MONTE CARLO COMPUTER SIMULATION OF THE LENNARD-JONES AND STOCKMAYER FLUID PHASE DIAGRAMS

by

Victor Paul Gregory, Jr.

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Chemistry

APPROVED:

J.C. Schug, Chairman

P.E. Field

H. Marand

G.A. Schick

C.D. Williams

February, 1994

Blacksburg, Virginia

MONTE CARLO COMPUTER SIMULATION OF THE LENNARD-JONES AND STOCKMAYER FLUID PHASE DIAGRAMS

by

Victor Paul Gregory, Jr.

Committee Chairman: John C. Schug
Chemistry

(ABSTRACT)

The isotherms of the Lennard-Jones fluid and the Stockmayer fluid are calculated by Monte Carlo computer simulation using the constant NpT ensemble. Empirical coefficients are determined for a truncated virial equation of state fitted to our data. Spinodal points are located for each temperature and fluid. For temperatures less than 0.90 of the critical temperature, we succeeded in temporarily isolating clusters during the gas to liquid transition for the LJ fluid. Density profiles are calculated for clusters at and above the spinodal pressures. The clusters above the spinodal pressure have liquid-like densities at their centers and are identified as critical condensation clusters. The clusters at the spinodal increase in size with temperature and have densities roughly half as dense as the equilibrium liquid at their centers. It is found that the results are essentially system size independent.

ACKNOWLEDGEMENTS

I would like to thank Dr. John Schug for his guidance in the research reported in this dissertation. I have the utmost respect for his understanding and knowledge of chemistry and physics. He is an outstanding person as well as a scientist.

Thanks also go to Dr. Paul Field, Dr. Herve Marand, Dr. Alan Schick and Dr. Clayton Williams for their support.

Lastly, special thanks to my friends and family, who have encouraged me in numerous ways during my education.

Table of Contents

Table of Figures

Table of Tables

I. INTRODUCTION

This dissertation describes the isotherms of particles interacting with the Lennard-Jones (LJ) potential and the Stockmayer potential (LJ + point dipole). In particular, it is concerned with the possibility of observing the onset of the gas-to-liquid transition. This is done through the use of the constant-NpT Monte Carlo simulation technique.

Gases and liquids have states in which they are stable, metastable, or unstable. The stable gas and stable liquid states are single-phase. The metastable and unstable states are distinguished on a pressure-volume isotherm by the sign of $(\partial p/\partial V)_T$, which is negative for metastable states and positive for unstable states. The points where $(\partial p/\partial V)_T = 0$ define the spinodes. The stable liquid and stable gas are in equilibrium with each other at a specific pressure and corresponding temperature. Locating the equilibrium points for each temperature up to the critical temperature, it is possible to construct the coexistence curve which defines the thermodynamic two-phase region inside this curve. The spinodal curve can also be constructed by connecting the spinodes for each temperature. On the isotherm, the gas spinode occurs at a higher pressure than the equilibrium pressure, and the liquid spinode occurs at a lower pressure than the equilibrium pressure. Refer to Figure 1. Thus, the isotherms exhibit the familiar van der Waal's loops obtained from the van der Waal's equation of state. In usual undergraduate physical chemistry texts, the loops are not included in the phase diagram. They are replaced by horizontal lines drawn so that the loops define equal areas above and below the lines. This is the Maxwell construction, which gives the vapor pressure of the liquid at the temperature. Thus, according to classical thermodynamics, it is not possible to observe a gas in a metastable state, termed supersaturated vapor because the gas is over-saturated with clusters. Nor is it possible to

have a liquid at a pressure less than the equilibrium pressure, termed supersaturated liquid because the liquid is over-saturated with cavities.

The computational results indicate that gases and liquids can exist in metastable states. Nucleation theory attempts to describe the supersaturated gas state through the use of clusters. Clusters are studied because they are the precursors to liquid formation, which has eluded characterization. Many prominent scientists have formulated cluster theories [1,2,3], but most theories have not agreed with experimental pressures, critical supersaturations, or nucleation rates [4] obtained with cloud chambers, aerosol generators, or supersonic nozzles. The goal of this study is to elaborate on the present theories with the help of Monte Carlo computer simulations. [5]. The present study is interested with the liquid-vapor two-phase region. We are concerned with the formation of liquid-like clusters from the vapor, which indicates a phase transition.

A cluster is a collection of particles in which each one is within a certain distance of its nearest neighbors. The distance has to be arbitrarily defined for each study, and is usually defined according to the taste of the investigator. Because we are interested in the formation of liquids from vapors, we shall use a distance that is comparable to the largest near-neighbor distance in liquids. Many different systems characterized by the interaction potential have been studied with a continuum lattice, also referred to as a random lattice because the particles are randomly located in the system, and seem to exhibit universal behavior. The present study uses a continuum system and incorporates the realistic Lennard-Jones potential and the Stockmayer potential. The liquid-gas coexistence properties for the LJ fluid are well established, and the phase diagram used for the LJ fluid is that of Panagiotopoulos, *et al.* [6]. For the Stockmayer potential, the coexistence properties are those calculated by Smit, *et al.* [7].

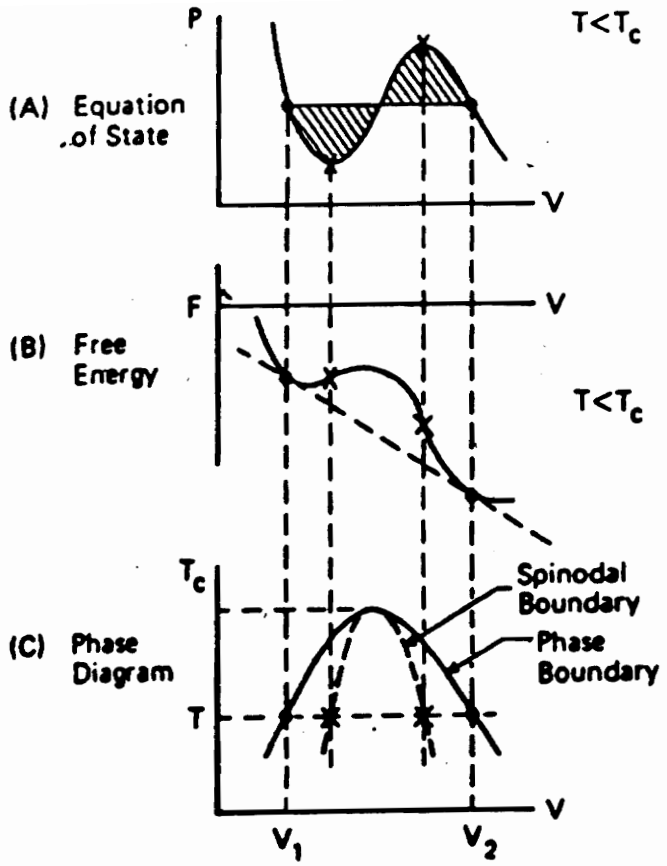


Figure 1. Equation of state, free energy-volume relation, and phase diagram for a "van der Waals-like" fluid. From reference [9].

II. THEORETICAL BACKGROUND

A. NUCLEATION

A homogeneous vapor can exist in a state of supersaturation. Although such a state is thermodynamically metastable, it is prevented from condensing by kinetic effects [8]. Small droplets have enhanced vapor pressures that cause them to reevaporate rather than grow. A cluster must be larger than some minimum critical size before it can grow spontaneously. The supersaturated vapor is of uniform density (single-phase) and exists in the two-phase region of the phase diagram, where coexisting liquid and vapor are the thermodynamically stable phases. Thus, there is a state of lower free energy for the metastable supersaturated vapor -- the stable two-phase state of liquid and vapor [9].

Isotherms for fluids exhibit van der Waals loops [9]; as the volume is increased through the two phase region, the isotherms possess minima and maxima that occur at $(\partial p/\partial V)_T = 0$. In principle, the homogeneous supersaturated vapor can be maintained up to the maximum in the isotherm, the spinode, without the appearance of the liquid phase. The spinodal is the line going through the minimum and the maximum points of the isotherms and represents the limit of meta-stability of a homogeneous phase [10]. Refer to Figure 1. The area between the saturated vapor line and the spinodal is the metastable supersaturated vapor region. Between the spinodals the system is unstable. Experimentally, a metastable state can be obtained by isothermal compression, isobaric cooling, or isochoric cooling. In actual processes, condensation, separation of the supersaturated vapor into liquid and vapor, is usually observed before the spinodal is reached due to random fluctuations in the homogeneity of the supersaturated vapor phase [4].

A cluster is formed when monomers come together and act as one entity.

Condensation occurs when the number of particles (g) in the cluster is above some critical nucleus number, g^*. Below g^* the cluster can gain and lose particles freely. Once g^* is reached, the cluster can grow spontaneously [4]. In other words, the cluster grows until the appropriate amount of liquid and vapor is obtained, corresponding to phase separation. This would be appropriate in an NVT ensemble, however, in the NpT ensemble the density is a changing variable that enables the system to completely change phase -- two phases are not simultaneously observable.

Classical and statistical nucleation theory have been described in many review articles [2,4,11]. Classical nucleation theory describes nucleation from a thermodynamic point of view, usually incorporating the macroscopic surface tension and assuming stationary clusters. Statistical nucleation theory uses partition functions to describe the nucleation process, making use of the rotational and translational partition functions to describe the motion of the clusters.

In classical nucleation theory, the most important assumption is that each embryo (cluster) is a liquid drop of spherical shape. This notion permits the use of the capillarity approximation, which allows the free energy of the droplet surface to be expressed as $4\pi r^2\gamma$, where γ is the surface tension and r is the radius of the spherical droplet. The surface tension is assumed to be independent of the number of particles in the cluster. The number of clusters that contain g particles is represented by N_g. The cluster distribution is derived on the basis that the Gibbs free energy is a minimum for equilibrium at constant pressure and temperature. Springer's [4] equation for the difference in the Gibbs free energy between the cluster-vapor system and the pure vapor system at the same temperature T and pressure p is

$$\Delta G = N_1\mu_V + \sum_g N_g(\mu_L g + 4\pi r^2\gamma) + \left[k_B T\left(N_1 \ln\frac{N_1}{F} + \sum_g N_g \ln\frac{N_g}{F}\right)\right] - N\mu_V . \quad (2.1)$$

where μ_V and μ_L are the chemical potentials in the vapor and liquid phases, N is the total number of particles ($N = \sum_g gN_g$) and F is the total number of clusters (including monomers) in the system ($F = \sum_g N_g$). Equation (2.1) can be written for the change in free energy due to the formation of a single cluster from the vapor as

$$\Delta G = N_1\mu_V + g\mu_L + 4\pi r^2\gamma - N\mu_V . \quad (2.2)$$

Using the requirement of thermodynamic equilibrium for a system that changes phase, $(dG)_{p,T} = 0$, and the constraint of constant number of particles, the Gibbs free energy of formation of a cluster from the vapor in equilibrium with the surrounding vapor is

$$\Delta G = -\frac{4}{3}\pi r^3\left(\frac{k_B T}{v_L}\right)\ln S + 4\pi r^2\gamma, \quad (2.3)$$

in which v_L is the molecular volume in the liquid phase, and the supersaturation ratio, S, is defined as

$$S = \frac{p(T)}{p_{eq}(T)} \quad (2.4)$$

where $p(T)$ is the vapor pressure on the isotherm and $p_{eq}(T)$ is the vapor pressure of the liquid at the same temperature. Equation (2.3) can be written using cluster number (g defined as the number of particles in the cluster) instead of cluster size (r defined as the radius of the cluster) through $g = 4\pi r^3/3v_L$ as

$$\Delta G = -gk_B T \ln S + (4\pi)^{1/3}(3g v_L)^{2/3}\gamma . \quad (2.5)$$

This equation has a negative contribution from the free energy difference between the bulk liquid and the bulk vapor and a positive contribution from the surface free energy. Since ΔG of a cluster has a maximum at the critical nucleus radius (r^*), the critical nucleus represents a metastable equilibrium state. The free energy of a cluster as a function of supersaturation ratio, S, illustrates the large effect of the supersaturation on g^* and r^*. As S increases from saturation ($S = 1$) to supersaturation ($S > 1$), the critical nucleus size decreases as the inverse of the $\ln S$. This means that the number of particles required to form the critical nucleus is smaller the further the system is from the saturated vapor line toward the liquid state. In other words, the maximum in ΔG is shifted to smaller g as the system becomes more supersaturated.

Springer [4] obtained the equilibrium cluster distribution, n_g, by minimizing the Gibbs free energy subject to the constraints of constant N and F. The equilibrium number of g-mers can be expressed as

$$n_g = N \exp\left\{ g \ln S - \frac{(4\pi)^{1/3}}{k_B T}\left(\frac{3g}{\rho_L}\right)^{2/3}\gamma \right\} \quad . \tag{2.6}$$

B. CAVITATION THEORY

Just as nucleation theory describes the route from a gas to a liquid, cavitation theory describes the liquid to gas transition. Nucleation experiments have paid much less attention to cavity formation than to droplet formation. To form the bubbles, the pressure of the liquid must be decreased such that the bubble has a high vapor pressure, which complies with the Laplace equation: $p_V = p_L + 2\gamma/r$, where γ is the surface tension and r is the radius of the cavity [8,12]. This causes $\Delta p = p_L - p_V$ to be negative, which corresponds to applying a tensile force on the liquid. Cavities in liquids are at equilibrium when the tendency for their surface area to decrease is balanced by the rise of internal pressure which would then result. The vapor pressure in the bubbles must exceed p_L by $2\gamma/r$ if they are to grow. Once the vapor pressure is sufficient to maintain the bubble and the bubble grows, the vapor pressure can decrease because r is increasing. This spontaneous growth makes the liquid explode.

The theory just presented is dependent on the capillarity approximation, and thus has problems similar to nucleation theory. In our computer simulations, it is easy to locate groups of particles that form clusters and characterize them. However, it is very difficult to locate empty spaces that form cavities. Therefore, there is a lot more information on the gas to liquid phase transition than there is on the reverse.

C. Scaling Laws and Percolation Theory

Observed universal behavior between systems on different lattices is described through critical exponents and scaling laws. A critical point, ρ_c, called the percolation threshold density, exists such that for ρ below ρ_c all clusters are finite, whereas for ρ above ρ_c one infinite cluster is found in addition to many finite clusters. This infinite cluster is said to be percolating; it spans the entire system. Thus, the percolation threshold density indicates a phase transition since a percolating network exists for $\rho > \rho_c$ and no percolating network exists for $\rho < \rho_c$. The region ρ close to ρ_c is called the scaling region because this is where the critical behavior of percolation theory, represented by scaling laws, is applied to percolating clusters [13]. The behavior of systems close to a phase transition is described by the critical exponents. The percolation problem serves as an illustration of the scaling laws of phase transitions and critical phenomena. There are seven critical exponents of which two are independent [13]. The exponents can be defined by various moments of the cluster distribution. For example,

$$\sum_g n_g(\rho) \propto |\rho - \rho_c|^{2-\alpha} \tag{2.7}$$

and

$$\sum_g g n_g(\rho) \propto (\rho - \rho_c)^{\beta}. \tag{2.8}$$

These exponents were derived from studies of Ising spin models and lattice gases, which are computationally easy systems relative to realistic gases and liquids.

There are two additional scaling laws and two exponents that are applied to clusters. We are only concerned with the cluster distribution scaling law. This scaling law, which is given by Bauchspiess and Stauffer [14], is based on a surface term and a correction. Since the cluster distribution is usually written in the form, $n_g \propto \exp(-\Delta G/k_B T)$ and the free energy

of cluster formation is mainly a surface term, $4\pi r^2\gamma$, the cluster number distribution can be represented by

$$n_g \propto g^{-\tau}\exp(-const \cdot g^{\zeta}), \tag{2.9}$$

where the exponential term represents the surface and the power law is considered a correction. According to Bauchspiess and Stauffer [14], the surface term has no effect on the cluster distribution at ρ_c and can be neglected, leaving only the term involving the power law [14], as predicted by the Fisher droplet model [15]. For $\rho \neq \rho_c$ the cluster numbers decay exponentially and the power law is not needed [14]. In d dimensions $\zeta = 1 - 1/d$ when $\rho > \rho_c$, and $\zeta = 1$ when $\rho < \rho_c$. In practice, researchers [16,17] using highly dissimilar systems employ the simple power law form of the cluster distribution scaling law:

$$n_g \propto g^{-\tau} \tag{2.10}$$

It is generally accepted that τ is a universal exponent with value 2.2 [18]. In our analysis of the simulated cluster distribution, we found that the simple scaling law is the principal component and the exponential factor is small [19].

III. PROCEDURE

A. Simulation Technique

1. Metropolis Monte Carlo Algorithm

The original Metropolis [20] Monte Carlo algorithm allows molecular systems to be simulated by a computer. The goal is to sample all phase space in a chosen statistical ensemble. In our simulations, we use the isothermal-isobaric, constant-NpT, ensemble. In theory, all configurations can be sampled by repeatedly, randomly placing the particles in the ensemble and calculating the Boltzmann factor for the potential energy, $\mathcal{V}(t)$, for each configuration, t. After a very large number of trials, $t_{\max}$, the configurational integral can be estimated using

$$Z_{NVT} \approx \frac{V^N}{t_{\max}} \sum_{t=1}^{t_{\max}} \exp(-\beta \mathcal{V}(t)) . \tag{3.1}$$

A reasonable minimum $t_{\max}$ value is 10^{3N}, which corresponds to 10 function evaluations for each independent coordinate. It is not feasible to generate this many configurations for a system of, say, 500 particles.

This problem was greatly simplified with use of the importance sampling technique [5]. Random configurations of the system are chosen from a distribution which allows the function evaluation to be concentrated in the regions of space that make important contributions to the configurational integral. A Markov chain samples these important regions. A Markov chain of sequential states is set up so the outcome of each trial belongs to a finite set of outcomes and depends only on the state that immediately precedes it. It is essential that the Markov chain samples a representative portion of phase space in a reasonable number of moves. To get from the present state to the next state, an atom is displaced from its position with equal probability to any point inside a cube. This is done

by picking a uniform random displacement along each of the coordinate axes. The maximum displacement, or step size, is an adjustable parameter that governs the size of the cube and controls the convergence of the Markov chain. The acceptance of the trial move depends on the relative probabilities of the initial state, m , and the final state, n . The move is accepted if $\Delta\mathcal{V}_{n\,m} = \mathcal{V}_n - \mathcal{V}_m$ is negative since the probability of the new state is greater than that of the previous state. If $\Delta\mathcal{V}_{n\,m}$ is positive, then the move is accepted with a probability of the Boltzmann factor, $\exp(-\beta\,\Delta\mathcal{V}_{n\,m})$. If a randomly chosen number between zero and one is less than the Boltzmann factor, the move is accepted, otherwise the move is rejected. If the move is discarded, the system remains in the previous state, and it is counted as a new configuration in the Markov chain. There is also a volume displacement associated with the NpT ensemble. A volume is perturbed by randomly selecting a displacement within a maximum value. The new energy is calculated and accepted with the probability $\min\{1, \exp(-\beta B)\}$, where

$$B = \Delta\mathcal{V}_{nm} + p\Delta V - NT\ln(V_n/V_m) \tag{3.2}$$

and $\Delta V = V_n - V_m$. The min function selects the smaller value. This procedure is designed to take the system from a state to any of its neighboring states with equal probability. The NpT ensemble is suitable for the present calculations because it prefers to be homogeneous. Also, the problem of calculating the pressure in the ensemble by various approximations is nonexistent. The pressure is a constant parameter. The density is calculated from the number of particles and the volume, which changes throughout the simulation.

When the Markov chain converges the system is at equilibrium. This way equilibrium is obtained by minimizing the free energy according to a Boltzmann distribution of potential energy states. The kinetic energy need not be included in the Monte Carlo simulation. In the model used, the energy is expressible as a sum of kinetic

($\mathbf{p}$-dependent) and potential ($\mathbf{q}$-dependent) contributions, so the classical partition function separates into a product of kinetic and potential parts.

$$Q_{NVT} = (N!h^{3N})^{-1} \int d\mathbf{p} \exp(-\mathcal{K}/k_B T) \int d\mathbf{q} \exp(-\mathcal{V}/k_B T) \tag{3.3}$$

The average value of any quantity, A, which is only dependent on the system configuration, is

$$\langle A \rangle = \frac{(N!h^{3N})^{-1} \int d\mathbf{q} A \exp(-\mathcal{V}/k_B T) \int d\mathbf{p} \exp(-\mathcal{K}/k_B T)}{Q_{NVT}}. \tag{3.4}$$

Since the kinetic factors cancel, the kinetic energy does not contribute to the average value of a structural quantity, such as the cluster distribution, and does not need to be taken into consideration in the Monte Carlo simulation. All other quantities, thermodynamic and statistical, must be determined by ensemble averaging and, for any particular state point, the instantaneous values of the appropriate phase function will deviate from this average value, illustrating the non-static character of equilibrium.

The potential energy function is chosen to represent the interactions between the molecules of interest. The form of the potential energy function is crucial to the validity of the simulation results. For example, if the simulation was for water, the results would reflect the accuracy of the empirical potential energy function used to represent the water-water interaction.

2. Lennard-Jones Potential Energy Function

We employ the Lennard-Jones [21] potential energy function in our calculations. The pair-wise function has the following form:

$$U^{LJ}(r_{ij}) = 4\varepsilon\left\{\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right\}, \tag{3.5}$$

so that the potential energy of the ensemble is

$$\mathcal{V}=\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} U^{\mathrm{LJ}}(r_{ij})\,. \tag{3.6}$$

No many-body terms are included in the ensemble potential energy. The LJ function rises sharply as the particles come within a certain separation, $r < \sigma$. Here, the $(1/r_{ij})^{12}$ repulsion term completely dominates the $(1/r_{ij})^{6}$ attractive term, as shown in Figure 2. It is not a square-well since the attractive term is continuous. Both terms approach zero at large separation. In the potential, σ is the separation at which $U^{\mathrm{LJ}} = 0$, ε corresponds to the depth of the well, and r_{ij} is the internuclear separation between particles i and j. The parameters, σ and ε, are well established for many molecules. The Lennard-Jones potential energy function best simulates spherically symmetric molecules, such as atoms and methane, although the LJ parameters have been determined for diatomics and hydrocarbons.

All quantities in the present study are in the usual reduced units so as not to specify the constants for any particular molecule. The reduced variables are in terms of the Lennard-Jones parameters, σ and ε. For example, reduced number density ($\rho^{*} = \rho\sigma^{3}$), reduced temperature ($T^{*} = k_{\mathrm{B}}T/\varepsilon$), reduced separation ($r_{ij}^{*} = r_{ij}/\sigma$), reduced surface tension ($\gamma^{*} = \gamma\sigma^{2}/\varepsilon$) and reduced energy ($\mathcal{V}^{*} = \mathcal{V}/\varepsilon$) are in terms of the LJ parameters. The conventional way to represent reduced variables is by a superscripted asterisk.

The pressure can be calculated in the simulation run as an ensemble average. It is calculated using the virial theorem expressed as a form of generalized equipartition: [5]

$$\langle \mathbf{q}_k \partial\mathcal{H}/\partial\mathbf{q}_k \rangle = k_{\mathrm{B}}T\,, \tag{3.7}$$

where $\mathcal{H} = \mathcal{K} + \mathcal{V}$. Each coordinate derivative is the negative of a component of the force on the particle. So, the formula for the pressure calculation is

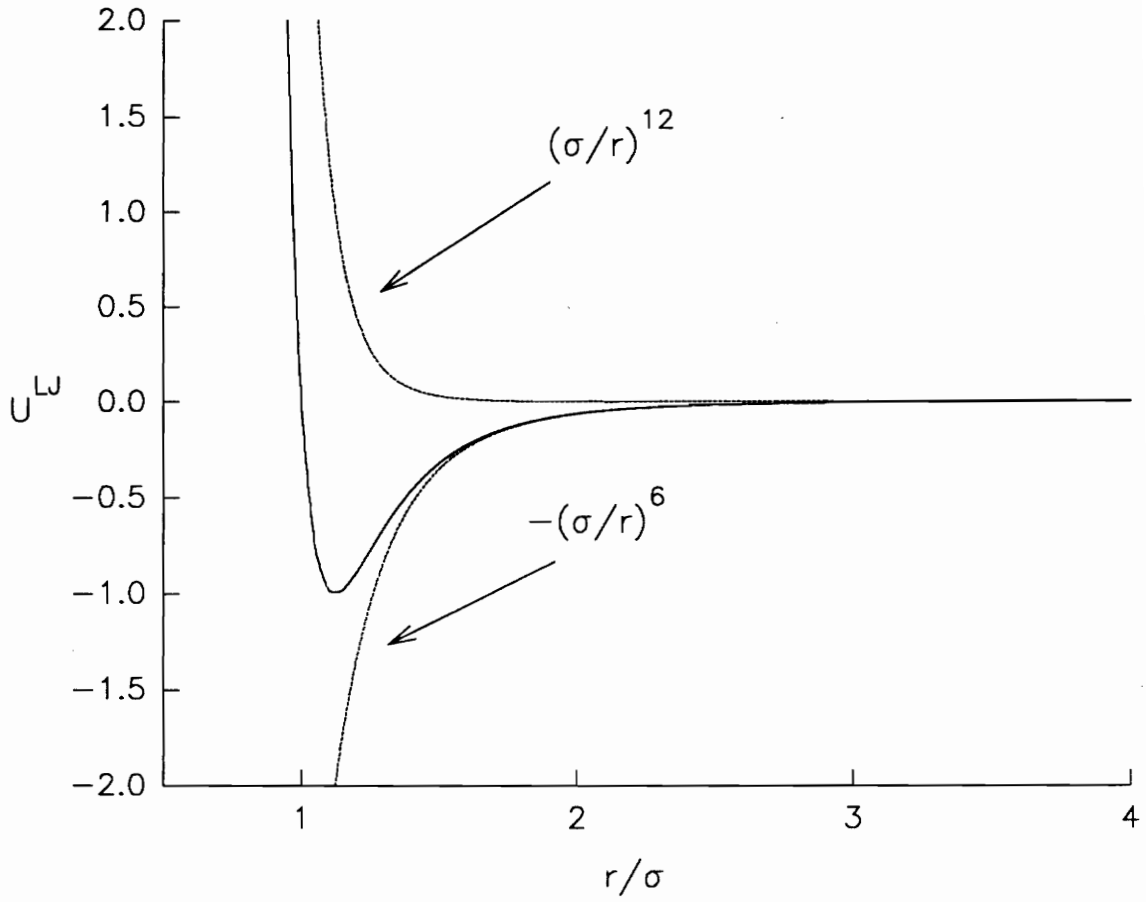


Figure 2. Lennard-Jones potential energy function.

$$pV = Nk_B T - \frac{1}{3}\left\langle \sum_i \sum_{j>i} r_{ij} \frac{\partial U(r_{ij})}{\partial r_{ij}} \right\rangle. \tag{3.8}$$

The derivative is expressed analytically since we are using potential energy functions. Thus, the pressure is expressed as

$$p^* V^* = NT^* - \frac{1}{3}\left\langle \sum_{i}^{N-1} \sum_{i>j}^{N} 4\left\{ -12(r_{ij}^*)^{-12} + 6(r_{ij}^*)^{-6} \right\} \right\rangle \tag{3.9}$$

in reduced variables for the LJ potential. Since the pressure is specified in the calculation, this is a nice test of the program and of the virial pressure calculation. Calculations have shown that the virial pressure method works well for the determination of equilibrium gas and liquid pressures. The method fails for supersaturated gases and supersaturated liquids. The error increases the closer the system is to the spinodal point.

3. Stockmayer Potential Energy Function

For two spheres, i and j, with point dipoles at their centers of magnitude μ_i and μ_j, the potential energy function is a superposition of the LJ function and the dipole-dipole interaction. The interaction between two ideal point dipoles can be written as

$$U^{\text{dip}}(r_{ij}, \mu_i, \mu_j) = \frac{1}{4\pi\varepsilon_o r_{ij}^3}\left\{ (\vec{\mu}_i \bullet \vec{\mu}_j) - \frac{3(\vec{\mu}_i \bullet \vec{r}_{ij})(\vec{\mu}_j \bullet \vec{r}_{ij})}{r_{ij}^2} \right\} \tag{3.10}$$

where ε_o is the vacuum permittivity. Thus, the Stockmayer [22] potential energy function is

$$U^{\text{SM}} = U^{\text{LJ}} + U^{\text{dip}} \tag{3.11}$$

It describes the interaction between polar molecules with negligible dipole-quadrupole and higher multipole interactions [23]. The reduced dipole moment is defined as $\mu^* = \mu(4\pi\varepsilon_o \varepsilon\sigma^3)^{-1/2}$. The Stockmayer potential energy function is shown in Figure 3 for

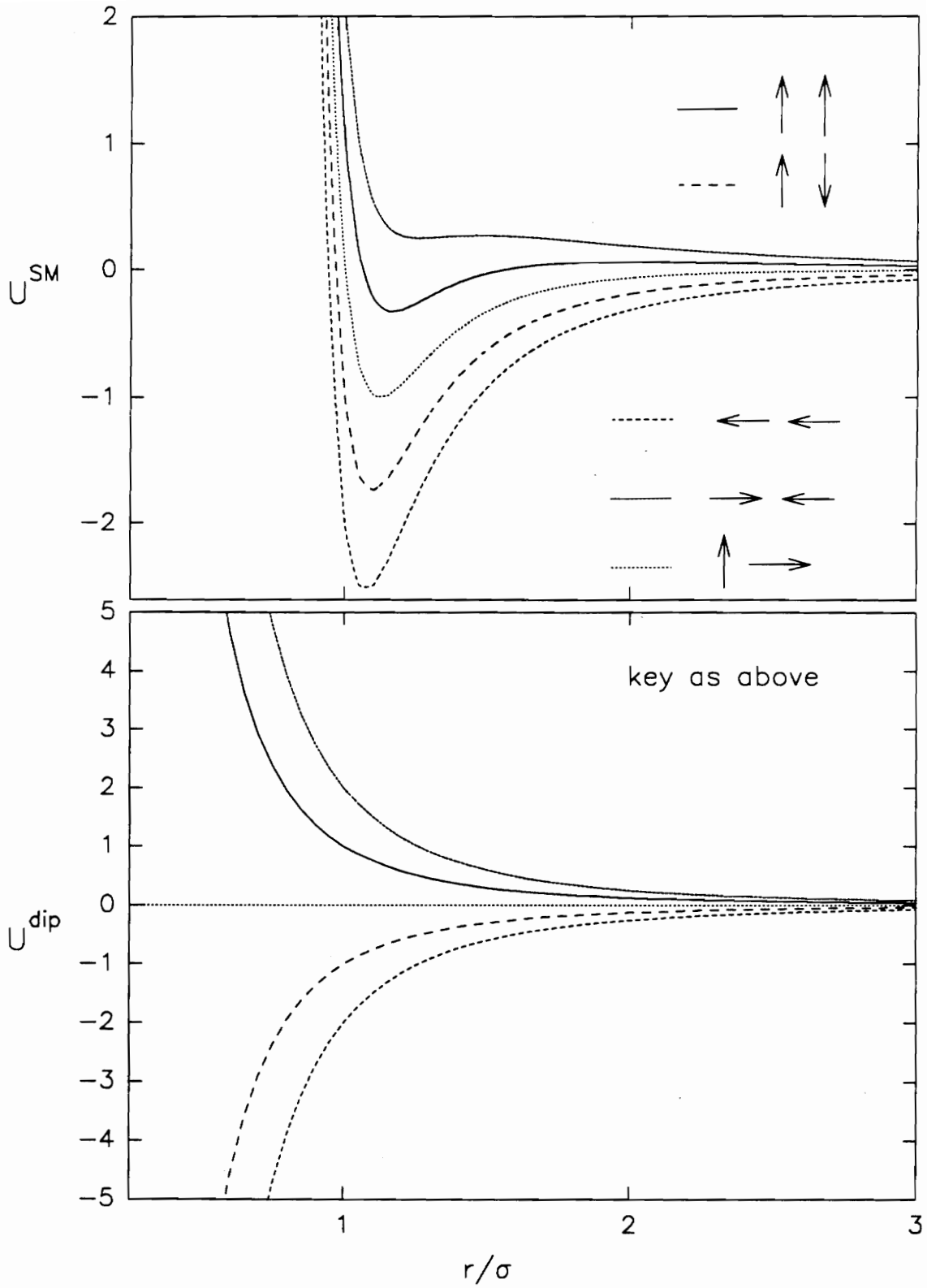


Figure 3. Stockmayer potential energy function for different orientations of the dipole moment.

several orientations of the dipole moment. One dipole was held at the origin while the other was moved along the axis. At small distance the dipole-dipole interaction is dominated by the LJ interaction.

4. Definition of a Cluster

There are many ways to define a cluster. The RKC (Reiss, Katz, Cohen) [24] cluster consists of a group of molecules contained within a spherical shell whose center is fixed on the center of mass of the group. This definition assumes a stable equilibrium within the cluster, but no natural choice for the shell radius. McGinty [25] introduced a cluster similar to the RKC cluster, only the constraining shell is replaced by the requirement that the net rate of evaporation of the cluster into the supersaturated vapor should vanish. Stillinger [26] defined yet another cluster based on physical grounds. An arbitrary maximum separation, c is used to determine if two neighboring particles are in the same cluster. To be part of a cluster every particle must be closer than c to at least one other member of the cluster. Depending on the choice of c , the cluster can have interactions with particles that are not part of the cluster. This is called an excluded volume effect and is the reason Reiss [27] does not like this cluster definition. In the present study, Stillinger's cluster definition is used with an appropriate choice of c , called the cluster criterion. Since is our intention to study coexistence properties between gases and liquids, the value of c was chosen as the distance where the first minimum occurs in the radial distribution function of the LJ liquid. The radial distribution function for the liquid is shown in Figure 4 for several temperatures. Based on this figure, we chose $c = 1.5\sigma$. Unlike the RKC and McGinty cluster, our cluster is allowed to grow and evaporate, for our Monte Carlo simulation does not treat clusters explicitly, it just counts them. It is our opinion that this choice of a cluster definition is physically consistent with our modeling goals.

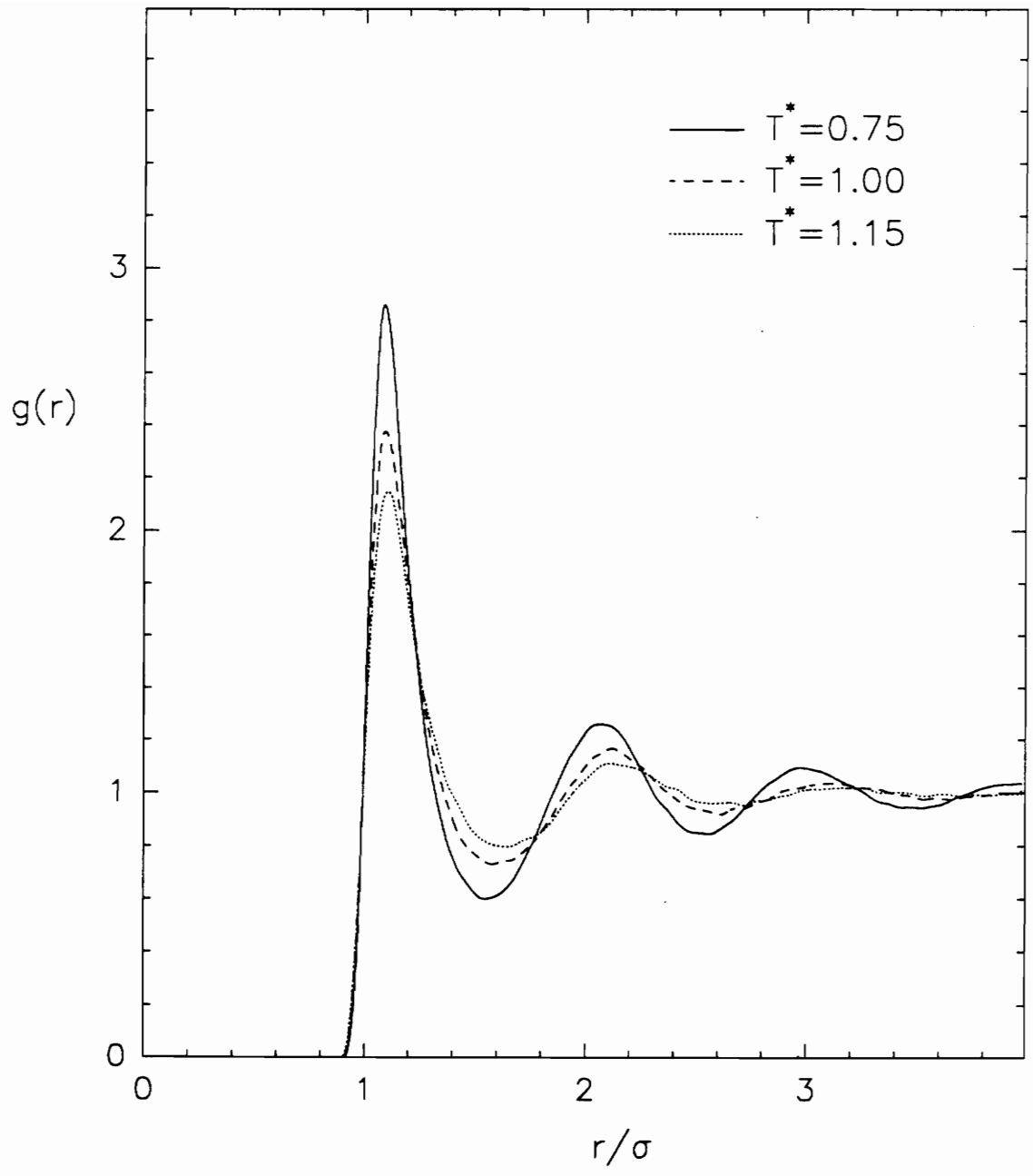


Figure 4. Radial distribution function for the Lennard-Jones liquid. Liquid densities used in simulation from reference [6].

5. Periodic Boundary Conditions

The system uses periodic boundary conditions to overcome the problem of surface effects. The system is represented by a central cubic box, which is replicated throughout space to form an infinite lattice. During the simulation, as a particle moves in the original box, its periodic image moves in exactly the same way in each of the neighboring boxes. If a molecule leaves the central box, one of its images will enter through the opposite face. See Figure 5, which is an example of two dimensional periodicity. There are no walls at the boundary of the central box, and no surface molecules. As the density approaches the vapor, the importance of periodic boundary conditions is reduced. The LJ potential is cut off after $L^*/2$, where L^* is the reduced length of a side of the cube. This means that the interaction of particles separated by a length greater than $L^*/2$ is zero. At most, there can be only one distance less than $L^*/2$ between a particle and the periodic images of another particle. The cutoff distance is scaled to the volume changes in the NpT simulation.

The simulation of larger systems, say greater than 1000 particles, can be carried out very efficiently by utilizing the cell index method [28,29]. The cube is divided into sub-cubes. Instead of calculating the distance between one particle and every other particle in the system, this method only calculates the distance between particles in the same cube and adjacent cubes. When the cell index method is used, the potential is cut off at a distance R_c, which is not necessarily equal to $L^*/2$. The number of sub-cubes is determined by R_c as $N_{\text{cells}} = [\text{int}\{(N/density)^{1/3}/R_c\}]^3$, where int$\{x\}$ means the integral part of x. The effectiveness of the method increases greatly with smaller cutoff size. However, one needs to be careful that the cutoff distance used has a minimal effect on the simulation properties. Simulations were carried out using a smaller cutoff distance than is normally employed, half the box edge length, in order to make more efficient use of the cell index method. This

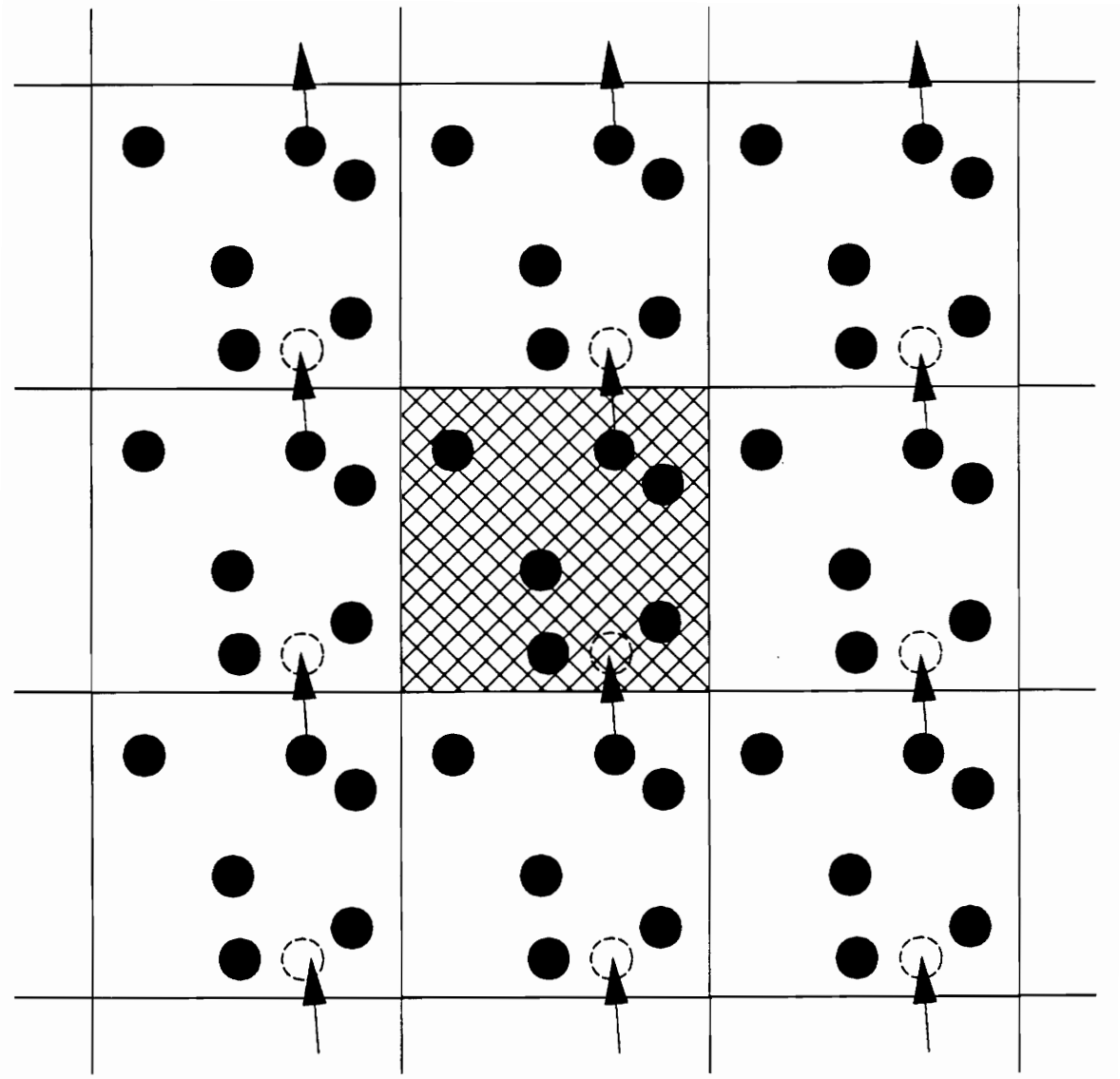


Figure 5. A 2-D example of periodic boundary conditions.

resulted in different properties between the two systems with different cutoff lengths. There usually exists a trade-off between efficiency and obtaining correct results. If essentially equal cutoff distances are used with small and large system sizes, the cell index method can be used with the larger system effectively.

Operationally, two arrays are assigned, LIST and HEAD. HEAD is dimensioned to the number of cells while the dimension of LIST is N. The particles are cycled through and assigned a cell. The particles that are in each cell is maintained by HEAD and LIST. The following example illustrates this point:

There are 10 particles: $N = 10$

There are 4 cells: $N_{cells} = 4$

	1	2	3	4	5	6	7	8	9	10	
LIST (	0	1	0	0	3	2	0	4	6	5	)

	1	2	3	4	
HEAD (	10	7	9	8	)

To extract the cell information, follow the pointers. For example, using cell one, HEAD(1)=10. This points to LIST(10), which means that particle 10 is in cell one. LIST(10)=5, which points to LIST(5); particle 5 is in cell one. LIST(5)=3; particle 3 is in cell one. LIST(3)=0, which means that the particles in cell one are 10, 5 and 3. Likewise, the particles in cell 3 are 9, 6, 2 and 1. This is a very efficient method to search by computer through the cells for interactions. It is called a linked list [29].

6. Percolation Algorithm

To study percolation, periodic boundary conditions must be used. Above a certain density, the percolation threshold, there is a finite probability, P , that a percolating cluster will exist. It is generally accepted that this percolation probability is one-half at the

percolation threshold [17]. In other words, half of the configurations will percolate at ρ_c^*. $P(\rho_c^*) = 0.50$. This value is chosen because the percolation threshold density exhibits small system size effects at this point [17]. In order to find ρ_c^* for a particular temperature, the density is incrementally increased, starting with a low density, to determine the percolation probability for each density. A method has to be used to determine if a configuration is percolating.

The operational definition of a percolating cluster is that it must be possible to start at a particle and, by moving through its neighbors, reach an image of the particle in another replica of the system. If this can be done the cluster is percolating. The percolation test is conducted in the accumulation of the cluster distribution. As a cluster is formed, the algorithm keeps track of the cube image where a neighbor is found. If a neighbor has been seen before and in a different cube, a cycle has been found, and the configuration is percolating. This method is very efficient because it makes a queue list of particles that need to be checked and divides the cube into sub-cubes that are slightly larger than the cluster criterion length. This way there are a minimum number of distances that need to be calculated.

There are other ways of indirectly calculating ρ_c^* based on the cluster distributions and scaling laws. Seaton and Glandt [17] apply finite-size scaling theory to estimate the percolation threshold density for a macroscopic system. The percolation transition becomes sharper as the size of the system increases, indicated by the slope of the P vs. ρ^* plot close to ρ_c^* [16]. Thus, a very small change in the density would result in a large change in P. This accounts for the difficulty in precisely locating ρ_c^*. For the macroscopic system, the slope would be infinite. According to the theory [13], the relationship between

the percolation point for various size systems, characterized by the length (L^*) of the side of the cube, and the percolation point of the macroscopic system is

$$\rho^*(P = 0.50) - \rho_c^\infty \propto (L^*)^{-1/\nu}, \tag{3.12}$$

where ν is the correlation length exponent and ρ_c^∞ is the reduced percolation threshold density for the infinite system. The exponent, ν, is universal and independent of the nature of the system. Its accepted value is 0.88 [13]. Thus, the macroscopic percolation threshold density can be determined as the intercept of a plot of $\rho^*(P = 0.50)$ vs. $(L^*)^{-1/\nu}$ for different sized systems. The problem of locating ρ_c^* accurately for each system size remains.

Locating the percolation point for a system can be done without actually calculating the percolation probability. Usually, this is accomplished by measuring some form of the average cluster size. Sevick *et al.* [30] define the mean cluster size, η, from the cluster distribution as

$$\eta = \frac{\sum_g g^2 n_g}{\sum_g g n_g}. \tag{3.13}$$

Of course, η should approach N as the percolation threshold density is approached for a finite size system. However, the mean cluster size should be infinite for an infinite system at ρ_c^*. In order to find the mean cluster size for the infinite system at each density, a plot of $1/\eta$ against $1/N$ gives the inverse average cluster size for the infinite system, η_∞^{-1}, as the intercept since $1/N \to 0$ approaches the infinite system. This should account for most of the finite size effects. Then, plotting η_∞^{-1} for each density versus density and extrapolating down to $\eta_\infty^{-1} = 0$ gives the percolation threshold density for the infinite system.

Another procedure for finding the percolation threshold density is proposed by Hoshen and Kopelman [31]. Their reduced average cluster size is defined as

$$\eta' = N^{-1} \sum_{g=1}^{g_{max}} g^2 n_g - N^{-1} g_{max}^2 \tag{3.14}$$

where g_{max} is the size of the largest cluster. Below ρ_c^*, η' increases with concentration. Above the critical density the largest cluster grows rapidly, leading to a sharp decline in the value of η'. By plotting η' versus ρ^*, a sharp maximum should occur in the region of the critical percolation density. Thus, there are alternate ways of finding the percolation threshold density without directly calculating the percolation probability in the computer simulation.

7. Equilibrium

The simulation is begun by filling the simulation box with particles. They can be arranged in a number of ways. For example, the particles can be placed on a simple cubic lattice, on a face centered lattice, or on a random lattice. It is our experience that equilibrium is obtained much faster by placing particles on a lattice than by placing them at random in the cube because there is a distinct possibility of the hard spheres overlapping and of local high density areas. The maximum step size is adjusted to ensure an acceptance ratio of 20-40 percent for particle displacement. Most researchers suggest a 50% acceptance ratio, but it is not clear that this is an optimum value [5]. At the start of a simulation, the step size is initialized to be one-third the length of the side of the cube. If the density is approaching that of a liquid, the step size will decrease to approximately half σ, when requiring 40% acceptance. Thus, the acceptance rate is decreased to 20% for dense systems to insure ensemble sampling. The simulation can become trapped in deep potential energy wells if the step size is not large enough to facilitate escaping. Here the

phase space of the system is explored slowly, and consecutive states are highly correlated. There is also little movement through phase space if the maximum displacement is too large because nearly all the trial moves are rejected. Thus, there is an optimum value for the acceptance ratio, which depends on the density. The maximum step size for the volume moves is also adjusted throughout the simulation to achieve a 30 percent acceptance of a new volume. The displacement and volume step sizes are adjusted independently.

The particles are cycled through or chosen randomly until apparent equilibrium is reached. We have tried several techniques to help us reach equilibrium rapidly. One such operation was to do cluster moves. These moves helped reach equilibrium faster, but calculating the cluster statistics and performing this more complicated move offset the savings in iterations; the CPU time was about equal to not doing cluster moves. Cluster moves become less effective as equilibrium is reached in the system.

By looking at energy averages over Wood's blocks [32], and comparing them to the running average energy, we are able to determine if the system is at equilibrium. The total number of steps is broken up into M_q blocks of q steps each. These are the Wood's blocks. The over-all average of some property, f, is

$$\langle f \rangle = \frac{1}{M_q} \sum_{s=1}^{M_q} f_s \tag{3.15}$$

with

$$f_s = \frac{1}{q} \sum_{t=1}^{q} f(t) \tag{3.16}$$

being the average of a block. A plot of f_s and the running over-all average vs. s is very useful. If the system is at equilibrium, the running average energy should be stable, as shown in Figure 6. Also, the block averages should not have a trend away from the average

value, as in Figure 7, which shows the block average density for a system that undergoes a transition. Once equilibrium is reached, the cluster distribution, percolation probability and density profile data are calculated twice every cycle of N particle trial moves. Through use of many simulation runs with different cluster gathering intervals, it was decided that this is sufficient to determine these quantities, for the particle coordinates do not change rapidly.

Equilibrium involving two bulk phases has been studied using the Gibbs ensemble developed by Panagiotopoulos in which each phase has its own separate 3d periodic cube [33]. There is a problem studying two bulk phases with 3d periodicity. It appears feasible to have two bulk phases in a single cube if the cube were to have 2d periodicity. However, such a study has not been completed.

The NpT ensemble works well for studying bulk phases because the equilibrium density is automatically calculated. In studying the metastable states in the present paper with the Monte Carlo technique, the author realizes that there is a fundamental problem with non-equilibrium systems, such as metastable gases and liquids. The bulk liquid is a state of lower free energy than the metastable gas. Likewise, the bulk gas is a state of lower free energy than the metastable liquid. There is a free energy barrier that must be overcome in order make a transition from a metastable state to a stable state. When the spinode is approached, the barrier disappears. On the basis of experience, it appears that local equilibrium is achieved in the metastable regions.

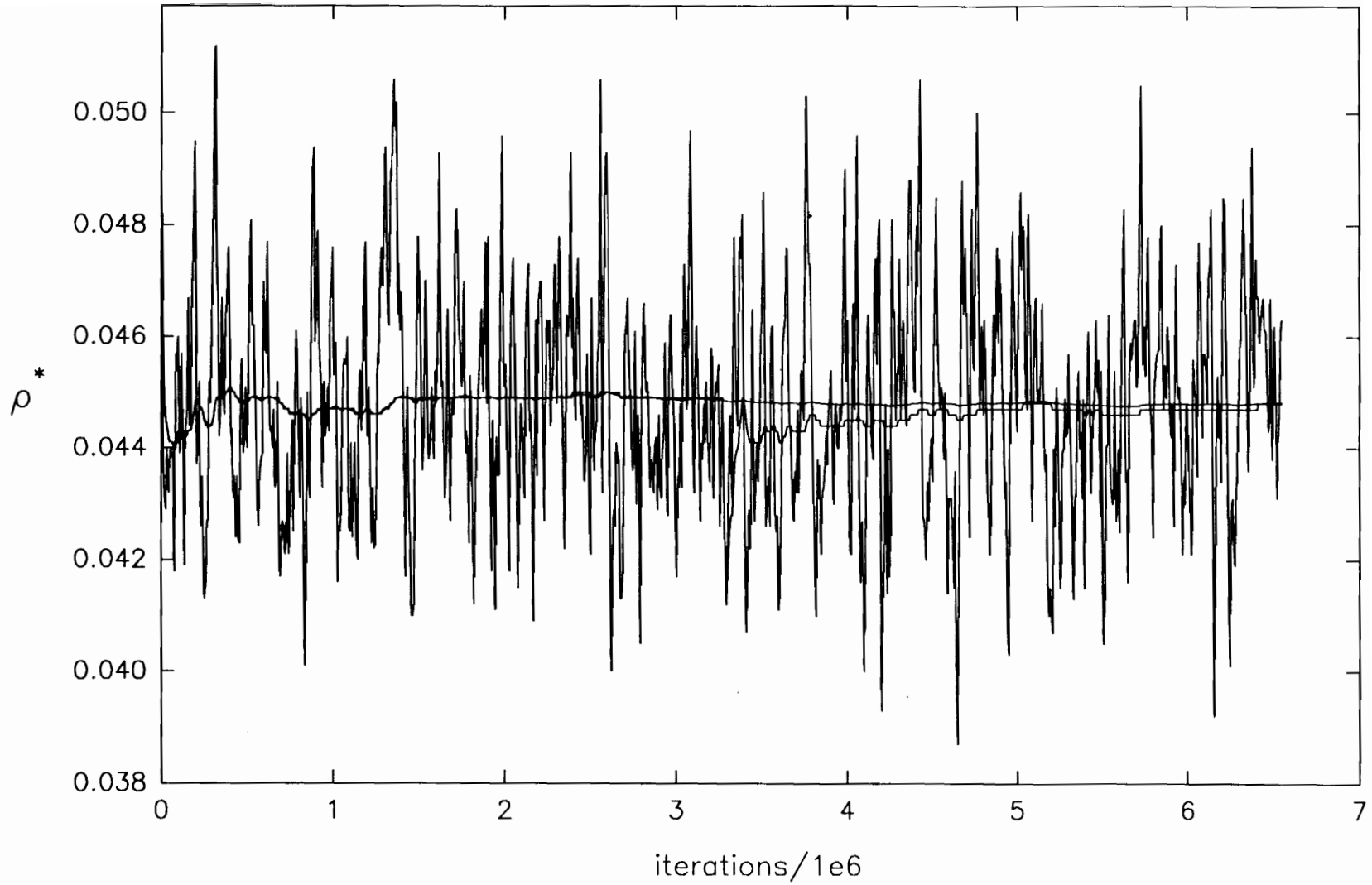


Figure 6. Wood's block type averaging for a stable equilibrated system.

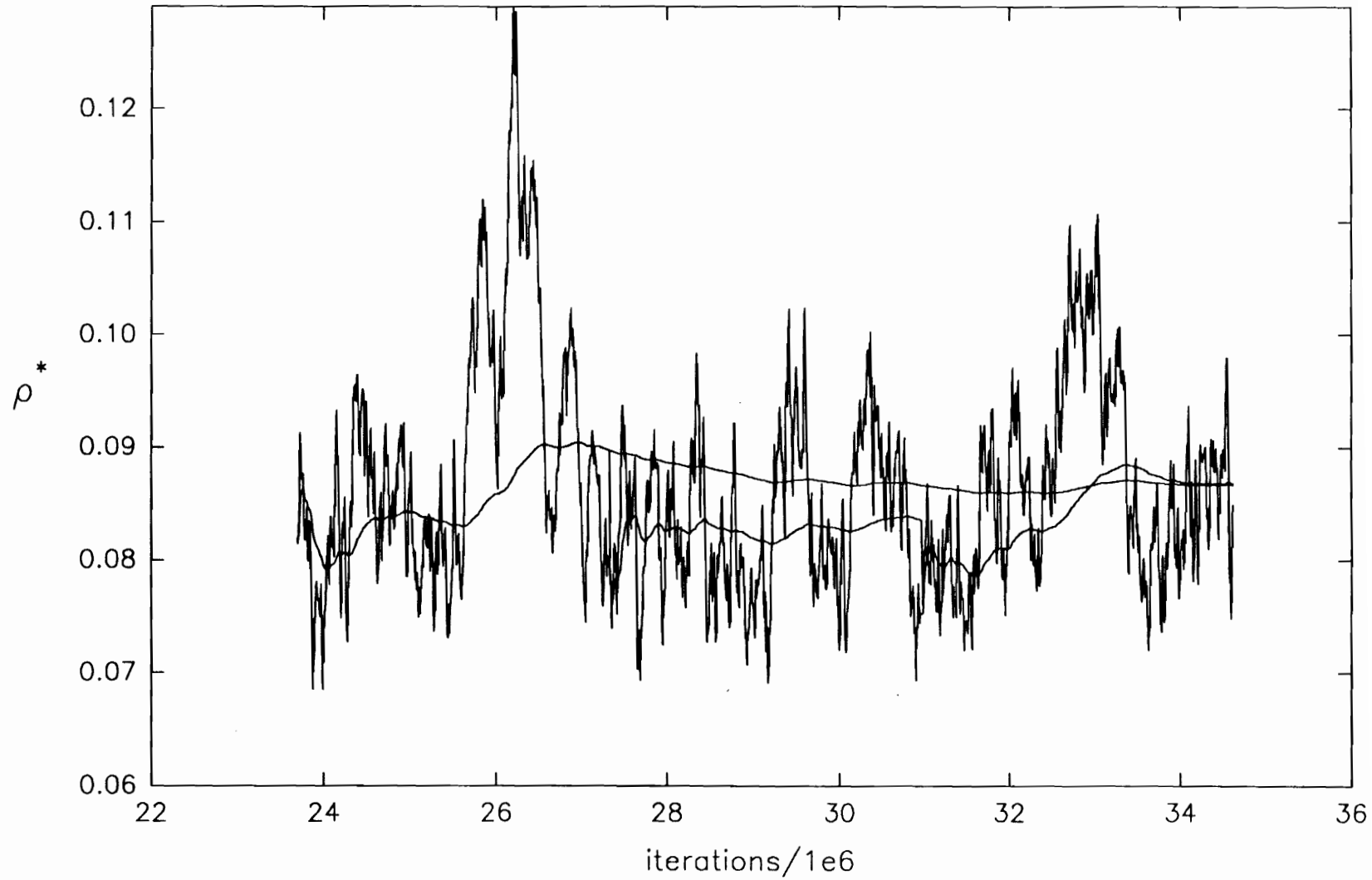


Figure 7. Wood's block type averaging for a system that undergoes a transition.

B. Simulation Details

Unless otherwise stated, our system is made up of 729 particles in a cube of length $L^* = (N/\rho^*)^{1/3}$. Initially, the particles were placed on a simple cubic lattice or the configuration from another pressure, adjusted for the desired pressure, was used. The pressures range from unsaturated vapor to the liquid at the corresponding temperatures. The reduced temperatures studied are 1.00, 1.15, 1.20 and 1.25. The particles are cycled through until equilibrium is reached, usually requiring about 8.6 million steps. The equilibration steps are not included in the averaging of properties, which are usually accumulated in 8.6 million additional steps, although systems that are very close to the spinode usually require greater than 20 or 30 million steps to acquire good statistics. On each isotherm, several pressures between the liquid and the gas density were simulated. In the simulation, the equilibrium number of clusters, n_g, is usually divided by the total number of particles. This makes it possible to compare cluster distributions for different size systems. Ensembles with 512, 1000 and 5000 particles were also investigated to determine finite size effects, if any.

All runs were carried out on an IBM RISC System/6000 350 workstation. With 729 LJ particles 4374000 iterations (6000 cycles) required approximately 7 hours real time to complete. With 729 Stockmayer particles 4374000 iterations (6000 cycles) required approximately 13 hours real time to complete. With 5000 LJ particles using the cell index method with 27 cells, 3000000 iterations required 18 hours of real time to complete. Of course this run time would be much shorter with more cells. These times are for runs that normally received 99.3% of the CPU. The computer code was written in Fortran and is given in Appendix 2.

IV. RESULTS AND DISCUSSION

A. THE LENNARD-JONES FLUID

The accepted critical temperature for a LJ fluid is $T_c^* = 1.35$ [34]. We have simulated isotherms for 729 particles at four temperatures below the critical point, namely $T^* = 1.00, 1.15, 1.20$, and 1.25. The state points determined are shown in Figure 8. The isothermal curves shown in the figure will be discussed in the next section. The horizontal lines are the vapor-liquid tie-lines found by Panagiotopoulos[6] using the Gibbs ensemble. On both sides of the diagram, the isotherms extend past the tie-line and show metastable phases. However, on each isotherm, there is a maximum pressure that the metastable gas can have without transforming into a liquid. There is also a minimum pressure that the metastable liquid can have without transforming into a gas. The maximum and minimum pressures and the corresponding densities are listed in Table 1 for the four isotherms. The supersaturation ratios at the pressure maxima are also given; these were calculated as $S = p^*_{max}/p^*_{eq}$ where p^*_{eq} was taken from Panagiotopoulos [6]. Appendix A.1 lists the calculation details for each isotherm.

Table 1. The spinodal points for the three-dimensional Lennard-Jones fluid. S is the supersaturation ratio at the gas spinodal. The pressures are accurate to the last decimal place given. The numbers in parentheses indicate the uncertainty in units of the last decimal digit: 0.047(3) means 0.047±0.003.

T^*	p^*_{max}	ρ^*_{max}	S	p^*_{min}	ρ^*_{min}
0.90	0.0293	0.047(3)	2.09	-0.376	0.66(2)
1.00	0.0488	0.073(6)	1.86	-0.198	0.62(2)
1.15	0.0753	0.13(2)	1.28	0.0016	0.53(3)
1.20	0.0888	0.15(2)	1.12	0.054	0.49(3)
1.25	0.105	0.19(3)	1.04	0.99	0.46(4)

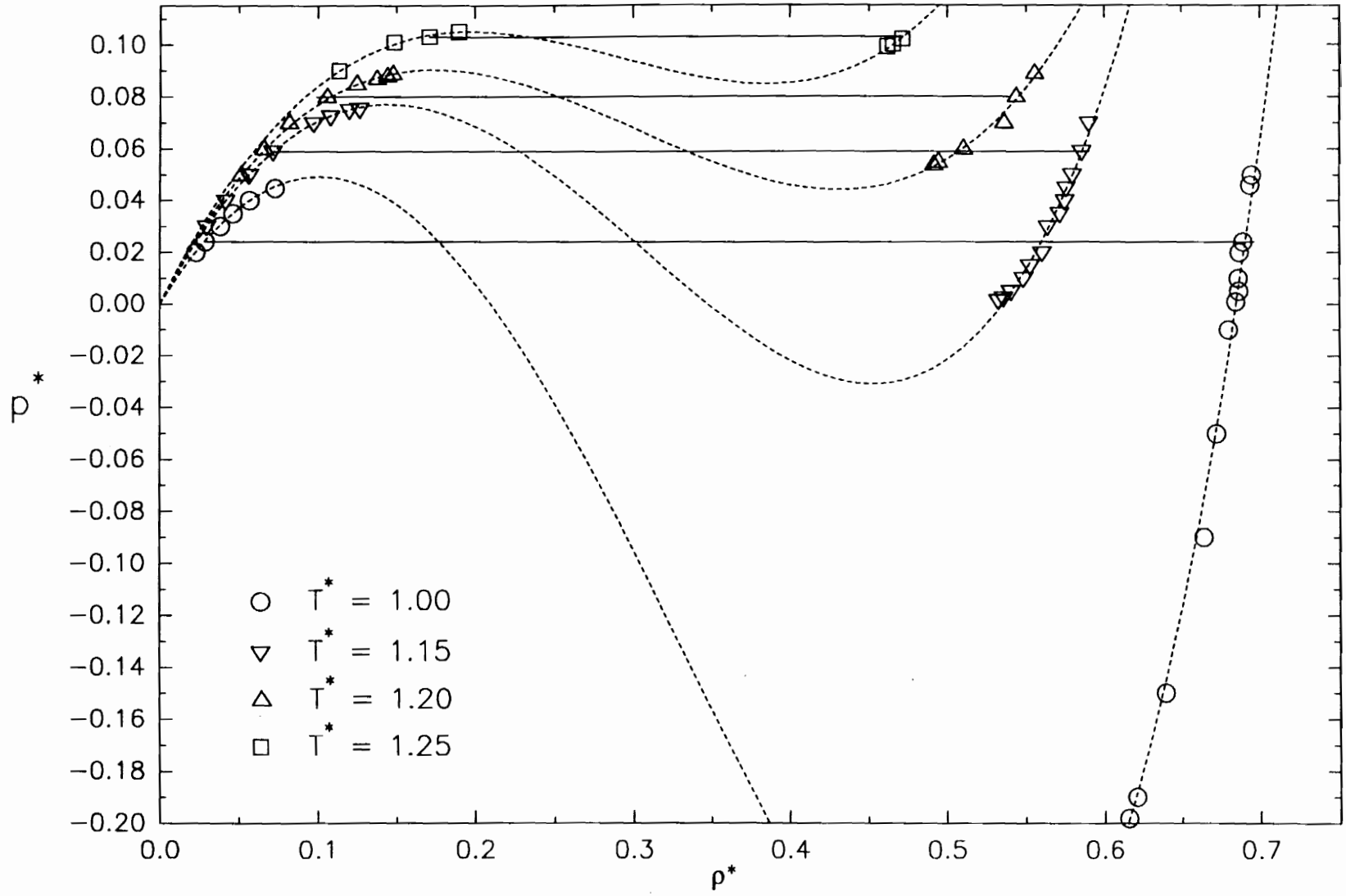


Figure 8. Isotherms for the 3D Lennard-Jones fluid. Solid lines are the liquid-vapor equilibrium tie-lines. Dashed lines are the fits of the truncated virial EOS to our data.

Figure 9 shows a comparison of points for $T^* = 1.15$ with those obtained by Hansen and Verlet [35] using the constant-NVT ensemble with homogeneity constraints. The points calculated here are shown by open triangles and the filled circles are points calculated by Hansen and Verlet. The two sets of points are compatible for the gas but exhibit appreciable differences for the liquid. Hansen and Verlet's calculations yielded a maximum pressure slightly higher than the present calculation and a minimum pressure considerably lower. A more important difference is the fact that the present constant-NpT calculations do not produce any points on the unstable portions of the isotherms between the pressure maxima and minima. This situation is probably physically realistic, as the van der Waals loops are artifacts of small system size [36].

The maximum and minimum of each isotherm are identified as the spinodes. In determining the spinodes, it was a concern that random fluctuations in the density might trigger phase transitions prematurely. However, this was not found to be a problem. Instead, the reverse was true, and rather long simulation runs were required to find the spinodes. For example, at $T^* = 1.15$, $p^* = 0.0016$ is the lowest pressure obtainable for the metastable liquid. When p^* was reduced to 0.0015, the simulation required 16 million iterations to reach the gas region. Once the transition to the gas began, it was very rapid. The system appeared to be a supersaturated liquid for about 15.3M iterations, but after 16M iterations the density was that of a dilute gas. Similar effects were observed on the gas side. There are metastable gas simulations that appeared to be at equilibrium for 20 million iterations that abruptly moved into the liquid region.

The spinodal curve can be constructed by connecting the spinodes of all isotherms, as shown in Figure 10. This figure shows a comparison of the calculated spinodal curve with those calculated by Abraham [9] and by Evans and Telo da Gama [37]. Abraham used

liquid state perturbation theory and Evans and Telo da Gama used the Percus-Yevick approximation. The present results agree with Abraham's, especially at higher temperatures. Our gas spinodal is also close to that of Evans and Telo da Gama; however, their liquid spinodes occur at appreciably lower densities than ours. Zeng and Oxtoby [12] have calculated the spinodal curve using nonclassical density functional theory. Our results agree with theirs for the gas region, but their liquid spinodal points are at somewhat lower densities than ours.

Supersaturated vapors and supersaturated liquids are metastable states. They are systems trapped in local free energy minima, where the system's fluctuations are insufficient to overcome the barriers separating local minima from global minima. As the spinode is approached, the barrier decreases and the fluctuations succeed in causing phase transitions. It would not be surprising if the spinodes determined by MC simulation depended on the size of the system used. Therefore, the spinodes have been partially checked for finite size effects. We searched for spinodes in systems containing 512, 729, 1000, and 5000 particles and found that the supersaturation ratios at the gas spinodes varied by less than one percent in these calculations, as shown in Figure 11. This independence of system size supports the validity of our reported spinodes.

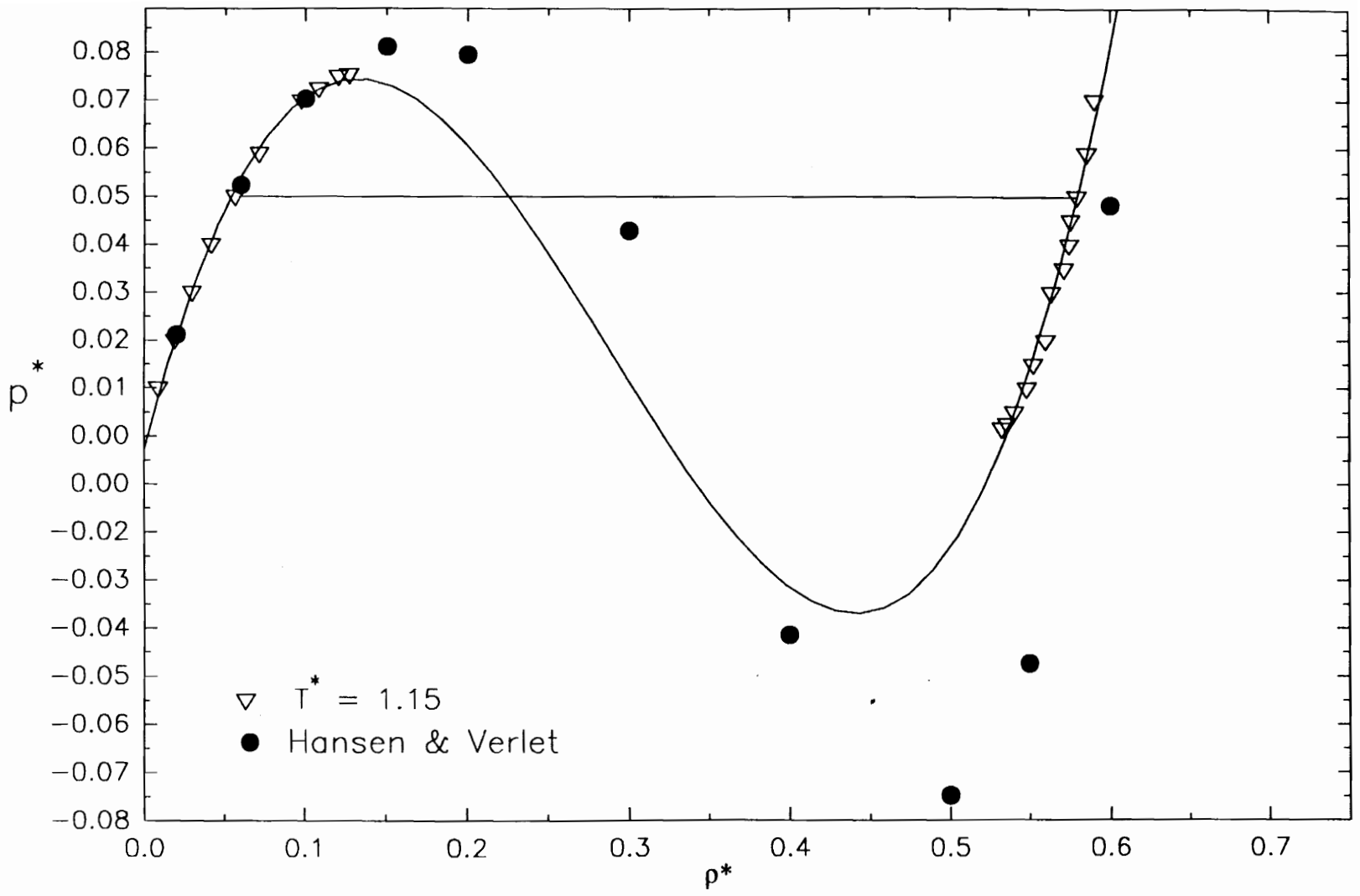


Figure 9. Pressure, density plot of our data and and Hansen & Verlet's points for the isotherm $T^* = 1.15$. The solid line is the truncated virial fitted to our data.

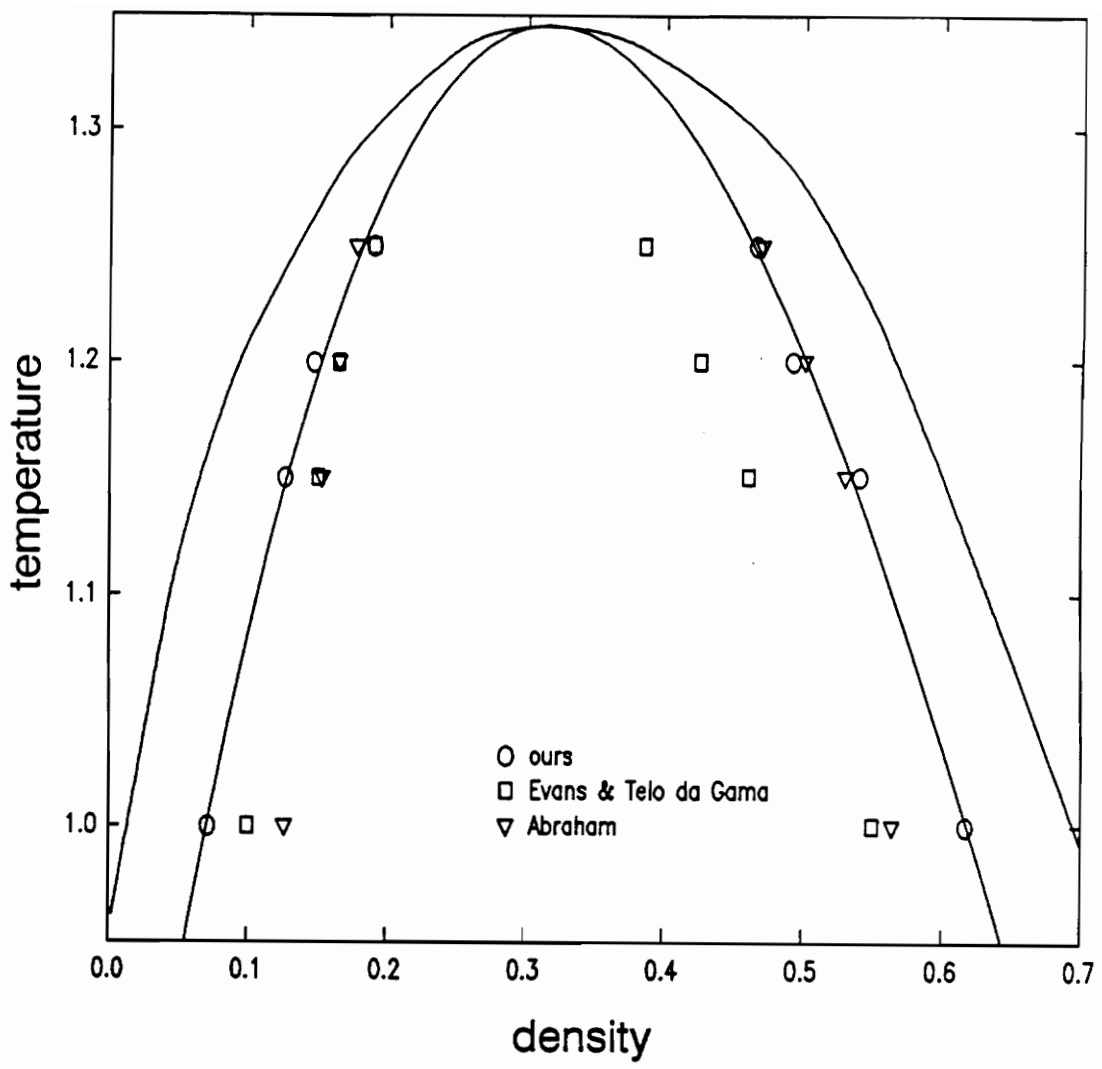


Figure 10. Phase diagram of the 3D Lennard-Jones fluid showing the liquid-gas coexistence curve and the spinodal curve. The spinodal curve is drawn through our data. The spinodes of Evans and Telo da Gama [37] and Abraham [9] are also shown.

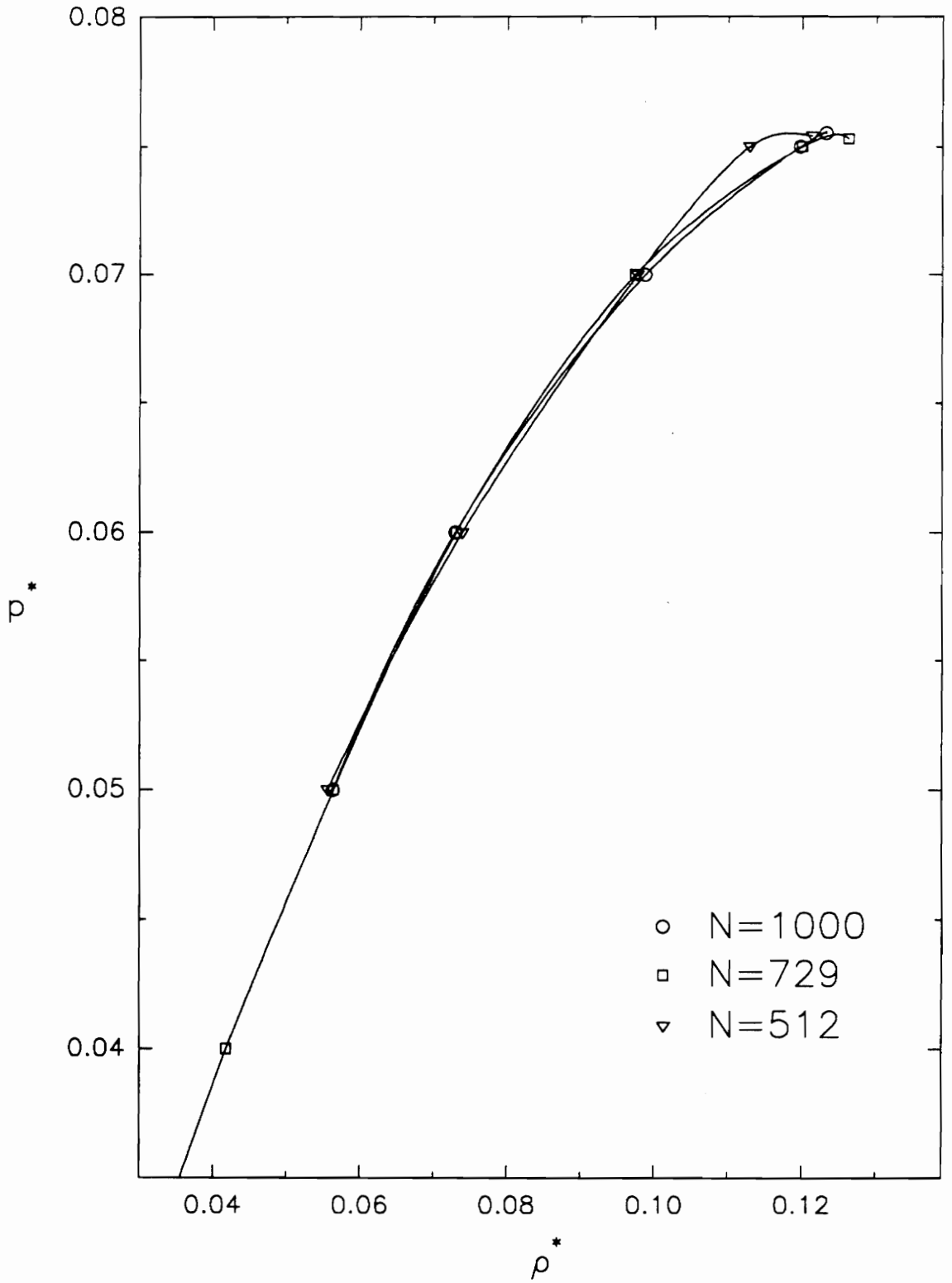


Figure 11. The LJ gas isotherm for $T^* = 1.15$ for different size systems.

1. Equations of State

To describe our simulated points, we attempted to fit them with several different equations of state (EOS). None of the standard equations was found to be completely suitable.

The simplest form tried was the van der Waals equation, which is written in reduced units as:

$$[p^* + a^*(\rho^*)^2][1/\rho^* - b^*] = T^* \tag{4.1}$$

where a^* and b^* are related to the usual van der Waals a and b by $a^* = a/\varepsilon\sigma^3 N_A^2$ and $b^* = b/\sigma^3 N_A$. We used a non-linear least-squares program to fit this equation to all of our calculated points and obtained the values $a^* = 4.59 \pm 0.03$ and $b^* = 0.995 \pm 0.003$. The van der Waals equation gives a reasonable approximation of the LJ equation of state in this range of pressures and temperatures, although major discrepancies appear in the neighborhoods of the spinodes. If the accepted values of $\varepsilon/k_B = 124$ and $\sigma = 342$pm for Ar [8] are used to scale the reduced constants, the differences are 16% in a and 34% in b from the normally accepted van der Waals values for Ar [8]. Figure 12 compares the van der Waals isotherms with the calculated points.

Isotherms calculated from the 33-parameter modified Benedict-Webb-Rubin EOS of Nicolas *et al.* [34] are plotted in Figure 13 as solid lines for $T^* = 1.00, 1.15, 1.20, 1.25$. Except for $T^* = 1.25$, the pressure maxima obtained from the Nicolas EOS, are higher than the present simulations were able to reach. Also, its isotherms do not match the calculated points in the liquid region. Adachi *et al.* [38] recently determined improved parameters for the Nicolas EOS. Isotherms calculated with the improved parameters, shown as dotted lines in Figure 13, are much closer to the simulated isotherms. The pressure maxima are

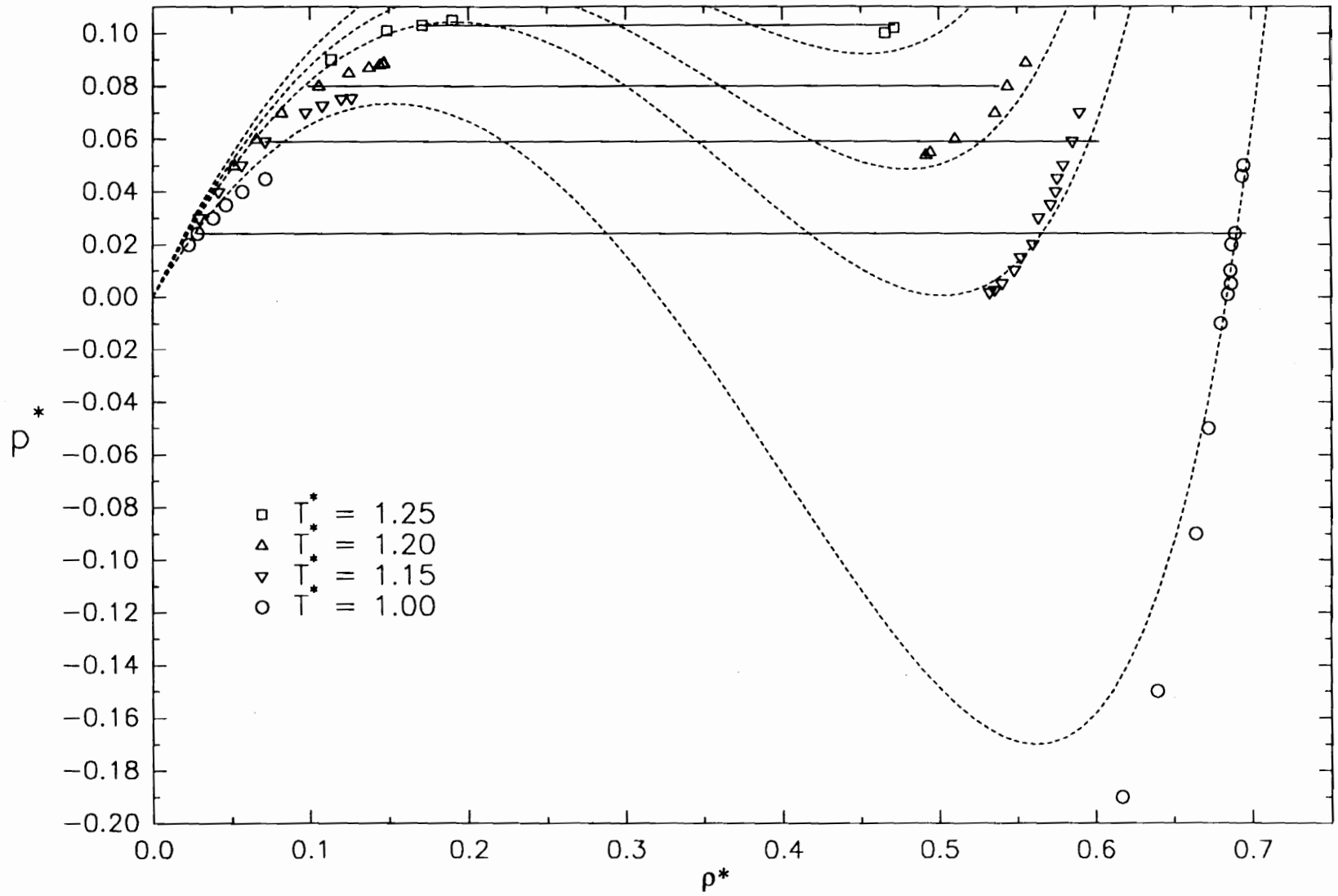


Figure 12. Isotherms of the van der Waals EOS obtained by fitting our data points using a least-squares procedure. The best values of the reduced van der Waals constants were $a^* = 4.59$ and $b^* = 0.995$.

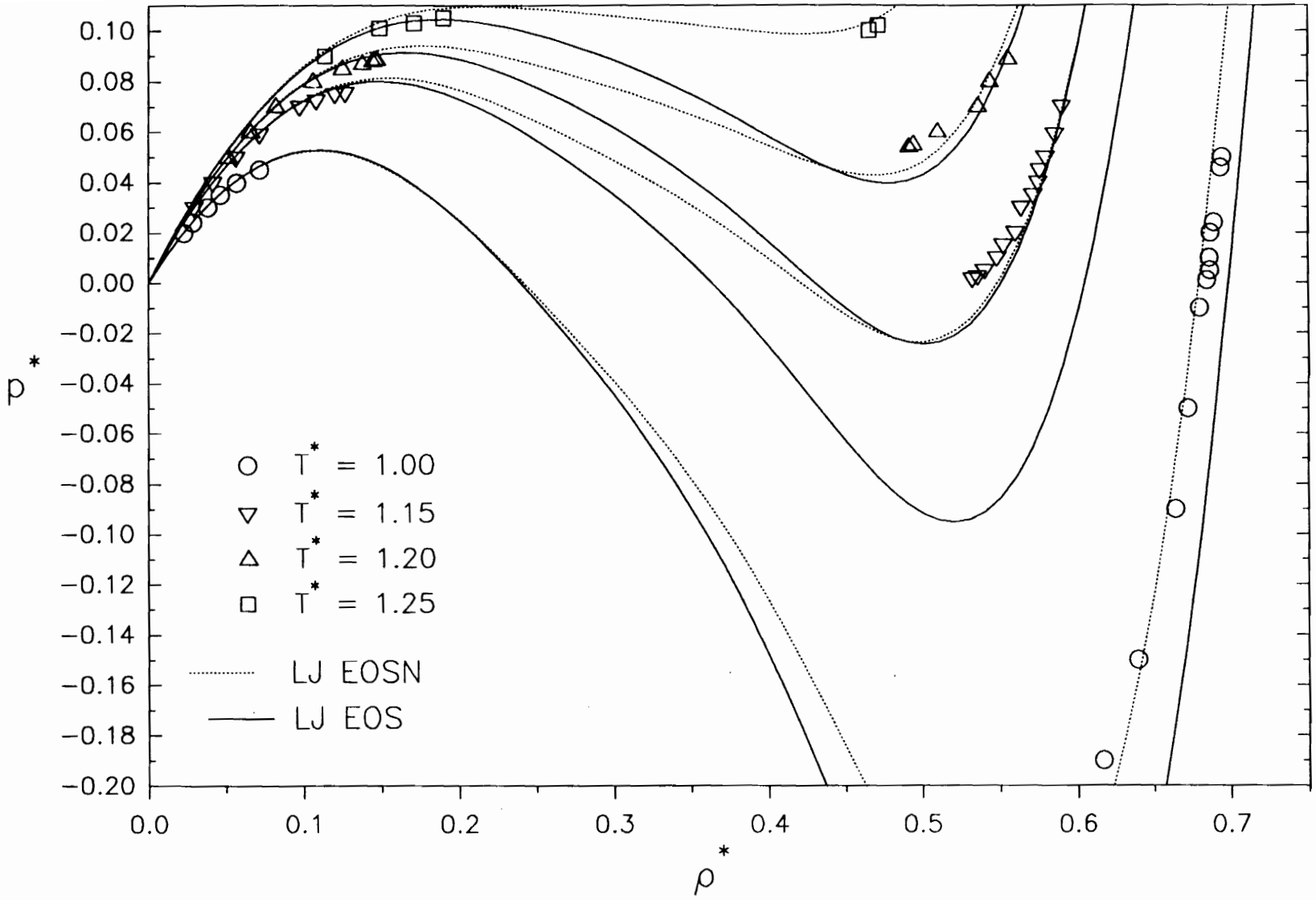


Figure 13. Isotherms of Nicolas [34] (LJ EOS) and Adachi [38] (LJ EOSN) compared with our data points.

still higher than the calculated maxima, but the liquid isotherms agree with the simulated isotherms. In the liquid region, isotherms calculated with either set of parameters give minimum pressures that are lower than the calculations indicate. The new parameters of Adachi *et al.* describe the simulation data better than the original parameters of Nicolas *et al.* However, neither set of parameters gives spinodes that correspond with the present results.

Barker and co-workers [39] have calculated accurate second, third, fourth, and fifth virial coefficients for a Lennard-Jones fluid. The virial EOS is written in reduced units as:

$$p^* = \rho^* T^* \{1 + B^* \rho^* + C^* (\rho^*)^2 + D^* (\rho^*)^3 + \ldots\} \qquad (4.2)$$

where the reduced virial coefficients are related to the normal ones by $B^* = B/N_A\sigma^3$ and $C^* = C/(N_A\sigma^3)^2$, etc. The virial equation gives isotherms that fit our points very well in the vapor region. However, on the liquid side, the densities are so high that one must extend the virial series past the fifth term to get accurate isotherms.

The best description of the calculated state points was obtained by truncating the virial EOS after the fourth term $[D^*(\rho^*)^3]$ and determining an empirical set of coefficients B^*, C^*, D^* for each temperature. The coefficients obtained in this way are given in Table 2.

Table 2. Coefficients for the truncated virial EOS of the LJ fluid.

T^*	B^*	C^*	D^*
1.00	-5.2(1.1)	0.43(35)	7.3(2.6)
1.15	-4.14(12)	2.5(4)	3.2(4)
1.20	-4.02(5)	4.3(2)	0.15(24)
1.25	-3.62(9)	3.3(7)	1.3(1.1)

The empirical values for B^* are close to the values calculated by Barker *et. al.*, but higher coefficients differ appreciably from theirs. The isotherms obtained by this procedure are the curves shown in Figure 8. They describe both the vapor and the liquid portions with essentially equal accuracy. However, the maximum and minimum pressures from the empirical equations do not agree with the spinodes from the simulations. The empirical EOS is not applicable for pressures between the maxima and minima listed in Table 1. This implies that the use of an empirical EOS in the unstable region may be unjustified.

2. Phase Transition

We have observed liquid-gas phase transitions in both directions. When the pressure of a metastable gas is above the gas spinode, the system transforms to a liquid. When the pressure of a metastable liquid is below the liquid spinode, the system transforms to a gas. An allowed cycle of transitions is shown in Figure 14. The figure shows a cycle of a saturated gas (A), supersaturated gas (B), dense liquid (C), saturated liquid (D), supersaturated liquid (E), dilute gas (F), back to saturated gas (A). However, when the liquid spinode occurs at a negative pressure, there is no such cycle. The liquid-to-gas transition is not defined in such cases, because the gas isotherm does not exist for negative pressures. Attempts to simulate these transitions resulted in unbounded increases in the system's volume.

We computed several cluster statistics at the gas spinode. Because we were interested in observing liquid-like clusters, we used the definition that two particles are in the same cluster and neighbors if the distance between them is less than 1.5σ. This distance is approximately the largest near-neighbor distance in the LJ liquid [5,40]. The cluster distribution at the gas spinode for $T^* = 1.15$ is given in Figure 15 as a log-log plot. The number of particles in a cluster is designated as g, and the equilibrium number of clusters with g particles is n_g. The exponent, τ, in $n_g \propto g^{-\tau}$ is 2.08±0.01. The accepted universal value of τ is 2.2 [18]. At this T^* and p^* there was always a large cluster. This large cluster percolated 32% of the time with an average number of 458 particles. Because of the periodic boundary conditions the percolating cluster is actually an infinite cluster. When the largest cluster was not percolating, it contained, on average, 219 particles. Cluster distributions at the spinodes at other temperatures gave similar τ values, which are comparable to the τ values given in our earlier NVT work [19]. The cluster distribution of

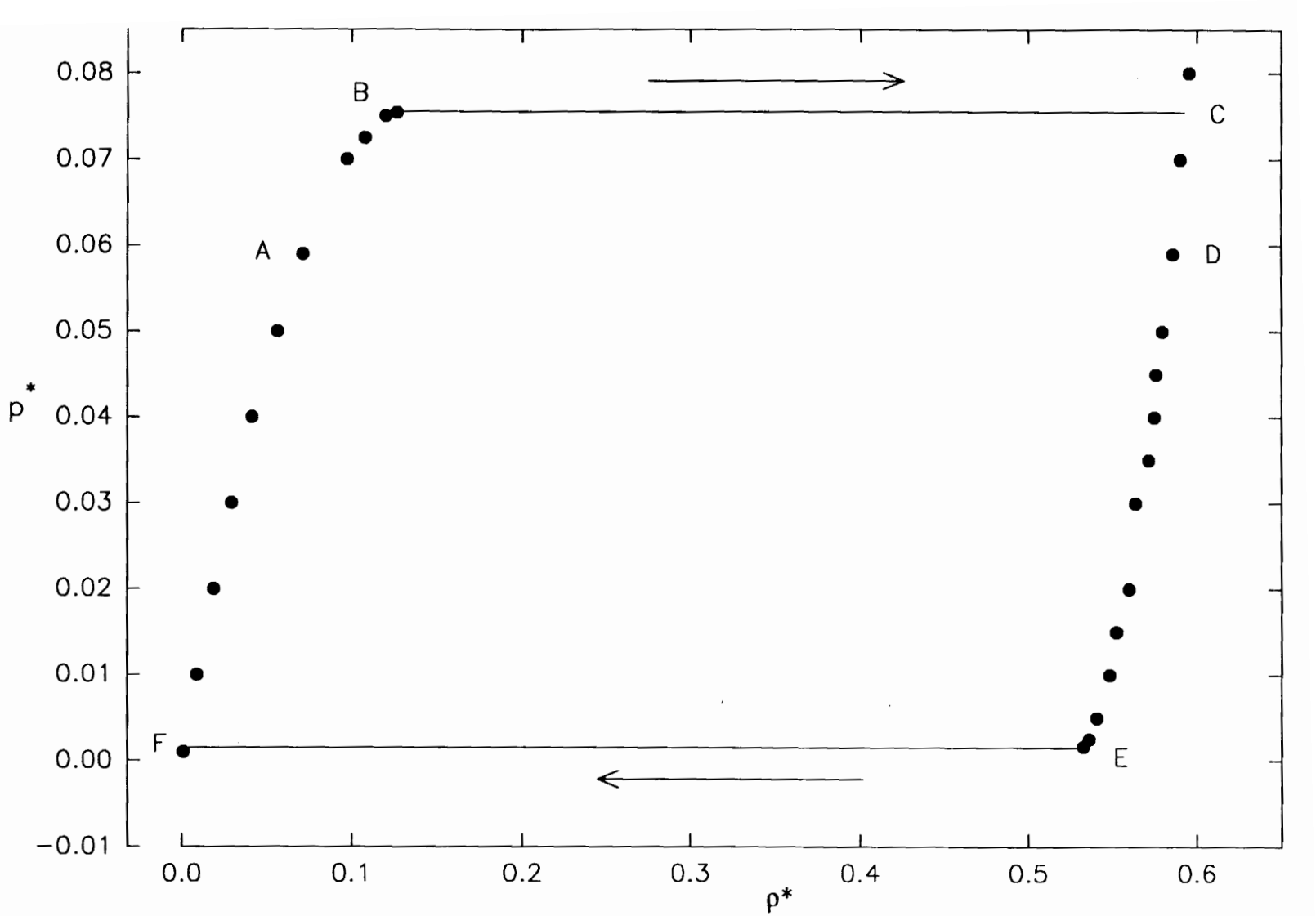


Figure 14. Cycle of transitions observed at $T^* = 1.15$. (A) equilibrium gas, (B) supersaturated gas, (C) dense liquid, (D) equilibrium liquid, (E) supersaturated liquid, (F) rarefied gas. Point B is the gas spinode and point E is the liquid spinode.

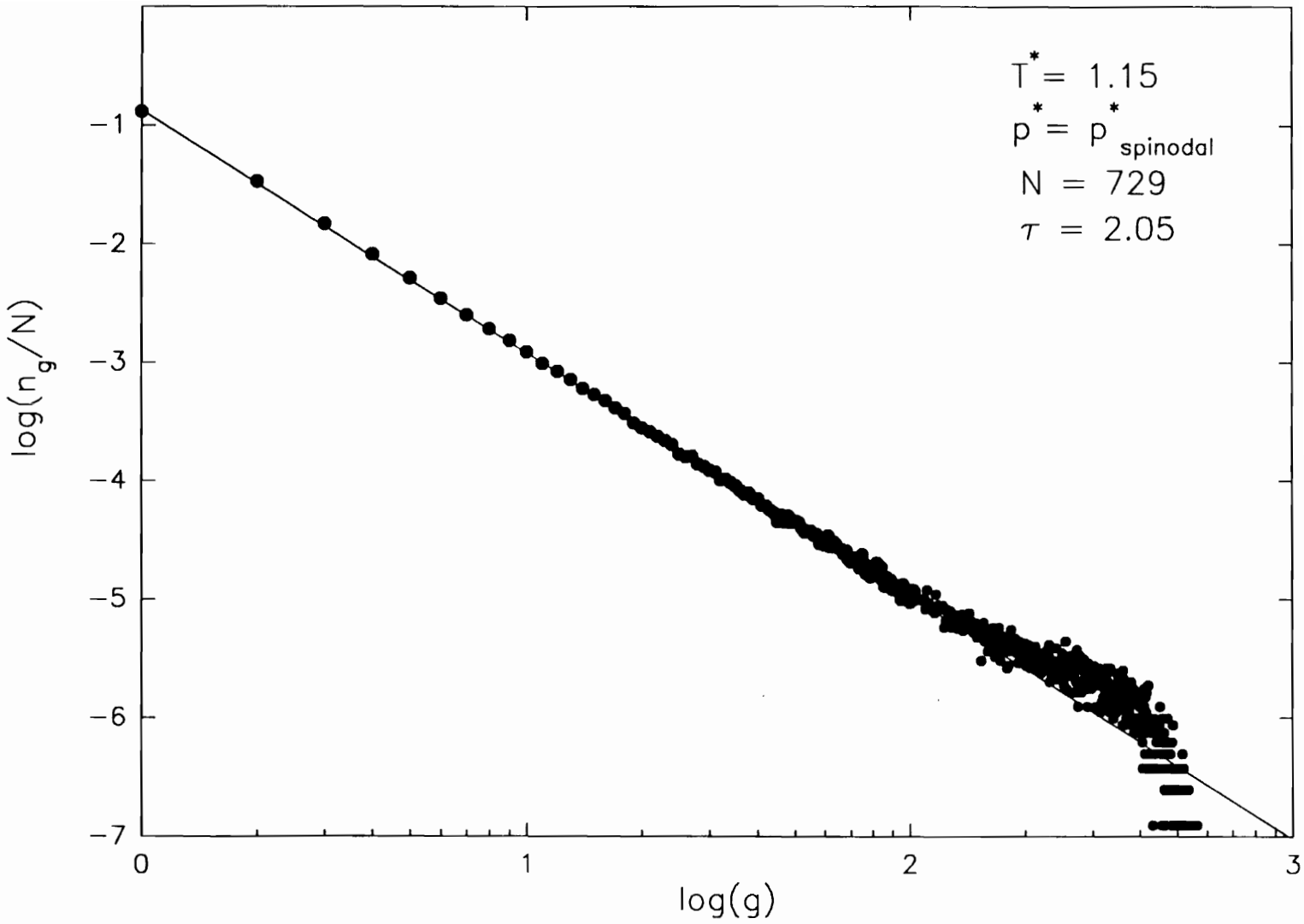


Figure 15. The cluster distribution for $T^* = 1.15$ at the gas spinode for 729 LJ particles.

the gas spinode is given in Figure 16 for 5000 particles. The cluster distributions (CD) of the two systems are very similar until $g \approx 400$. The 5000 particle system's CD does not have the curve at high g found in the 729 particle system's CD, but it does include the curve points. The 5000 particle CD does have extensions into the $g \approx 1900$ range. Of course, the 729 particle system cannot form clusters that large.

The number density profiles of the largest non-percolating clusters are given in Figure 17 for three temperatures. At their centers, these clusters are roughly half as dense as the equilibrium liquid at each temperature. Table 3 summarizes the cluster statistics for each temperature. The percolation probability increases with the temperature as would be expected since the cluster increases in spatial extent, as shown by the density profile. The smaller clusters at lower temperatures are more compact and do not possess the finger-like structure that the clusters at the highest temperature have, which enables half of the configurations to percolate. As seen in the table, the percolation ratio is less than 0.50 for all but the highest temperature studied. The percolation threshold density cannot be found using the NpT ensemble for the lower temperatures because this ensemble will not allow half of the configurations to percolate. Since these clusters are at the spinodes, the number of particles they contain should mirror the size of the critical condensation nucleus. The size of the critical nucleus increases with temperature. At higher temperatures, the critical condensation nucleus is a cluster that spans the simulation cube (percolates) but does not have a high density center. At lower temperatures, the critical nuclei do not percolate but have higher density centers.

According to classical nucleation theory [4], if a cluster is large enough to be a critical nucleus, it will grow spontaneously and consume all the particles in the system to form a liquid. To try to observe a critical nucleus, we increased the pressure slightly above the gas

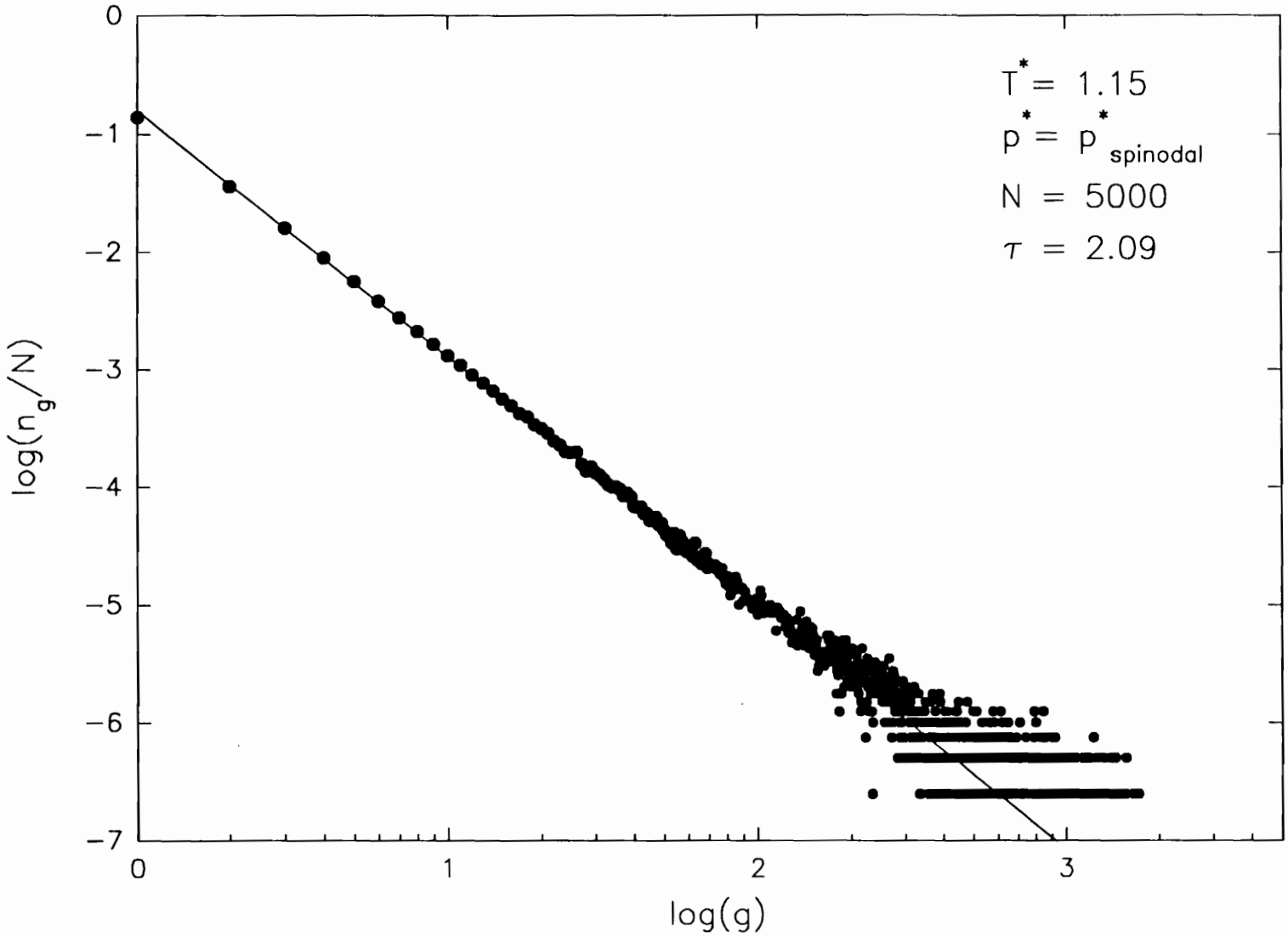


Figure 16. The cluster distribution for $T^* = 1.15$ at the gas spinode for 5000 LJ particles.

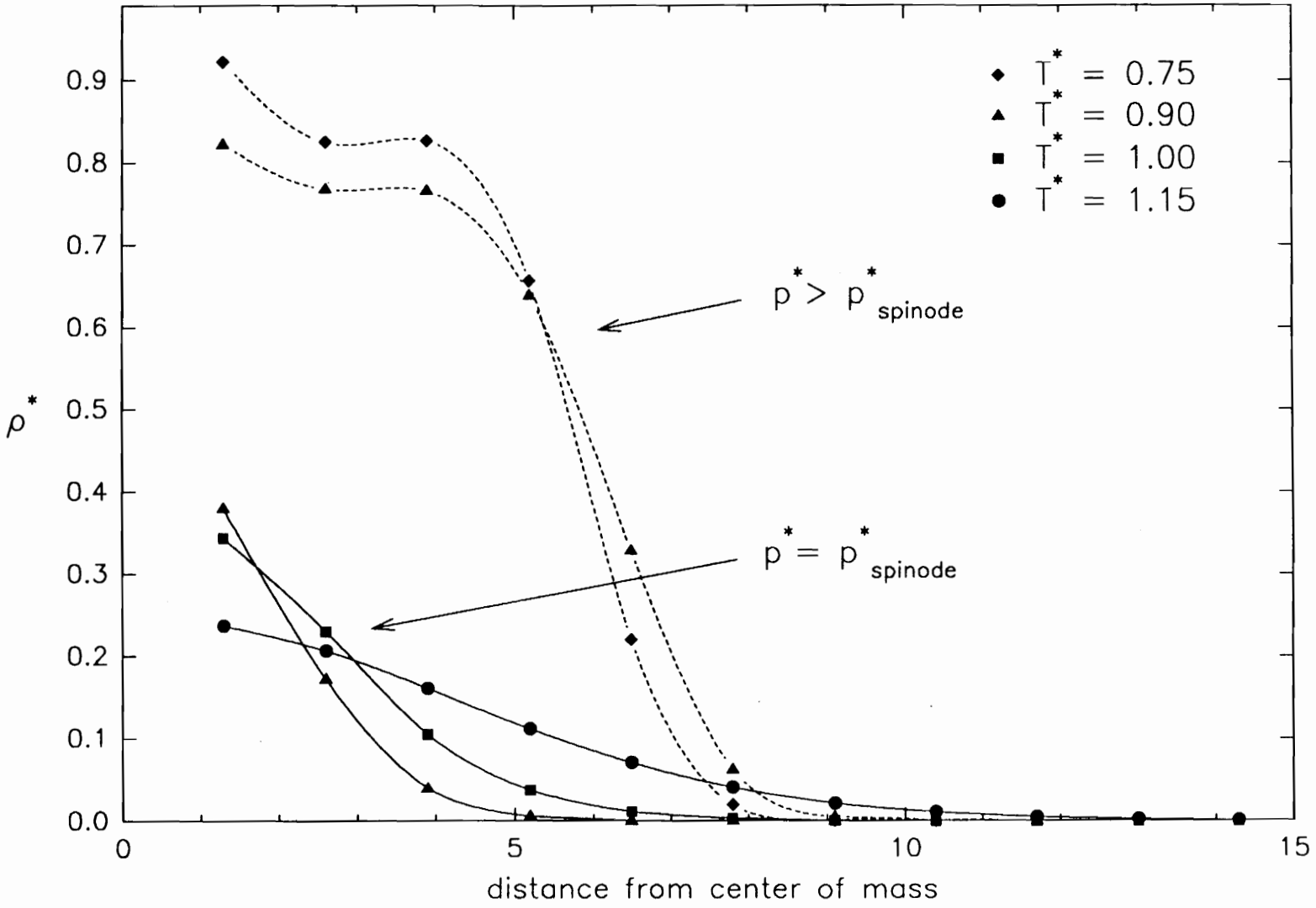


Figure 17. Density profiles for the largest non-percolating LJ clusters at the gas spinodes and for the largest non-percolating clusters at pressures slighlty above the spinodal pressures.

spinode. At $T^* = 1.15$, the transition to the liquid was very rapid and we were unable to observe the critical nucleus. The system density changed linearly with system energy during the transition.

Table 3. Cluster statistics for the Lennard-Jones gas spinode at $T^* = 0.90, 1.00, 1.15, 1.20$.

T^*	$< g_{max} >$	τ	% percolation	ρ^*_{center}	$\rho^*_{sat'd\ liquid}$
0.90	24	2.48	0.00	0.380	0.758
1.00	60	2.22	0.03	0.344	0.703
1.15	219	2.08	32.3	0.237	0.605
1.20	276	2.21	68.9	0.208	0.537

At sufficiently low temperatures, we were able to trap what appears to be a critical nucleus. For example, when $T^* = 0.90$, the gas spinode occurs at $p^* = 0.0293$. At this temperature, we performed a simulation starting with a gas at $p^* = 0.0300$. In this case, a large cluster formed rapidly, and the energy of the system immediately became comparable to that of a liquid. However, the density of the system was still that of a gas. Thus, soon after the start of the simulation, we had a quasi two-phase system. As the simulation proceeded, the system energy changed nonlinearly with system density, in contrast to the transition at the higher temperature in which the system density varied linearly with the system energy. The cluster was confined temporarily as a large dense region while the volume changes slowly carried the system density to an appropriate value for the liquid. Figure 18 shows the energy and density as the system transformed from a gas to an unstable "trapped" two-phase system and finally to a stable liquid. It appears that the slow volume changes of the simulation could not "keep up" with the cluster formation to maintain a

single-phase system. Attempts to speed up the transition by increasing the frequency of volume changes were unsuccessful.

The average maximum cluster size for each block of ten cycles is shown in Figure 19 as this transition proceeded. The average maximum cluster size at the spinode for $T^* = 0.090$ is 24 (see Table 3). The transition occurred after a cluster with 68 particles formed, as shown in the figure. However, the largest cluster formed at the spinode contained 66 particles. Hence, it is difficult to determine the size of the condensation nucleus. A similar plot for a transition with 5000 particles reveals that the cluster size is essentially identical to the system with 729 particles. Thus, the average cluster size and the size of the condensation nucleus are not influenced by finite size effects. A study of the largest cluster's density profile during this process indicated that the cluster grew in several stages. Particles were first added to the cluster's exterior, while the density at the center remained low. When the number of particles in the cluster reached about 100, there occurred a period of internal rearrangement, in which the density in the cluster's interior became liquid-like as particles were being added. Finally, the whole system collapsed to a liquid. After the second stage, the density profile of the largest non-percolating cluster had the shape shown in Figure 17. The density at the cluster's center was $\rho^* = 0.78$. This indicates that the cluster was liquid-like in the center, for the density of the equilibrium liquid is 0.76 at this temperature [6].

The same procedure was repeated for $T^* = 0.75$. This calculation gave rise to a large non-percolating cluster with a liquid-like density, 0.92, at its center "trapped" in a dilute gas. This cluster's density profile is also shown in Figure 17. Fifty-two million iterations

were required for the simulation to reach a uniform condensed phase. The density profiles for the trapped clusters are not as precise as those at the spinodal because the system is unstable and is slowly transforming to liquid during the calculation.

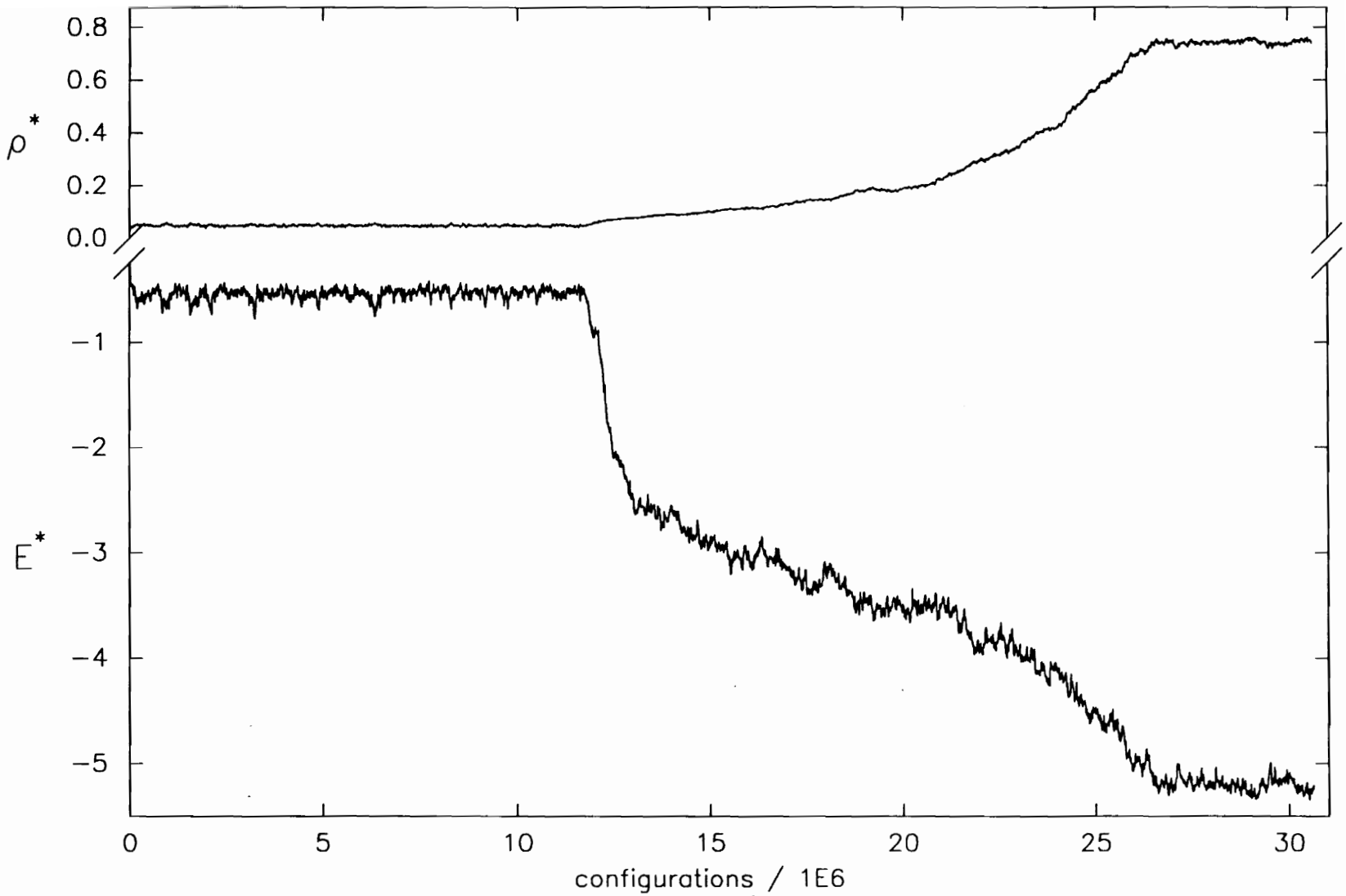


Figure 18. Block energy and density vs iteration number for $T^* = 0.90$ showing the transition occurring at approximately 12M iterations.

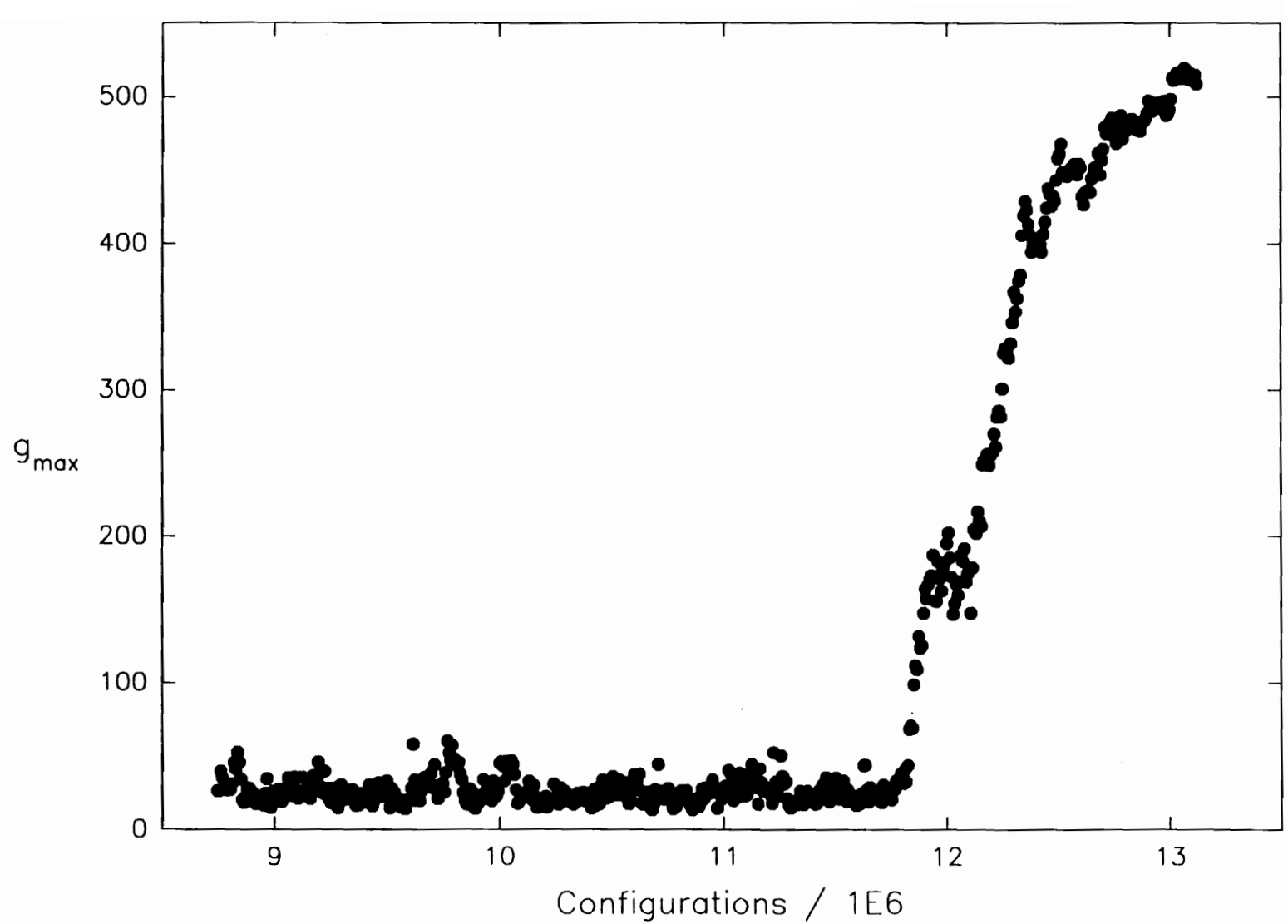


Figure 19. Maximum cluster size during the gas-liquid transition for $T^* = 0.90$ averaged over 10 cycles.

B. THE STOCKMAYER FLUID

Adding a point dipole to the LJ fluid changes the isotherms and spinodal points of the fluid. Referring to Figure 20, it is seen at $T^* = 1.15$ that the addition of the dipole ($\mu^* = 1.00$) causes the system on the gas side to be at higher density at the same pressure. Also, the spinodal point occurs at lower pressure and density than those with no dipole. Similar results occur at $T^* = 1.25$, as shown in the figure. On the liquid side of the phase diagram, the dipole has an increased effect on the isotherm. The density is much higher with the dipole moment than without it. The liquid spinode occurs at appreciably lower pressures with the inclusion of the dipole. The effect of the dipole on the liquid isotherm is much greater than on the gas isotherm, as would be expected since the particles are much closer. Appendix A.2 lists the simulation details for the Stockmayer isotherms.

If the strengths of dipole moments vary between zero and one, the calculated isotherms lie between those calculated at $\mu^* = 0.00$ and $\mu^* = 1.00$. The isotherm with $\mu^* = 0.50$ does not exist half-way between the other systems. Instead, it occurs very close to the $\mu^* = 0.00$ system, as shown in Figure 21 and magnified in Figure 22. Table 4 lists the calculated spinodes for the Stockmayer fluid. There are no known literature values for the spinode with which to compare these results. Points were calculated for different dipole moment strengths at the same pressure and temperature and their properties are given in Table 5. These points are included in Figure 22 as solid circles. It seems that there is a non-linear relationship between the magnitude of the dipole moment and the location of the isotherm. This can be seen from the isotherm of the system with $\mu^* = 0.50$. The isotherms for the greater dipole moments were not completely calculated, but the points shown in the table reveal the non-linear relationship. Thus, one can dictate the spinodal point by specifying the magnitude of the dipole moment.

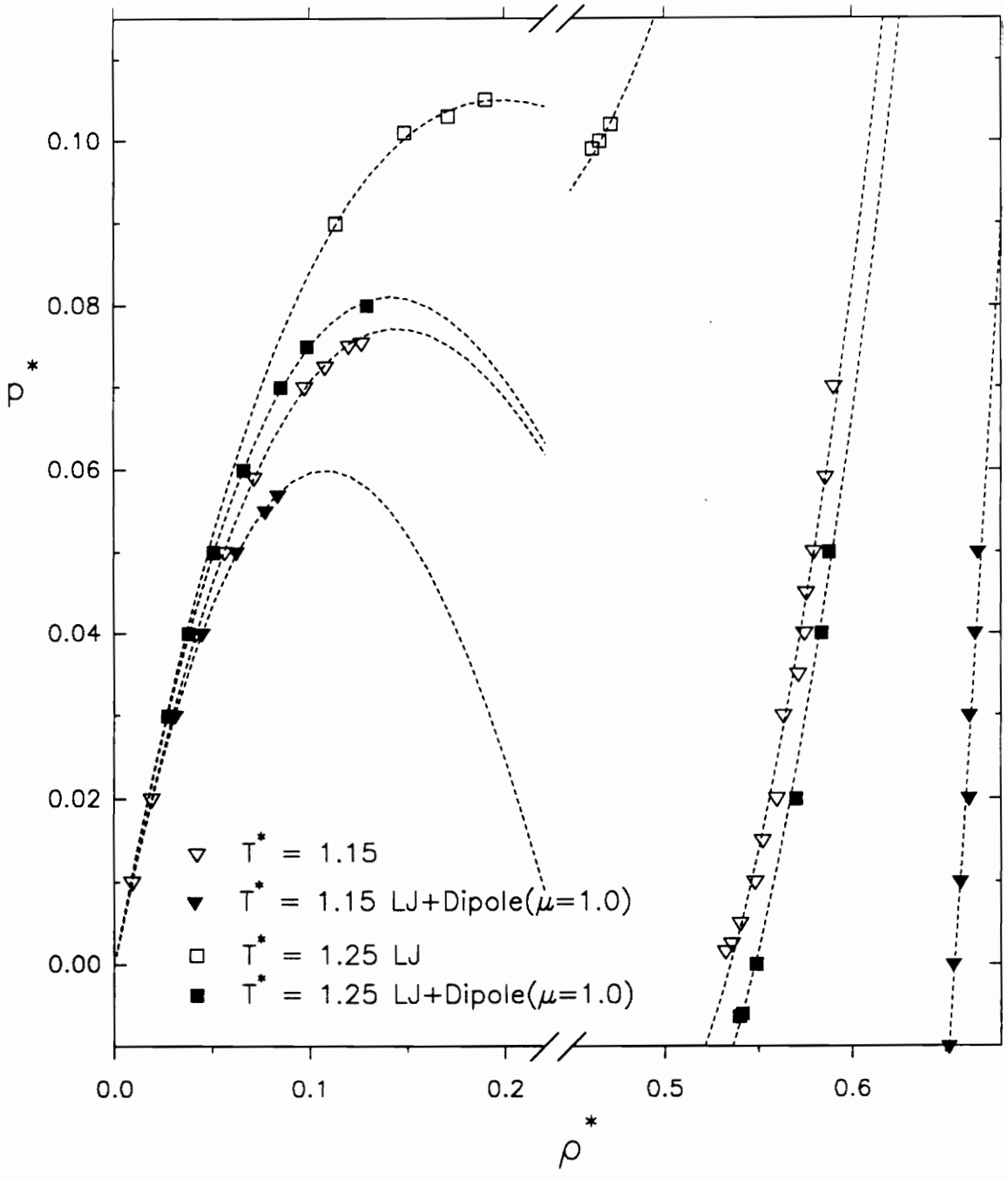


Figure 20. Isotherms for the LJ and Stockmayer fluids at $T^* = 1.15$ and $T^* = 1.25$.

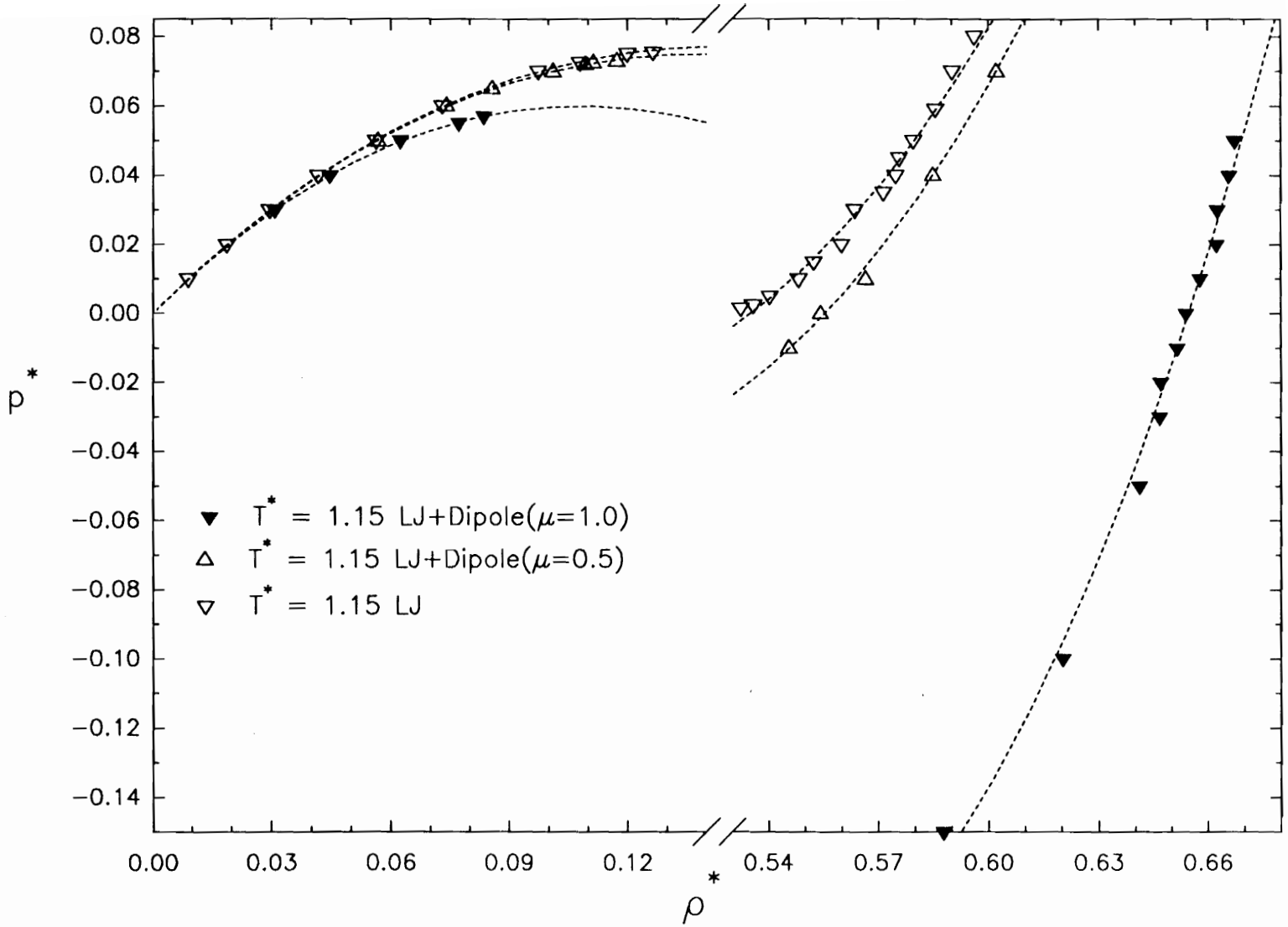


Figure 21. Comparison of the Stockmayer isotherms with different dipole moment strengths for $T^* = 1.15$.

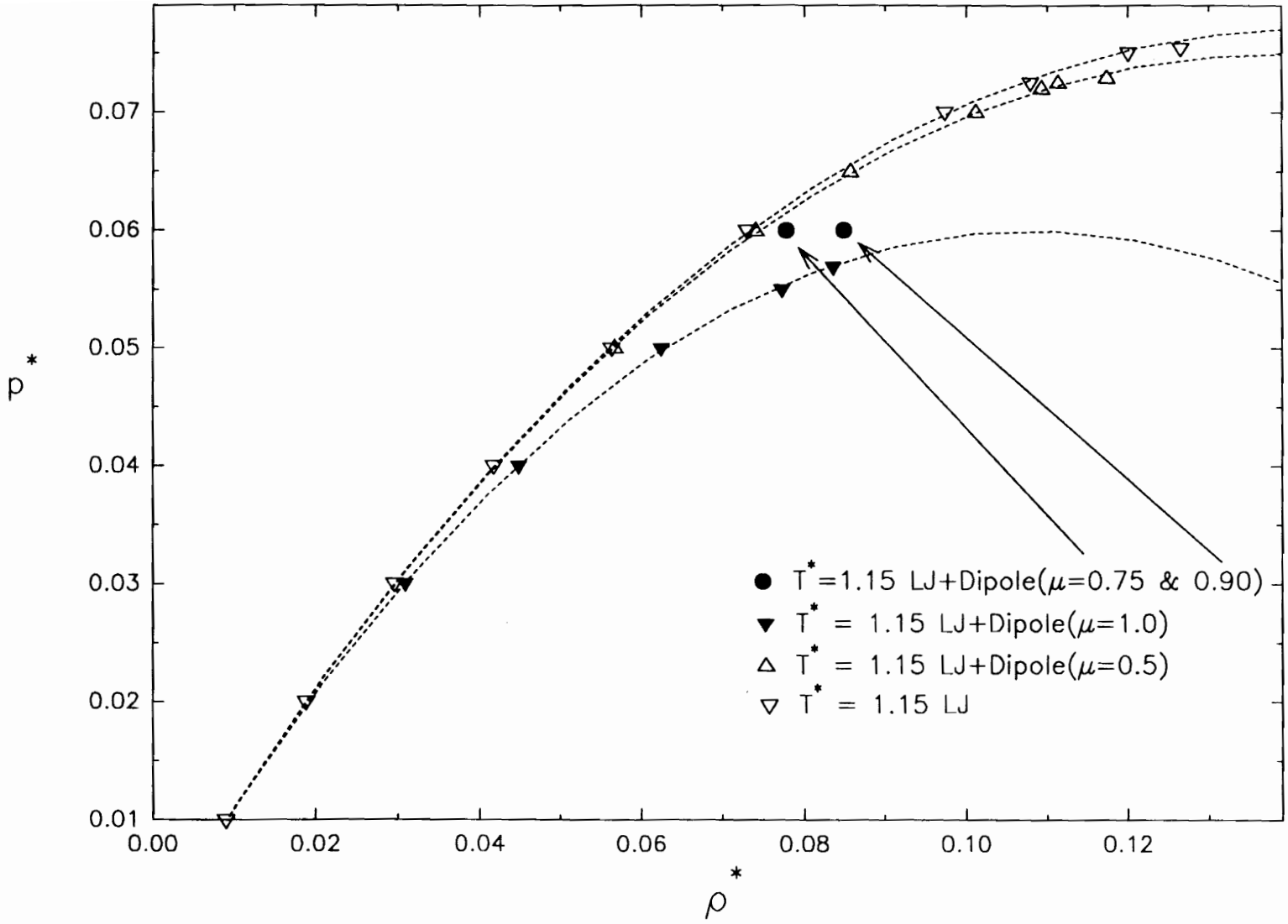


Figure 22. Gas isotherms at $T^* = 1.15$ comparing points with varying dipole moment magnitudes.

Table 4. Spinodal points for the Stockmayer fluid.

T^*	μ^*	p^*_{max}	ρ^*_{max}	S[7]	p^*_{min}	ρ^*_{min}
1.15	0.50	0.0729	0.117(15)		-0.012	0.540(21)
1.15	1.00	0.0569	0.0836(7)	1.8	-0.150	0.588(23)
1.25	1.00	0.0801	0.123(15)	1.2	-0.0064	0.54(3)

Note: The supersaturation ratio cannot be calculated for the $\mu^* = 0.50$ system because the saturated liquid pressure is not known.

Table 5. The properties of the systems with different dipole moment strengths at $T^* = 1.15$ and $p^* = 0.060$

μ^*	ρ^*	U^*
0.00	0.073(4)	-0.63(4)
0.50	0.074(5)	-0.65(4)
0.75	0.078(5)	-0.74(5)
0.90	0.085(6)	-0.91(8)
1.00	-	-

Note: The system with $\mu^* = 1.00$ does not have a point at $p^* = 0.060$ on the gas isotherm, for the system's gas spinode is at $p^* = 0.0569$.

The isotherms are fitted with a truncated virial EOS, like the LJ fluid, with three coefficients. The results are given in Table 6 and shown in Figure 20 and Figure 21. The first virial coefficients agree well with those of Bird [41] with only 1% discrepancy. However, the second coefficients are inconsistent with the values of Bird. The C^* value for $T^* = 1.15$ and $\mu^* = 0.50$ agrees reasonably (10 percent error) while the other dipole points have errors as high as sixty percent. The fact that C^* was interpolated from a table that did

not explicitly include the temperature and dipole strength could have caused the error and the success. Equivalent to the LJ fluid, the truncated virial EOS does a respectable job of approximating the state points of the Stockmayer fluid.

Table 6. Stockmayer fluid coefficients for the truncated virial EOS compared to those of R.B. Bird [41] (indicated with a prime).

T^*	μ^*	B^*	$B^{*\prime}$	C^*	$C^{*\prime}$	D^*
1.15	0.50	-4.21(1)	-4.22	2.27(8)	2.55	3.7(1)
1.15	1.00	-5.1(2)	-5.1	1.9(6)	4.73	5.3(5)
1.25	1.00	-4.36(9)	-4.35	3.5(3)	4.61	2.1(3)

The cluster distributions are shown in Figures 23 and 24 for the Stockmayer fluid at the spinode for $T^* = 1.15$ for the two different magnitude dipoles. They are plotted in log-log fashion as before and fitted with the scaling theory, $n_g \propto g^{-\tau}$. The τ values agree with the accepted value and those obtained earlier. The cluster properties are given in Table 7. The equilibrium properties for $\mu^* = 0.50$ are not given in the literature. Notice from the cluster distributions that the system with the most monomers is the stronger dipole system. Also, the weaker dipole system has clusters that are much bigger than in the stronger dipole system. This is evident from $< g_{max} >$ given in the table. The big clusters are not present in the $\mu^* = 1.00$ system because the critical cluster has been formed and the system condensed to the liquid state. This occurs at a much lower density and pressure than the $\mu^* = 0.50$ or $\mu^* = 0.00$ systems, which indicates that the particles are attracted to each other in a manner such that the barrier to condensation is overcome at lower supersaturation ratios than without the dipole moment interaction.

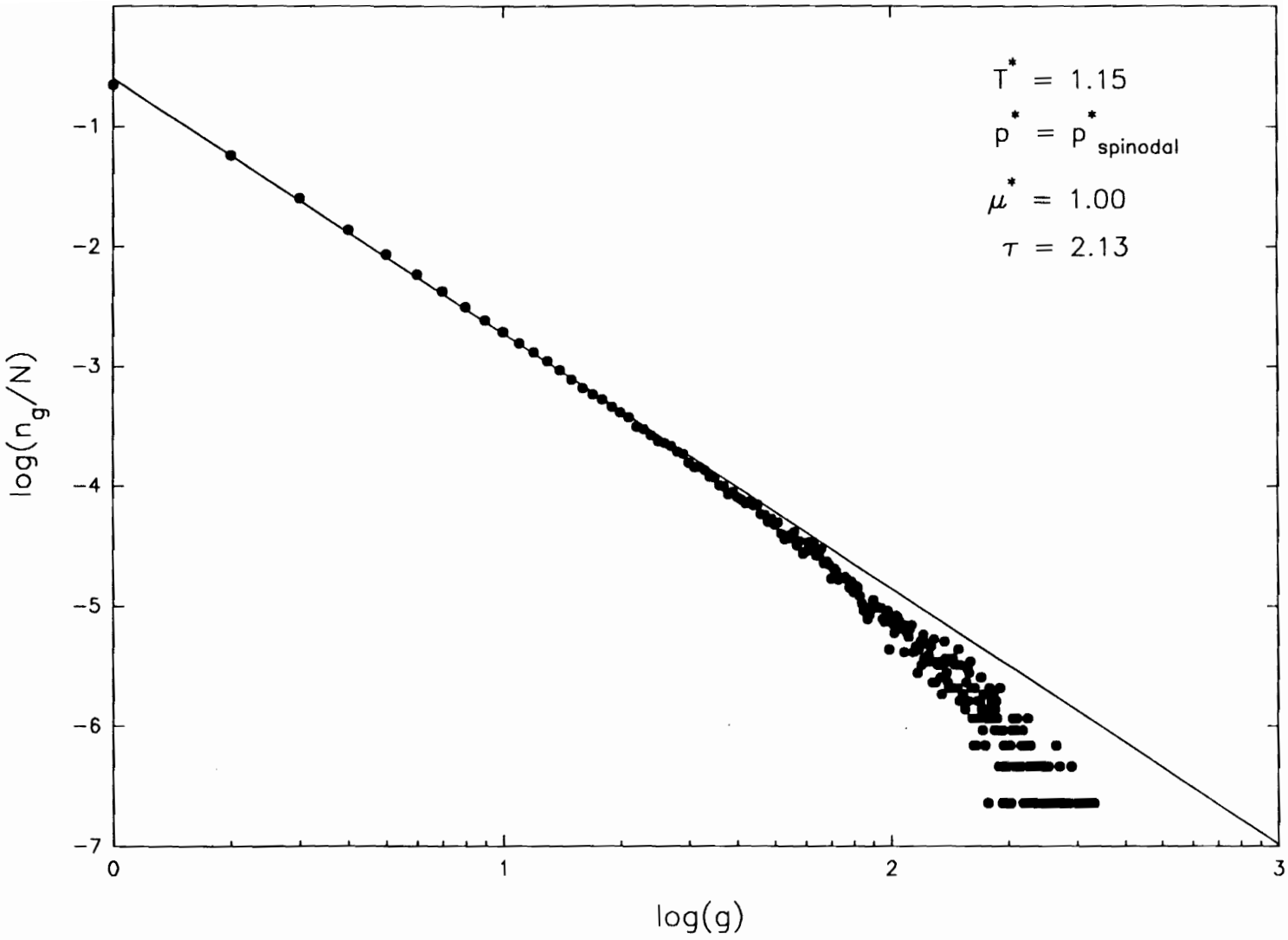


Figure 23. The cluster distribution at the spinode for the Stockmayer fluid at $T^* = 1.15$ and $\mu^* = 0.50$.

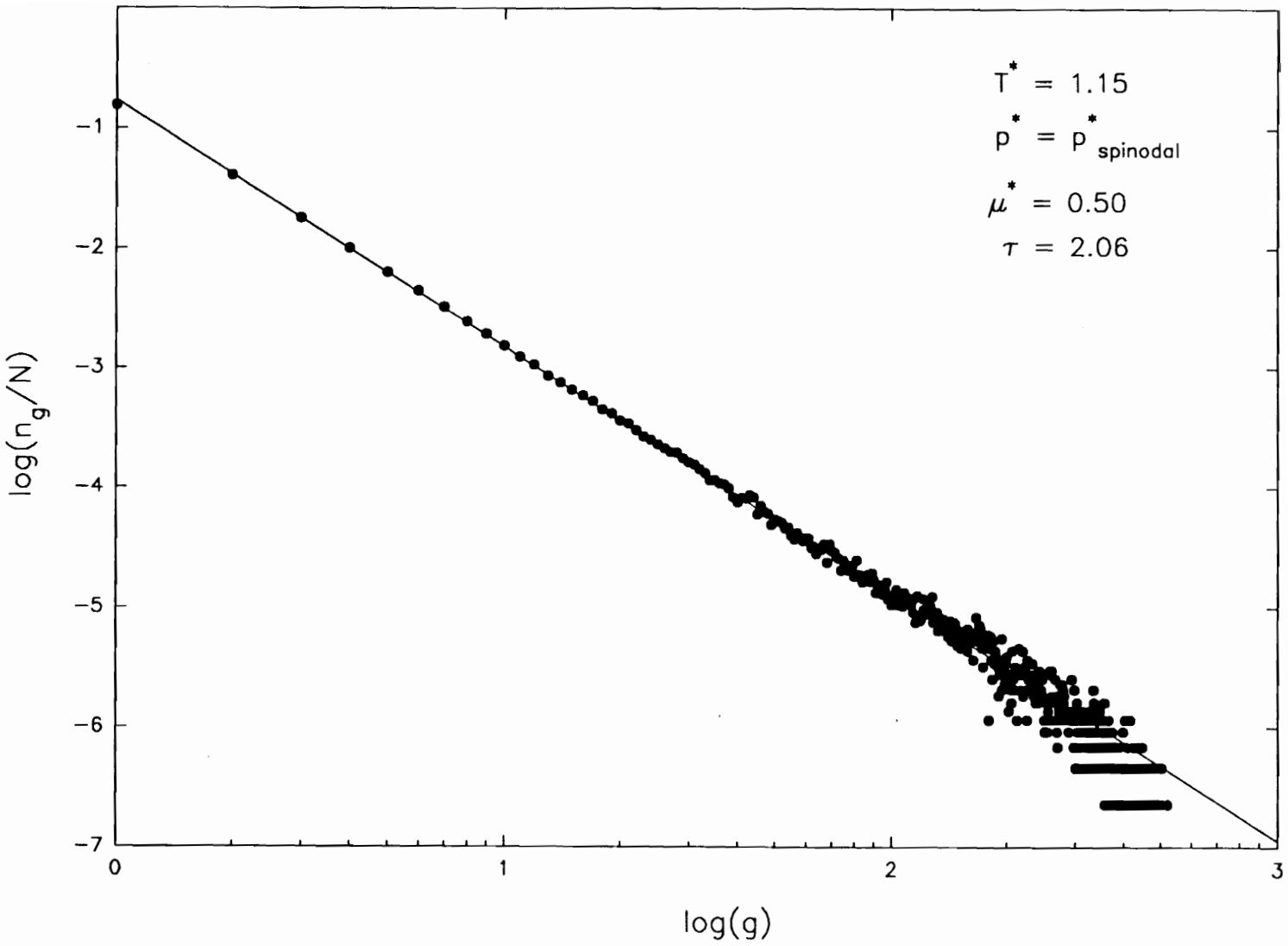


Figure 24. The cluster distribution at the spinode for the Stockmayer fluid at $T^* = 1.15$ and $\mu^* = 1.00$

Table 7. Cluster properties for the Stockmayer fluid spinodal points.

T^*	μ^*	$< g_{max} >$	τ	% perc.	ρ^*_{center}	$\rho^*_{eq\ liquid}$[7]
1.15	0.50	152	2.06	6.5	0.309	-
1.15	1.00	85	2.13	1.5	0.253	0.684
1.25	1.00	209	2.07	19.6	0.257	0.648

Note: The $\rho^*_{eq\ liquid}$ value for $T^* = 1.15$ and $\mu^* = 0.50$ is not given in the reference.

The density profile is calculated as it was for the LJ fluid for each supersaturated vapor at the spinode. As shown in Figure 25, the clusters do not have a liquid-like center. Their centers' densities are about half the density of the equilibrium liquid. The sharpness of the fall-off in the density profile is an indication of how well the surface is defined. Thus, the system with the most definite surface is the system with the largest dipole moment magnitude. Likewise, the system with the most tenuous surface has no dipole-dipole interaction. The orientational profile for the $\mu^* = 1.00$ system is given in Figure 26. It reveals the angle that the dipole makes with a vector from the center of mass of the cluster to the point dipole. It is calculated for each shell of the cluster in hopes of finding different orientations at the surface. The number calculated in the simulation is the average number of times a certain angle was formed between the dipole and the vector. To find the density orientational profile, the $\sin\theta$ has to be taken out. This is done by dividing the number by $2\pi \sin\theta \partial\theta$. 2π is the integral over all the ϕ angles and $\partial\theta = \pi/18$. This could be explicitly done in the simulation by accumulating the angles in intervals of $\cos\theta$ and then dividing by 2π. As seen in the figure, all the shells are have a flat density orientational profile. This means that the dipole angles are equally distributed with respect to the center of mass

vector. The noise in the larger shells is from poor averaging because there are not many points there. The density orientational profile for the $\mu^* = 0.50$ system is essentially identical to that of the $\mu^* = 1.00$ system.

Thus, we can change many characteristics of a system by adding a dipole moment. The spinodal point can be altered. The structure of the condensation cluster can be changed depending on the magnitude of the dipole moment. The density profile and surface characteristics will change with dipole strength. The cluster distribution will be different for different dipole moments. Therefore, one can design a molecule to have desired properties.

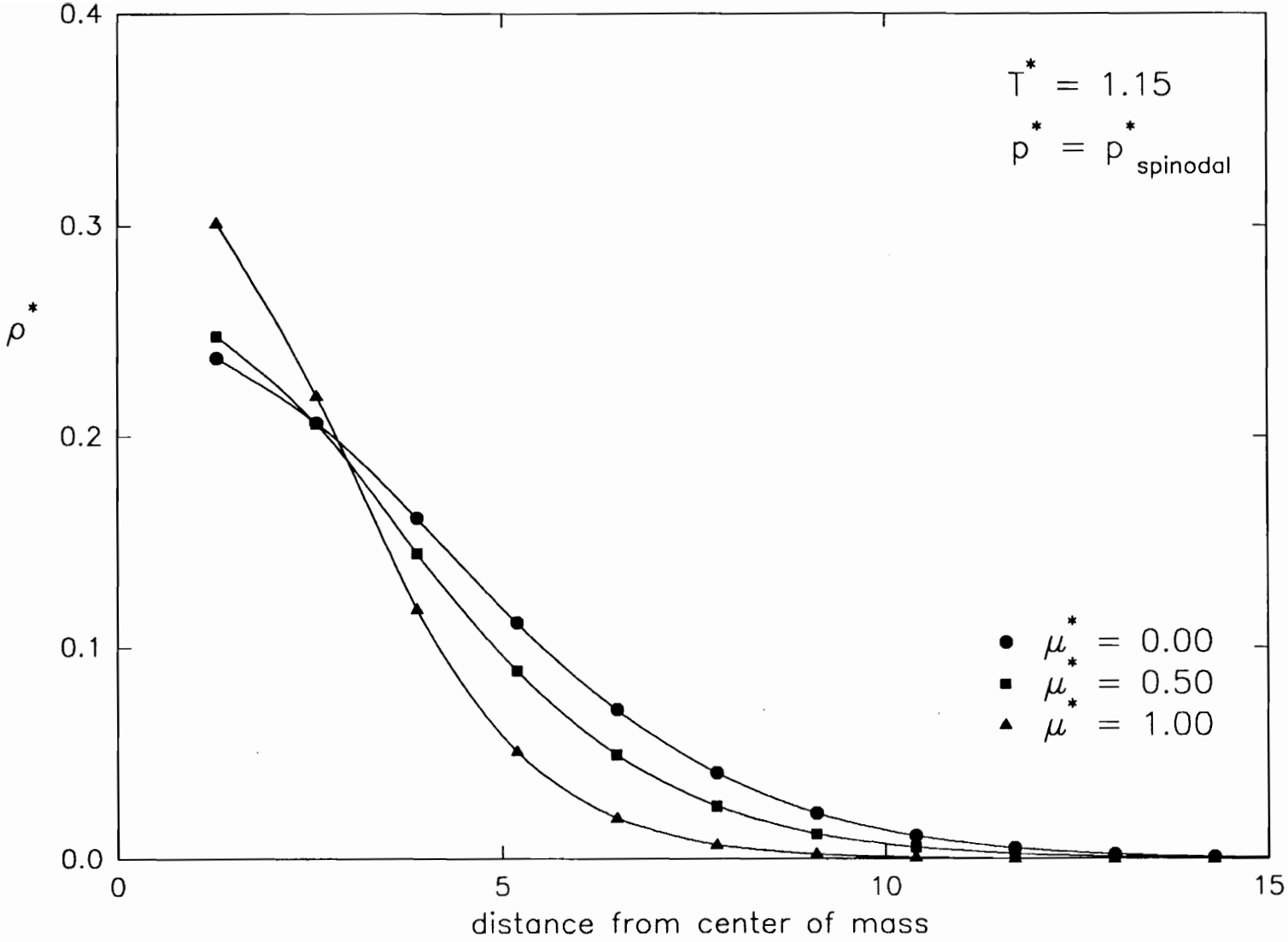


Figure 25. The density profile for the Stockmayer fluid at $\mu^* = 0.00, 0.50,$ and 1.00.

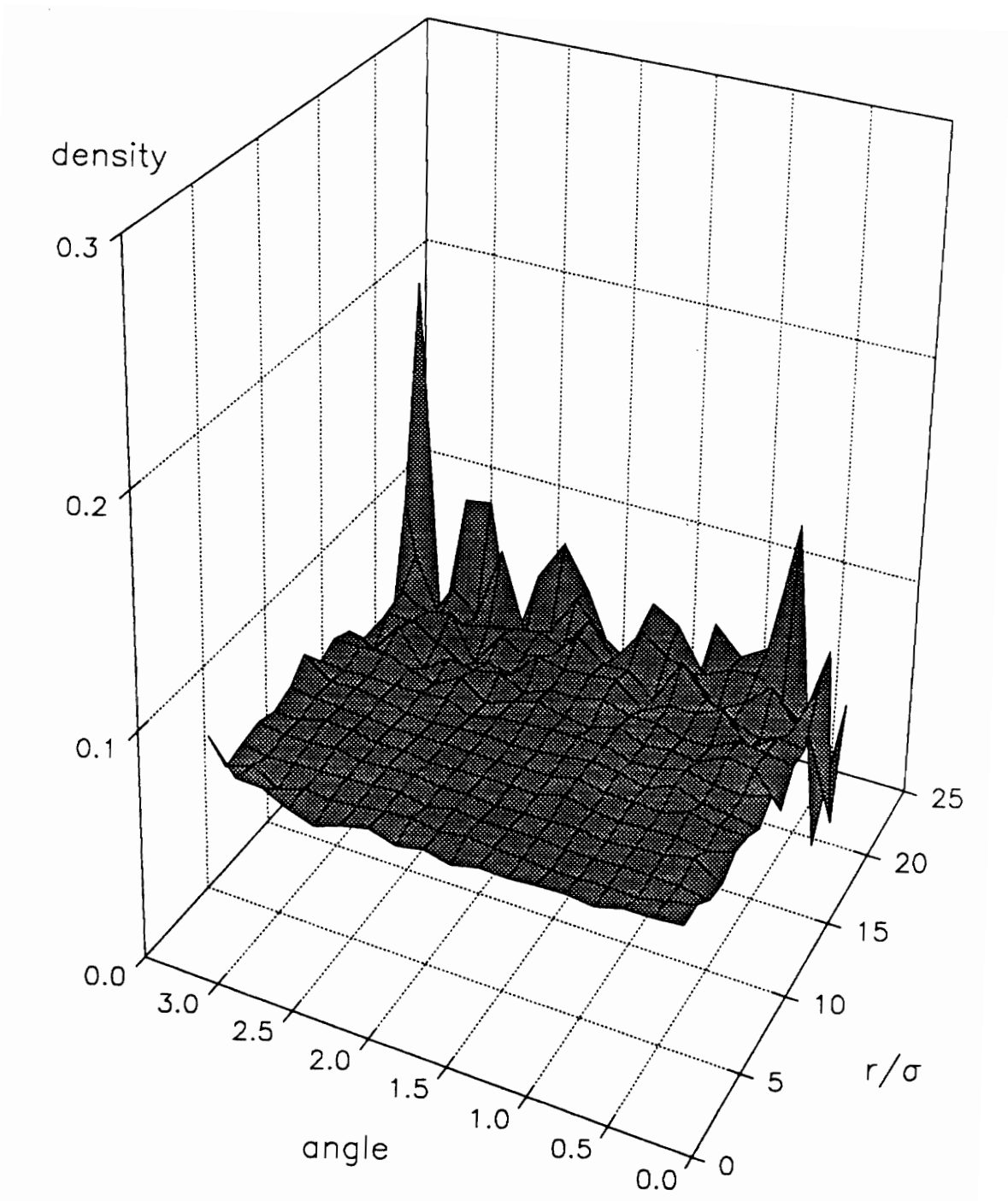


Figure 26. The orientational density profile for the Stockmayer fluid at the spinode for $T^* = 1.15$ and $\mu^* = 1.00$.

V. CONCLUSIONS

The Monte Carlo method is a useful tool in simulating equilibrium systems. Even though a supersaturated vapor and a supersaturated liquid are metastable states, the method allows the determination of their properties. Isotherms and cluster properties in the liquid-gas two-phase region of the LJ phase diagram have been calculated. The constant NpT ensemble is well suited for this type of study because the simulation does not allow the system to be in an unstable region. The best representation of the simulated pressure-volume behavior is by a truncated virial expression, for which empirical coefficients were determined. We have observed a system, whose initial state is a supersaturated gas with pressure greater than the spinode, transform into a uniform liquid via configurations which have a very dense cluster surrounded by gas. The number of particles in the critical cluster is hard to determine. It seems that not only the number of particles, but the structure of the cluster, determines if it is a critical nucleus.

The addition of the dipole-dipole interaction of the Stockmayer potential energy function changed many characteristics of the system. The spinodal points can be altered. The structure of the condensation cluster can be changed depending on the magnitude of the dipole monent. The density profile and surface characteristics changed with dipole strength. The cluster distribution was different for different dipole moments. Therefore, one can design a molecule to have desired properties.

Real fluids should be studied. Hopefully, the cluster properties of water and carbon dioxide will be calculated in the future. The spinodes of these molecules can be determined and compared with experiment.

VI. APPENDICES

Appendix A. Simulation Details

1. The Lennard-Jones Fluid

N	T^*	p^*	ρ^*	U^*	iterations[1]
729	1.25	0.090	0.113(5)	-0.925(51)	1494450
729	1.25	0.101	0.149(13)	-1.20(11)	4374000
729	1.25	0.103	0.171(13)	-1.38(10)	1800000
729	1.25	0.105	0.190(26)	-1.52(20)	6561000
729	1.25	0.099	0.461(39)	-3.19(22)	13122000
729	1.25	0.100	0.466(31)	-3.21(20)	10935000
729	1.25	0.102	0.471(28)	-3.25(17)	7290000
729	1.20	0.0500	0.0516(22)	-0.434(22)	2916000
729	1.20	0.0600	0.0659(31)	-0.550(29)	1458000
729	1.20	0.0700	0.0821(42)	-0.686(42)	1458000
729	1.20	0.0800	0.106(6)	-0.884(71)	2916000
729	1.20	0.0850	0.125(11)	-1.05(11)	4374000
729	1.20	0.0870	0.138(13)	-1.17(13)	7290000
729	1.20	0.0880	0.144(16)	-1.22(14)	5103000
729	1.20	0.0888	0.148(16)	-1.25(14)	9112500
729	1.20	0.054	0.492(30)	-3.40(17)	5103000
729	1.20	0.055	0.495(25)	-3.43(15)	5103000
729	1.20	0.060	0.510(21)	-3.51(14)	4374000
729	1.20	0.070	0.536(21)	-3.66(14)	4374000
729	1.20	0.080	0.544(16)	-3.71(11)	2916000
729	1.20	0.089	0.556(19)	-3.79(12)	2916000
512	1.15	0.0500	0.0556(33)	-0.478(37)	1024000
512	1.15	0.0600	0.0738(47)	-0.638(49)	3072000
512	1.15	0.0700	0.0977(70)	-0.849(74)	2048000
512	1.15	0.0750	0.112(9)	-0.973(88)	2713600
512	1.15	0.0754	0.121(17)	-1.06(16)	11776000
729	1.15	0.0100	0.00906(23)	-0.0778(39)	2187000
729	1.15	0.0200	0.0189(6)	-0.162(8)	2187000
729	1.15	0.0300	0.0296(10)	-0.253(11)	2187000
729	1.15	0.0400	0.0417(16)	-0.359(20)	2187000
729	1.15	0.0500	0.0564(22)	-0.487(30)	568620
729	1.15	0.0590	0.0714(25)	-0.611(32)	1458000
729	1.15	0.0700	0.0974(60)	-0.848(65)	4374000
729	1.15	0.0725	0.108(9)	-0.946(98)	4374000
729	1.15	0.0750	0.120(11)	-1.06(11)	11664000
729	1.15	0.0753	0.126(16)	-1.12(16)	8019000
729	1.15	0.0016	0.533(29)	-3.68(18)	29160000

729	1.15	0.0025	0.536(16)	-3.72(16)	7290000
729	1.15	0.0500	0.540(24)	-3.72(15)	9477000
729	1.15	0.0100	0.548(23)	-3.78(14)	4374000
729	1.15	0.0150	0.552(20)	-3.79(13)	11664000
729	1.15	0.0200	0.560[2]	-3.84[2]	[2]
729	1.15	0.0300	0.565(25)	-3.88(16)	5832000
729	1.15	0.0350	0.572(18)	-3.92(12)	8748000
729	1.15	0.0400	0.575(17)	-3.93(12)	13122000
729	1.15	0.0450	0.576(18)	-3.95(12)	17496000
729	1.15	0.0500	0.580(19)	-3.97(12)	8748000
729	1.15	0.0590	0.586(16)	-4.00(11)	8748000
729	1.15	0.0700	0.591(19)	-4.04(13)	17496000
1000	1.15	0.0500	0.0563(19)	-0.485(21)	2000000
1000	1.15	0.0600	0.0729(30)	-0.629(31)	4000000
1000	1.15	0.0700	0.0988(54)	-0.858(56)	4000000
1000	1.15	0.0750	0.120(9)	-1.06(9)	4000000
1000	1.15	0.0755	0.123(9)	-1.09(10)	12000000
5000	1.15	0.0600	0.0731(17)	-0.633(23)	5000000
5000	1.15	0.0750	0.122(7)	-1.09(9)	15000000
5000	1.15	0.0751	0.123(4)	-1.10(5)	12000000
729	1.00	0.0200	0.0230	-0.216	[2]
729	1.00	0.0241	0.0285	-0.264	[2]
729	1.00	0.0300	0.0382	-0.361	[2]
729	1.00	0.0350	0.0464	-0.438	[2]
729	1.00	0.0400	0.0567(3)	-0.548(40)	8748000
729	1.00	0.0448	0.0729(57)	-0.734(85)	13122000
729	1.00	-0.198	0.615(22)	-4.29(14)	17496000
729	1.00	-0.190	0.621(18)	-4.33(12)	10935000
729	1.00	-0.150	0.640(14)	-4.44(10)	17496000
729	1.00	-0.090	0.664(12)	-4.60(19)	17496000
729	1.00	-0.050	0.672(13)	-4.66(10)	17496000
729	1.00	-0.010	0.680(12)	-4.71(9)	17496000
729	1.00	0.0010	0.685(11)	-4.74(8)	17496000
729	1.00	0.0050	0.686(11)	-4.76(8)	4374000
729	1.00	0.0100	0.686(10)	-4.76(8)	4374000
729	1.00	0.0200	0.68728		
729	1.00	0.0241	0.690(13)	-4.77(9)	13122000
729	1.00	0.0460	0.694(7)	-4.80(6)	729000
729	1.00	0.0500	0.695(7)	-4.81(6)	4374000

729	0.90	0.0123	0.0151(5)	-0.153(9)	8748000
729	0.90	0.0130	0.0161(5)	-0.162(9)	4374000
729	0.90	0.0150	0.0190(6)	-0.192(11)	4374000
729	0.90	0.0200	0.0269(11)	-0.275(17)	17496000
729	0.90	0.0250	0.0363(17)	-0.379(25)	17496000
729	0.90	0.0270	0.0407(20)	-0.430(31)	17496000
729	0.90	0.0290	0.0459(25)	-0.495(43)	17496000
729	0.90	0.0293	0.0469(25)	-0.508(49)	17496000
729	0.90	-0.376	0.660(15)	-4.64(11)	17496000
729	0.90	-0.375	0.659(16)	-4.64(11)	17496000
729	0.90	-0.370	0.661(15)	-4.66(11)	23969520
729	0.90	-0.350	0.675(14)	-4.74(10)	17496000
729	0.90	-0.300	0.688(13)	-4.83(9)	17496000
729	0.90	0.0140	0.742(9)	-5.19(8)	8748000
729	0.75	0.0023	0.00315(8)	-0.036(3)	8748000
729	0.75	0.0050	0.0071(2)	-0.083(6)	8474000
729	0.75	0.0080	0.0119(4)	-0.141(9)	8474000
729	0.75	0.0100	0.0155(5)	-0.185(12)	17496000
729	0.75	0.0130	0.0215(10)	-0.267(24)	4374000
729	0.75	0.0140	0.0239(11)	-0.303(28)	9637380
729	0.75	0.0150[3]	0.813	-5.78	52000000

[1]This number does not include the pre-equilibrium steps. It is the number of configurations that the averages were carried over.

[2]This information is not available.

[3]This run is above the spinodal point. It was used to trap the condensing cluster. The run started as a gas and ended as a liquid.

2. The Stockmayer Fluid

N	T^*	μ^*	p^*	ρ^*	U^*	iterations
729	1.25	1.00	0.0300	0.0273(9)	-0.288(18)	867569
729	1.25	1.00	0.0400	0.0381(13)	-0.404(22)	3324240
729	1.25	1.00	0.0500	0.0508(22)	-0.538(32)	7574310
729	1.25	1.00	0.0600	0.0660(32)	-0.697(42)	11999340
729	1.25	1.00	0.0700	0.0852(51)	-0.903(64)	10935000
729	1.25	1.00	0.0750	0.0986(72)	-1.05(9)	8748000
729	1.25	1.00	0.0801	0.122(14)	-1.32(19)	26244000
729	1.25	1.00	-0.0064	0.540(32)	-4.31(22)	30618000
729	1.25	1.00	-0.0060	0.541(30)	-4.32(20)	30618000
729	1.25	1.00	0.0000	0.549(23)	-4.37(17)	17496000
729	1.25	1.00	0.0200	0.570(20)	-4.52(14)	8748000
729	1.25	1.00	0.0400	0.584(17)	-4.61(13)	8748000
729	1.25	1.00	0.0500	0.588(16)	-4.63(12)	8748000
729	1.15	1.00	0.0300	0.0310(11)	-0.353(20)	6925500
729	1.15	1.00	0.0400	0.0448(20)	-0.514(31)	6554159
729	1.15	1.00	0.0500	0.0625(37)	-0.722(54)	2901420
729	1.15	1.00	0.0550	0.0773(58)	-0.914(90)	4359420
729	1.15	1.00	0.0565	0.0822(67)	-0.98(10)	13107420
729	1.15	1.00	0.0569	0.0836(72)	-0.99(11)	16387920
729	1.15	1.00	-0.150	0.588(23)	-4.73(16)	29152710
729	1.15	1.00	-0.125	0.603(20)	-4.83(14)	13114710
729	1.15	1.00	-0.100	0.620(17)	-4.96(13)	17496000
729	1.15	1.00	-0.075	0.633(15)	-5.05(12)	8748000
729	1.15	1.00	-0.050	0.642(13)	-5.12(10)	17496000
729	1.15	1.00	-0.030	0.647(13)	-5.15(10)	17496000
729	1.15	1.00	-0.020	0.648(13)	-5.15(10)	18225000
729	1.15	1.00	-0.010	0.652(13)	-5.19(11)	21735000
729	1.15	1.00	0.000	0.654(13)	-5.20(10)	18225000
729	1.15	1.00	0.010	0.658(13)	-5.23(10)	18225000
729	1.15	1.00	0.020	0.662(11)	-5.26(9)	18225000
729	1.15	1.00	0.030	0.663(13)	-5.27(10)	18225000
729	1.15	1.00	0.040	0.666(12)	-5.29(9)	18225000
729	1.15	1.00	0.050	0.667(12)	-5.30(10)	10935000
729	1.15	0.50	0.050	0.0567(25)	-0.497(28)	8748000
729	1.15	0.50	0.060	0.0740(40)	-0.654(42)	8748000
729	1.15	0.50	0.065	0.0856(53)	-0.758(56)	8748000
729	1.15	0.50	0.070	0.101(8)	-0.903(78)	8748000
729	1.15	0.50	0.072	0.109(9)	-0.978(99)	13122000

729	1.15	0.50	0.0725	0.111(10)	-1.00(11)	13122000
729	1.15	0.50	0.0727	0.115(15)	-1.04(16)	17496000
729	1.15	0.50	0.0729	0.117(15)	-1.06(16)	17496000
729	1.15	0.50	-0.010			
729	1.15	0.50	0.000	0.554(29)	-3.87(18)	17496000
729	1.15	0.50	0.010	0.567(20)	-3.95(13)	17496000
729	1.15	0.50	0.040	0.589(16)	-4.09(11)	8748000
729	1.15	0.50	0.070	0.602(13)	-4.17(9)	8748000
729	1.15	0.75	0.060	0.0778(46)	-0.743(55)	8694000
729	1.15	0.90	0.060	0.0849(57)	-0.905(78)	13122000

Appendix B. NpT ensemble computer program

```
!**********************************************************************
! Program: lj.f
! Author: v. paul gregory (paul@simulate.chem.vt.edu)
! Date: Jan 07, 1994
! Purpose:
!        this program implements the monte carlo technique to reach
! equilibrium of a system of lennard-jones spheres in the npt ensemble.
! this is a fast version of the program ljnpt.f, which calculates all
! the cluster statistics as well as the average energy and density.
!
! version: 2.2
! system: AIX
! language: XL Fortran 2.3
!
! full calculation of percolation probability, density,
!     cluster distribution, cluster energies, radius of gyration,
!     density profile, maximun cluster radius, cluster neighbors,
!     radial distribution function
!     all distances are squared.
!     cluster calculations are done with cells allocated in the
!            cluster subroutine with malloc()
!
!**********************************************************************
@process ddim
      program lj
      implicit real*8 (a-h,o-z)
      parameter (pi=3.1415272654)
      parameter (maxbin=50,delr=0.10)

      logical do_energy/F/, do_coordi/F/, do_denpro/F/, do_radgyr/F/
     :     , do_raddis/F/, full_calc/F/
      namelist /nl1/ ntrig,Num_Part,nmc,trans,f1,f2,cut,temp,den,
     :               presc,nea,infil,full_calc
      namelist /nl2/ seed

      namelist /nl3/ do_energy,do_coordi,do_denpro,do_radgyr,do_raddis
      integer trt(3,27) / 0,0,0, 0,1,0, 0,0,1, 0,1,1, 0,-1,0, 0,0,-1,
     :     0,-1,-1, 0,-1,1, 0,1,-1,
     :     1,0,0, 1,1,0, 1,1,1, 1,0,1, 1,-1,1, 1,-1,0, 1,-1,-1, 1,0,-1,
     :     1,1,-1,
     :     -1,0,0, -1,-1,0, -1,0,-1, -1,1,0, -1,0,1, -1,1,1, -1,-1,1,
     :     -1,1,-1, -1,-1,-1/
      pointer ( addr_coor, coor )
      pointer ( addr_pot_lj, pot_lj )
      pointer ( addr_work, work )
      pointer ( addr_sig, sig )
      pointer ( addr_itmp, itmp(1) )  ! temporary integer storage
```

```
pointer ( addr_ipccnt, ipccnt ) ! cluster calc storage
pointer ( addr_iccnt, iccnt )
pointer ( addr_aclus, aclus )
pointer ( addr_taclus, taclus )
pointer ( addr_neigh, neigh )
pointer ( addr_nnei, nnei )
pointer ( addr_neic, neic )
pointer ( addr_aneic, aneic )
pointer ( addr_aneicp, aneicp )
pointer ( addr_eclus, eclus )
pointer ( addr_epclus, epclus )
pointer ( addr_rgyro, rgyro )
pointer ( addr_ipart, ipart )
pointer ( addr_iclus, iclus )
pointer ( addr_jclus, jclus )
pointer ( addr_vden, vden )
real*8 coor(Num_Part,3),xwt(3),xwti(3)
integer*4 nged(125),ngoo(maxbin)
real*8 ged(50),goo(maxbin)
real*8 pot_lj(Num_Part,Num_Part),trp(8,3), trm(8,3)
real seed
integer*4 IPCCNT(Num_Part), ICCNT(Num_Part)
real*8 ACLUS(Num_Part), TACLUS(Num_Part)
integer neigh(Num_Part), nnei(Num_Part), neic(Num_Part)
real*8 ANEIC(Num_Part), ANEICP(Num_Part)
real*8 eclus(Num_Part), epclus(Num_Part), rgyro(Num_Part)
integer ipart(Num_Part), iclus(Num_Part), jclus(Num_Part)
integer vden(100)
character*30 outfil
character*30 infil
logical imv
type iar_date
   sequence
   integer*4 iday
   integer*4 imonth
   integer*4 iyear
end type
type(iar_date) idat_str
type iar_time
   sequence
   integer*4 ihr
   integer*4 imin
   integer*4 isec
end type
type(iar_time) itim_str
type std
   sequence
   real*8 ene
```

```
         real*8 den
      end type
      type(std) sig(1)

      type tmp_ene
         sequence
         real*8 E6
         real*8 E12
      end type
      type (tmp_ene) work(Num_Part*8*2)

      data trp/3*0.0,1.0,0.0,3*1.0,2*0.0,1.0,0.0,1.0,0.0,
     : 2*1.0,0.0,1.0,2*0.0,2*1.0,0.0,1.0/
      data ged/50*0.0/
      data nged/125*0/
      prv=0.
      kea = 0.
      call itime_(itim_str)
      call idate_(idat_str)
      seed=float(itim_str%isec * idat_str%iday)/
     :     float(idat_str%imonth)/pi+pi
      call srand(seed)

      idummy = iargc()
      if ( idummy == 0 ) then
         write(*,*) 'you have to input a file name'
         write(*,*) 'example:   ljnpt lj115d54'
         stop
      end if

      call getarg(1,outfil)
      len_out = index(outfil,' ')
      f1 = 0.05
      f2 = 0.05           ! scales for initial displacements and volume
changes
      open(unit=5,file=outfil(1:len_out)//'.data',status='old')
      read(5,NML=nl1)  ! Num_Part, steps, temp, pressure, etc.
      read(5,NML=nl2)  ! seed
      close(5)
      addr_coor = malloc ( %val(Num_Part*3*8) )
!  allocate cluster storage according to what is turned on
      if ( full_calc ) then
         addr_ipart = malloc ( %val(Num_Part*4) )
         addr_iclus = malloc ( %val(Num_Part*4) )
         addr_jclus = malloc ( %val(Num_Part*4) )
         addr_ipccnt = malloc ( %val(Num_Part*4) )
         addr_iccnt = malloc ( %val(Num_Part*4) )
         addr_aclus = malloc ( %val(Num_Part*8) )
         addr_taclus = malloc ( %val(Num_Part*8) )
         if ( do_coordi ) then
```

```
            addr_neigh = malloc ( %val(Num_Part*4) )
            addr_nnei = malloc ( %val(Num_Part*4) )
            addr_neic = malloc ( %val(Num_Part*4) )
            addr_aneic = malloc ( %val(Num_Part*8) )
            addr_aneicp = malloc ( %val(Num_Part*8) )
         end if
         if ( do_energy ) then
            addr_eclus = malloc ( %val(Num_Part*8) )
            addr_epclus = malloc ( %val(Num_Part*8) )
         end if
         if ( do_radgyr ) addr_rgyro = malloc ( %val(Num_Part*8) )
         if ( do_denpro ) then
            addr_vden = malloc ( %val(100*4) )
         end if
      end if
! vden() is the density profile:  100 shells, max.

c..Num_Part=no. of molecules
      n33=3*Num_Part
      xn3=dfloat(Num_Part)
      xn3s=xn3*xn3
      xnpr=dfloat(Num_Part*(Num_Part-1))/2.0
      if(ntrig <> 0) then
         len_in = index(infil,' ')-1
         open(10,access='sequential',file=infil(:len_in)//'.bin',
     :        form='unformatted',status='old')
         do i=1,Num_Part
            read(10,err=9801) (coor(i,k), k=1,3)
         end do
 9801    close(10)
         open(unit=5,file=infil(:len_in)//'.pun',status='old')
         if ( full_calc ) then
            read(5,809) (taclus(i), i=1,Num_Part)      ! full
         else
            read(5,809) (taclus, i=1,Num_Part)         ! fast
         end if
         read(5,*) ebar                                 ! average energy
         read(5,*) avdensity                           ! average density
         read(5,*) trans                                  ! cube length
         read(5,*) nprv                          ! # of previous mc steps
         read(5,*) nprvs                ! # of previous mc steps in stats
         read(5,*) nprvc                 ! # of previous calls to cluster
         read(5,*) nprvv                   ! # of previous volume changes
         read(5,*) step                      ! step size of displacements
         read(5,*) stepv                    ! step size of volume changes
         read(5,*) tjperc            ! # of previous percolation configs
         close(5)
```

```
         prv=dfloat(nprv)
         prvs=dfloat(nprvs)
         prvc=dfloat(nprvc)
         volume=trans**3  ! calc vol from trans (box size) for last run
         density=Num_Part/volume
         cut=trans/2.
         cut2=cut*cut
      end if
      if ( ntrig == 0 ) then      ! set up initial coordinates
         volume=(xn3/den)
         trans=volume**(1./3.)
         cut=trans/2.
         cut2=cut*cut
         density=den

!--------initial_coors
         X=TRANS/Num_Part**(1./3.)
         NEND=NINT(Num_Part**(1./3.))
         M=0
         DO II=1,NEND
            DO J=1,NEND
               DO K=1,NEND
                  M=M+1
                  COOR(M,1)=(K-1)*X
                  COOR(M,2)=(J-1)*X
                  COOR(M,3)=(II-1)*X
               END DO
            END DO
         END DO
         if ( m .lt. Num_Part ) then
            addr_itmp = malloc ( %val(Num_Part*4) )
            do II = m+1, Num_Part
 234           icell = int ( rand() * NEND**3 ) + 1
               if ( itmp(icell) .eq. 0 ) then
                  jcell = icell
                  ix = 0
                  iy = 0
                  iz = 0
                  do while ( jcell-NEND**2 .ge. 0 )
                     iz = iz + 1
                     jcell = jcell - NEND**2
                  end do
                  iz = iz + 1
                  do while ( jcell-NEND .ge. 0 )
                     iy = iy + 1
                     jcell = jcell - NEND
                  end do
                  iy = iy + 1
                  ix = jcell
                  m=m+1
```

```
                  coor(m,1)=(ix-1)*X+X/2.
                  coor(m,2)=(iy-1)*X+X/2.
                  coor(m,3)=(iz-1)*X+X/2.
                  itmp(icell) = 1
               else
                  goto 234
               end if
            end do
            call free( %val(addr_itmp) )
         end if
         step=f1*trans
         stepv=f2*trans
      end if

c..calculate initial energy and distributions
      addr_pot_lj = malloc ( %val( Num_Part*Num_Part*8 ) )

      enew=0.0
      nt1=0
      loop1: do i=2,Num_Part
         do kc=1,3
           xwt(kc)=coor(i,kc)
         end do
c..for each molecule, loop over all other molecules
         loop2: do j=1,i-1
            pot_lj(j,i)=0.0
            pot_lj(i,j)=0.0
            e6=0.0d0
            e12=0.0d0
            do k=1,3
               dir=1.0
               y=coor(j,k)-xwt(k)
               if (y > 0.0) then
                  dir= -1.0
               end if
               tmp=dir*trans
               do l=1,8          ! calc. 8 possible periodic translations
                  trm(l,k)=tmp*trp(l,k)
               end do
            end do
            loop3: do k8=1,8
               xwti(1)=coor(j,1)+trm(k8,1)
               xwti(2)=coor(j,2)+trm(k8,2)
               xwti(3)=coor(j,3)+trm(k8,3)
               roo=(xwt(1)-xwti(1))**2+(xwt(2)-xwti(2))**2+
     :                (xwt(3)-xwti(3))**2
               if (roo <= cut2) then
                  ro2 = (1.0d0/roo)**3
                  e6 = -4*ro2
                  e12 = 4*(ro2**2)
```

```
                  exit loop3
               end if
            end do loop3
            if ( (e6+e12 > -7.d0) .and. (e6+e12 <= 3.d0) ) then
               ned = int((e6+e12 + 7.d0)/0.2d0) + 1
               nged(ned) = nged(ned) + 1
               ged(ned)=ged(ned)+1.d0
               nt1=nt1+1
            end if
            pot_lj(j,i)=e6
            pot_lj(i,j)=e12
         end do loop2
      end do loop1
      do i=2,Num_Part
         do j=1,i-1
            enew = enew + pot_lj(j,i) + pot_lj(i,j)
         end do
      end do
      if (ntrig == 2)  then
         t=0.
         ttc=0.
         iattv=0
         nprvs=0
         nprvc=0
         nprvv=0
         tjperc=0
         ojperc=0
      else
         t=prvs
         ttc=prvc
         iattv=nprvv
      end if
      if (ntrig == 1) then
         ojperc=int(tjperc)
      end if

      open(unit=6,file=outfil(1:len_out)//'.out',status='new')
!-----initial_output
      if ( idat_str%imonth < 10 ) then     ! used to format the time &
date
         mon_len = 1
      else
         mon_len = 2
      end if
      if ( itim_str%imin < 10 ) then
         min_len = 1
      else
         min_len = 2
      end if
```

```
 if ( full_calc ) then
    WRITE(6,*) 'FULL CALCULATION, NPT ensemble'
 else
    WRITE(6,*) 'FAST CALCULATION, NPT ensemble'
 end if
 write(6,*)
 WRITE(6,*) 'JOB STARTED ON'
 WRITE(6,9040) idat_str%iday,idat_str%imonth,idat_str%iyear,
:      itim_str%ihr,itim_str%imin,itim_str%isec
 write(6,*)
 WRITE(6,*) 'PARAMETERS FOR THIS RUN:'
 if ( ntrig == 0 ) then
   write(6,*) 'Starting new run'
 end if
 if ( ntrig == 1 ) then
    write(6,*) 'Continuing previous run and averaging'
    write(6,*) '     data filename: ',INFIL(:len_in)
 end if
 if ( ntrig == 2 ) then
    write(6,*) 'Continuing previous run, starting averaging over'
    write(6,*) '     data filename: ',INFIL(:len_in)
 end if
 write(6,*)
 WRITE(6,*) '# particles ', N3
 WRITE(6,*) '# prev. steps ', NPRV
 WRITE(6,*) '# prev. steps in stats ', NPRVS
 WRITE(6,*) '# Monte Carlo steps', NMC
 WRITE(6,*) '# steps in variance calculation ', NEA
 WRITE(6,'('' Reduced constant temperature'',1f10.6)') TEMP
 WRITE(6,'('' Reduced Constant Pressure'',1f11.7)') PRESC
 WRITE(6,*) 'SEED',SEED
 WRITE(6,*) 'IRAD : ',IRAD,'  IRAP: ',IRAP
 WRITE(6,*) 'Reduced initial box size', TRANS
 WRITE(6,*) 'Reduced initial cutoff distance', CUT
 WRITE(6,*) 'Initial step size', STEP
 WRITE(6,*) 'Initial volume step size', STEPV
 IF ( NTRIG. EQ. 0 ) THEN
    WRITE(6,*) 'Initial reduced density', DEN
    WRITE(6,*) '     in this run, STEP=F1*TRANS ...'
    WRITE(6,*) '     fraction of box size ', F1
 end if
 write(6,*)
 write(6,*) 'This run includes:'
 write(6,*) '     average energy'
 write(6,*) '     average density'
 write(6,*) '     cluster distribution'
 write(6,*) '     percolation probability'
 if ( full_calc ) then
   if ( do_energy ) write(6,*) '     cluster energy'
   if ( do_coordi ) write(6,*) '     coordination number'
```

```
        if ( do_denpro ) write(6,*) '    density profile'
        if ( do_radgyr ) write(6,*) '    radius of gyration'
        if ( do_raddis ) write(6,*) '    radial distribution function'
        write(6,*) '    average number of clusters'
        write(6,*) '    average size of the largest non-percolating ',
     :     'cluster'
        write(6,*) '    average size of the largest percolating cluster'
      end if
      write(6,*)
      WRITE(6,*)'ORIG. NGED: ED=-7 TO +3, INC. OF 0.2, COUNT= ',NT1
      WRITE(6,'(10i7)')(NGED(I),I=1,50)
 9040 FORMAT(5X,'DD/MM/YY',5X,I2,'/',I<mon_len>,'/',I4,
     :       5X,'HH:MM:SS',5X,I2,':',I<min_len>,':',I2)

c...do cluster distribution for initial configuration
      if ( full_calc ) then
        call cluster
(Num_Part,ncells,cell_len,trans,coor,trt,iclus,jclus,
     :     iperc,num_clus,ipart,iccnt,ipccnt,do_energy,eclus,epclus,
     :     do_radgyr,rgyro,nmax,pnmax,do_denpro,vden)
        if (iperc == 1) write(6,*) 'initial config. percolates.'
        write(6,*)'particles in clusters:'
        iend = 0
        do j = 1 , num_clus
           istart = iend + 1
           iend = istart + iclus(j) -1
           write(6,609) j,(ipart(i),i=istart,iend)
        end do
        if (irad <> 2 .and. ntrig <> 1) then
           ebar=enew/xn3
           avdensity=density
           do ii=1,Num_Part
              taclus(ii)=jclus(ii)
              aclus(ii)=jclus(ii)
           end do
           write(6,*) 'averages set equal to initial values'
        else
           write(6,*) 'averages continued from previous run'
           tnumcl=num_clus
        end if
      end if

      open(unit=8,file=outfil(1:len_out)//'.pnn',status='new')
      if ( ntrig <> 1 ) then
         if ( full_calc ) then
            write(8,994) nprv,enew/xN3,ebar,real(jperc),tjperc,
     :           density,avdensity
         else
            write(8,994) nprv,enew/xn3,ebar,density,avdensity
         end if
```

```
      else
         if ( full_calc ) then
            write(8,994) nprv,enew/xn3,ebar,real(jperc)/tc,
     :      tjperc/ttc,density,avdensity
         else
            write(8,994) nprv,enew/xn3,ebar,density,avdensity
         end if
      end if
      write(6,*) '                    initial     average     constant'
      write(6,210) enew/xn3,ebar
      write(6,211) pres,avpres,presc
      write(6,213) density,avdensity
      if (ttc > 0) then
         write(6,212) float(iperc),tjperc/ttc
      else
         write(6,212) float(iperc)
      end if
      addr_sig = malloc ( %val(nmc/nea*8*2) )    ! vector for std
c*********************************************************************
c...ready for monte carlo
      write(6,219)
      iac=0        !  counter of accepted particle moves for a block
      iacc=0       !  counter of total accepted partilce moves
      iacv=0       !  counter of accepted volume changes for a block
      iaccv=0      !  counter of total accepted volume changes
      iatv=0       !  counter of attempted volume changes for a block
      xn31=xn3-1
      jsig=0
      tw=0.0       ! wood's block counter
      m=0
      iperc=0
      addr_work = malloc ( %val(Num_Part * 8 * 2) )
      main: do ns=1,nmc     ! this is the main monte carlo loop
      isw = 0
      kea = kea + 1
      if ( kea == nea ) then
        kea = 0
        jsig=jsig+1
        isw=1
      end if
c...pick a molecule...we are cycling thru them
      m=m+1          !  m is the particle number
      if(m > Num_Part) m=1
c..try separate step size for each molecule
      r=rand()
      x=step*(2.0*r-1.0)
      r=rand()
      y=step*(2.0*r-1.0)
      r=rand()
      z=step*(2.0*r-1.0)
```

```
c..generate its new coordinates
      xn=coor(m,1)+x
      if(xn > trans) xn=xn-trans
      if(xn < 0.) xn=xn+trans
      yn=coor(m,2)+y
      if(yn > trans) yn=yn-trans
      if(yn < 0.) yn=yn+trans
      zn=coor(m,3)+z
      if(zn > trans) zn=zn-trans
      if(zn < 0.) zn=zn+trans
      xwt(1)=xn
      xwt(2)=yn
      xwt(3)=zn
c..loop over all other molecules to get its new energy
      de=0.0
      ep=0.0
      do k=1,Num_Part
         if (k == m) cycle
         E6 = 0.0d0
         E12 = 0.0d0
         do kk=1,3
            dir=1.0
            y=coor(k,kk)-xwt(kk)
            if (y > 0.0) then
               dir= -1.0
            end if
            tmp=dir*trans
            do l=1,8
               trm(l,kk)=tmp*trp(l,kk)
            end do
         end do
         do k8=1,8
            xwti(1)=coor(k,1)+trm(k8,1)
            xwti(2)=coor(k,2)+trm(k8,2)
            xwti(3)=coor(k,3)+trm(k8,3)
            roo=(xwt(1)-xwti(1))**2+(xwt(2)-xwti(2))**2+
     :           (xwt(3)-xwti(3))**2
            if (roo <= cut2) then
               ro2=(1.0d0/roo)**3
               e6=-4*ro2
               e12=4*(ro2**2)
               exit
            end if
         end do
         work(k)%E6=e6
         work(k)%E12=e12
         if (k < m) then
            de=de+work(k)%e6-pot_lj(k,m)
            de=de+work(k)%e12-pot_lj(m,k)
         else
```

```
            de=de+work(k)%e6-pot_lj(m,k)
            de=de+work(k)%e12-pot_lj(k,m)
         end if
      end do

c..test for acceptance
      imv = .false.
      if (de > 0.0) then
         bf = dexp(-de/temp)
         r = rand()
         if (r < bf) imv = .true.       ! move accepted
      else
         imv = .true.                   ! move accepted
      end if
      if (imv ) then
         do lm = 1,m-1
            pot_lj(lm,m) = work(lm)%E6
            pot_lj(m,lm) = work(lm)%E12
         end do
         do lm=m+1,Num_Part
            pot_lj(m,lm) = work(lm)%E6
            pot_lj(lm,m) = work(lm)%E12
         end do
         enew=enew+de
         coor(m,1)=xwt(1)          !  replace the coordinates
         coor(m,2)=xwt(2)
         coor(m,3)=xwt(3)

         iac=iac+1
      end if
c. update energy averages
      t=t+1.        ! attempt anything counter
      ebar=( t*ebar- ebar + enew/xn3 ) / t
      tw=tw+1.             ! wood's block counter
      ebr2 = ( tw*ebr2 - ebr2 + enew/xn3 ) / tw
c...end of molecule move
c. attempt volume change
      if ( m == Num_Part ) then
         iatv=iatv+1
         transn = trans + stepv*(2.*rand() -1.0 )
         volumen = transn**3

         e6=0.0
         e12=0.0
         do i=1,Num_Part-1
            do j=i+1,Num_Part
               e6= e6 + pot_lj(i,j)
```

```
                e12= e12 + pot_lj(j,i)
             end do
          end do

          dev = e12*((trans/transn)**12-1) + e6*((trans/transn)**6-1)

          de = dev + presc*(volumen-volume) -
     :         Num_Part*temp*dlog(volumen/volume)

c..test for acceptance
          imv=.false.
          if (de > 0.0) then
             bf=dexp(-de/temp)
             r=rand()
             if (r < bf) imv=.true.   ! move accepted
          else
             imv=.true.                ! move accepted
          end if

          if ( imv ) then
c..update coor, pot_lj, and enew
             tnot = transn/trans

             do i=1,Num_Part
                coor(i,1)=coor(i,1)*tnot
                coor(i,2)=coor(i,2)*tnot
                coor(i,3)=coor(i,3)*tnot
             end do

             tt12 = (trans/transn)**12
             tt6 = (trans/transn)**6

             do i=1,Num_Part-1
                do j=i+1,Num_Part
                   pot_lj(i,j) = pot_lj(i,j)*tt6
                   pot_lj(j,i) = pot_lj(j,i)*tt12
                end do
             end do

             enew = enew + dev
             trans=transn
             cut2=trans*trans/4
             volume=volumen
             density=Num_Part/volume

             iacv=iacv+1

          end if

c.    update averages for volume move
```

```
         t=t+1                    ! attempt anything counter
         ebar = ( t*ebar - ebar + enew/xn3 ) / t
         tw=tw+1.                 ! wood's block counter
         ebr2 = ( tw*ebr2 - ebr2 + enew/xn3 ) / tw

         bdensity= ( iatv*bdensity - bdensity + density ) / iatv

      end if

c.    end of volume change
c. do cluster statistics
      if ( m == Num_Part .or. m == int(Num_Part/2) .and. full_calc) then

         call cluster
(Num_Part,ncells,cell_len,trans,coor,trt,iclus,jclus,
     :        iperc,num_clus,ipart,iccnt,ipccnt,do_energy,eclus,epclus,
     :        do_radgyr,rgyro,nmax,pnmax,do_denpro,vden)

c...calculate the reduced average cluster size.  hosen & kopelman
         racs = 0.0
         do i=1,Num_Part
            racs= racs + jclus(i)*i*i
         end do
         racs = racs / Num_Part
c...end calculate the racs
         tracs = tracs + racs
c....test for percolation
         if (iperc == 1) then    ! don't INCLUDE in clu.dis. or clu.ene.
            jperc=jperc+1
            avenmaxp=avenmaxp+nmax
            aneicp(nmax)=aneicp(nmax)+neic(nmax)
            neic(nmax)=0
         else
            avenmax=avenmax+nmax
         end if

c....update the cluster averages (distribution,energy,#)
         tc=tc+1                  ! call cluster counter, block
         qc1=(tc-1)/tc
         qc2=1.0d0/tc
         do ii=1,Num_Part
            aclus(ii) = qc1*aclus(ii) + qc2*jclus(ii) ! block clus dist
            iccnt(ii) = iccnt(ii) + jclus(ii) ! total jclus
            aneic(ii) = aneic(ii) + neic(ii) ! total cluster neighbors
         end do
         avnumcl=qc1*avnumcl+qc2*num_clus

      end if  ! end of cluster calculations
```

```
 if (isw == 1) then
    if ( real(iac)/real(nea)>0.30 .and. step<trans/3 ) then
       step=step*1.10
    else
       if ( step >= 0.30 ) then
          step=step*0.90
       end if
    end if
    iacc=iacc+iac
    iac=0
    if ( real(iacv)/real(iatv) > 0.30 ) then
       stepv=stepv*1.10
    else
       stepv=stepv*0.90
    end if
    iaccv=iaccv+iacv
    iacv=0
    iattv=iattv+iatv
    tv0=float(iattv-iatv)/float(iattv)
    tv1=float(iatv)/float(iattv)
    avdensity=tv0*avdensity+tv1*bdensity
    iatv=0
    tjperc=tjperc+jperc
    ttc=ttc+tc              ! call cluster counter, total
    tq0=(ttc-tc)/(ttc)
    tq1=tc/(ttc)
    do ii=1,Num_Part
       taclus(ii)=tq0*taclus(ii)+tq1*aclus(ii)
    end do
    tnumcl=tq0*tnumcl+tq1*avnumcl
    if ( full_calc ) then
       write(8,994) ns+nprv,ebr2,ebar,real(jperc)/tc,
:           tjperc/ttc,bdensity,avdensity
       write(6,209) ebr2,jperc/tc
    else
       write(8,994) ns+nprv,ebr2,ebar,
:           bdensity,avdensity
       write(6,209) ebr2
    end if
    open(unit=9,file=outfil(1:len_out)//'.009',status='unknown')
    write(9,*) ns,' out of ',nmc,'  block den',bdensity
    close(9)
    sig(jsig)=std(ebr2,bdensity)
    tw=0.0
    tc=0.0
    jperc=0
    avnumcl=num_clus
    do ii=1,Num_Part
       aclus(ii)=0
    end do
```

```
         bdensity=density
         ebr2=enew/xn3
         open(unit=11,file=outfil(1:len_out)//'.kill',
     :         iostat=ios,status='old')
         if ( ios .eq. 0 ) then
            write(6,*) 'program killed by user'
            exit main
         end if
       end if
       end do main
c*****************************************************************************

       if ( full_calc ) then
         do ii=1,Num_Part
            a2=1./ii
            if (iccnt(ii)>0) then
               acc=1./iccnt(ii)
               rgyro(ii)=rgyro(ii)*acc
               eclus(ii)=eclus(ii)*acc*a2
               aneic(ii)=aneic(ii)*acc*a2
            end if
            if (ipccnt(ii)>0) then
               apcc=1./ipccnt(ii)
               epclus(ii)=epclus(ii)*apcc*a2
               aneicp(ii)=aneicp(ii)*apcc*a2
            end if
         end do

         write(6,*)'particles in clusters:'
         do 637 i=1,num_clus
 637        write(6,609) i,(ipart(i,j), j=1,iclus(i))

          maxclus=1
          do ii=1,num_clus
             if (iclus(ii)>iclus(maxclus)) maxclus=ii
          end do
          nmax=iclus(maxclus)

          do j=1,iclus(maxclus)
            jj=ipart(maxclus,j)
            write(9,998) (coor(k,jj), k=1,3)
          end do

          do i=1,Num_Part
            write(6,609) i,(nnei(i,j),j=1,neigh(i))
          end do
          write(6,*)
```

```
         if ( do_raddis ) then
           const = 4.0 * pi * density / 3.0
           do bin = 1,maxbin
              rlow = real ( bin -1 ) * delr
              rupp = rlow + delr
              xideal = const * ( rupp**3 - rlow**3 )
              goo(bin) = real(ngoo(bin)) / real(jsig) /
     :             real(Num_Part) / xideal
           end do
         end if
       end if ! full_calc
       sum=0.0
       do ii=1,jsig
         sum=sum+(sig(ii)%ene-ebar)**2
       end do
       var_ene=sum/jsig                                    ! variance in
energy
       sdev_ene=dsqrt(var_ene)                             ! std in energy
       sum = 0.0
       do ii = 1, jsig
          sum = sum + (sig(ii)%den-avdensity)**2
       end do
       var_den = sum/jsig
       sdev_den = dsqrt(var_den)

       write(6,*) 'finial simulation results:'
       write(6,'('' energy'',1f15.10)') ebar
       write(6,'('' std and var'',2f15.10)') sdev_ene,var_ene
       write(6,'('' last energy'',1f15.10)') enew/Num_Part
       write(6,*)
       write(6,'('' density'',1f15.10)') avdensity
       write(6,'('' std and var'',2f15.10)') sdev_den,var_den
       write(6,'('' last density'',1f15.10)') density
       write(6,*)
       write(6,*) 'particle displacement results'
       write(6,'(2i10,5x,1f5.2,''%'')') ns,iacc,float(iacc)/ns*100
       write(6,*)
       write(6,*) 'volume change results'
       write(6,'(5x,2i10,5x,1f5.2,''%'')') iattv,iaccv,
     :     float(iaccv)/iattv*100
       write(6,*)
       write(6,*) 'final step size:'
       write(6,1029) step
       write(6,*) 'final volume step size:'
       write(6,1029) stepv
       write(6,*)
```

```
 if ( full_calc ) then
   write(6,*) 'average number of clusters:' ,tnumcl
   write(6,*) 'reduced average cluster size',tracs/ttc
   write(6,*)
   write(6,*) 'average non-perc cluster distribution (size wise)'
   write(6,809) (taclus(i),i=1,Num_Part)
   write(6,*)
   if ( do_energy ) then
      write(6,*) 'average non-percolating cluster energy'
      write(6,809) (eclus(i),i=1,Num_Part)
      write(6,*)
   end if
   if ( do_coordi ) then
      write(6,*) 'average non-perc # neighbors for clusters'
      write(6,809) (aneic(i),i=1,Num_Part)
      write(6,*)
   end if
   write(6,*) 'iccnt'
   write(6,811) (iccnt(i), i=1,Num_Part)
   write(6,*)
   write(6,*) 'percolation results (ratio)'
   write(6,*) int(tjperc)
   write(6,'(f15.7)') tjperc/ttc
   write(6,*)
   if (tjperc > 0.0) then
     write(6,*) 'average perc. cluster distribution (size wise)'
     if (int(tjperc)<>0 .or. int(tjperc-ojperc)<>0 ) then
       write(6,809) (ipccnt(i)/(tjperc-ojperc),i=1,Num_Part)
     end if
     write(6,*)
     if ( do_energy ) then
        write(6,*) 'average percolating cluster energy'
        write(6,809) (epclus(i), i=1,Num_Part)
        write(6,*)
     end if
     if ( do_coordi ) then
        write(6,*) 'average # perc neighbors for clusters'
        write(6,809) (aneicp(i),i=1,Num_Part)
     end if
   end if
   if ( do_denpro ) then
      write(6,*) 'density profile data'
      write(6,*) '    r          density        neighbors    '
      do i=1,100
         if (vden(i)>1e-6) then
           r = i*1.3
           write(6,'(3f15.8)') r,vden(i)/(ttc-nprvc-tjperc+ojperc)/
:            (4./3.*pi*( 2.197 - 5.07*r + 3.9*r*r ))
         end if
      end do
```

```
            write(6,*)
          end if
          write(6,*) 'average max non-percolating cluster size',
     :        avenmax/(ttc-nprvc- tjperc+ojperc)
          if (tjperc>0) then
            write(6,*) 'average percolating cluster size',avenmaxp/
     :           (tjperc-ojperc)
          end if
          if ( do_raddis ) then
             write(6,*)'average radius of gyration : rms'
             write(6,610) (rgyro(m), m=1,Num_Part)
          end if

          if ( do_raddis ) then
             write(6,819)
          end if

        end if ! full_calc

        write(6,*) 'final step size :'
        write(6,1029) step

        write(6,*) 'last configuration energy',enew/xn3
c.. test energy scaling
        do  kc=1,50
           nged(kc)=0
        end do
        enew=0.0
        nt1=0
        do i=2,Num_Part
           do kc=1,3
              xwt(kc)=coor(i,kc)
           end do
c..for each molecule, loop over all other molecules
           do j=1,i-1
              pot_lj(j,i)=0.0
              do k=1,3
                 dir=1.0
                 y=coor(j,k)-xwt(k)
                 if (y>0.0) then
                    dir= -1.0
                 end if
                 tmp=dir*trans
                 do l=1,8
                    trm(l,k)=tmp*trp(l,k)
                 end do
              end do
              do k8=1,8
                 xwti(1)=coor(j,1)+trm(k8,1)
                 xwti(2)=coor(j,2)+trm(k8,2)
```

```
                xwti(3)=coor(j,3)+trm(k8,3)
                ro2=(xwt(1)-xwti(1))**2+(xwt(2)-xwti(2))**2+
     :              (xwt(3)-xwti(3))**2
                if (ro2>cut2) then
                   e11=0.0
                else
                   ro2=(1./ro2)**3
                   e11=4*(ro2**2-ro2)
                   exit
                end if
             end do
             pot_lj(j,i)=e11
          end do
          do j=1,i-1
             enew=enew+pot_lj(j,i)
          end do
       end do
       write(6,*) enew/xn3
c.     end of test energy
       close(8)                    ! .pnn file
       close(6)                    ! .out file
       open(10,access='sequential',file=outfil(:len_out)//'.bin',   !
write coors
      :     form='unformatted',status='new')
       do i=1,Num_Part
          write(10,err=9802) (coor(i,k), k=1,3)
       end do
 9802  close(10)
       open(unit=7,file=outfil(:len_out)//'.pun',status='new')      !
write stats
       if ( full_calc ) then
           write(7,'(5f14.7)') (taclus(i), i=1,Num_Part)
           continue
       else
          write(7,'(5f14.7)') (rdummy, i=1,Num_Part)
       end if
       write(7,*) ebar
       write(7,*) avdensity
       write(7,*) trans
       write(7,*) nprv+nmc
       write(7,*) nprvs+nmc
       write(7,*) int(ttc)
       write(7,*) iattv
       write(7,*) step
       write(7,*) stepv
       write(7,*) tjperc
       close(7)
```

```
 9     format(i5,5x,i10)
 19    format(3f10.5)
 109   format(i5,f10.5,f10.5)
 209   format(5f15.5)
 210   format(2x,'energy:',5x,2f15.5)
 211   format(2x,'presure:',4x,3f15.5)
 213   format(2x,'density:',4x,2f15.5)
 212   format(2x,'percolation:',2f15.5,/)
 219   format(3x,'monte carlo')
 608   format(10i7)
 609   format(i4,70(t10,15i4,/))
 610   format(10f7.3)
 767   format(f16.6)
 809   format(5f14.7)
 811   format(5i12)
 819   format(//,2x,'radial distribution functions',//)
 829   format(//,2x,'o--o number counted= ',2f10.3,//)
 839   format(//,2x,'o--h number counted= ',2f10.3,//)
 849   format(//,2x,'dimerization energies',//)
 859   format(//,2x,'bonding energies',//)
 989   format(f15.9)
 993   format(8f10.6)
 994   format(2x,i8,2x,2f9.4,2x,4f9.5)
 995   format(5i15)
 998   format(3f15.9)
 999   format(2i10,f10.5,2x,i10)
 1029  format(5x,2f10.6)
       stop
       end
c-------------------------------------------------------------------------

       subroutine cluster (Num_Part,ncells,cell_len,trans,xox,
      :     trt,iclus,jclus,iperc,num_clus,ipart,
      :     iccnt,ipccnt,do_energy,eclus,epclus,
      :     do_radgyr,rgyro,nmax,pnmax,do_denpro,vden)
       implicit real*8 (a-h,o-z)
       parameter ( pi=3.1415927 )
       parameter ( rneg = 1.5d0*1.5d0 )
       integer jclus(Num_Part), iclus(Num_Part), trt(3,27)
       integer ipart(Num_Part), que(Num_Part), image(3,Num_Part)
       real*8 xox(Num_Part,3), xwt(3), xwti(3), xyz(Num_Part,3)
       integer iccnt(Num_Part), ipccnt(Num_Part)
       real*8 eclus(Num_Part),epclus(Num_Part)
       real*8 rgyro(Num_Part)
       integer vden(100)
       integer list(Num_Part)
       integer num_clus
       logical do_energy,do_radgyr,do_denpro
       pointer ( addr_head, ihead(1) )
       pointer ( addr_checklist, ichecklist(1) )
```

```
      pointer ( addr_dist, dist(1) )     ! temporary distances for denpro
      pointer ( addr_xyz, xyz )
      pointer ( addr_list, list )
      pointer ( addr_que, que )
      pointer ( addr_image, image )
      nmax = 0        ! the size of the largest cluster
      imax = 0        ! is the index of the largest cluster
      pnmax = 0       ! equals one if nmax is percolating
      num_clus=0      ! is the number of clusters
      icum = 0        ! tracks the number of particles put into clusters
      iperc = 0       ! equals one if the configuration percolates
      do i=1,Num_Part
         ipart(i)=0    ! is the particles in the clusters
         iclus(i)=0    ! is the size of the clusters
         jclus(i)=0    ! is the cluster distribution (size wise)
         image(1,i)=128 ! is 128 if haven't been checked, holds the
periodicity
         image(2,i)=0
         image(3,i)=0
      end do
      ncells = int ( trans/1.6d0 )
      ncells3 = ncells**3
      addr_head = malloc ( %val(ncells3 * 4) )
      addr_checklist = malloc ( %val(ncells3*27 * 4) )
      addr_xyz = malloc ( %val(Num_Part*3 *8) )
      addr_list = malloc ( %val(Num_Part * 4) )
      addr_que = malloc ( %val(Num_Part * 4) )
      addr_image = malloc ( %val(Num_Part*3 * 4) )
      call setcells(ncells,trt,ichecklist(1))
      c_length = ncells/trans
C     PUT THE PARTICLES IN THE COORESPONDING CELL
      do i=1,Num_Part
         list(i)=0
      end do
      do i=1,ncells3
         ihead(i)=0
      end do
      DO I=1,Num_Part
         ICELL = 1 + INT( ( XOX(I,1) ) * C_length )
     :             + INT( ( XOX(I,2) ) * C_length ) * NCELLS
     :             + INT( ( XOX(I,3) ) * C_length ) * NCELLS * NCELLS
         LIST(i) = ihead(icell)
         ihead(icell) = i
      end do
c      rneg = 1.5*1.5    ! cluster criterion
C..set up que with first particle
```

```
do ij=1,Num_Part
   if ( image(1,ij) == 128 ) then
      iper = 0          ! equals one if the cluster is percolating
      istart = ij             ! the first particle in the cluster
      in_que=1             ! the number of particles to be checked
      que(in_que)=istart        ! list of particles to be checked

      in_clus=1           ! the number of particles in the cluster
      icum = icum + 1

      image(1,istart)=0
      image(2,istart)=0
      image(3,istart)=0
      ipart(icum)=istart
      xyz(icum,1)=xox(istart,1)         ! the periodic coordinates
      xyz(icum,2)=xox(istart,2)
      xyz(icum,3)=xox(istart,3)

      do while ( in_que .ne. 0 )
         ip = que(in_que)
         in_que = in_que - 1
         icell = 1 + int( ( xox(ip,1) ) * C_length )
:              + int( ( xox(ip,2) ) * C_length ) * NCELLS
:              + int( ( xox(ip,3) ) * C_length ) * NCELLS * NCELLS
         xwt(1)=xox(ip,1)
         xwt(2)=xox(ip,2)
         xwt(3)=xox(ip,3)
         M = ihead(icell)
         DO while ( M .ne. 0 )    ! LOOP OVER # PARTICLES IN iCELL
            if ( m <> ip ) then
               RO2=(XWT(1)-Xox(m,1))**2+(XWT(2)-Xox(m,2))**2+
:                                       (XWT(3)-Xox(m,3))**2
               if ( ro2 <= rneg ) then
                  if ( image(1,m) == 128 ) then
                     in_clus=in_clus+1
                     icum=icum+1
                     ipart(icum)=m
                     image(1,m)=image(1,ip)
                     image(2,m)=image(2,ip)
                     image(3,m)=image(3,ip)
                     xyz(icum,1)=xox(m,1)+image(1,m)*trans
                     xyz(icum,2)=xox(m,2)+image(2,m)*trans
                     xyz(icum,3)=xox(m,3)+image(3,m)*trans
                     in_que=in_que+1
                     que(in_que)=m
                  else
                     if ( iper == 0 ) then
                        if ( image(1,m) <> image(1,ip) .or.
:                            image(2,m) <> image(2,ip) .or.
:                            image(3,m) <> image(3,ip) ) then
```

```
                        iper=1
                        iperc=1
                     end if
                  end if
               end if
            end if
         end if
         M = list(M)
      end do ! while

      call translate (NCELLS,icell,idimx,idimy,idimz)
      DO IDO = 2,27              ! loop over the other 26 cells
         idirx=idimx
         idiry=idimy
         idirz=idimz
         if ( trt(1,ido) <> idimx ) idirx=0
         if ( trt(2,ido) <> idimy ) idiry=0
         if ( trt(3,ido) <> idimz ) idirz=0
         kcell = ichecklist((ido-1)*ncells3+icell)
         M = ihead(kcell)
         do while ( M .ne. 0 )  ! loop over particles in cells
            xwti(1)=xox(m,1)+idirx*trans
            xwti(2)=xox(m,2)+idiry*trans
            xwti(3)=xox(m,3)+idirz*trans
            RO2=(XWT(1)-XWTI(1))**2+(XWT(2)-XWTI(2))**2+
     :           (XWT(3)-XWTI(3))**2
            if ( ro2 <= rneg ) then
               if ( image(1,m) == 128 ) then
                  in_clus=in_clus+1
                  icum=icum+1
                  ipart(icum)=m
                  image(1,m)=idirx + image(1,ip)
                  image(2,m)=idiry + image(2,ip)
                  image(3,m)=idirz + image(3,ip)
                  xyz(icum,1)=xox(m,1)+image(1,m)*trans
                  xyz(icum,2)=xox(m,2)+image(2,m)*trans
                  xyz(icum,3)=xox(m,3)+image(3,m)*trans
                  in_que=in_que+1
                  que(in_que)=m
               else
                  if ( iper == 0 ) then
                     if (image(1,m)<>idirx+image(1,ip) .or.
     :                    image(2,m)<>idiry+image(2,ip) .or.
     :                    image(3,m)<>idirz+image(3,ip) )then
                        iper=1
                        iperc=1
                     end if
                  end if
               end if
            end if
```

```
                M = list(M)
             end do
          end do
       end do
       num_clus = num_clus + 1
       iclus(num_clus) = in_clus

       if ( in_clus > nmax ) then
          nmax=in_clus
          imax=num_clus
          if ( iper == 1 ) then
             pnmax=1
          else
             pnmax=0
          end if
       end if
       if ( iper == 0 ) then
          iccnt(in_clus)=iccnt(in_clus)+1
          jclus(in_clus)=jclus(in_clus)+1
       else
          ipccnt(in_clus)=ipccnt(in_clus)+1
       end if
       if ( (do_energy .or. do_radgyr) .and. in_clus<>1 ) then
!            calculate cluster energy & radius of gyration
          ene = 0.0
          sum_r2 = 0.0
          do i=icum-in_clus+1,icum-1
             ip = ipart(i)
             xwt(1)=xyz(i,1)
             xwt(2)=xyz(i,2)
             xwt(3)=xyz(i,3)
             do j=i+1,icum
                jp = ipart(j)
                RO2=(xwt(1)-xyz(j,1))**2+(xwt(2)-xyz(j,2))**2+
:                    (xwt(3)-xyz(j,3))**2
                if ( do_radgyr ) sum_r2 = sum_r2 + RO2
                if ( do_energy ) then
                   RO2 = (1./RO2)**3
                   ene = ene + 4*(RO2*RO2-RO2)
                end if
             end do
          end do
          if ( iper == 0 ) then
             if ( do_energy ) eclus(in_clus)=eclus(in_clus)+ene
             if ( do_radgyr ) rgyro(in_clus)=rgyro(in_clus)+
:                  sqrt(sum_r2/in_clus/(in_clus-1))
          else
             if ( do_energy ) epclus(in_clus)=epclus(in_clus)+ene
          end if
       end if
```

```
         end if
      end do
      if ( do_denpro ) then
         if ( pnmax == 0 ) then
            istart = 0
            do i=1,imax-1
               istart = istart + iclus(i)
            end do
            sumx = 0.d0
            sumy = 0.d0
            sumz = 0.d0
            do i=istart+1,istart+nmax
               sumx = sumx + xyz(i,1)
               sumy = sumy + xyz(i,2)
               sumz = sumz + xyz(i,3)
            end do
            xwt(1) = sumx / nmax
            xwt(2) = sumy / nmax
            xwt(3) = sumz / nmax

            addr_dist = malloc ( %val(nmax*8) )

            j = 0
            do i = istart+1,istart+nmax
               j = j + 1
               roo=(xwt(1)-xyz(i,1))**2 + (xwt(2)-xyz(i,2))**2 +
     :              xwt(3)-xyz(i,3)**2
               dist(j) = sqrt(roo)
            end do

            tmax = 0.
            do i=1,nmax
               if ( dist(i) > tmax ) tmax=dist(i)
            end do

            do i = 1, nmax
               nco = int(dist(i)/1.3) + 1
               vden(nco) = vden(nco) + 1
            end do

            call free( %val(addr_dist) )

         end if

      end if
c      write(6,*) 'ipart'
c      write(6,*) 'number of clusters',num_clus
c      iend = 0
c      do j=1,num_clus
c        istart = iend + 1
```

```
c         iend = istart + iclus(j) - 1
c          write(6,'(i4,70(T10,15i4,/))') j,(ipart(i),i=istart,iend )
c       end do
c       write(6,*) 'eclus'
c       do i=1,Num_Part
c          if ( jclus(i) <> 0 ) then
c             eclus(i) = eclus(i)/jclus(i)/i
c          end if
c       end do
c       write(6,'(10f7.3)') (eclus(j),j=1,Num_Part)
        call free( %val(addr_head) )
        call free( %val(addr_checklist) )
        return
        end
c-----------------------------------------------------------------
        subroutine translate (M,nc,idimx,idimy,idimz)
c       set up the neighbors for the cells
c       M = # cells
c       NC is the cell number
        IDIMx=0
        IDIMy=0
        IDIMz=0
        IF  ( MOD(NC,M) == 0 ) THEN
           IDIMx=1
        END IF
        IF ( MOD(NC+M-1,M) == 0 ) THEN
           IDIMx=-1
        END IF
        do i=1,M
           IF ( NC >= i*M**2-M+1 .and. NC <= i*M**2 ) THEN
              IDIMy=1
           END IF
           if ( NC >= (i-1)*M**2+1 .and. NC <= M+(i-1)*M**2 ) then
              IDIMy=-1
           end if
        end do
        IF ( NC >= M**3-M**2+1 .and. NC <= M**3 ) then
           IDIMz=1
        end if
        IF ( NC <= M**2 ) THEN
           IDIMz=-1
        END IF

        return
        end
!-----------------------------------------------------------------
        subroutine setcells (m,trt,checklist)
        integer trt(3,27),checklist(m**3,27)
C        M=number of cells
        DO NC=1,M**3
```

```
          IDIMx=0
          IDIMy=0
          IDIMz=0
          IF  ( MOD(NC,M) == 0 ) THEN
             IDIMx=1
          END IF
          IF ( MOD(NC+M-1,M) == 0 ) THEN
             IDIMx=-1
          END IF
          do i=1,M
             IF ( NC >= i*M**2-M+1 .and. NC <= i*M**2 ) THEN
                IDIMy=1
             END IF
             if ( NC >= (i-1)*M**2+1 .and. NC <= M+(i-1)*M**2 ) then
                IDIMy=-1
             end if
          end do
          IF ( NC >= M**3-M**2+1 .and. NC <= M**3 ) then
             IDIMz=1
          end if
          IF ( NC <= M**2 ) THEN
             IDIMz=-1
          END IF
          DO I=1,27
             neighbor = NC + TRT(1,i) + TRT(2,i)*M + TRT(3,i)*M*M
             if ( trT(1,i) == IDIMx ) then
                neighbor = neighbor + (-1*IDIMx*M)
             end if
             if ( trT(2,i) == IDIMy ) then
                neighbor = neighbor + (-1*IDIMy*M**2)
             end if
             if ( trT(3,i) == IDIMz ) then
                neighbor = neighbor + (-1*IDIMz*M**3)
             end if
             checklist(nc,i) = neighbor
          END DO
       END DO
c      write(6,*) 'check list'
c      do i=1,M**3
c         write(6,'(27i4)') ( checklist(i,j),j=1,27 )
c      end do

       return
       end
c-----------------------------------------------------------------------
```

VII. BIBLIOGRAPHY

1. J.W. Gibbs, *The Scientific Papers of J. Willard Gibbs* (Dover, New York, 1961), Vol.1.

2. F.F. Abraham, *Homogeneous Nucleation Theory* (Academic Press, New York, 1974).

3. J. Lothe and G.M. Pound, *J. Chem. Phys.*, **36**, 2080 (1962).

4. G.S. Springer, Homogeneous Nucleation in *Advances in Heat Transfer* (Academic Press, New York, 1978).

5. M.P. Allen and D.J. Tildesley, *Computer Simulation of Liquids* (Oxford Science Publications, Oxford, 1987).

6. A.Z. Panagiotopoulos, N. Quirke, M. Stapleton, and D.J. Tildesley, *Molec. Phys.*, **63**, 527 (1988).

7. B. Smit, C.P. Williams, E.M. Hendriks, and S.W. De Leeuw, *Molec. Phys.*, **68**, 765 (1989).

8. P.W. Atkins, *Physical Chemsitry* (W.H. Freeman and Company, New York, 1986).

9. F.F. Abraham, *Phys. Rep.* **53**, 93 (1979).

10. R.S. Berry, S.A. Rice, and J. Ross, *Physical Chemistry* (John Wiley & Sons, New York, 1980).

11. J. Lothe and G.M. Pound, Statistical Mechanics of Nucleation. In *Nucleation* (A.C. Zettlemoyer, ed.), pp 109-149, Dekker, New York, 1969.

12. X.C. Zeng and D.W. Oxtoby, *J. Chem. Phys.*, **94**, 4472 (1991).

13. D. Stauffer, *Phys. Rep.*, **54**, 1 (1979).

14. K.R. Bauchspiess and D. Stauffer, *J. Aerosol Sci.*, **9**, 567 (1978).

15. M.E. Fisher, *Physics*, **3**, 255 (1967).

16. D.M. Heyes and J.R. Melrose, *Molec. Phys.*, **66**, 1057 (1989).

17. N.A. Seaton and E.D. Glandt, *J. Chem. Phys.*, **86**, 4668 (1987).

18. D. Stauffer, *Introduction to Percolation Theory* (Taylor and Francis, London, 1985).

19. V.P. Gregory and J.C. Schug, *Molec. Phys.*, **78**, 407 (1993).

20. N. Metropolis and S. Ulam, *J. Am. Stat. Ass.*, **44**, 335 (1949).

21. J.E. Lennard-Jones, *Proc. Roy. Soc.*, **A106**, 463 (1924).

22. W.H. Stockmayer, *J. Chem. Phys.*, **9**, 398 (1941).

23. J.O. Hirschfelder, C.F. Curtiss, and R.B. Bird, *Molecular Theory of Gases and Liquids* (John Wiley & Sons, Inc., 1954).

24. H. Reiss, J.L. Katz, and E.R. Cohen, *J. Chem. Phys.*, **48**, 5553 (1968).

25. D.J. McGinty, *J. Chem. Phys.*, **58**, 4733 (1973).

26. F.H. Stillinger, *J. Chem. Phys.*, **38**, 1486 (1963).

27. H. Reiss, A. Tabazadeh, and J. Talbot, *J. Chem. Phys.*, **92**, 1266 (1990).

28. B. Quentrec and C. Brot, *J. Comput. Phys.*, **13**, 430 (1975).

29. R.W. Hockney and J.W. Eastwood, *Computer Simulation using Particles* (McGraw-Hill, New York, 1981).

30. E.M. Sevick, P.A. Monson, and J.M. Ottino, *J. Chem. Phys.*, **88**, 1198 (1988).

31. J. Hoshen and R. Kopelman, *Phys. Rev. B.*, **14**, 3438 (1976).

32. W.W. Wood, *The Physics of Simple Liquids*, ed. H.N.V. Temperly, J.S. Rowlinson, and G.S. Rushbrooke (Amsterdam, North-Holland Pub. Co., 1968).

33. A.Z. Panagiotopoulos, *Molec. Phys.*, **61**, 813 (1987).

34. J.J. Nicolas, K.I. Gubbins, W.B. Streett, and D.J. Tildesley, 1979, *Molec. Phys.*, **37**, 1429.

35. J. Hansen and L. Verlet, 1969, *Phys. Rev.*, **184**, 151.

36. M.S.S. Challa and J.H. Hetherington, 1988, *Phys. Rev. A.*, **38**, 6324.

37. R. Evans, and M.M. Telo da Gama, 1979, *Molec. Phys.*, **38**, 687.

38. Y. Adachi, I. Fijihara, M. Takamiya, and K. Nakanishi, 1988, *Fluid Phase Equilibria*, **39**, 1.

39. J.A. Barker, P.J. Leonard, and A. Pompe, 1966, *J. Chem. Phys.*, **44**, 4206.

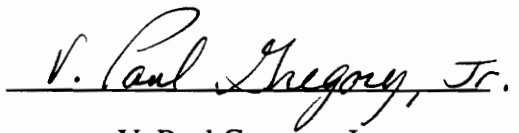
40. J.P. Hansen and I.R. McDonald, 1976, *Theory of Simple Liquids* (Academic Press).

41. Table given in ref. [23]. Prepared by J.S. Rowlinson from the calculations by R.B. Bird, PhD. Dissertation, University of Wisconsin (1950).

VITA

Victor Paul Gregory, Jr. was born May 19, 1966, the son of Victor Paul and Mary LeGrand Gregory. He graduated from South Stokes Senior High School in 1984. He received his BS in Chemistry from Wake Forest University in 1988. He received the undergraduate analytical research award while at WFU and also helped optimize a CI/SCF program package. He holds a masters degree in physical chemistry from Virginia Polytechnic Institute and State University.

He is interested in the usefulness of computers to simulate chemical systems. In particular, he is fascinated with the application of the Monte Carlo simulation technique to understand nucleation theory and predict states of matter when given their thermodynamic properties. He is also interested in being able to develop materials for specific purposes based on simulation techniques, and in the role of the interaction potential on the properties.



V. Paul Gregory, Jr.