INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Dissertation

The Role of Polymer Architecture in Determining the Static and Dynamic Properties of Polymers in the Melt

Submitted by

Scott P. Brown

Chemistry Department

In partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Colorado State University
Fort Collins, CO

Spring 2001

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

UMI Number: 3013827

UMI®

UMI Microform 3013827
Copyright 2001 by Bell & Howell Information and Learning Company.
All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Colorado State University

December 15, 2000

We hereby recommend that the dissertation prepared under our supervision by Scott P. Brown entitled The Role of Polymer Architecture in Determining the Static and Dynamic Properties of Polymers in the Melt be accepted as fulfilling in part requirements for the degree of Doctor of Philosophy.

Committee on Graduate Work

Grzegorz Szamel (advisor)

Kenneth E. DeBruin

Ellen R. Fisher

Marshall Fixman

Martin P. Gelfand

Department Head

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ABSTRACT

The Role of Polymer Architecture in Determining the Static and Dynamic Properties of Polymers in the Melt

The static and dynamic properties of ring polymers and three-arm star polymers are investigated via computer simulation. Ring and star polymer simulations have been performed for both melts and single chains.

We find that rings in the melt are smaller and more compact than corresponding linear chains: for rings and linear chains in the melt the mean-square radius of gyration scales with degree of polymerization as $R_g^2 \propto N^{0.82}$ and $R_g^2 \propto N$, respectively. Simulations in which bonds are allowed to cross show that the smaller ring size is a consequence of the constraint of non-concatenation: for crossing rings in the melt the mean-square radius of gyration scales as $R_g^2 \propto N$.

We find that rings in the melt have faster dynamics than linear chains of comparable N. For rings the self-diffusion coefficient and orientational relaxation time scale as $D \propto N^{-1.6}$ and $\tau_{ee} \propto N^{2.5}$, respectively. Faster ring dynamics is shown to persist for ring sizes up to twenty times greater than the entanglement crossover chain length for linear chains. For crossing rings in the melt we find that the self-diffusion coefficient and orientational relaxation time scale as $D \propto N^{-1}$ and $\tau_{ee} \propto N^{2.0}$, respectively.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

For crossing and non-crossing single rings we find that the scaling of the mean-square radius of gyration is identical in both cases, $R_g^2 \propto N^{1.17}$. Additionally we find that for both crossing and non-crossing single rings the self-diffusion coefficient and orientational relaxation time scale identically as $D \propto N^{-1}$ and $\tau_{ee} \propto N^{2.1}$, respectively.

For three-arm star polymers in the melt we find that the mean-square radius of gyration scales as $R_g^2 \propto N$. For the dynamics we observe a possible non-power-law dependence of the self-diffusion coefficient on star size. The limited range of star sizes simulated precludes any stronger comment; however, we argue that the largest star lies in at least a semi-entangled dynamics regime.

For the simulations of single stars we find that the mean-square radius of gyration scales as $R_g^2 \propto N^{1.2}$, and that the self-diffusion coefficient and relaxation time scale as $D \propto N^{-1}$ and $\tau_{ce} \propto N^{2.3}$, respectively.

Scott P. Brown
Chemistry Department, Colorado State University, Fort Collins, CO 80523
Spring 2001

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Contents

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of Tables

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

List of Figures

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 1

Introduction to Polymers

The industrial production of polymer materials first began in the United States around 1910[1]. Today the demand for industrial polymers exceeds that of steel[2]. The high versatility of polymer compounds has led to their wide-spread use in a large variety of applications. Some examples from a long list are foams, medicinals, insulators, fibers, films, packaging materials, *etc*[3].

A major class of polymer materials produced are the so-called thermoplastics. Thermoplastics are polymers which are capable of being reversibly softened and hardened by heating and cooling. They are the recyclable plastics. Some typical thermoplastics are polystyrene, polyethylene, and polypropylene. Most thermoplastic processing is done in the melt in a procedure which typically involves relatively large and rapid deformations in order to mold the polymer[2]. Thus an understanding of polymer melt dynamics is of significant practical interest in the processing of thermoplastics. Despite this fact there exists no universally accepted

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

theoretical treatment that accounts for the dynamical behavior of polymers in the melt. The approaches to date have been largely from a phenomenological point of view.

The dynamics of polymer liquids is fundamentally different from the dynamics of low molecular weight fluids[4]. Qualitatively, differences in the dynamics of the two fluids can be readily seen by observing their macroscopic flow behavior. Such demonstrations as flow recoil in polymeric fluids (indicating the presence of "memory") illustrate the often counter-intuitive nature of polymer dynamics when compared with that of low molecular weight fluids[5]. One of the more striking demonstrations compares the results of placing a rotating rod into beakers containing each type of fluid[6]. For the low molecular weight fluid. the liquid near the rod is pushed outwards resulting in a depression around the rod. In contrast to this, the polymeric fluid moves toward the rotating rod and climbs up it. This is known as the "Weissenberg effect."

The unique features of polymer liquids are a direct consequence of the high molecular weight of the constituent polymer molecules. It is interesting to note that polymers were not believed to be macromolecules until the early part of the 20th century, despite the fact that the unusual properties of naturally occurring polymer compounds had been known for quite some time prior: as early as 1861 Thomas Graham noted that certain solutions of naturally occuring polymers were characterized by unusually sluggish dynamics[7]. The introduction of the word

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

"polymer" is credited to an 1833 publication by Jons Jacob Berzelius[8], but it was not until the 1920's that the work of Staudinger brought to the forefront the macromolecular nature of polymers[7].

The subject matter of this dissertation is the role of polymer architecture in determining the static and dynamic properties of polymers in the melt. The specific architectures examined are the non-linear ring and star polymer architectures.

In order to discuss ring and star polymers it is instructive to first begin with the topic of linear polymer chains, as they have been the most extensively studied architecture. This is the subject matter of Chapter 2, which presents a general overview of the properties of melts of linear polymer chains. Chapter 3 then describes the simulation model used to perform the computer simulations of ring and star polymers. Chapters 4 and 5 represent the main body of this thesis, namely the presentation and analysis of data gathered from the simulations of ring and star polymers. Chapter 6 closes this dissertation with a summary of the major results and a brief discussion.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 2

Background

2.1 Introduction

Before talking about ring and star polymers it is useful to first introduce properties particular to linear polymer chains. A discussion of linear polymers leads naturally into some of the concepts that are central to the currently accepted understanding of polymer dynamics, namely entanglements and reptation.

2.2 Linear polymers

Equilibrium properties

At room temperature a polymer chain in the melt has available to it many different conformations, each perhaps corresponding to a different total size of the chain. Thus it makes sense to talk about the size of polymer chains as an average quantity.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The customary measure of average size is the mean-square radius of gyration R_g^2, which is simply a measure of the average monomer distance from the polymer center of mass,

$$R_g^2 = \frac{1}{n_{pol}} \frac{1}{N} \sum_{\alpha=1}^{n_{pol}} \sum_{i=1}^{N} \left\langle (\mathbf{r}_i^\alpha - \mathbf{R}_{cm}^\alpha)^2 \right\rangle ; \tag{2.1}$$

where $\mathbf{R}_{cm}$ is the position of the polymer center of mass, $\mathbf{r}_i$ is the position of the ith monomer and n_{pol} is the total number of polymer chains in the system. Note that the end-to-end distance $R^2 = \frac{1}{n_{pol}} \sum_{\alpha=1}^{n_{pol}} \langle (\mathbf{r}_1^\alpha - \mathbf{r}_N^\alpha)^2 \rangle$ can also be used as a measure of average chain size; however, this measure becomes ambiguous when applied to ring and star polymers.

The radius of gyration for a single, ideal linear polymer chain follows the scaling law $R_g^2 \propto N^{2\nu}$ with $\nu = 1/2$. The scaling exponent, $\nu = 1/2$, is that of an ideal random walk. For real polymer chains, repulsive interactions between segments that are distant along the chain cause the polymer to swell[9]. This is known as the excluded volume effect. A random walk under excluded volume conditions is known as a self-avoiding walk (SAW), and results in an increase in the scaling exponent[10] $\nu_{excl.vol.} \approx 3/5$.

For linear polymer chains in the melt excluded volume effects become "screened out."[7] The screening of excluded volume is due to repulsive interactions with monomers from *other* chains, which exactly cancel out the swelling effects due to the intra-chain excluded volume interactions. Under conditions in which excluded volume is fully screened the scaling exponent regains its ideal value $\nu = 1/2$. Thus

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

it is said that linear polymer chains in the melt resemble a random walk.

Dynamics

For linear chains in the melt the self-diffusion coefficient D and zero-shear coefficient of viscosity η_o follow power law scaling with the degree of polymerization. A truly unique property of linear-chain melts comes from the dramatic change in the scaling laws for these transport coefficients at a critical, crossover molecular weight N_c. This crossover transition is marked by the following change in the scaling laws: for chains with $N < N_c$, $D \propto N^{-1}$ and $\eta_o \propto N$; whereas for $N > N_c$, $D \propto N^{-2}$ and $\eta_o \propto N^{3.4}$. This transition in the dynamical scaling laws is observed in melts and concentrated solutions of all flexible, linear polymer chains, independent of the chemical identity of the constituent monomers.

Qualitatively, the observed transition in the dynamics of linear-chain melts is said to result from the onset of intermolecular "entanglements." The exact nature of an entanglement is not well defined; however, it is undoubtedly a consequence of chain uncrossability. Uncrossability refers to the fact that polymer chains have short-range repulsive forces which prevent any two chains from passing through one another. Hence entanglements are classified as "topological constraints" on polymer motion. It will be shown in the next Chapter that removal of the constraint of uncrossability in an entangled melt results in complete recovery of unentangled dynamics.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2.3 Entanglements & reptation theory

The challenge that any theoretical treatment of linear polymer chains must meet is the description of the phenomenon of entanglement. Several different theoretical approaches have been proposed[11–14]. One treatment in particular that has had success in predicting most dynamical attributes of entangled polymer liquids is reptation theory.

Reptation theory postulates that an entangled polymer chain in the melt is entrapped inside an imaginary "tube," the size and shape of which are determined by the topological constraints imposed by surrounding chains. A given polymer chain must then execute motions consistent with confinement within the tube. This amounts to suppression of transverse motions, and a center-of-mass displacement that occurs through longitudinal "slithering" motions along the tube contour. De Gennes [11] and Doi and Edwards [4] have constructed an elegant, detailed theory based upon reptative assumptions.

Strict reptation predicts a scaling of the self-diffusion coefficient $D \propto N^{-2}$, and of the zero-shear coefficient of viscosity $\eta_o \propto N^3$; in close agreement with the generally accepted scaling laws determined from experiment. With a few exceptions, reptation theory is able to predict the experimentally observed dynamical properties of entangled melts of linear polymer chains. In short, it forms the foundation for the present understanding of linear-chain melt dynamics.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reptation theory is appealing due its inherent simplicity: it reduces a complicated, many-body problem to the problem of a single chain executing a special type of curvilinear motion. Yet despite its appeal and many successes, reptation theory remains controversial and no universally accepted theoretical treatment of entangled polymer melts exists. Objections to reptation theory call into question the validity of its dramatic simplification of an inherently many-body problem. In particular, the assumption of the phenomenon of entanglement, which reptation theory presents without any justification for why such entanglements occur. In early simulations of entangled linear polymer chains[15,16] no evidence was found to support the basic assumption of reptation theory that entangled polymer chain motion is highly anisotropic. A later simulation[17] demonstrated primitive chain motion which appeared to be confined to within a tube of a diameter estimated from monomer mean-square displacement. More recently, it has been shown that incorporation of an enhanced chain stiffness, the presence of which does not disrupt the overall Gaussian statistics of the chain, results in observable anisotropic polymer chain motion[18].

One justification for the study of non-linear polymer architectures such as rings and stars is to investigate the validity of reptation theory. The properties of entangled polymer melts are strongly dependent upon the architectural type of the constituent polymer chains. For instance, the viscosity of an entangled melt can be drastically altered by simply modifying the number of branching points

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

in the polymer molecules[19]. Invoking concepts of reptation, one might envision that each polymer architecture has a unique type of characteristic motion that enables it to escape from the reptative tube. The different characteristic motions lead to different predictions for the molecular-weight dependence of the transport properties which are specific to each architecture. Thus the goal is to use the ideas of reptation theory to make predictions for the dynamics of entangled melts of non-linear polymer chains, and to then test the predictions against data from experiment, or simulation, in order to comment on the possible existence of any reptative tube.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 3

Simulation Model

3.1 Introduction

Computer simulation has played a valuable role in the development of our understanding of the dynamical properties of linear polymer chains[20,21]. A variety of different types of simulation models have been proposed, each having their own unique advantages and disadvantages[22,23]. As the goal here is to investigate dense multi-chain systems of large polymer chains we have chosen a simulation method applicable to this. Using a lattice representation of polymer chains, we implement a modified version of the bond fluctuation model (BFM). The BFM uses a dynamic Monte Carlo (MC) algorithm in which polymer chain motion occurs through local monomer jumps. For any monomer jump, the lengths of the bonds connecting the monomer to the rest of the polymer chain are allowed to vary; hence the name "bond fluctuation." The original version of BFM was developed

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

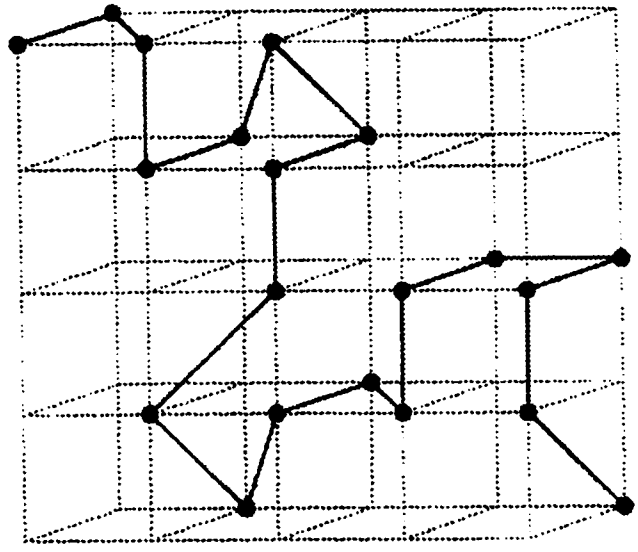

Figure 3.1: Shown is a depiction of a linear polymer chain ($N = 20$) on the lattice. Picture taken from Ref.[25].

by Carmesin and Kremer[24]. The modified version of BFM we use here was developed by J. S. Shaffer[25]. In Shaffer's version of BFM, each monomer occupies a single lattice site, as opposed to the original BFM in which monomers occupy an entire unit cell. The number of allowed bond vectors is also much smaller in Shaffer's version.

The specifics for this implementation of BFM are as follows. The polymer beads are placed on a simple cubic lattice (Fig. 3.1) with periodic boundary conditions in place in all coordinate directions (note that the polymer "beads" in this model are actually Kuhn statistical segment lengths). Excluded volume is enforced through a single-occupancy condition on each lattice site, *i.e.* only one

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

bead may occupy a given site at any given time. To connect the beads, we use bonds which are restricted to a set of three possible lengths: 1, $\sqrt{2}$, and $\sqrt{3}$; here the units are the lattice spacing. We control the crossing of bonds by monitoring a sub-lattice of bond midpoints. By virtue of the model construction any bond crossing must proceed through a configuration in which bond mid-points intersect. This makes it a simple matter to control the ability of chains to pass through one another (a fact which will be used to disable the topological interactions in ring polymers). Simulations in which bonds are *allowed* to pass through one another will be termed "crossing" throughout this text, and like-wise simulations for which bond crossing is *forbidden* will be termed "non-crossing."

The motion of polymer chains is achieved by attempting to displace a randomly selected bead to a randomly selected nearest-neighbor lattice site. The move is accepted if it does not violate the excluded volume or allowed bond length constraints, or (for non-crossing simulations) the bond-crossing restriction, and rejected if it does. $N \times n_p$ attempted moves constitutes one Monte Carlo time step (MCS), which is hereafter used as the time unit (N is the number of beads per chain and n_p is the total number of polymers in the simulation box). As we use a MC algorithm it is particularly important that a robust random number generator be used. For this purpose we chose the function *ran3*. A description of this random number generator and its source code can be found in Ref.[26]. We list the entire source code we have used for the ring and star simulations in Appendices

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A and B.

All simulations were performed with a bead density of $\phi = 0.5$. In a detailed study Shaffer has shown that the salient features of polymer melts are captured at this density[25,27]. In particular it has been shown that excluded volume is being screened on all length scales greater than a few monomer segments. Secondly, it has been shown that there is an entanglement transition in the dynamics, which for this model occurs at $N_c \approx 40$. Finally, for intermediate wavelengths it has been found that the Rouse mode relaxation times scale with mode index as $\tau_p \propto 1/p^3$. This same scaling has been found in molecular dynamics simulations by Kremer *et al*[28].

Generally speaking, one advantage to using lattice models is that the integer arithmetic involved is typically faster than the floating-point calculations that must be done for off-lattice models. An advantage to this version of BFM is that it is very efficient to maintain excluded volume by monitoring the lattice site occupation. Another advantage is the relatively low value for the critical chain length at which the crossover to entangled dynamics occurs for linear chains. This allows simulations to be done for chains with a relatively high degree of entanglement (more precisely, with relatively large N/N_c). This is advantageous as we wish to investigate melts of non-linear chains with sizes for which the corresponding linear chains would be highly entangled.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

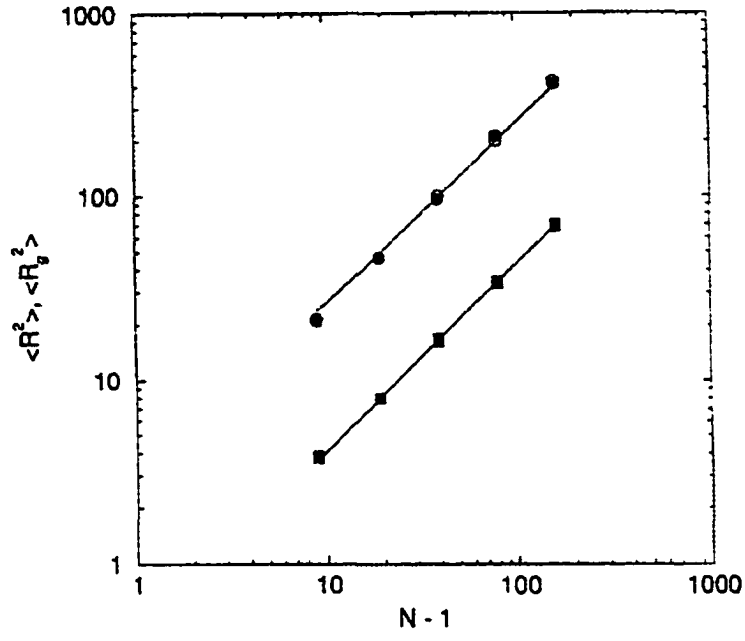

Figure 3.2: Mean-square end-to-end distance $\langle R^2 \rangle$ (circles) and mean-square radius of gyration $\langle R_g^2 \rangle$ (squares) versus number of bonds $N-1$ for linear chains in the melt (graph taken from Ref.[25]). The filled symbols are data for non-crossing chains. The open symbols are data for crossing chains, the values of which are virtually superimposed with the non-crossing data. The solid lines indicate the power law fits $R_g^2 \propto R^2 \propto N$.

3.2 Linear chains

Shown in Fig. 3.2 is the N-dependence for the two customary measures of the average size of linear polymer chains: the mean-square end-to-end distance R^2; and the mean-square radius of gyration R_g^2. Both follow the power law scaling expected for an ideal random walk $R_g^2 \propto R^2 \propto N$. Note that data for both crossing and non-crossing linear chains are shown in the plot.

Fig. 3.3 shows the scaling of the center-of-mass self-diffusion coefficient for

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

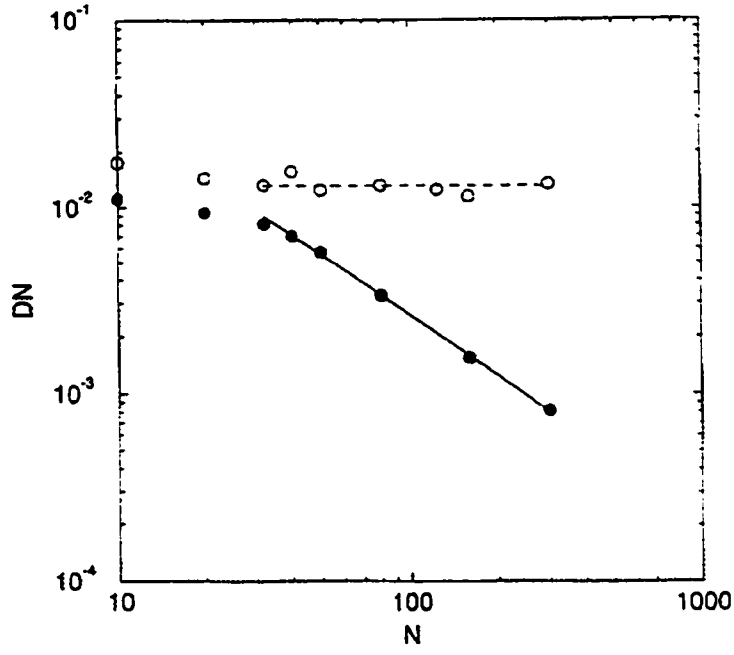

Figure 3.3: Center-of-mass self-diffusion coefficient times number of monomers DN versus number of monomers N for linear chains in the melt (graph taken from Ref.[25]). The filled symbols are data for the non-crossing chains, and the open circles are the data for crossing chains. The solid line shows the power law scaling $DN \propto N^{-1}$, indicative of the onset of entangled dynamics. The dashed line shows the unentangled scaling $DN \approx const.$

both crossing and non-crossing linear chains. For the non-crossing chains there is a crossover transition to an entangled dynamics which occurs at $N_c \approx 40$. Note that there is no such transition for the crossing chain data. As was claimed in the previous Chapter, removal of the constraint of uncrossability in an entangled melt results in the complete recovery of unentangled dynamics.

In order to simulate non-linear polymer chains certain modifications to the simulation construction for linear chains were required. In the next two sections the specific details pertinent to the simulations of rings and stars are presented.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.3 Rings

In ring polymer simulations (and experiments) it is critical to avoid configurations that have self-knots and concatenations. The non-trivial effects of such configurations have been previously shown[29]. To avoid self-knots and concatenations we begin all ring polymer simulations with initial configurations in which rings are isolated from one another. This requires placing the rings on the lattice as compact thin loops. As a consequence of this the necessary equilibration time is increased.

To relax the ring polymer initial configurations we perform equilibration runs for more than 20 (smaller rings) to about 4 (800mers) orientational relaxation times. Note that this requires some guesswork as the relaxation times are determined with the data acquired from production runs; however, center-of-mass mean-square displacement and monomer pair-correlation functions can be monitored "on the fly" to evaluate the extent of equilibration.

We performed simulations of both crossing and non-crossing rings in the melt. Additionally, we performed simulations for crossing and non-crossing single (isolated) rings. The data for the melt and single ring simulations are shown in Tables 3.1 and 3.2, respectively.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 3.1: Relevant simulation parameters and results for rings in the melt: degree of polymerization N; number of ring polymers in simulation box n_{ring}; number of independent simulation runs n_{run}; side length of simulation box L; equilibration time τ_{eq}; run time τ_{run}; mean-square radius of gyration R_g^2; mean-square ring diameter R_e^2; self-diffusion coefficient D; orientational relaxation time τ_{ee}. All time values are measured in Monte Carlo time steps, and all distances are in units of the lattice constant.

non-crossing									
N	n_{ring}	n_{run}	L	τ_{eq}	τ_{run}	R_g^2	R_e^2	D	τ_{ee}
10	400	6	20	2.5×10^4	2.5×10^4	2.1	6.6	1.1×10^{-3}	2.4×10^2
20	200	6	20	5×10^4	5×10^4	4.1	12.8	5.1×10^{-4}	1.0×10^3
40	100	6	20	5×10^4	1×10^5	8.1	24.5	2.2×10^{-4}	4.1×10^3
100	135	6	30	1×10^6	1×10^6	18.7	55.3	7.0×10^{-5}	3.0×10^4
150	90	6	30	2.5×10^6	2.5×10^6	26.8	78.3	3.8×10^{-5}	7.0×10^4
200	160	3	40	5×10^6	5×10^6	34.4	100	2.6×10^{-5}	1.3×10^5
300	45	6	30	1×10^7	1×10^7	48.4	140	1.3×10^{-5}	4.0×10^5
500	64	3	40	2×10^7	1×10^8	73.8	210	6.0×10^{-6}	1.4×10^6
800	40	4	40	1.9×10^7	2×10^8	105	297	2.7×10^{-6}	4.9×10^6

crossing									
N	n_{ring}	n_{run}	L	τ_{eq}	τ_{run}	R_g^2	R_e^2	D	τ_{ee}
10	400	6	20	2.5×10^4	2.5×10^4	2.13	6.74	1.4×10^{-3}	1.8×10^2
20	200	6	20	2.5×10^4	2.5×10^4	4.37	13.6	6.8×10^{-4}	7.5×10^2
40	100	6	20	5×10^4	1×10^5	8.88	27.9	3.3×10^{-4}	2.9×10^3
100	135	6	30	1×10^6	1×10^6	22.5	70.3	1.3×10^{-4}	1.9×10^4
300	45	3	30	5×10^6	1×10^7	68.8	212	4.1×10^{-5}	1.8×10^5
500	64	2	40	6×10^6	2.5×10^7	114	350	2.4×10^{-5}	5.0×10^5

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 3.2: Relevant simulation parameters and results for isolated rings: degree of polymerization N; number of independent simulation runs n_{run}; side length of simulation box L; equilibration time τ_{eq}; run time τ_{run}; mean-square radius of gyration R_g^2; mean-square ring diameter R_e^2; self-diffusion coefficient D; orientational relaxation time τ_{ee}. All time values are measured in Monte Carlo time steps, and all distances are in units of the lattice constant.

non-crossing								
N	n_{run}	L	τ_{eq}	τ_{run}	R_g^2	R_e^2	D	τ_{ee}
10	400	40	2×10^4	2×10^4	2.27	7.37	4.1×10^{-3}	72
20	200	40	2×10^4	2×10^4	4.96	16.1	2.0×10^{-3}	3.9×10^2
40	100	40	8×10^4	8×10^4	11.0	35.7	1.0×10^{-3}	2.0×10^3
100	135	50	4×10^5	4×10^5	32.1	104	4.2×10^{-4}	1.4×10^4
300	40	60	1×10^6	5×10^6	116	379	1.4×10^{-4}	1.6×10^5
500	64	100	4×10^6	2×10^7	210	683	8.4×10^{-5}	4.0×10^5
crossing								
N	n_{run}	L	τ_{eq}	τ_{run}	R_g^2	R_e^2	D	τ_{ee}
10	400	40	2×10^4	2×10^4	2.24	7.21	4.6×10^{-3}	1.1×10^2
20	200	40	2×10^4	2×10^4	4.86	15.6	2.2×10^{-3}	3.1×10^2
40	100	40	8×10^4	8×10^4	10.8	35.0	1.1×10^{-3}	1.6×10^3
100	135	50	4×10^5	4×10^5	31.2	101	4.5×10^{-4}	1.0×10^4
300	40	60	1×10^6	5×10^6	113	368	1.6×10^{-4}	1.0×10^5
500	64	100	4×10^6	2×10^7	204	661	9.1×10^{-5}	3.2×10^5

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3.4 Three-arm stars

To inspect the feasibility of using Shaffer's version of BFM for three-arm stars (for simulations of stars using the original BFM see Ref. [24]), we initially simulated single stars. It should be emphasized here that the branching point (center-bead) moves according to the same rules as other beads (*i.e.* no special moves are necessary). Once the model had been checked for isolated stars, we proceeded to simulate melts of star polymers. Data from all star runs is shown in Table 3.3.

For computational convenience we use stars with slightly different arm lengths. In all simulations star polymers have two arms of length $N_{arm1} = N_{arm2} = N_{arm}$ and one arm of length $N_{arm3} = N_{arm} - 1$. Thus the total number of monomers in a star polymer is $N = 1 + N_{arm1} + N_{arm2} + N_{arm3} = 3 \times N_{arm}$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 3.3: Relevant simulation parameters and results for star polymer simulations: number of monomers per arm N_{arm}; number of star polymers in simulation box n_{star}; number of independent simulation runs n_{run}; side length of simulation box L; equilibration time τ_{eq}; run time τ_{run}; mean-square radius of gyration R_g^2; mean-square distance from center bead to arm end bead R_{ce}^2; mean-square distance between arm end beads R_{ee}^2; self-diffusion coefficient D; orientational relaxation time τ_{ce}. All time values are measured in Monte Carlo time steps, and all distances are in units of the lattice constant.

N_{arm}	n_{star}	n_{run}	L	τ_{eq}	τ_{run}	R_g^2	R_{ce}^2	R_{ee}^2	D	τ_{ce}
10	1	50	30	2.5×10^4	1×10^5	11.4	21.7	56.9	1.4×10^{-3}	1.1×10^3
20	1	50	30	5×10^4	2×10^5	25.3	49.4	129	7.0×10^{-4}	4.9×10^3
30	1	50	30	1×10^5	4×10^5	40.6	79.7	210	4.6×10^{-4}	1.2×10^4
50	1	50	50	1.5×10^5	5×10^5	74.9	148	395	2.8×10^{-4}	4.7×10^4
90	1	50	50	3×10^5	1.5×10^6	147	294	768	1.6×10^{-4}	1.5×10^5
N_{arm}	n_{star}	n_{run}	L	τ_{eq}	τ_{run}	R_g^2	R_{ce}^2	R_{ee}^2	D	τ_{ce}
10	450	2	30	5×10^5	1×10^6	9.7	23.2	47.6	2.7×10^{-4}	3.9×10^3
20	100	3	20†	5×10^5	1×10^6	20.1	49.3	101	9.7×10^{-5}	2.4×10^4
30	150	3	30	5×10^5	5×10^6	30.3	75.6	153	4.8×10^{-5}	6.6×10^4
50	90	2	30	1×10^6	1×10^7	51.2	128	259	1.6×10^{-5}	2.9×10^5
90	50	3	30	5×10^6	2×10^8	93.1	235	475	4.6×10^{-6}	1.8×10^6
150	64	2	40‡	2×10^7	4×10^8	157	398	801	1.3×10^{-6}	1.3×10^7

†$L_x = L_y = 20$, $L_z = 30$; ‡$L_x = L_y = 40$, $L_z = 36$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 4

Ring Polymers

4.1 Introduction

This chapter presents the results of simulations for ring polymers both in the melt and as isolated chains on the lattice. For melt and single-ring systems both crossing and non-crossing simulations were performed, the results of which are both presented here. A limited number of linear-chain simulations were done for the sake of comparison (and also to initially check the simulation code): however, the majority of linear-chain simulation data used here for comparison is taken from the published works of Shaffer[25,27]. The results of this Chapter have been published[30,31] or submitted for publication[32].

The original motivation for simulating ring polymers was to study other non-reptative modes of motion. To this end, we searched for a polymer chain in which reptation would not be the fastest mode of relaxation. Polymer rings have no

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

chain ends and hence cannot reptate in the original sense of reptation theory: slithering motions are only possible when rings adopt special double-stranded (pseudo-linear) configurations. These double-stranded structures were termed "non-ramified" configurations by Klein[33]. To adopt a non-ramified configuration both halves of a ring must follow the same path through space. This condition greatly reduces the likelihood for the formation of non-ramified configurations: the probability of adopting such a double-stranded configuration decreases exponentially with ring size. It was argued by Klein that this leads to an exponential dependence on molecular weight for both the self-diffusion coefficient and longest relaxation time for large rings in the melt.

However, existing experimental evidence for rings in the melt is not able to substantiate this conjecture. For example, McKenna *et al.*[34] find very similar dynamics for rings and linear chains in the melt. They also report a molecular weight dependence for the viscosity of rings in the melt which shows a similar transition to that of linear chains. They estimate the crossover molecular weight of rings to be about a factor 2 larger than that of linear chains. A more recent study by Tead *et al.*[35] found that tracer diffusion of linear chains in linear chain and ring matrices was nearly identical. Additionally, tracer diffusion coefficients for rings in linear chain matrices were found to be close to those expected for linear tracers of similar molecular weight. Preliminary study of ring tracer diffusion in ring matrices was mentioned in which diffusion coefficients were again close to those

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of corresponding linear chains. Tead *et al.* interpreted these results as supporting the hypothesis that ring polymers can, in some unspecified way, reptate.

In contrast to this, the main result of the ring polymer simulations presented herein is that rings in the melt have much faster dynamics than linear chains of comparable degree of polymerization. Moreover, we show that faster ring dynamics persists well into the entanglement regime for linear chains: rings have significantly faster dynamics even for ring sizes which are 20 times greater than the entanglement crossover chain length for linear chains. We rationalize that faster ring dynamics is a consequence of their smaller size: dependence of the self-diffusion coefficient and single-chain relaxation time on the radius of gyration of rings and linear chains is remarkably similar. Additionally, we show that for our largest ring sizes ($N \geq 300$) there is evidence for the beginning of a transition to a new dynamical regime characterized by a new time scale.

In an interesting simulation study using the original BFM, Müller *et al.*[36] showed that ring diffusion in a ring melt is faster than that of linear chains (in a linear chain matrix) of the same degree of polymerization. Their analysis was limited to degrees of polymerization smaller than the entanglement crossover degree of polymerization for linear chains. In an earlier study[37] it was found that the self-diffusion coefficients for rings were smaller than those for linear chains: however, a "cooperative rearrangement" algorithm was used that is very different from the model used by Müller *et al.* and the one employed herein.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The organization of this Chapter is as follows. The results for the equilibrium and dynamical properties of crossing and non-crossing rings in the melt are presented first. Following this, the results for crossing and non-crossing single-ring simulations are briefly summarized. The reason for performing the single-ring simulations was simply to contrast the effects of topological constraints on isolated rings with that of rings in the melt, and as such they require only brief presentation.

4.2 Rings in the melt

Table 3.1 lists the main ring polymer simulation parameters and results for crossing and non-crossing rings in the melt.

4.2.1 Equilibrium properties

Non-crossing rings

Fig. 4.1 shows a typical conformation of a ring and linear chain taken from equilibrated melt configurations. At least qualitatively, it can be seen from Fig. 4.1 that the linear chain conformation has a much more open and extended structure.

However, we require quantifiable measure of the equilibrium properties, and to this end we introduce the following quantities. As a measure of average chain size we use the mean-square radius of gyration R_g^2, and the mean-square diameter vector R_e^2. The diameter vector is the ring analog of the end-to-end vector in

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

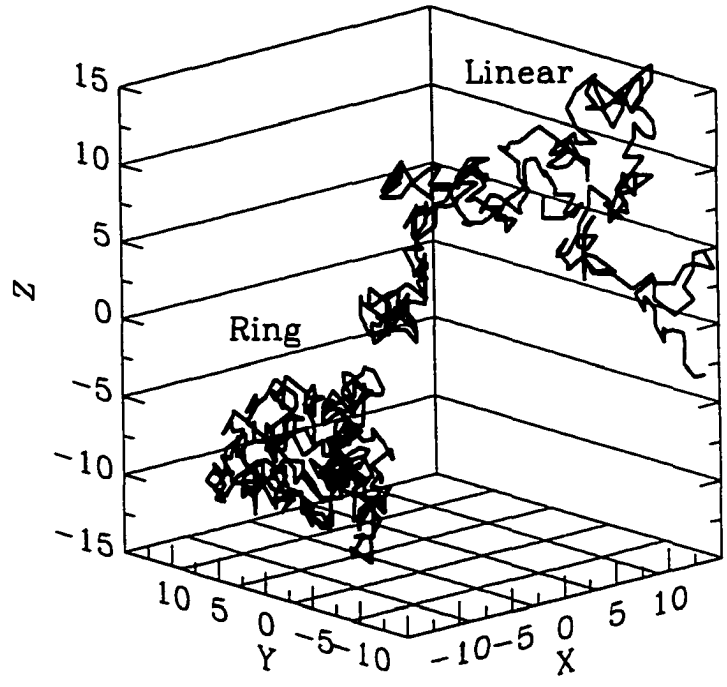

Figure 4.1: Example conformations for non-crossing 300mer ring and 300mer linear chain (taken from equilibrated melt configurations).

linear polymers and is defined by

$$R_e^2 = \frac{1}{n_{ring}} \frac{2}{N} \sum_{\alpha=1}^{n_{ring}} \sum_{i=1}^{N/2-1} \left\langle \left(\mathbf{r}_i^\alpha - \mathbf{r}_{i+N/2}^\alpha \right)^2 \right\rangle ; \tag{4.1}$$

where α is a sum over the total number of ring polymers n_{ring}. Other quantities of use in analyzing the equilibrium structure are several pair-correlation functions: the intra-chain monomer pair-correlation function $g_{intra}(r)$

$$\phi g_{intra}(r) = \frac{1}{n_{ring}} \frac{1}{N} \sum_{\alpha=1}^{n_{ring}} \sum_{i \neq j=1}^{N} \left\langle \delta \left(\mathbf{r} + \mathbf{r}_i^\alpha - \mathbf{r}_j^\alpha \right) \right\rangle ; \tag{4.2}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the inter-chain monomer pair-correlation function $g_{inter}(r)$

$$\phi g_{inter}(r) = \frac{1}{N n_{ring}} \sum_{\alpha \neq \beta = 1}^{n_{ring}} \sum_{i,j=1}^{N} \left\langle \delta \left(\mathbf{r} + \mathbf{r}_i^{\alpha} - \mathbf{r}_j^{\beta} \right) \right\rangle ; \tag{4.3}$$

lastly, we have the center-of-mass pair-correlation function $g_{cm}(r)$

$$\rho g_{cm}(r) = \frac{1}{n_{ring}} \sum_{\alpha \neq \beta = 1}^{n_{ring}} \left\langle \delta \left(\mathbf{r} + \mathbf{R}_{\alpha} - \mathbf{R}_{\beta} \right) \right\rangle , \tag{4.4}$$

where $\mathbf{R}$ is the position of the center of mass. In the above equations ϕ is the monomer density and ρ is the density of polymer chains. Finally, we measure the intra-chain static structure factor

$$S(k) = \frac{1}{n_{ring}} \frac{1}{N} \sum_{\alpha=1}^{n_{ring}} \left\langle \sum_{i,j=1}^{N} exp\left(-i\mathbf{k} \cdot \left(\mathbf{r}_i^{\alpha} - \mathbf{r}_j^{\alpha} \right) \right) \right\rangle . \tag{4.5}$$

$g_{intra}(r)$ is a measure of the probability of finding a second monomer (belonging to the same chain) a distance r away given that the first monomer lies at the origin ($r = 0$). Thus $g_{intra}(r)$ measures the pair-wise distribution of intra-chain monomer density. Its counterpart $g_{inter}(r)$ gives the probability of finding a second monomer (belonging to a *different* chain) a distance r away given the first monomer lies at the origin. This quantity provides a measure of the distribution of inter-chain monomer density, and hence contains information on the correlation hole. Every monomer in the melt is surrounded by a correlation hole in which the concentration of monomers from other chains is reduced. The size of the correlation hole is comparable to the radius of gyration.

The quantity $g_{cm}(r)$ is a measure of the probability of finding a second polymer center of mass a distance r away given the first polymer center of mass lies at

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the origin. $g_{cm}(r)$ thus provides a qualitative indication of the degree of interpenetration present in the system.

Finally, $S(k)$ measures the intra-chain density fluctuations in k-space. $S(k)$ can be determined experimentally by inelastic neutron scattering[10]. $S(k)$ is proportional to $k^{-1/\nu}$ for intermediate values of k, and so can be used as another measure of the scaling exponent ν.

Shown in Fig. 4.2 is the scaling of the mean-square radius of gyration with degree of polymerization for rings and linear chains. From the plot, it can be seen that for a given size N rings have a smaller radius of gyration than linear chains. Additionally, the radii of gyration for rings and linear chains follow different scaling laws. For ideal (Gaussian) rings and linear chains, the ratio of the radii of gyration is expected to equal $\sqrt{2}$, independent of N.

The scaling exponent ν, defined through the relation $R_g \propto N^{\nu}$, has a smaller value for rings than the ideal linear chain value of 1/2. A fit to the $N > 100$ ring data for R_g^2 yields $\nu = 0.41 \pm 0.01$. The scaling exponent is obtained through a linear least-squares fit of the *log* of the data, that is, $logR_g = \nu logN + const.$

The scaling exponent we obtain agrees with the value found in the ring simulations of Müller *et al.*[36]. A fit to the ring diameter vector R_e (Fig. 4.3) yields the scaling exponent $\nu = .40 \pm .01$. This value is slightly lower than the value obtained from the R_g^2 fit; however, it is in perfect agreement with the theoretical prediction of Cates and Deutsch[38]. Note that a careful finite-size analysis by

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Müller *et al.* results in values which are also in perfect agreement with the theory. We expect that similar analysis of our R_g^2 data would produce agreement between its exponent and the exponent obtained from the R_e^2 fit. Note that R_e^2 is the more averaged quantity here. There are several measurements of R_e^2 possible per individual ring, whereas for R_g^2 there is only one measurement possible per ring. It is interesting to note that although R_g^2 and R_e^2 reveal non-Gaussian statistics, their ratio is close to the Gaussian value of 1/3.

As rings are smaller on average than linear chains they must be more dense. Therefore we expect that their intra-chain correlations should be characterized by larger values (higher correlations) for small r. Moreover, we expect that the correlation hole in the inter-chain correlation function should be more pronounced for rings than for linear chains. This follows from the fact that the total monomer pair-correlation function, *i.e.* the sum of intra- and inter-chain contributions, is expected to be roughly the same for rings and linear chains.

Fig. 4.4 shows results for the monomer pair-correlation functions $g_{intra}(r)$ and $g_{inter}(r)$. We find that the correlation hole is deeper and wider for 300mer rings than for 300mer linear chains. In addition we find that the 300mer ring correlation hole is also deeper than for 100mer linear chains. This is significant as 300mer rings and 100mer linear chains have almost equal radii of gyration. We also find that rings have stronger intra-chain correlations. It should be noted that since rings are smaller on average than linear chains the range of the ring intra-chain

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

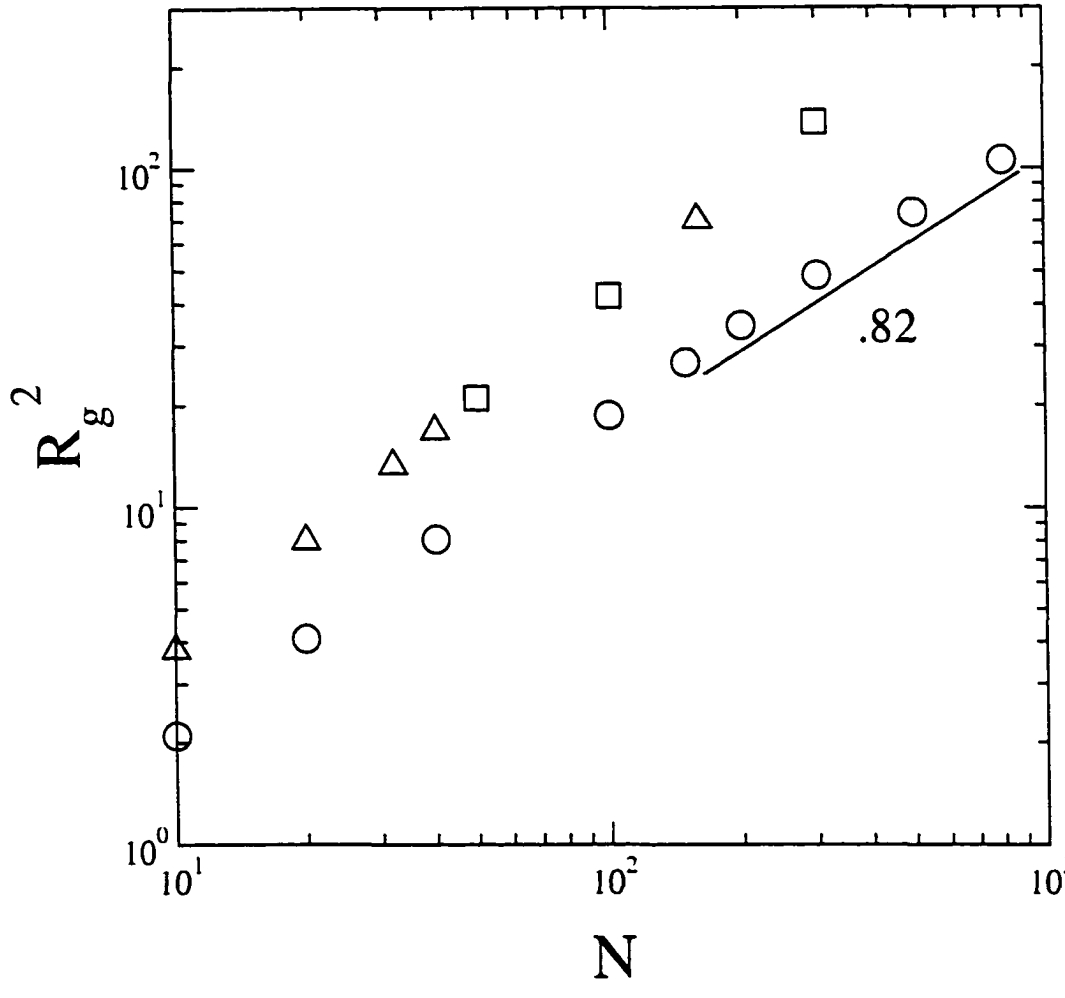

Figure 4.2: Mean-square radius of gyration R_g^2 versus degree of polymerization N for rings and linear chains in the melt. Circles: rings; triangles: linear chains, data from Ref.[25]; squares: linear chains, our data. The line indicates the power law fit to the ring data with scaling exponent indicated.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

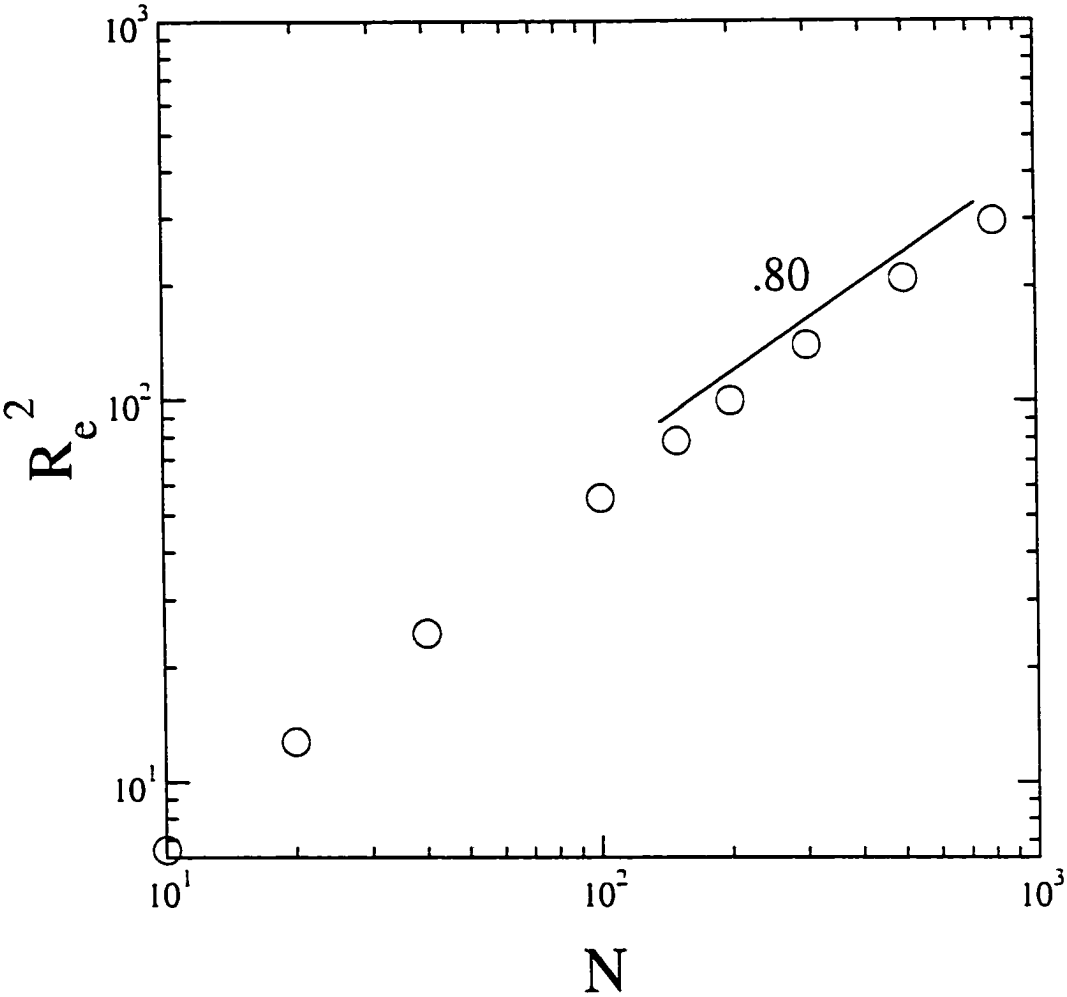

Figure 4.3: Mean-square diameter vector R_e^2 versus degree of polymerization N for rings in the melt. The line indicates the power law fit with scaling exponent indicated.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

correlation function is expected to also be smaller. Finally, in Fig. 4.5 we show that the correlation hole for rings deepens and widens with increasing degree of polymerization.

Rings are more compact, have deeper and wider correlation holes, and stronger intra-chain correlations. As a consequence of this, we expect them to have less inter-penetration than linear chains. It is difficult to quantify inter-penetration: however, we attempt to evaluate it by looking at the number of polymers in the correlation hole, $R_g^3\phi/N$. ϕ is the bead density, therefore $R_g^3\phi$ gives the number of beads present within the volume R_g^3, which, when scaled by the number of monomers per chain N, gives the average number of polymer chains within the correlation hole. In the large N regime, we find for rings $R_g^3\phi/N \approx 0.3N^{0.23}\phi$. Comparing this formula with the number of linear chains in the correlation hole at the linear-chain entanglement crossover, $R_g^3(N = 40) \times 0.5/40 \approx 0.86$, we see that in order to achieve the degree of inter-penetration of rings equal to that of linear chains at $N = N_{c,linear} \approx 40$ we would need to consider rings of approximately 2000 monomers!

To further quantify the degree of inter-penetration we monitor the center-of-mass pair-correlation function $g_{cm}(r)$. Fig. 4.6 indicates that both rings and linear chains have correlation holes in $g_{cm}(r)$; however, the hole for rings is deeper than for linear chains of roughly the same radius of gyration (*i.e.* 300mer rings and 100mer linear chains). Fig. 4.6 confirms the supposition that rings inter-penetrate

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

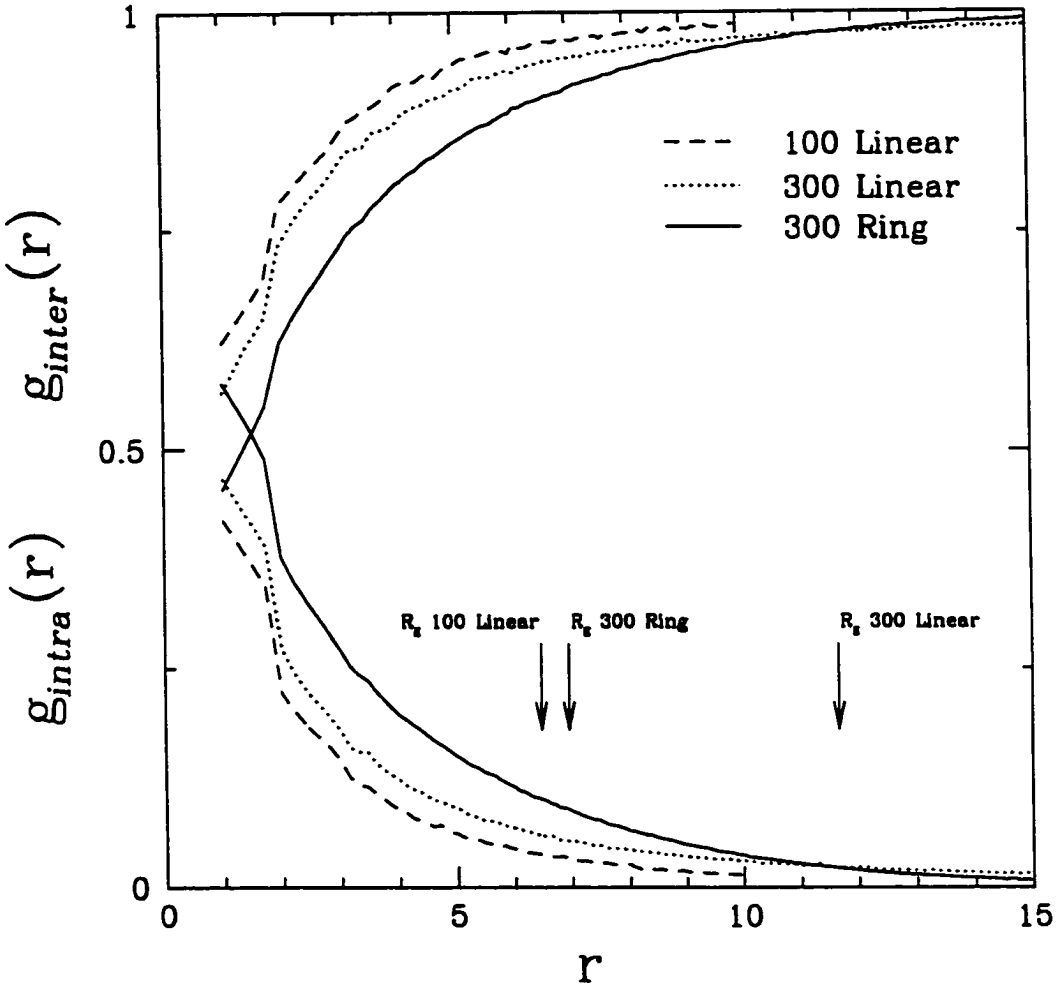

Figure 4.4: Monomer pair-correlation functions $g_{intra}(r)$ and $g_{inter}(r)$ versus separation distance r for rings and linear chains. Note that 300mer rings and 100mer linear chains have comparable radii of gyration (as indicated in plot).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

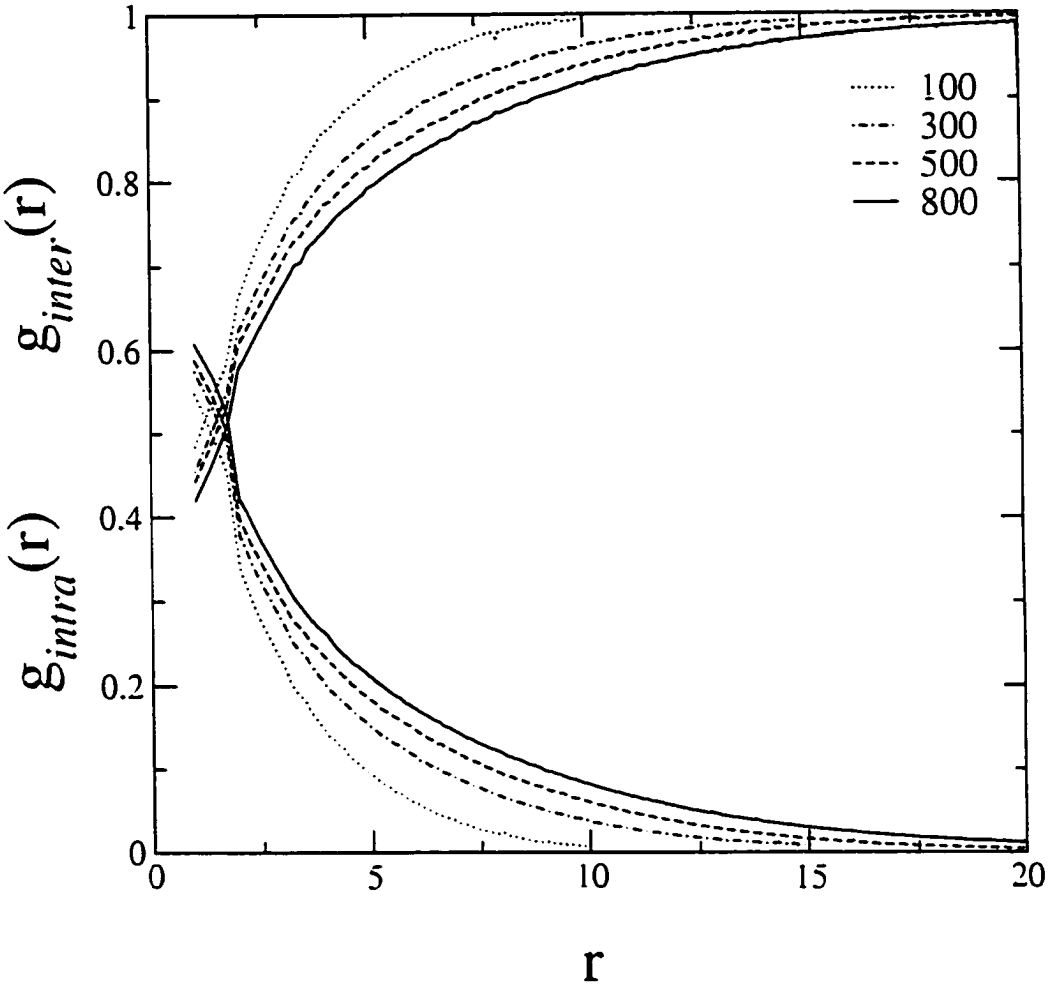

Figure 4.5: Monomer pair-correlation functions $g_{intra}(r)$ and $g_{inter}(r)$ versus distance r for rings of different degrees of polymerization (as indicated in the legend). The correlation hole in $g_{inter}(r)$ deepens and widens with increasing degree of polymerization.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

less than linear chains. It can be seen that the correlation hole in $g_{cm}(r)$ widens with increasing ring size, and also roughly scales with R_g (shown in the insert in Fig. 4.6).

Finally, in Fig. 4.7 we show the master curve for the intra-chain static structure factor $S(k)$. To obtain the curve we scale the structure factor by the degree of polymerization N, and multiply the wave-vector by the radius of gyration R_g. The results for differing degrees of polymerization superimpose well. The scaling part of the structure factor is compatible with the power law decay $S(k)/N \propto (kR_g)^{-1/\nu}$ with exponent $\nu \approx 0.4$. This provides another measure of the scaling exponent ν, which is in close agreement with the power law fits to R_g and R_e.

As a final note. because the non-crossing rings are initially fully compact (unit lattice density) and are allowed to expand for finite equilibration times. it might be claimed that they are residing in meta-stable conformations. To test for this possibility we performed a simulation of 200mers in which the rings were initially highly extended and entangled. During the simulation run we monitored R_g^2, the data for which is shown in Fig. 4.8. The highly extended rings collapse down and eventually acquire the same value for R_g^2 as the rings which were initially collapsed. Therefore, we conclude that the more collapsed ring structure is an inherent property of the non-crossing rings in the melt, and not result of any meta-stable state.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

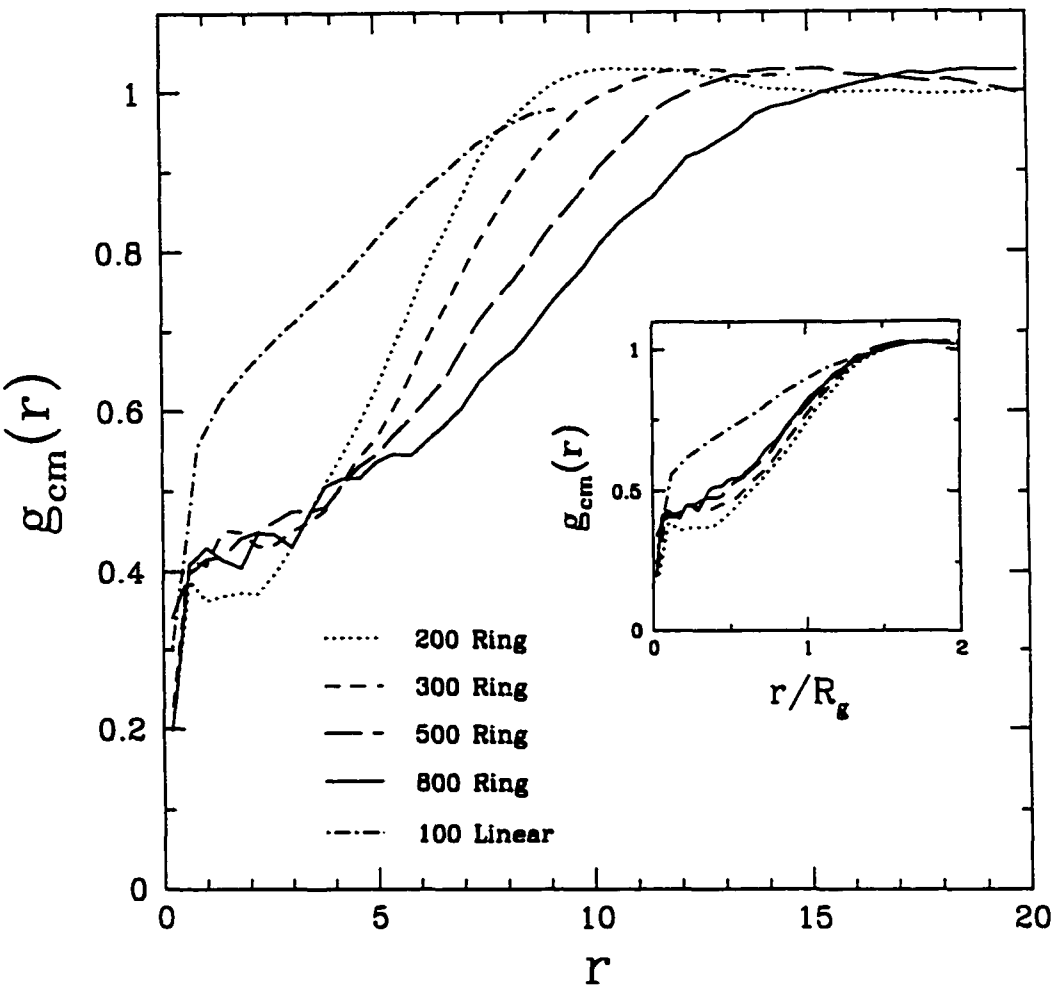

Figure 4.6: Center-of-mass pair-correlation function $g_{cm}(r)$ plotted versus separation distance r for non-crossing rings and a linear chain. The insert shows the same data with the distance r scaled by the radius of gyration.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

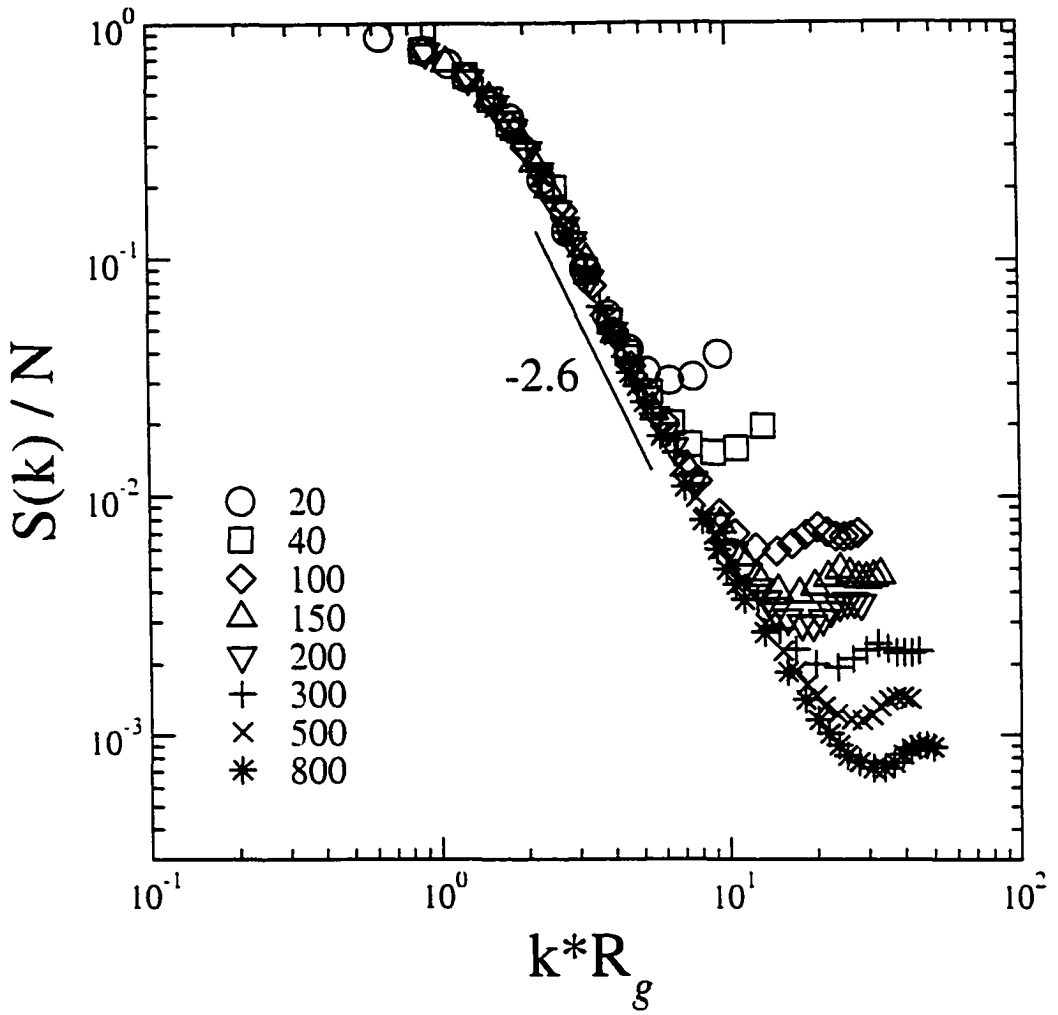

Figure 4.7: Static structure factor scaled by degree of polymerization $S(k)/N$ versus wave vector times the radius of gyration $k * R_g$ for non-crossing rings. In the intermediate scaling regime the plots reduce down to a single curve. The solid line shows the observed scaling: $S(k)/N \propto (kR_g)^{-2.6}$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

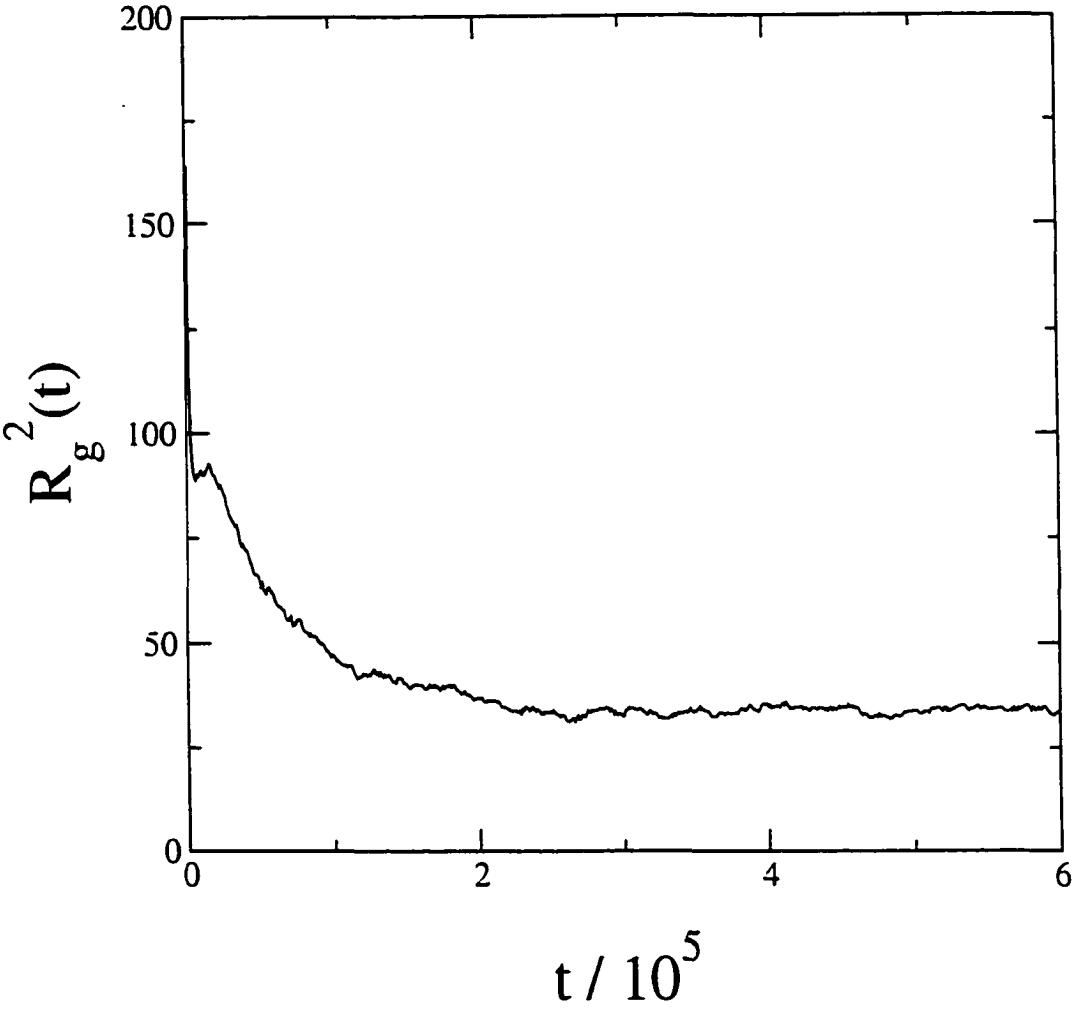

Figure 4.8: Mean-square radius of gyration $R_g^2(t)$ versus time step t for initially extended and entangled 200mer rings. The long-time value of $R_g^2(t)$ approaches the same value as is seen with the initially compact 200mer ring simulations. Note that the time axis has been rescaled merely for convenience.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

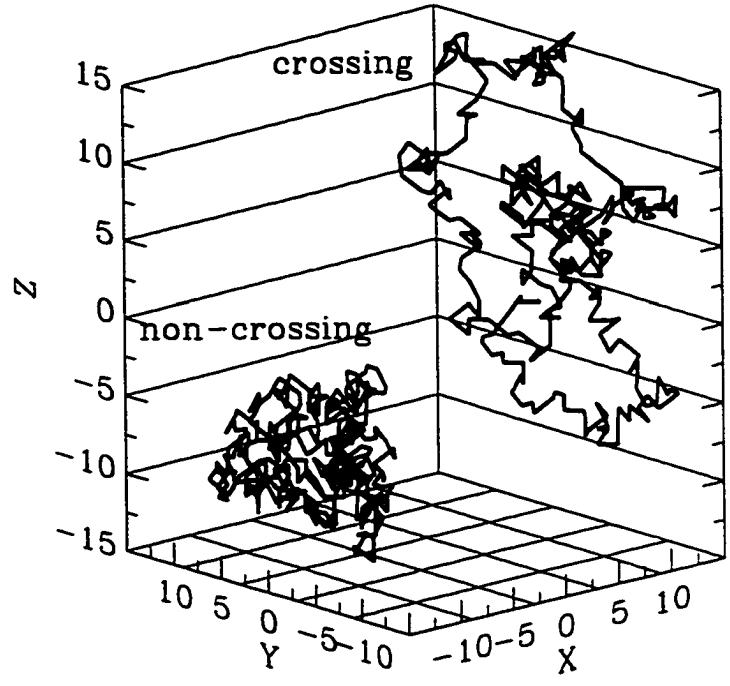

Figure 4.9: Typical conformations of crossing and non-crossing 300mer rings (taken from melt configurations).

Crossing rings

We now briefly look at ring polymer melts in which the constraint of non-concatenation has been removed, *i.e.* crossing rings in the melt. Fig. 4.9 shows the qualitative difference in conformational structure between crossing and non-crossing 300mer rings. The obvious difference between the two conformations is somewhat reminiscent of the ring / linear chain comparison in Fig. 4.1. The crossing ring appears to have a much more open and extended structure.

We find that in removing the constraint of non-concatenation we recover the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

equilibrium Gaussian statistics seen for linear chains in the melt. That is, for crossing rings in the melt we find that the radius of gyration scales as $R_g^2 \propto N$ (Fig. 4.10). Removal of the topological interactions has a significant effect on the average size of rings in the melt.

In Fig. 4.11, we plot the center-of-mass pair-correlation function for crossing rings. Also included in the figure are 500mer non-crossing rings and 100mer linear chains. The crossing rings have a much less pronounced correlation hole than the non-crossing rings. In fact the correlation hole for crossing rings is qualitatively similar to that for linear chains. The absence of the topological constraints frees the crossing rings to explore more extended conformations. The result is a more open ring structure, which allows for closer approach of the ring centers of mass. Shown in the insert in Fig. 4.11 is the same data plotted with the separation distance scaled by the radius of gyration. The crossing rings collapse onto the same curve as the linear chain data, whereas the non-crossing 500mer ring falls onto a curve that is quite separate from the rest.

Fig. 4.12 shows the reduced curve for the intra-chain static structure factor. The results for the smaller rings do not superimpose onto the master curve as well as the larger rings. This is not too surprising as the crossing rings are more swollen than non-crossing rings, and hence, at least for the smaller rings, feel the effect of having to form a loop more strongly than the non-crossing rings. For this reason the fit to the scaling part of the curve is done only for $N \geq 100$. The

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

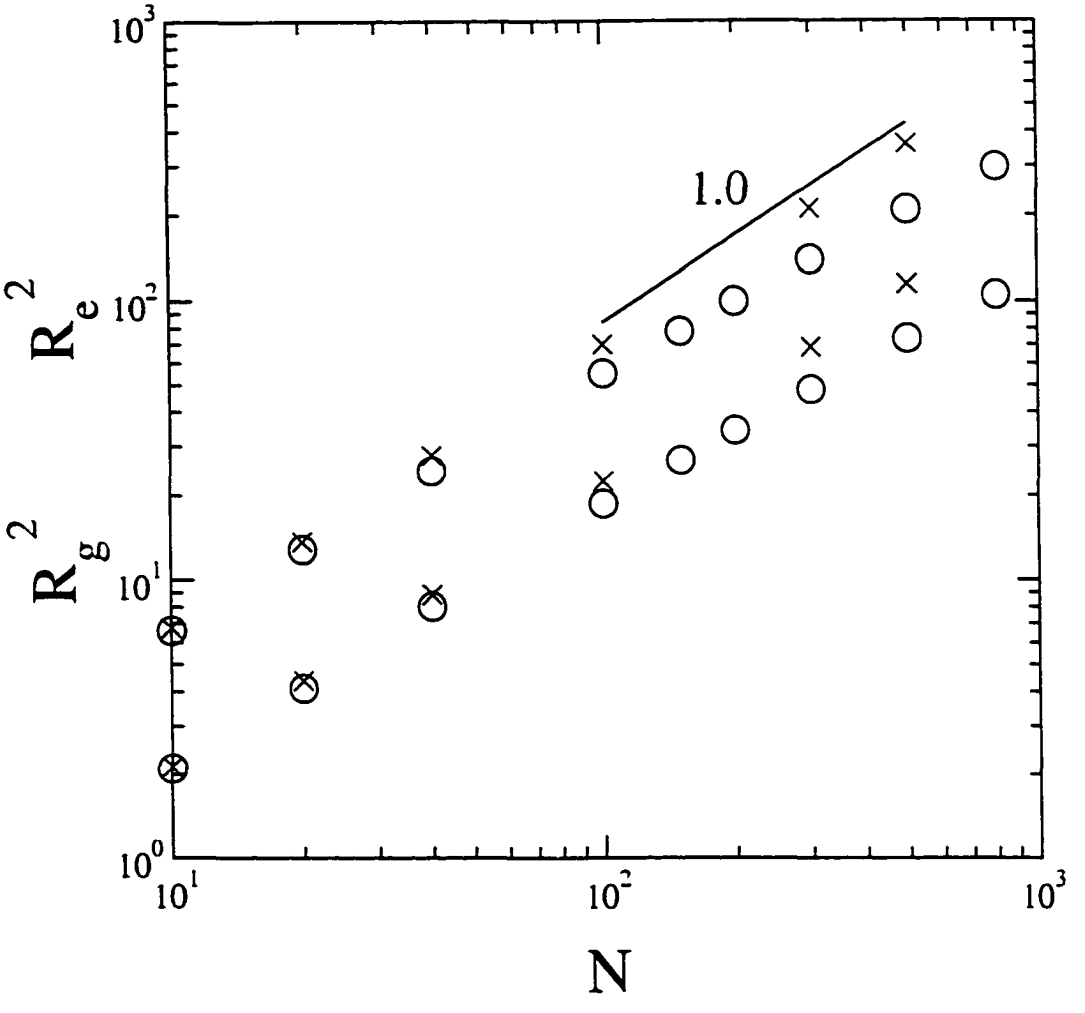

Figure 4.10: Mean-square radius of gyration R_g^2 and mean-square diameter vector R_e^2 versus degree of polymerization N for rings in the melt. Circles: non-crossing rings; crosses: crossing rings. The line indicates the power law fits to the crossing data for both R_g^2 and R_e^2 with scaling exponent indicated.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

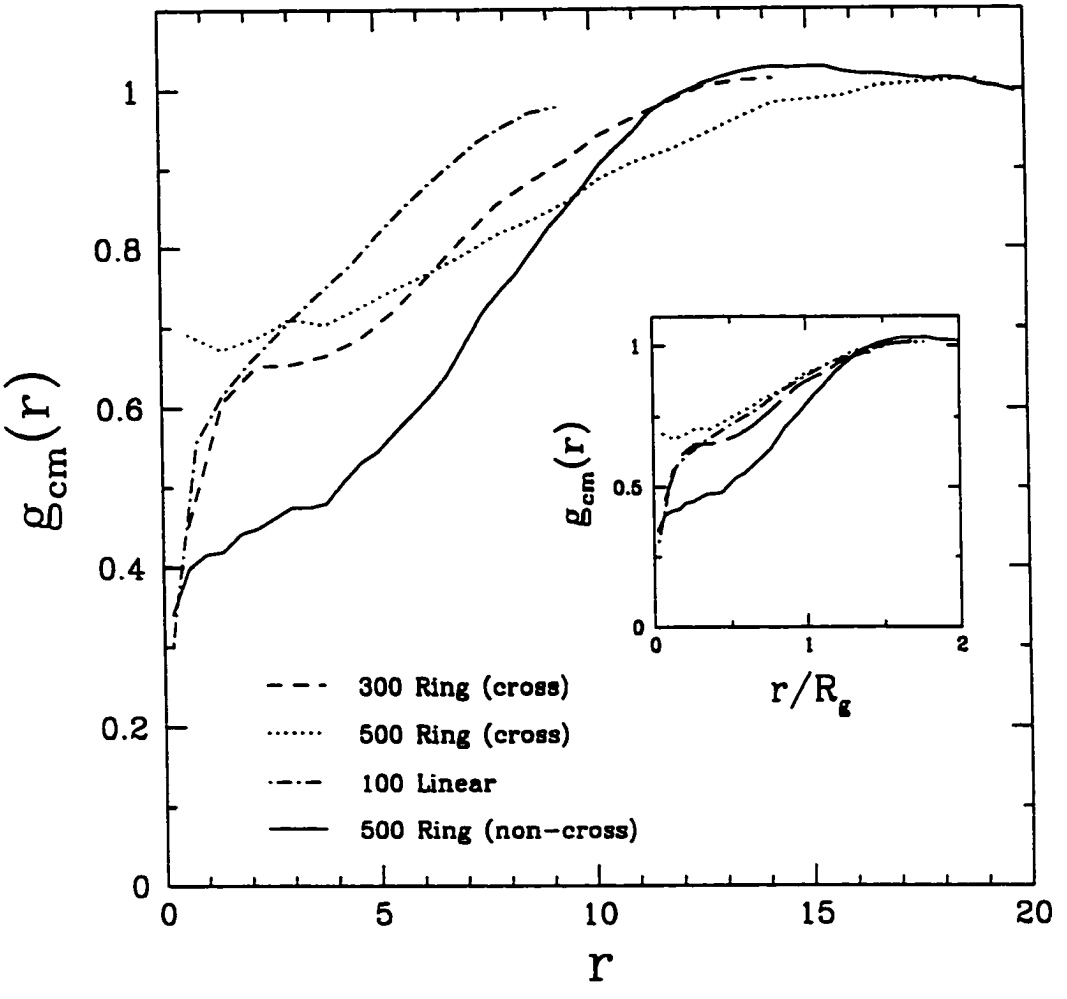

Figure 4.11: Center-of-mass pair-correlation function $g_{cm}(r)$ plotted versus separation distance r. The insert shows the same data with the separation distance r scaled by the radius of gyration.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

exponent from this fit gives the Gaussian scaling exponent $\nu = 0.5$.

Conclusion

The main conclusion that we can draw from the investigation of the equilibrium structure of rings in the melt is that the presence of topological constraints forces a given ring into a structure which is smaller and more compact than rings of similar degree of polymerization *without* the topological interactions. The absence of topological constraints leads to Gaussian scaling of the ring polymer size with the degree of polymerization, $R_g^2 \propto N$, whereas their presence results in the scaling $R_g^2 \propto N^{0.82}$.

Additionally, we find that the constraint of non-concatenation produces ring polymer configurations which have much less inter-penetration than is present for linear chains of similar degree of polymerization.

4.2.2 Dynamics

Non-crossing rings

As a first look at the dynamics of non-crossing rings we inspect the average monomer mobility $\langle\mu\rangle$. We define a mobility μ_n as the fraction of accepted MC moves of the nth bead. $\langle\mu\rangle$ is then the nth bead mobility averaged over all beads in the simulation box. It should be emphasized that the average monomer mobility is different from the long-time, center-of-mass mobility (in particular the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

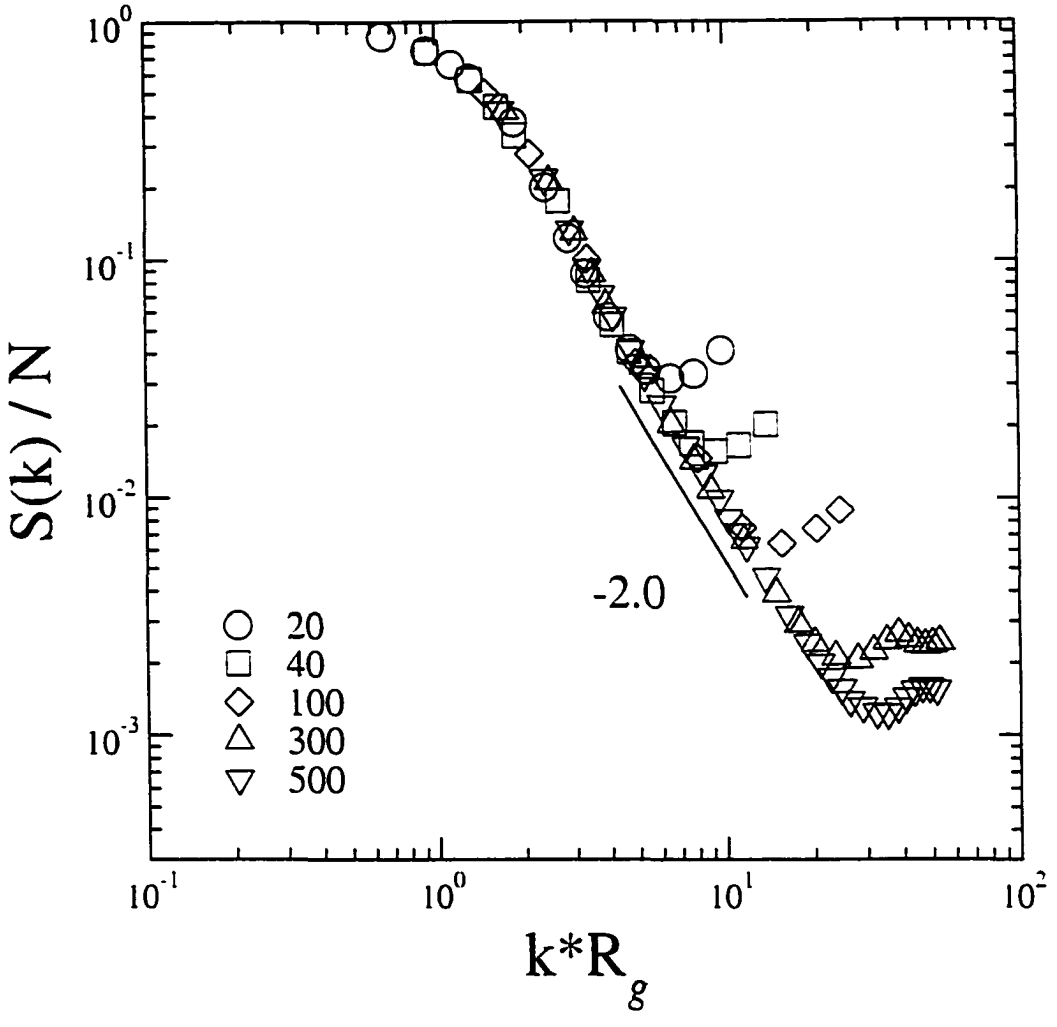

Figure 4.12: Static structure factor scaled by degree of polymerization $S(k)/N$ versus wave vector times the radius of gyration $k*R_g$ for crossing rings. The solid line shows the observed intermediate scaling: $S(k)/N \propto (kR_g)^{-2}$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

former does not depend on N whereas the latter does). Shown in Fig. 4.13 is the average mobility of all beads in the simulation box plotted versus ring size N. Apart from the small N values it can be seen that the average bead mobility is independent of N. The local bead motions are N-independent, and it is only on long time and length scales that N becomes important.

To analyze the time-dependent properties of ring polymers in the melt we monitor the following quantities: monomer mean-square displacement

$$g_1(t) = \frac{1}{n_{ring}} \sum_{\alpha=1}^{n_{ring}} \sum_{i=1}^{N} \left\langle (\mathbf{r}_i^\alpha(t) - \mathbf{r}_i^\alpha)^2 \right\rangle ; \tag{4.6}$$

monomer mean-square displacement in the center-of-mass reference frame

$$g_2(t) = \frac{1}{n_{ring}} \sum_{\alpha=1}^{n_{ring}} \sum_{i=1}^{N} \left\langle (\mathbf{r}_i^\alpha(t) - \mathbf{R}_{cm}^\alpha(t) - \mathbf{r}_i^\alpha + \mathbf{R}_{cm}^\alpha)^2 \right\rangle /2: \tag{4.7}$$

center-of-mass mean-square displacement

$$g_3(t) = \frac{1}{n_{ring}} \sum_{\alpha=1}^{n_{ring}} \left\langle (\mathbf{R}_{cm}^\alpha(t) - \mathbf{R}_{cm}^\alpha)^2 \right\rangle : \tag{4.8}$$

and finally, the diameter vector autocorrelation function

$$C_{ee}(t) = \frac{1}{n_{ring}} \sum_{\alpha=1}^{n_{ring}} \langle \mathbf{R}_e^\alpha(t) \cdot \mathbf{R}_e^\alpha \rangle . \tag{4.9}$$

For $g_1(t)$ and $g_3(t)$ it is expected that in the long-time limit there will be a linear dependence on time $g_1(t) \propto g_3(t) \propto t$, which represents free diffusive motion. In the case of $g_2(t)$ the long-time value is expected to converge to the mean-square radius of gyration. Finally, the function $C_{ee}(t)$ is expected to exhibit single-exponential decay at long times.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

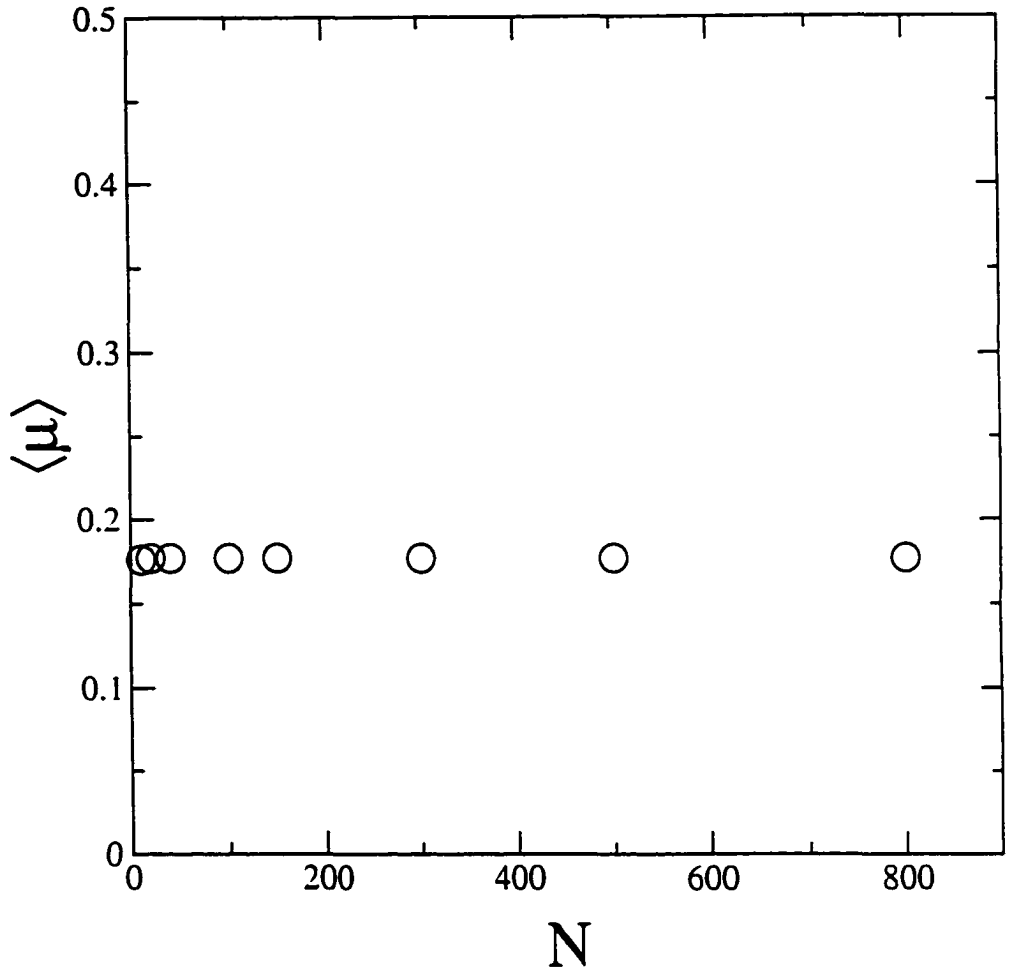

Figure 4.13: Average monomer mobility $\langle \mu \rangle$ versus ring size N for non-crossing rings in the melt.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The self-diffusion coefficients are obtained from the convergence of the ratio $g_3(t)N/6t$, shown in Fig. 4.14 for all non-crossing ring sizes studied. Note that the long-time asymptote of $g_3(t)N/6t$ converges to DN, the quantity of interest.

In Fig. 4.15 the values for DN are plotted versus N. For comparison the data for linear chains is included in the plot. Self-diffusion for the linear chains undergoes an entanglement transition at $N_{c,linear} \approx 40$. An analogous crossover for the rings, if there is one, is less obvious and much more gradual.

We observe faster ring diffusion for all ring sizes investigated. For the smaller rings (*i.e.* for $N \leq N_{c,linear}$) this result agrees with the computer simulations of Müller *et al.* The fact that faster ring diffusion persists for ring sizes as large as twenty times greater than $N_{c,linear}$ is the main result of this section. As indicated in Fig. 4.15 the power law fit of $N > 100$ data results in $DN \propto N^{-0.6}$ scaling of the ring center-of-mass self-diffusion coefficient.

In Fig. 4.16 we plot the mean-square displacements $g_1(t)$, $g_2(t)$ and $g_3(t)$ for all ring sizes. For the center-of-mass displacement $g_3(t)$, we observe a slower-than-linear time dependence for short times. A similar anomalous diffusion has been observed at short times for linear chains[39] and also for small rings[36]. This has been interpreted as a signature of a net force acting on the test chain. The existence of such a force is not surprising as the test ring is partially collapsed precisely due to the topological interaction with surrounding rings.

Also shown in Fig. 4.16 are fits for the monomer mean-square displacement

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

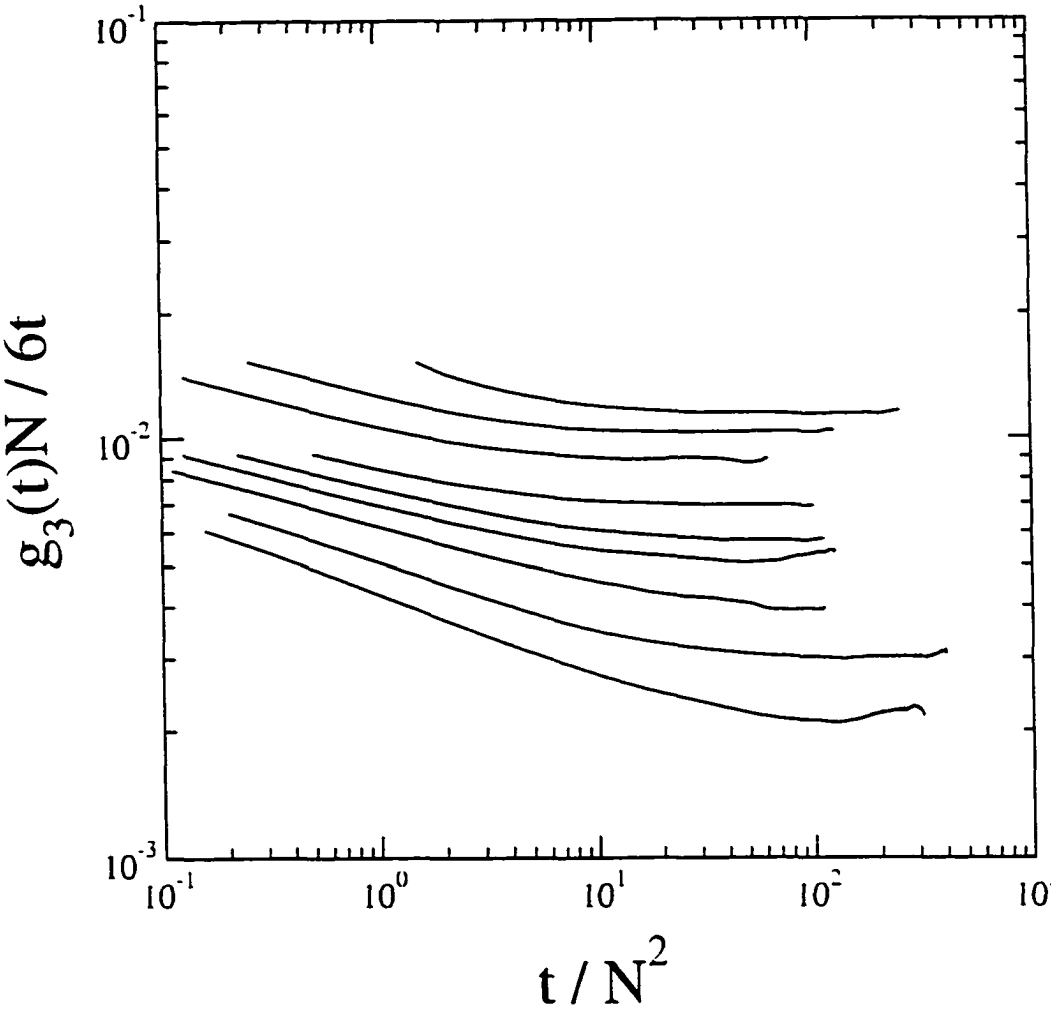

Figure 4.14: Mean-square center-of-mass displacement times degree of polymerization divided by six times time $g_3(t)N/6t$ versus time scaled by the degree of polymerization squared t/N^2 for non-crossing rings. Note that the time axis is rescaled merely for convenience.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

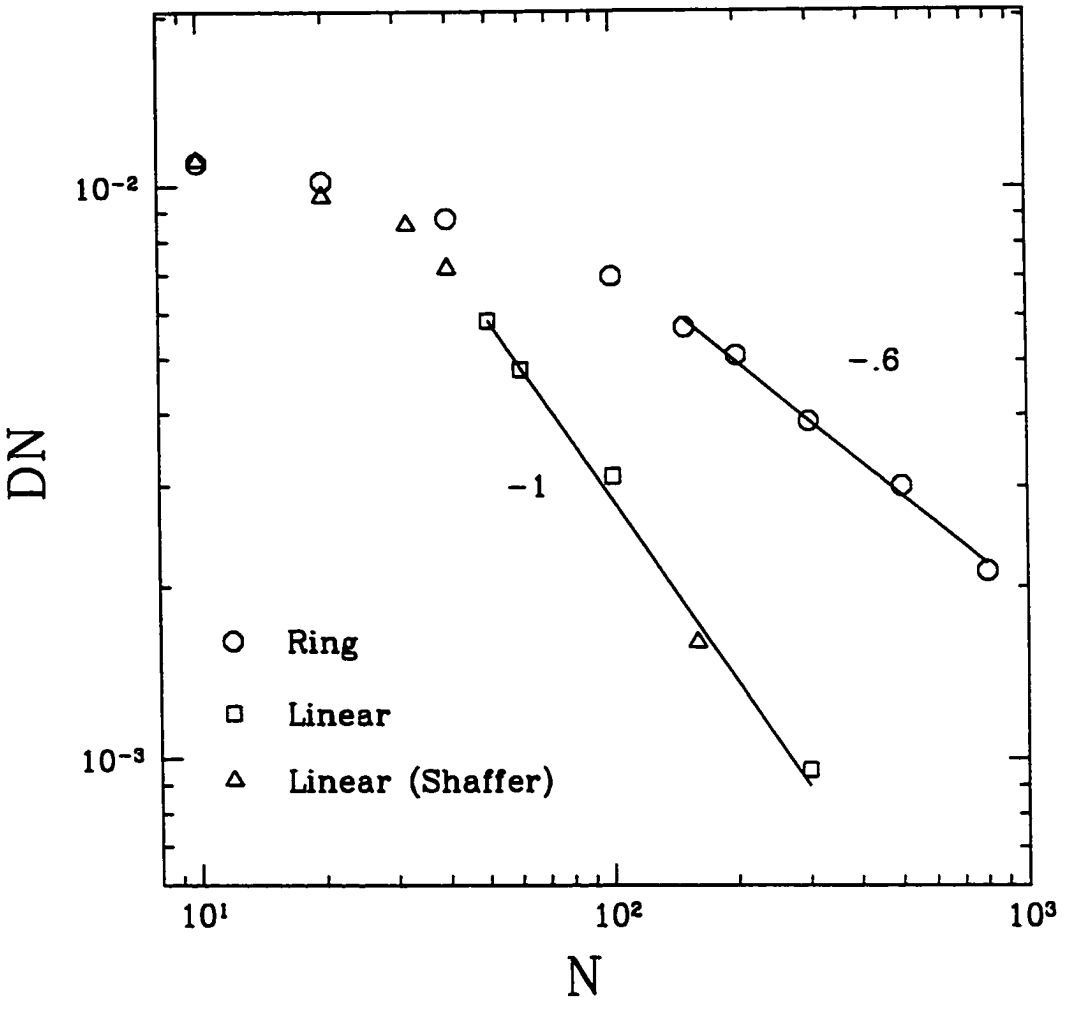

Figure 4.15: Center-of-mass self-diffusion coefficient times degree of polymerization DN versus degree of polymerization N for non-crossing rings and linear chains. The lines indicate power law fits with scaling exponent indicated.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

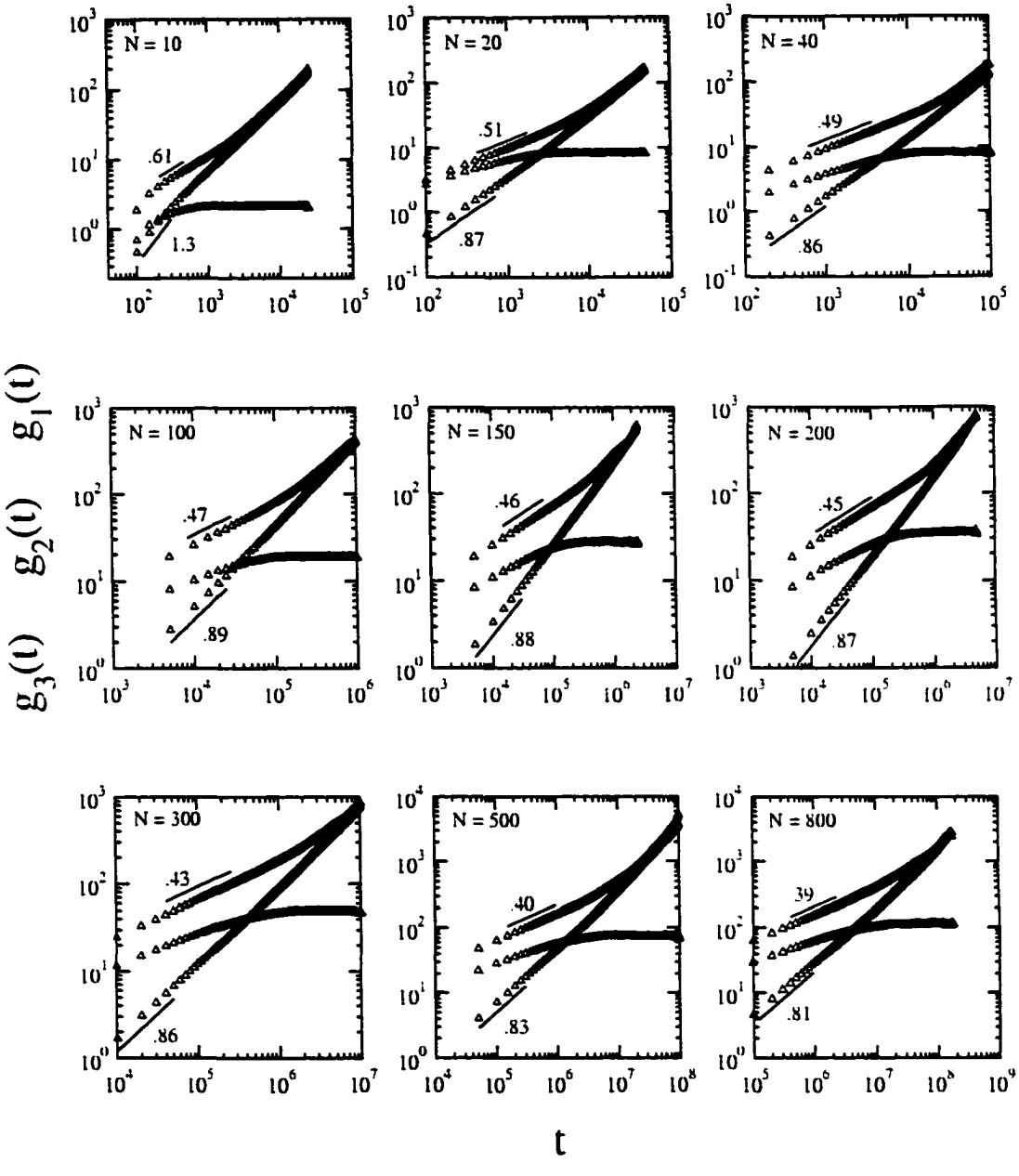

Figure 4.16: Mean-square monomer displacement $g_1(t)$, mean-square monomer displacement in the center-of-mass frame $g_2(t)$, and mean-square center-of-mass displacement $g_3(t)$ versus time t. The solid lines show power-law fits, with scaling exponents indicated.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$g_1(t)$ at intermediate times. We obtain fits of the form $g_1(t) \propto t^a$, with the exponent a slightly N-dependent and monotonically decreasing from the Rouse-like value $a = 0.5$ (for $N = 20$) to $a = 0.38$ (for $N = 800$). The value for 10mer rings, $a = .61$, is disregarded as 10mers appear to have quite anomalous dynamics. The precise source of this is unclear. Presumably the chains are simply too short to be accurate representations of polymer chains in this model; however, the anomalous value disappears for 10mer crossing rings. Similar effective exponents ($a \leq 0.5$) have been observed in Ref.[36]. They were rationalized in terms of a non-Gaussian equilibrium structure of rings. Clearly they are indicative of the constrained motion of the rings, which appears to increase with increasing ring size.

Next we look at the diameter vector autocorrelation function $C_{ee}(t)$, which is a measure of the time-dependent orientational correlation of rings in the melt. It is analogous to the end-to-end vector correlation function monitored for linear chains. In Fig. 4.17 we show the diameter vector correlation functions for all ring sizes simulated. For long times $C_{ee}(t)$ shows an exponential decay. This asymptotic decay time can be used as a definition for the single-chain orientational relaxation time, denoted by τ_{ee}. It should be emphasized that $C_{ee}(t)$ does not show an exponential decay for all times.

Shown in Fig. 4.18 are the relaxation times plotted versus degree of polymerization. In addition to the orientational relaxation time τ_{ee} we also plot the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

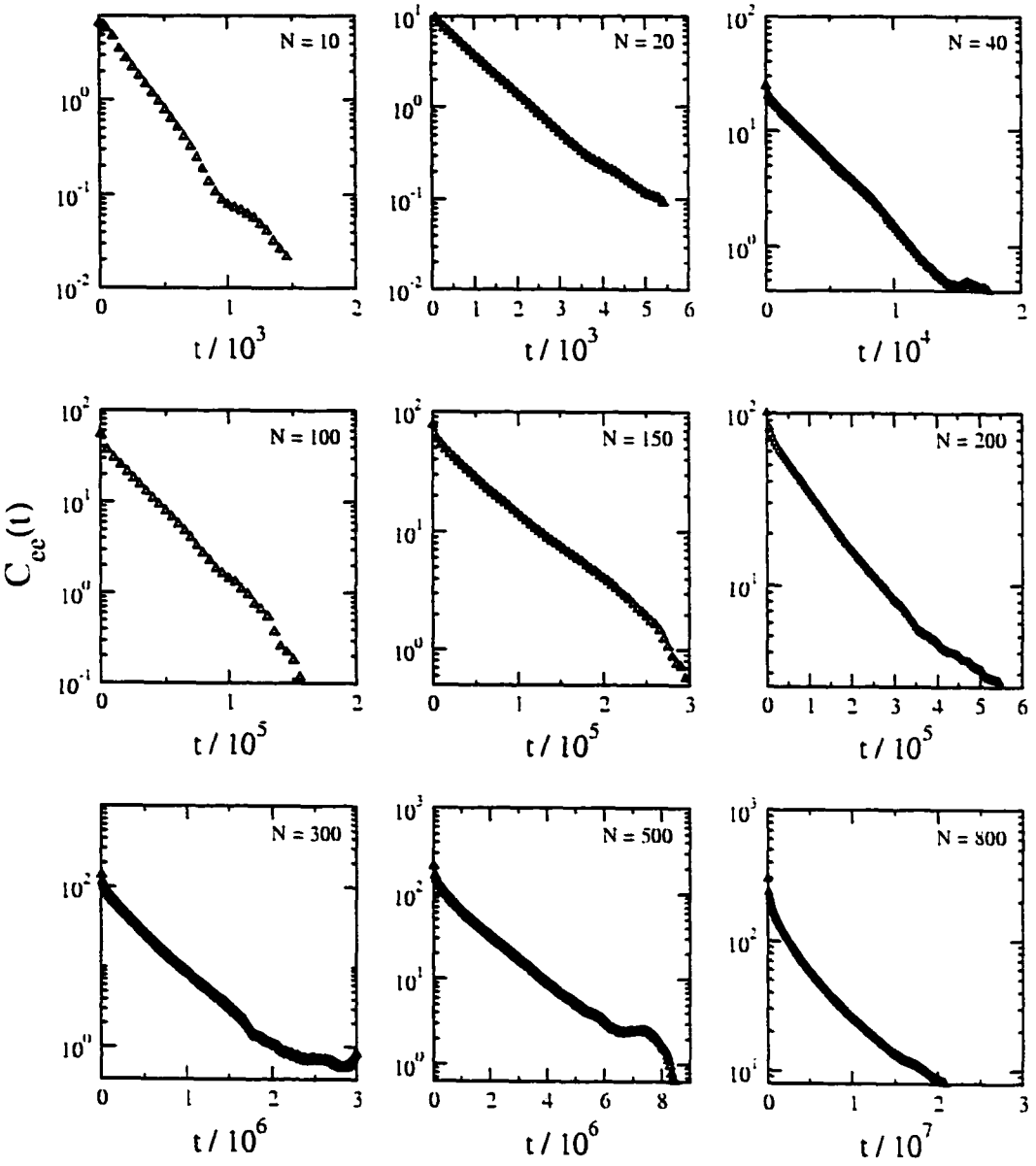

Figure 4.17: Diameter vector auto-correlation function $C_{ee}(t)$ versus time t for non-crossing rings. Note that for convenience the time axes have been rescaled by differing orders of magnitude (as indicated on each time axis).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

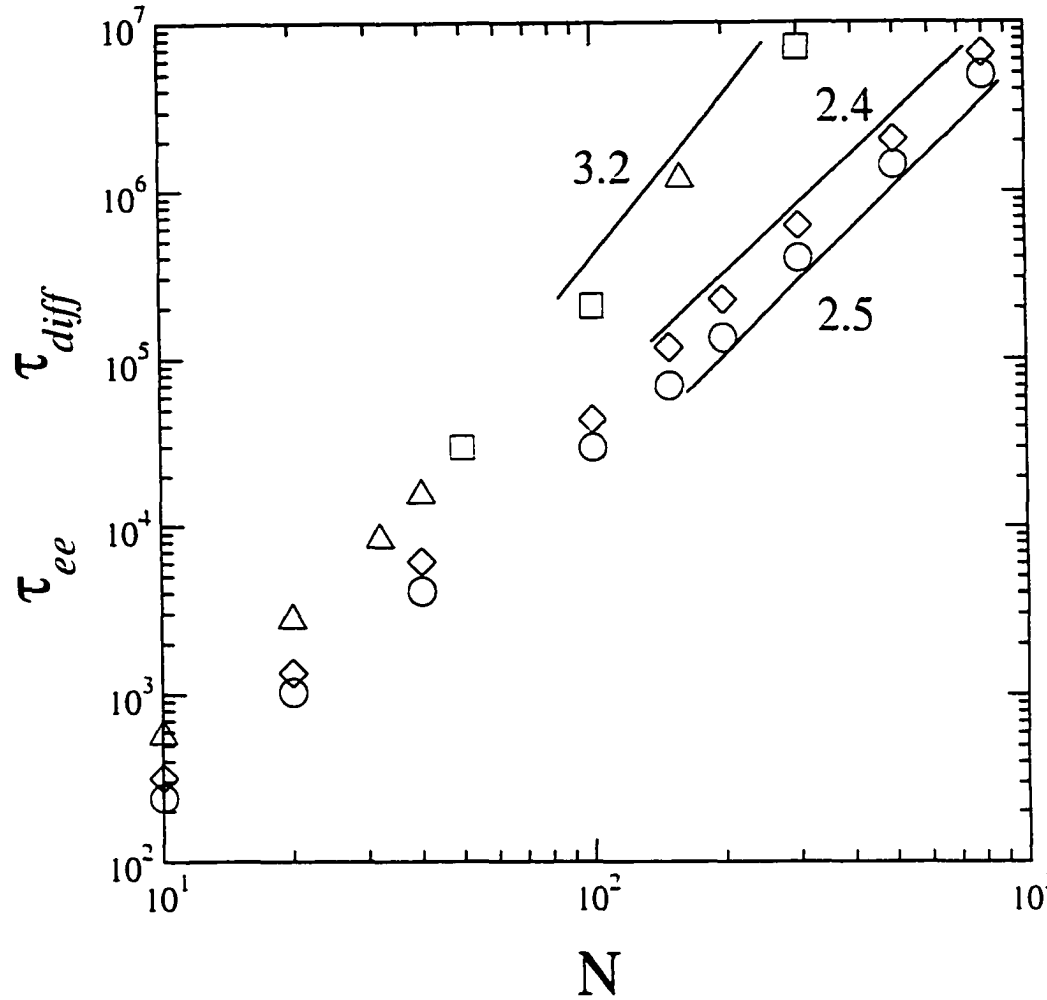

Figure 4.18: Single-chain relaxation times τ_{ee} and τ_{diff} versus degree of polymerization N. Circles: exponential long-time fit to $C_{ee}(t)$ for rings; diamonds: characteristic diffusion time $\tau_{diff} = R_g^2/6D$ for rings; triangles: linear chain data for τ_{diff}, taken from Ref.[25]; squares: linear chain data for τ_{diff}, taken from our work. The lines indicate power law fits with scaling exponents indicated.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

characteristic diffusion time $\tau_{diff} = R_g^2/6D$. This is a translational relaxation time: it is the time required for a chain to diffuse a distance equal to the radius of gyration. We find the two times to have similar magnitude and to scale with N in approximately the same way. For linear chains τ_{diff} has a much stronger N-dependence.

To determine the power-law scaling of the relaxation times we fit $N > 100$. For the orientational decay time τ_{ee} the power law fit results in $\tau_{ee} \propto N^{2.5}$, in reasonable agreement with the scaling found by Müller *et al* for smaller rings. For the characteristic diffusion time we find $\tau_{diff} \propto N^{2.4}$, in close agreement with the fit for τ_{ee}. These scaling dependencies should be contrasted with the $\tau_{diff} \propto N^{3.2}$ scaling of the characteristic diffusion time of long linear chains.

Crossing rings

In analogy with the dynamics of non-crossing rings we first present data for the average bead mobility (shown in Fig. 4.19). As with non-crossing rings, we see that apart from small values of N the average bead mobility is independent of N. We do notice an approximate 8 percent increase in the average mobility of crossing rings compared to non-crossing rings.

We obtain the self-diffusion coefficients in a similar way as for non-crossing rings, *i.e.* from the convergence of the ratio $g_3(t)/6t$. Shown in Fig. 4.20 is $g_3(t)/6t$ versus time for all crossing rings sizes simulated. Note that in order to obtain lines which are not clustered together in Fig. 4.20 the quantity plotted is

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

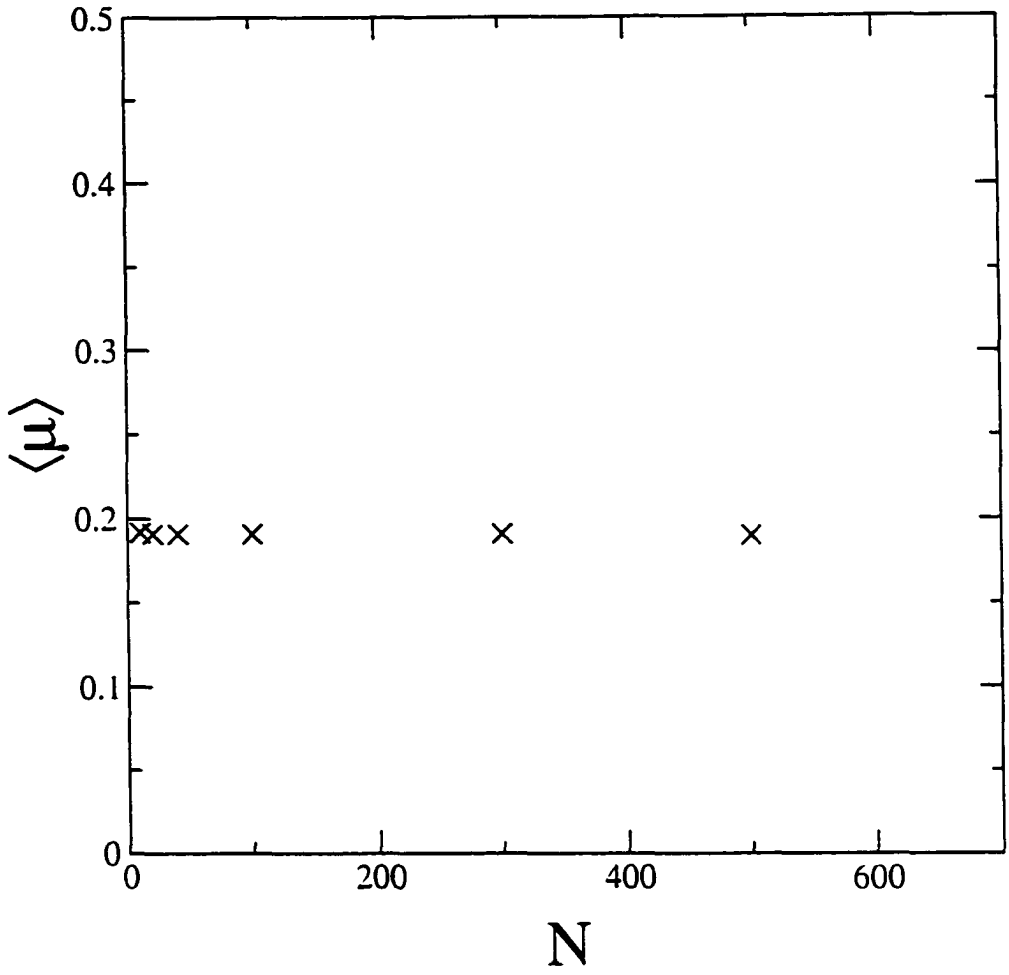

Figure 4.19: Average monomer mobility $\langle \mu \rangle$ versus ring size N for crossing rings in the melt.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$g_3(t)/6t$, as opposed to $g_3(t)N/6t$ plotted for non-crossing rings.

Fig. 4.21 shows the self-diffusion coefficients for the crossing rings. For all ring sizes studied we observe a Rouse-like scaling $DN \approx const.$ This is to be contrasted with the result for non-crossing rings $DN \propto N^{-0.6}$. Clearly removing the constraint of non-concatenation results in a significant alteration of the dynamics.

As with the non-crossing rings we now look at the mean-square displacements $g_1(t)$, $g_2(t)$ and $g_3(t)$ (shown in Fig. 4.22). For all crossing rings we observe Rouse-like behavior of the center-of-mass mean-square displacement, $g_3(t) \propto t^1$, for all times (fits not shown in plot). Additionally, we observe the Rouse-like square-root dependence for the monomer mean-square displacement at intermediate times, $g_1(t) \propto t^{1/2}$. This is consistent with what is to be expected for the dynamics of an unentangled linear chain.

In Fig. 4.23 we show the diameter vector autocorrelation functions for crossing rings. From a long-time exponential fit of $C_{ee}(t)$ we obtain the single-chain orientational relaxation time τ_{ee}.

The orientational relaxation times τ_{ee} and the characteristic diffusion times τ_{diff} are plotted in Fig. 4.24. Fits of the $N \geq 100$ data yield the same scaling law $\tau_{ee} \propto \tau_{diff} \propto N^{2.0}$. These fits are also consistent with that of an unentangled linear chain system. This should be contrasted with the stronger N-dependence of the scaling for non-crossing rings $\tau_{ee} \propto N^{2.5}$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

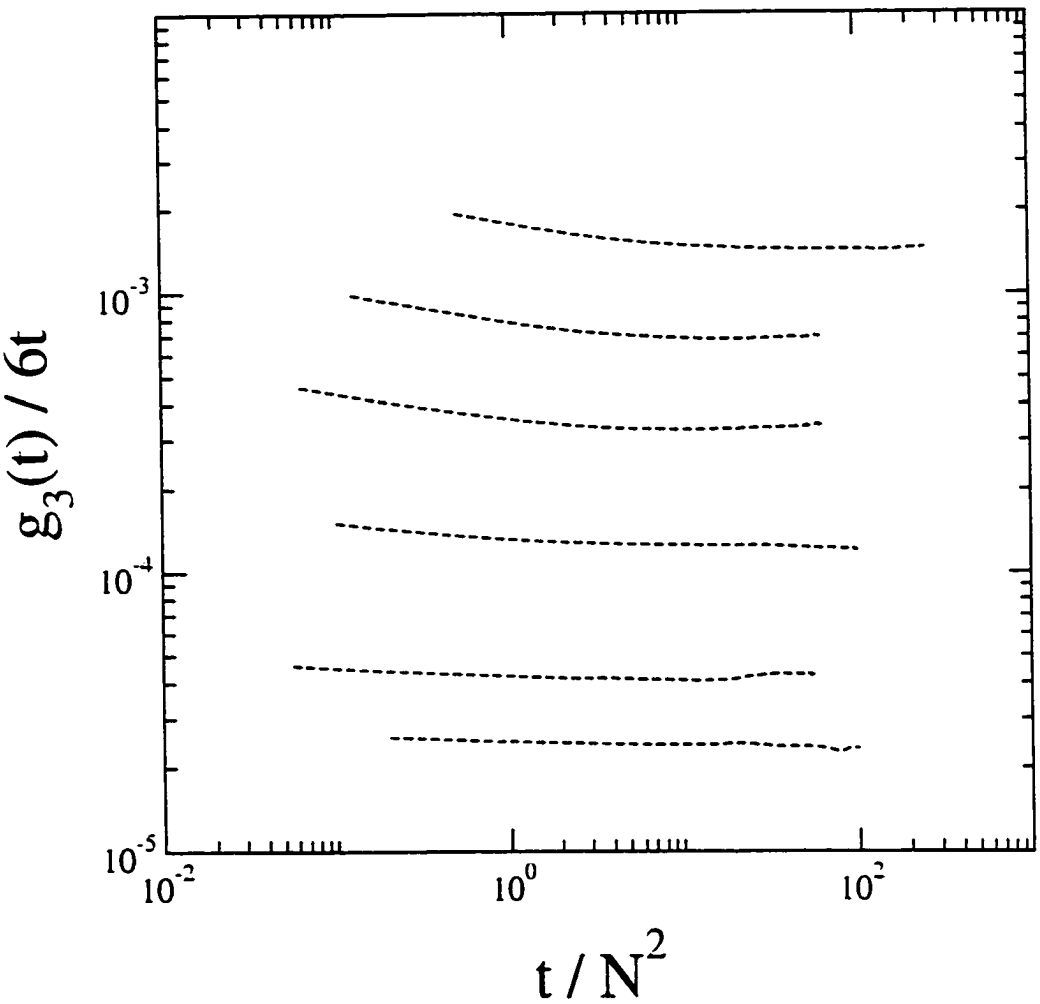

Figure 4.20: Mean-square center-of-mass displacement divided by six times time $g_3(t)/6t$ versus time scaled by the degree of polymerization squared t/N^2 for crossing rings in the melt. Note that the time axis is rescaled merely for convenience.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

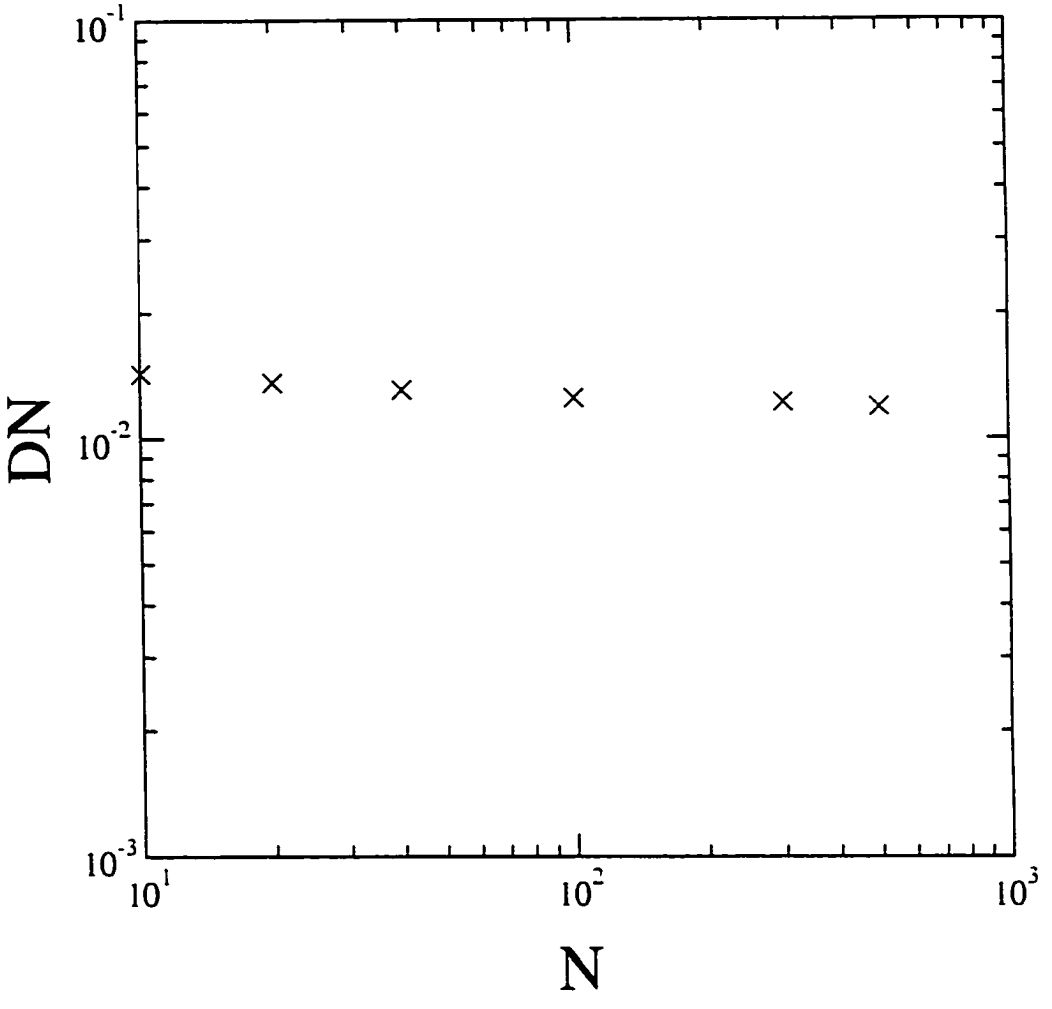

Figure 4.21: Center-of-mass self-diffusion coefficient times degree of polymerization DN versus degree of polymerization N for crossing rings in the melt.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

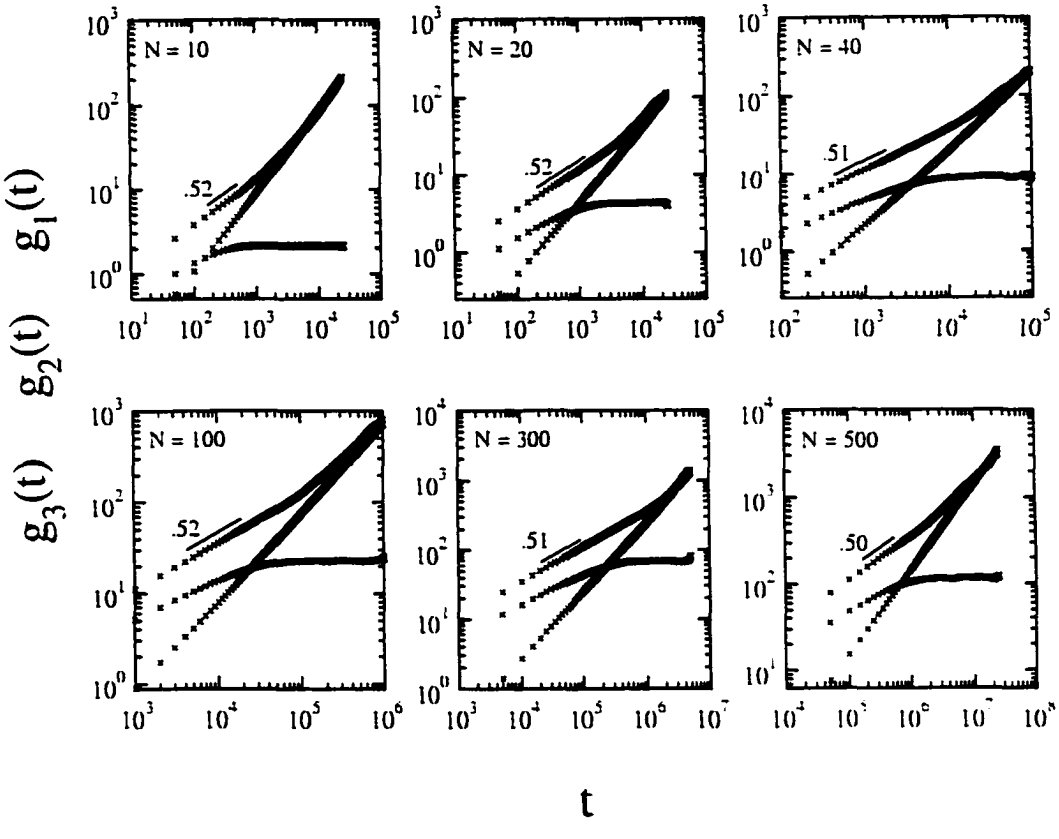

Figure 4.22: Mean-square monomer displacement $g_1(t)$, mean-square monomer displacement in the center-of-mass frame $g_2(t)$, and mean-square center-of-mass displacement $g_3(t)$ versus time t for crossing rings in the melt. The solid lines show intermediate-time power-law behavior, with scaling exponents indicated.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

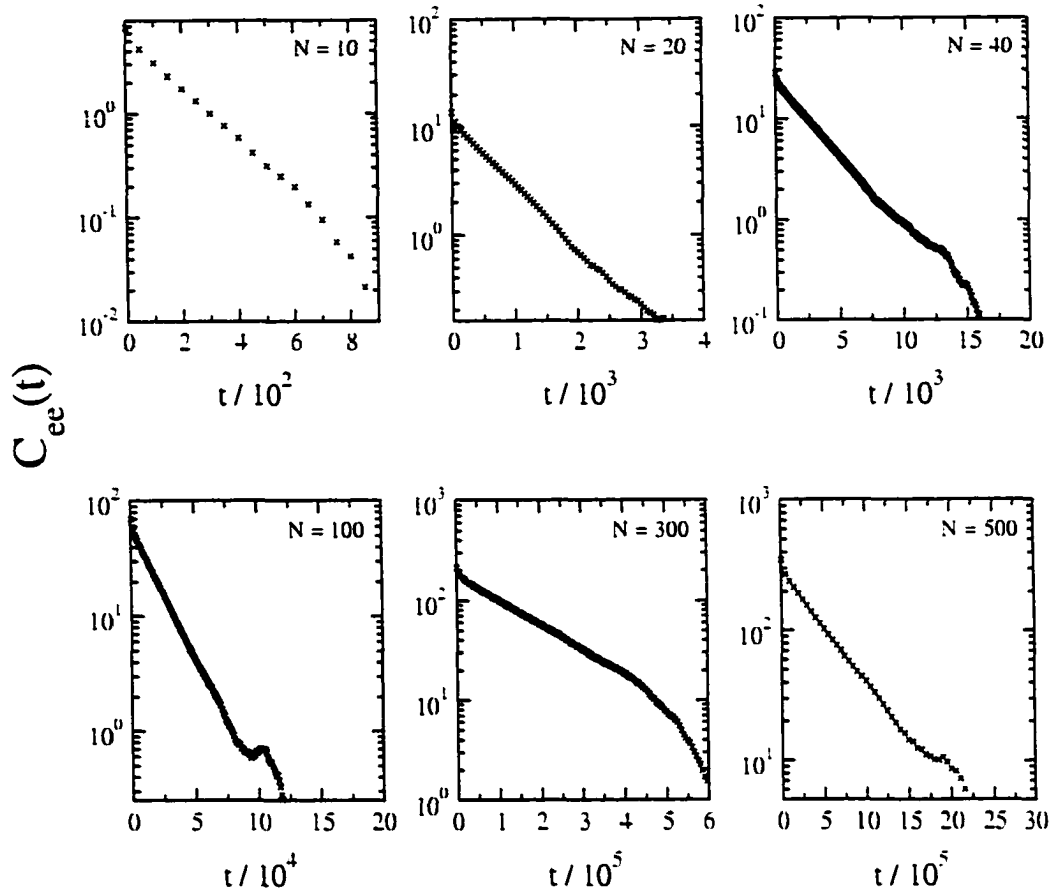

Figure 4.23: Diameter vector autocorrelation function $C_{ee}(t)$ versus time t for crossing rings in the melt. Note that for convenience the time axes have been rescaled by differing orders of magnitude (as indicated on each time axis)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

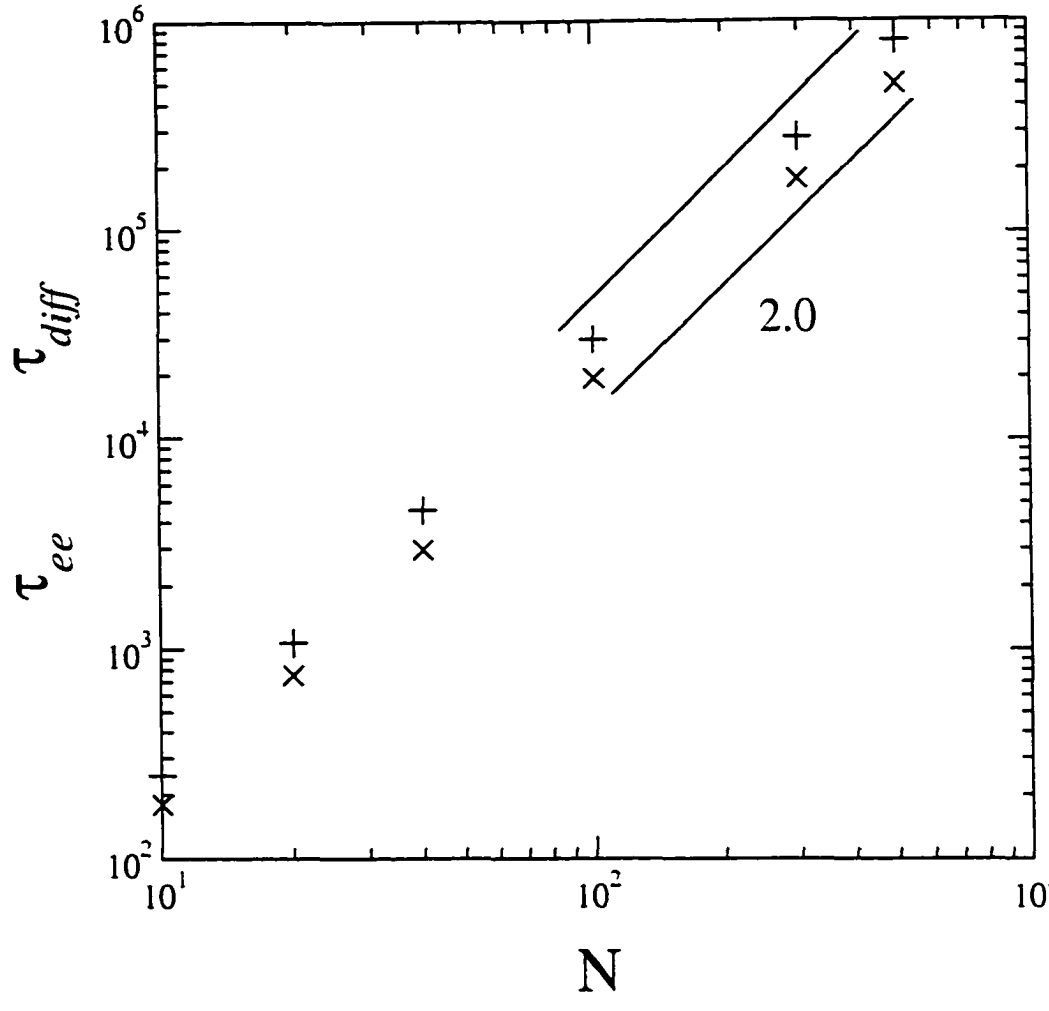

Figure 4.24: Single-chain relaxation times τ_{ee} and τ_{diff} versus degree of polymerization N for crossing rings in the melt. Crosses: exponential long-time fit to $C_{ee}(t)$; plus signs: characteristic diffusion time $\tau_{diff} = R_g^2/6D$. The lines indicate power law fits with the scaling exponents indicated.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Conclusion

In conclusion we find that topological interactions have significant consequences for the dynamics of ring polymers in the melt. The absence of topological constraints leads to the scaling of the self-diffusion coefficient for crossing rings $D \propto N^{-1}$, whereas for non-crossing rings the scaling observed is $D \propto N^{-1.6}$. Additionally, the relaxation times follow different power laws: for crossing and non-crossing rings the orientational relaxation times scale as $\tau_{ee} \propto N^{2.0}$ and $\tau_{ee} \propto N^{2.5}$, respectively.

4.3 Single rings

We now investigate isolated rings on the lattice. For such rings the non-crossing constraint results in a restriction forbidding configurations which contain self-knots. The simulations of single rings thus provide some insight into the importance of self-knots in determining static and dynamic properties.

The equilibrium size of isolated, unknotted ring polymers with zero excluded volume has been investigated by Deutsch[40]. It was claimed that the scaling exponent shows approach to a SAW value, which is in agreement with an analytical argument[41] that topological constraints lead to the same scaling behavior as excluded volume interactions.

The purpose of the simulations presented in this section is to investigate the influence of topological constraints on the equilibrium structure and dynamics of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

isolated ring polymers. The main result is that topological interactions have only a very mild quantitative influence on the structure and dynamics of isolated ring polymers. See Table 3.2 for pertinent simulation parameters and data from the single-ring crossing and non-crossing simulations.

4.3.1 Equilibrium properties

As in previous sections we use the mean-square radius of gyration R_g^2 and the mean-square ring diameter R_e^2 as our measure of average ring size. Fits for both crossing and non-crossing isolated rings for $N \geq 100$ (Fig. 4.25) yield the scaling relationships $R_g^2 \propto R_e^2 \propto N^{1.17}$. Apparently self-knots play little role in determining the scaling of the average size for isolated rings.

Off-lattice simulations by Baumgärtner[42] of a single unknotted ring polymer find the scaling $N^{1.18}$ for both R_e^2 and R_g^2. For single rings on the lattice Sikorski[43] finds $R_e^2 \propto N^{1.180}$ and $R_g^2 \propto N^{1.186}$.

The recent simulations by Deutsch[40] of rings with zero excluded volume find the scaling relationship for larger rings of $R_g^2 \propto N^{1.17}$. It is argued that the topological interaction alone is sufficient to produce SAW scaling for large rings. We see no difference in the scaling exponents for our single rings in either the crossing or non-crossing systems; however, non-crossing rings are slightly larger: topological interactions produce an increase of the excluded volume as argued in Ref.[41].

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

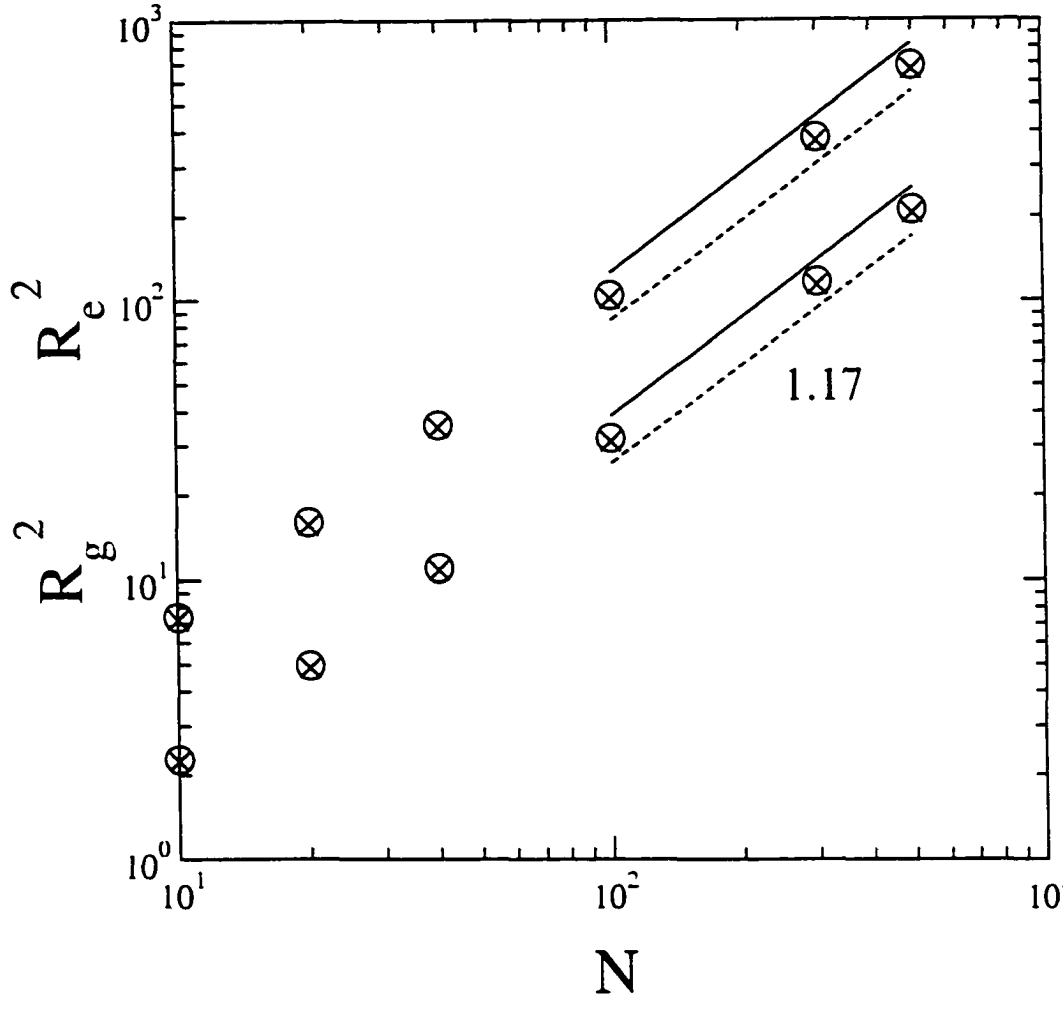

Figure 4.25: Mean-square radius of gyration R_g^2 and mean-square diameter vector R_e^2 versus degree of polymerization N for crossing and non-crossing isolated rings. Circles: non-crossing rings; crosses: crossing rings. The solid lines indicate the fits to the non-crossing rings, and the dashed lines indicated the crossing fits. The scaling exponent obtained from all fits is the same.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.3.2 Dynamics

In Fig. 4.26 we show self-diffusion data for single rings. It can be seen that DN is roughly independent of N for both crossing and non-crossing simulations, indicating the Rouse-like scaling $D \propto N^{-1}$.

If we look at characteristic orientational relaxation times for single rings (shown in Fig. 4.27), we find that both the crossing and non-crossing rings exhibit the identical scaling $\tau_{ee} \propto N^{2.1}$. This is consistent with the previous simulations of isolated rings on the lattice by Skolnick and Kolinski[29] in which they report $\tau_{ee} \propto N^{2.1}$. We find then that the dynamics of isolated rings occurs on similar time scales regardless of the presence of topological constraints in the system.

Conclusion

To conclude the analysis, we find that for isolated ring polymers the presence or absence of topological interactions does not affect, *qualitatively*, either static or dynamic properties: scaling exponents in both cases do not change.

4.4 Discussion

To discuss the dynamics of rings in the melt we now turn to a comparison between crossing and non-crossing simulation results. It is clear that non-crossing rings in the melt are experiencing constrained dynamics that are a consequence of the topological interactions between rings. This can be seen in the fact that removal

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

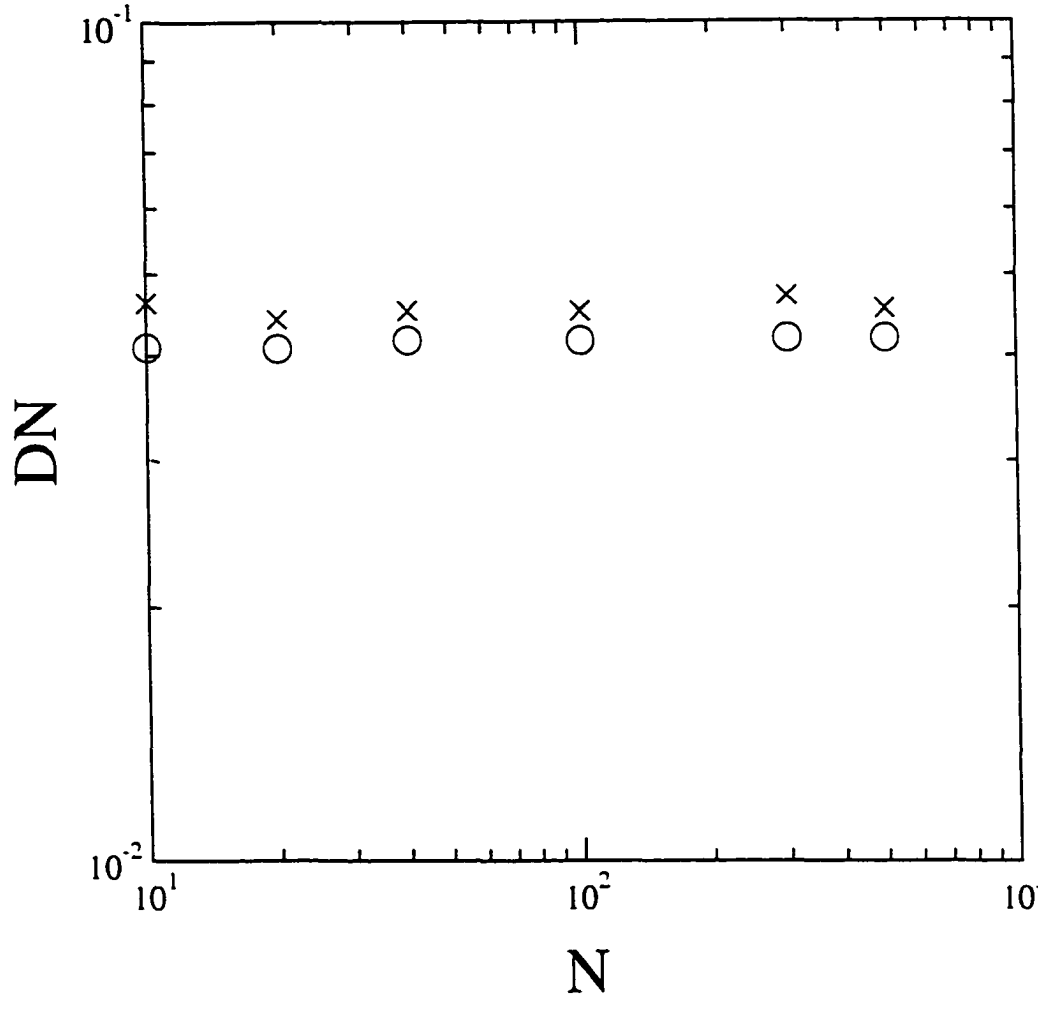

Figure 4.26: Center-of-mass self-diffusion coefficient times degree of polymerization DN versus degree of polymerization N for crossing and non-crossing isolated rings. Circles: non-crossing rings; crosses: crossing rings.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

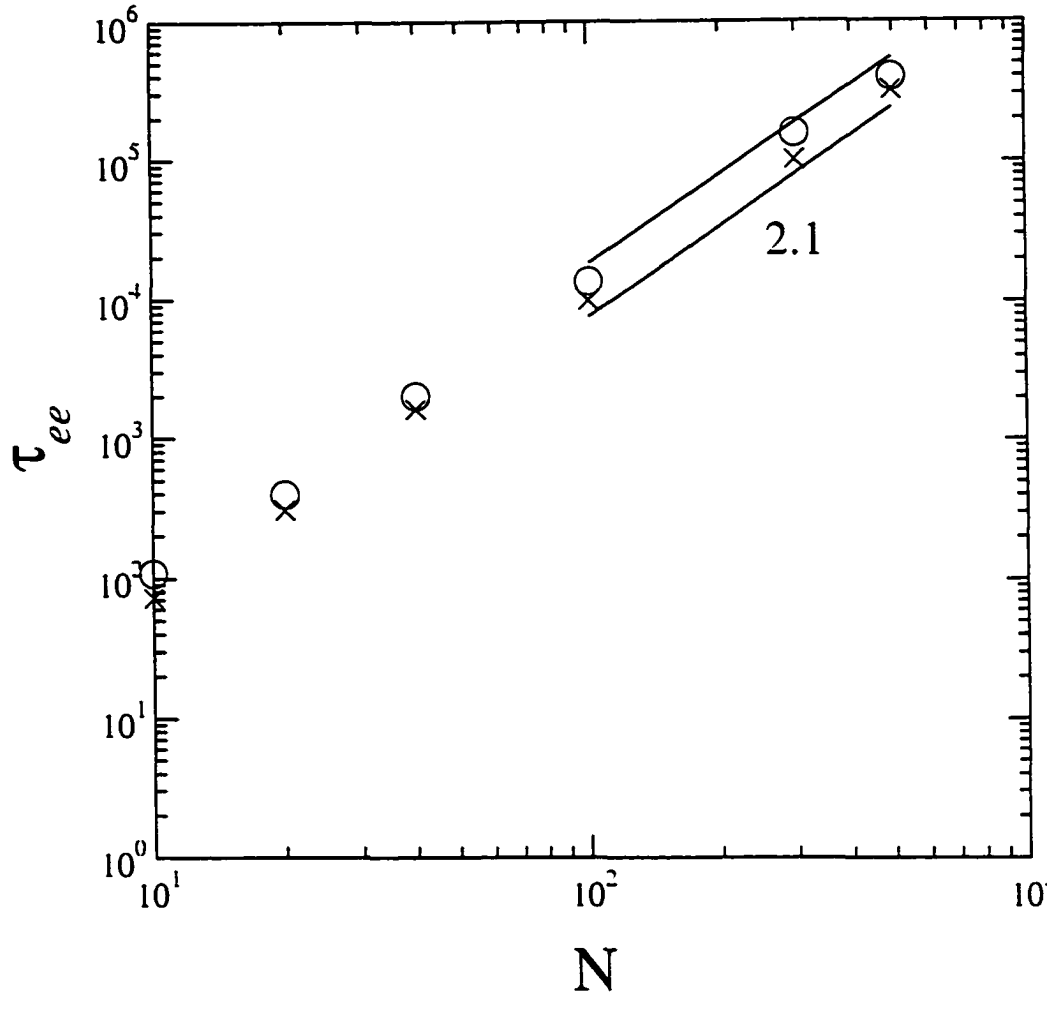

Figure 4.27: Single-chain orientational relaxation time τ_{ee} versus degree of polymerization N for crossing and non-crossing isolated rings. Circles: non-crossing rings; crosses: crossing rings. The lines indicate power the law fits with scaling exponent indicated.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of the topological constraints in the crossing ring simulations results in Rouse-like scaling of the self-diffusion coefficient and longest relaxation time. Unanswered is the question of whether or not the larger rings are experiencing a fully entangled (asymptotic) dynamics, or whether they are in a crossover regime to a new dynamics.

To address this issue we investigate the different time scales on which the dynamics of different size rings occurs. In Fig. 4.28 we show rescaled values for all three mean-square displacements (g_1, g_2 and g_3), plotted versus a rescaled time coordinate. The displacements are divided by the mean-square radius of gyration and the time is divided by the orientational relaxation time. From the plot we see that the rings with $N = 100$, 150, and 200 superimpose quite well; however, for the larger rings with $N = 300$, 500, and 800 there is a scatter which appears to have a systematic drift associated with it. When contrasted with Fig. 4.29, which shows the data for crossing rings scaled in the same way, it can be seen that for a similar range of N the crossing data collapses perfectly.

We can perform a similar scaling for the diameter vector autocorrelation function. In Fig. 4.30 we plot the normalized diameter vector autocorrelation function versus reduced time. We see that, as in Fig. 4.28, the rings with $N = 100$, 150, and 200 superimpose essentially perfectly, whereas the larger rings appear to show the same systematic drift. Again, if we compare this with the rescaled data for crossing rings we see no such drift, and all data collapses to a single curve

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

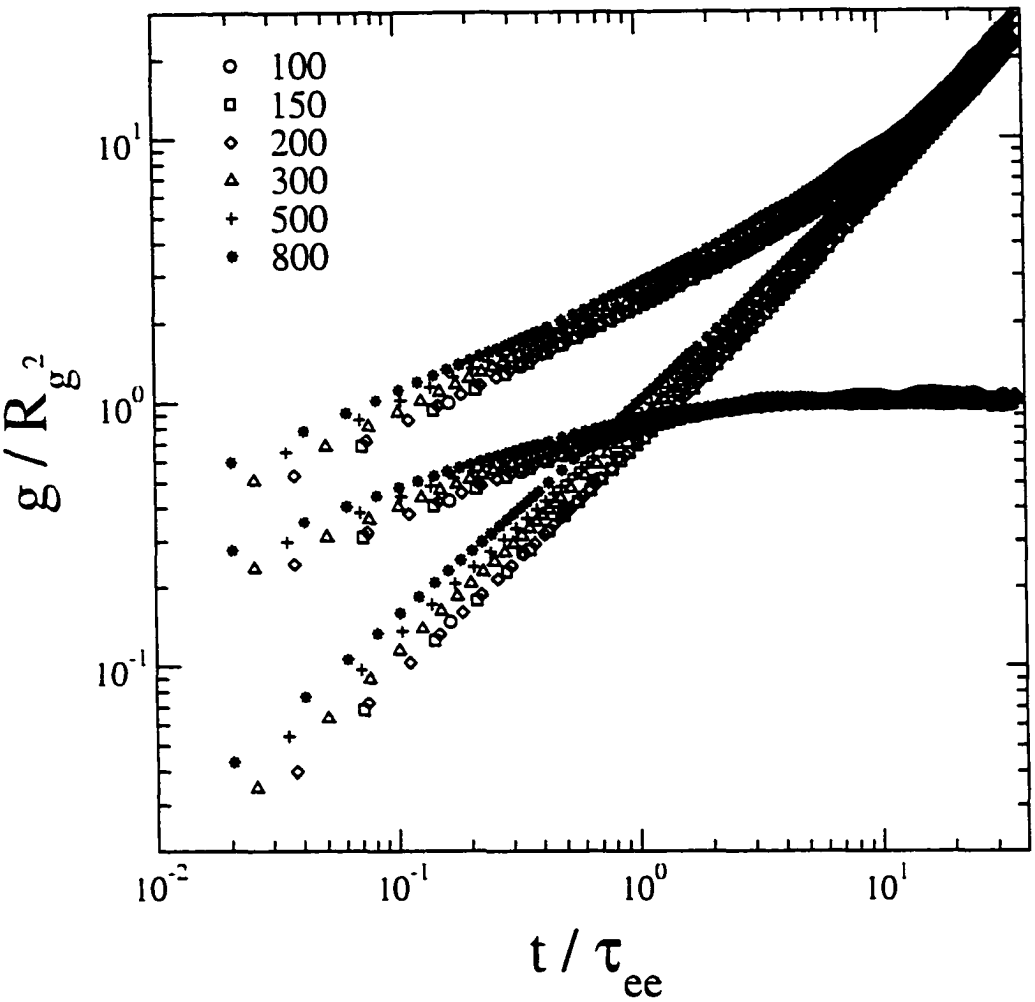

Figure 4.28: Mean-square displacements (g_1, g_2 and g_3) scaled by the mean-square radius of gyration g/R_g^2 versus time scaled by the orientational relaxation time t/τ_{ee} for non-crossing rings in the melt.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

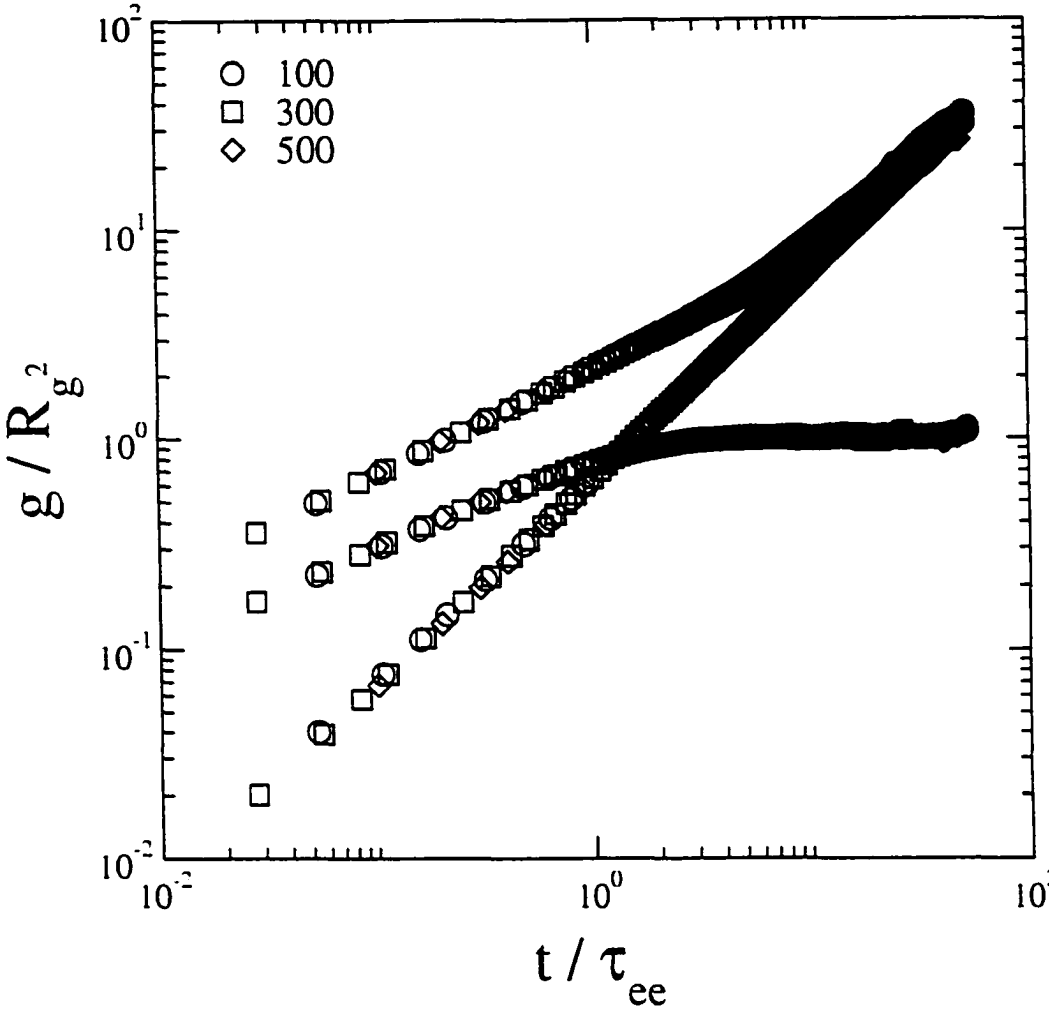

Figure 4.29: Mean-square displacements (g_1, g_2 and g_3) scaled by the mean-square radius of gyration g/R_g^2 versus time scaled by the orientational relaxation time t/τ_{ee} for crossing rings in the melt.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(Fig. 4.31). This suggests that for the larger non-crossing rings ($N \geq 300$) there is the emergence of a crossover to a new time scale for the dynamics.

In obtaining the relaxation times τ_{ee} we performed exponential fits to the long-time parts of $C_{ee}(t)$. Each of these exponential fits has a corresponding amplitude which indicates a length scale associated with the relaxation mode. If we plot the amplitudes obtained from the long-time exponential fits of $C_{ee}(t)$, $C_{ee,fit}(t=0)$, we see an interesting scaling behavior. The amplitudes for $N \leq 200$ scale roughly with the average size of the entire ring, $C_{ee,fit}(t=0) \propto N^{0.8}$; however, for $N > 200$ the scaling undergoes a transition to $C_{ee,fit}(t=0) \propto N^{0.6}$. It appears that the longest relaxation time is becoming associated with a smaller portion of the chain for the larger rings. Stated in another way, this seems to indicate the appearance of new long-lived constraints upon ring polymer motion, the effects of which are beginning to play an appreciable role in the ring dynamics (unlike reptation, these new constraints "act" on a length scale smaller than R_g). Combined with the evidence for the appearance of a new time scale, this implies that there is indeed the beginning of a transition to a new dynamical regime for the larger rings.

Finally, in Fig. 4.33 we look at D as a function of R_g for rings and linear chains. Fit to the $N > 100$ data for rings yields the scaling $D \propto R_g^{-3.9}$. This is consistent with the observed scaling $D \propto N^{-1.6}$. For linear chains, a fit to the $N \geq 50$ data yields an identical scaling $D \propto R_g^{-3.9}$, which is also consistent with the observed entangled scaling for long linear chains $D \propto N^{-2.0}$. The scaling of the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

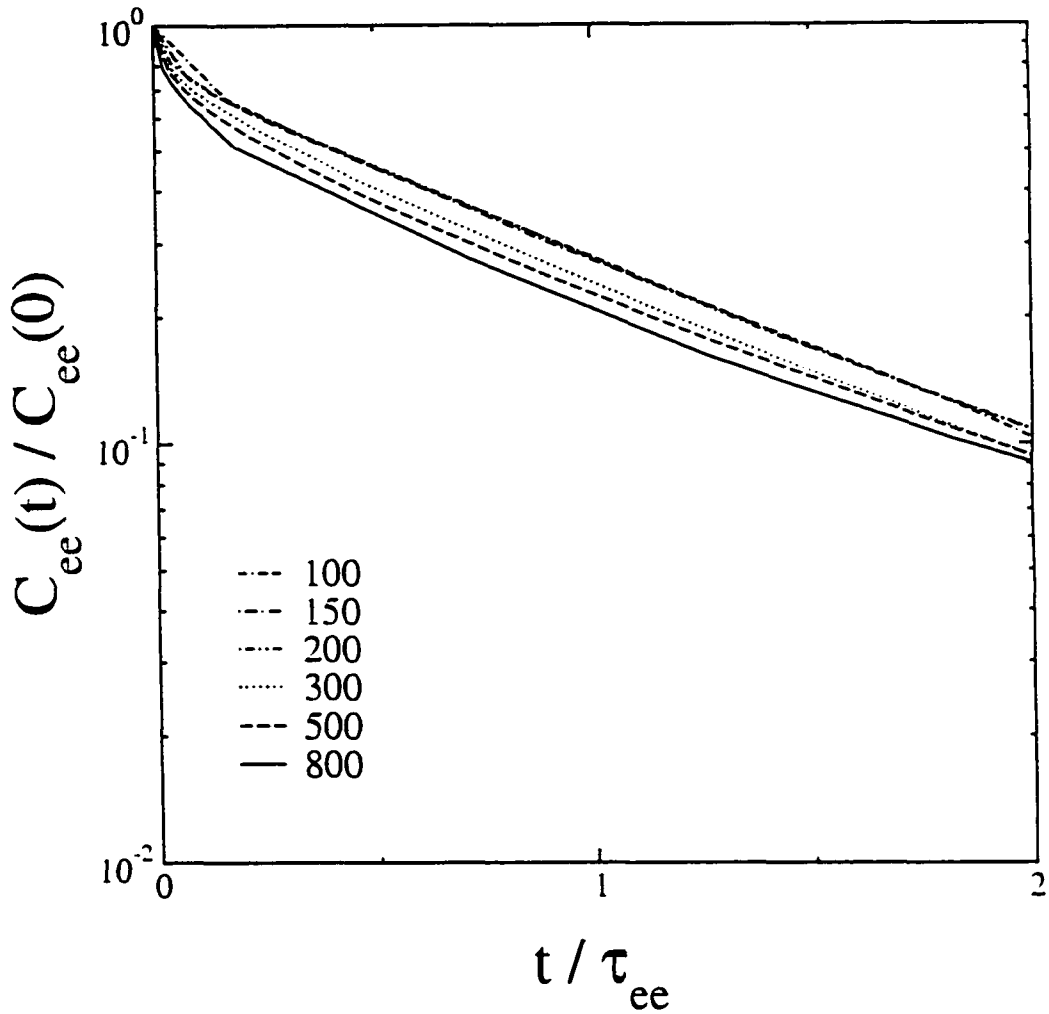

Figure 4.30: Normalized diameter vector auto-correlation function $C_{ee}(t)/C_{ee}(0)$ versus time scaled by the orientational relaxation time t/τ_{ee} for non-crossing rings.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

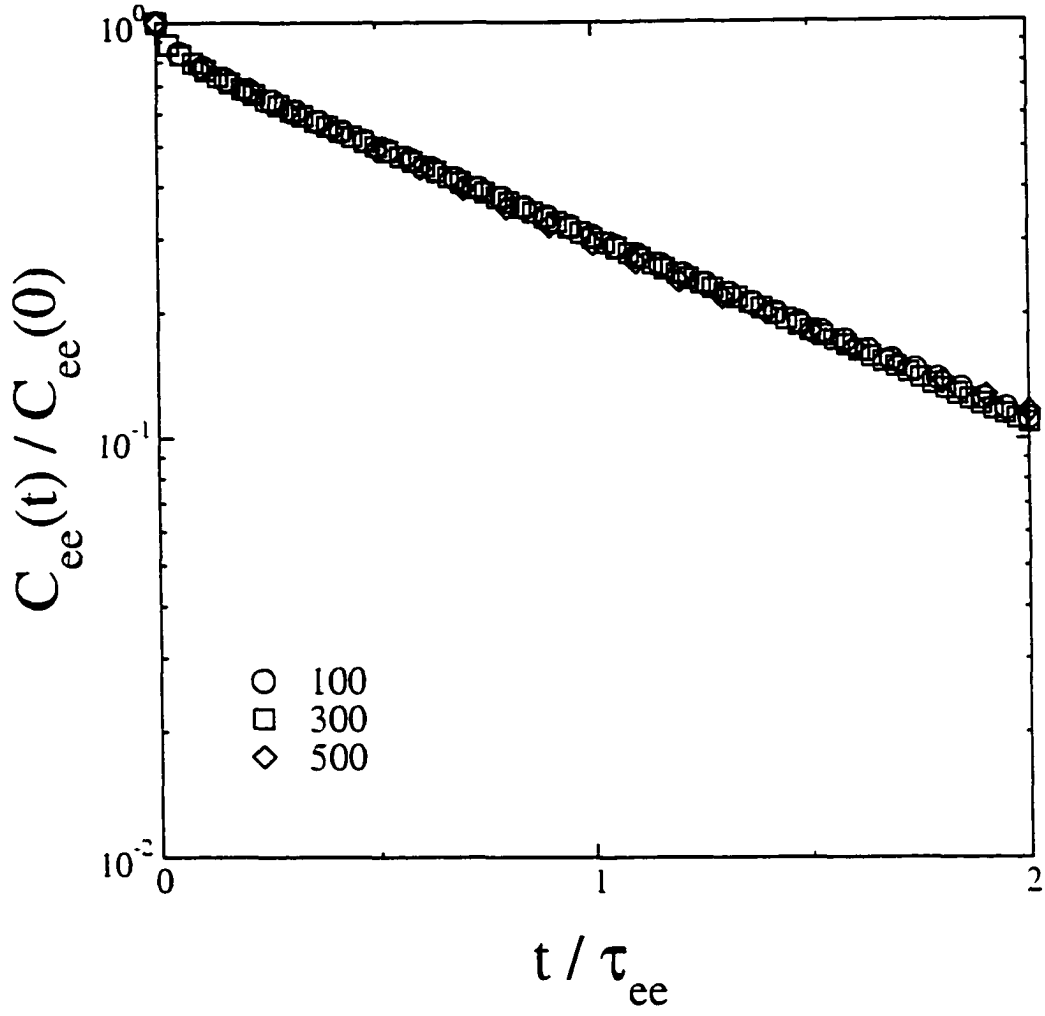

Figure 4.31: Normalized diameter vector auto-correlation function $C_{ee}(t)/C_{ee}(0)$ versus time scaled by the orientational relaxation time t/τ_{ee} for crossing rings.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

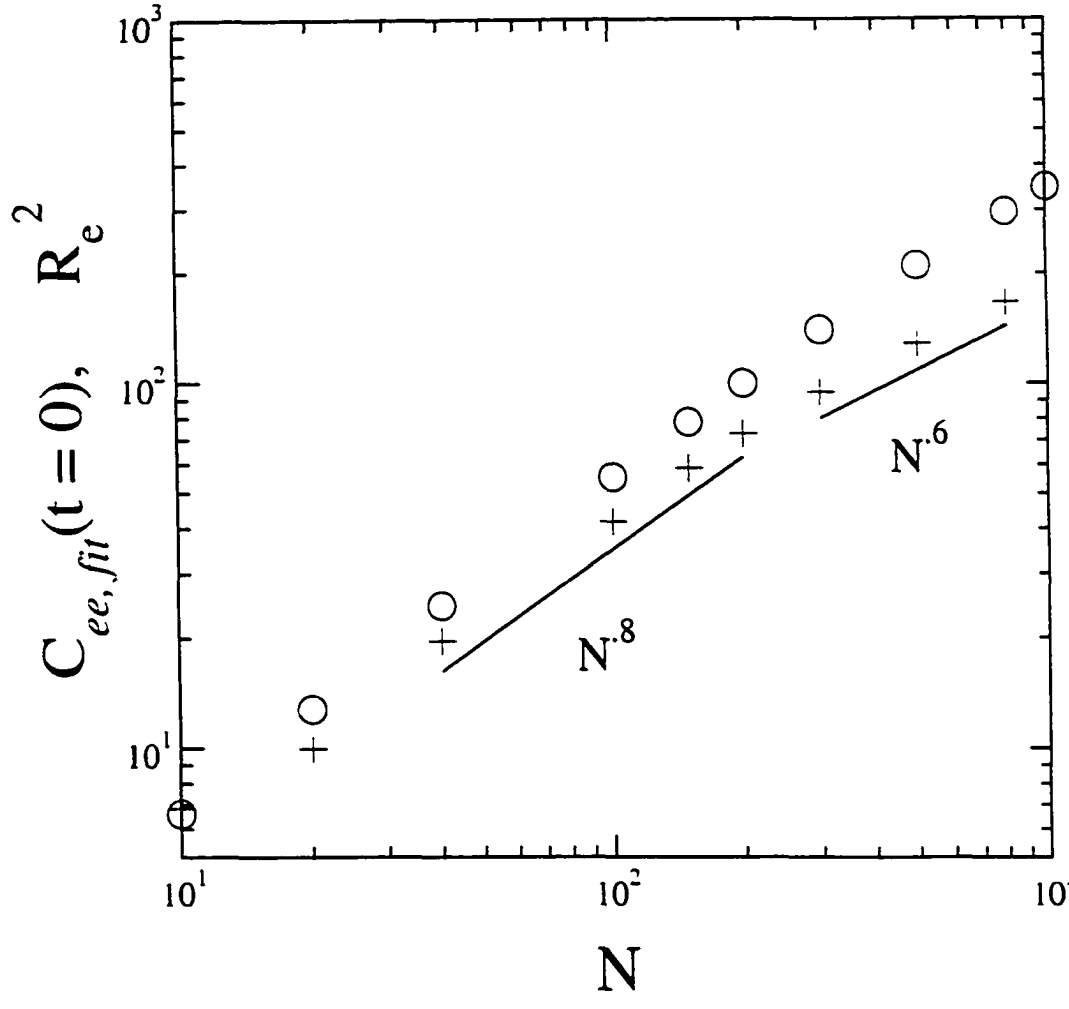

Figure 4.32: Amplitudes obtained from the long-time exponential fits of the diameter vector autocorrelation function $C_{ee,fit}(t = 0)$ versus degree of polymerization N for non-crossing rings. Also included in the plot is data for the mean-square diameter vector R_e^2 (circles).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

self-diffusion coefficient with size for the larger rings and entangled linear chains is quite similar. This implies that size is the determining factor in the dynamics. We know that the linear-chain dynamics is in an entangled (asymptotic) regime for $N > 50$. Could then the larger rings, with their similar dependence on R_g, also be in an asymptotic regime? The evidence appears to indicate that the rings with $N \geq 300$ are at least in a crossover regime to a new dynamics. Further comment on any asymptotic scaling is not possible without simulating larger ring sizes.

However, if we assume an asymptotic form $D \propto N^{-2}$ we can attempt to estimate an entanglement crossover chain length for rings using a formula from Ref.[29]. We fit the self-diffusion coefficient data to the equation $D = c/(N + N^2/N_e)$; where c and N_e are adjustable parameters. From a least-squares fit we obtain $N_{c,ring} = 2 \times N_e \approx 250$. This value is consistent with the observation that rings with $N \geq 300$ are showing evidence for a transition to a new dynamical regime.

4.5 Conclusion

The main result of this Chapter is that rings diffuse faster than linear chains, and that this faster ring diffusion persists up to ring sizes at least twenty times greater than the entanglement crossover chain length of linear chains. Furthermore, we show that, for all ring sizes studied, the dependence of the single-chain relaxation time on ring size is weaker than that expected for linear chains. Additionally we

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

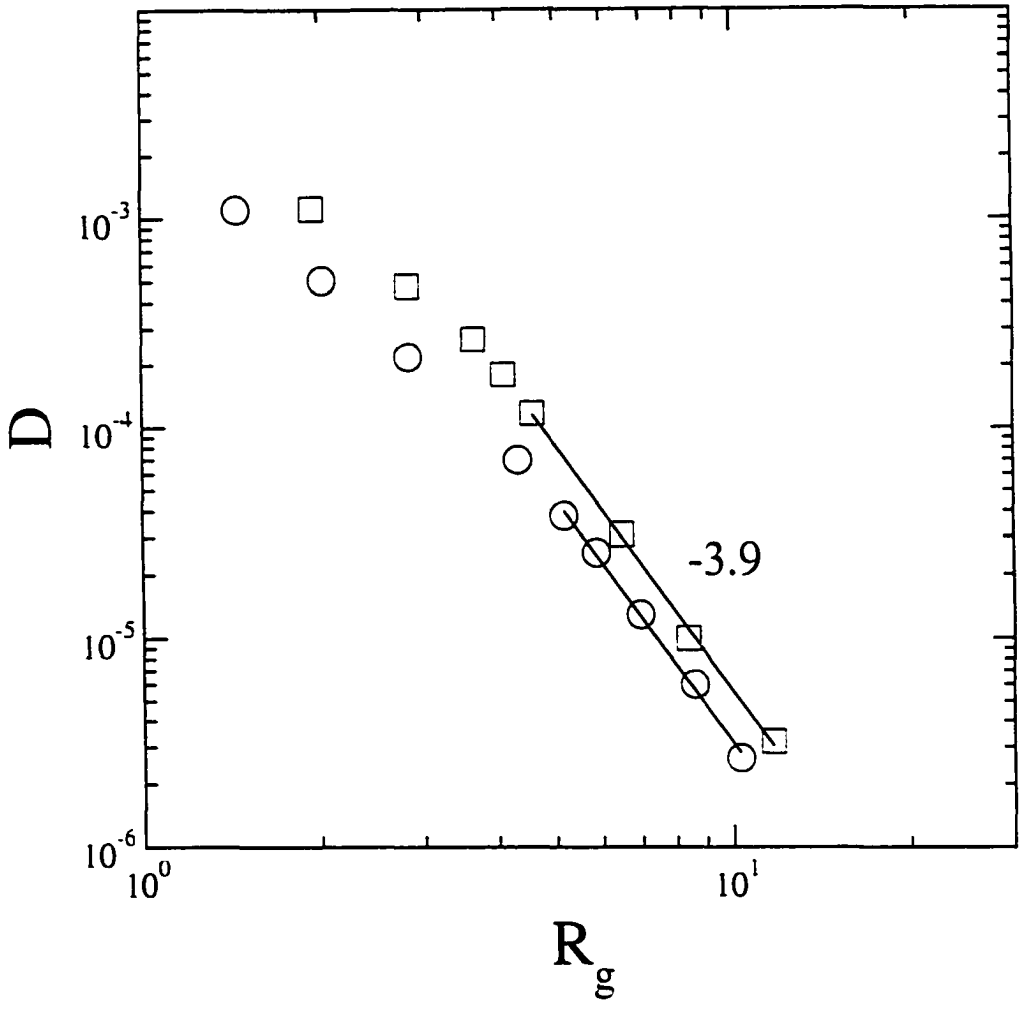

Figure 4.33: Center-of-mass self-diffusion coefficient D versus radius of gyration R_g for non-crossing rings and linear chains. Circles: rings; squares: linear chains (note that the data from our work and Ref.[25] is included here).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

have shown that for the rings with $N \geq 300$ there appears to be the beginning of a transition to a new time scale. We suggest that this is indicative of the presence of longer-lived constraints in the system, and that it implies the onset of some form of new (asymptotic?) dynamics.

We also show that both faster diffusion and faster relaxation of rings can be rationalized by their smaller size: dependence of the self-diffusion coefficient and relaxation time on the radius of gyration of rings and linear chains is remarkably similar.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 5

Three-Arm Star Polymers

5.1 Introduction

In this Chapter we present the results of computer simulations of three-arm star polymers. We have simulated both single stars and stars in the melt. The single star simulations were performed to see whether the simulation model could be applied to star polymers. As such the results from the single star simulations are presented here merely for completeness. The majority of the results from this Chapter have already been published[44].

Our motivation for studying star polymers in the melt is to investigate the different theoretical approaches to melts of entangled linear chains (as discussed in Chapter 2); however, additional incentives to study star polymers in the melt come from their practical importance: most of the polymers used in industry are branched chains of various types. It is known that the dynamical properties

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of branched chains (especially non-linear flow properties) are very different from those of linear chains[45]. The systematic study of branched polymer dynamics has just begun[46,47].

It is believed that in dense systems of star polymers reptation is strongly suppressed. De Gennes[48] and other workers[33,49] have argued that large-length-scale diffusive motions proceed through an "arm-retraction" mechanism which leads to an exponential dependence of both the self-diffusion coefficient and viscosity on the length of the arm. Experimental data of Bartels *et al.*[50] on three-arm star polymer melts shows a stronger-than-power-law dependence of the self-diffusion coefficient on arm length. A plot of their data shows exponential dependence over roughly five orders of magnitude. Similarly, experiments by Fetters *et al.*[51] find that the coefficient of viscosity exhibits a stronger-than-power-law behavior consistent with an exponential increase of viscosity with arm length. These experiments are probably the strongest argument in favor of reptation. It should be emphasized that in these experiments there is no direct evidence for any arm retraction mechanism.

There have been several simulations of isolated star polymers embedded within a lattice of fixed constraints. A simulation by Needs and Edwards[52] suggests an exponential slowing of the dynamics with increasing arm length. A more recent simulation by Barkema and Baumgärtner[53] shows over several orders of magnitude that diffusion decreases exponentially with increasing arm length. The results of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the above simulations are perhaps not entirely surprising as the immobile linear constraints strongly suppress motion of the center bead with respect to motion of the arms, leading to a separation of time scales between the two. This is the underlying assumption of the arm retraction mechanism.

A recent simulation by Sikorski *et al.*[54] of a single star in a matrix of linear chains shows a power-law scaling of the self-diffusion coefficient with arm length; however, the star arm lengths simulated appear to have been too short for an entangled dynamics.

The goal of this study is to investigate the dynamics of entangled melts of star polymers. To the best of our knowledge a simulation of entangled star polymers in the melt has not previously been done. Such a simulation could provide direct evidence for the arm-retraction mechanism, which, again, has never been directly observed.

This Chapter is organized in the following way. The results for the equilibrium and dynamical properties of melts of star polymers are presented, followed by a brief presentation of the main results for the single star simulations.

5.2 Stars in the melt

Table 3.3 lists the main simulation parameters and results for our star melt simulations. We have simulated stars with arm lengths up to $N_{arm} = 150$. As $N_{c,linear} \approx 40$, it is expected that the dynamics for the melt of stars with

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

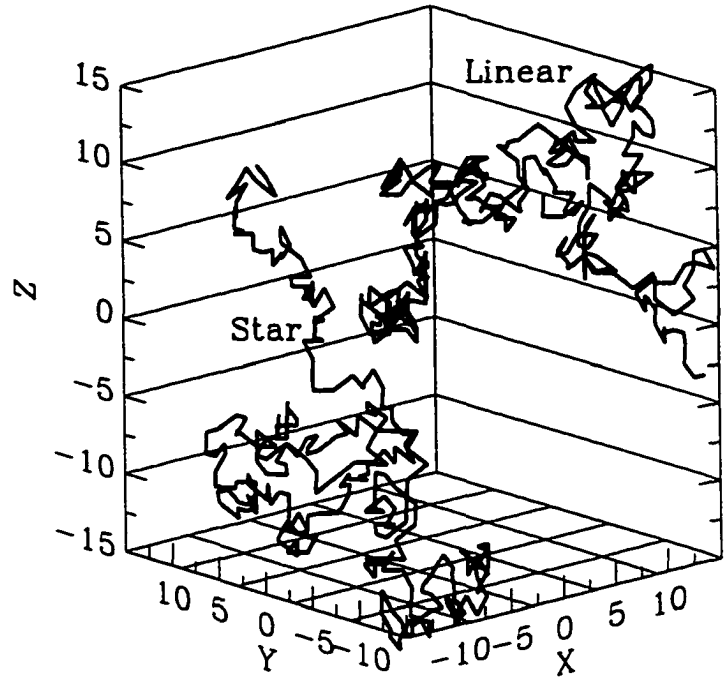

Figure 5.1: Example conformations for 3×90 star and 300mer linear chain (taken from equilibrated melt configurations).

$N_{arm} = 150$ should lie in at least a semi-entangled regime.

5.2.1 Equilibrium properties

Shown in Fig. 5.1 is a typical conformation of a three-arm star polymer compared with that of a linear chain. The two chains appear to have reasonably similar structures.

As in previous chapters we monitor the mean-square radius of gyration R_g^2 in order to quantify the average size of the star polymers. We also use two additional

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

measures which are specific to stars: the mean-square distance between the center bead (branching point) and an arm end bead

$$R_{ce}^2 = \frac{1}{n_{star}} \frac{1}{3} \sum_{\alpha=1}^{n_{star}} \sum_{i=1}^{3} \left\langle \left(\mathbf{R}_c^\alpha - \mathbf{R}_{e,i}^\alpha \right)^2 \right\rangle , \tag{5.1}$$

where $\mathbf{R}_c$ is the position of the star center bead, $\mathbf{R}_{e,i}$ is the position of the end bead on the ith arm (there are three arms), n_{star} is the total number of stars in the system; and the mean-square distance between the end beads of any two arms

$$R_{ee}^2 = \frac{1}{n_{star}} \frac{1}{3} \sum_{\alpha=1}^{n_{star}} \sum_{j>i=1}^{3} \left\langle \left(\mathbf{R}_{e,i}^\alpha - \mathbf{R}_{e,j}^\alpha \right)^2 \right\rangle . \tag{5.2}$$

Shown in Fig. 5.2 is a plot of all three quantities. We observe the same scaling for all three measures of star polymer size, $R_g^2 \propto R_{ce}^2 \propto R_{ee}^2 \propto N$, indicating that excluded volume is being screened.

Fig. 5.3 shows results for the monomer pair-correlation functions $g_{intra}(r)$ and $g_{inter}(r)$. We find that the correlation hole and intra-chain monomer correlations for stars are quite similar to what is seen for linear chains. In Fig. 5.4 we show that the correlation hole deepens with increasing degree of polymerization.

As a further characterization of equilibrium structure, we show the center-of-mass pair-correlation function $g_{cm}(r)$ in Fig. 5.5. The correlation hole in $g_{cm}(r)$ decreases with increasing arm length, indicating that as the stars get larger there is an increasing degree of inter-penetration.

Finally, in Fig. 5.6 we show the master curve for the intra-chain static structure factor $S(k)$. As indicated in the plot, we obtain the power law fit from the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

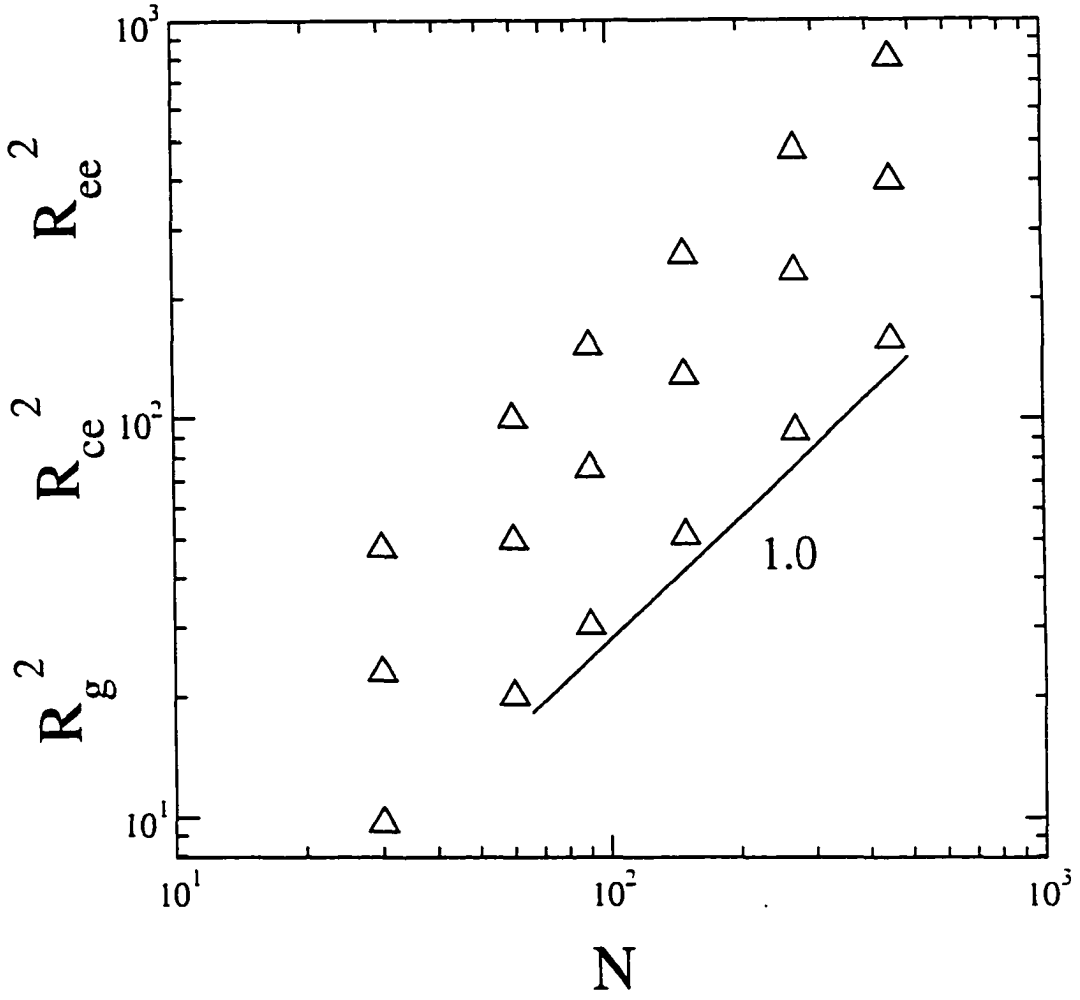

Figure 5.2: Mean-square radius of gyration R_g^2, mean-square center-to-end distance R_{ce}^2 and mean-square distance between arm end beads R_{ee}^2 versus degree of polymerization N for stars in the melt. The line indicates the power law fit with scaling exponent indicated.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

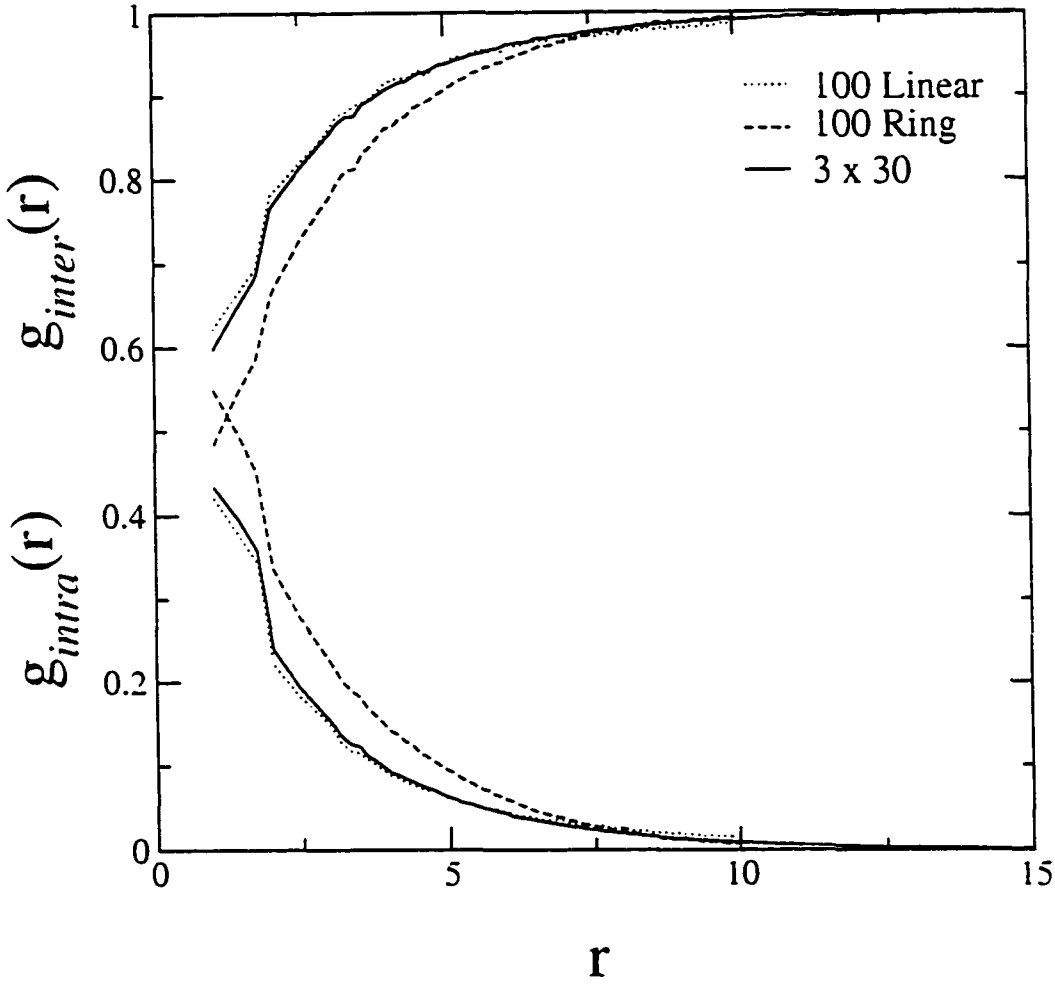

Figure 5.3: Monomer pair-correlation functions $g_{intra}(r)$ and $g_{inter}(r)$ versus separation distance r for selected ring, linear and star polymers.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

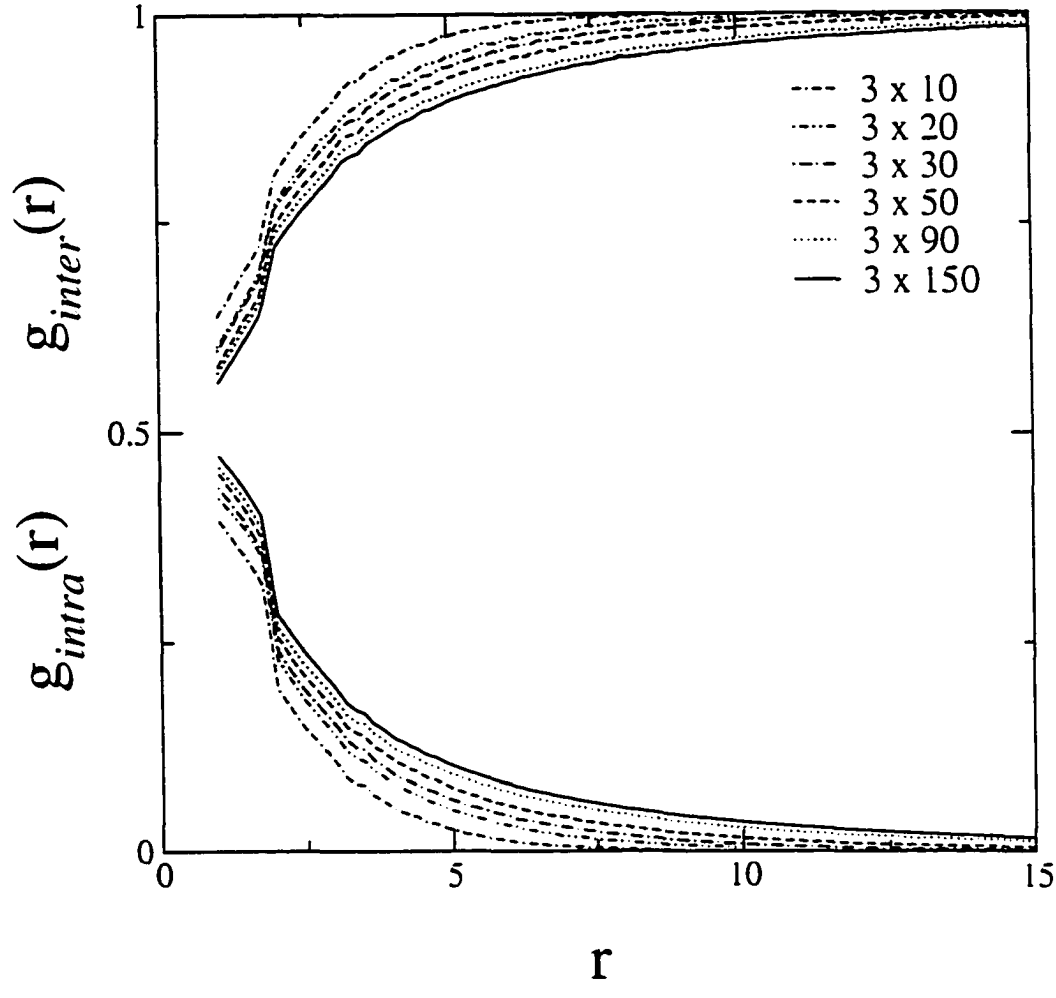

Figure 5.4: Monomer pair-correlation functions $g_{intra}(r)$ and $g_{inter}(r)$ versus distance r for stars of different sizes (as indicated in the legend). The correlation hole in $g_{inter}(r)$ deepens and widens with increasing degree of polymerization.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

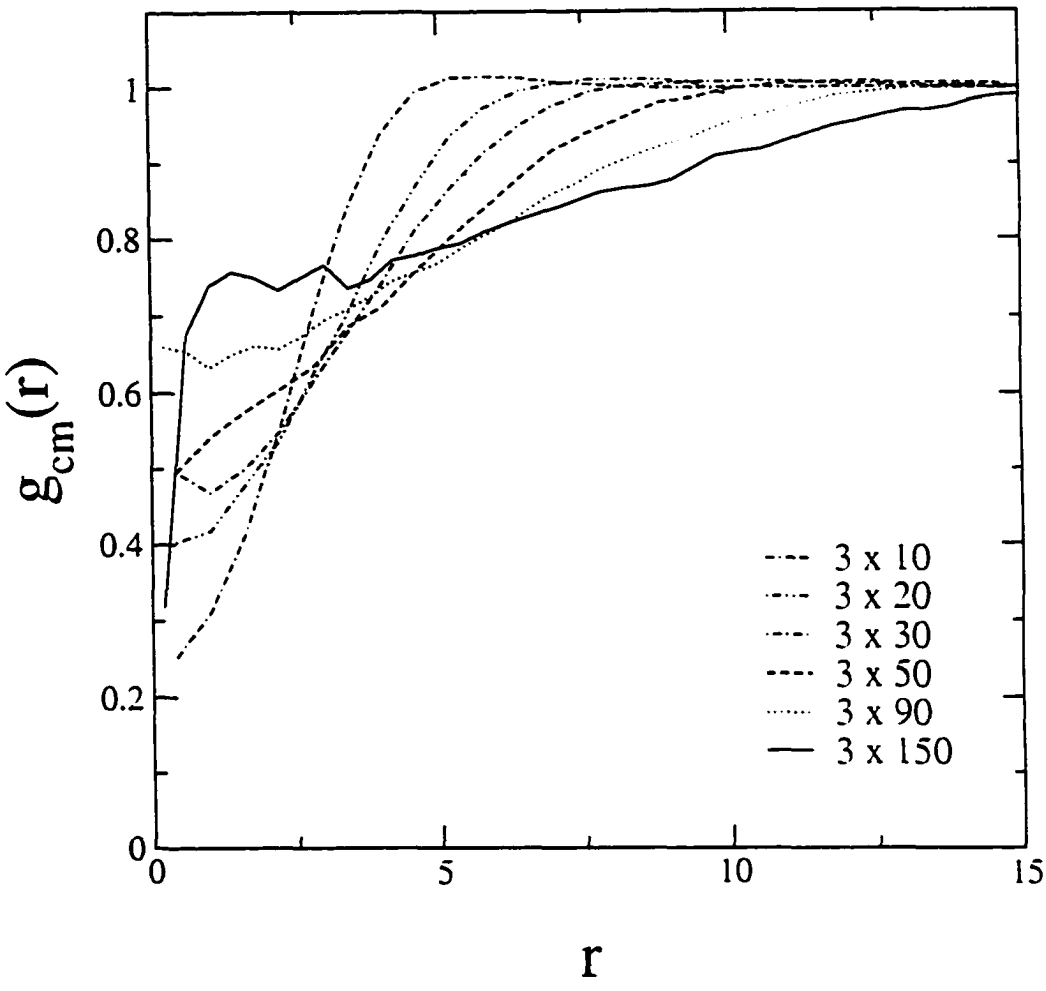

Figure 5.5: Center-of-mass pair-correlation function $g_{cm}(r)$ plotted versus separation distance r for stars in the melt.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

collapsed part of the curve $S(k)/N \propto (kR_g)^{-1/\nu}$ with the exponent $\nu = 0.5$. This is of course one of the signatures of Gaussian statistics, and is consistent with the scaling exponents obtained from the fits of R_g, R_{ce}, and R_{ee}.

5.2.2 Dynamics

As for ring polymers, we first present results for monomer mobility. Shown in Fig. 5.7(a) is the bead mobility along the arm μ_n. There is a significant decrease in μ_n for the center bead ($n = 0$). Of note is that the mobility of intermediate beads along the arm is essentially the same for all arm lengths. Also shown is the average mobility for all the beads in the simulation box $\langle\mu\rangle$ (Fig. 5.7(b)). Apart from small N values the average bead mobility is independent of N.

To analyze time-dependent properties for stars we utilize the same three measures of mean-square displacement as was used for rings: the monomer mean-square displacement $g_1(t)$; the monomer mean-square displacement in the center-of-mass reference frame $g_2(t)$; and the center-of-mass mean-square displacement $g_3(t)$ (see Chapter 4 for the precise definition of these quantities). We also measure the center-to-end vector autocorrelation function

$$C_{ce}(t) = \frac{1}{n_{star}} \sum_{\alpha=1}^{n_{star}} \langle \mathbf{R}_{ce}^{\alpha}(t) \cdot \mathbf{R}_{ce}^{\alpha} \rangle . \tag{5.3}$$

In Fig. 5.8 we show the convergence of $g_3(t)N/6t$, from which values for D are extracted. The self-diffusion coefficients obtained from the plot are shown in Fig. 5.9. We show D plotted in (a) log-log and (b) log-linear fashion. Following

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

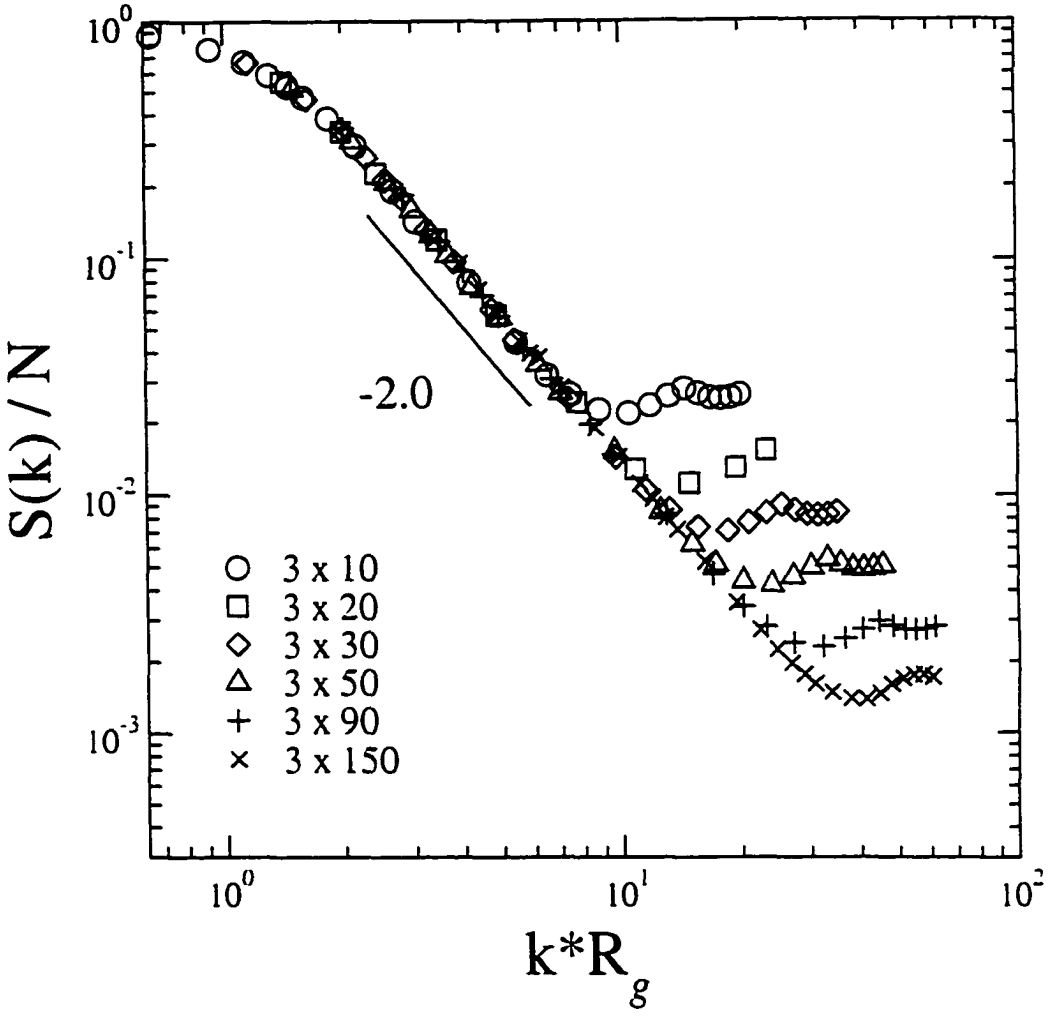

Figure 5.6: Static structure factor scaled by degree of polymerization $S(k)/N$ versus wave vector times the radius of gyration $k*R_g$ for stars. In the intermediate scaling regime the plots reduce down to a single curve. The solid line shows the observed scaling: $S(k)/N \propto (kR_g)^{-2.0}$, which gives a value $\nu = 0.5$ consistent with Gaussian statistics.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

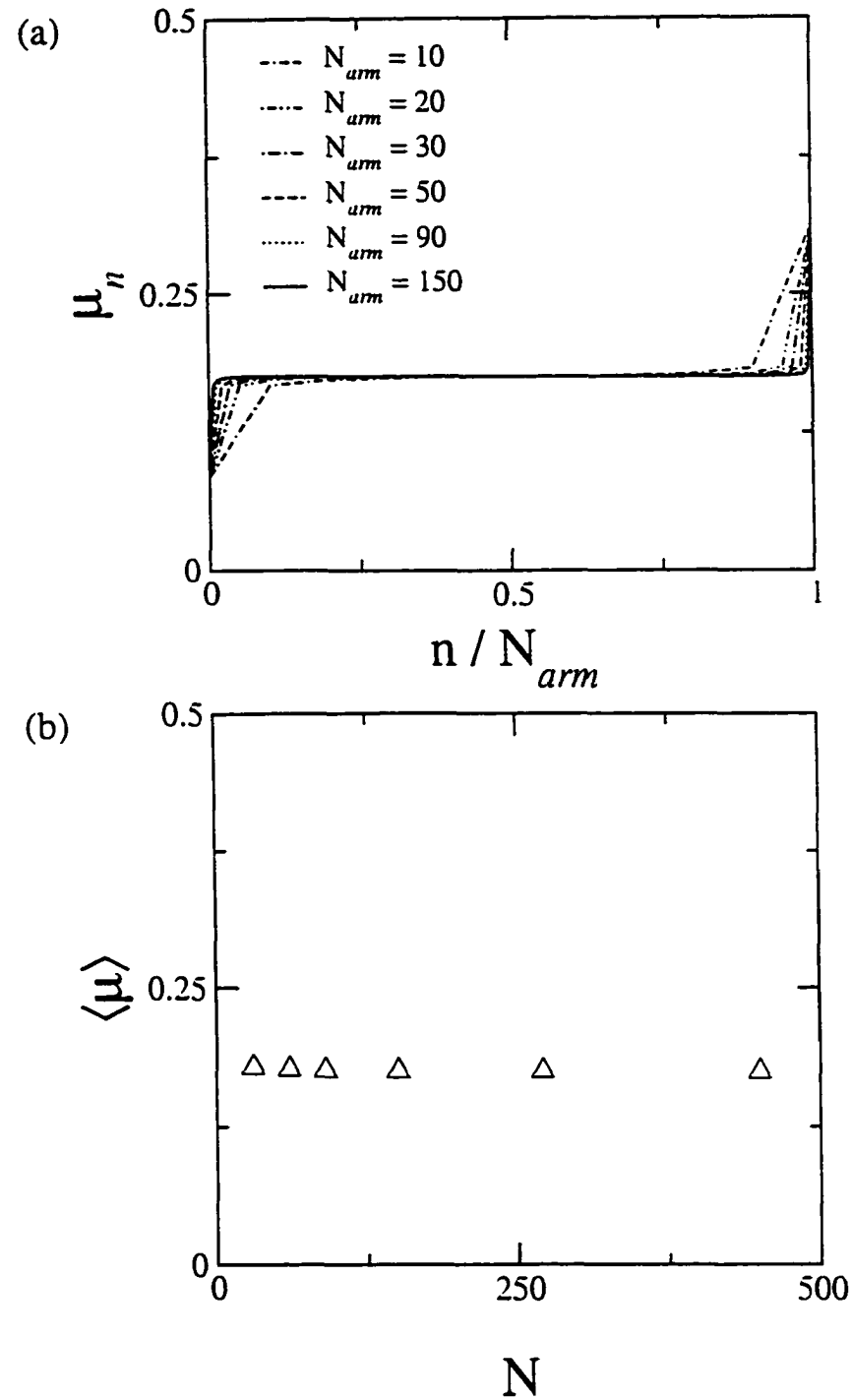

Figure 5.7: (a) Individual bead mobility μ_n versus fractional distance along arm n/N_{arm}. Note that bead $n = 0$ corresponds to the star central bead. (b) Average bead mobility $\langle\mu\rangle$ versus star size N.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the convention used in experimental papers[51] we plot these quantities versus so-called span degree of polymerization, $N_{span} = 2 \times N_{arm}$. We compare to D for linear chains (note that for linear chains $N_{span} = N$).

A power law fit to our data gives $D \propto N_{span}^{-2.3}$. Exponential fit to the experimental data of Bartels *et al.*[50] for star polymers in the melt gives a dependence of the form $D \propto exp(-0.59 M_{arm}/M_e)$. Exponential fit to our data gives $D \propto exp(-0.0128 N_{span})$, which when rearranged into the form of the experimental fit gives $D \propto exp(-0.51 N_{arm}/N_e)$. Graphically it appears that the power law fit is a better representation of the data; however, neither fit appears to be ideal. It is interesting that the exponential fit appears to be in close agreement with the experimental result. This is suggestive that perhaps the data for the largest stars already lies in a crossover regime to an entangled dynamics.

In Fig. 5.10 we plot the mean-square displacements $g_1(t)$, $g_2(t)$ and $g_3(t)$ for all stars. For the center-of-mass displacement, $g_3(t)$, we observe a slower-than-linear time dependence for short times. This is similar to what we observed for the ring polymer simulations; however, for the largest star the value is much smaller than the value for the largest ring (compare 0.69 for $N_{star} = 450$ to 0.81 for $N_{ring} = 800$).

We also show in Fig. 5.10 fits to the monomer mean-square displacement of the form $g_1(t) \propto t^a$. As with the ring polymers the exponent a is N-dependent and monotonically decreasing from a Rouse-like value $a = 0.53$ (for $N_{arm} = 10$)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

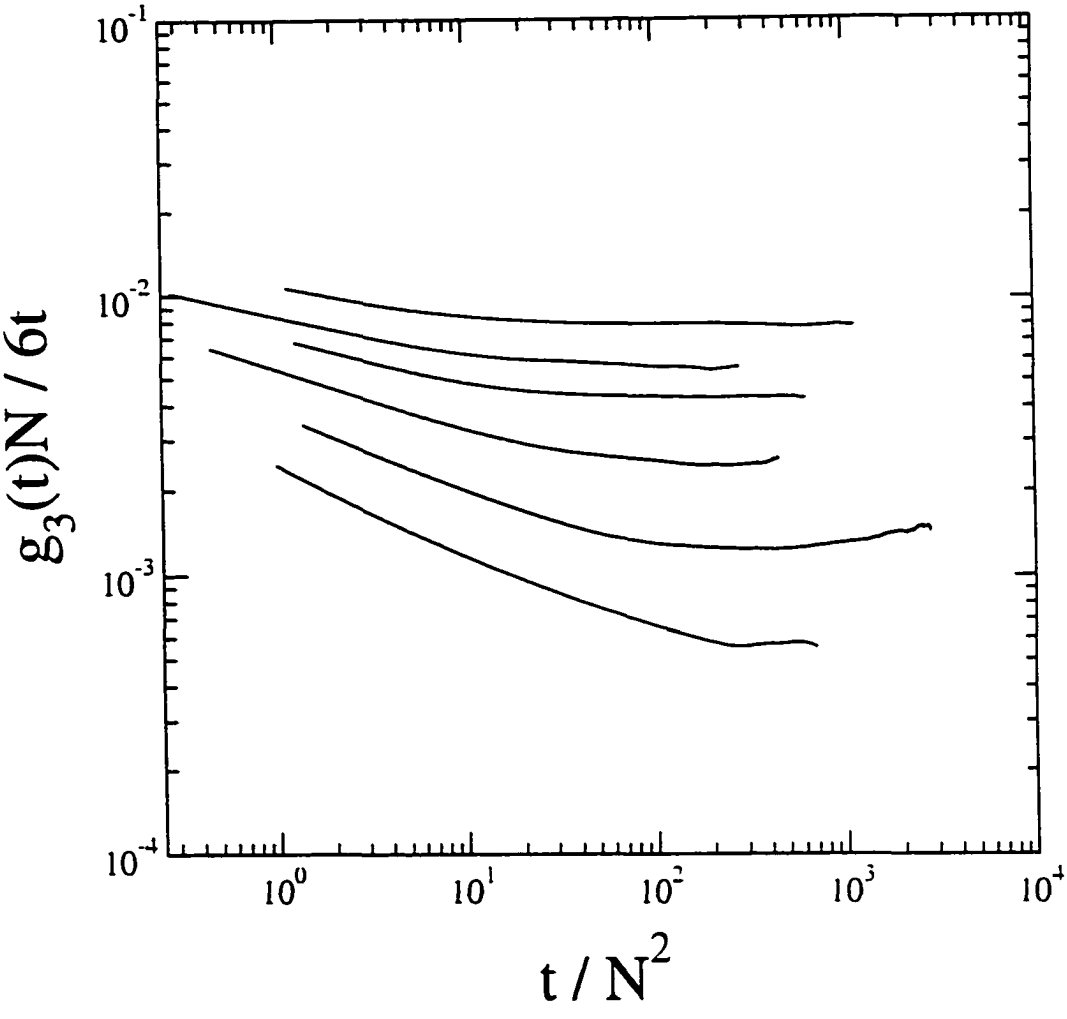

Figure 5.8: Mean-square center-of-mass displacement times degree of polymerization divided by six times time $g_3(t)N/6t$ versus time scaled by the degree of polymerization squared t/N^2 for stars in the melt. Note that the time axis is rescaled merely for convenience.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

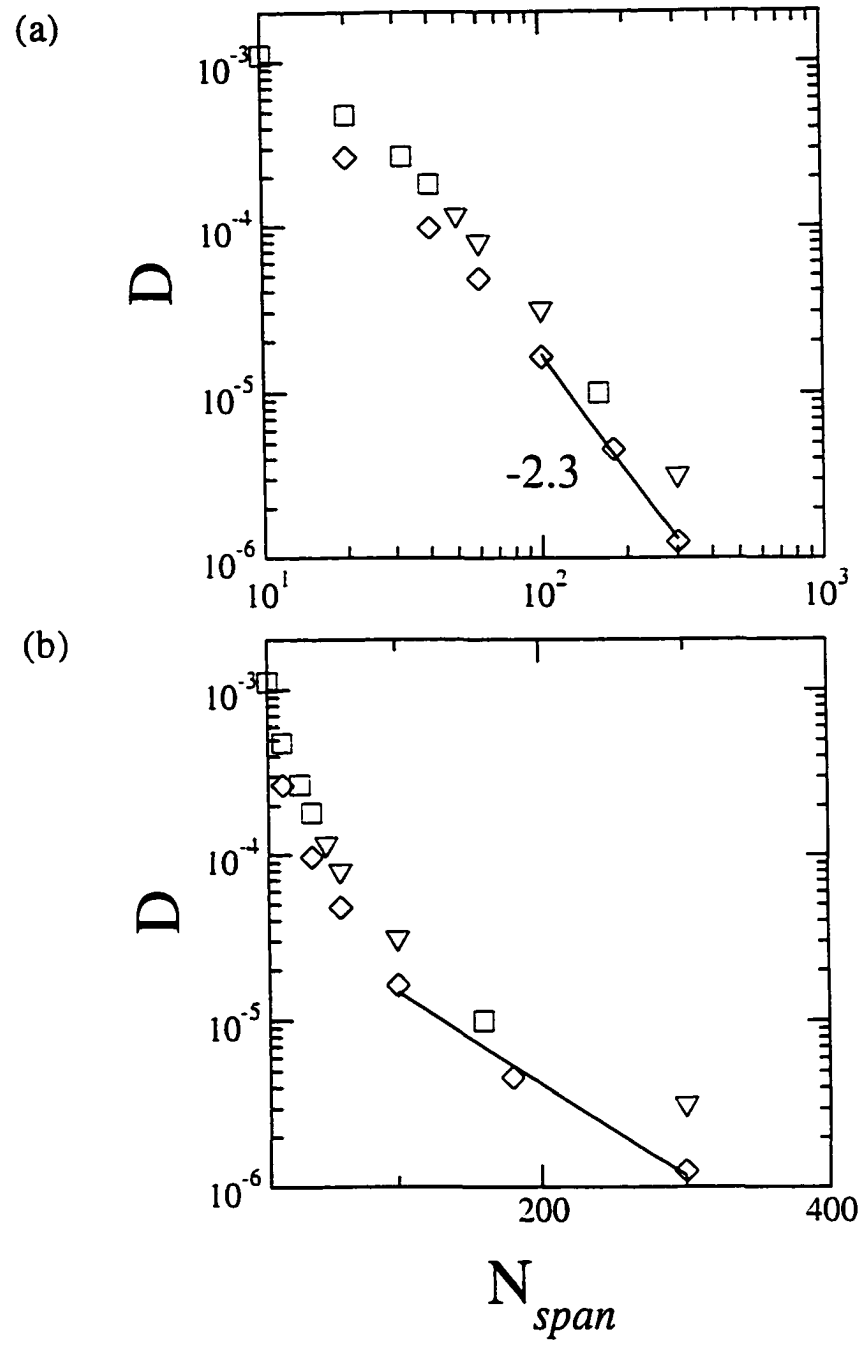

Figure 5.9: Center-of-mass self-diffusion coefficient D for stars and linear chains. (a) Log-log plot of D versus N_{span} ($N_{span} = 2 \times N_{arm}$ for stars; $N_{span} = N$ for linear chains). (b) Same data plotted in log-linear fashion. Diamonds: data for stars; triangles: linear chain data from Ref.[25]; squares: linear chain data from our work. Power law fit to the star data in (a) gives $D \propto N_{span}^{-2.3}$; exponential fit to the star data in (b) gives $D \propto exp(-0.0128 N_{span})$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

to $a = 0.43$ (for $N_{arm} = 150$).

We now look at the decay of the center-to-end vector autocorrelation function $C_{ce}(t)$. In Fig. 5.11 we show $C_{ce}(t)$ for all stars simulated. Fit to the long-time decay of $C_{ce}(t)$ is used to obtain the single-chain orientational relaxation time τ_{ce}.

Shown in Fig. 5.12 is a plot of the orientational relaxation times versus degree of polymerization. As for the self-diffusion coefficient, the data for τ_{ce} appears to be best fit by a power law; however, the data is too limited for any conclusion to be drawn.

5.3 Single stars

As noted in the introduction, the single-star simulations are performed as a check to ensure that there are no problems associated with the star branching point on the lattice. The equilibrium properties should be a reasonable representation of a star polymer in a good solvent. On the other hand, it should be emphasized that our single stars are not a good model for a star polymer in dilute solution, as we do not include any hydrodynamic interactions.

5.3.1 Equilibrium properties

To quantify the average star size we monitor the customary mean-square radius of gyration R_g^2. Additionally we monitor the mean-square distance between center bead and arm end bead R_{ce}^2, and the mean-square distance between arm end beads

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

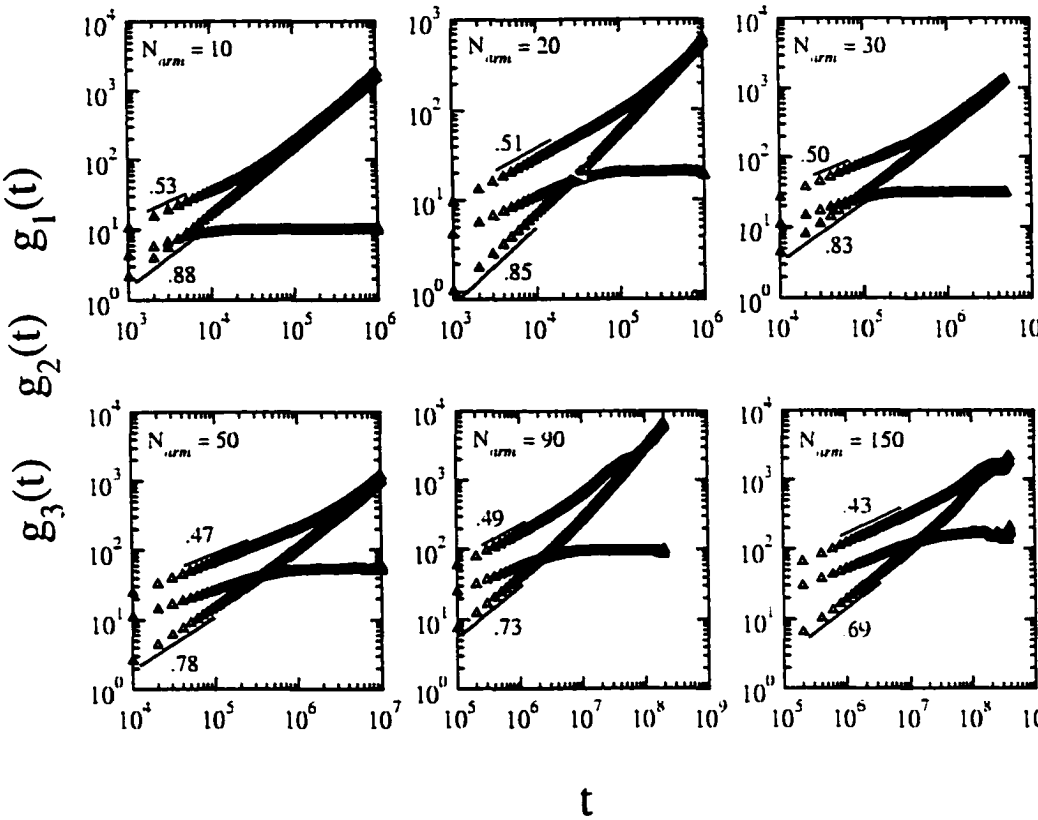

Figure 5.10: Mean-square monomer displacement $g_1(t)$, mean-square monomer displacement in the center-of-mass frame $g_2(t)$, and mean-square center-of-mass displacement $g_3(t)$ versus time t for stars in the melt. The solid lines show power-law fits, with scaling exponent indicated.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

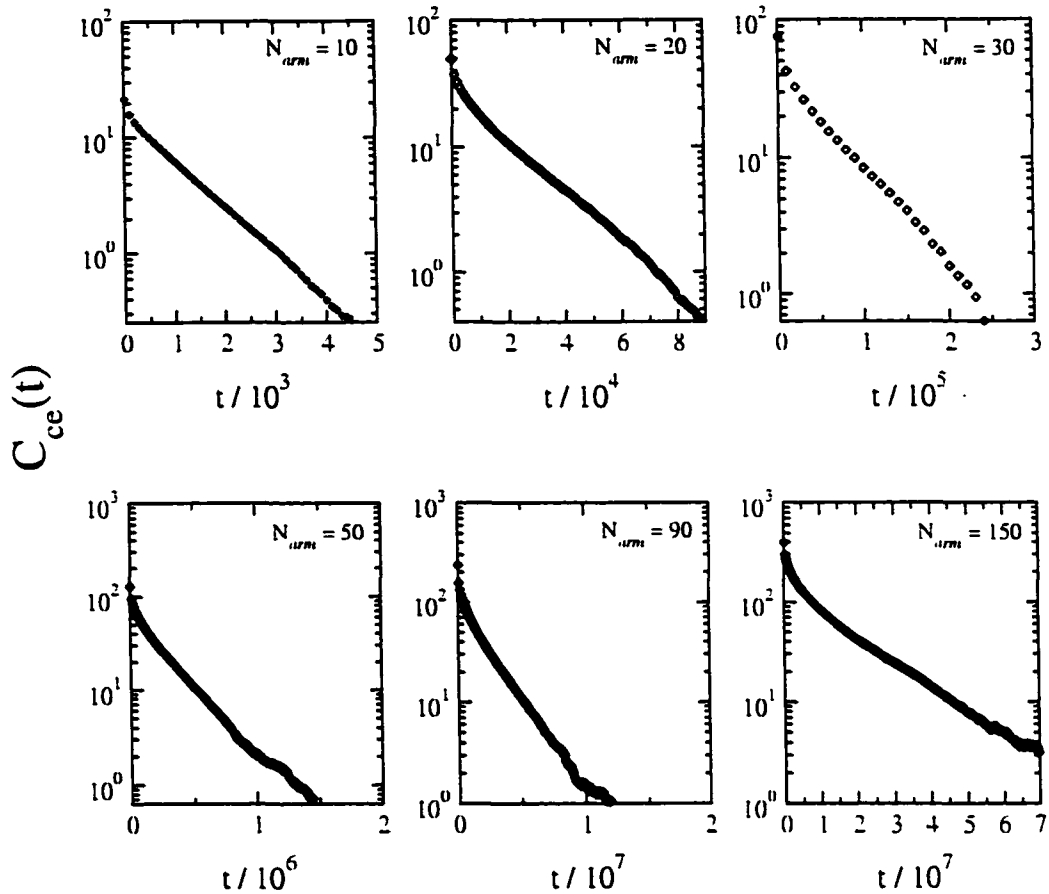

Figure 5.11: Center-to-end vector auto-correlation function $C_{ce}(t)$ versus time t for stars in the melt. Note that for convenience the time axes have been rescaled by differing orders of magnitude (as indicated on each time axis).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

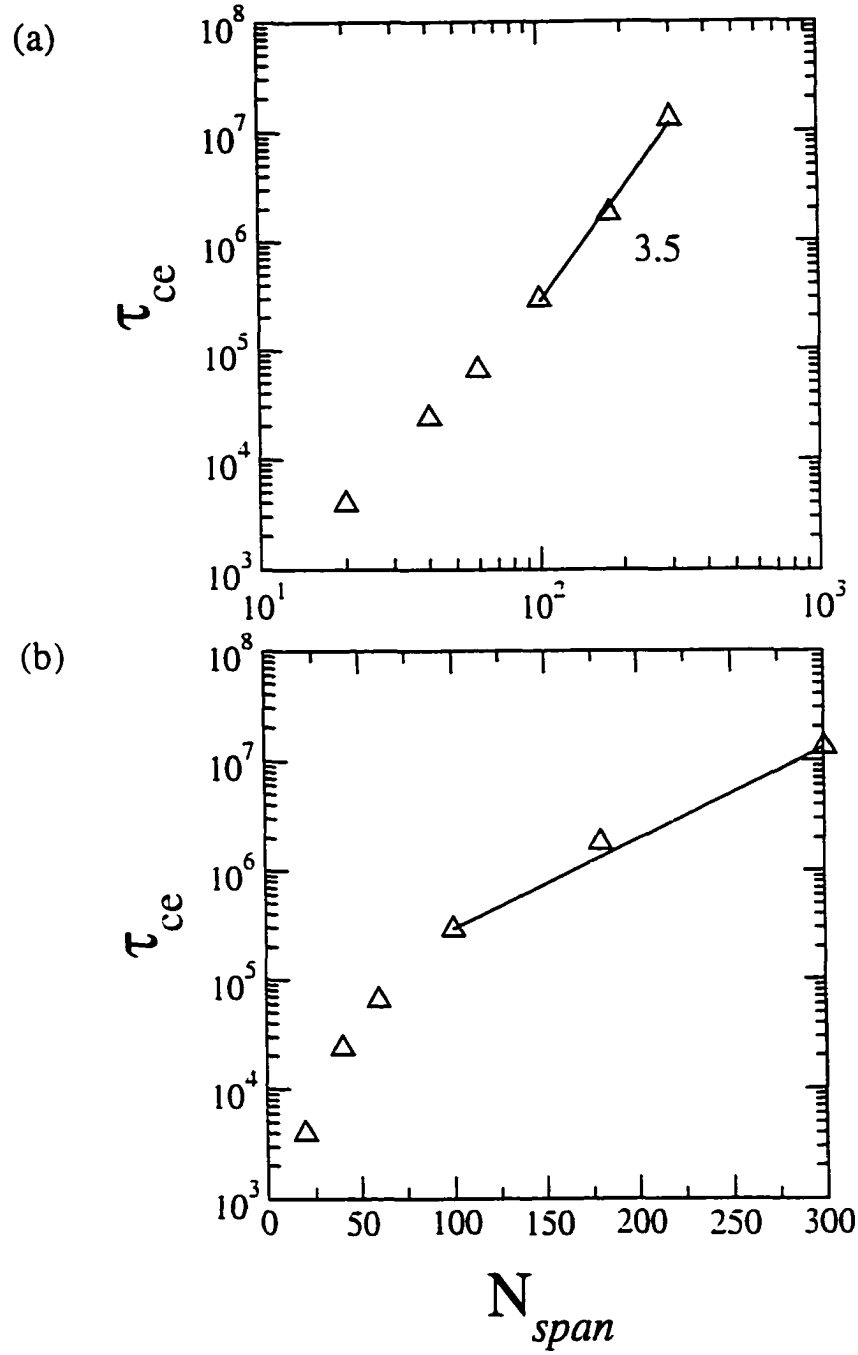

Figure 5.12: Center-to-end vector relaxation time τ_{ce} versus degree of polymerization N. The line in (a) indicates the power law fit $\tau_{ce} \propto N_{span}^{3.5}$. The line in (b) indicates the fit $\tau_{ce} \propto exp(0.019 N_{span})$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

R^2_{ee}. As can be seen from Fig. 5.13 the observed scaling with N is identical for all three measures of star polymer size $R^2_g \propto R^2_{ce} \propto R^2_{ee} \propto N^{1.2}$. The scaling exponent 1.2 agrees with the SAW exponent. This is consistent with results from previous MC lattice simulations by Evans[55], Batoulis & Kremer[56], Sikorski[57], and Molina & Freire[58].

5.3.2 Dynamics

Here we are interested in the mobility of the branching point relative to the mobilities of the other monomers. As with the other sections on dynamics, we introduce a mobility μ_n, which is the fraction of accepted MC moves of the nth bead. In Fig. 5.14(a) we show μ_n as a function of fractional position along the arm. Note that in this plot bead 0 is the branching point (center bead). It can be seen that the center bead has a reduced mobility compared to the other beads. This is of course due to the center bead having three bond constraints upon it, as opposed to two for the other beads inside the star, and one for the arm end beads. μ_n is virtually constant for intermediate values of n. In Fig. 5.14(b) we show the average bead mobility $\langle\mu\rangle$ as a function of star size. For small N $\langle\mu\rangle$ is slightly N dependent due to the end beads (with their significantly higher mobilities) representing a larger fraction of the total number of beads in the simulation. This effect becomes less pronounced as N increases, and for higher N $\langle\mu\rangle$ is essentially constant.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

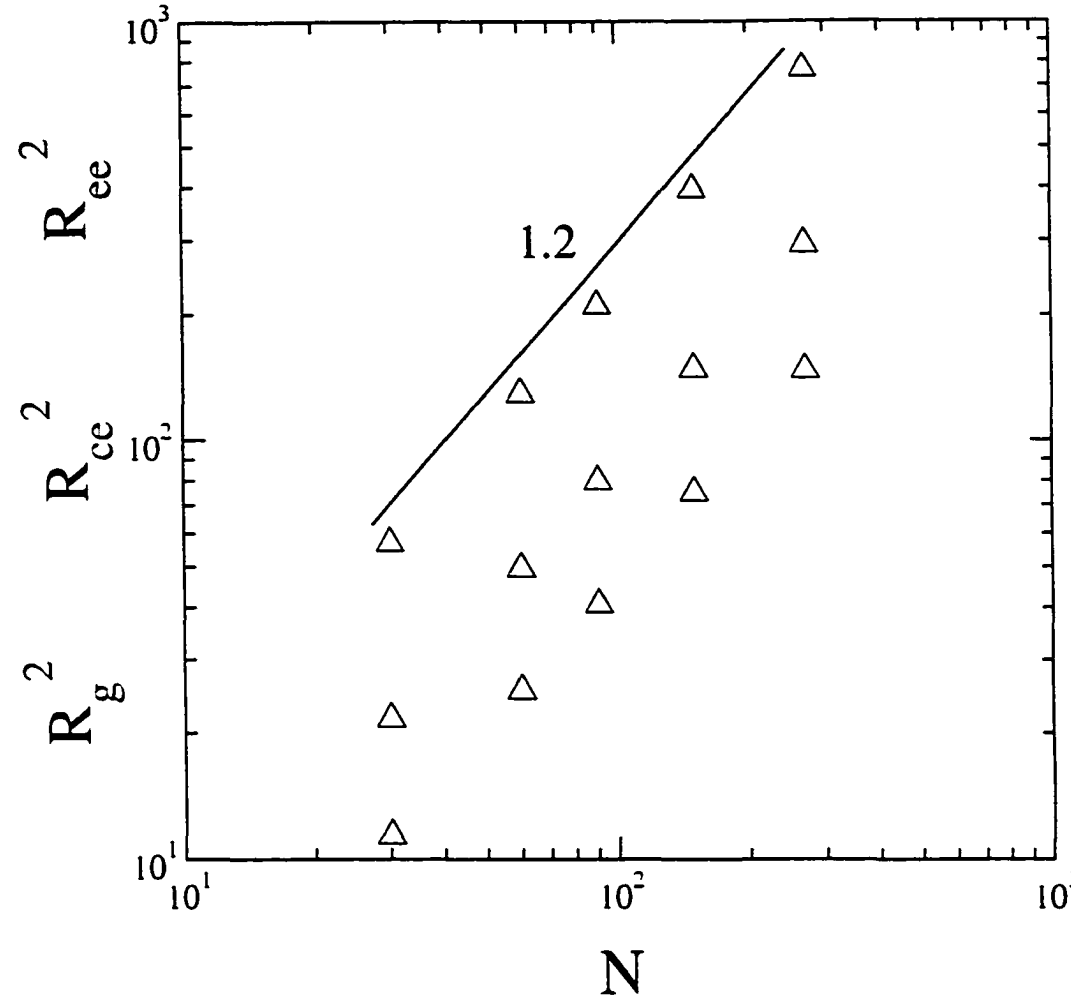

Figure 5.13: Mean-square radius of gyration R_g^2, mean-square center-to-end distance R_{ce}^2 and mean-square distance between arm end beads R_{ee}^2 versus degree of polymerization N for single stars. The line indicates the power law fit with scaling exponent indicated.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

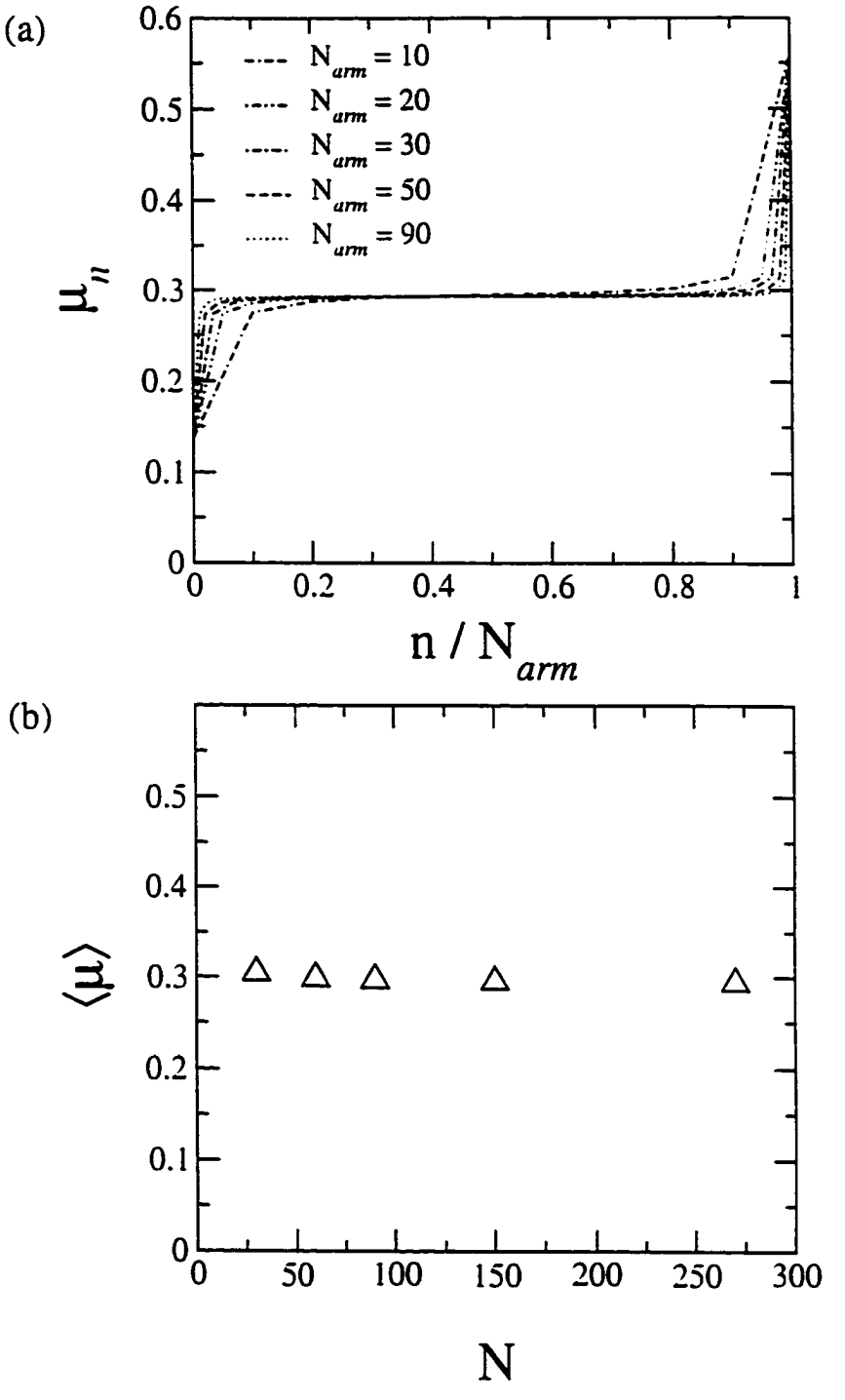

Figure 5.14: (a) Bead mobility along the star arm μ_n versus fractional distance along arm n/N_{arm} for single stars. Bead $n = 0$ corresponds to the star center bead. (b) Average monomer mobility $\langle\mu\rangle$ versus star size N.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

To characterize other dynamical aspects of the single stars we monitor the mean-square center-of-mass displacement $g_3(t)$, and the center-to-end vector auto-correlation function $C_{ce}(t)$, as defined above. From these time-dependent functions we obtain the self-diffusion coefficient D, and the center-to-end vector relaxation time τ_{ce}. The values for these quantities are listed in Table 3.3.

Fig. 5.15 shows results for the self-diffusion coefficient. It can be seen that DN is independent of N, indicating a Rouse-like scaling of $D \propto N^{-1}$. In a single-chain MC simulation by Sikorski[57] a slightly stronger N dependence is found with an exponent -1.1.

In Fig. 5.16 we show the center-to-end vector relaxation time τ_{ce} as a function of N. A power-law fit gives $\tau_{ce} \propto N^{2.3}$. The exponent 2.3 is close to the value 2.18 predicted from scaling arguments for the N_{arm} dependence[19]. From single-star MC simulations, Molina & Freire[58] report an exponent of 2.17, Sikorski[57] reports an exponent of 2.2, and Needs & Edwards[52] report an exponent of 1.94.

5.4 Conclusion

In this Chapter we have presented the results of computer simulations of three-arm star polymers for both single stars and stars in the melt.

For stars in the melt we find that the excluded volume interaction is screened and we observe a scaling for the radius of gyration of $R_g^2 \propto N$. We find that the correlation hole deepens and widens with increasing degree of polymerization

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

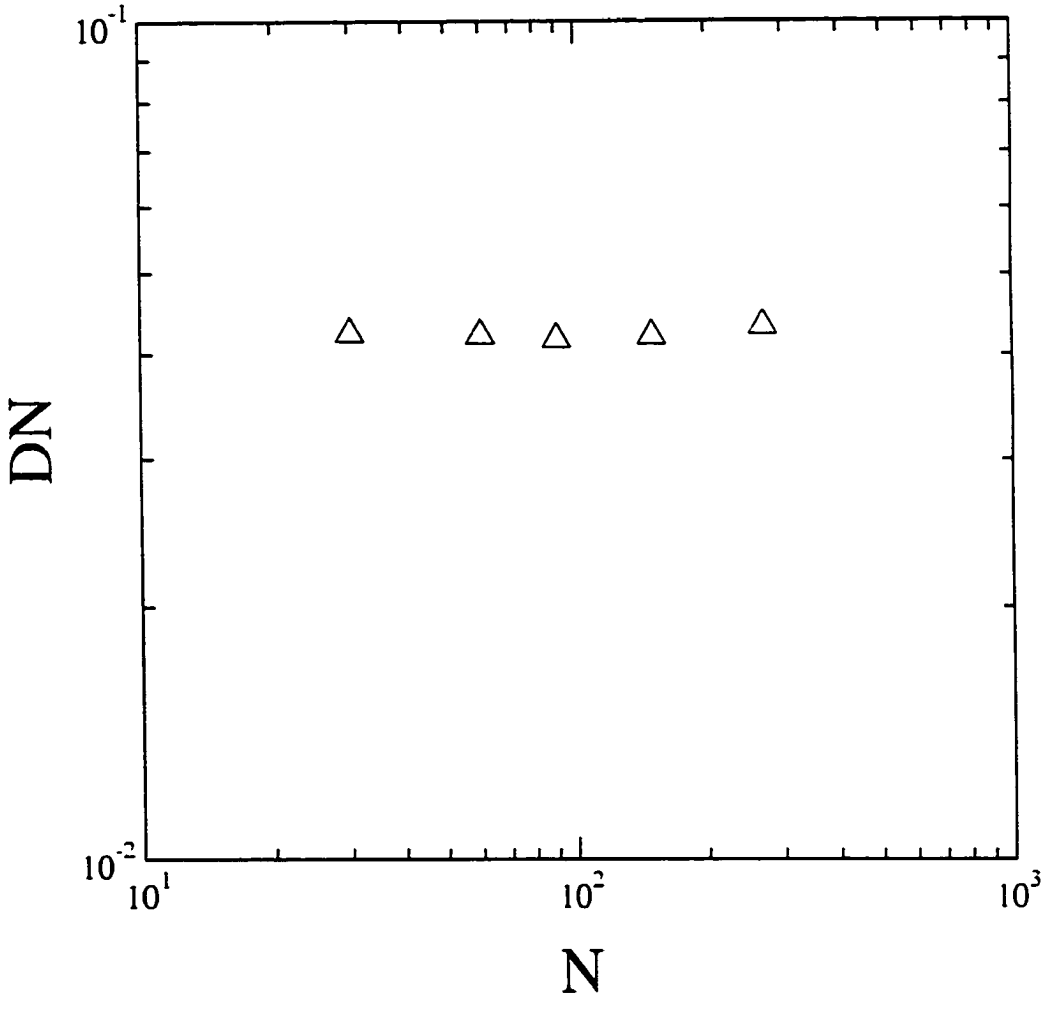

Figure 5.15: Center-of-mass self-diffusion coefficient times degree of polymerization DN versus degree of polymerization N for single stars. For all star sizes $DN \approx const.$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

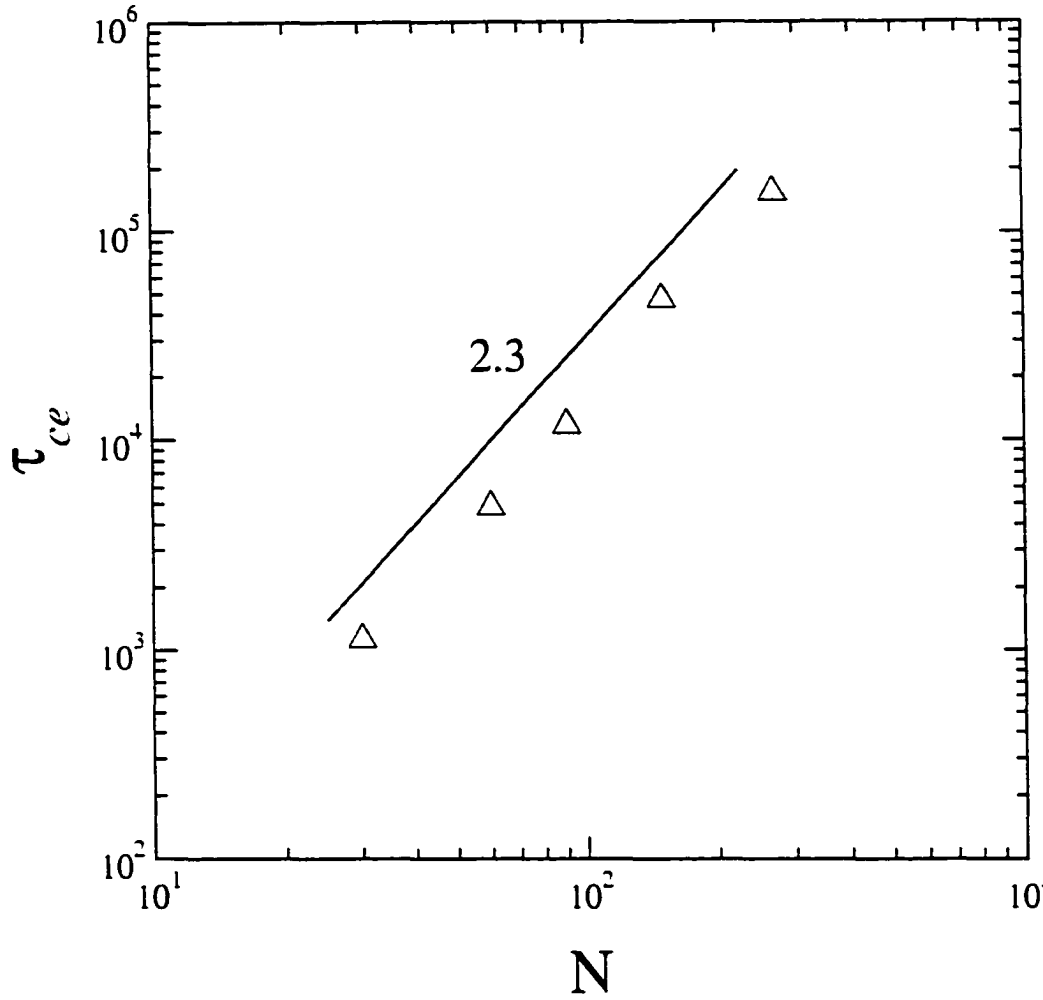

Figure 5.16: Orientational relaxation time τ_{ce} versus degree of polymerization N for single stars. The line indicates power the law fit with scaling exponent indicated.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

for stars in the melt. Furthermore, we show that the extent of inter-penetration increases with the degree of polymerization.

For the dynamics in the melt we observe a possible non-power-law dependence of the self-diffusion coefficient on star size. Our limited data set prevents us from reaching any stronger conclusions. We note that while the exponential fit to the self-diffusion data appears to be of lower quality when compared to the power law fit, it does show close agreement to an exponential fit of experimental data for stars in the melt.

However, we concentrate here on the power law fit to the self-diffusion data for the three largest stars $D \propto N_{span}^{2.3}$. There is a stronger dependence on the degree of polymerization for the dynamics of the three largest stars than for the dynamics of fully entangled linear chains. This seems to imply that the dynamics of the three largest stars lies in an entanglement crossover regime. It seems reasonable then to conclude that the dynamics of the largest star size simulated ($N_{arm} = 150$) should at least lie in a semi-entangled regime.

We find for the single-star simulations that the static and dynamic results agree with what is expected for a single star on the lattice performing constraint-free diffusive motion. We find that the radius of gyration scales as $R_g^2 \propto N^{1.2}$, and the self-diffusion coefficient scales as $D \propto N^{-1}$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 6

Summary

In this dissertation we have looked at the equilibrium structure and dynamics for the non-linear architectures of ring and star polymers. We observe different behavior between the two non-linear architectures when compared with linear chains, and also when compared with each other. The main results for ring and star polymers are summarized below. Following this there is a brief inspection into possible architecturally independent generalizations for the dynamics.

6.1 Ring & star polymer results

The main conclusion for the equilibrium structure of rings in the melt is that the constraint non-concatenation results in rings which are much more compact. It was shown that the radius of gyration for non-crossing rings follows the scaling law $R_g^2 \propto N^{0.82}$. Whereas the radius of gyration for crossing rings, with the absence

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of topological constraints, follows the Gaussian scaling law $R_g^2 \propto N$.

We also find that the non-crossing constraint produces ring polymer configurations which have a much lower degree of inter-penetration than is the case for linear chains of comparable N.

The main result for the dynamics of ring polymers in the melt is that non-crossing rings diffuse faster than linear polymer chains of comparable N, and that faster ring diffusion persists up to ring sizes at least twenty times greater than the entanglement crossover chain length of linear chains. The observed scaling laws for linear chains and rings are $D \propto N^{-2}$ and $D \propto N^{-1.6}$, respectively.

Additionally, we find that the characteristic diffusion time for non-crossing rings has an N-dependence that is much weaker than for linear chains. Specifically, we observe for linear chains and non-crossing rings the scaling $\tau_{diff} \propto N^{3.2}$ and $\tau_{diff} \propto N^{2.4}$, respectively. We also report a scaling of the orientational relaxation time for non-crossing rings of $\tau_{ee} \propto N^{2.5}$.

We demonstrate that the scaling seen for non-crossing rings in the melt is a direct consequence of the topological constraint of uncrossability. In particular, we demonstrate that removal of the constraint of uncrossability results in a scaling for the self-diffusion coefficient of $D \propto N^{-1}$ for crossing rings in the melt. Moreover, we show that for crossing rings in the melt the characteristic diffusion time and the orientational relaxation time follow the power law scaling $\tau_{diff} \propto \tau_{ee} \propto N^{2.0}$. These scaling laws characterize Rouse-like dynamics, and hence show that removal

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of the constraint of uncrossability in a melt of ring polymers leads to unentangled dynamics.

Finally, we show that the constraint of uncrossability has little effect on static and dynamic quantities for single rings. For both crossing and non-crossing single rings we observe that the radius of gyration scales as $R_g^2 \propto N^{1.17}$. Also we observe a self-diffusion coefficient and orientational relaxation time which scale identically for crossing and non-crossing single rings as $D \propto N^{-1}$ and $\tau_{ee} \propto N^{2.1}$, respectively.

The main results for the equilibrium structure of three-arm star polymers in the melt is that the excluded volume interaction is fully screened and we observe the scaling $R_g^2 \propto N$. We also show that the inter-penetration in a melt of stars increases with the degree of polymerization.

For the dynamics of three-arm stars in the melt we observe a possible non-power-law dependence of the self-diffusion coefficient on star size. Our limited data prevents any further comment from being made. We argue that the largest star size simulated, $N_{arm} = 150$, is at least in a semi-entangled dynamical regime.

For the single star simulations the static and dynamic results agree with what is expected for a single chain on the lattice executing constraint-free diffusive motion. We find that the mean-square radius of gyration scales as $R_g^2 \propto N$, and the self-diffusion coefficient as $D \propto N^{-1}$. It is also shown that the orientational relaxation time scales as $\tau \propto N^{2.3}$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.2 Size determines dynamics?

In the Chapter 4 discussion of ring polymer dynamics, a comparison of $D(R_g)$ for rings and linear chains revealed a surprisingly similar dependence. This was used to explain faster ring diffusion for a given degree of polymerization N based upon the smaller ring size. We now include star polymers, and look at the the data for all three architectures.

In Fig. 6.1 we plot D versus R_g for all ring, star and linear polymers simulated. The plot includes two different fits to the star polymer data. The first fit is done in a similar range of R_g as the fits for rings and linear chains, and gives the scaling $D \propto R_g^{-4.0}$. This fit agrees well with the ring and linear chain fits (recall that both fits in a similar range gave $D \propto R_g^{-3.9}$).

However, a second fit to the three largest star sizes gives $D \propto R_g^{-4.6}$. Note that fits to the three largest rings and three largest linear chains give $D \propto R_g^{-4.0}$ (not shown in plot). This result suggests that the star polymer dependence for $D(R_g)$ obtained from the first fit is only a finite-N crossover effect.

It is interesting that a very similar agreement is obtained between all architectures for $D(R_g)$ when the fit includes only the intermediate range of R_g values for the stars. At least for the star polymers this appears to be only a finite-N result. Of course it doesn't resolve any issue regarding the similar dependence seen for $D(R_g)$ between rings and linear chains. It does, however, emphasize the need for simulations of larger ring sizes.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

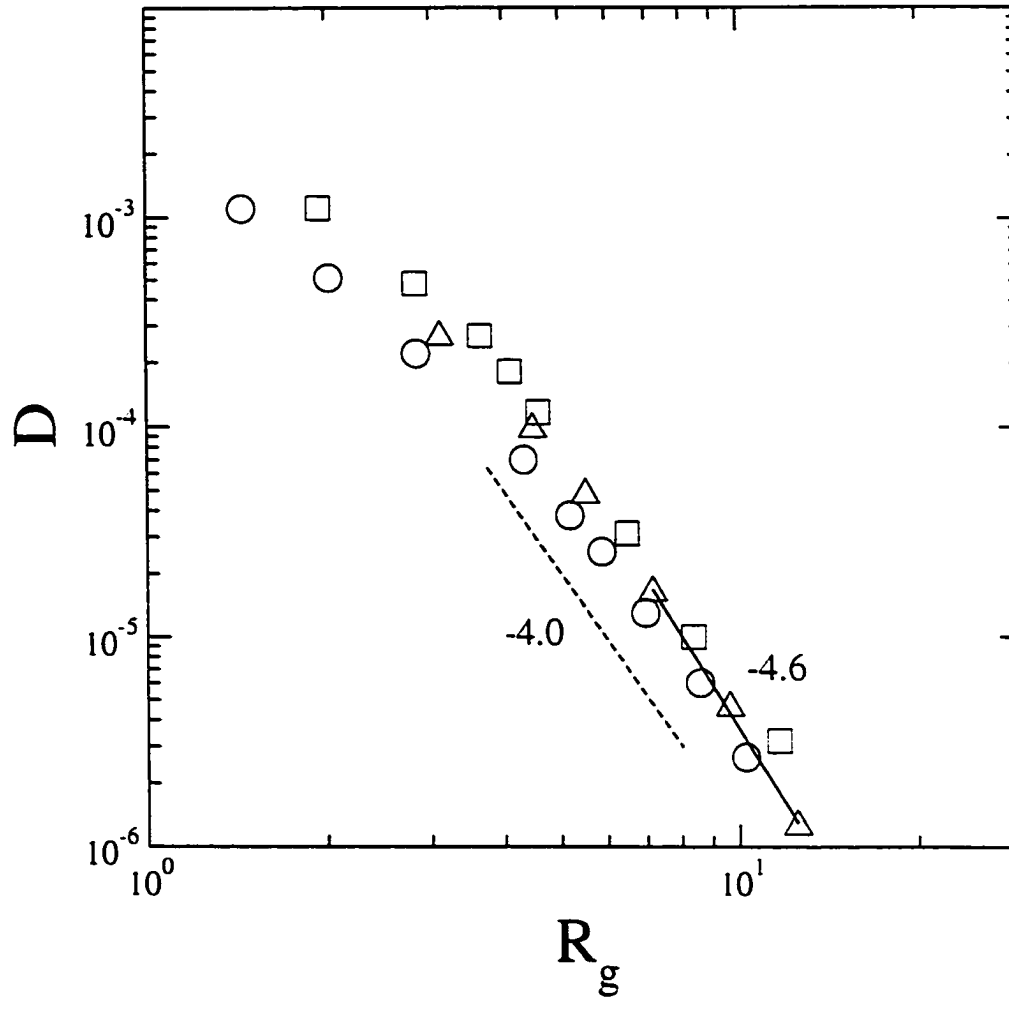

Figure 6.1: Center-of-mass self-diffusion coefficient D versus radius of gyration R_g for non-crossing rings, stars and linear chains. Circles: rings; squares: linear chains (note that data from our work and Ref.[25] is included here); triangles: stars. Two different fits to the star data are shown. The first fit (dashed line) shows the scaling $D \propto R_g^{-4.0}$, which is in close agreement to the ring and linear chain fits in a similar range $D \propto R_g^{-3.9}$. A fit to the three largest stars (solid line) gives $D \propto R_g^{-4.6}$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.3 What next?

To resolve the issues regarding ring polymer dynamics, obviously it is desirable to perform simulations of larger ring sizes. This is relatively straightforward and is simply a matter of obtaining computers with greater computational power.

However, even with data from larger ring polymer simulations we are still faced with the more challenging problem of developing a theoretical explanation for the observed ring dynamics. As a next logical step then, it makes sense to attempt to construct a model of ring polymer dynamics which accounts for the observed behavior. Linear polymer chains have the snake-like motions of reptation, and star polymers have the arm retraction mechanism, what characteristic motions enable ring polymer dynamics? A reasonable starting point for developing an intuitive picture of ring polymer motion would be to understand the change that occurs in the scaling of the amplitude associated with the long-time exponential fit to $C_{ee}(t)$. An initial question to address might be, under what conditions does one expect to observe the $N^{0.6}$ scaling law? Speculation along these lines could allow comment to be made on the asymptotic scaling expected for ring polymer self-diffusion. In an ideal scenario such predictions for ring dynamics would stimulate further experimental studies of ring polymers.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix A

Ring source code

```
      program rngncrv11

c Stochastic lattice dynamics simulation of a system of cyclic,
c monodisperse polymer chains with excluded volume in a periodic box.

c In this implementation, the polymer chains are modeled with the
c ONE-SITE bond fluctuation model. The stochastic dynamics is a simple
c dynamic Monte Carlo process.

c In our implementation of the ONE-SITE bond fluctuation model, each
c polymer bead occupies one site on the lattice. The bonds between
c connected beads are restricted to the set (1,0,0), (1,1,0), and
c (1,1,1). Excluded volume conditions are enforced for all polymer
c beads by restricting the occupancy of the lattice sites to at most one
c polymer bead. An array of occupation numbers for the primary lattice,
c on which the beads reside, is maintained to check for bead overlaps.

c To forbid the crossing of polymer chains during the dynamics, we also
c track the MIDPOINTS of the polymer bonds. These coordinates of bond
c midpoints are always integer multiples of 1/2, so we maintain an
c additional lattice of occupation numbers for bond midpoints. This
c lattice has twice the number of entries in each coordinate direction
c as the lattice of occupation numbers for polymer beads. Any bond
c crossing must involve the simultaneous overlap of the coordinates of
c two bond midpoints, so this bond crossing condition is easy to detect.

c Note that the occupation numbers for the beads must be either zero or
c one because the excluded volume condition is always enforced. In
c contrast, the occupation numbers for the bond midpoints may be greater
c than one, because several bond midpoints can overlap when the chains
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
c are allowed to cross.

c The connectivity information is provided by the array KCONN, described
c below, so that polydisperse systems can be simulated with relative
c ease, if we choose do so later.

c The program can use previously equilibrated configuration
c if the flag equil is set to TRUE

c PARAMETERS                                                          40
c ----------
c
c     N.....Number of beads per polymer chain
c     npoly.....Number of polymer chains
c     nbead.....Total number of polymer beads
c     nx,ny,nz.....Size of the lattice box for polymer segments
c     nxmid,
c     nymid,
c     nzmid.....Size of the lattice box for bond midpoints
c                                                                     50
c     nrelax.....Estimate of the relaxation time
c     kdata.....Number of MCS between accumulation of data
c     ndata.....Number of data points that will be calculated.
c
c        (The total number of MCS will be KDATA*NDATA.)
c
c     equil.....Logical flag: TRUE if equilibriated configuration
c                       is used; FALSE otherwise
c
c     rpoly.....number of chains whose bead displacements will be tracked   60
c     rbead.....total number of beads in the rpoly polymer chains
c     maxbin.....max number of bins in radial distribution function
c
c ----------

      implicit none

      common/block1/cmpoly,r,Rg,lpoly,hist

      integer       N, npoly, nbead, nx,ny,nz, nxmid,nymid,nzmid       70
      integer       nrelax, kdata, ndata, rpoly, rbead

c note these parameters need to be set in subroutines core, data, and start

      parameter     (N     = 1000)
      parameter     (npoly = 65)
      parameter     (rpoly = 1)

      parameter     (nbead = N*npoly)
      parameter     (rbead = N*rpoly)                                   80
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      parameter    (nx = 50)
      parameter    (ny = 50)
      parameter    (nz = 52)

      parameter    (nxmid = 2*nx)
      parameter    (nymid = 2*ny)
      parameter    (nzmid = 2*nz)

      parameter    (kdata = 250000)                                    90
      parameter    (ndata = 2000)
      parameter    (nrelax = 406700000)

      logical      equil
      parameter    (equil = .true.)

c runmax is the total number of desired independent trajectories

      integer      runmax
      parameter    (runmax = 1)                                        100

c nstream is the number of desired stream.coord files for _each_
c trajectory.  nconfig evenly divides up the total number of
c configurations for a single trajectory into the nstream
c stream.coord files

c maxstream is the total number of stream.coord files for all
c trajectories
c you specify the names of the stream.coord files in the array
c "coordfile"                                                          110

      integer      nstream, nconfig, maxstream
      parameter    (nstream = 10)
      parameter    (nconfig = ndata/nstream)
      parameter    (maxstream = nstream*runmax)

      integer      maxbin
      parameter    (maxbin = 125)

c VARIABLES                                                            120
c ———
c
c     lpoly.....Lattice coordinates of all polymer beads
c     locc.....Lattice occupation numbers for polymer beads:
c
c                    LOCC(jx,jy,jz) = 0 if (jx,jy,jz) is unoccupied;
c                    LOCC(jx,jy,jz) = 1 if (jx,jy,jz) is occupied.
c
c     loccmid.....Lattice occupation numbers for bond midpoints:
c                                                                      130
c                    LOCCMID(jxm,jym,jzm) = 0 if no bond midpoint
c                    is present at position (jxm/2,jym/2,jzm/2).
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
c
c                    LOCCMID(jxm,jym,jzm) is a positive integer, M,
c                    if M bond midpoints are present at position
c                    (jxm/2,jym/2,jzm/2) on the lattice;
c
c
c     kconn.....Array containing the connectivity information:
c                                                                      140
c                    For polymer beads not at chain ends,
c                    KCONN(1,I) = I-1 and KCONN(2,I) = I+1,
c                    signifying that the bead I is bonded to
c                    its two neighbors,
c
c                    for the first bead of a polymer chain,
c                    KCONN(1,I) = 0 and KCONN(2,I) = I+1,
c                    because there is no bond to the "left",
c
c                    for the last bead of a polymer chain,             150
c                    KCONN(1,I) = I-1 and KCONN(2,I) = 0,
c                    because there is no bond to the "right".
c
c     cmpoly.....center-of-mass coordinates of all polymer beads
c     msd.....center-of-mass "autocorrelation" function
c     Rg.....radius of gyration
c     r.....records displacement of each bead
c     Cr.....time correlation for bead displacement
c     Ce.....time correlation for diameter vector
c     Ccmf.....time correlation for bead motion in center-of-mass frame  160
c     attmov.....array for attempted moves
c     accmov.....array for accepted moves
c     tRg.....accumulate Rg for averaging
c     tRe.....accumulate Re for averaging
c     moblty.....final mobility
c     nrm.....normalization constant
c     hist.....histogram array for center-of-mass pair-correlation function
c     grcm.....center-of-mass radial distribution function
c     dr.....bin width for pair distribution
c                                                                      170
c ----------

      integer          imc1, imc2, jpoly, iseed
      integer          jbead, i, j, lper, jn2, k
      integer          istream, iconfig, itrash, istart
      integer          kconn(2,nbead), lpoly(3,nbead)
      integer          locc(nx,ny,nz), loccmid(nxmid,nymid,nzmid)
      external         lper

      double precision cmpoly(ndata,3,npoly)                           180
      double precision msd(0:ndata), Cr(0:ndata), ran3
      double precision Rg(ndata,3), r(ndata,3,rbead), tRg, tRe
      double precision attmov(nbead), accmov(nbead), moblty(n)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      double precision Ce(0:ndata), Ccmf(0:ndata), nrm
      double precision grcm, const, pi, d, dr, hist(0:maxbin)

      character*(75) coordfile(maxstream)
      character*(10) msdfile(runmax), grfile(runmax)

c ------------------------------------------------------------------- 190


      msdfile(1) = "msd1"

      grfile(1) = "grcm1"

      coordfile(1) = "configs/stream.coord1"
      coordfile(2) = "configs/stream.coord2"
      coordfile(3) = "configs/stream.coord3"
      coordfile(4) = "configs/stream.coord4"                          200
      coordfile(5) = "configs/stream.coord5"
      coordfile(6) = "configs/stream.coord6"
      coordfile(7) = "configs/stream.coord7"
      coordfile(8) = "configs/stream.coord8"
      coordfile(9) = "configs/stream.coord9"
      coordfile(10) = "configs/stream.coord10"

      pi = acos(-1.d0)
      dr = 0.2d0
                                                                      210
      do i=1,nbead
        attmov(i) = 0.d0
        accmov(i) = 0.d0
      enddo

      do i=0,maxbin
        hist(i)=0.d0
      enddo

c read in run parameters                                              220

      open(unit=10,file="runtime.cfg",status="old")
      read(10,*)iseed
      read(10,*)istart
      read(10,*)istream
      read(10,*)iconfig
      close(10)

c create new output file
                                                                      230
      open(unit=10,file="rng.out",status="unknown")
      write(10,*)" "
      close(10)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      call start(nx,ny,nz,nxmid,nymid,nzmid,kconn,locc,loccmid,
     :     attmov,accmov,kdata,pi,dr,nconfig,iconfig,iseed,equil,
     :          nrelax,istream,nstream,istart,maxstream,coordfile)


C ====================                                              240
C START THE MAIN LOOP
C ====================

      do k=istart, runmax

        open(unit=10,file="rng.out",status="old",access="append")
        write(10,*)" "
        write(10,35)" Beginning run ", k, " of ", runmax, "..."
        write(10,*)" "
        close(10)                                                  250

        if (k .gt. istart) then
          iconfig = 0
          do i=0,maxbin
            hist(i)=0.d0
          enddo
        endif

        do imc1=iconfig+1, ndata
                                                                   260
c ==================
c  Save coordinates
c ==================

          istream = 1 + (imc1-1)/nconfig + (k-1)*nstream

          open(35,file=coordfile(istream),status="unknown",
     :                 access="append", form="unformatted")
          write(35)lpoly
          close(35)                                                270

c ==========================
c  Save configuration info
c ==========================

          itrash = -int(5.0d5*ran3(iseed))
          open(unit=10,file="runtime.cfg",status="unknown")
          write(10,10)itrash, " :  Random number seed"
          write(10,10)k, " : current trajectory"
          write(10,10)istream, " : existing coord files "//        280
     :                    "(starts at zero)"
          write(10,10)imc1, " : current number of configs "//
     :                  "(starts at zero)"
          close(10)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
c ==================
c  Accumulate data
c ==================

        call data(imcl,dr)                                          290

        if (mod((imcl-10),10) .eq. 0) then

c ====================
c  Calculate Re & Rg
c ====================

          tRg=0.d0
          tRe=0.d0
          nrm=0.d0                                                  300

          do i=1,imcl
            tRg=tRg+Rg(i,1)+Rg(i,2)+Rg(i,3)
          enddo

          do i=1,imcl
            do jpoly=1,rpoly
              do jbead=1+(jpoly-1)*n,jpoly*n-(n/2)-1

                jn2 = jbead+(n/2)                                   310

                tRe=tRe+(r(i,1,jbead)-r(i,1,jn2))**2+
     :                  (r(i,2,jbead)-r(i,2,jn2))**2+
     :                  (r(i,3,jbead)-r(i,3,jn2))**2

                nrm=nrm+1.d0

              enddo
            enddo
          enddo                                                     320

          tRg=tRg/dble(imcl)
          tRe=tRe/nrm

          call corr(imcl,msd,Cr,Ce,Ccmf)

c ========================
c  Save intermediate data
c ========================
                                                                    330
          open(unit=80,file=grfile(k),status="unknown")
          write(80,25)"#kd=",kdata,"nd=",ndata,"imcl=",imcl,
     :              "<Rg^2>=",tRg,"<Re^2>=",tRe

          write(80,5)"#"
          write(80,5)"#      r       g_cm(r)"
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      const = 4.0*pi*dble(npoly)/(3.0*dble(nx*ny*nz))
      i = 0
      d = dble(i)*dr                                                    340

      grcm=hist(i)/(.5**3*const*dr**3*dble(npoly)*
     :                                        dble(imc1))

      write(80,102)d,grcm

      do i=1,maxbin

        d = dble(i)*dr
                                                                        350
        grcm=hist(i)/(((dble(i)+.5)**3-(dble(i)-.5)**3)*
     :                    const*dr**3*dble(npoly)*dble(imc1))

        write(80,102)d,grcm

      enddo
      close(80)


      open(unit=30,file=msdfile(k),status="unknown")                    360
      write(30,25)"#kd=",kdata,"nd=",ndata,"imc1=",imc1,
     :            "<Rg^2>=",tRg,"<Re^2>=",tRe
      write(30,5)"#"
      write(30,5)"# tstep      Rcm2     NRcm2/6t",
     :           "    Cce         r2        rcmf2"

      write(30,101)0,0,0,Ce(0),0,0
      do i=1, imc1-1
        write(30,101)i*kdata,msd(i),
     :               msd(i)*N/(6.d0*dble(i*kdata)),                   370
     :               Ce(i),Cr(i),Ccmf(i)
      enddo
      close(30)

    endif

c ========================================
c  Advance the configuration kdata steps
c ========================================
                                                                        380
    do imc2=1, kdata
      call advance (N,npoly,nbead,nx,ny,nz,nxmid,nymid,nzmid,
     :              kconn,lpoly,locc,loccmid,iseed,attmov,accmov)
    enddo

  enddo
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
c ================================================
c  Run for kdata*ndata time steps without collecting data
c ================================================                    390

      if (k .lt. runmax) then

        do imc2=1, kdata*ndata

          call advance (N,npoly,nbead,nx,ny,nz,nxmid,nymid,nzmid,
     :                  kconn,lpoly,locc,loccmid,iseed,attmov,accmov)

          if (mod((imc2-100000),100000) .eq. 0) then
                                                                       400
            open(65,file="intermed.coord",status="unknown")

            write(65,30)"kd*nd=",kdata*ndata,"imc2=",imc2

            do i=1, nbead
              write(65,100)lpoly(1,i),lpoly(2,i),lpoly(3,i)
            enddo

            close(65)
                                                                       410
          endif

        enddo

      endif

    enddo


c ===============                                                      420
c  save mobility
c ===============

      do i=1,nbead
        accmov(i)=accmov(i)/attmov(i)
      enddo

      do jpoly=1,npoly
        do i=1,n
          jbead = i + N*(jpoly-1)                                      430
          moblty(i)=moblty(i)+accmov(jbead)
        enddo
      enddo

      do i=1,n
        moblty(i)=moblty(i)/dble(npoly)
      enddo
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      open(unit=70,file="final.data",status="unknown")
      write(70,*)"# Bead Mobility"                                          440
      do i=1,n
        write(70,101)i,moblty(i)
      enddo

      close(70)


    5 format (4(a))
   10 format (2x,i8,a)
   25 format (a4,i6,1x,a4,i5,1x,a6,i5,3x,a7,e10.4,2x,a7,e10.4)              450
   30 format (1x,a,i10,3x,a,i10)
   35 format (a,i2,a,i2,a)
  100 format (1x, i4, 1x, i4, 1x, i4)
  101 format (1x,i10,6(1x,e10.4))
  102 format (1x,f10.4,2x,e10.4,1x,e10.4)


      stop
      end
                                                                            460


c ********************* SUBROUTINE CORR ***************************

c ================================================
c  subroutine for time correlatation functions
c ================================================

      subroutine corr(imc1,msd,Cr,Ce,Ccmf)                                  470 corr

      implicit none

      common/block1/cmpoly,r,Rg,lpoly,hist

      integer       n,npoly,nbead,rpoly,rbead,ndata,maxbin
      parameter     (N      = 1000)
      parameter     (npoly  = 65)
      parameter     (nbead  = n*npoly)
                                                                            480
      parameter     (rpoly  = 1)
      parameter     (rbead  = n*rpoly)

      parameter     (ndata  = 2000)
      parameter     (maxbin = 125)

      integer        imc1,t,to,tto,jn2
      integer        jpoly,jbead,bin
      integer        lpoly(3,nbead)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
                                                                      490
      double precision cmpoly(ndata,3,npoly),msd(0:ndata),cmo(3)
      double precision norm(0:ndata),r(ndata,3,rbead)
      double precision Cr(0:ndata),Cro(3)
      double precision Ce(0:ndata),Ceo(3)
      double precision Ccmf(0:ndata),Ccmfo(3)
      double precision hist(0:maxbin),Rg(ndata,3)

c ------------------------------

                                                                      500
c ================================================
c  Polymer center-of-mass displacement correlation
c ================================================

      do t=0,imcl-1
        msd(t) = 0.d0
        norm(t) = 0.d0
      enddo

      do jpoly=1,npoly                                                510

        do to=1,imcl

          cmo(1) = cmpoly(to,1,jpoly)
          cmo(2) = cmpoly(to,2,jpoly)
          cmo(3) = cmpoly(to,3,jpoly)

          do tto=to,imcl

            t = tto - to                                              520
            msd(t) = msd(t) + (cmo(1)-cmpoly(tto,1,jpoly))**2 +
     :                       (cmo(2)-cmpoly(tto,2,jpoly))**2 +
     :                       (cmo(3)-cmpoly(tto,3,jpoly))**2

            norm(t) = norm(t) + 1.d0

          enddo

        enddo
                                                                      530
      enddo

      do t=0,imcl-1
        msd(t) = msd(t)/norm(t)
      enddo

c ================================================
c  Bead displacement correlation in normal and center-of-mass frame
c ================================================
                                                                      540
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      do t=0,imcl-1
        Cr(t)=0.d0
        Ccmf(t)=0.d0
        norm(t)=0.d0
      enddo

      do jpoly=1,rpoly

        do jbead=1+(jpoly-1)*n,jpoly*n
                                                                        550
          do to=1,imcl

            Cro(1) = r(to,1,jbead)
            Cro(2) = r(to,2,jbead)
            Cro(3) = r(to,3,jbead)
            Ccmfo(1) = cmpoly(to,1,jpoly)-r(to,1,jbead)
            Ccmfo(2) = cmpoly(to,2,jpoly)-r(to,2,jbead)
            Ccmfo(3) = cmpoly(to,3,jpoly)-r(to,3,jbead)

            do tto=to,imcl                                              560

              t = tto - to

              Cr(t) = Cr(t) + (Cro(1)-r(tto,1,jbead))**2 +
                              (Cro(2)-r(tto,2,jbead))**2 +
                              (Cro(3)-r(tto,3,jbead))**2

              Ccmf(t) = Ccmf(t) + (Ccmfo(1)+r(tto,1,jbead)-
                                   cmpoly(tto,1,jpoly))**2 +
                                  (Ccmfo(2)+r(tto,2,jbead)-             570
                                   cmpoly(tto,2,jpoly))**2 +
                                  (Ccmfo(3)+r(tto,3,jbead)-
                                   cmpoly(tto,3,jpoly))**2

              norm(t) = norm(t) + 1.d0

            enddo

          enddo
                                                                        580
        enddo

      enddo

      do t=0,imcl-1
        Cr(t) = Cr(t)/norm(t)
        Ccmf(t) = Ccmf(t)/(norm(t)*2.d0)
      enddo

c ==============================                                      590
c  Diameter vector correlation
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
c ============================

      do t=0,imcl-1
        Ce(t)=0.d0
        norm(t)=0.d0
      enddo

      do jpoly=1,rpoly                                                  600

        do jbead=1+(jpoly-1)*n,jpoly*n-(n/2)-1

          jn2 = jbead+(n/2)

          do to=1,imcl

            Ceo(1) = r(to,1,jbead)-r(to,1,jn2)
            Ceo(2) = r(to,2,jbead)-r(to,2,jn2)
            Ceo(3) = r(to,3,jbead)-r(to,3,jn2)
                                                                        610
            do tto=to,imcl

              t = tto - to
              Ce(t) = Ce(t) + Ceo(1)*(r(tto,1,jbead)-
     :                                 r(tto,1,jn2))+
     :                        Ceo(2)*(r(tto,2,jbead)-
     :                                 r(tto,2,jn2))+
     :                        Ceo(3)*(r(tto,3,jbead)-
     :                                 r(tto,3,jn2))
                                                                        620
              norm(t) = norm(t) + 1.d0

            enddo

          enddo

        enddo

      enddo
                                                                        630
      do t=0,imcl-1
        Ce(t) = Ce(t)/norm(t)
      enddo


      return
      end


                                                                        640
c ******************** SUBROUTINE DATA ***************************
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      subroutine data(i,dr)                                              data

      implicit none

      common/block1/cmpoly,r,Rg,lpoly,hist

      integer       n,npoly,nbead,rpoly,rbead,ndata,maxbin
      integer       nx, ny, nz                                           650
      parameter     (N      = 1000)
      parameter     (npoly  = 65)
      parameter     (nbead  = n*npoly)

      parameter     (rpoly  = 1)
      parameter     (rbead  = n*rpoly)

      parameter     (ndata  = 2000)
      parameter     (maxbin = 125)
                                                                         660
      parameter     (nx = 50)
      parameter     (ny = 50)
      parameter     (nz = 52)

      integer       jx, jy, jz, imcl
      integer       jbead, j, jpoly, i, k, bin
      integer       lpoly(3,nbead)

      double precision cmpoly(ndata,3,npoly)
      double precision Rg(ndata,3), r(ndata,3,rbead)                     670
      double precision rxij, ryij, rzij
      double precision rxxij, ryyij, rzzij
      double precision rij, rijsq, d, dr, hist(0:maxbin)

c ------------------------------.


      Rg(i,1)=0.d0
      Rg(i,2)=0.d0
      Rg(i,3)=0.d0                                                       680

      do jpoly=1, npoly

        cmpoly(i,1,jpoly)=0.d0
        cmpoly(i,2,jpoly)=0.d0
        cmpoly(i,3,jpoly)=0.d0

        do jx=1, N

          jbead = jx + N*(jpoly-1)                                       690
          cmpoly(i,1,jpoly)=cmpoly(i,1,jpoly)+
     :            dble(lpoly(1,jbead))
          cmpoly(i,2,jpoly)=cmpoly(i,2,jpoly)+
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
     :              dble(lpoly(2,jbead))
            cmpoly(i,3,jpoly)=cmpoly(i,3,jpoly)+
     :              dble(lpoly(3,jbead))

          enddo

          cmpoly(i,1,jpoly)=cmpoly(i,1,jpoly)/dble(N)                    700
          cmpoly(i,2,jpoly)=cmpoly(i,2,jpoly)/dble(N)
          cmpoly(i,3,jpoly)=cmpoly(i,3,jpoly)/dble(N)

c =====================
c  Radius of gyration
c =====================

          do jx=1,n

            jbead = jx + N*(jpoly-1)                                     710

            Rg(i,1)=Rg(i,1)+(dble(lpoly(1,jbead))-
     :                       cmpoly(i,1,jpoly))**2
            Rg(i,2)=Rg(i,2)+(dble(lpoly(2,jbead))-
     :                       cmpoly(i,2,jpoly))**2
            Rg(i,3)=Rg(i,3)+(dble(lpoly(3,jbead))-
     :                       cmpoly(i,3,jpoly))**2

          enddo
                                                                         720
        enddo

        Rg(i,1)=Rg(i,1)/(dble(n)*dble(npoly))
        Rg(i,2)=Rg(i,2)/(dble(n)*dble(npoly))
        Rg(i,3)=Rg(i,3)/(dble(n)*dble(npoly))

        do jbead=1, rbead

          r(i,1,jbead)=dble(lpoly(1,jbead))
          r(i,2,jbead)=dble(lpoly(2,jbead))                              730
          r(i,3,jbead)=dble(lpoly(3,jbead))

        enddo

c ================================================
c  Radial distribution function of center-of-mass
c ================================================

        do j=1, npoly-1
          do k=j+1, npoly                                                740

            rxij=cmpoly(i,1,j)-cmpoly(i,1,k)
            ryij=cmpoly(i,2,j)-cmpoly(i,2,k)
            rzij=cmpoly(i,3,j)-cmpoly(i,3,k)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      rxxij = rxij-anint(rxij/dble(nx))*dble(nx)
      ryyij = ryij-anint(ryij/dble(ny))*dble(ny)
      rzzij = rzij-anint(rzij/dble(nz))*dble(nz)

      rijsq = rxxij**2 + ryyij**2 + rzzij**2                          750
      rij = sqrt(rijsq)

      bin = anint(rij/dr)

      if (bin .le .maxbin) then
        hist(bin)=hist(bin) + 2.d0
      endif

    enddo
  enddo                                                               760


  return
  end


c ******************** SUBROUTINE START ***************************

      subroutine start(nx,ny,nz,nxmid,nymid,nzmid,kconn,locc,loccmid,  770
     :          attmov,accmov,kdata,pi,dr,nconfig,iconfig,iseed,equil,
     :               nrelax,istream,nstream,istart,maxstream,coordfile)

      implicit none

      common/block1/cmpoly,r,Rg,lpoly,hist

      integer        n,npoly,nbead,rpoly,rbead,ndata,maxbin
      parameter      (N      = 1000)
      parameter      (npoly  = 65)                                    780
      parameter      (nbead  = n*npoly)

      parameter      (rpoly  = 1)
      parameter      (rbead  = n*rpoly)

      parameter      (ndata  = 2000)
      parameter      (maxbin = 125)

      integer        nx, ny, nz, nxmid, nymid, iconfig, jconfig
      integer        nzmid, nrelax, kdata, jx, jy, jz, istart         790
      integer        iseed, jxxx, jyyy, jzzz, jcon2, itrash
      integer        jbead, j, jpoly, jxmid, jymid, jzmid, imc1
      integer        lper, iu, i, k, nconfig, bin, istream
      integer        kconn(2,nbead), lpoly(3,nbead), nstream
      integer        locc(nx,ny,nz), loccmid(nxmid,nymid,nzmid)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      integer         maxstream
      external        lper

      logical         equil
                                                                        800
      double precision attmov(nbead),accmov(nbead)
      double precision cmpoly(ndata,3,npoly)
      double precision Rg(ndata,3), r(ndata,3,rbead)
      double precision rxij, ryij, rzij
      double precision rxxij, ryyij, rzzij, const
      double precision rij, rijsq, pi, d, dr, hist(0:maxbin)

      character*(75) coordfile(maxstream)

c ------------------------------                                       810


      if(istream .eq. 0) then

        if(equil .eqv. .false.) then

          open (10, file = "init.coord", status = "old")

          read (10, *) itrash
          if(itrash .ne. N) then                                        820
            write(*, *)"Number of beads does not match input"
            stop
          endif

          read (10, *) itrash
          if (itrash .ne. npoly) then
            write (*, *) 'Number of polymers does not match input'
            stop
          endif
                                                                        830
          do jbead = 1, nbead
            read(10,*)lpoly(1,jbead),lpoly(2,jbead),lpoly(3,jbead)
          enddo

          close (10)

        else

          open (unit=20,file='start.coord',status='old')
                                                                        840
          read (20, *) itrash
          if (itrash .ne. N) then
            write (*, *) 'Number of beads does not match input'
            stop
          endif
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
        read (20, *) itrash
        if (itrash .ne. npoly) then
          write (*, *) 'Number of polymers does not match input'
          stop                                                          850
        endif

        do jbead = 1, nbead
          read(20,*)lpoly(1,jbead),lpoly(2,jbead),lpoly(3,jbead)
        enddo

        close (20)

      endif
                                                                        860
c ==============
c RUN PARAMETERS
c ==============

      iu = 9
      open (unit=iu,file='vital.stats',status='unknown',
     :                    form='formatted')
      write (iu, *) ' '
      write (iu,55) '            Chain length: ', N
      write (iu,55) '       Number of chains: ', npoly                 870
      write (iu, *) ' '
      write (iu,60) '     Simulation box size: ', nx, ny, nz
      write (iu, *) ' '
      write (iu,50) '         Average density: ',
     :                dble(nbead)/dble(nx*ny*nz)
      write (iu, *) ' '
      write (iu,55) ' MCS between data recording: ', kdata
      write (iu,55) '   Number of data records: ', ndata
      write (iu,55) '      Total number of MCS: ', kdata*ndata
      write (iu,55) '       Equilibration MCS: ', nrelax              880
      write (iu, *) ' '
      close (unit = iu)

    else

c ================================
c  We are restarting from a crash
c ================================

      open(unit=10,file="rng.out",status="old",access="append")         890
      write(10,*)" "
      write(10,*)" Restarting..."
      write(10,*)" "
      close(10)

c ========================
c  Open stream.coord files
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
c ==========================

      do k=1+(istart-1)*nstream, istream                                  900

        open(unit=10,file="rng.out",status="old",access="append")
        write(10,20)" Accessing file: ", coordfile(k)
        close(10)

        open(15,file=coordfile(k),status="old",
     :                        form="unformatted")

        do i=1, nconfig
                                                                          910
          jconfig = i + (k-1-(istart-1)*nstream)*nconfig

          if(jconfig .le. iconfig) then
            read(15)lpoly

            call data(jconfig,dr)
          endif

        enddo
                                                                          920
        close(15)

      enddo

      open(unit=10,file="rng.out",status="old",access="append")
      write(10,*)" "
      write(10,*)" Done reading files."
      write(10,*)" "
      close(10)
                                                                          930
    endif

c =============
c Connectivity
c =============

    call connect(npoly,n,kconn,nbead)

c ================================
c  Initialize occupation numbers                                          940
c ================================

    call occunum(nbead,nx,ny,nz,nxmid,nymid,nzmid,locc,loccmid,
   :             kconn,lpoly)


C Check again whether restarting, if true then advance kdata steps
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      if(istream .ne. 0) then
                                                                      950
        open(unit=10,file="rng.out",status="old",access="append")
        write(10,*)" "
        write(10,*)" Advancing configuration kdata steps..."
        write(10,*)" "
        close(10)

        do imc1=1, kdata
          call advance (N,npoly,nbead,nx,ny,nz,nxmid,nymid,nzmid,
     :                  kconn,lpoly,locc,loccmid,iseed,attmov,accmov)
        enddo                                                         960

        open(unit=10,file="rng.out",status="old",access="append")
        write(10,*)" "
        write(10,*)" Done restarting."
        write(10,*)" "
        close(10)

      endif

                                                                      970
C Check again whether the system is equilibrated

      if( equil .eqv. .false. ) then

        if( istream .ne. 0 ) goto 1000

c ====================
c EQUILIBRATION PERIOD
c ====================
                                                                      980
        open(unit=10,file="rng.out",status="old",access="append")
        write(10,*)" "
        write(10,*)" Equilibrating..."
        write(10,*)" "
        close(10)

        do imc1 = 1, nrelax
          call advance (N,npoly,nbead,nx,ny,nz,nxmid,nymid,nzmid,
     :                  kconn,lpoly,locc,loccmid,iseed,attmov,accmov)
                                                                      990
          if(mod((imc1-100000),100000) .eq. 0) then
            open(68,file="run.coord",status="unknown")
            write(68,30)"nrelax=",nrelax,"imc1=",imc1
            do i=1, nbead
              write(68,100)lpoly(1,i),lpoly(2,i),lpoly(3,i)
            enddo
            close(68)
          endif
        enddo
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
                                                                        1000
c     Write equilibrated coordinates...

      open (unit=77, file='equil.coord', status='unknown')

      write (77, 90) N,    ' : Chain length '
      write (77, 90) npoly, ' : Number of chains'

      do jbead = 1, nbead
        write(77,100)lpoly(1,jbead),lpoly(2,jbead),lpoly(3,jbead)
      enddo                                                             1010
      close (unit=77)

c ----------
c Check for bond crossing
c ----------

      do jbead=1, nbead
        jcon2 = kconn(2,jbead)

        jxmid = lpoly(1,jbead) + lpoly(1,jcon2)                         1020
        jymid = lpoly(2,jbead) + lpoly(2,jcon2)
        jzmid = lpoly(3,jbead) + lpoly(3,jcon2)

        jxxx = lper(jxmid,nxmid)
        jyyy = lper(jymid,nymid)
        jzzz = lper(jzmid,nzmid)

        if (loccmid(jxxx,jyyy,jzzz) .ne. 1) then
          write (*, *) 'ERROR: Bond crossing detected: ',
     :                 jxxx,jyyy,jzzz, loccmid(jxxx,jyyy,jzzz)          1030
          stop
        endif
      enddo

      open(unit=10,file="rng.out",status="old",access="append")
      write(10,*) ' '
      write(10,*) ' No bond crossings remain...'
      write(10,*) ' '
      close(10)
                                                                        1040
1000  endif


   20 format (a,a)
   30 format (1x,a,i10,3x,a,i10)
   50 format (a, f8.4)
   55 format (a, i10)
   60 format (a, i4, 1x, i4, 1x, i4)
   90 format (1x, i6, a)
  100 format (1x, i4, 1x, i4, 1x, i4)                                   1050
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      return
      end


c ********** SUBROUTINE ADVANCE **********

c ==========================================                          1060
c ADVANCE: This subroutine advances the dynamic Monte Carlo time
c evolution of a lattice polymer system by one MC step. A total of NBEAD
c time steps are attempted. The polymer chains are modeled by the
c ONE-SITE bond fluctuation model.
c Periodic boundary conditions are applied in all coordinate directions.
c ==========================================

      subroutine advance ( N, npoly, nbead, nx, ny, nz, nxmid, nymid,
     :     nzmid, kconn, lpoly, locc, loccmid, iseed, attmov, accmov )
                                                                    1070
c INPUT
c     nbead.....Total number of polymer beads
c     nx,ny,nz.....Size of the lattice box for polymer beads
c     nxmid,
c     nymid,
c     nzmid.....Size of the lattice box for bond midpoints
c     lpoly.....Initial lattice coordinates of all polymer beads
c     locc.....Lattice occupation numbers for polymer beads:
c
c                    LOCC(jx,jy,jz) = 0 if (jx,jy,jz) is unoccupied;    1080
c                    LOCC(jx,jy,jz) = 1 if (jx,jy,jz) is occupied.
c
c     loccmid.....Lattice occupation numbers for bond midpoints:
c
c                    LOCCMID(jxm,jym,jzm) = 0 if no bond midpoint
c                    is present at position (jxm/2,jym/2,jzm/2).
c
c                    LOCCMID(jxm,jym,jzm) is a positive integer, M,
c                    if M bond midpoints are present at position
c                    (jxm/2,jym/2,jzm/2) on the lattice;               1090
c
c     kconn.....Array containing the connectivity information:
c
c                    For polymer beads not at chain ends,
c                    KCONN(1,I) = I-1 and KCONN(2,I) = I+1,
c                    signifying that the bead I is bonded to
c                    its two neighbors,
c
c                    for the first bead of a polymer chain,
c                    KCONN(1,I) = 0 and KCONN(2,I) = I+1,              1100
c                    because there is no bond to the "left",
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
c
c                    for the last bead of a polymer chain,
c                    KCONN(1,I) = I-1 and KCONN(2,I) = 0,
c                    because there is no bond to the "right".
c
c
c     iseed.....Initial random number seed
c
c OUTPUT                                                          1110
c     lpoly.....Updated lattice coordinates of all beads
c     locc.....Updated occupation numbers for beads
c     loccmid.....Updated occupation numbers for bond midpoints
c     iseed.....New random number seed
C     attmov.....array storing attempted moves for each bead
C     accmov.....array for storing accepted moves
c
c ----------

      implicit none                                               1120

c - ARGUMENTS ----
      integer         N, npoly
      integer         nbead, nx, ny, nz, nxmid, nymid, nzmid, iseed
      integer         kconn(2,nbead)
      integer         lpoly(3,nbead)
      integer         locc(nx,ny,nz)
      integer         loccmid(nxmid,nymid,nzmid)

c - LOCAL -----                                                   1130
      integer         iatt, jbead, jmove, jcon1, jcon2, jx, jy, jz,
     :                jxmid1, jymid1, jzmid1, jxmid2, jymid2, jzmid2,
     :                jxxx, jyyy, jzzz, LSQR, jpoly

      integer         lper
      double precision ran3, rannum
      double precision attmov(nbead), accmov(nbead)
      external        lper, ran3

c ------------------------------                                  1140

c ====================
c  --------
c    MOVE ATTEMPTS
c  --------
c ====================

      do iatt = 1, nbead

c ================                                                1150
c Select the event
c ================
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      jbead = 1 + int(dble(nbead)*ran3(iseed))
      attmov(jbead)=attmov(jbead) + 1.d0

      jmove = 1 + int(6.D0*ran3(iseed))

c ====================
c Generate trial move                                          1160
c ====================

      jx = lpoly(1,jbead)
      jy = lpoly(2,jbead)
      jz = lpoly(3,jbead)

      if (jmove .eq. 1) then
        jx = jx + 1
      else if (jmove .eq. 2) then
        jx = jx - 1                                            1170
      else if (jmove .eq. 3) then
        jy = jy + 1
      else if (jmove .eq. 4) then
        jy = jy - 1
      else if (jmove .eq. 5) then
        jz = jz + 1
      else
        jz = jz - 1
      endif
                                                               1180

      jcon1 = kconn(1,jbead)
      jcon2 = kconn(2,jbead)

      jxmid1 = jx + lpoly(1,jcon1)
      jymid1 = jy + lpoly(2,jcon1)
      jzmid1 = jz + lpoly(3,jcon1)

      jxmid2 = jx + lpoly(1,jcon2)
      jymid2 = jy + lpoly(2,jcon2)                             1190
      jzmid2 = jz + lpoly(3,jcon2)


c ================
c Test trial move
c ================

c As soon as we find a violation of a necessary condition for the
c acceptance of the move, we jump to the end of the loop over trials...
                                                               1200
c ---------------
c Excluded volume constraints
c ---------------
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      jxxx = lper(jx,nx)
      jyyy = lper(jy,ny)
      jzzz = lper(jz,nz)

      if (locc(jxxx,jyyy,jzzz) .ne. 0) goto 1000
c ----------                                                    1210
c Allowed bond vectors
c ----------

c We have already checked for bond lengths of zero
c by enforcing the excluded volume condition...

      LSQR = (jx - lpoly(1,jcon1))**2
     :     + (jy - lpoly(2,jcon1))**2
     :     + (jz - lpoly(3,jcon1))**2
                                                                1220
      if (LSQR .gt. 3) goto 1000

      LSQR = (jx - lpoly(1,jcon2))**2
     :     + (jy - lpoly(2,jcon2))**2
     :     + (jz - lpoly(3,jcon2))**2

      if (LSQR .gt. 3) goto 1000

c -------
c Bond crossing                                                 1230
c -------

      jxxx = lper(jxmid1,nxmid)
      jyyy = lper(jymid1,nymid)
      jzzz = lper(jzmid1,nzmid)

      if (loccmid(jxxx,jyyy,jzzz) .ne. 0) goto 1000

      jxxx = lper(jxmid2,nxmid)
      jyyy = lper(jymid2,nymid)                                 1240
      jzzz = lper(jzmid2,nzmid)

      if (loccmid(jxxx,jyyy,jzzz) .ne. 0) goto 1000


c =============
c UPDATE SYSTEM
c =============

c If we get to this point, then the move has been accepted...   1250

      accmov(jbead)=accmov(jbead)+1.d0

c Update occupation numbers:
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      jxxx = lper(lpoly(1,jbead),nx)
      jyyy = lper(lpoly(2,jbead),ny)
      jzzz = lper(lpoly(3,jbead),nz)
      locc(jxxx,jyyy,jzzz) = 0

      jxxx = lper(jx,nx)
      jyyy = lper(jy,ny)
      jzzz = lper(jz,nz)
      locc(jxxx,jyyy,jzzz) = 1


c *Update bond midpoints:*

      jxxx = lper(lpoly(1,jbead) + lpoly(1,jcon1), nxmid)
      jyyy = lper(lpoly(2,jbead) + lpoly(2,jcon1), nymid)
      jzzz = lper(lpoly(3,jbead) + lpoly(3,jcon1), nzmid)
      loccmid(jxxx,jyyy,jzzz) = loccmid(jxxx,jyyy,jzzz) - 1

      jxxx = lper(jxmid1,nxmid)
      jyyy = lper(jymid1,nymid)
      jzzz = lper(jzmid1,nzmid)
      loccmid(jxxx,jyyy,jzzz) = loccmid(jxxx,jyyy,jzzz) + 1

      jxxx = lper(lpoly(1,jbead) + lpoly(1,jcon2), nxmid)
      jyyy = lper(lpoly(2,jbead) + lpoly(2,jcon2), nymid)
      jzzz = lper(lpoly(3,jbead) + lpoly(3,jcon2), nzmid)
      loccmid(jxxx,jyyy,jzzz) = loccmid(jxxx,jyyy,jzzz) - 1

      jxxx = lper(jxmid2,nxmid)
      jyyy = lper(jymid2,nymid)
      jzzz = lper(jzmid2,nzmid)
      loccmid(jxxx,jyyy,jzzz) = loccmid(jxxx,jyyy,jzzz) - 1


c *Update coordinates:*

      lpoly(1,jbead) = jx
      lpoly(2,jbead) = jy
      lpoly(3,jbead) = jz

 1000 enddo

      return
      end


c ********** SUBROUTINE OCCUNUM **********
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
c ================================
c  Initializes occupation numbers
c ================================

      subroutine occunum(nbead,nx,ny,nz,nxmid,nymid,nzmid,locc,loccmid,
     :                   kconn,lpoly)

      implicit none

      integer  nx,ny,nz,nxmid,nymid,nzmid,jx,jy,jz,jxmid
      integer  jymid,jzmid,jxxx,jyyy,jzzz,jcon2,nbead
      integer  jbead,lper

      integer  locc(nx,ny,nz),loccmid(nxmid,nymid,nzmid)
      integer  kconn(2,nbead),lpoly(3,nbead)

      external  lper

c ---------------------------------

        open(unit=10,file="rng.out",status="old",access="append")
        write(10,*)" "
        write(10,*)" Initializing occupation numbers..."
        write(10,*)" "
        close(10)

        do jz = 1, nz
          do jy = 1, ny
            do jx = 1, nx
              locc(jx,jy,jz) = 0
            enddo
          enddo
        enddo

        do jzmid = 1, nzmid
          do jymid = 1, nymid
            do jxmid = 1, nxmid
              loccmid(jxmid,jymid,jzmid) = 0
            enddo
          enddo
        enddo

        do jbead = 1, nbead
          jx = lpoly(1,jbead)
          jy = lpoly(2,jbead)
          jz = lpoly(3,jbead)

          jxxx = lper(jx,nx)
          jyyy = lper(jy,ny)
          jzzz = lper(jz,nz)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      if(locc(jxxx,jyyy,jzzz) .ne. 0)then
        write (*,*)'ERROR: Bead overlap detected'
        stop
      endif                                                          1360

      locc(jxxx,jyyy,jzzz) = 1

      jcon2 = kconn(2,jbead)

      if (jcon2 .ne. 0) then
        jxmid = lpoly(1,jbead) + lpoly(1,jcon2)
        jymid = lpoly(2,jbead) + lpoly(2,jcon2)
        jzmid = lpoly(3,jbead) + lpoly(3,jcon2)
                                                                     1370
        jxxx = lper(jxmid,nxmid)
        jyyy = lper(jymid,nymid)
        jzzz = lper(jzmid,nzmid)

        if (loccmid(jxxx,jyyy,jzzz) .ne. 0) then
          write (*, *) 'WARNING: Bond crossing detected: ',
     :                 jxxx,jyyy,jzzz, jbead
        endif

c NOTE: There may be more than one bond midpoint at a given location...   1380

        loccmid(jxxx,jyyy,jzzz) = loccmid(jxxx,jyyy,jzzz) + 1

      endif
    enddo

    return
    end

                                                                     1390


c ******************** SUBROUTINE CONNECT ***************************

c =====================
c  Connects the beads
c =====================

      subroutine connect(npoly,n,kconn,nbead)                        connect
                                                                     1400
      implicit none

      integer npoly,n,nbead,jbead,jpoly,J

      integer kconn(2,nbead)

c ---------------------------------
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      open(unit=10,file="rng.out",status="old",access="append")
      write(10,*)" "                                                    1410
      write(10,*)" Setting up chain connectivity..."
      write(10,*)" "
      close(10)

      do jpoly = 1, npoly
        J = 1
        jbead = J + N*(jpoly-1)

        kconn(1,jbead) = jbead+n-1
        kconn(2,jbead) = jbead+1                                        1420

        do J = 2, (N-1)
          jbead = J + N*(jpoly-1)

          kconn(1,jbead) = jbead-1
          kconn(2,jbead) = jbead+1
        enddo

        J = N
        jbead = J + N*(jpoly-1)                                         1430

        kconn(1,jbead) = jbead-1
        kconn(2,jbead) = jbead-n+1
      enddo

      return
      end


                                                                        1440

c ******************** FUNCTION LPER ***************************

c ==========================================
c LPER: Given an uncorrected lattice coordinate LCOORD, and the size
c of the lattice (in the corresponding coordinate direction) N, this
c function returns the corrected lattice coordinates by applying
c periodic boundary conditions...
c Here we assume that the primary lattice box runs from 1 through N.
c ==========================================                            1450

      integer function lper ( lcoord, n )                               lper

      implicit none

      integer         lcoord, n

c ---------------------------
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      if (lcoord .lt. 1) then                                          1460
         lper = n + mod(lcoord,n)
      else
         lper = 1 + mod(lcoord-1,n)
      endif

      return
      end


                                                                       1470
c ******************** FUNCTION RAN3 ***************************

c =========================================
c RAN3: Returns a uniform random deviate on [0,1). Set IDUM to
c any negative value to initialize or reinitialize the sequence.
c From "Numerical Recipes", Chapter 7.
c =========================================

      double precision function ran3 (idum)                           ran3
                                                                       1480
      implicit none

      integer        idum

      double precision mbig, mseed, mz, fac
      parameter (mbig = 4000000.D0)
      parameter (mseed = 1618033.D0)
      parameter (mz = 0.D0)
      parameter (fac = 2.5D-07)
                                                                       1490
      integer        i, ii, iff, inext, inextp, k
      double precision mj, mk
      double precision ma(55)

      save           iff, inext, inextp
      save           ma

      data iff /0/

c ---------------------------------                                    1500

      if(idum.lt.0.or.iff.eq.0)then
         iff=1
         mj=mseed-iabs(idum)
         mj=mod(mj,mbig)
         ma(55)=mj
         mk=1
         do 11 i=1,54
            ii=mod(21*i,55)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
        ma(ii)=mk                                                    1510
        mk=mj-mk
        if(mk.lt.mz)mk=mk+mbig
        mj=ma(ii)
11    continue
      do 13 k=1,4
        do 12 i=1,55
          ma(i)=ma(i)-ma(1+mod(i+30,55))
          if(ma(i).lt.mz)ma(i)=ma(i)+mbig
12      continue
13    continue                                                       1520
      inext=0
      inextp=31
      idum=1
    endif

    inext = inext + 1
    if (inext .eq. 56) inext = 1
    inextp = inextp + 1
    if (inextp .eq. 56) inextp = 1
                                                                   1530
    mj = ma(inext) - ma(inextp)
    if (mj .lt. mz) mj = mj + mbig
    ma(inext) = mj

    ran3 = mj*fac

    return
    end

                                                                   1540
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Appendix B

Star source code

```
      program sncrv6

c Stochastic lattice dynamics simulation of a system of tri-functional,
c monodisperse polymer stars with excluded volume in a periodic box.

c In this implementation, the polymer chains are modeled with the
c ONE-SITE bond fluctuation model. The stochastic dynamics is a simple
c dynamic Monte Carlo process.

c In our implementation of the ONE-SITE bond fluctuation model, each
c polymer bead occupies one site on the lattice. The bonds between
c connected beads are restricted to the set (1,0,0), (1,1,0), and
c (1,1,1). Excluded volume conditions are enforced for all polymer
c beads by restricting the occupancy of the lattice sites to at most one
c polymer bead. An array of occupation numbers for the primary lattice,
c on which the beads reside, is maintained to check for bead overlaps.

c To forbid the crossing of polymer chains during the dynamics, we also
c track the MIDPOINTS of the polymer bonds. These coordinates of bond
c midpoints are always integer multiples of 1/2, so we maintain an
c additional lattice of occupation numbers for bond midpoints. This
c lattice has twice the number of entries in each coordinate direction
c as the lattice of occupation numbers for polymer beads. Any bond
c crossing must involve the simultaneous overlap of the coordinates of
c two bond midpoints, so this bond crossing condition is easy to detect.

c Note that the occupation numbers for the beads must be either zero or
c one because the excluded volume condition is always enforced. In
c contrast, the occupation numbers for the bond midpoints may be greater
c than one, because several bond midpoints can overlap when the chains
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
c are allowed to cross.

c The connectivity information is provided by the array KCONN, described
c below, so that polydisperse systems can be simulated with relative
c ease, if we choose do so later. Because KCONN is a one dimensional
c array and one bead, the star center bead, is tri-functional we declare
c a parameter that labels the initial bead of the third arm (i.e. the
c bead that represents the connectivity of the third bond on center
c bead).

c The program can use previously equilibrated configuration
c if the flag equil is set to TRUE

c PARAMETERS
c ------
c
c     N.....Number of beads per polymer chain
c     npoly.....Number of polymer chains
c     nbead.....Total number of polymer beads
c     nx,ny,nz.....Size of the lattice box for polymer segments
c     nxmid,
c     nymid,
c     nzmid.....Size of the lattice box for bond midpoints
c
c     nrelax.....Estimate of the relaxation time
c     kdata.....Number of MCS between accumulation of data
c     ndata.....Number of data points that will be calculated.
c
c        (The total number of MCS will be KDATA*NDATA.)
c
c     equil.....Logical flag: TRUE if equilibriated configuration
c                       is used; FALSE otherwise
c
c     rpoly.....number of chains whose bead displacements will be tracked
c     rbead.....total number of beads in the rpoly polymer chains
c     maxbin.....max number of bins in radial distribution function
c     abead.....bead of third arm that is connected to the center bead
c     lbead.....this is the center bead of the star
c
c ------

      implicit none

      common/block1/cmpoly,r,Rg,lpoly,hist

      integer       N, npoly, nbead, nx,ny,nz, nxmid,nymid,nzmid
      integer       nrelax, kdata, ndata, rpoly, rbead

c note these parameters need to be set in subroutines core, data, and start

      parameter     (N     = 450)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      parameter    (npoly = 64)
      parameter    (rpoly = 1)

      parameter    (nbead = N*npoly)
      parameter    (rbead = N*rpoly)

      parameter    (nx = 40)
      parameter    (ny = 40)
      parameter    (nz = 36)                                          90

      parameter    (nxmid = 2*nx)
      parameter    (nymid = 2*ny)
      parameter    (nzmid = 2*nz)

      parameter    (kdata = 200000)
      parameter    (ndata = 2000)
      parameter    (nrelax = 14800000)

      logical      equil                                              100
      parameter    (equil = .true.)

c nstream is the number of desired stream.coord files. nconfig
c evenly divides up the total number of configurations into the
c nstream stream.coord files
c the names of the stream.coord files are specified in the array
c "coordfile"

      integer      nstream, nconfig
      parameter    (nstream = 10)                                     110
      parameter    (nconfig = ndata/nstream)

      integer      maxbin
      parameter    (maxbin = 100)

      integer      irunmax, istart
      parameter    (istart = 1, irunmax = 1)

      integer      lbead, abead
      parameter    (lbead = 150, abead = 301)                         120

c VARIABLES
c ----
c
c     lpoly.....Lattice coordinates of all polymer beads
c     locc.....Lattice occupation numbers for polymer beads:
c
c                    LOCC(jx,jy,jz) = 0 if (jx,jy,jz) is unoccupied;
c                    LOCC(jx,jy,jz) = 1 if (jx,jy,jz) is occupied.
c                                                                     130
c     loccmid.....Lattice occupation numbers for bond midpoints:
c
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
c                    LOCCMID(jxm,jym,jzm) = 0 if no bond midpoint
c                    is present at position (jxm/2,jym/2,jzm/2).
c
c                    LOCCMID(jxm,jym,jzm) is a positive integer, M,
c                    if M bond midpoints are present at position
c                    (jxm/2,jym/2,jzm/2) on the lattice;
c
c                                                                    140
c   kconn.....Array containing the connectivity information:
c
c                    For polymer beads not at chain ends,
c                    KCONN(1,I) = I-1 and KCONN(2,I) = I+1,
c                    signifying that the bead I is bonded to
c                    its two neighbors,
c
c                    for the first bead of a polymer chain,
c                    KCONN(1,I) = 0 and KCONN(2,I) = I+1,
c                    because there is no bond to the "left",             150
c
c                    for the last bead of a polymer chain,
c                    KCONN(1,I) = I-1 and KCONN(2,I) = 0,
c                    because there is no bond to the "right".
c
c   cmpoly.....center-of-mass coordinates of all polymer beads
c   msd.....center-of-mass "autocorrelation" function
c   Rg.....radius of gyration
c   r.....records displacement of each bead
c   Cr.....time correlation for bead displacement                       160
c   Ccmf.....time correlation for bead motion in center-of-mass frame
c   arm.....time correlation for star arm vectors
c   attmov.....array for attempted moves
c   accmov.....array for accepted moves
c   tRg.....accumulate Rg for averaging
c   tRe.....accumulate Re for averaging
c   moblty.....final mobility
c   nrm.....normalization constant
c   hist.....histogram array for center-of-mass pair-correlation function
c   grcm.....center-of-mass radial distribution function               170
c   dr.....bin width for pair distribution
c   endbead.....sets arm end beads (value set in beginning of program)
c
c ----------

      integer       imc1, imc2, jpoly, iseed
      integer       jbead, i, j, lper, k, endbead(3)
      integer       istream, iconfig, itrash
      integer       kconn(2,nbead), lpoly(3,nbead)
      integer       locc(nx,ny,nz), loccmid(nxmid,nymid,nzmid)            180
      external      lper

      double precision cmpoly(ndata,3,npoly)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      double precision msd(0:ndata), Cr(0:ndata), ran3
      double precision Rg(ndata,3), r(ndata,3,rbead), tRg
      double precision attmov(nbead), accmov(nbead), moblty(n)
      double precision arm(3,0:ndata), Ccmf(0:ndata), nrm
      double precision grcm, const, pi, d, dr, hist(0:maxbin)

      character*(25) coordfile(nstream)                                  190
      character*(10) msdfile(10)
      character*(10) grfile(10)

c ---------------------------------.

      msdfile(1)="msd1"

      grfile(1)="grcm1"
                                                                         200
      coordfile(1) = "configs/stream.coord1"
      coordfile(2) = "configs/stream.coord2"
      coordfile(3) = "configs/stream.coord3"
      coordfile(4) = "configs/stream.coord4"
      coordfile(5) = "configs/stream.coord5"
      coordfile(6) = "configs/stream.coord6"
      coordfile(7) = "configs/stream.coord7"
      coordfile(8) = "configs/stream.coord8"
      coordfile(9) = "configs/stream.coord9"
      coordfile(10) = "configs/stream.coord10"                           210

c ====================
c  Set arm end beads
c ====================

      endbead(1) = 1
      endbead(2) = 300
      endbead(3) = 450

      pi = acos(-1.0)                                                    220
      dr = .2

      do i=1,nbead
        attmov(i) = 0.d0
        accmov(i) = 0.d0
      enddo

      do i=0,maxbin
        hist(i)=0.d0
      enddo                                                              230

c ================================
c  Read in runtime configuration
c ================================
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      open(unit=10,file="runtime.cfg",status="old")
      read(10,*)iseed
      read(10,*)istream
      read(10,*)iconfig
      close(10)                                                    240

      call start(nx,ny,nz,nxmid,nymid,nzmid,kconn,locc,loccmid,
     :           attmov,accmov,kdata,pi,dr,nconfig,iconfig,iseed,
     :                 equil,nrelax,istream,coordfile,lbead,abead)

      write(*,*)" "
      write(*,*)" Beginning run..."
      write(*,*)" "

C ====================                                              250
C START THE MAIN LOOP
C ====================

      do k=istart,irunmax

        if (k .gt. istart) then
          do i=0,maxbin
            hist(i)=0.d0
          enddo
        endif                                                      260

        do imc1=iconfig+1, ndata

c ==========================================
c  Save coordinates & runtime configuration
c ==========================================

          istream = 1 + (imc1-1)/nconfig

          open(35,file=coordfile(istream),status='unknown',        270
     :           access='append',form='unformatted')
          write(35)lpoly
          close(35)

          itrash = -int(5.0d5*ran3(iseed))
          open(unit=10,file="runtime.cfg",status="unknown")
          write(10,'(2x,i,a)')itrash, " : Random number seed"
          write(10,'(2x,i,a)')istream, " : Number of stream files"
          write(10,'(2x,i,a)')imc1, " : Number of configurations"
          close(10)                                                280

c =================
c  Accumulate data
c =================
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      call data(imcl,dr)

      if (mod((imcl-10),10) .eq. 0) then

c ====================                                              290
c  Calculate Re & Rg
c ====================

        tRg=0.d0

        do i=1,imcl
          tRg=tRg+Rg(i,1)+Rg(i,2)+Rg(i,3)
        enddo

        tRg=tRg/dble(imcl)                                         300

c ======================================
c  Calculate time correlation functions
c ======================================

        call corr(imcl,msd,Cr,arm,endbead,Ccmf)

c ========================
c  Save intermediate data
c ========================                                        310

        open(unit=80,file=grfile(k),status="unknown")
        write(80,25)"#kd=",kdata,"nd=",ndata,"imcl=",imcl,
     :              "<Rg^2>=",tRg

        write(80,5)"#"
        write(80,5)"#      r       g_cm(r)"

        const = 4.0*pi*dble(npoly)/(3.0*dble(nx*ny*nz))
        i = 0                                                      320
        d = dble(i)*dr
        grcm=hist(i)/(.5**3*const*dr**3*dble(npoly)*
     :                                      dble(imcl))

        write(80,102)d,grcm

        do i=1,maxbin

          d = dble(i)*dr
          grcm=hist(i)/(((dble(i)+.5)**3-(dble(i)-.5)**3)*         330
     :                  const*dr**3*dble(npoly)*dble(imcl))

          write(80,102)d,grcm

        enddo
        close(80)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
          open(unit=30,file=msdfile(k),status='unknown')
          write(30,25)"#kd=",kdata,"nd=",ndata,"imc1=",imc1,
     :                "<Rg^2>=",tRg

          write(30,5)"#"
          write(30,5)"# tstep <Rcm(t)^2> D(t)N  <r(t)^2> ",
     :               "<Rcmf(t)^2> arm1   arm2      arm3"

          write(30,101)0,0,0,0,0,arm(1,0),arm(2,0),arm(3,0)

          do i=1, imc1-1
            write(30,101)i*kdata,msd(i),
     :                  msd(i)*N/(6.d0*dble(i*kdata)),
     :                  Cr(i),Ccmf(i),arm(1,i),arm(2,i),
     :                  arm(3,i)
          enddo
          close(30)

        endif

c ========================================
c  Advance the configuration kdata steps
c ========================================

        do imc2=1, kdata
          call advance (N,npoly,nbead,nx,ny,nz,nxmid,nymid,nzmid,
     :    kconn,lpoly,locc,loccmid,iseed,attmov,accmov,lbead,abead)
        enddo

      enddo

c ========================================
c  Run for kdata*ndata time steps without collecting data
c ========== ================

        if (k .lt. irunmax) then

          do imc2=1, kdata*ndata

            call advance (N,npoly,nbead,nx,ny,nz,nxmid,nymid,nzmid,
     :      kconn,lpoly,locc,loccmid,iseed,attmov,accmov,lbead,abead)

            if (mod((imc2-100000),100000) .eq. 0) then

              open(65,file="run.coord",status="unknown")

              write(65,25)"kd*nd=",kdata*ndata,"imc2=",imc2

              do i=1, nbead
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
            write(65,100)lpoly(1,i),lpoly(2,i),lpoly(3,i)
          enddo

          close(65)

        endif

      enddo

    endif

  enddo

c ========================
c  Work-up and save data
c ========================

  do i=1,n
    moblty(i)=0.d0
  enddo

  do i=1,nbead
    accmov(i)=accmov(i)/attmov(i)
  enddo

  do jpoly=1,npoly
    do i=1,n
      jbead = i + N*(jpoly-1)
      moblty(i)=moblty(i)+accmov(jbead)
    enddo
  enddo

  do i=1,n
    moblty(i)=moblty(i)/dble(npoly)
  enddo

  open(unit=70,file="final.data",status="unknown")

  write(70,*)"# Bead Mobility"
  do i=1,n
    write(70,101)i,moblty(i)
  enddo

  close(70)


  open (16, file='end.coord', status='unknown')

  write (16, 90) N,   ' : Chain length '
  write (16, 90) npoly, ' : Number of chains'
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      do jbead = 1, nbead
        write(16,100)lpoly(1,jbead),lpoly(2,jbead),lpoly(3,jbead)
      enddo


    5 format (4(a))
   25 format (a,i10,1x,a,i10,1x,a,i9,3x,a,e10.4,2x,a,e10.4)
   90 format (1x,i5,1x,a)
  100 format (1x, i4, 1x, i4, 1x, i4)
  101 format (1x,i10,7(1x,e10.4))
  102 format (1x,f10.4,2x,e10.4,1x,e10.4)


      stop
      end



c ******************** SUBROUTINE CORR ***************************

c ================================================
c  subroutine for time correlatation functions
c ================================================

      subroutine corr(imc1,msd,Cr,arm,endbead,Ccmf)

      implicit none

      common/block1/cmpoly,r,Rg,lpoly,hist

      integer       n,npoly,nbead,rpoly,rbead,ndata,maxbin
      parameter     (N = 450)
      parameter     (npoly = 64)
      parameter     (nbead = n*npoly)

      parameter     (rpoly = 1)
      parameter     (rbead = n*rpoly)

      parameter     (ndata = 2000)
      parameter     (maxbin = 100)

      integer        imc1,t,to,tto,mod
      integer        jpoly,jbead,i
      integer        lpoly(3,nbead),endbead(3),e(3)

      double precision cmpoly(ndata,3,npoly),msd(0:ndata),cmo(3)
      double precision norm(0:ndata),r(ndata,3,rbead)
      double precision Cr(0:ndata),Cro(3)
      double precision arm(3,0:ndata)
      double precision Ccmf(0:ndata),Ccmfo(3)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      double precision hist(0:maxbin),Rg(ndata,3)                    490

c ---------------------------------

c ================================
c  Polymer center-of-mass displacement correlation
c ================================

      do t=0,imc1-1
        msd(t) = 0.d0                                                500
        norm(t) = 0.d0
      enddo

      do jpoly=1,npoly

        do to=1,imc1

          cmo(1) = cmpoly(to,1,jpoly)
          cmo(2) = cmpoly(to,2,jpoly)
          cmo(3) = cmpoly(to,3,jpoly)                                510

          do tto=to,imc1

            t = tto - to
            msd(t) = msd(t) + (cmo(1)-cmpoly(tto,1,jpoly))**2 +
     :                       (cmo(2)-cmpoly(tto,2,jpoly))**2 +
     :                       (cmo(3)-cmpoly(tto,3,jpoly))**2

            norm(t) = norm(t) + 1.d0
                                                                     520
          enddo

        enddo

      enddo

      do t=0,imc1-1
         msd(t) = msd(t)/norm(t)
         do i=1,3
           arm(i,t)=arm(i,t)/norm(t)                                 530
         enddo
      enddo

c ========================================
c  Bead displacement correlation in normal and center-of-mass frame
c ========================================

      do t=0,imc1-1
        Cr(t)=0.d0
        Ccmf(t)=0.d0                                                 540
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      norm(t)=0.d0
    enddo

    do jpoly=1,rpoly

      do jbead=1+(jpoly-1)*n,jpoly*n

        do to=1,imc1

          Cro(1) = r(to,1,jbead)                                          550
          Cro(2) = r(to,2,jbead)
          Cro(3) = r(to,3,jbead)
          Ccmfo(1) = cmpoly(to,1,jpoly)-r(to,1,jbead)
          Ccmfo(2) = cmpoly(to,2,jpoly)-r(to,2,jbead)
          Ccmfo(3) = cmpoly(to,3,jpoly)-r(to,3,jbead)

          do tto=to,imc1

            t = tto - to
                                                                          560
            Cr(t) = Cr(t) + (Cro(1)-r(tto,1,jbead))**2 +
     :                      (Cro(2)-r(tto,2,jbead))**2 +
     :                      (Cro(3)-r(tto,3,jbead))**2

            Ccmf(t) = Ccmf(t) + (Ccmfo(1)+r(tto,1,jbead)-
     :                          cmpoly(tto,1,jpoly))**2 +
     :                          (Ccmfo(2)+r(tto,2,jbead)-
     :                          cmpoly(tto,2,jpoly))**2 +
     :                          (Ccmfo(3)+r(tto,3,jbead)-
     :                          cmpoly(tto,3,jpoly))**2                 570

            norm(t) = norm(t) + 1.d0

          enddo

        enddo

      enddo

    enddo                                                                 580

    do t=0,imc1-1
       Cr(t) = Cr(t)/norm(t)
       Ccmf(t) = Ccmf(t)/(norm(t)*2.d0)
    enddo

    do t=0,imc1-1
      norm(t)=0.d0
      do i=1,3
        arm(i,t)=0.d0                                                     590
      enddo
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      enddo

      do jpoly=1,rpoly

        mod = n*(jpoly-1)
        e(1) = endbead(1)+mod
        e(2) = endbead(2)+mod
        e(3) = endbead(3)+mod                                          600

        do to=1,imcl
          do tto=to,imcl

            t = tto - to

            do i=1,3
              arm(i,t)=arm(i,t)+( ( r(to,1,e(i))-
     :                              r(to,1,mod) )*
     :                            ( r(tto,1,e(i))-
     :                              r(tto,1,mod) ) )+                610
     :                          ( ( r(to,2,e(i))-
     :                              r(to,2,mod) )*
     :                            ( r(tto,2,e(i))-
     :                              r(tto,2,mod) ) )+
     :                          ( ( r(to,3,e(i))-
     :                              r(to,3,mod) )*
     :                            ( r(tto,3,e(i))-
     :                              r(tto,3,mod) ) )
            enddo
                                                                       620
            norm(t) = norm(t) + 1.d0

          enddo
        enddo

      enddo

      do t=0,imcl-1
        do i=1,3
          arm(i,t)=arm(i,t)/norm(t)                                    630
        enddo
      enddo


      return
      end


c ********************* SUBROUTINE DATA ************************** 640

      subroutine data(i,dr)                                            data
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      implicit none

      common/block1/cmpoly,r,Rg,lpoly,hist

      integer        n,npoly,nbead,rpoly,rbead,ndata,maxbin
      integer        nx, ny, nz
      parameter      (N = 450)                                        650
      parameter      (npoly = 64)
      parameter      (nbead = n*npoly)

      parameter      (rpoly = 1)
      parameter      (rbead = n*rpoly)

      parameter      (ndata = 2000)
      parameter      (maxbin = 100)

      parameter      (nx = 40)                                        660
      parameter      (ny = 40)
      parameter      (nz = 36)

      integer        jx, jy, jz, imcl
      integer        jbead, j, jpoly, i, k, bin
      integer        lpoly(3,nbead)

      double precision cmpoly(ndata,3,npoly)
      double precision Rg(ndata,3), r(ndata,3,rbead)
      double precision rxij, ryij, rzij                               670
      double precision rxxij, ryyij, rzzij
      double precision rij, rijsq, d, dr, hist(0:maxbin)

c ------------------------------------------------------------------

      Rg(i,1)=0.d0
      Rg(i,2)=0.d0
      Rg(i,3)=0.d0
                                                                      680
      do jpoly=1, npoly

        cmpoly(i,1,jpoly)=0.d0
        cmpoly(i,2,jpoly)=0.d0
        cmpoly(i,3,jpoly)=0.d0

        do jx=1, N

          jbead = jx + N*(jpoly-1)
          cmpoly(i,1,jpoly)=cmpoly(i,1,jpoly)+                        690
     :          dble(lpoly(1,jbead))
          cmpoly(i,2,jpoly)=cmpoly(i,2,jpoly)+
     :          dble(lpoly(2,jbead))
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
          cmpoly(i,3,jpoly)=cmpoly(i,3,jpoly)+
     :          dble(lpoly(3,jbead))

        enddo

        cmpoly(i,1,jpoly)=cmpoly(i,1,jpoly)/dble(N)
        cmpoly(i,2,jpoly)=cmpoly(i,2,jpoly)/dble(N)                    700
        cmpoly(i,3,jpoly)=cmpoly(i,3,jpoly)/dble(N)

c ====================
c  Radius of gyration
c ====================

        do jx=1,n

          jbead = jx + N*(jpoly-1)
                                                                       710
          Rg(i,1)=Rg(i,1)+(dble(lpoly(1,jbead))-
     :                     cmpoly(i,1,jpoly))**2
          Rg(i,2)=Rg(i,2)+(dble(lpoly(2,jbead))-
     :                     cmpoly(i,2,jpoly))**2
          Rg(i,3)=Rg(i,3)+(dble(lpoly(3,jbead))-
     :                     cmpoly(i,3,jpoly))**2

        enddo

      enddo                                                            720

      Rg(i,1)=Rg(i,1)/(dble(n)*dble(npoly))
      Rg(i,2)=Rg(i,2)/(dble(n)*dble(npoly))
      Rg(i,3)=Rg(i,3)/(dble(n)*dble(npoly))

      do jbead=1, rbead

        r(i,1,jbead)=dble(lpoly(1,jbead))
        r(i,2,jbead)=dble(lpoly(2,jbead))
        r(i,3,jbead)=dble(lpoly(3,jbead))                              730

      enddo

c ================================================
c  Radial distribution function of center-of-mass
c ================================================

      do j=1, npoly-1
        do k=j+1, npoly
                                                                       740
          rxij=cmpoly(i,1,j)-cmpoly(i,1,k)
          ryij=cmpoly(i,2,j)-cmpoly(i,2,k)
          rzij=cmpoly(i,3,j)-cmpoly(i,3,k)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      rxxij = rxij-anint(rxij/dble(nx))*dble(nx)
      ryyij = ryij-anint(ryij/dble(ny))*dble(ny)
      rzzij = rzij-anint(rzij/dble(nz))*dble(nz)

      rijsq = rxxij**2 + ryyij**2 + rzzij**2
      rij = sqrt(rijsq)                                                  750

      bin = anint(rij/dr)

      if (bin .le .maxbin) then
        hist(bin)=hist(bin) + 2.d0
      endif

    enddo
  enddo
                                                                         760

  return
  end


c ******************* SUBROUTINE START *************************

      subroutine start(nx,ny,nz,nxmid,nymid,nzmid,kconn,locc,loccmid,
     :                 attmov,accmov,kdata,pi,dr,nconfig,iconfig,iseed,  770
     :                      equil,nrelax,istream,coordfile,lbead,abead)

      implicit none

      common/block1/cmpoly,r,Rg,lpoly,hist

      integer       n,npoly,nbead,rpoly,rbead,ndata,maxbin
      parameter     (N = 450)
      parameter     (npoly = 64)
      parameter     (nbead = n*npoly)                                    780

      parameter     (rpoly = 1)
      parameter     (rbead = n*rpoly)

      parameter     (ndata = 2000)
      parameter     (maxbin = 100)

      integer       nx, ny, nz, nxmid, nymid, nzmid, lbead, abead
      integer       nrelax, kdata, jx, jy, jz, iconfig, jconfig
      integer       iseed, jxxx, jyyy, jzzz, jcon2, itrash             790
      integer       jbead, j, jpoly, jxmid, jymid, jzmid, imc1
      integer       lper, iu, i, k, irestart, bin, istream
      integer       kconn(2,nbead), lpoly(3,nbead), nconfig
      integer       locc(nx,ny,nz), loccmid(nxmid,nymid,nzmid)
      external      lper
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      logical          equil

      double precision attmov(nbead),accmov(nbead)
      double precision cmpoly(ndata,3,npoly)                              800
      double precision Rg(ndata,3), r(ndata,3,rbead)
      double precision rxij, ryij, rzij
      double precision rxxij, ryyij, rzzij, const
      double precision rij, rijsq, pi, d, dr, hist(0:maxbin)

      character*(25) coordfile(10)

c ---------------------------------------------------------------

      if(istream .eq. 0) then                                             810

        if (equil .eqv. .false.) then

          open (10, file = 'start.coord', status = 'old')

          read (10, *) itrash
          if (itrash .ne. N) then
            write (*, *) 'Number of beads does not match input'
            stop
          endif                                                           820

          read (10, *) itrash
          if (itrash .ne. npoly) then
            write (*, *) 'Number of polymers does not match input'
            stop
          endif

          read (10, *)

          do jbead = 1, nbead                                             830
            read(10,*)jx,jy,jz

            lpoly(1,jbead) = jx
            lpoly(2,jbead) = jy
            lpoly(3,jbead) = jz
          enddo

          close (10)

        else                                                              840

          open (unit=20,file='equil.coord',status='old')

          read (20, *) itrash
          if (itrash .ne. N) then
            write (*, *) 'Number of beads does not match input'
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
        stop
      endif

      read (20, *) itrash                                              850
      if (itrash .ne. npoly) then
        write (*, *) 'Number of polymers does not match input'
        stop
      endif

      do jbead = 1, nbead
        read (20,100) jx, jy, jz

        lpoly(1,jbead) = jx
        lpoly(2,jbead) = jy                                            860
        lpoly(3,jbead) = jz
      enddo

      close (20)

    endif

c ==============
c RUN PARAMETERS
c ==============                                                       870

    iu = 9
    open (unit=iu,file='vital.stats',status='unknown',
   :                                  form='formatted')
    write (iu, *) ' '
    write (iu,55) '            Chain length: ', N
    write (iu,55) '        Number of chains: ', npoly
    write (iu, *) ' '
    write (iu,60) '     Simulation box size: ', nx, ny, nz
    write (iu, *) ' '                                                  880
    write (iu,50) '         Average density: ',
   :                  dble(nbead)/dble(nx*ny*nz)
    write (iu, *) ' '
    write (iu,55) ' MCS between data recording: ', kdata
    write (iu,55) '   Number of data records: ', ndata
    write (iu,55) '      Total number of MCS: ', kdata*ndata
    if(equil .eqv. .false.) then
      write (iu,55) '       Equilibration MCS: ', nrelax
    else
      write (iu,55) '       Equilibration MCS: N/A'                  890
    endif
    write (iu, *) ' '

    close (unit = iu)

  else
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
c ================================
c  We are restarting from a crash
c ================================                                    900

      write(*,*)" "
      write(*,*)" Restarting..."
      write(*,*)" "

c ========================
c  Open stream.coord files
c ========================

      do k=1, istream                                                 910

        write(*,'(a,a)')" Accessing file: ", coordfile(k)

        open(15,file=coordfile(k),status='old',
     :                        form='unformatted')

        do i=1, nconfig

          jconfig = i + (k-1)*nconfig
                                                                      920
          if(jconfig .le. iconfig) then
            read(15)lpoly

            call data(jconfig,dr)
          endif

        enddo

        close(15)
                                                                      930
      enddo

      write(*,*)" "
      write(*,*)" Done reading files."
      write(*,*)" "


    endif

c ============                                                        940
c Connectivity
c ============

    call connect(npoly,n,kconn,nbead,lbead,abead)

c ================================
c  Initialize occupation numbers
c ================================
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      call occunum(n,nbead,nx,ny,nz,nxmid,nymid,nzmid,locc,loccmid,
     :                                          kconn,lpoly,abead)


C Check again whether restarting, if true then advance kdata steps

      if (istream .ne. 0) then

        write(*,*)" "
        write(*,*)" Advancing configuration kdata steps..."
        write(*,*)" "

        do imc1=1, kdata
          call advance (N,npoly,nbead,nx,ny,nz,nxmid,nymid,nzmid,kconn,
     :                lpoly,locc,loccmid,iseed,attmov,accmov,lbead,abead)
        enddo

        write(*,*)" "
        write(*,*)" Done restarting."
        write(*,*)" "

      endif


C Check again whether the system is equilibrated

      if (equil .eqv. .false.) then

c ====================
c EQUILIBRATION PERIOD
c ====================

        write(*,*)" "
        write(*,*)" Equilibrating..."
        write(*,*)" "

        do imc1 = 1, nrelax

          call advance (N,npoly,nbead,nx,ny,nz,nxmid,nymid,nzmid,kconn,
     :                lpoly,locc,loccmid,iseed,attmov,accmov,lbead,abead)

          if (mod((imc1-100000),100000) .eq. 0) then
            open(68,file="run.coord",status="unknown")
            write(68,30)"nrelax=",nrelax,"imc1=",imc1
            do i=1, nbead
              write(68,100)lpoly(1,i),lpoly(2,i),lpoly(3,i)
            enddo
            close(68)
          endif
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      enddo                                                          1000

c     Write equilibrated coordinates...

      open (unit=77, file='equil.coord', status='unknown')

      write (77, 90) N,   ' : Chain length '
      write (77, 90) npoly, ' : Number of chains'

      do jbead = 1, nbead
        write(77,100)lpoly(1,jbead),lpoly(2,jbead),lpoly(3,jbead)    1010
      enddo
      close (unit=77)

c ------------
c Check for bond crossing
c ------------

      call  bcross(n,nbead,nxmid,nymid,nzmid,kconn,lpoly,
     :                                  loccmid,lbead,abead)
                                                                     1020
      endif


   30 format (1x,a,i10,3x,a,i10)
   50 format (a, f8.4)
   55 format (a, i10)
   60 format (a, i4, 1x, i4, 1x, i4)
   90 format (1x, i6, a)
  100 format (1x, i4, 1x, i4, 1x, i4)
                                                                     1030
      return
      end



c ******************** SUBROUTINE ADVANCE *************************

c ==========================================
c ADVANCE: This subroutine advances the dynamic Monte Carlo time
c evolution of a lattice polymer system by one MC step. A total of NBEAD   1040
c time steps are attempted. The polymer chains are modeled by the
c ONE-SITE bond fluctuation model.
c Periodic boundary conditions are applied in all coordinate directions.
c ==========================================

      subroutine advance (N,npoly,nbead,nx,ny,nz,nxmid,nymid,
     :    nzmid,kconn,lpoly,locc,loccmid,iseed,attmov,accmov,
     :                                    lbead,abead)

c INPUT                                                              1050
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
c     nbead.....Total number of polymer beads
c     nx,ny,nz.....Size of the lattice box for polymer beads
c     nxmid,
c     nymid,
c     nzmid.....Size of the lattice box for bond midpoints
c     lpoly.....Initial lattice coordinates of all polymer beads
c     locc.....Lattice occupation numbers for polymer beads:
c
c                    LOCC(jx,jy,jz) = 0 if (jx,jy,jz) is unoccupied;
c                    LOCC(jx,jy,jz) = 1 if (jx,jy,jz) is occupied.
c
c     loccmid.....Lattice occupation numbers for bond midpoints:
c
c                    LOCCMID(jxm,jym,jzm) = 0 if no bond midpoint
c                    is present at position (jxm/2,jym/2,jzm/2).
c
c                    LOCCMID(jxm,jym,jzm) is a positive integer, M,
c                    if M bond midpoints are present at position
c                    (jxm/2,jym/2,jzm/2) on the lattice;
c
c     kconn.....Array containing the connectivity information:
c
c                    For polymer beads not at chain ends,
c                    KCONN(1,I) = I-1 and KCONN(2,I) = I+1,
c                    signifying that the bead I is bonded to
c                    its two neighbors,
c
c                    for the first bead of a polymer chain,
c                    KCONN(1,I) = 0 and KCONN(2,I) = I+1,
c                    because there is no bond to the "left",
c
c                    for the last bead of a polymer chain,
c                    KCONN(1,I) = I-1 and KCONN(2,I) = 0,
c                    because there is no bond to the "right".
c
c
c     iseed.....Initial random number seed
c
c OUTPUT
c     lpoly.....Updated lattice coordinates of all beads
c     locc.....Updated occupation numbers for beads
c     loccmid.....Updated occupation numbers for bond midpoints
c     iseed.....New random number seed
C     attmov.....array storing attempted moves for each bead
C     accmov.....array for storing accepted moves
c
c ----------

      implicit none

c - ARGUMENTS ----
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      integer         N, npoly, lbead, abead, ajb
      integer         nbead, nx, ny, nz, nxmid, nymid, nzmid, iseed
      integer         kconn(2,nbead)
      integer         lpoly(3,nbead)
      integer         locc(nx,ny,nz)
      integer         loccmid(nxmid,nymid,nzmid)

c - LOCAL ------
      integer         iatt, jbead, jmove, jcon1, jcon2, jx, jy, jz,          1110
     :                jxmid1, jymid1, jzmid1, jxmid2, jymid2, jzmid2,
     :                jxxx, jyyy, jzzz, LSQR, jpoly
      integer         jxmid3, jymid3, jzmid3

      integer         lper
      double precision ran3, rannum
      double precision attmov(nbead), accmov(nbead)
      external        lper, ran3

c ---------------------------------------------------------------------  1120


c =====================
c  -------
c    MOVE ATTEMPTS
c  -------
c =====================

      do iatt = 1, nbead
                                                                           1130
c ================
c Select the event
c ================

        jbead = 1 + int(dble(nbead)*ran3(iseed))
        attmov(jbead)=attmov(jbead) + 1.d0
        ajb = int(jbead/n)*n

        jmove = 1 + int(6.D0*ran3(iseed))
                                                                           1140
c ==================
c Generate trial move
c ==================

        jx = lpoly(1,jbead)
        jy = lpoly(2,jbead)
        jz = lpoly(3,jbead)

        if (jmove .eq. 1) then
          jx = jx + 1                                                      1150
        else if (jmove .eq. 2) then
          jx = jx - 1
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      else if (jmove .eq. 3) then
        jy = jy + 1
      else if (jmove .eq. 4) then
        jy = jy - 1
      else if (jmove .eq. 5) then
        jz = jz + 1
      else
        jz = jz - 1                                              1160
      endif


      jcon1 = kconn(1,jbead)
      jcon2 = kconn(2,jbead)

      if (jcon1 .ne. 0) then
        jxmid1 = jx + lpoly(1,jcon1)
        jymid1 = jy + lpoly(2,jcon1)
        jzmid1 = jz + lpoly(3,jcon1)                             1170
      endif

      if (jcon2 .ne. 0) then
        jxmid2 = jx + lpoly(1,jcon2)
        jymid2 = jy + lpoly(2,jcon2)
        jzmid2 = jz + lpoly(3,jcon2)
      endif

c ===============
c Test trial move                                                1180
c ===============

c As soon as we find a violation of a necessary condition for the
c acceptance of the move, we jump to the end of the loop over trials...

c ------------
c Excluded volume constraints
c ------------

      jxxx = lper(jx,nx)                                         1190
      jyyy = lper(jy,ny)
      jzzz = lper(jz,nz)

      if (locc(jxxx,jyyy,jzzz) .ne. 0) goto 1000
c ----------
c Allowed bond vectors
c ----------

c We have already checked for bond lengths of zero
c by enforcing the excluded volume condition...                   1200

      if(jcon1 .ne. 0) then
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
          LSQR = (jx - lpoly(1,jcon1))**2
     :         + (jy - lpoly(2,jcon1))**2
     :         + (jz - lpoly(3,jcon1))**2

          if (LSQR .gt. 3) goto 1000

        endif                                                    1210

        if(jcon2 .ne. 0) then

          LSQR = (jx - lpoly(1,jcon2))**2
     :         + (jy - lpoly(2,jcon2))**2
     :         + (jz - lpoly(3,jcon2))**2

          if (LSQR .gt. 3) goto 1000

        endif                                                    1220

        if(jbead-ajb .eq. lbead) then

          LSQR = (jx - lpoly(1,jbead-1))**2
     :         + (jy - lpoly(2,jbead-1))**2
     :         + (jz - lpoly(3,jbead-1))**2

          if (LSQR .gt. 3) goto 1000

          LSQR = (jx - lpoly(1,jbead+1))**2                      1230
     :         + (jy - lpoly(2,jbead+1))**2
     :         + (jz - lpoly(3,jbead+1))**2

          if (LSQR .gt. 3) goto 1000

          LSQR = (jx - lpoly(1,abead+ajb))**2
     :         + (jy - lpoly(2,abead+ajb))**2
     :         + (jz - lpoly(3,abead+ajb))**2

          if (LSQR .gt. 3) goto 1000                             1240

        endif

c -------
c Bond crossing
c -------

        if (jcon1 .ne. 0) then
          jxxx = lper(jxmid1,nxmid)
          jyyy = lper(jymid1,nymid)                              1250
          jzzz = lper(jzmid1,nzmid)

          if (loccmid(jxxx,jyyy,jzzz) .ne. 0) goto 1000
        endif
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      if (jcon2 .ne. 0) then
        jxxx = lper(jxmid2,nxmid)
        jyyy = lper(jymid2,nymid)
        jzzz = lper(jzmid2,nzmid)
                                                                    1260
        if (loccmid(jxxx,jyyy,jzzz) .ne. 0) goto 1000
      endif

      if (jbead-ajb .eq. lbead) then
        jxmid3 = jx + lpoly(1,abead+ajb)
        jymid3 = jy + lpoly(2,abead+ajb)
        jzmid3 = jz + lpoly(3,abead+ajb)

        jxxx = lper(jxmid3,nxmid)
        jyyy = lper(jymid3,nymid)                                   1270
        jzzz = lper(jzmid3,nzmid)

        if (loccmid(jxxx,jyyy,jzzz) .ne. 0) goto 1000
      endif

c ==============
c UPDATE SYSTEM
c ==============

c If we get to this point, then the move has been accepted...       1280

       accmov(jbead)=accmov(jbead)+1.d0

c Update occupation numbers:

      jxxx = lper(lpoly(1,jbead),nx)
      jyyy = lper(lpoly(2,jbead),ny)
      jzzz = lper(lpoly(3,jbead),nz)
      locc(jxxx,jyyy,jzzz) = 0
                                                                    1290
      jxxx = lper(jx,nx)
      jyyy = lper(jy,ny)
      jzzz = lper(jz,nz)
      locc(jxxx,jyyy,jzzz) = 1


c Update bond midpoints:

      if (jcon1 .ne. 0) then
        jxxx = lper(lpoly(1,jbead) + lpoly(1,jcon1), nxmid)         1300
        jyyy = lper(lpoly(2,jbead) + lpoly(2,jcon1), nymid)
        jzzz = lper(lpoly(3,jbead) + lpoly(3,jcon1), nzmid)
        loccmid(jxxx,jyyy,jzzz) = loccmid(jxxx,jyyy,jzzz) - 1

        jxxx = lper(jxmid1,nxmid)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
            jyyy = lper(jymid1,nymid)
            jzzz = lper(jzmid1,nzmid)
            loccmid(jxxx,jyyy,jzzz) = loccmid(jxxx,jyyy,jzzz) + 1
          endif
                                                                          1310
          if (jcon2 .ne. 0) then
            jxxx = lper(lpoly(1,jbead) + lpoly(1,jcon2), nxmid)
            jyyy = lper(lpoly(2,jbead) + lpoly(2,jcon2), nymid)
            jzzz = lper(lpoly(3,jbead) + lpoly(3,jcon2), nzmid)
            loccmid(jxxx,jyyy,jzzz) = loccmid(jxxx,jyyy,jzzz) - 1

            jxxx = lper(jxmid2,nxmid)
            jyyy = lper(jymid2,nymid)
            jzzz = lper(jzmid2,nzmid)
            loccmid(jxxx,jyyy,jzzz) = loccmid(jxxx,jyyy,jzzz) + 1         1320
          endif

          if (jbead-ajb .eq. lbead) then
            jxxx = lper(lpoly(1,jbead) + lpoly(1,abead+ajb), nxmid)
            jyyy = lper(lpoly(2,jbead) + lpoly(2,abead+ajb), nymid)
            jzzz = lper(lpoly(3,jbead) + lpoly(3,abead+ajb), nzmid)
            loccmid(jxxx,jyyy,jzzz) = loccmid(jxxx,jyyy,jzzz) - 1

            jxxx = lper(jxmid3,nxmid)
            jyyy = lper(jymid3,nymid)                                     1330
            jzzz = lper(jzmid3,nzmid)
            loccmid(jxxx,jyyy,jzzz) = loccmid(jxxx,jyyy,jzzz) + 1
          endif

c Update coordinates:

          lpoly(1,jbead) = jx
          lpoly(2,jbead) = jy
          lpoly(3,jbead) = jz
                                                                          1340
 1000 enddo

      return
      end



c ******************** SUBROUTINE OCCUNUM ***************************
                                                                          1350
c ==================================
c  Initializes occupation numbers
c ==================================

      subroutine occunum(n,nbead,nx,ny,nz,nxmid,nymid,nzmid,locc,
     :                                  loccmid,kconn,lpoly,abead)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      implicit none

      integer  nx,ny,nz,nxmid,nymid,nzmid,jx,jy,jz,jxmid                1360
      integer  jymid,jzmid,jxxx,jyyy,jzzz,jcon2,nbead
      integer  jbead,lper,n

      integer  locc(nx,ny,nz),loccmid(nxmid,nymid,nzmid)
      integer  kconn(2,nbead),lpoly(3,nbead),jcon1,abead

      external lper

c ------------------------------------------------------------------
                                                                      1370

        write(*,*)" "
        write(*,*)" Initializing occupation numbers..."
        write(*,*)" "

        do jz = 1, nz
          do jy = 1, ny
            do jx = 1, nx
              locc(jx,jy,jz) = 0
            enddo                                                     1380
          enddo
        enddo

        do jzmid = 1, nzmid
          do jymid = 1, nymid
            do jxmid = 1, nxmid
              loccmid(jxmid,jymid,jzmid) = 0
            enddo
          enddo
        enddo                                                         1390

        do jbead = 1, nbead
          jx = lpoly(1,jbead)
          jy = lpoly(2,jbead)
          jz = lpoly(3,jbead)

          jxxx = lper(jx,nx)
          jyyy = lper(jy,ny)
          jzzz = lper(jz,nz)
                                                                      1400
          if(locc(jxxx,jyyy,jzzz) .ne. 0)then
            write (*,*)'ERROR: Bead overlap'
            write (*,*)'locc(jxxx,jyyy,zzz)=',locc(jxxx,jyyy,jzzz)
            write (*, *) 'jbead, jx, jy, jz: ',jbead,jx,jy,jz
            stop
          endif
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      locc(jxxx,jyyy,jzzz) = jbead

      jcon2 = kconn(2,jbead)                                          1410

      if (jcon2 .ne. 0) then
        jxmid = lpoly(1,jbead) + lpoly(1,jcon2)
        jymid = lpoly(2,jbead) + lpoly(2,jcon2)
        jzmid = lpoly(3,jbead) + lpoly(3,jcon2)

        jxxx = lper(jxmid,nxmid)
        jyyy = lper(jymid,nymid)
        jzzz = lper(jzmid,nzmid)
                                                                      1420
        if (loccmid(jxxx,jyyy,jzzz) .ne. 0) then
          write(*,*)"WARNING: Bond crossing: ",
     :                jxxx,jyyy,jzzz,loccmid(jxxx,jyyy,jzzz)
          write(*,*)"jbead, jx, jy, jz: ",jbead,jx,jy,jz
          write(*,*)"jcon1, jcon2: ",kconn(1,jbead),jcon2
        endif

c NOTE: There may be more than one bond midpoint at a given location...

        loccmid(jxxx,jyyy,jzzz) = loccmid(jxxx,jyyy,jzzz) + 1         1430

        if(jbead-int(jbead/n)*n .eq. abead) then
          jcon1 = kconn(1,jbead)
          jxmid = lpoly(1,jbead) + lpoly(1,jcon1)
          jymid = lpoly(2,jbead) + lpoly(2,jcon1)
          jzmid = lpoly(3,jbead) + lpoly(3,jcon1)

          jxxx = lper(jxmid,nxmid)
          jyyy = lper(jymid,nymid)
          jzzz = lper(jzmid,nzmid)                                    1440

          if (loccmid(jxxx,jyyy,jzzz) .ne. 0) then
            write(*,*)"WARNING: Bond crossing: ",
     :              jxxx,jyyy,jzzz, loccmid(jxxx,jyyy,jzzz)
            write(*,*)"jbead, jx, jy, jz: ",jbead,jx,jy,jz
            write(*,*)"jcon1, jcon2: ",jcon1,jcon2
          endif
          loccmid(jxxx,jyyy,jzzz) = loccmid(jxxx,jyyy,jzzz) + 1
        endif
                                                                      1450
      endif
    enddo

    return
    end
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
c ******************** SUBROUTINE CONNECT ************************** 1460

c ====================
c  Connects the beads
c ====================

      subroutine connect(npoly,n,kconn,nbead,lbead,abead)                connect

      implicit none

      integer npoly,n,nbead,jbead,jpoly,J                                1470

      integer kconn(2,nbead),na,lbead,abead

c ---------------------------------------.


       write(*,*)" "
       write(*,*)" Setting up chain connectivity..."
       write(*,*)" "
                                                                         1480
       do jpoly=1,npoly

          j = 1
          jbead = j + n*(jpoly-1)

          kconn(1,jbead) = 0
          kconn(2,jbead) = jbead+1

          do j = 2,abead-2
                                                                         1490
            jbead = j + n*(jpoly-1)

            kconn(1,jbead) = jbead-1
            kconn(2,jbead) = jbead+1

          enddo

          j = abead-1
          jbead = j + n*(jpoly-1)
                                                                         1500
          kconn(1,jbead) = jbead-1
          kconn(2,jbead) = 0

          j = abead
          jbead = j + n*(jpoly-1)

          kconn(1,jbead) = lbead + n*(jpoly-1)
          kconn(2,jbead) = jbead+1
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      do j = abead+1,n-1                                                1510

        jbead = j + n*(jpoly-1)

        kconn(1,jbead) = jbead-1
        kconn(2,jbead) = jbead+1

      enddo

      j = n
      jbead = j + n*(jpoly-1)                                           1520

      kconn(1,jbead) = jbead-1
      kconn(2,jbead) = 0

    enddo

    return
    end

                                                                        1530

c ******************* FUNCTION bcross *************************

    subroutine bcross(n,nbead,nxmid,nymid,nzmid,kconn,lpoly,
   :                  loccmid,lbead,abead)

    integer jbead,jcon2,jxmid,jymid,jzmid,jxxx,jyyy,jzzz
    integer lpoly(3,nbead),kconn(2,nbead),lper
    integer nxmid,nymid,nzmid,lbead,abead
    integer loccmid(nxmid,nymid,nzmid),n                                1540

    external lper

c ---------------------------------

    do jbead = 1, nbead
      jcon2 = kconn(2,jbead)

      if (jcon2 .ne. 0) then
        jxmid = lpoly(1,jbead) + lpoly(1,jcon2)                         1550
        jymid = lpoly(2,jbead) + lpoly(2,jcon2)
        jzmid = lpoly(3,jbead) + lpoly(3,jcon2)

        jxxx = lper(jxmid,nxmid)
        jyyy = lper(jymid,nymid)
        jzzz = lper(jzmid,nzmid)

        if (loccmid(jxxx,jyyy,jzzz) .ne. 1) then
          write (*, *) 'ERROR: Bond crossing: ',
   :               jxxx,jyyy,jzzz, loccmid(jxxx,jyyy,jzzz)              1560
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
          write(*,*)'jbead, jx, jy, jz: ',jbead,jx,jy,jz
          write(*,*)'jcon1, jcon2: ',kconn(1,jbead),jcon2
          stop
        endif

        if(jbead-int(jbead/n)*n .eq. abead) then
          jcon1 = kconn(1,jbead)
          jxmid = lpoly(1,jbead) + lpoly(1,jcon1)
          jymid = lpoly(2,jbead) + lpoly(2,jcon1)
          jzmid = lpoly(3,jbead) + lpoly(3,jcon1)                    1570

          jxxx = lper(jxmid,nxmid)
          jyyy = lper(jymid,nymid)
          jzzz = lper(jzmid,nzmid)

          if (loccmid(jxxx,jyyy,jzzz) .ne. 1) then
            write (*, *) 'ERROR: Bond crossing: ',
     :           jxxx,jyyy,jzzz, loccmid(jxxx,jyyy,jzzz)
            write(*,*)'jbead, jx, jy, jz: ',jbead,jx,jy,jz
            write(*,*)'jcon1, jcon2: ',jcon1,jcon2                   1580
            stop
          endif
        endif

      endif

    enddo

    write(*,*)" "
    write(*,*)" No bond crossings remain."                           1590
    write(*,*)" "

    return
    end



c ******************** FUNCTION LPER ***************************
                                                                     1600
c ================================================
c LPER: Given an uncorrected lattice coordinate LCOORD, and the size
c of the lattice (in the corresponding coordinate direction) N, this
c function returns the corrected lattice coordinates by applying
c periodic boundary conditions...
c Here we assume that the primary lattice box runs from 1 through N.
c ================================================

    integer function lper ( lcoord, n )                              lper
                                                                     1610
    implicit none
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      integer         lcoord, n

c ---------------------------------

      if (lcoord .lt. 1) then
        lper = n + mod(lcoord,n)
      else
        lper = 1 + mod(lcoord-1,n)                                    1620
      endif

      return
      end



c ******************** FUNCTION RAN3 *************************
                                                                      1630
c ==========================================
c RAN3: Returns a uniform random deviate on [0,1). Set IDUM to
c any negative value to initialize or reinitialize the sequence.
c From "Numerical Recipes", Chapter 7.
c ==========================================

      double precision function ran3 (idum)                          ran3

      implicit none
                                                                      1640
      integer         idum

      double precision mbig, mseed, mz, fac
      parameter (mbig = 4000000.D0)
      parameter (mseed = 1618033.D0)
      parameter (mz = 0.D0)
      parameter (fac = 2.5D-07)

      integer         i, ii, iff, inext, inextp, k
      double precision mj, mk                                         1650
      double precision ma(55)

      save            iff, inext, inextp
      save            ma

      data iff /0/

c ---------------------------------

      if(idum.lt.0.or.iff.eq.0)then                                   1660
        iff=1
        mj=mseed-iabs(idum)
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
      mj=mod(mj,mbig)
      ma(55)=mj
      mk=1
      do 11 i=1,54
        ii=mod(21*i,55)
        ma(ii)=mk
        mk=mj-mk
        if(mk.lt.mz)mk=mk+mbig                                    1670
        mj=ma(ii)
11    continue
      do 13 k=1,4
        do 12 i=1,55
          ma(i)=ma(i)-ma(1+mod(i+30,55))
          if(ma(i).lt.mz)ma(i)=ma(i)+mbig
12      continue
13    continue
      inext=0
      inextp=31                                                   1680
      idum=1
    endif

    inext = inext + 1
    if (inext .eq. 56) inext = 1
    inextp = inextp + 1
    if (inextp .eq. 56) inextp = 1

    mj = ma(inext) - ma(inextp)
    if (mj .lt. mz) mj = mj + mbig                                1690
    ma(inext) = mj

    ran3 = mj*fac

    return
    end
```


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Bibliography

[1] C. S. Marvel. *Polymer Science Overview*, page 53. American Chemical Society, Washington, D.C., 1981.

[2] D. H. Morton-Jones, *Polymer Processing*, (Chapman and Hall, New York, 1989).

[3] R. B. Seymour and H. F. Mark, editors, *Applications of Polymers*, (Plenum Press, New York, 1988).

[4] M. Doi and S.F. Edwards, *The Theory of Polymer Dynamics*, (Oxford University Press, Oxford, 1986).

[5] R. B. Bird and C. F. Curtiss, Physics Today page 36 (Jan. 1984).

[6] R. B. Bird, R. C. Armstrong, and O. Hassager, *Dynamics of Polymeric Liquids*, (John Wiley & Sons, New York, 1987).

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

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[8] G. A. Stahl. *Polymer Science Overview*, page 25. American Chemical Society, Washington, D.C., 1981.

[9] B. H. Zimm, W. H. Stockmayer, and M. Fixman, J. Chem. Phys. **21**, 1716 (1953).

[10] P.-G. de Gennes, *Scaling Concepts in Polymer Physics*, (Cornell University Press, Ithaca, New York, 1979).

[11] P.-G. de Gennes, J. Chem. Phys. **55**, 572 (1971).

[12] J. Skolnick, R. Yaris, and A. Kolinski, J. Chem. Phys. **88**, 1407 (1988).

[13] M. Fixman, J. Chem. Phys. **89**, 3892 (1988).

[14] K. S. Schweizer, J. Chem. Phys. **91**, 5802 (1989).

[15] A. Kolinski, J. Skolnick, and R. Yaris, J. Chem. Phys. **86**, 1567 (1987).

[16] A. Kolinski, J. Skolnick, and R. Yaris, J. Chem. Phys. **86**, 7164; 7174 (1987).

[17] K. Kremer and G. S. Grest, J. Chem. Phys. **92**, 5057 (1990).

[18] R. Faller, F. Müller-Plathe, and A. Heuer, Macromol. **33**, 6602 (2000).

[19] G. S. Grest, L. J. Fetters, J. S. Huang, and D. Richter, Adv. Chem. Phys. **94**, 67 (1996).

[20] A. Baumgärtner, Annu. Rev. Phys. Chem. **35**, 419 (1984).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[21] T. P. Lodge, N. A. Rotstein, and S. Prager, Adv. Chem. Phys. **LXXIX**, 1 (1990).

[22] K. Binder, editor, *Monte Carlo and molecular dynamics simulations in polymer science*, (Oxford University Press, New York, 1995).

[23] K. Binder and W. Paul, J. Polym. Sci. Poly. Phys. **35**, 1 (1997).

[24] I. Carmesin and K. Kremer, Macromol. **21**, 2819 (1988).

[25] J. S. Shaffer, J. Chem. Phys. **101**, 4205 (1994).

[26] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran*, (Cambridge University Press. New York. 1992).

[27] J. S. Shaffer, J. Chem. Phys. **103**, 761 (1995).

[28] K. Kremer, G. S. Grest, and I. Carmesin, Phys. Rev. Lett. **61**, 566 (1988).

[29] J. Skolnick and A. Kolinski, Adv. Chem. Phys. **57**. 223 (1990).

[30] S. Brown and G. Szamel, J. Chem. Phys. **108**, 4705 (1998).

[31] S. Brown and G. Szamel, J. Chem. Phys. **109**, 6184 (1998).

[32] S. Brown and G. Szamel, Phys. Rev. E (Submitted 2000).

[33] J. Klein, Macromol. **19**, 105 (1986).

[34] G. B. McKenna, B. J. Hostetter, N. Hadjichristidis, L. J. Fetters, and D. J. Plazek, Macromol. **22**, 1834 (1989).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[35] S. F. Tead, E. J. Kramer, G. Hadziioannou, M. Antonietti, H. Sillescu, P. Lutz, and C. Strazielle, Macromol. **25**, 3942 (1992).

[36] M. Müller, J. P. Wittmer, and M. E. Cates, Phys. Rev. E **53**, 5063 (1996).

[37] T. Pakula and S. Geyler, Macromol. **21**, 1665 (1988).

[38] M. E. Cates and J. M. Deutsch, J. Phys. **47**, 2121 (1986).

[39] W. Paul, K. Binder, D. Heermann, and K. Kremer, J. Chem. Phys. **95**, 7726 (1991).

[40] J. M. Deutsch, Phys. Rev. E **59**, 2539 (1999).

[41] J. des Cloizeaux, J. Physique **42**, L-433 (1981).

[42] A. Baumgäertner, J. Chem. Phys. **76**, 4275 (1982).

[43] A. Sikorski, Poly. **35**, 3792 (1994).

[44] S. Brown and G. Szamel, Macromol. Theory Simul. **9**, 14 (2000).

[45] T. C. B. McLeish and S. T. Milner, Adv. Poly. Sci. **143**, 195 (1999).

[46] T. C. B. McLeish, Physics World **8**, 32 (1995).

[47] S. T. Milner and T. C. B. McLeish, Macromol. **30**, 2159 (1997).

[48] P.-G. de Gennes, J. Phys. (Paris) **36**, 1199 (1975).

[49] D. S. Pearson and E. Helfand, Macromol. **17**, 888 (1984).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[50] C. R. Bartels, B Crist, Jr., L. J. Fetters, and W. Graessley, Macromol. **19**, 785 (1986).

[51] L. J. Fetters, A. D. Kiss, D. S. Pearson, G. Quack, and F. J. Vitus, Macromol. **26**, 647 (1993).

[52] R. J. Needs and S. F. Edwards, Macromol. **16**, 1492 (1983).

[53] G. T. Barkema and A. Baumgärtner, Macromol. **32**, 911 (1999).

[54] A. Sikorski, A. Kolinski, and J. Skolnick, Macromol. Theory Simul. **3**, 715 (1994).

[55] K. E. Evans, J. Chem. Soc., Faraday Trans. 2 **77**, 2385 (1981).

[56] J. Batoulis and K. Kremer, Macromol. **22**, 4277 (1989).

[57] A. Sikorski, Makromol. Chem. **2**, 309 (1993).

[58] L. A. Molina and J. J. Freire, Macromol. **32**, 499 (1999).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.