
                      TPX(CNT) = X1C(CNT+ARRSFT)
                      TPY(CNT) = Y1C(CNT+ARRSFT)
                      TPZ(CNT) = Z1C(CNT+ARRSFT)
30300               CONTINUE
                    DO 30350 CNT = CM1XBD+1, CM1XBD+BEADS
                      X1TP(CNT) = TPX(CNT)
                      Y1TP(CNT) = TPY(CNT)
                      Z1TP(CNT) = TPZ(CNT)
30350               CONTINUE
C
C  ***   End  CHAINS:  All of the chains have been moved.  ***
C
31000             CONTINUE
C
30000         CONTINUE
C
C       Move the chain center of masses so that the values of
C       XCM/(box length), YCM/(box length), ZCM/(box length)
C       remain unchanged for the new box.  The internal confor-
```

```
C       mation of the chain remains the same.  The new positions
C       are put in temporary storage.
C
            DO 41000 C2 = 1, CHAINS
              RATIO = LENTMP/LENGTH - 1.D0
              DELX = XCMTP(C2)*RATIO
              DELY = YCMTP(C2)*RATIO
              DELZ = ZCMTP(C2)*RATIO
C
              XCMTP(C2) = XCMTP(C2) + DELX
              YCMTP(C2) = YCMTP(C2) + DELY
              ZCMTP(C2) = ZCMTP(C2) + DELZ
              DO 42000 B2 = 1, BEADS
                BDCNT = (C2-1)*BEADS + B2
                X1TP(BDCNT) = X1TP(BDCNT) + DELX
                Y1TP(BDCNT) = Y1TP(BDCNT) + DELY
                Z1TP(BDCNT) = Z1TP(BDCNT) + DELZ
42000         CONTINUE
41000       CONTINUE
C
            IF( VOLTMP .GE. VOLUME ) THEN
              IF( CONFGS .GT. UNOBSR ) THEN
                VOLUP = VOLUP + 1
                VOLCNG = VOLCNG + 1
                VOLUFG = 1
              ENDIF
            ELSE
              IF( CONFGS .GT. UNOBSR ) THEN
                VOLDN = VOLDN + 1
                VOLCNG = VOLCNG + 1
                VOLDFG = 1
              ENDIF
            ENDIF
C
C     Compute and write the average distance through which a
C     chain (center-of-mass) moves during the course of this
C     volume loop.
C
            DO 44500 C = 1, CHAINS
              XDIST = XDIST + (XCM(C) - XCMTP(C))**2
              YDIST = YDIST + (YCM(C) - YCMTP(C))**2
              ZDIST = ZDIST + (ZCM(C) - ZCMTP(C))**2
44500       CONTINUE
            DIST = DIST + (XDIST + YDIST + ZDIST)/DBLE(CHAINS)
C
C           Perform the excluded volume test.
C           NB:  The box size for this test is (LENTMP)**3.
C           (Compare with the previous test).
C
            LDIFF = VOLTMP - VOLUME
            print*, 'VOLUME, VOLTMP, diff ='
            print*, VOLUME, VOLTMP, ldiff
```

```
C
C  NOTE:  ***          This is the brute force method.          ***
C        ***  Incorporate the neighbor table method ASAP.      ***
C
            GFLAG = GFLAG + 1
            FLAG = 1
            DO 45000 C = 1, CHAINS-1
              CM1XB = (C-1)*BEADS
              DO 45100 B = 1, BEADS
                J = CM1XB + B
                XTEMP = X1TP(J)
                YTEMP = Y1TP(J)
                ZTEMP = Z1TP(J)
                DO 45200 B2 = 1, BEADS
                  DO 45300 C2 = C+1, CHAINS
                    I = (C2-1)*BEADS + B2
                    XTMP = ( DABS(XTEMP - X1TP(I)) )*RLENTP
                    YTMP = ( DABS(YTEMP - Y1TP(I)) )*RLENTP
                    ZTMP = ( DABS(ZTEMP - Z1TP(I)) )*RLENTP
                    IRX = XTMP + 0.5D+00
                    IRY = YTMP + 0.5D+00
                    IRZ = ZTMP + 0.5D+00
                    BDDIS2 = ( (XTMP - IRX)**2 + (YTMP - IRY)**2
     &                           + (ZTMP - IRZ)**2  )*LENTP2
C
                    IF (BDDIS2 .LT. DIAM2 ) FLAG = 2
45300               CONTINUE
                    IF(FLAG.GT.1) GOTO 46000
45200             CONTINUE
45100           CONTINUE
45000         CONTINUE
C



46000         IF(FLAG .GT. 1 ) THEN
                IF( CONFGS .GT. UNOBSR ) THEN
                  IF(VOLUFG.EQ.1) VLNULU = VLNULU + 1
                  IF(VOLDFG.EQ.1) VLNULD = VLNULD + 1
                ENDIF
                GOTO 40000
              ENDIF
C
              ANGLSM=0.0D0
              DO 43000 CNT = 1, BM1XCH
                ANGLSM = ANGLSM + ANG1TP(CNT)
43000         CONTINUE
              BNDNEW = (0.50D+00 / TAU) * ((BEADS-2)*CHAINS + ANGLSM)
```

```
C
C    Metropolis Test.
C
                HAMNEW = BNDNEW  +  OSMOT * VOLTMP * DBLE(CHAINS)
C               print*, 'HAMNEW, HAMINI =', HAMNEW, HAMINI
                IF (HAMNEW .GT. HAMINI) THEN
                  DELHT = (HAMNEW - HAMINI)
                  IF (DELHT .GE. 170.0D0) THEN
                    ENGCNT = ENGCNT + 1
                    IF( CONFGS .GT. UNOBSR) THEN
                      IF( VOLTMP .GT. VOLUME) THEN
                        FAILUP = FAILUP + 1
                       ELSE
                        FAILDN = FAILDN + 1
                      ENDIF
                    ENDIF
                    GO TO 40000
                  ELSE
                    BOLTZ = DEXP(- DELHT)
                    rantmp = dran(iseed)
                    IF (BOLTZ .LT. rantmp) THEN
                      ENGCNT = ENGCNT + 1
                      IF( CONFGS .GT. UNOBSR) THEN
                        IF( VOLTMP .GT. VOLUME) THEN
                          FAILUP = FAILUP + 1
                         ELSE
                          FAILDN = FAILDN + 1
                        ENDIF
                      ENDIF
                      GO TO 40000
                    ENDIF
                  ENDIF
                ENDIF
C
C     Metropolis test was passed.
C     New positions are acceptable.  Place in permanent storage.
C
                  ACCEPT = ACCEPT + 1
                  MOVED = 0.0D0
                  DO 44000 CNT = 1, 3*BDXCH
                    IF (POS1TP(CNT) .NE. POS1(CNT)) MOVED = MOVED + 1.0D0
44000             CONTINUE
                  IF (MOVED .NE. 0.0D0) ACCMOV = ACCMOV + 1
                  ACCFLG = 1
C
                  BNDINI = BNDNEW
                  HAMINI = HAMNEW
                  ANG1(1:BM1XCH) = ANG1TP(1:BM1XCH)
                  CM1(1: 3*CHAINS) = CM1TP(1: 3*CHAINS)
                  POS1(1:3*BDXCH) = POS1TP(1:3*BDXCH)
                  VEC1(1:3*BM1XCH) = VEC1TP(1:3*BM1XCH)
C
                  ENRAT = DBLE(ACCEPT)/DBLE(LENSTP)
```

```
C
C       Compute the percentage of excluded volume failures thus far.
C
C             VOLRAT = DBLE(VOLCNT)/DBLE(CHAINS*MCLIM*LENSTP*CONFGS)

              IF( CONFGS .GT. UNOBSR) THEN
                IF(VOLTMP.GT.VOLUME ) THEN
                  VFLGUP = VFLGUP + 1
                 ELSE
                  VFLGDN = VFLGDN + 1
                ENDIF
              ENDIF
C
              LENGTH = LENTMP
              VOLUME = VOLTMP
C
              SMVOL2 = SMVOL2 + VOLTMP**2
C
C    ***         This is the end of the  LENSTP loop.          ***
C
40000       CONTINUE
C
C  ***  The Measurement Section.  ***
C
C    The system is given  UNOBSR  steps to come to equilibrium, or at
C  least to shed the initial character of the system, before statistics
C  are accumulated.  The number of samples is  NSAMP, and  SAMPL
C  serves as the counter.
C
            IF (CONFGS .GT. UNOBSR) THEN
              SUCCES = SUCCES + ACCEPT
              SUCMOV = SUCMOV + ACCMOV
              SAMPL = CONFGS - UNOBSR
C
              BNDAVS = BNDAVS + BNDINI
              BNDS2 = BNDS2 + BNDINI**2
              HAMLTN = HAMLTN + HAMINI
              HAMS2 = HAMS2 + HAMINI**2
C
              VOLAVS = VOLAVS + VOLUME
               print*,'volavs, volume=', volavs, volume
              VOLSM2 = VOLSM2 + (VOLUME)**2
C
              DISTAV = DIST/DBLE(LENLIM) + DISTAV
              DSTSM2 = DSTSM2 + (DIST/DBLE(LENLIM))**2
C
```

```
                RNN2 = 0.0D+00
                RGY2 = 0.0D+00
                DO 51000 C = 1, CHAINS
                  CM1XBD = (C - 1)*BEADS
                  I = CM1XBD + BEADS
                  J = CM1XBD + 1
                  RNN2 = RNN2 + (X1(I) - X1(J))**2 + (Y1(I) - Y1(J))**2
     &                        + (Z1(I) - Z1(J))**2
                  DO 51100 B = 1, BEADS
                    K = CM1XBD + B
                    RGY2 = RGY2 + (X1(K) - XCM(C))**2
     &                            + (Y1(K) - YCM(C))**2
     &                            + (Z1(K) - ZCM(C))**2
51100             CONTINUE
51000           CONTINUE
                RNTON2(SAMPL) = RNN2 / DBLE(CHAINS)
                RGYRA2(SAMPL) = RGY2 / DBLE(BDXCH)
C
C           ***  This is the end of the  CONFGS  loop.  ***
C
              ENDIF
C
              PNTRL = DFLOAT(CONFGS-UNOBSR)/DFLOAT(NPRINT)
              PNTIN = DFLOAT( (CONFGS-UNOBSR)/NPRINT )
              IF(CONFGS.GT.UNOBSR .AND. ABS(PNTRL-PNTIN).LT.1.D-08 ) THEN
C
C     The average pseudo-Hamiltonian, (HAMLTN), the average pseudo-
C     Hamiltonian squared (HAMS2), and the average end-to-end distance
C     of the polymer chains (RNTON) over  NSAMP  tries are calculated.
C
                BDAVS = BNDAVS / DBLE(SAMPL)
                BDS2 = BNDS2 / DBLE(SAMPL )
C
                HMLTN = HAMLTN / DBLE(SAMPL )
                HMS2 = HAMS2 / DBLE(SAMPL )
C
                VLAVS = VOLAVS/ DBLE(SAMPL )
                 print*, 'vlavs, sampl= ', vlavs, sampl
                VLSM2 = VOLSM2/ DBLE(SAMPL )
C
                DSTAV = DISTAV/ DBLE(SAMPL )
                DTSM2 = DSTSM2/ DBLE(SAMPL )
C
                RNNAVG = DASUM(NSAMP, RNTON2, 1) / DBLE(SAMPL)
                RGYAVG = DASUM(NSAMP, RGYRA2, 1) / DBLE(SAMPL)
C
C         The value of the variance in the volume and energies are
C         computed, and the Flory exponent (STIFF) is computed.
C
                HAMVAR = HMS2 - HMLTN**2
                BNDVAR = BDS2 - BDAVS**2
                VOLVAR = VLSM2 - VLAVS**2
                DISVAR = DTSM2 - DSTAV**2
                STIFF  = DLOG(RNNAVG * RBON2) / DLOG(DBLE(BEADS - 1))
```

```
C
                IF(SUCCES.GT.0) THEN
                  fracup = dble(vflgup)/dble(succes )
                  fracdn = dble(vflgdn)/dble(succes )
                ENDIF
C
C     The next calculation gives the fraction of attempts to change
C   the PERMANENT configuration of the system that were successful.
C   This value (SUCCES) only depends on the results of the energy test.
C
C   NOTE 1:  The denominator is obtained from the maximum values for
C            the counters of the loops involved:
C
C                                CONFGS * LENLIM.
C
                SCCES = SUCCES / DBLE(SAMPL * LENLIM )
                SCMOV = SUCMOV / DBLE(SAMPL * LENLIM )
C
C
C
C       Next, the percentage of excluded volume failures is calculated,
C       along with the percentage of failures due to self-interaction
C       of the chains.
C
                DENOM = DBLE(CHAINS)*DBLE(MCLIM)*DBLE(LENLIM)*DBLE(CONFGS)
                VOLRAT = DBLE(VOLCNT) / DENOM
                SELRAT = DBLE(SELCNT) / DENOM
                DENOM2 = dble(lenlim)*dble(SAMPL)
                chgrad = dble(VOLDN)/DENOM2
                chgrau = dble(VOLUP )/DENOM2
                IF(VOLDN.GT.0) chgnd = dble(VLNULD)/(dble(VOLDN))
                IF(VOLUP.GT.0) chgnu = dble(VLNULU)/(dble(VOLUP))
C
C              ******************  DONE  ******************
C
C       PRINT  statements for the run statistics.
C
                print*, '   '
                print*, '   '
                print*, '   '
                print*, 'Run statistics up to SAMPL =',SAMPL
                print*, ' '
                print*, 'Initial parameters.'
                print*, 'BEADS, CHAINS =', BEADS, CHAINS
                print*, 'BDVOL, BONDL =' , BDVOL, BONDL
                print*, 'OSMOT, TAU  = ', OSMOT, TAU
                print*, 'VDIFMX =', VDIFMX
                print*, 'Initial value of ISEED =', INAUGH
                print*, ' '
                print*, 'Loop parameters'
                print*, 'EQSTEP =', EQSTEP
                print*, 'MCLIM, LENLIM, =', MCLIM, LENLIM
                print*, 'NSAMP, UNOBSR =', NSAMP, UNOBSR
```

```
               print*, '  '
               print*, 'Results'
               print*, 'VOLAVS, VOLVAR =', VLAVS, VOLVAR
               print*, 'HAMLTN, HAMVAR =', HMLTN, HAMVAR
               print*, 'BNDAVS, BNDVAR =', BDAVS, BNDVAR
               print*, 'End-to-end and radius of gyration =',
     &                  RNNAVG, RGYAVG
               print*, 'Flory exponent =', STIFF
               print*, 'DISTAV, DISVAR =', DSTAV, DISVAR
               print*, 'SELRAT, VOLRAT =', SELRAT, VOLRAT
               print*, 'Energy success ratio =', SCCES
               print*, 'ACCEPTED: FRACDN =',  fracdn
               print*, 'ACCEPTED: FRACUP =',  fracup
               print*, 'OVERALL: fraction vol decreases =', chgrad
               print*, 'OVERALL: fraction vol increases =', chgrau
               print*, 'fraction vol decreases w/ exc. vol. failure =',
     &                 chgnd
               print*, 'fraction vol increases w/ exc. vol. failure =',
     &                 chgnu
               print*, 'FAILURES: number with vol decreased =', FAILDN
               print*, 'FAILURES: number with vol increased =', FAILUP
C-------------------------------------------------------------------03E
C
               print*, ' '
               print*, ' '
               print*, ' '
C
                 ENDIF
C
C              ***  This is the end of the  CONFGS  loop.  ***
C
50000          CONTINUE
C
C     The average pseudo-Hamiltonian, (HAMLTN), the average pseudo-
C   Hamiltonian squared (HAMS2), and the average end-to-end distance
C   of the polymer chains (RNTON) over  NSAMP  tries are calculated.
C
               BNDAVS = BNDAVS / DBLE(SAMPL)
               BNDS2 = BNDS2 / DBLE(SAMPL)
C
               HAMLTN = HAMLTN / DBLE(SAMPL)
               HAMS2 = HAMS2 / DBLE(SAMPL)
C
               VOLAVS = VOLAVS/ DBLE(SAMPL)
               VOLSM2 = VOLSM2/ DBLE(SAMPL)
C
               DISTAV = DISTAV/ DBLE(SAMPL)
               DSTSM2 = DSTSM2/ DBLE(SAMPL)
C
               RNNAVG = DASUM(NSAMP, RNTON2, 1) / DBLE(SAMPL)
               RGYAVG = DASUM(NSAMP, RGYRA2, 1) / DBLE(SAMPL)
```

```
C
C       The value of the variance in the volume and energies are
C       computed, and the Flory exponent (STIFF) is computed.
C
        HAMVAR = HAMS2 - HAMLTN**2
        BNDVAR = BNDS2 - BNDAVS**2
        VOLVAR = VOLSM2 - VOLAVS**2
        DISVAR = DSTSM2 - DISTAV**2
        STIFF  = DLOG(RNNAVG * RBON2) / DLOG(DBLE(BEADS - 1))
C
        fracup = dble(vflgup)/dble(succes )
        fracdn = dble(vflgdn)/dble(succes )
C
C   The next calculation gives the fraction of attempts to change
C  the PERMANENT configuration of the system that were successful.
C  This value (SUCCES) only depends on the results of the energy test.
C
C  NOTE 1:  The denominator is obtained from the maximum values for
C           the counters of the loops involved:
C
C                              NSAMP * LENLIM.
C
        SUCCES = SUCCES / DBLE(SAMPL * LENLIM )
        SUCMOV = SUCMOV / DBLE(SAMPL * LENLIM )
C
C
C
C       Next, the percentage of excluded volume failures is calculated,
C       along with the percentage of failures due to self-interaction
C       of the chains.
C
        DENOM =  DBLE(CHAINS)*DBLE(MCLIM)*DBLE(LENLIM)*DBLE(NSAMP)
        VOLRAT = DBLE(VOLCNT) / DENOM
        SELRAT = DBLE(SELCNT) / DENOM
        DENOM2 = dble(lenlim)*dble(nsamp)
        chgrad = dble(VOLDN)/DENOM2
        chgrau = dble(VOLUP )/DENOM2
        chgnd = dble(VLNULD)/( dble(VOLDN)   )
        chgnu = dble(VLNULU)/( dble(VOLUP)   )
C
C            ******************  DONE  ******************
C       print statements for the run statistics.
C
        print*, 'Run statistics.'
        print*, ' '
        print*, 'Initial parameters.'
        print*, 'BEADS, CHAINS =', BEADS, CHAINS
        print*, 'BDVOL, BONDL =' , BDVOL, BONDL
        print*, 'OSMOT, TAU  = ', OSMOT, TAU
        print*, 'VDIFMX =', VDIFMX
        print*, 'Initial value of ISEED =', INAUGH
```

```
      print*, ' '
      print*, 'Loop parameters'
      print*, 'EQSTEP =', EQSTEP
      print*, 'MCLIM, LENLIM, =', MCLIM, LENLIM
      print*, 'NSAMP, UNOBSR =', NSAMP, UNOBSR
      print*, '  '
      print*, 'Results'
      print*, 'VOLAVS, VOLVAR =', VOLAVS, VOLVAR
      print*, 'HAMLTN, HAMVAR =', HAMLTN, HAMVAR
      print*, 'BNDAVS, BNDVAR =', BNDAVS, BNDVAR
      print*, 'End-to-end and radius of gyration =', RNNAVG, RGYAVG
      print*, 'Flory exponent =', STIFF
      print*, 'DISTAV, DISVAR =', DISTAV, DISVAR
      print*, 'SELRAT, VOLRAT =', SELRAT, VOLRAT
      print*, 'Energy success ratio =', SUCCES
      print*, 'ACCEPTED: FRACDN =',  fracdn
      print*, 'ACCEPTED: FRACUP =',  fracup
      print*, 'OVERALL: fraction vol decreases =', chgrad
      print*, 'OVERALL: fraction vol increases =', chgrau
      print*, 'fraction vol decreases w/ exc. vol. failure =', chgnd
      print*, 'fraction vol increases w/ exc. vol. failure =', chgnu
      print*, 'FAILURES: number with vol decreased =', FAILDN
      print*, 'FAILURES: number with vol increased =', FAILUP
C-----------------------------------------------------------------03E

      ELSPTL = DTIME(TARR1)
      print*, 'Timing for the running of the system:', ELSPTL
C
      STOP 'UP TO 2ND PRINTING SECTION.'
      END
C
C
C
      SUBROUTINE ST_REP(ALPHA, BDXCH, BEADS, BM1XCH, CHAINS,
     &                  BONDL, DIAM, DIAM2, PI,
     &                   RBON2,
     &                  ISEED,
     &                  HAMINI, OSMOT, TAUTMP,
     &                  BDPERC,
     &                  ANG1, X1, Y1, Z1, XCM, YCM, ZCM,
     &                  XCMTP, YCMTP, ZCMTP,
     &                  XVEC1, YVEC1, ZVEC1)
C
C    Program to generate the initial configuration of a set of
C  CHAINS  linear polymers, each consisting of BEADS particles.  The
C  method is a simple self-avoiding walk mechanism.
C
C  Last modified 10/25/88.  Scott K. Starry .  Removed the box
C  constraints in the z direction. 11/4/88, dmj.
C
      IMPLICIT NONE
C
C    Parameters passed.
C
```

```
      INTEGER   BDXCH, BEADS, BM1XCH, CHAINS
      REAL*8    ALPHA, BONDL, DIAM, DIAM2, OSMOT, PI, RBON2
C
CC    Scalar variables passed.
C
      INTEGER   ISEED
      REAL*8    HAMINI, TAUTMP
C
CC    Array variables passed.
C
      INTEGER   BDPERC(CHAINS)
      REAL*8    ANG1(BM1XCH),
     &           X1(BDXCH), Y1(BDXCH), Z1(BDXCH),
     &           XCM(CHAINS), YCM(CHAINS), ZCM(CHAINS),
     &           XCMTP(CHAINS), YCMTP(CHAINS), ZCMTP(CHAINS),
     &           XVEC1(BM1XCH), YVEC1(BM1XCH), ZVEC1(BM1XCH)
C
CC    Local scalar variables.
C
c     integer   ierr
      INTEGER   B, B1, C, C1, C1M1XB, C1XB1, CM1XBD, CNT,
     &           I, J
      REAL*8    ANGLSM, BNDNEW, CSTHET, DIST2,
     &           PHI, SNTHET,
     &           XTMP, YTMP, ZTMP
C
C     Declarations of Functions.
C
      REAL*8    DRAN
      EXTERNAL  DRAN
C
C
CC    Generate the initial beads of each polymer.
CC    This is done by systematically laying down the first bead in
CC    successsive "cells" that are  ALPHA  times  DIAM  in width.
CC    XTMP, YTMP, ZTMP  are the coordinates of the first bead, at the
CC    center of a given cell.  The excluded volume test is kept in
CC    as a check, although it should not be necessary.
C

      XTMP = -DIAM*alpha/2.d0
      YTMP =  DIAM*alpha/2.d0
      ZTMP =  DIAM*alpha/2.d0

      DO 1000 C = 1, CHAINS
        I = (C-1)*BEADS + 1
1010    XTMP = XTMP + ALPHA*DIAM
        IF( XTMP .GT. 1.D0 - 0.5*DIAM ) THEN
           XTMP = ALPHA*DIAM*0.5
           YTMP = YTMP + DIAM*ALPHA
        ENDIF
```

```
            IF( YTMP .GT. 1.D0 - 0.5*DIAM ) THEN
               YTMP = ALPHA*DIAM*0.5
               ZTMP = ZTMP + DIAM*ALPHA
            ENDIF
            IF( ZTMP .GT. 1.D0 - 0.5*DIAM ) THEN
               STOP 'COULD NOT FIT ALL INITIAL BEADS.  LOWER ALPHA.'
            ENDIF
C
C   Perform the "excluded volume test" at this stage by checking
C to make sure that the last bead position generated is not within
C DIAM  of any other bead generated previously.
C
            DO 1100 C1 = 1, C-1
              J = (C1 - 1)*BEADS + 1
              DIST2 =  (XTMP - X1(J))**2 + (YTMP - Y1(J))**2
     &                + (ZTMP - Z1(J))**2
              IF (DIST2 .LE. DIAM2) GO TO 1010
 1100       CONTINUE
            X1(I) = XTMP
            Y1(I) = YTMP
            Z1(I) = ZTMP
 1000     CONTINUE
C
C    The initial beads of each polymer have now been generated
C  in the box.
C
          DO 2000 C = 1, CHAINS
            I = (C - 1)*BEADS + 1
            XCM(C) = X1(I)
            YCM(C) = Y1(I)
            ZCM(C) = Z1(I)
            XCMTP(C) = XCM(C)
            YCMTP(C) = YCM(C)
            ZCMTP(C) = ZCM(C)
 2000     CONTINUE
C
C    Next, the second bead of each polymer is generated.  The second
C  bead is added to each polymer before proceeding to the next step.
C
          DO 3000 C = 1, CHAINS
            I = (C - 1)*BEADS + 1
C
C    Randomly choose the orientation of the second bead relative
C  to the first.  The next two statements return  PHI  and the cosine
C  of THETA, CSTHET.  (THETA  is the azimuthal angle; PHI  is the
C  polar angle.)
C
 3010       PHI = 2.0D+00 * PI * DRAN(ISEED)
            CSTHET = 2.0D+00*DRAN(ISEED) - 1.0D+00
            SNTHET = DSQRT(1.0D+00 - CSTHET**2)
```

```
C
C     Compute the coordinates for the second bead in chain C, and
C   store these results in  XTMP, YTMP, ZTMP.
C
            XTMP = X1(I) + BONDL*SNTHET*DSIN(PHI)
            YTMP = Y1(I) + BONDL*SNTHET*DCOS(PHI)
            ZTMP = Z1(I) + BONDL*CSTHET
C
            XCMTP(C) = (X1(I) + XTMP)/2.d0
            YCMTP(C) = (Y1(I) + YTMP)/2.d0
            ZCMTP(C) = (Z1(I) + ZTMP)/2.d0
C
C     Perform the excluded volume test, comparing bead (2,C)
C   to the positions of all other beads generated so far.
C
C   NOTE:  No test needs to be performed between bead 1 and bead 2 of
C          chain C here as the second bead was just laid down with
C          respect to the first bead.
C
C   NOTE:  There should not be any trouble laying down the second
C          bead due to the expansion of the box.  Thus, returning
C          to Label 3010 without a trap should be sufficient in the
C          unlikely event of a failure of the excluded volume test.
C
            DO 3100 C1 = 1, C-1
              C1M1XB = (C1 - 1)*BEADS
              DO 3110 B = 1, 2
                J = C1M1XB + B
                DIST2 =  (XTMP - X1(J) - DNINT(XTMP - X1(J)))**2
     &                 + (YTMP - Y1(J) - DNINT(YTMP - Y1(J)))**2
     &                 + (ZTMP - Z1(J) - DNINT(ZTMP - Z1(J)))**2
                IF (DIST2 .LE. DIAM2) GO TO 3010
 3110         CONTINUE
 3100       CONTINUE
C
C       If the excluded volume test is passed, then accept the
C   generated values of the second bead in polymer C.
C
            X1(I+1) = XTMP
            Y1(I+1) = YTMP
            Z1(I+1) = ZTMP
            XCM(C) = XCMTP(C)
            YCM(C) = YCMTP(C)
            ZCM(C) = ZCMTP(C)
 3000     CONTINUE
C
C     At this point, all of the chains have two beads, and the values
C   of BDPERC(C) must all be set to 2.
C
          DO 4000 C = 1, CHAINS
            BDPERC(C) = 2
 4000     CONTINUE
```

```
C
C     Next, further beads are added to each chain.  This is done until
C   all of the beads for a single chain have been placed.  Then, the
C   beads for the next chain are laid down.  Since not all chains have
C   the same length, the array  BDPERC  is used to handle the excluded
C   volume tests.
C
          DO 5000 C = 1, CHAINS
            CM1XBD = (C - 1)*BEADS
            DO 5100 B = 3, BEADS
              I = CM1XBD + B
C
C     Attempt to propagate a new bead.
C
C     Generate the coordinates of the new bead.
C
 5010         PHI = 2.0D+00 * PI * DRAN(ISEED)
              CSTHET = 2.0D+00*DRAN(ISEED) - 1.0D+00
              SNTHET = DSQRT(1.0D+00 - CSTHET**2)
C
              XTMP = X1(I-1) + BONDL*SNTHET*DCOS(PHI)
              YTMP = Y1(I-1) + BONDL*SNTHET*DSIN(PHI)
              ZTMP = Z1(I-1) + BONDL*CSTHET
C
C     Compute the temporary center-of-mass of the chain I.
C
              XCMTP(C) = ( XCM(C)*BDPERC(C) + XTMP )/
     &                   ( DBLE( BDPERC(C) + 1) )
              YCMTP(C) = ( YCM(C)*BDPERC(C) + YTMP )/
     &                   ( DBLE( BDPERC(C) + 1) )
              ZCMTP(C) = ( ZCM(C)*BDPERC(C) + ZTMP )/
     &                   ( DBLE( BDPERC(C) + 1) )
C
C     Perform the excluded volume test for the new bead.
C
C   NOTE:  In this test, even beads within the chain  C  must
C          be tested.
C
              DO 5110 C1 = 1, CHAINS
                C1M1XB = (C1 - 1)*BEADS
                DO 5111 B1 = 1, BDPERC(C1)
                  J = C1M1XB + B1
                  IF ( .NOT. ((C1 .EQ. C) .AND. (B1 .EQ. B)) ) THEN
                    DIST2 =   (XTMP - X1(J) - DNINT(XTMP - X1(J)))**2
     &                      + (YTMP - Y1(J) - DNINT(YTMP - Y1(J)))**2
     &                      + (ZTMP - Z1(J) - DNINT(ZTMP - Z1(J)))**2
C
C    If the excluded volume test fails, find a new position for
C   the bead being placed.
C
C   NOTE:  There is presently no trap here for chains that are caught
C          in a tight spot.  This is because Ho believes that in
C          three dimensions there is always space to place the next
C          monomer.  This must be tested at high concentrations
```

```
C         to see if the time for creating the initial configuration
C         is acceptable.
C
                IF (DIST2 .LE. DIAM2) GO TO 5010
              ENDIF
 5111       CONTINUE
 5110     CONTINUE
C
C    If the excluded volume test is passed, store the new bead
C  coordinates in X, Y, and Z.
C
            X1(I) = XTMP
            Y1(I) = YTMP
            Z1(I) = ZTMP
            BDPERC(C) = BDPERC(C) + 1
C
 5100     CONTINUE
          XCM(C) = XCMTP(C)
          YCM(C) = YCMTP(C)
          ZCM(C) = ZCMTP(C)
 5000   CONTINUE
C
C    Loops 6000 to 8000 calculate the bond vectors, centers of mass,
C  and bond angles for the chains.
C
        DO 6000 C = 1, CHAINS
          C1XB1 = (C - 1)*(BEADS - 1)
          CM1XBD = (C - 1)*BEADS
          DO 6100 B = 1, BEADS-1
            XVEC1(C1XB1+B) = X1(CM1XBD+B+1) - X1(CM1XBD+B)
            YVEC1(C1XB1+B) = Y1(CM1XBD+B+1) - Y1(CM1XBD+B)
            ZVEC1(C1XB1+B) = Z1(CM1XBD+B+1) - Z1(CM1XBD+B)
 6100     CONTINUE
 6000   CONTINUE
C
        DO 8000 C = 1, CHAINS
          C1XB1  = (C - 1)*(BEADS - 1)
          ANG1(C1XB1+1) = 0.0D0
          DO 8100 B = C1XB1+2, C1XB1+(BEADS-1)
            ANG1(B) = -RBON2 * (    XVEC1(B-1) * XVEC1(B)
     &                          +  YVEC1(B-1) * YVEC1(B)
     &                          +  ZVEC1(B-1) * ZVEC1(B) )
 8100     CONTINUE
 8000   CONTINUE
C
        ANGLSM = 0.0D+00
        DO 10500 CNT = 1, BM1XCH
          ANGLSM = ANGLSM + ANG1(CNT)
10500   CONTINUE
        BNDNEW = (0.50D0 / TAUTMP) * ((BEADS-2)*CHAINS + ANGLSM)
C
        HAMINI = BNDNEW + OSMOT * DBLE(CHAINS)
```

```
C
C     Output for the initial configuration.
C
C        OPEN (UNIT=10, ACCESS='SEQUENTIAL', STATUS='NEW',
C      &       FORM='UNFORMATTED', IOSTAT=IERR)
C        IF (IERR .NE. 0) STOP ' OPEN to UNIT 10 failed.'
C         OPEN (UNIT=15, ACCESS='SEQUENTIAL', STATUS='UNKNOWN',
C      &       FORM='FORMATTED', IOSTAT=IERR)
C         IF (IERR .NE. 0) STOP ' OPEN to UNIT 15 failed.'
C
C        WRITE(10) BEADS, BONDL, CHAINS, DIAM
C        WRITE(15,9910) BEADS, BONDL, CHAINS, DIAM
C         DO 9000 CNT = 1, BDXCH
C          DO 9100 B = 1, BEADS
C            WRITE(10) X(B,C), Y(B,C), Z(B,C)
C             WRITE(15,9920) X1(CNT), Y1(CNT), Z1(CNT)
C            WRITE(15,9920) X(B,C), Y(B,C), Z(B,C)
C 9100     CONTINUE
C 9000    CONTINUE
C 9910   FORMAT (3X, I4, 3X, 1PD13.6, 3X, I4, 1PD13.6, /)
C 9920    FORMAT (3(3X, 1PD13.6))
C
C        CLOSE (UNIT=10)
C        CLOSE (UNIT=15)
C
         RETURN
         END
C
C
C
         SUBROUTINE EQ_REP (BDXCH, BDVOL, BEADS, BM1XCH, CHAINS,
      &                     EQSTEP, MCLIM,
      &                     BONDL, DIAM2, RBON2, PI,
      &                     ISEED,
      &                     HAMINI, OSMOT, TAUTMP, TRANMX,
      &                     ANG1, ANG1TP, CM1, CM1TP,
      &                     POS1, POS1TP,
      &                     TPA, TPVX, TPVY, TPVZ, TPX, TPY, TPZ,
      &                     VEC1, VEC1TP,
      &                     X1TP, Y1TP, Z1TP, X1C, Y1C, Z1C,
      &                     XCMTP, YCMTP, ZCMTP,
      &                     XVC1TP, YVC1TP, ZVC1TP)
C
C        Last modified 09/29/88.  Scott K. Starry
C
         IMPLICIT NONE
C
C     Parameters passed.
C
         INTEGER   BDXCH, BEADS, BM1XCH, CHAINS, EQSTEP, MCLIM
         REAL*8    BDVOL, BONDL, DIAM2, OSMOT, PI, RBON2,
      &            TRANMX
```

```
C
CC    Scalar variables passed.
C
      INTEGER   ISEED
      REAL*8    HAMINI, TAUTMP
C
CC    Array variables passed.
C
      REAL*8    ANG1(BM1XCH), ANG1TP(BM1XCH),
     &           CM1(3*CHAINS), CM1TP(3*CHAINS),
     &           POS1(3*BDXCH), POS1TP(3*BDXCH)
      REAL*8    TPA(BM1XCH), TPVX(BM1XCH), TPVY(BM1XCH),
     &           TPVZ(BM1XCH), TPX(BDXCH), TPY(BDXCH),
     &           TPZ(BDXCH), VEC1(3*BM1XCH), VEC1TP(3*BM1XCH)
      REAL*8     X1C(BDXCH), Y1C(BDXCH), Z1C(BDXCH),
     &           X1TP(BDXCH), Y1TP(BDXCH), Z1TP(BDXCH),
     &           XCMTP(CHAINS), YCMTP(CHAINS), ZCMTP(CHAINS),
     &           XVC1TP(BM1XCH), YVC1TP(BM1XCH), ZVC1TP(BM1XCH)
C
CC    Local scalar variables.
C
      INTEGER   ACCEPT, ACCFLG, ACCMOV, ARRSFT, ATMPTS,
     &           B, B2, BEADLW, BEADHI, C, C1, C1XB1, C2,
     &           CM1XBD, CNT,
     &           END, ENDMV, END_AV, ENGCNT, EQUIL, FLAG,
     &           HIGHCT, HI_AV, I, IRX, IRY, IRZ, J,
     &           LOWCT, LOW_AV, MCSTEP, PENULT, PEN_AV, SELFG, VOLCNT
      REAL*8    ANGLSM, BDDIS2, BNDNEW, BNDINI, BOLTZ, COTHE,
     &           CXTEMP, CYTEMP, CZTEMP,
     &           DELHT, GAMMA, HAMNEW, MOVED,
     &           RX, RY, RZ, RXNEW, RYNEW, RZNEW,
     &           SITHE, SUCCES, SUCMOV, XCMC, YCMC, ZCMC,
     &           XTEMP, YTEMP, ZTEMP, XTEST, YTEST, ZTEST,
     &           XTMP, YTMP, ZTMP,
     &           XTRANS, YTRANS, ZTRANS
C
CC    Declarations of functions.
C
      EXTERNAL  DRAN
      REAL*8    DRAN
C
      ATMPTS = 0
      SUCCES = 0.0D0
      SUCMOV = 0.0D0
      DO 40000 EQUIL = 1, EQSTEP
        ACCEPT = 0
        ACCFLG = 0
        ACCMOV = 0
        DO 30000 MCSTEP = 1, MCLIM
C
CC    Transfer the information to the working arrays.
C
```

```
                IF (ACCFLG .EQ. 0) THEN
                  ANG1TP(1:BM1XCH) = ANG1(1:BM1XCH)
                  CM1TP(1: 3*CHAINS) = CM1(1: 3*CHAINS)
                  POS1TP(1:3*BDXCH) = POS1(1:3*BDXCH)
                  VEC1TP(1:3*BM1XCH) = VEC1(1:3*BM1XCH)
                ENDIF
                ACCFLG = 0
                DO 31000 C1 = 1, CHAINS
C
                   ATMPTS = ATMPTS + 1
                   C = CHAINS*DRAN(ISEED) + 1
                   CM1XBD = (C - 1) * BEADS
                   C1XB1  = (C - 1) * (BEADS - 1)
                   ENDMV = 2*DRAN(ISEED) + 1
C
C    Translate chain C through a randomly chosen vector.
C    X1C, Y1C, and Z1C are arrays for the temporary storage
C    of the positions of chain C.  These arrays are taken to
C    be of dimension BEADS*CHAINS for convenience.
C
                   XTRANS = TRANMX * ( 2.D0*DRAN(ISEED) - 1.D0 )
                   YTRANS = TRANMX * ( 2.D0*DRAN(ISEED) - 1.D0 )
                   ZTRANS = TRANMX * ( 2.D0*DRAN(ISEED) - 1.D0 )
                   DO 35000 B2 = 1, BEADS
                     I = CM1XBD + B2
                     X1C(I) = X1TP(I) + XTRANS
                     Y1C(I) = Y1TP(I) + YTRANS
                     Z1C(I) = Z1TP(I) + ZTRANS
35000              CONTINUE
                   XCMC = XCMTP(C) + XTRANS
                   YCMC = YCMTP(C) + YTRANS
                   ZCMC = ZCMTP(C) + ZTRANS
C
C     For ENDMV=1, the vector  A  is  -VEC(2,C)  and the vector  R
C  is  -VEC(1,C).  For  ENDMV=BEADS, vector  A  is  VEC(BEADS-2,C)  and
C  vector  R  is  VEC(BEADS-1,C).
C
                IF (ENDMV .EQ. 1) THEN
                    ENDMV = 1
                    END = CM1XBD + 1
                    PENULT = END + 1
                    LOWCT  = END + 1
                    HIGHCT = CM1XBD + BEADS
                    END_AV = C1XB1 + 1
                    PEN_AV = END_AV + 1
                    LOW_AV = PEN_AV
                    HI_AV  = C1XB1 + (BEADS - 1)
                    ARRSFT = -1
                    BEADLW = 0
                    BEADHI = BEADS-1

                    RX = -XVC1TP(END_AV)
                    RY = -YVC1TP(END_AV)
                    RZ = -ZVC1TP(END_AV)
```

```
                  ELSE
                    ENDMV = BEADS
                    END = CM1XBD + BEADS
                    PENULT = END - 1
                    LOWCT  = CM1XBD + 1
                    HIGHCT = END - 1
                    END_AV = C1XB1 + (BEADS - 1)
                    PEN_AV = END_AV
                    LOW_AV = C1XB1 + 1
                    HI_AV  = END_AV - 1
                    ARRSFT = 1
                    BEADLW = 2
                    BEADHI = BEADS+1

                    RX = XVC1TP(PEN_AV)
                    RY = YVC1TP(PEN_AV)
                    RZ = ZVC1TP(PEN_AV)
                ENDIF

C
C     Compute the cosine and sine of the rotation angles GAMMA and
C   THETA.  GAMMA varies from -PI to PI.  THETA  varies from 0 to PI.
C
                GAMMA = PI * (2.0D+00*DRAN(ISEED) - 1.0D+00)
                COTHE = 2.0D0*DRAN(ISEED) - 1.0D0
                SITHE = DSQRT(1.0D0 - COTHE**2)

C     Calculate the new coordinates for the end bead.  This amounts
C   to applying a vector decomposition as if the vector formed by
C   the present positions of the penultimate and end beads lies along
C   the laboratory Z-axis.  This really has no bearing on the result
C   as long as the new vector has a length equal to  BONDL  which
C   is assured by this method.
C
                XTEMP = X1C(END) + BONDL*SITHE*DCOS(GAMMA)
                YTEMP = Y1C(END) + BONDL*SITHE*DSIN(GAMMA)
                ZTEMP = Z1C(END) + BONDL*COTHE
C
C       Perform the excluded volume test for the moved chain.
C       NB:  The box size here is (1.00d0)**3
C
C   NOTE:  ***        This is the brute force method.        ***
C          ***  Incorporate the neighbor table method ASAP.  ***
C
                FLAG = 1
                SELFG = 1

C       First, check for failures due to chain C interacting
C       with itself.  This happens often, and is relatively
C       efficient to check.
C
                C2 = C
                DO 31101 B2 = 2, BEADS-1
```

```
C
C       XTEMP, YTEMP, ZTEMP give the coordinates of the
C       beads being checked against.
C
                  I = (C2-1)*BEADS + B2
                  XTMP = ( DABS(XTEMP - X1C(I)) )
                  YTMP = ( DABS(YTEMP - Y1C(I)) )
                  ZTMP = ( DABS(ZTEMP - Z1C(I)) )
                  IRX = XTMP + 0.5D+00
                  IRY = YTMP + 0.5D+00
                  IRZ = ZTMP + 0.5D+00
                  BDDIS2 = ( (XTMP - IRX)**2 + (YTMP - IRY)**2
     &                     + (ZTMP - IRZ)**2  )
                  IF (BDDIS2 .LT. DIAM2) THEN
                    FLAG = 2
                    SELFG = 2
                  ENDIF
31101           CONTINUE
C
31115           IF(FLAG.GT.1) GOTO 32000
C
C       Two different versions of the excluded volume check are
C       included depending on whether TRANMX = 0.d0.  If this parameter
C       is zero, then there is no translational movement of the
C       chain, and it is satisfactory to check against the end
C       bead of chain C only.  Otherwise, all of the beads in
C       C must be checked for excluded volume.
C
                IF( TRANMX .EQ. 0.D0 ) THEN
                  DO 31100 B2 = 1, BEADS
                    DO 31110 C2 = 1, C-1
C
C       xtest,ytest,ztest  give the coordinates of the
C       bead being checked against.
C
                      I = (C2-1)*BEADS + B2
                      XTMP = ( DABS(XTEMP - X1TP(I)) )
                      YTMP = ( DABS(YTEMP - Y1TP(I)) )
                      ZTMP = ( DABS(ZTEMP - Z1TP(I)) )
                      IRX = XTMP + 0.5D+00
                      IRY = YTMP + 0.5D+00
                      IRZ = ZTMP + 0.5D+00
                      BDDIS2 = ( (XTMP - IRX)**2 + (YTMP - IRY)**2
     &                       + (ZTMP - IRZ)**2  )
                      IF (BDDIS2 .LT. DIAM2) FLAG = 2
31110               CONTINUE
                    IF( FLAG .GT. 1) GO TO 32000
C
                    DO 31111 C2 = C+1, CHAINS
C
C       xtest, ytest, ztest  give the coordinates of the
C       beads being checked against.
C
```

```
                I = (C2-1)*BEADS + B2
                XTMP = ( DABS(XTEMP - X1TP(I)) )
                YTMP = ( DABS(YTEMP - Y1TP(I)) )
                ZTMP = ( DABS(ZTEMP - Z1TP(I)) )
                IRX = XTMP + 0.5D+00
                IRY = YTMP + 0.5D+00
                IRZ = ZTMP + 0.5D+00
                BDDIS2 = ( (XTMP - IRX)**2 + (YTMP - IRY)**2
     &                 + (ZTMP - IRZ)**2  )
                IF (BDDIS2 .LT. DIAM2) FLAG = 2
31111         CONTINUE
              IF( FLAG .GT. 1) GO TO 32000

C
31100       CONTINUE
            ELSE
             DO 31300 B = BEADLW, BEADHI
               IF( B.EQ.0 .OR. B.EQ.(BEADS+1) )  THEN
                 XTEST = XTEMP
                 YTEST = YTEMP
                 ZTEST = ZTEMP
               ELSE
                 J = (C-1)*BEADS + B
                 XTEST = X1C(J)
                 YTEST = Y1C(J)
                 ZTEST = Z1C(J)
               ENDIF
               DO 31200 B2 = 1, BEADS
                 DO 31210 C2 = 1, C-1
C
CC      xtest,ytest,ztest  give the coordinates of the
C       bead being checked against.
C
                   I = (C2-1)*BEADS + B2
                   XTMP = ( DABS(XTEST - X1TP(I)) )
                   YTMP = ( DABS(YTEST - Y1TP(I)) )
                   ZTMP = ( DABS(ZTEST - Z1TP(I)) )
                   IRX = XTMP + 0.5D+00
                   IRY = YTMP + 0.5D+00
                   IRZ = ZTMP + 0.5D+00
                   BDDIS2 = ( (XTMP - IRX)**2 + (YTMP - IRY)**2
     &                    + (ZTMP - IRZ)**2  )
                   IF (BDDIS2 .LT. DIAM2) FLAG = 2
31210            CONTINUE
                 IF( FLAG .GT. 1) GO TO 32000
C
                 DO 31211 C2 = C+1, CHAINS
                   I = (C2-1)*BEADS + B2
                   XTMP = ( DABS(XTEST - X1TP(I)) )
                   YTMP = ( DABS(YTEST - Y1TP(I)) )
                   ZTMP = ( DABS(ZTEST - Z1TP(I)) )
                   IRX = XTMP + 0.5D+00
                   IRY = YTMP + 0.5D+00
                   IRZ = ZTMP + 0.5D+00
```

```
     &                    BDDIS2 = ( (XTMP - IRX)**2 + (YTMP - IRY)**2
                                + (ZTMP - IRZ)**2 )
                          IF (BDDIS2 .LT. DIAM2) FLAG = 2
31211                   CONTINUE
                        IF( FLAG .GT. 1) GO TO 32000
31200                 CONTINUE
31300               CONTINUE
                  ENDIF
C
32000             IF (FLAG .NE. 1) THEN
                    VOLCNT = VOLCNT + 1
                    GO TO 31000
                  ENDIF
C
C  ZTP(B,C)  equal to  XTEMP, YTEMP, ZTEMP, respectively.
C
C    Calculate the new end vector and angle.
C
                  RXNEW = XTEMP - X1C(END)
                  RYNEW = YTEMP - Y1C(END)
                  RZNEW = ZTEMP - Z1C(END)
C
C    Compute the new vectors VEC(1,C)  or  VEC(BEADS-1,C),
C  as necessary, and the new center of mass for the chain.
C
                  IF (ENDMV .EQ. 1) THEN
                      I = END + (BEADS - 1)
                      XCMTP(C) = XCMC
     &                         + (XTEMP - X1C(I)) / DBLE(BEADS)
                      YCMTP(C) = YCMC
     &                         + (YTEMP - Y1C(I)) / DBLE(BEADS)
                      ZCMTP(C) = ZCMC
     &                         + (ZTEMP - Z1C(I)) / DBLE(BEADS)
C
                      TPVX(END_AV) = -RXNEW
                      TPVY(END_AV) = -RYNEW
                      TPVZ(END_AV) = -RZNEW
                    ELSE
                      I = END - (BEADS - 1)
                      XCMTP(C) = XCMC
     &                         + (XTEMP - X1C(I)) / DBLE(BEADS)
                      YCMTP(C) = YCMC
     &                         + (YTEMP - Y1C(I)) / DBLE(BEADS)
                      ZCMTP(C) = ZCMC
     &                         + (ZTEMP - Z1C(I)) / DBLE(BEADS)
C
                      TPVX(PEN_AV) = RXNEW
                      TPVY(PEN_AV) = RYNEW
                      TPVZ(PEN_AV) = RZNEW
                  ENDIF
C
```

```
                DO 31400 CNT = LOW_AV, HI_AV
                  TPVX(CNT) = XVC1TP(CNT+ARRSFT)
                  TPVY(CNT) = YVC1TP(CNT+ARRSFT)
                  TPVZ(CNT) = ZVC1TP(CNT+ARRSFT)
31400           CONTINUE
                DO 31450 CNT = C1XB1+1, C1XB1+BEADS-1
                  XVC1TP(CNT) = TPVX(CNT)
                  YVC1TP(CNT) = TPVY(CNT)
                  ZVC1TP(CNT) = TPVZ(CNT)
31450           CONTINUE
C
C  Calculate the angles for the new system.
C
                TPA(C1XB1+1) = 0.0D0
                TPA(PEN_AV) = -(RXNEW*RX + RYNEW*RY + RZNEW*RZ) * RBON2
                DO 30200 CNT = LOW_AV+1, HI_AV
                  TPA(CNT) = ANG1TP(CNT+ARRSFT)
30200           CONTINUE
                DO 30250 CNT = C1XB1+1, C1XB1+BEADS-1
                  ANG1TP(CNT) = TPA(CNT)
30250           CONTINUE
C
C  Calculate the positions for the new system.
C
                TPX(END) = XTEMP
                TPY(END) = YTEMP
                TPZ(END) = ZTEMP
                DO 30300 CNT = LOWCT, HIGHCT
                  TPX(CNT) = X1C(CNT+ARRSFT)
                  TPY(CNT) = Y1C(CNT+ARRSFT)
                  TPZ(CNT) = Z1C(CNT+ARRSFT)
30300           CONTINUE
                DO 30350 CNT = CM1XBD+1, CM1XBD+BEADS
                  X1TP(CNT) = TPX(CNT)
                  Y1TP(CNT) = TPY(CNT)
                  Z1TP(CNT) = TPZ(CNT)
30350           CONTINUE
C
C  ***   End  CHAINS:  All of the chains have been moved.  ***
C
31000       CONTINUE
C
C    Metropolis sampling.
C
            ANGLSM=0.0D0
            DO 33000 CNT = 1, BM1XCH
              ANGLSM = ANGLSM + ANG1TP(CNT)
33000       CONTINUE
            BNDNEW = (0.50D+00 / TAUTMP) * ((BEADS-2)*CHAINS + ANGLSM)
C
            HAMNEW = BNDNEW + OSMOT * DBLE(CHAINS)
C
C           print*, 'In eq_rep, HAMNEW, HAMINI' , HAMNEW, HAMINI
```

```
                IF (HAMNEW .GT. HAMINI) THEN
                  DELHT = (HAMNEW - HAMINI)
                  IF (DELHT .GE. 170.0D0) THEN
                      ENGCNT = ENGCNT + 1
                      GO TO 30000
                    ELSE
                      BOLTZ = DEXP(- DELHT)
                      IF (BOLTZ .LT. DRAN(ISEED)) THEN
                        ENGCNT = ENGCNT + 1
                        GO TO 30000
                      ENDIF
                  ENDIF
                ENDIF
C
C     New positions are acceptable.  Place in permanent storage.
C
                ACCEPT = ACCEPT + 1
                MOVED = 0.0D0
                DO 34000 CNT = 1, 3*BDXCH
                  IF (POS1TP(CNT) .NE. POS1(CNT)) MOVED = MOVED + 1.0D0
34000           CONTINUE
                IF (MOVED .NE. 0.0D0) ACCMOV = ACCMOV + 1
                ACCFLG = 1
                BNDINI = BNDNEW
                HAMINI = HAMNEW
                ANG1(1:BM1XCH) = ANG1TP(1:BM1XCH)
                CM1(1: 3*CHAINS) = CM1TP(1: 3*CHAINS)
                POS1(1:3*BDXCH) = POS1TP(1:3*BDXCH)
                VEC1(1:3*BM1XCH) = VEC1TP(1:3*BM1XCH)
C
C     ***           This is the end of the  MCSTEP  loop.          ***
C
30000         CONTINUE
              SUCCES = SUCCES + ACCEPT
              SUCMOV = SUCMOV + ACCMOV
C             print*, 'suc, acc', succes, accept
C             print*, 'smov, amov', sucmov, accmov
C             print*, 'vol, eng', volcnt, engcnt
C     ***           This is the end of the  EQUIL   loop.          ***
40000     CONTINUE
          SUCCES = SUCCES / DBLE(EQSTEP * MCLIM)
          SUCMOV = SUCMOV / DBLE(EQSTEP * MCLIM)
          PRINT*, 'Success in the energy tests of REP_EQ =', SUCCES
          PRINT*, 'Success in the movement tests of REP_EQ =', SUCMOV
          RETURN
          END
```

APPENDIX D

DRAN: A RANDOM NUMBER GENERATOR

Used in All Simulations Excluding Those Performed
on the CDC CYBER 205s and the CDC ETA^10
at the John von Neumann Computing Center, Princeton, New Jersey

---

This program was orginally titled URAND for **Uni**form **Rand**om Number Generator, and was printed in George E Forsythe, Michael A. Malcolm, and Cleve B. Moler, Computer Methods for Mathematical Computations [Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1977], p. 246. The only changes to the program were to redefine the single precision variable S as double precision, and to introduce a SAVE statement to ensure that the machine constants (IA, IC, M2, MIC, and S) were kept for the next invocation of the subroutine.

This function is not particularly fast, but only consumed 5% or less of the overall execution time of our programs and was therefore deemed to be acceptable. With knowledge of the machine architecture, one may eliminate the three IF statements and substitute a simple assignment statement to IY after statement number 20 to increase the speed of execution. This was done for the Alliant FX/40-4 that only required the final mechanism in which integer overflow affects the sign bit. In this case, all IF statements were removed, and the statement before the assignment of a value to DRAN was

```
IY = IAND(IY, '7FFFFFFF'X)
```

This assignment statement inverts the sign bit on the variable IY.

Notes:

While this pseudo-random number generator provides machine dependent values, these values are dependent on the word size. All machines used in this study, excluding the Control Data Corporation's CYBER 205s and ETA^10, have 32-bit word sizes so that a series of values generated from any machine was exactly the same given the same initial seed value.

This program has been run successfully without modification on the following machines: Ardent Titan-2, Celerity 1200, Convex C-210, FPS 500, IBM 3090VF, MIPS M/120 and M/2000, Multiflow Trace 14/200, and Silicon Graphics Personal Iris 4D/20. This program was used on the Alliant FX/40-4 with the modification stated above.

The reference to "D. E. Knuth (1969), Vol. 2" mentioned below is: Donald E. Knuth, The Art of Computer Programming, Vol. 2: Seminumerical Algorithms [Reading, Massachusetts: Addison-Wesley Publishing Company, 1969] Chp. 3.

```
      DOUBLE PRECISION FUNCTION DRAN(IY)
      INTEGER IY
C
C     DRAN is a uniform random number generator based on theory and
C  suggestions given in D. E. Knuth (1969), Vol. 2.  The integer IY
C  should be initialized to an arbitrary integer prior to the first call
C  to DRAN.  The calling program should not alter the value of IY
C  between subsequent calls to DRAN.  Values of DRAN will be returned
C  in the interval (0,1).
C
      INTEGER             IA, IC, ITWO, M2, M, MIC
      DOUBLE PRECISION    HALFM, S
      DOUBLE PRECISION    DATAN, DSQRT
      SAVE                IA, IC, M2, MIC, S
      DATA  M2/0/, ITWO/2/
      IF (M2 .NE. 0) GO TO 20
C
C  If first entry, compute machine integer word length.
C
      M = 1
   10 M2 = M
      M = ITWO * M2
      IF (M .GT. M2) GO TO 10
      HALFM = M2
C
C  Compute multiplier and increment for linear congruential method.
C
      IA = 8*IDINT(HALFM * DATAN(1.D0) / 8.D0) + 5
      IC = 2*IDINT(HALFM * (0.5D0 - DSQRT(3.D0)/6.D0)) + 1
      MIC = (M2 - IC) + M2
C
C  S is the scale factor for converting to floating point.
C
      S = 0.5 / HALFM
C
C  Compute next random number.
C
   20 IY = IY * IA
C
C  The following statement is for computers which do not allow
C  integer overflow upon addition.
C
      IF (IY .GT. MIC) IY = (IY - M2) - M2
C
      IY = IY + IC
C
C  The following statement is for computers where the word length
C  for addition is greater than for multiplication.
C
      IF (IY/2 .GT. M2) IY = (IY - M2) - M2
```

```
C
C  The following statement is for computers where integer
C  overflow affects the sign bit.
C
      IF (IY .LT. 0) IY = (IY + M2) + M2
C
      DRAN = DBLE(IY) * S
      RETURN
      END
```

APPENDIX E

DEVELOPMENT OF THE SCATTERING FUNCTION $S(q)$

The intensity of X-ray scattering is represented by the equation [Alexander, 1969, p.40]

$$I(hkl) = P \cdot L \cdot j \cdot A \cdot |F|^2, \tag{A–1}$$

where

A $\equiv$ Absorption factor, which accounts for differences from expected intensities due to elemental composition, source wavelength, and the size and shape of the sample;

F $\equiv$ Structure factor. This is the most important factor for it represents the contribution from the positions of the scatterers relative to some crystallographic plane;

h, k, l $\equiv$ Indices for the crystallographic planes;

j $\equiv$ Multiplicity factor, arising from different sets of crystallographic planes contributing to a singly observed reflection;

L $\equiv$ Lorentz factor. This coefficient has many forms, but represents corrections for the reflecting time;

P $\equiv$ Polarization factor $= \frac{1}{2}(1 + \cos^2 \theta)$.

The factors A, j, L, and P will be ignored here as will the temperature corrections that are normally included in the structure factor F. Doing this provides the "ideal" intensity I_i as

$$I_i = |F|^2. \tag{A–2}$$

The structure factor for N scatterers may be represented as [Alexander, 1969, p. 39]

$$F(hkl) = \sum_{n=1}^{N} f_n \exp[2\pi i(hx_n + ky_n + lz_n)]$$

$$= \sum_{n=1}^{N} f_n \exp[2\pi i(h\hat{\mathbf{x}} + k\hat{\mathbf{y}} + l\hat{\mathbf{z}}) \cdot (x_n\hat{\mathbf{x}} + y_n\hat{\mathbf{y}} + z_n\hat{\mathbf{z}})], \qquad \text{(A–3)}$$

where f_n is the atomic scattering factor for the n^{th} scatterer. Using the definitions for the wavevector $\mathbf{q} \equiv 2\pi(h\hat{\mathbf{x}}+k\hat{\mathbf{y}}+l\hat{\mathbf{z}})$ and the position vector for the n^{th} scatterer $\mathbf{R}_n \equiv x_n\hat{\mathbf{x}} + y_n\hat{\mathbf{y}} + z_n\hat{\mathbf{z}}$, Equation A–3 may be rewritten in the form

$$F(\mathbf{q}) = \sum_{n=1}^{N} f_n \exp[i\mathbf{q} \cdot \mathbf{R}_n]. \qquad \text{(A–4)}$$

Thus, the "ideal" intensity becomes

$$I_i(\mathbf{q}) = |F|^2 = F \cdot F^* = \left(\sum_{n=1}^{N} f_n \exp[i\mathbf{q} \cdot \mathbf{R}_n]\right)\left(\sum_{m=1}^{N} f_m \exp[-i\mathbf{q} \cdot \mathbf{R}_m]\right)$$

$$= \sum_{n=1}^{N}\sum_{m=1}^{N} f_n f_m \exp[i\mathbf{q} \cdot (\mathbf{R}_n - \mathbf{R}_m)] \qquad \text{(A–5)}$$

$$= \sum_{n=1}^{N}\sum_{m=1}^{N} f_n f_m \exp[i\mathbf{q} \cdot \mathbf{r}_{nm}],$$

where $\mathbf{r}_{nm} = \mathbf{R}_n - \mathbf{R}_m$ and is the difference vector between scatterers n and m.

Since all of the scatterers here are of the same type, $f_m = f_n = f$, a constant. A more important consideration arises from the aspect that in this development for solutions, the scatterers may be continuously distributed in space, as opposed to crystal structures where the atomic positions are fixed (ignoring such effects as thermal vibrations). As such, an ensemble average must be taken to consider all of the possible configurations of the system that may contribute to the intensity. This implies that the "ideal" intensity $I_i(q) = \langle I_i(\mathbf{q})\rangle$ becomes

$$I_i(q) = \langle f^2 \sum_{n=1}^{N}\sum_{m=1}^{N} \exp[i\mathbf{q} \cdot \mathbf{r}_{nm}]\rangle. \qquad \text{(A–6)}$$

(Note that the intensity is independent of the angular part of q.)

Polymeric chains have a constraint on connectivity that will become important below. To facilitate the representation of this constraint, a change in the notation is made here. Using $\mathbf{R}_n$ as the position vector for the n^{th} scatterer, the scatterer is relabeled to be identified by the number of the chain to which the scatterer belongs, k, and the scatterer's position along the chain, u (*i.e.*, the bead number). Thus, a sum over all of the position vectors to find the center of mass for a system could be written in either of these ways

$$\frac{1}{N}\sum_{n=1}^{N}\mathbf{R}_n \equiv \frac{1}{C\,B}\sum_{k=1}^{C}\sum_{u=1}^{B}\mathbf{R}_{ku}. \tag{A–7}$$

where C is the number of chains in the system and B is the number of beads *per* chain. For example, if a system consists of two chains each having five beads *per* chain, the middle bead on the second chain would be denoted $k = 2$, $u = 3$; in the original notation this bead would have been denoted $n = 8$. Mathematically, the relationships may be expressed as $k = \lceil n/B \rceil$ and $u = [(n-1) \bmod B] + 1$ so that $I_i(q)$ becomes

$$I_i(q) = \left\langle f^2 \sum_{k=1}^{C}\sum_{u=1}^{B}\sum_{l=1}^{C}\sum_{v=1}^{B}\exp(i\mathbf{q}\cdot\mathbf{r}_{ku,lv})\right\rangle. \tag{A–8}$$

Separating this equation into scattering from beads on the same chain and scattering from beads on different chains provides

$$I_i(q) = \left\langle f^2 \left[\sum_{k=1}^{C}\sum_{u=1}^{B}\sum_{v=1}^{B}\exp(i\mathbf{q}\cdot\mathbf{r}_{ku,kv}) + \sum_{k=1}^{C}\sum_{u=1}^{B}\sum_{l\neq k=1}^{C}\sum_{v=1}^{B}\exp(i\mathbf{q}\cdot\mathbf{r}_{ku,lv})\right]\right\rangle. \tag{A–9}$$

Interchanging the sums and the ensemble average, which presumes absolute convergence of the sums and averaging integrals results in the form

$$I_i(q) = f^2 \left[\sum_{k=1}^{C}\sum_{u=1}^{B}\sum_{v=1}^{B}\langle\exp(i\mathbf{q}\cdot\mathbf{r}_{ku,kv})\rangle + \sum_{k=1}^{C}\sum_{u=1}^{B}\sum_{l\neq k=1}^{C}\sum_{v=1}^{B}\langle\exp(i\mathbf{q}\cdot\mathbf{r}_{ku,lv})\rangle\right]. \tag{A–10}$$

Using the definition of the ensemble average as an integral over all space for the probability of existence P, Equation A–10 may be transformed into

$$I_i(q) = f^2 \left[\sum_{k=1}^{C} \sum_{u=1}^{B} \sum_{v=1}^{B} \int_V d\mathbf{r} P^s_{ku,kv}(|\mathbf{r}|) \exp(i\mathbf{q}\cdot\mathbf{r}) + \sum_{k=1}^{C} \sum_{u=1}^{B} \sum_{l\neq k=1}^{C} \sum_{v=1}^{B} \int_V d\mathbf{r} P^o_{ku,lv}(|\mathbf{r}|) \exp(i\mathbf{q}\cdot\mathbf{r}) \right], \qquad \text{(A–11)}$$

where

$P^s_{ku,kv}(|\mathbf{r}|)$ $\equiv$ Probability of finding bead v on chain k at a distance r from bead u on the same chain k;

$P^o_{ku,lv}(|\mathbf{r}|)$ $\equiv$ Probability of finding bead v on a chain other than k at a distance r from bead u on chain k.

By definition, $\int_V d\mathbf{r} P(r) = 1$. To meet this condition for the two different types of probabilities above, separate approaches are required. First, for the beads on the same chain as the observer, all of the beads on that chain must be found within a distance L, the contour length of the chain, since the beads are rigidly connected. So,

$$P^s_{ku,kv}(|\mathbf{r}|) = \begin{cases} g^s_{ku,kv}(r), & \text{for } r \leq L; \\ 0, & \text{for } r > L. \end{cases} \qquad \text{(A–12)}$$

where $g^s_{ku,kv}(r)$ is the pair radial distribution function for beads on the same chain k. This quantity has the additional constraint that

$$g^s_{ku,ku}(r) = \delta(r) = \delta(0) = 1. \qquad \text{(A–13)}$$

Equation A–13 simply represents the obvious idea that a bead's position must be completed correlated to itself. However, this condition is necessary to guarantee that the integral

$$\int_V d\mathbf{r} P^s_{ku,kv}(|\mathbf{r}|) \exp(i\mathbf{q}\cdot\mathbf{r}) = \int_V d\mathbf{r}\ g^s_{ku,kv}(r) \exp(i\mathbf{q}\cdot\mathbf{r}). \qquad \text{(A–14)}$$

is convergent. Second, for beads on chains other than the observer, there is no limit to the region of space in which the beads may be found, but, if *all* of

space is considered, then the other beads *must* be encountered. The pair radial distribution function for beads on other chains then meets the constraint

$$\lim_{r\to\infty} g^{o}_{ku,lv}(r) = 1. \tag{A–15}$$

For the probability distribution to be normalized correctly, the relation

$$P^{o}_{ku,lv}(|\mathbf{r}|) = \frac{g^{o}_{ku,lv}(r)}{V} \tag{A–16}$$

must be used so that

$$\int_V d\mathbf{r} P^{o}_{ku,lv}(|\mathbf{r}|)\exp(i\mathbf{q}\cdot\mathbf{r}) = \int_V d\mathbf{r}\,\frac{g^{o}_{ku,lv}(r)}{V}\exp(i\mathbf{q}\cdot\mathbf{r}). \tag{A–17}$$

Hence, Equation A–11 for the "ideal" intensity may be rewritten in the following form:

$$I_i(q) = f^2\left[\sum_{k=1}^{C}\sum_{u=1}^{B}\sum_{v=1}^{B}\int_V d\mathbf{r}\; g^{s}_{ku,kv}(r)\exp(i\mathbf{q}\cdot\mathbf{r}) + \sum_{k=1}^{C}\sum_{u=1}^{B}\sum_{l\neq k=1}^{C}\sum_{v=1}^{B}\int_V d\mathbf{r}\,\frac{g^{o}_{ku,lv}(r)}{V}\exp(i\mathbf{q}\cdot\mathbf{r})\right]. \tag{A–18}$$

Since all chains are, on average, equivalent in this system, the sums over the contributions of the chains are equal to the number of chains multiplied by the contribution of a single chain. Choosing an arbitrary chain in the system and labeling it "1" gives

$$I_i(q) = f^2\left[C\sum_{u=1}^{B}\sum_{v=1}^{B}\int_V d\mathbf{r}\; g^{s}_{1u,1v}(r)\exp(i\mathbf{q}\cdot\mathbf{r}) + \frac{C}{V}\sum_{l=2}^{C}\sum_{u=1}^{B}\sum_{v=1}^{B}\int_V d\mathbf{r}\; g^{o}_{1u,lv}(r)\exp(i\mathbf{q}\cdot\mathbf{r})\right]. \tag{A–19}$$

Using the same argument for the indistinguishability of some chain $l = 2$, Equation A–19 may be rewritten as

$$I_i(q) = f^2\left[C\sum_{u=1}^{B}\sum_{v=1}^{B}\int_V d\mathbf{r}\; g^{s}_{1u,1v}(r)\exp(i\mathbf{q}\cdot\mathbf{r}) + \frac{C}{V}(C-1)\sum_{u=1}^{B}\sum_{v=1}^{B}\int_V d\mathbf{r}\; g^{o}_{1u,2v}(r)\exp(i\mathbf{q}\cdot\mathbf{r})\right]. \tag{A–20}$$

To properly evaluate Equation A–20, the term

$$\int_V d\mathbf{r}\; g^{o}_{1u,2v}(r)\,\exp(i\mathbf{q}\cdot\mathbf{r}) \tag{A–21}$$

should be placed in an absolutely convergent form. Hence,

$$\begin{aligned}\frac{1}{V}\int_V d\mathbf{r}\; g^{o}_{1u,2v}(r)\,\exp(i\mathbf{q}\cdot\mathbf{r}) &= \frac{1}{V}\int_V d\mathbf{r}\;[g^{o}_{1u,2v}(r)-1+1]\,\exp(i\mathbf{q}\cdot\mathbf{r})\\ &= \frac{1}{V}\int_V d\mathbf{r}\;[g^{o}_{1u,2v}(r)-1]\,\exp(i\mathbf{q}\cdot\mathbf{r})\\ &\quad+\frac{1}{V}\int_V d\mathbf{r}\;\exp(i\mathbf{q}\cdot\mathbf{r}) \qquad\qquad (A–22)\\ &= \frac{1}{V}\int_V d\mathbf{r}\;[g^{o}_{1u,2v}(r)-1]\,\exp(i\mathbf{q}\cdot\mathbf{r})\\ &\quad+\frac{1}{V}\delta(\mathbf{q}-\mathbf{0}).\end{aligned}$$

Thus, excluding $\mathbf{q}=\mathbf{0}$, which means that infinitely long length scales are avoided,

$$\frac{1}{V}\int_V d\mathbf{r}\; g^{o}_{1u,2v}(r)\,\exp(i\mathbf{q}\cdot\mathbf{r}) = \frac{1}{V}\int_V d\mathbf{r}\;[g^{o}_{1u,2v}(r)-1]\,\exp(i\mathbf{q}\cdot\mathbf{r}),\quad \mathbf{q}\neq\mathbf{0}. \tag{A–23}$$

To compare intensities from differently sized systems, the intensity is normalized by the volume of the system by defining the Scattering Function $S(q)$ and the chain density ρ_c as

$$S(q)\equiv\frac{I_i(q)}{V}\quad\text{and}\quad\rho_c\equiv\frac{C}{V}. \tag{A–24}$$

Equation A–20 may then be written in the form (for $\mathbf{q}\neq\mathbf{0}$)

$$\begin{aligned}S(q) = f^2\Bigg\{&\rho_c\sum_{u=1}^{B}\sum_{v=1}^{B}\int_V d\mathbf{r}\; g^{s}_{1u,1v}(r)\,\exp(i\mathbf{q}\cdot\mathbf{r})\\ &+\rho_c(\rho_c-\frac{1}{V})\sum_{u=1}^{B}\sum_{v=1}^{B}\int_V d\mathbf{r}\;[g^{o}_{1u,2v}(r)-1]\,\exp(i\mathbf{q}\cdot\mathbf{r})\Bigg\}.\end{aligned} \tag{A–25}$$

As the system (not the periodic box here) is large, $1/V\rightarrow 0$. Also, since these simulations are being performed on generic beads, one may set $f=1$ without any

loss of generality. Finally, one obtains the expression for the Scattering Function $S(q)$ as

$$S(q) = \left\{ \rho_c \sum_{u=1}^{B} \sum_{v=1}^{B} \int_V d\mathbf{r}\; g^s_{1u,1v}(r)\; \exp(i\mathbf{q}\cdot\mathbf{r}) \right.$$
$$\left. + \rho_c^2 \sum_{u=1}^{B} \sum_{v=1}^{B} \int_V d\mathbf{r}\; [g^o_{1u,2v}(r) - 1]\; \exp(i\mathbf{q}\cdot\mathbf{r}) \right\}, \quad \mathbf{q} \neq \mathbf{0}$$

(A–26)

BIBLIOGRAPHY

Alexander, Leroy E. [1969], *X-ray Diffraction Methods in Polymer Science* [New York: John Wiley and Sons, Inc., 1969]. Reprint with corrections. [Malabar, Florida: Robert E. Krieger Publishing Company, Inc., 1985].

Allen, M. P.; and Tildesley, D. J. [1987], *Computer Simulations of Liquids* [New York: Oxford University Press, 1987].

Baumgärtner, A. [1980], "Statics and Dynamics of the Freely Jointed Polymer Chain with Lennard-Jones Interaction," *Journal of Chemical Physics*, **72**, 2 (January 15, 1980) 871–879.

Baumgärtner, A. [1985], "Hard Rods on Lattices," *Le Journal de Physique—Lettres*, **46**, 15 (August 1, 1985) L-659–L-666.

Baumgärtner, Artur [1986], "Phase Transitions of Semiflexible Lattice Polymers," *Journal of Chemical Physics*, **84**, 3 (February 1, 1986) 1905–1908.

Baumgärtner, A. [1987], "Simulations of Polymer Models," *Topics in Current Physics, Vol 36: Applications of the* Monte Carlo Method *in Statistical Physics*, 2nd Edition, K. Binder, ed. [New York: Springer-Verlag, 1987] 145–180.

Baumgärtner, A.; and Binder, K. [1979], "Monte Carlo Studies on the Freely Jointed Polymer Chain with Excluded Volume Interaction," *Journal of Chemical Physics*, **71**, 6 (September 15, 1979) 2541–2545.

Baumgärtner, A.; and Yoon, D. Y. [1983], "Phase Transition of Lattice Polymer Systems," *Journal of Chemical Physics*, **79**, 1 (July 1, 1983) 521–522.

Bellemans, A.; and de Vos, E. [1973], "On the Combinatorial Entropy of Athermal Polymer Solutions," *Journal of Polymer Science: Polymer Symposia, No. 42: International Symposium on Macromolecules, Helsinki, 1972, Part 3*, **42** (1973) 1195–1197.

Bellemans, A.; and Janssens, M. [1974], "On the Osmotic Second Virial Coefficient of Athermal Polymer Solutions," *Macromolecules*, **7**, 6 (November-December, 1974) 809–811.

Billmeyer, Fred W., Jr. [1984], *Textbook of Polymer Science*, 3rd Ed. [New York: John Wiley and Sons, Inc., 1984].

Boyd, Richard H. [1989], "An Off-Lattice Constant-Pressure Simulation of Liquid Polymethylene," *Macromolecules*, **22**, 5 (May, 1989) 2477–2481.

Candau, Françoise; Strazielle, Claude; and Benoit, Henri [1976], "Etude par Pression Osmotique de Polystyrenes Lineaires et Ramifies en Solution: Determination de Leurs Parametres Thermodynamiques," *European Polymer Journal*, **12**, 2 (1976) 95–103.

Chandler, David [1987], *Introduction to Modern Statistical Mechanics* [New York: Oxford University Press, 1987].

des Cloizeaux, J. [1975], "The Lagrangian Theory of Polymer Solutions at Intermediate Concentrations," *Le Journal de Physique*, **36**, 4 (April, 1975) 281–291.

Cotton, J. P.; Decker, D.; Benoit, H.; Farnoux, B.; Higgins, J.; Jannink, G.; Ober, R.; Picot, C.; and des Cloizeaux, J. [1974], "Conformation of Polymer Chain in the Bulk," *Macromolecules*, **7**, 6 (November-December, 1974) 863–872.

Cotton, J. P.; Farnoux, B.; Jannick, G.; and Strazielle, C. [1973], "Dilute and Semidilute Solutions: Light and Neutron Scattering and Osmotic Pressure," *Journal of Polymer Science: Polymer Symposia, No. 42: International Symposium on Macromolecules, Helsinki, 1972, Part 2*, 42 (1973) 981–985

Croxton, Clive A. [1979], "Configurational Properties of Athermal Self-Avoiding Polymer Chains at Intermediate to High Concentrations," *Journal of Physics A: Mathematical and General*, **12**, 12 (December, 1979) 2497–2508.

Curro, John G. [1976], "Computer Simulation of Multiple Chain Systems—Equation of State of Hard Sphere Chains," *Journal of Chemical Physics*, **64**, 6 (March 15, 1976) 2496–2500.

Curro, J. G. [1980], "Monte Carlo Simulation of Polymers in Solution and Bulk," *Journal of Macromolecular Science—Physics*, **B18**, 3 (1980) 343–362.

Curro, John G.; and Schweizer, Kenneth S. [1987a], "Equilbrium Theory of Polymer Liquids: Linear Chains," *Journal of Chemical Physics*, **87**, 3 (August 1, 1987) 1842–1846.

Curro, John G.; and Schweizer, Kenneth S. [1987b], "Theory of Polymer Melts: An Integral Equation Approach," *Macromolecules*, **20**, 8 (August, 1987) 1928–1934.

Dadmun, Mark A. [1987], Unpublished results: Dimensions of short chains (1987).

Dadmun, Mark A. [1988], Unpublished results: Exclusions of phase space due to double culs-de-sacs (1988).

Daoud, M.; Cotton, J. P.; Farnoux, B.; Jannink, G.; Sarma, G.; Benoit, H.; Duplessix, R.; Picot, C.; and de Gennes, P. G. [1975], "Solutions of Flexible Polymers. Neutron Experiments and Interpretation," *Macromolecules*, **8**, 6 (November-December, 1975) 804–818.

Debye, P. [1915], "Zerstreuung von Röntgenstrahlen," *Annalen der Physik*, **46**, Ser. 4 (1915) 809–823.

Dickman, Ronald [1987], "New Simulation Method for the Equation of State of Lattice Chains," *Journal of Chemical Physics*, **87**, 4 (August 15, 1987) 2246–2248.

Dickman, Ronald; and Hall, Carol K. [1986a], "High Density Monte Carlo Simulations of Chain Molecules: Bulk Equation of State and Density Profile Near Walls," Unpublished manuscript (*ca.* 1986).

Dickman, Ronald; and Hall, Carol K. [1986b], "Equation of State for Athermal Lattice Chains," *Journal of Chemical Physics*, **85**, 5 (September 1, 1986) 3023–3026.

Dickman, Ronald; and Hall, Carol K. [1986c], "Equation of State for Chain Molecules: Continuous-Space Analog of Flory Theory," *Journal of Chemical Physics*, **85**, 7 (October 1, 1986) 4108–4115.

Dickman, Ronald; and Hall, Carol K. [1988], "High Density Monte Carlo Simulations of Chain Molecules: Bulk Equation of State and Density Profile Near Walls," *Journal of Chemical Physics*, **5**, (September 1, 1988) 3168–3174.

Edwards, S. F. [1965], "The Statistical Mechanics of Polymers with Excluded Volume," *Proceedings of the Physical Society*, **85**, 4, Issue No. 546 (April, 1965) 613–624.

Edwards, S. F. [1966], "The Theory of Polymer Solutions at Intermediate Concentration," *Proceedings of the Physical Society*, **88**, 2, Issue No. 560 (June, 1966) 265–280.

Flory, Paul J. [1941], "Thermodynamics of High Polymer Solutions," *Journal of Chemical Physics*, **9**, 8 (August, 1941) 660–661.

Flory, Paul J. [1953], *Principles of Polymer Chemistry* [Ithaca, New York: Cornell University Press, 1953].

Flory, Paul J. [1970], "Thermodynamics of Polymer Solutions," *Discussions of the Faraday Society, No. 49: Polymer Solutions*, 49 (1970) 7–29.

Forsythe, George E.; Malcolm, Michael A.; and Moler, Cleve B. [1977], *Computer Methods for Mathematical Computations* [Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1977].

Freed, Karl F.; and Bawendi, M. G. [1989], "Lattice Theories of Polymeric Fluids," *Journal of Physical Chemistry*, 93, 6 (June, 1989) 2194–2203.

de Gennes, P. G. [1971], "Reptation of a Polymer Chain in the Presence of Fixed Obstacles," *Journal of Chemical Physics*, 55, 2 (July 15, 1971) 572–579.

de Gennes, P. G. [1972], "Exponents for the Excluded Volume Problem as Derived by the Wilson Method," *Physics Letters*, 38A, 5 (February 28, 1972) 339–340.

de Gennes, Pierre-Gilles [1979], *Scaling Concepts in Polymer Physics* [Ithaca, New York: Cornell University Press, 1979].

Gujrati, P. D.; and Goldstein, Martin [1981], "On the Validity of the Flory-Huggins Approximation for Semiflexible Chains," *Journal of Chemical Physics*, 74, 4 (February 15, 1981) 2596–2603.

Ho, Jyh-Shyong; and Baumgärtner, Arthur [1989], Private communication: Discussion of finite size effects (October 30, 1989).

Hobson, Arthur [1971], *Concepts in Statistical Mechanics* [New York: Gordon and Breach, Science Publishers, Inc., 1971].

Honnell, Kevin G.; and Hall, Carol K. [1989], "A New Equation of State for Athermal Chains," *Journal of Chemical Physics*, 90, 3 (February 1, 1989) 1841–1855.

Huggins, Maurice L. [1941], "Solutions of Long Chain Compounds," *Journal of Chemical Physics*, 9, 5 (May, 1941) 440.

Jones, Dumont M. [1989], Private communication: Discussions on statistical bias and microreversibility (August 11, 1989).

Klug, Harold P.; and Alexander, Leroy E. [1974], *X-ray Diffraction Procedures for Polycrystalline and Amorphous Materials*, 2nd Ed. [New York: John Wiley and Sons, Inc., 1974].

Knuth, Donald E. [1969], *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms* [Reading, Massachusetts: Addison-Wesley Publishing Company, 1969].

Kolinski, Andrzej [1984], "On the Entropy of the Multichain Athermal Lattice Systems," *Journal of Polymer Science: Polymer Letters Edition*, **22**, 7 (July, 1984) 407–411.

Kolinski, Andrzej; Skolnick, Jeffrey; and Yaris, Robert [1986a], "On the Short Time Dynamics of Dense Polymeric Systems and the Origin of the Glass Transition: A Model System," *Journal of Chemical Physics*, **84**, 3 (February 1, 1986) 1922–1931.

Kolinski, Andrzej; Skolnick, Jeffrey; and Yaris, Robert [1986b], "The Collapse Transition of Semiflexible Polymers. A Monte Carlo Simulation of a Model System," *Journal of Chemical Physics*, **85**, 6 (September 15, 1986) 3585–3597.

Kolinski, Andrzej; Skolnick, Jeffrey; and Yaris, Robert [1986c], "Monte Carlo Study of Local Orientational Order in a Semiflexible Polymer Melt Model," *Macromolecules*, **19**, 10 (October, 1986) 2550–2560.

Kolinski, Andrzej; Skolnick, Jeffrey; and Yaris, Robert [1986d], "Order-Disorder Transitions in Tetrahedral Lattice Polymer Systems," *Macromolecules*, **19**, 10 (October, 1986) 2560–2567.

Kolinski, Andrzej; Skolnick, Jeffrey; and Yaris, Robert [1987a], "Does Reptation Describe the Dynamics of Entangled, Finite Length Polymer Systems? A Model Simulation," *Journal of Chemical Physics*, **86**, 3 (February 1, 1987) 1567–1585.

Kolinski, Andrzej; Skolnick, Jeffrey; and Yaris, Robert [1987b], "Dynamic Monte Carlo Study of the Conformational Properties of Long Flexible Polymers," *Macromolecules*, **20**, 2 (February, 1987) 438–440.

Kolinski, Andrzej; Skolnick, Jeffrey; and Yaris, Robert [1987c], "Monte Carlo Studies on the Long Time Dynamic Properties of Dense Cubic Lattice Multichain Systems. I. The Homopolymeric Melt," *Journal of Chemical Physics*, **86**, 12 (June 15, 1987) 7164–7173.

Koningsveld, R.; and Kleintjens, L. A. [1971], "Liquid-Liquid Phase Separation in Multicomponent Polymer Systems X. Concentration Dependence of the Pair-Interaction Parameter in the System Cyclohexane-Polystyrene," *Macromolecules*, 4, 5 (September-October, 1971) 637–641.

Kron, A. K. [1965], "The Monte Carlo Method in Statistical Calculations of Macromolecules," *Polymer Science U.S.S.R.*, 7, 7 (1965) 1361–1367.

Kron, A. K.; and Ptitsyn, O. B. [1967], "Calculation of Volume Effects in Macromolecules by Means of a Monte Carlo Method: Non-Self-Intersecting Chains on a Cubic Lattice," *Polymer Science U.S.S.R.*, **9**, 4 (1967) 847–853.

Liu, K. S.; Kalos, M. H.; and Chester, G. V. [1974], "Quantum Hard Spheres in a Channel," *Physical Review A*, **10**, 1 (July, 1974) 303–308.

Longman, G. W.; Wignall, G. D.; and Sheldon, R. P. [1976], "Radial Distribution Functions from Molten Polyethylene by X-ray Diffraction," *Polymer: The Science and Technology of Polymers and Biopolymers*, **17**, 6 (June, 1976) 485–487.

McKenzie, D. S. [1976], "Polymers and Scaling," *Physics Reports: A Review Section of Physics Letters (Section C)*, **27C**, 2 (September, 1976) 35–88.

McQuarrie, Donald A. [1976], *Statistical Mechanics* [New York: Harper and Row, Publishers, Inc., 1976].

Metropolis, Nicholas; Rosenbluth, Arianna W.; Rosenbluth, Marshall N.; Teller, Augusta H.; and Teller, Edward [1953], "Equation of State Calculations by Fast Computing Machines," *Journal of Chemical Physics*, **21**, 6 (June, 1953) 1087–1092.

Meyer, K. H. [1939], "Entropy of Mixing for Systems with Long-Chain Compounds and Its Statistical Explanation," *Zeitschrift für physikalische Chemie, Abteilung B: Chemie der Elementarprozesse aufbau der Materie*, 44 (1939) 383–391.

Meyer, Kurt H. [1940], "Propriétés de Polymères en Solution XVI. Interprétation Statistique des Propriétés Thermodynamiques de Systèmes Binaires Liquides," *Helvetica Chimica Acta*, **23** (1940) 1063–1070.

Muthukumar, M. [1985], Private communication: Development of the compressibility form of Flory-Huggins theory for simulations (1985).

Muthukumar, M. [1986], "Thermodynamics of Polymer Solutions," *Journal of Chemical Physics*, **85**, 8 (October 15, 1986) 4722–4728.

Muthukumar, M.; and Edwards, S. F. [1982], "Extrapolation Formulas for Polymer Solution Properties," *Journal of Chemical Physics*, **76**, 5 (March 1, 1982) 2720–2730.

Okamoto, Hiroshi [1976], "Monte Carlo Study of Systems of Linear Oligomers in Two-Dimensional Spaces," *Journal of Chemical Physics*, **64**, 6 (March 15, 1976) 2686–2691.

Okamoto, Hiroshi; and Bellemans, A. [1979], "Thermodynamical Properties Related to Chemical Potentials of Chain Molecules in Solutions by Computer Simulation," *Nippon Butsurigakkai Journal: Journal of the Physical Society of Japan*, **47**, 3 (September, 1979) 955–959.

Okamoto, Hiroshi; Itoh, Kazuya; and Araki, Takabumi [1983], "Scaling Relations of Two-Dimensional Athermal Multichain Systems by Computer Simulation," *Journal of Chemical Physics*, **78**, 2 (January 15, 1983) 975–979.

Percus, J. K. [1976], "Model for Density Variation at a Fluid Surface," *Journal of Statistical Physics*, **15**, 5 (November, 1976) 423–435.

Rosenbluth, Marshall N.; and Rosenbluth, Arianna W. [1955], "Monte Carlo Calculation of the Average Extension of Molecular Chains," *Journal of Chemical Physics*, **23**, 2 (February, 1955) 356–359.

Schweizer, Kenneth S.; and Curro, John G. [1988a], "Equation of State of Polymer Melts: General Formulation of a Microscopic Integral Equation Theory," *Journal of Chemical Physics*, **89**, 5 (September 1, 1988) 3342–3349.

Schweizer, Kenneth S.; and Curro, John G. [1988b], "Equation of State of Polymer Melts: Numerical Results for Athermal Freely Jointed Chain Fluids," *Journal of Chemical Physics*, **89**, 5 (September 1, 1988) 3350–3362.

Schweizer, Kenneth S.; and Curro, John G. [1988c], "Integral Equation Theory of Polymer Melts: Intramolecular Structure, Local Order, and the Correlation Hole," *Macromolecules*, **21**, 10 (October, 1988) 3070–3081.

Schweizer, Kenneth S.; and Curro, John G. [1988d], "Integral Equation Theory of Polymer Melts: Density Fluctuations, Static Structure Factor, and Comparison with Incompressible and Continuum Limit Models," *Macromolecules*, **21**, 10 (October, 1988) 3082–3087.

Snook, Ian K.; and Henderson, Douglas [1978], "Monte Carlo Study of a Hard-Sphere Fluid Near a Hard Wall," *Journal of Chemical Physics*, **68**, 5 (March 1, 1978) 2134–2139.

Starry, Scott K. [1987], Unpublished results: Low temperature studies on a simple cubic lattice (December, 1987).

Starry, Scott K. [1989], Unpublished results: Effect of the aspect ratio of a bead-link chain on the bead-bead radial distribution functions (March, 1989).

Sullivan, D. E.; Levesque, D.; and Weis, J. J. [1980], "Structure of a Simple Fluid Near a Wall. II. Comparison with Monte Carlo," *Journal of Chemical Physics*, **72**, 2 (January 15, 1980) 1170–1174.

Tolman, Richard C. [1938], *The Principles of Statistical Mechanics* [New York: Oxford University Press, 1938]. Unabridged reprint. [New York: Dover Publications, Inc., 1979].

Vacatello, Michele; Avitabile, Gustavo; Corradini, Paolo; and Tuzi, Angela [1980], "A Computer Model of Molecular Arrangement in a *n*-Paraffinic Liquid," *Journal of Chemical Physics*, **73**, 1 (July 1, 1980) 548–552.

Vaz, Roy [1989], Private communication: Discussions on Skolnick's dodecahedral lattice (April, 1989).

Wall, Frederick T.; and Mandel, Frederic [1975], "Macromolecular Dimensions Obtained by an Efficient Monte Carlo Method without Sample Attrition," *Journal of Chemical Physics*, **63**, 11 (December 1, 1975) 4592–4595.

Warren, B. E. [1937], "X-ray Determination of the Structure of Liquids and Glass," *Journal of Applied Physics*, **8**, 10 (October, 1937) 645–654.

Webman, I.; Lebowitz, J. L.; and Kalos, M. H. [1980], "Monte-Carlo Studies of a Polymer between Planes, Crossover between Dimensionalities," *Le Journal de Physique*, **41**, 6 (June, 1980) 579–583.

Webman, I.; Lebowitz, Joel L.; and Kalos, M. H. [1981],"A Monte Carlo Study of the Collapse of a Polymer Chain," *Macromolecules*, 14, 5 (September-October, 1981) 1495–1501.

Wertheim, M. S. [1987], "Thermodynamic Perturbation Theory of Polymerization," *Journal of Chemical Physics*, 87, 12 (December 15, 1987) 7323–7331.

Wolf, B. A. [1975], "Polymer-Solvent Interaction Parameters" in *Polymer Handbook*, J. Brandup and E. H. Immergut, eds., with W. McDowell, collaborator [New York: John Wiley and Sons, Inc., 1975] IV-131–IV-134.

Yamakawa, Hiromi [1971], *Modern Theory of Polymer Solutions* [New York: Harper and Row, Publishers, Inc., 1971].