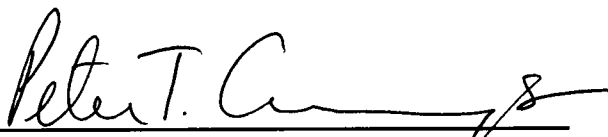
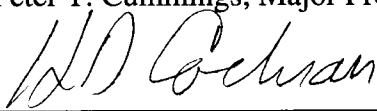


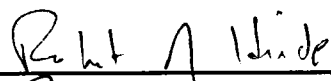
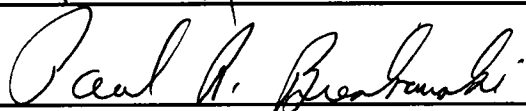
To the Graduate Council:

I am submitting herewith a thesis written by Harpreet S. Chopra entitled "Algorithms for the Gibbs Ensemble Monte Carlo Simulations". I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Chemical Engineering.


Peter T. Cummings, Major Professor


Hank D. Cochran, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:


Associate Vice Chancellor and
Dean of The Graduate School

**PARALLEL ALGORITHMS FOR THE
GIBBS ENSEMBLE MONTE CARLO
SIMULATIONS**

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Harpreet S. Chopra

December 1996

DEDICATION

This thesis is dedicated to my parents

Mr. Chanchal Singh Chopra

(ਚੰਚਲ ਸਿੰਘ ਚੌਪੜਾ)

and

Mrs. Harvinder Kaur Chopra

(ਹਰਵਿੰਦਰ ਕੌਰ ਚੌਪੜਾ)

who have been with me each step of the way.

Acknowledgments

I would like to thank Dr. Peter Cummings and Dr. Hank Cochran for their guidance and patience through out this work and during the writing of this thesis, Dr. Ariel Chialvo for his useful suggestions and discussions, the Laboratory directed research and development (LDRD) program of the Oak Ridge National Laboratory (ORNL) for the funding of this project and the Center for Computational Sciences at ORNL for the generous use of the Intel Paragon XPS/5.

Abstract

Conventional Gibbs ensemble Monte Carlo simulations have been carried out on serial machines with the samples of both the boxes being on the same processor. With the advent of distributed memory parallel supercomputers, it is necessary to develop new techniques to parallelize these simulations. In this work different procedures for the parallelization of the GEMC simulations have been tested in an attempt to find the most accurate and efficient method.

Since the GEMC simulations involve two simulation boxes which are almost independent of each other except for the time when the exchanges are taking place, we have experimented with techniques which involves placing the two boxes on different processors. Since the boxes are now on different nodes, they need to get paired before any transfers can be made. The different possibilities in which the nodes can get paired lead to a number of different procedures for carrying out the simulations.

It is seen that the parallel procedures are more efficient because they can sample more configurations in the same wall clock time, thus improving the statistical accuracy of the results. For the multi-processor codes, it is seen that the swapping of the simulation boxes after a certain number of cycles can lead to a faster equilibration of the processors. However swapping is effective only at lower temperatures when the convergence rates are slow.

Contents

CHAPTER	PAGE
1 Background.....	1
1.1 Molecular Simulation.....	1
1.2 Calculation of Phase Equilibrium Properties.....	6
1.3 Gibbs Ensemble Monte Carlo Method.....	8
1.3.1 Serial Algorithm.....	10
1.3.2 Computational Aspects.....	14
1.4 Potential Models and Energy Calculations.....	19
2 Parallel Gibbs Ensemble Monte Carlo Simulations.....	21
2.1 Synchronous Parallel Algorithm.....	21
2.2 Asynchronous Parallel Algorithm.....	23
2.3 Simulation Details of the Asynchronous Parallel Algorithm.....	26
2.3.1 Structure of the Code.....	26
2.3.2 Pairing of the Processors.....	30
2.3.3 Decoupling of the Processors.....	33
2.4 Random Selection Multi-Processor Algorithm.....	33
2.5 Perfectly Parallel Multi-Processor Algorithm.....	39
3 Calculation of the Isothermal Compressibility.....	43
4 Phase Equilibrium Calculations for the BFW Potential.....	48
5 Results and Discussion.....	49
5.1 Tests for the Accuracy of the Simulations.....	49
5.1.1 Results for the Serial Code.....	50
5.1.2 Results for the Asynchronous Two-Processor Code.....	52
5.1.3 Results for Multi-Processor Codes.....	54
5.2 Tests for the Efficiency of the Simulations.....	67
5.2.1 Comparison of the Serial and Asynchronous Parallel Procedures.....	68

CHAPTER	PAGE
5.2.2 Comparison of the Perfectly Parallel and Swapping Procedures.....	71
5.3 Results for Isothermal Compressibility Calculations.....	83
5.4 Results for the Phase Equilibrium Calculations of the BFW Potential.....	84
4 Conclusions and Recommendations.....	86
References.....	90
Appendices.....	94
Appendix A.....	95
Appendix B.....	97
Appendix C.....	101
Vita.....	103

List of Figures

1.1 The two boxes represent samples from within the bulk of the two phases.....	9
1.2 Monte Carlo moves in the Gibbs ensemble method.....	9
2.1 Flowchart for the 2-node synchronous parallel GEMC simulation.....	23
2.2 Decoupling of the boxes.....	24
2.3 Flowsheet of ' <i>Subtrans</i> ' subroutine.....	27
2.4 Flowsheet for ' <i>Subvolume</i> ' subroutine.....	29
2.5 Random selection multi-processor procedure.....	35
2.6 Swapping algorithm.....	42
5.1 Phase coexistence curve for the Lennard-Jones fluid (serial code) compared to values reported by Panagiotopoulos [11].....	50
5.2 Phase coexistence curve for the Lennard-Jones fluid (asynchronous code).....	52
5.3 Plot of average vapor energies, comparison of random-selection (r-s) and asynchronous codes for $T^*=0.9$	57
5.4 Plot of average vapor energies, comparison of random-selection and asynchronous codes for $T^*=1.0$	58
5.5 Plot of average liquid energies, comparison of random-selection and asynchronous codes for $T^*=0.9$	58
5.6 Plot of average liquid energies, comparison of random-selection and asynchronous codes for $T^*=1.0$	59
5.7 Plot of average vapor energies of all the 10 vapor processors for $T^*=0.9$	64
5.8 Plot of average liquid energies of all 10 liquid processors for $T^*=0.9$	64
5.9 Plots of block averages of vapor energies for the asynchronous and the serial code.....	69
5.10 Plot of block averages of liquid energies for the serial and asynchronous parallel procedures.....	70
5.11 Plots of the average energies (for all processors) for runs without swapping at $T^*=0.9$	72
5.12 Plots of the average energies (for all processors) for runs with swapping at 500 cycles for $T^*=0.9$	73
5.13 Plot of average energies for different swapping frequencies at $T^*=0.9$	75
5.14: Linear plots for different swapping frequencies at $T^*=0.9$	75

5.15 Linear plots for the average vapor energies for the case without swapping and for swapping at 500 cycles at $T^*=1.0$ (5 processors in vapor phase).....	76
5.16 Plot of the average vapor energies (over all vapor phases) at $T^*=0.9$	77
5.17 Plot of the difference from mean of the vapor averages at $T^*=0.9$	77
5.18 Plot of the average liquid energies (over all vapor phases) at $T^*=0.9$	78
5.19 Plot of the difference from mean of average liquid energies at $T^*=0.9$	78
5.20 Plot of the average vapor energies and difference from the mean values at $T^*=0.8$	79
5.21 Plot of the average liquid energies and difference from the mean values at $T^*=0.8$	80
5.22 Plot of the average vapor energies and difference from the mean values at $T^*=1.0$	80
5.23 Plot of the average liquid energies and difference from the mean values at $T^*=1.0$	81
5.24 Plot of the average vapor energies and difference from the mean values at $T^*=1.15$	81
5.25 Plot of the average liquid energies and difference from the mean values at $T^*=1.15$	82
B.1 Plot of $1/\kappa_T$ for simulation results and for the Lennard-Jones equation of state.....	97
B.2 Plot of $1/\kappa_T$ for simulation results and for the Lennard-Jones equation of state.....	99
B.3 Plot of $1/\kappa_T$ for simulation results and for the Lennard-Jones equation of state, showing only the vapor phases.....	100
C.1 Phase coexistence curve for Kr. Comparison of the simulation results with reported values.....	101

List of Tables

5.1 Coexistence properties for Lennard-Jones fluid (serial code).....	51
5.2 Coexistence properties for Lennard-Jones fluid (asynchronous code).....	53
5.3 Coexistence properties for the Lennard Jones fluid (random selection multi processor code) for $T^*=1.0, 0.9$	55
5.4 Results for perfectly parallel multi-processor code for $T^*= 0.9$ showing all 20 processors.....	61
5.5 Coexistence properties for Lennard-Jones fluid (perfectly parallel code).....	62
5.6 Coexistence properties for Lennard-Jones fluid (parallel multi-processor code with swapping).....	65
B.1 Results for isothermal compressibility calculations.....	98
C.1 Simulation results for the BFW potential.....	102

List of symbols

Latin Alphabets

E	energy
ΔE	change in energy
J_i	value of property J in the i^{th} state
k_b	Boltzmann constant
L	box length
LRC	long range correction
N	number of molecules
N_{trans}	total number of translational moves
N_{vol}	total number of volume exchanges
N_{mol}	total number of molecule exchanges
P	pressure
P_{mol}	ratio of probability of the new to the old state (for a molecule exchange)
P_{trans}	ratio of probability of the new to the old state (for a translational move)
P_{vol}	ratio of probability of the new to the old state (for a volume move)
$\mathcal{P}$	instantaneous pressure function
Q	partition function
Q_{Gibbs}	partition function for the Gibbs ensemble
r_c	cutoff distance for the potential models
r	particle position
r_{ij}	position of particle i relative to j
T	temperature
Δt_i	time interval

U	intermolecular potential
V	volume
δV_{max}	maximum volume exchange

Greek alphabet

β	inverse temperature ($1/k_bT$)
ϵ	energy parameter in pair potentials
κ	isothermal compressibility
ϕ	pair potential
Λ	thermal de Broglie wavelength
ρ	number density
σ	length parameter in pair potentials

Special conventions

$\langle \dots \rangle_{ens}$	general ensemble average
*	denotes reduced variables

Chapter 1: Background.

1.1 Molecular Simulation

Molecular simulation is an application of statistical mechanics which uses a relatively small number of particles to represent a macroscopic system. Simulations are usually carried out in a cubic simulation box, which is considered a sample from the bulk of the fluid. Periodic boundary conditions are applied to replicate the box infinitely in space, thus making the representation more realistic by minimizing the effect of interfaces. Statistical methods are then employed to obtain macroscopic properties from the microscopic properties calculated during the simulations.

The individual configurations of the particles in the simulation box represent unique points in configuration space, and the simulation can explore the configuration space by generating numerous configurations of the same system. New configurations of the system can be generated either randomly by using random numbers or by solving the classical Newton's equations of motion [1]. The random numbers used are generally pseudo-random numbers and are generated on the interval (0,1) by using computer random number generators. Depending on the procedure used for the generation of new configurations, the principal methods for conducting molecular simulations can be broadly divided into the Monte Carlo simulations (so named because of the use of random numbers in this procedure) and the molecular dynamics simulations.

Monte Carlo (MC) methods randomly generate new configurations of the system, using a Markov chain of events. A Markov chain is a sequence of trials which satisfies the following two conditions.

- a) The outcome of each trial belongs to a finite set of outcomes, called the state space.
- b) The outcome of each trial depends only on the outcome of the trial that immediately precedes it.

The Markov chain is used to generate configurations according to a probability distribution function such as the Boltzmann probability function. If the Boltzmann distribution is used, the ensemble average of the property J is calculated as

$$\langle J \rangle = \frac{1}{N} \sum_{i=1}^N J_i \quad (1.1)$$

where J_i is the property J in the i^{th} configuration and N is the total number of configurations generated. MC simulations statistically sample the configuration space by generating configurations distributed according to a prescribed distribution, but cannot follow the trajectory of a particle with time. Thus they cannot be used directly to calculate dynamic properties.

Molecular dynamics (MD) simulations are an alternative technique which track the trajectory of the system through phase space by solving the equations of motion, generating samples of the same system at different time intervals. Time averages are then used to obtain the macroscopic properties.

$$\langle J \rangle = \lim_{t \rightarrow \infty} \frac{1}{t} \sum_i^t J_i \Delta t_i \quad (1.2)$$

where J_i is the value of the property J in the i^{th} time step (Δt_i)

Since MD simulations follow the trajectory of particles through phase space with

time, they can be used to calculate dynamic properties. According to the ergodic hypothesis, time averages are equivalent to ensemble averages in the thermodynamic limit. This is reflected in the similarity between the macroscopic averages obtained from two techniques.

Compared to MC simulations, MD simulations have proven relatively easier to implement in parallel environments. Parallelization can be carried out either by domain decomposition [2], which involves the geometrical breakdown of the simulation box among different processors, or the replicated data approach, in which the same system is present on all the processors, the force calculations and the integration of the equations of motion for the molecules being distributed among the processors.

MC simulations on the other hand have been more difficult to parallelize because they are intrinsically sequential in nature and in the standard implementations, only one molecule moves at each step. Thus, the communication time relative to that of computation is intrinsically higher in MC molecular simulations than for MD simulations. A number of different approaches to the parallelization of the Monte Carlo simulations have been attempted. As for the MD simulations, these approaches can be broadly classified as domain decomposition and replicated data.

Domain Decomposition Approaches to Monte Carlo Simulations.

Two main domain decomposition strategies can be identified.

a) Making simultaneous moves in independent regions of the simulation [3]: The simulation can be divided into independent regions if the potential is short ranged, such

that a change in location of a particle in one region does not have an effect on the energy of any other region. The independent regions can then be simulated on different processors. In the event of two adjacent processors moving atoms within range of each other, the move initiated with the earlier clock time is assigned priority and completed first. The method becomes inefficient at higher densities due to this contention between the processors. The restriction on this method is that the moves and interactions cannot be long ranged, thus very long chain molecules and systems with long range forces cannot be simulated by this method.

b) Time warp technique [4]: This is an optimistic protocol for event driven simulation. The simulation is carried out optimistically assuming that all the events are occurring in the correct order until an error is detected. A rollback and recovery mechanism is then used to correct the errors. This is a domain decomposition algorithm and involves the physical breakdown of the simulation box among a number of processors.

Replicated Data Approaches

a) Perfectly Parallel MC: Another form of parallelism can be the perfectly parallel approach, in which the parallelism is introduced at the level of the complete simulation rather than to speed up one single run. Simultaneous simulations can be run on different processors using different seeds for the random number generators. This procedure could be considered a decomposition over MC time since exploring phase space in an MC simulation is analogous to stepping through time in an MD simulation. This has thus been termed 'the time decomposition method' [3].

This procedure may not be very beneficial if the equilibration time of a simulation

is not small compared to the production time. Also, this method may not be used for the simulation of very large systems, since the size of the system is restricted by the memory on an individual processor. Since this approach does not aim to improve the speed of one simulation, but involves simultaneous simulations running on individual processors in order to improve the statistical accuracy of the results, it scales perfectly with the number of processors in that the total number of molecules simulated (added up over all the processors) divided by the number of processors is a constant.

b) Distribution of energy calculations: This method involves distributing the computational load of a Monte Carlo simulation (energy calculations) over a number of processors, where each processor is responsible for calculating the contribution to the energy by a subset of the system's particles. This approach is likely to be inefficient because only a small subset of the system particles interact with the moved particle at a given time.

c) Esselink Parallel approach: As proposed by Esselink et. al. [5], this is a replicated data method and has been applied successfully for the MC simulations of long chain molecules [5,30]. It involves placing replicas of the system on all the processors. For choosing the new position of a chain, a number of trial conformations of the chain are generated in parallel. Out of these the most favorable conformation has the highest probability of being selected. If the move is accepted, the conformation is communicated to all the processors and another trial move is attempted. Since the most favorable move is attempted, the acceptance criterion is modified to remove the bias introduced by this procedure.

1.2 Calculation of Phase Equilibrium Properties.

Phase equilibrium properties play an important role in the chemical industry where they are used in the design of separation processes based on difference in the composition of two phases at equilibrium. The conventional engineering approach for the calculation of the phase equilibrium properties is to start from a macroscopic model without taking into consideration the microscopic nature of matter.

For many separation processes (such as recovery of biochemical products), few macroscopic models of the systems may be available. Also in the early stages of development, experimental data for a process under consideration may be unavailable. It might thus be difficult to predict the phase equilibrium properties based on standard macroscopic thermodynamic approaches. This is one of the reasons why molecular-based approaches that explicitly take into consideration the intermolecular interactions are gaining in popularity for the prediction of phase equilibrium properties.

Simulation techniques (MC and MD) can easily calculate mechanical properties such as energy, pressure and pair correlation functions but not non-mechanical properties such as entropy or free energy which are the properties required for the phase equilibrium calculations. Special techniques such as the grand canonical Monte Carlo simulation (GCMC) [1, 6] and the Widom test particle method [6, 7] need to be employed for the calculation of the free energy and chemical potentials. Some of the earlier simulation techniques used for phase equilibrium calculations are described below.

a) Indirect techniques.

Using indirect methods [8, 9], it is possible in principle to obtain the phase diagrams by performing a series of simulations at different densities (and compositions for mixtures) and numerically matching the values of the pressures and chemical potential in the coexisting phases. However, the number of simulations required increases with the number of components in a mixture and with the density. Also the GCMC simulations and the test particle insertion methods for the calculation of the chemical potential become very inefficient for polyatomic molecules (larger molecules) or for denser systems because it becomes difficult to sample adequately the configurations that significantly contribute to the free energy in a simulation of reasonable length. These procedures are more expensive close to the triple point since the number of attempted insertions required for an accurate estimate of chemical potential increases sharply with density.

b) Direct techniques.

These involve setting up a 2-phase system in a box with an interface [10]. The presence of an interface means that the simulations are slow to equilibrate. The simulation box should be large enough so that the bulk properties can be calculated beyond the influence of the interface. The long range corrections are difficult to calculate in these inhomogeneous systems and the densities close to the interfaces are susceptible to the surface effects. In many cases the system may not separate into distinct phases because of limited system size and the use of periodic boundary conditions [11]. However, if the interfacial region itself is the subject of interest, this technique is appropriate and widely used.

1.3 Gibbs Ensemble Monte Carlo Method.

First proposed in 1987 by Panagiotopoulos [12], the Gibbs Ensemble Monte Carlo (GEMC) technique is a method for the direct calculation of the phase coexistence properties of pure components and mixtures from a single simulation without the need for specifying the chemical potential of the system. The method attempts to simulate phase coexistence properties by following the evolution in phase space of a system composed of two distinct regions, each representing a sample from within the bulk of one phase (Figure 1.1). The simulation boxes representing each of the phases do not have any physical contact, and thus there are no interfaces. Thermodynamic equilibrium between these two boxes is brought about by carrying out a number of different Monte Carlo moves (Figure 1.2). The moves involve the translational displacement of the particles, volume exchanges between the two boxes, and particle exchanges between the two boxes. The translational moves are essential for the internal equilibration of the systems after the destabilization brought about by the particle or volume exchanges. The volume exchanges ensure the equality of the pressure in the two boxes, and the particle exchanges ensure the equality of the chemical potential. This method thus involves elements of the NVT (constant number of particles, volume and temperature) ensemble, the NPT (constant number of particles, pressure and temperature) ensemble and the μ VT (constant chemical potential, volume and temperature) ensemble. The combined effect of the three moves is to force the two systems to attain equality of chemical potential and pressure. These simulations are performed at the same temperature for both the systems, so that temperature equality is imposed. These conditions are the necessary and sufficient conditions for phase equilibrium as originally proposed by W. J. Gibbs [13] and thus the name 'Gibbs Ensemble' was given to this method.

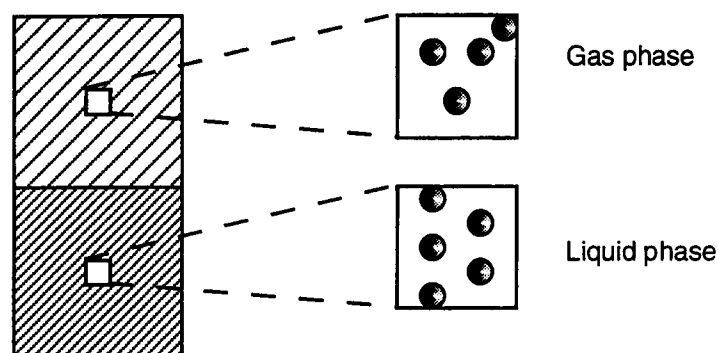


Figure 1.1 The two boxes represent samples from within the bulk of the two phases.

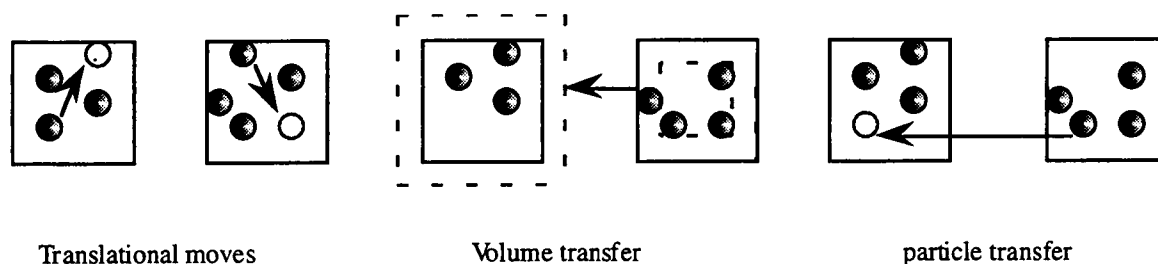


Figure 1.2 Monte Carlo moves in the Gibbs Ensemble method.

Conventional GEMC simulations are carried out on serial machines and have both the boxes conceptually and physically on the same processor. With the advent of distributed memory parallel supercomputers, it has become possible to implement variations of conventional algorithms in the parallel environments. This work investigates different approaches to the parallelization of the GEMC simulation and attempts to identify an efficient parallel version of the algorithm. Section 1.3.1 and 1.3.2 contain the simulation and computational details of the conventional serial GEMC simulations. The subsequent chapters discuss the various approaches for the parallelization of the technique.

1.3.1 Serial Algorithm

As noted above, the conventional simulations have been performed in a serial fashion where both the boxes conceptually and physically exist on the same processor. The total system is considered to be under NVT conditions since the total number of particles and the total volume of the system remain constant throughout the simulation. The simulation is carried out in two boxes where each box represents a sample from the bulk of the fluid. Each box is replicated in all three directions by applying periodic boundary conditions. The simulation is conducted in a manner which avoids the presence of interfaces between the two boxes, thus eliminating surface effects. Thus GEMC can be used with a relatively small number of molecules.

The two boxes attain equilibrium by performing different Monte Carlo moves. The three kind of moves performed are described in detail as follows.

a) Translational moves.

A particle is chosen randomly in either of the boxes and is moved to a random point within a cube of side $2d_{rmax}$, centered on the particle's original position (d_{rmax} is the maximum displacement). The move is accepted with a probability of $\min(1, P_{trans})$, where P_{trans} is the ratio of probabilities of the new and the old state and is given as

$$P_{trans} = \exp(-\beta\Delta E) \quad (1.3)$$

where ΔE is the change in energy for the trial move, $\beta = 1/k_b T$ where k_b is the

Boltzmann constant and T is the absolute temperature. If the move is rejected the old state is recounted in the Markov chain of states. The translational moves are required for the internal equilibration of the systems after the volume and the molecule exchanges. The particle to be moved can be chosen from one box while the other box remains idle, or particles may be chosen in both the boxes during a translational move and the move may be accepted or rejected by considering the change in energies in both the boxes [12].

The ΔE in this case is calculated as

$$\Delta E = \Delta E_1 + \Delta E_2 \quad (1.4)$$

where the subscripts 1 and 2 indicate the change in energy in boxes 1 and 2 respectively. The acceptance criterion remains the same as Equation (1.3).

b) Volume Transfers.

The second type of Monte Carlo move is the volume exchange between the two phases. In the constant volume GEMC simulations, the volumes of the two boxes are simultaneously changed so as to keep the total volume of the system constant. The perturbation in volume in one box is matched with an equal and opposite perturbation in the other box. The new configurations of the particles are obtained by scaling the position of the particles to the new volume, a procedure similar to that used in NPT ensemble simulations. If ΔE_1 and ΔE_2 are the changes in energy in boxes 1 and 2 respectively, then the ratio of the probabilities of the new and old states, P_{vol} , is given as

$$P_{vol} = -\beta \left[\Delta E_1 + \Delta E_2 - N_1 kT \ln \left(\frac{V_1 + \Delta V}{V_1} \right) - N_2 kT \ln \left(\frac{V_2 - \Delta V}{V_2} \right) \right] \quad (1.5)$$

for a transfer of volume from box 2 to box 1. The combined move is accepted with a probability $\min(1, P_{vol})$. The volume exchange move coupled with the translational moves helps the two boxes attain equality of pressure and also become internally equilibrated. The pressure in the two boxes is not specified externally (as in the NPT simulations), but the two boxes adjust themselves to arrive at the pressure value at that state point.

In the case of mixtures, it is possible to perform the Gibbs calculations where the total system is under NPT (constant number of particles, pressure and temperature). In the constant-pressure GEMC, as opposed to the constant-volume GEMC, the volume moves are uncoupled and are made independently in the two phases. Thus, the total volume of the system is not constant. In these kinds of simulations, the ratio of the probabilities of the new and old states, P_{vol} , is given as

$$P_{vol} = \exp \left(-\beta \left[\Delta E_1 + \Delta E_2 - N_1 kT \ln \left(\frac{V_1 + \Delta V_1}{V_1} \right) - N_2 kT \ln \left(\frac{V_2 + \Delta V_2}{V_2} \right) + P(\Delta V_1 + \Delta V_2) \right] \right) \quad (1.6)$$

For the case of mixtures the constant pressure Gibbs method is beneficial over the constant pressure Gibbs method [11].

c) Molecule Transfers.

The third kind of Monte Carlo move involves the transfer of a particle from one box to another. A particle is chosen at random and is made to disappear from the donor box. A particle is inserted into the acceptor box randomly at the same time and the move is

accepted with a probability of $\min(1, P_{mol})$ where the ratio of probabilities of the new and the old state, P_{mol} , is calculated as

$$P_{mol} = \exp\left(-\beta\left[\Delta E_1 + \Delta E_2 + kT \ln \frac{V_2(N_1 + 1)}{V_1 N_2}\right]\right) \quad (1.7)$$

for the transfer of a molecule from phase 2 to phase 1. The quantities ΔE_1 and ΔE_2 are the change in energies in the two phases, and N_1 and N_2 are the original number of particles.

To satisfy microscopic reversibility [1], creations and destructions in the two regions must be attempted with equal probability. This is achieved by choosing the boxes for insertions randomly with an equal probability. Incorporating this move into the simulation ensures the equality of the chemical potential of the components in the two phases without specifying a value for it. Thus, it is not necessary to calculate the chemical potential for the two boxes though, it could be easily calculated during the particle insertions in the molecule exchanges.

The overall procedure thus involves moves which sample the NVT, the NPT and the μ VT ensembles and result in two regions which are internally equilibrated and have equality of chemical potential and pressures.

The starting densities affect the results of the simulation to a great extent. If the initial densities are chosen in the unstable region, then the system should rearrange itself into two stable phases. If the simulation is started from a state which lies in the metastable region, the quantity $(\partial P / \partial V)$ will be negative. Such a state can be long-lived

and a phase separation may not be observed during a simulation of reasonable length, unless random fluctuations bring one of the regions past the stability limit [12]. For simple fluids such as the Lennard-Jones fluid, the phase diagram is well known and thus the starting densities for the liquid and vapor phases can be chosen close to the equilibrium densities to save on computation time. This will also allow sufficient number of particles to be present in each box at equilibrium (if the simulation is started with the same number of particles in each box). However, if the phase diagram is not known in advance, then short exploratory runs can be made to get approximate estimates of the final phase densities and to check if phase separation is taking place.

1.3.2 Computational Aspects.

The conventional GEMC is carried out using two simulation boxes representing samples from the two phases at equilibrium, typically the liquid and vapor phases. However, the GEMC simulations can also be used to simulate other fluid phase equilibria (such as liquid-liquid equilibria). Periodic boundary conditions are applied to each box separately so that a good representation of the macroscopic properties can be obtained. The simulation boxes should contain enough molecules so that the simulation results exhibit little dependence on the system size.

The simulation is performed in cycles where each cycle consists of N_{trans} attempted translational moves, N_{vol} attempted volume exchanges and N_{mol} attempted molecule exchanges. In most of the earlier work on GEMC [11,12], the moves have been grouped together in a cycle in a sequential manner, with the translational moves being performed first in the cycle (the NVT ensemble), followed by the volume exchanges (the NPT ensemble) and finally the molecule exchanges (the μ VT ensemble) carried out at the

end. One of the disadvantages of this procedure is that, since the molecule and the volume exchanges serve to destabilize the system, successive exchange moves will be performed on an unstable system and this might lead to erroneous results. A bias in results has been observed in the liquid pressures in [11] as pointed out by Smit in [21] due to the large number of molecule exchanges. The translational (displacement) moves are performed to equilibrate the system internally, and it is better to perform the exchanges interspersed randomly between the displacements. The nature of the move to be performed within a cycle is thus chosen randomly.

The details of the implementation of the simulation are as follows:

a) Initial Starting Conditions

The initial density and the number of particles should be such that after equilibration there are a large enough number of particles in each phase. This is important to ensure appropriate sampling of the equilibrium properties of both phases. Another alternative would be to start from densities close to the equilibrium densities at that state point. For the Lennard-Jones fluid the phase diagram is reasonably well known and the initial densities can be chosen so as to save computation cycles and also to ensure that there is always a sufficient number of molecules in the two phases. Otherwise, if the phase diagram is not known, the initial densities and number of particles must be determined by trial and error.

The initial configuration in each box can be a face-centered cubic (FCC) lattice (cold start). In this case, the initial number of molecules in each of the boxes will be a number from the series of $4n^3$, where $n=1,2,3,\dots$. If starting from an FCC lattice, the

properties should be accumulated from the point where the lattice structure has melted away. The length of the equilibration run (melting of the lattice structure) depends on the initial density. For larger densities, a larger number of configurations may be required to melt the lattice structure. Another alternative is to start the simulation from a saved configuration (hot start) from a previous run.

b) Long range corrections.

The calculation of the potential energy of a configuration is the most expensive part of the MC code computationally. In principle the interactions between a molecule and all its images in the neighboring boxes should also be considered while calculating the energy of that particle. But this would lead to an infinite number of calculations. For relatively short-ranged potentials such as the Lennard-Jones potential, it is possible to restrict the summation of energies by making an approximation. The Lennard-Jones potential is given by

$$\phi(r_{ij}) = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^6 \right] \quad r_{ij} \leq r_c \quad (1.8)$$

$$= 0 \quad r_{ij} > r_c$$

where ϵ is the well depth of the LJ potential, σ is the separation of particles at zero interaction energy and r_c is the cutoff distance.

A spherical cutoff is applied so that only the interactions between a particle and those lying within the cutoff are considered while calculating the energies. This means setting the potential U to zero for $r \geq r_c$ where r_c is the cutoff distance. In the simulation of the Lennard-Jones fluid, the cutoff is normally taken as 2.5σ where σ is the collision

distance between the particles. The approximation introduced by using a cutoff is corrected for by considering long-range corrections. Contributions to the energy and pressure for $r \geq r_c$ are estimated by assuming the pair correlation function $g(r) \approx 1$ in this region. For the Lennard-Jones potential the long range corrections for the energy and pressure are

$$E_{LRC}^* = (8/9)\pi N \rho^* (r_c^*)^{-9} - (8/3)\pi N \rho^* (r_c^*)^{-3}$$

$$P_{LRC}^* = (32/9)\pi N (\rho^*)^2 (r_c^*)^{-9} - (16/3)\pi N (\rho^*)^2 (r_c^*)^{-3}$$

where the * represents reduced units. The reduced variables are defined in terms of σ as the fundamental unit of length and ε as the fundamental unit of energy. The corrections for an NVT ensemble can be applied at the end of the run since the number of particles and the density remain the same. However for the NPT and the μ VT ensemble, the corrections have to be applied to the instantaneous energies and the pressures when these are calculated since the long range corrections also change during the course of these moves.

b) Translational move.

The number of translational moves N_{trans} should be such that on average, each particle will be moved once during a cycle. Thus

$$N_{trans} \approx N_1 + N_2$$

where N_1 and N_2 are the numbers of particles in the two boxes. The particle to be moved is chosen randomly from a randomly picked box. Thus in the serial procedure, a

translational move usually involves the movement of a particle in only one box. The new position of the particle is generated by adding randomly generated displacements to the three coordinates.

$$\begin{aligned} \text{PXNEW} &= \text{PXOLD} + (2.0 * \text{RAN}(\text{DSEED}) - 1.0) * \text{DRMAX} \\ \text{PYNEW} &= \text{PYOLD} + (2.0 * \text{RAN}(\text{DSEED}) - 1.0) * \text{DRMAX} \\ \text{PZNEW} &= \text{PZOLD} + (2.0 * \text{RAN}(\text{DSEED}) - 1.0) * \text{DRMAX} \end{aligned}$$

$\text{RAN}(\text{DSEED})$ is a library function for generating a random number on (0,1), DSEED being the seed value. DRMAX is the maximum displacement. The maximum displacement is adjusted so as to keep the acceptance ratio of the displacement moves at around 50%. This can be easily done by decreasing DRMAX if the acceptance ratio is below 50% so as to increase the probability of accepting moves and vice versa.

c) Volume Transfers.

The volume transfers involve the transfer of a volume from one box to another. The volume to be exchanged ΔV is chosen randomly from the interval 0 to δV_{max} where δV_{max} is the maximum volume exchange. The value of δV_{max} is also adjusted so as to keep the ratio of accepted moves as 50 %. The long-range corrections change after every volume exchange because of the difference in the densities of the two states and thus these need to be taken into consideration while calculating the acceptance criteria. For systems such as the Lennard-Jones fluid, the scaling properties of the potential can be used to calculate the change in the energy.

c) Molecule Transfers.

The particle transfers involve the transfer of a particle from one box to another. The donor and acceptor boxes are decided randomly. A particle is chosen from the donor box

and is made to disappear. It is inserted into the acceptor box at a random location. Since the calculated energy of the inserted particle corresponds to the test particle insertion, the chemical potential can be calculated at no additional cost to the simulation.

The number of particle exchanges attempted depends on the state of the system. Close to the critical point, the density of the liquid phase is smaller compared to that at the lower temperatures. Since the probability of accepting a particle transfer near criticality is higher, the number of moves attempted should be less than at lower temperatures. A possible check as to whether the number of attempted particle transfers is sufficient is to check the equality of chemical potential in the two phases. Thus the number of moves should be large enough to ensure equality of chemical potential but small enough to give correct results.

1.4 Potential Models and Energy Calculations.

In this study we used the Lennard-Jones (LJ) potential models for calculating the intermolecular energies of the systems. The LJ potential is given by

$$\phi(r_{ij}) = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^6 \right] \quad r_{ij} \leq r_c$$

$$= 0 \quad r_{ij} > r_c$$
(1.8)

where ϵ is the well depth of the LJ potential, σ is the separation of particles at zero interaction energy and r_c is the cutoff distance.

For the particle displacements, the change in energy of the particle moved has to be calculated. These energy calculations involves only a single loop in which the

distances between the particle under consideration and its neighbors within a distance of r_c are calculated. Thus there are at the most $2(N-1)$ calculations.

The volume exchanges require the recalculation of all $\frac{1}{2}N(N-1)$ interactions.

For the LJ potential the change in energy with volume can be calculated by using the scaling property of the potential. The energy of the box before the volume exchange, V_{old} , is given as

$$\begin{aligned} V_{old} &= 4\epsilon \sum_i \sum_{j>i} \left(\frac{\sigma}{r_{ij}} \right)^{12} - 4\epsilon \sum_i \sum_{j>i} \left(\frac{\sigma}{r_{ij}} \right)^6 \\ &= V_{old}^{(12)} + V_{old}^{(6)} \end{aligned} \quad (1.9)$$

The potential can be separated into its twelfth-power and the sixth-power components. The energy of the new state V_{new} can be calculated as

$$V_{new} = V_{old}^{(12)} \left(\frac{Boxlen_{old}}{Boxlen_{new}} \right)^{12} + V_{old}^{(6)} \left(\frac{Boxlen_{old}}{Boxlen_{new}} \right)^6 \quad (1.10)$$

and the change in energy as

$$\Delta V = V_{old}^{(12)} \left[\left(\frac{Boxlen_{old}}{Boxlen_{new}} \right)^{12} - 1 \right] + V_{old}^{(6)} \left[\left(\frac{Boxlen_{old}}{Boxlen_{new}} \right)^6 - 1 \right] \quad (1.11)$$

Chapter 2: Parallel Gibbs Ensemble Monte Carlo Simulations.

A number of different approaches to the parallelization of the GEMC have been investigated. Since the two boxes behave independently during the translational moves and have to be coupled only during the exchange moves, one way to parallelize the procedure would be to decouple the boxes and physically place them on different nodes. Thus, for the simple parallelization of the basic GEMC code, two processors would be required, each having one of the boxes. The statistical mechanical details of the simulation such as the acceptance criteria, the ratio of accepted moves, long range corrections remain the same in all the different techniques investigated. The only difference between the procedures described in the next few paragraphs is the extent to which the boxes are independent of each other and the way they come together to conduct an exchange.

2.1 Synchronous Parallel Algorithm

The synchronous parallel algorithm involves the simple parallelization of the serial GEMC code without any changes to the simulation procedure. The two boxes are physically on different nodes but are not independent and perform the MC moves synchronously. All the steps are performed in the same way and sequence as the serial code. The communication between the two boxes is carried out using a message passing interface. The Paragon system has a number of message passing libraries including LNX, NX, PVM, P4 and *tcgmsg*. This work is based upon the NX message passing library which offers the best performance.

The simulation is performed in cycles with each cycle attempting N_{trans} translational moves, N_{vol} volume exchanges and N_{mol} particle transfers. The nature of the Monte Carlo moves is decided randomly by using a pseudo random number generated by a random number generator. Both the boxes use the same seed value for the random number generators to generate the same sequences of random numbers. This is important for making synchronous moves in both the boxes. The code required for the synchronous simulation is very similar to the serial code and the subroutines for conducting the translational moves, volume and particle exchanges in the serial code can be used without any major modifications.

For a translational move in the serial code, the box in which the move is made is decided randomly with an equal probability. The other box remains idle during the move. However in this parallel procedure, we need not allow the other box to remain idle. In the serial code there was no incentive for performing the moves simultaneously in both the boxes, but in the parallel version, it would be a waste of resources not to do so. Thus a modification to the serial procedure would be to perform displacements in both the boxes simultaneously during the scheduled translational move. If N_{trans} is kept the same as the old value, this would mean conducting almost double the number of displacements as compared to the serial procedure. Thus N_{trans} on each node can be halved or be adjusted dynamically so that all the particles have an equal chance of being displaced at least once during the cycle.

The transfer moves are decided concurrently on both the processors. For a volume exchange, the volume to be transferred can be calculated in both the boxes simultaneously using the same chain of seed values for the random number generators. The changes in the energy are calculated separately on both the processors and are

communicated by messages. The acceptance criteria can then be calculated on both the nodes and the move can be accepted or rejected concurrently. A similar procedure is followed for the particle exchanges. A schematic flow diagram of the algorithm is presented in Figure 2.1.

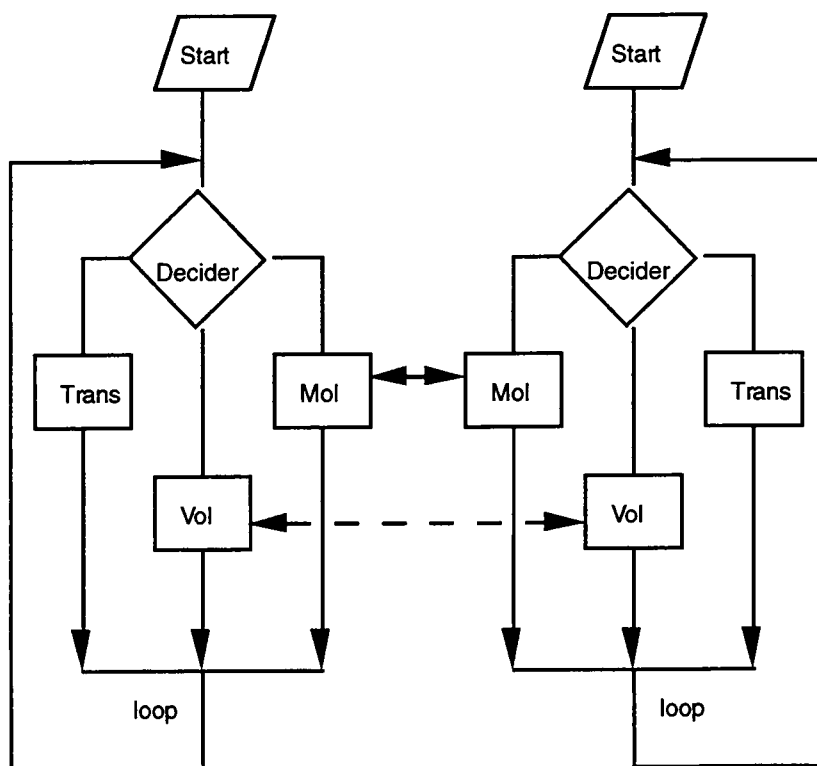


Figure 2.1: Flowchart for the 2-node synchronous parallel GEMC Simulation.

2.2 Asynchronous Parallel Algorithm

In the synchronous procedure, the nodes are not independent of each other as they perform all the moves concurrently. As the system approaches equilibrium, the number of particles differs in the two boxes. Usually the denser phases has a greater number of particles than the lower density phase, and thus the time required for a translational move is larger. The time for performing the energy calculations in a translational move (which

involves the calculation of energy of the particle moved, due to interactions with its neighbors) scales with the number of particles since the energy calculation in the translational moves has only one loop and involves the calculation of energy of only the displaced particle with the other particles. Thus the vapor phase, which completes a translational move in less time than the liquid phase, has to wait until the other phase completes its move. This is not very efficient. Also the synchronous procedure simulates only two boxes and cannot be extended to a larger number of processors. The primary goal of the development of the asynchronous procedure is to decouple the boxes totally and make them completely independent of each other (Figure 2.2). The boxes are now conducting translational moves asynchronously. For two nodes the transfer moves are conducted between the two existing processors. For the multi-node code, the partner for conducting a transfer can be chosen in a number of ways, leading to the different multi-processor algorithms.

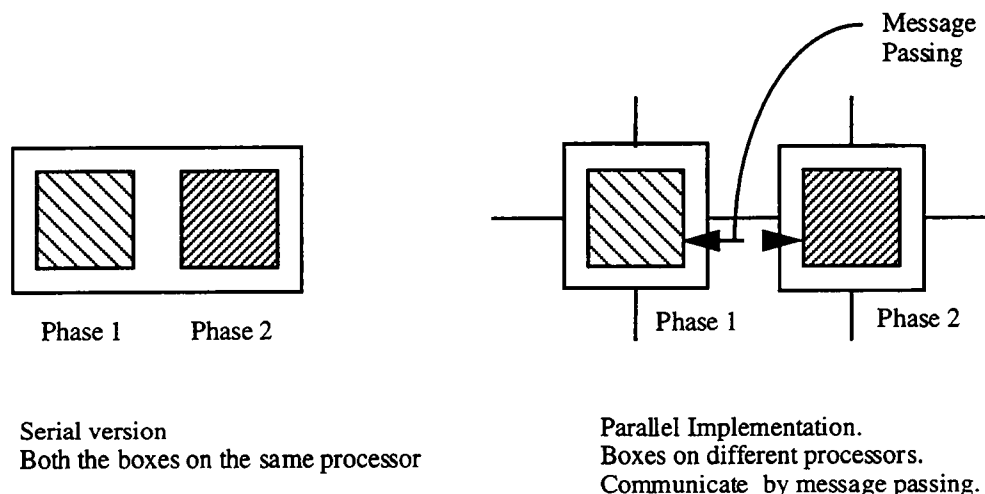


Figure 2.2 Decoupling of the boxes.

The boxes continue to make Monte Carlo moves in cycles, each cycle consisting of the same number of translational, volume exchange and molecule exchange moves as in the synchronous procedure. The nature of the moves made is decided randomly on

each processor. Each processor performs translational moves independently until it comes to a transfer (particle or volume) move, when it send a message to the other processor. This processor which initiates the move is termed the calling processor. The other (called) processor may be conducting translational moves itself and responds to the message after completing the translational move. Thus in this algorithm the processors have to continuously check for any possible messages which might be coming in. Once the 'called' processor has responded, the transfer move is conducted simultaneously on both the processors. The change in the energy is calculated separately on each processor and is communicated to one of the processors. The acceptance criterion is now calculated on only one processor and the decision (acceptance or rejection of the move) is communicated to the other. This is necessary because it is not possible to generate the same random number on both the processors, the nature of the moves being totally random.

In case both processors decide to initiate transfer moves at the same time (a very rare occurrence), the move is aborted, but not counted in the Markov chain. The asynchronous procedure can be easily extended to a larger number of nodes. In this case now the caller will decide upon a node either randomly or in an ordered manner, leading to the different multi-node procedures.

Since the processors keep on switching between the roles of the caller and the responder, the subroutines for the transfer moves are divided into the 'caller' part and the 'responder' part and are accessed accordingly. Further details about the working of the procedure are presented in the next Section (2.3).

The number of transfers per cycle (N_{vol}, N_{mol}) in the asynchronous mode needs

to be approximately half that of the synchronous procedure because each processor initiates N_{vol} volume exchanges, and the total number of volume exchanges between the two boxes (in case of two nodes) is $2N_{vol}$.

2.3 Simulation Details of the Asynchronous Parallel Algorithm

One of the major problems the asynchronous algorithm faces is to keep the processors aware of the current status of the others, and to make the processors respond to every call while performing translational moves themselves. This would have been easy if it were possible to change the value of a variable while a processor was busy with an exchange, which a calling node could read without sending a message and getting a reply for it. This could be possible in some processor chips in which the calling node could access a specific memory location and read off the current status, but the i860 XP processor chips on the Paragons do not support that function. The only way the calling node can know of the current status of the called node is to send a message and receiving a reply for it.

2.3.1 Structure of the Code.

The nature of the move to be undertaken is decided randomly on each processor and the appropriate subroutine is called. The subroutines for the asynchronous procedures need to be modified to handle both the caller and the responder modes of operations and are divided into the 'caller' and the 'responder' parts.

The translational moves are carried out by calling the subroutine '*Subtrans*' (Figure 2.3). This subroutine involves choosing a particle at random and displacing it to a random point within a cube of length equal to the maximum displacement. The change in

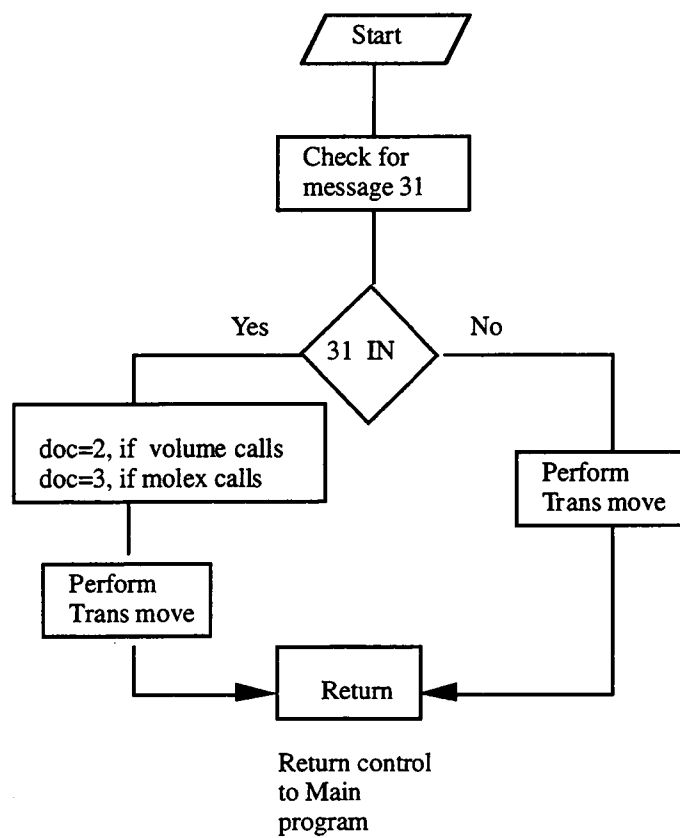


Figure 2.3 Flow sheet of 'Subtrans' subroutine.

energy is calculated and the move is accepted or rejected with the Boltzmann probability (Equation 1.3). The energy calculations are carried out in a separate subroutine ‘*energy*’, which involves calculation of the change in energy of the chosen particle with the other interacting particles using the Lennard-Jones potential. The subroutines used in the translational moves are almost the same as in the serial and the synchronous codes since it is only the transfer moves which are affected due to the asynchronous nature of the procedures. While the translational move is being conducted, the processor checks for any messages from the calling node using the `iprobe(message number)` command. If a message comes in, the translational move is completed and the next move is the response to the message. The next move can be a volume or particle transfer depending on the nature of the message. If no message comes in during the translational move, then the processor is free to choose the next move. Thus the next move can be either a translational move or the initiation of a volume transfer or a molecule transfer.

The processors can initiate a volume transfer move by calling the subroutine ‘*Subvolume*’ (Figure 2.4). This subroutine is divided into two sections, the responder section and the caller section. Depending on the value of a variable `nway` passed to the subroutine, either the caller part or the responder part of the subroutine is accessed by the processor. Once the two processors get paired together, the volume to be exchanged `delvol` (ΔV) is calculated on the caller and is communicated to the responder. The volume to be exchanged is not calculated on both the processors as done previously, because it is randomly decided and it is no longer possible to generate the same random number on both the processors at the same time. ΔV is added to the volume in the caller part and is subtracted from the volume in the responder part. Thus, depending on the sign of the `delvol` (it can be positive or negative), the donor and acceptors are decided. The change in energy is calculated on both the nodes and is sent to the caller by the responder,

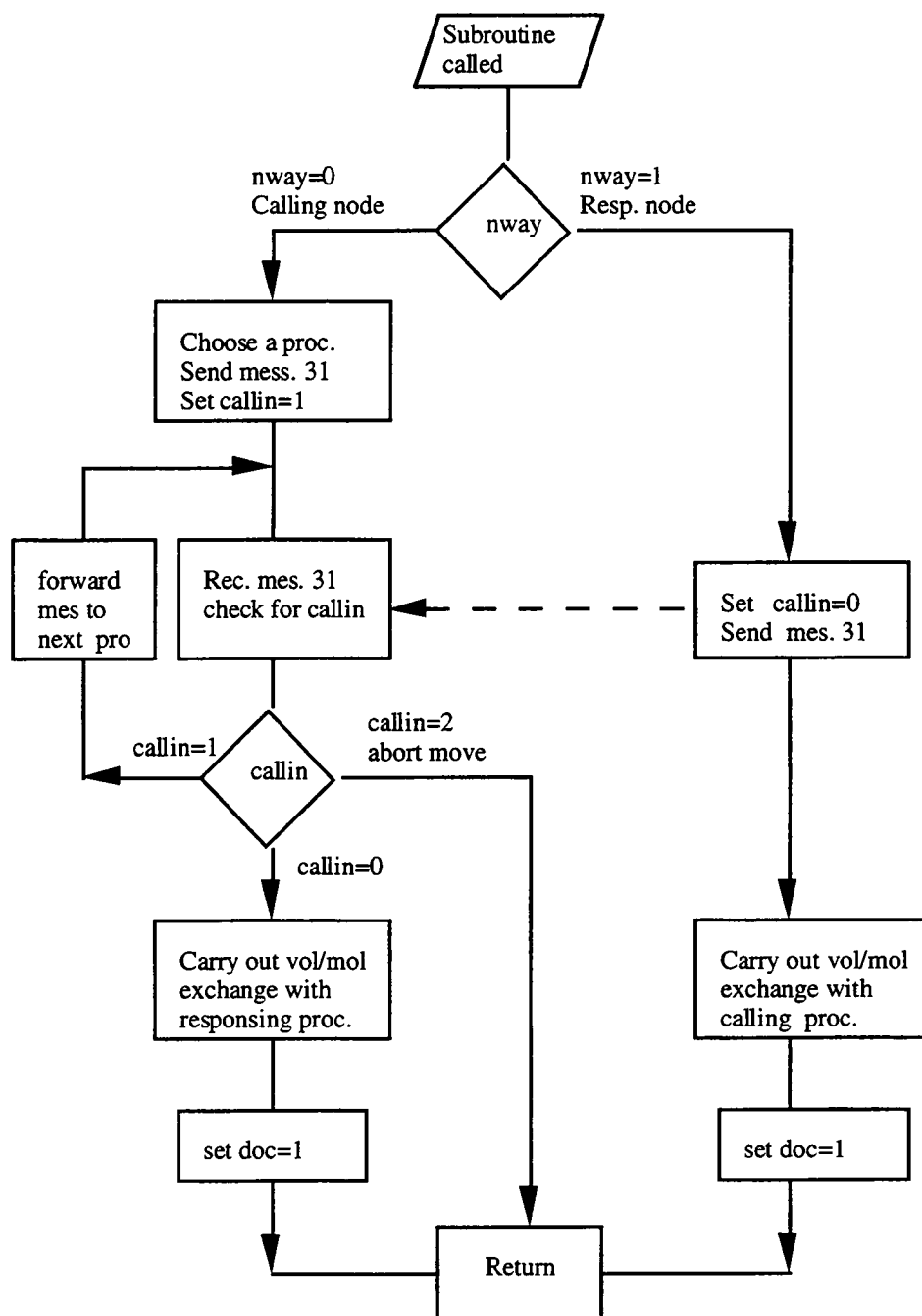


Figure 2.4 Flow sheet for 'Subvolume' subroutine

which then calculates the acceptance criterion, accepts or rejects the move and communicates the decision back to the responder node. It is not possible to calculate the acceptance criterion on both the nodes (as done in the synchronous code) because of the inability to generate the same random number on both the nodes, the moves and the identity of the acceptor-donor and the caller-responder being totally random.

The particle transfer subroutine '*Submolec*' is divided into four sections. The processors access either the 'caller' part ($nway=0$) or the 'responder' part ($nway=1$) depending on the value of $nway$ which is communicated to the subroutine. The donor and the acceptors in this case are decided randomly. Thus the caller can either be the donor or the acceptor. The donor (which can be the caller or the responder) now accesses the 'donor' part of the subroutine. In the donor part, one particle is chosen at random and is destroyed, while in the acceptor part of the subroutine a particle is inserted at a random location. The changes in the energy are calculated on both the nodes. The acceptor then communicates these to the donor which calculates the acceptance criterion, accepts or rejects the move and communicates the decision back to the acceptor node.

Once the two processors become paired together, the procedure is similar to that followed in the serial and the synchronous parallel procedures. The major challenge faced is to bring the processors together.

2.3.2 Pairing of Processors.

One of the approaches to bring the processors together for a transfer is to keep the other processor informed of the current status. This is done by sending a message after each

MC move. Before deciding on a move, a processor would check the message from the other processor. If the message is sent from the calling part of the transfer subroutines, then the responding processor has to access the responding part of the transfer subroutines. However, if the message was sent from the translational moves subroutine, then the processor is free to decide on one of the three moves. If it decided on a translational move it would then send a message to the other node indicating that it was free, but if it decided to do an exchange it would send out a message from the caller part of the transfer subroutines and would wait until the other processor responds.

This procedure involves message passing after every configuration and thus may require excessive communication time as compared to alternative procedures. For two nodes it would still be manageable, though much of the benefit of independent translational moves would be lost. Extending this kind of procedure to a larger number of processors would be chaotic and would likely be very inefficient. The alternative approach involves letting the nodes run free and dealing with a transfer when one arises.

The nature of the move is decided with the variable `doc`. The processors start off with the value `doc=1`. This signifies that the processor is free to choose one of the three MC moves. If it chooses a translational move it sets `mstep=1` and calls the translational move subroutine. In case it chooses to initiate the volume transfer it sets `mstep=2`, `nway=0` and calls the volume transfer subroutine. For initiating the particle transfer `mstep=3` and `nway=0`. A responding processor identifies the nature of the move called for by reading the variable `doc` from the message received. If `doc=2`, then it sets `mstep=2`, `nway=1` and calls the volume transfer subroutine accessing the responder part. If it reads `doc=3`, it sets `mstep=3`, `nway=1` and calls the particle transfer subroutine

accessing the responder part. For the initiating processor, `nway=0` and for the responding processor, `nway=1`.

On coming to a transfer move, the initiating processor sets the variable `nway=0`, and accesses the caller part of the transfer subroutine. It then sends a message (31) to the other node with the variable `callin=1` (which means that it is the calling node), informing it of the nature of move initiated and waits for a response. The other processor is constantly checking for the message (31) by using the `iprobe(31)` statement in the translational moves subroutine. On receiving the message (if `iprobe(31)` is equal to 1), the responding node completes the translational move, reads `doc=2` or `3` (depending on kind of transfer) and calls the subroutine desired (either volume transfer or particle transfer) with the variable `nway=1`. Thus, it accesses the responding part of the subroutine. Once inside the subroutine, it sends a response message (31) (with `callin=0`) to the caller which is waiting for it. Once the response message comes in, both the processors proceed normally with the transfer. It is important that the same message number used for both the calling and the response message. The value of variable `callin` distinguishes between the calling and the response messages.

In case both the nodes decide to initiate the move at the same time each of the nodes sends a message to the other with `callin=1` and waits for the reply. Since they are anticipating message 31, the calling messages are received as if they were the responses. But since the variable `callin` has a value of 1 instead of 0 (which is the value when it is a response message), the nodes realize that the other node is also initiating a move. The move is then aborted and both go back to performing translational moves (until the time when a node comes upon a transfer move again).

2.3.3 Decoupling of the Processors

The nodes conduct the MC moves in an asynchronous manner. Because of the disparity in the number of molecules and the speed at which the translational moves are conducted, it is likely that one of the processors will finish the allotted number of cycles earlier than the other. When this happens the node which finishes sends a global message. Both the processors then increment a variable `node_fin` which indicates the number of nodes that have finished their moves. However, the faster processor continues to do MC moves until the slower processor also finishes the allotted cycles. Once this is done, the finishing node (the slower one this time) sends the message, incrementing the variable `node_fin` on both the processors. When `node_fin` becomes equal to the number of processors (2 in this case) in the simulation, the processors stop conducting MC moves and the simulation ends.

2.4 Random Selection Multi-Processor Algorithm.

The motivation behind the development of the asynchronous algorithm was to make the procedure extendible to a larger number of processors. The basic algorithm developed for two nodes can be extended to multiple processors without any major changes.

For a larger number of processors also, each of the processors continues to conduct translational moves until it either initiates a transfer or receives a message calling for a transfer. When a processor initiates a transfer move, it accesses the calling part of the transfer subroutine. It chooses the other processor randomly from the pool of processors, sends the message (31) to the other processor with the variable `callin=1` (which signifies that the message is a calling message and not a response). For only two

nodes the other processor had to be free because it could conduct transfers with the calling processor only. But for multiple processors, the other processor might be busy conducting transfers with some other processor.

If the other processor is free (i.e. it is conducting a translational move), it completes the translational move and responds to the message in the same way as described earlier for the two-processor procedure. In case the other node is busy with a transfer move itself, it is running the volume transfer or the particle transfer subroutines and will pick up the message using the `iprobe(31)` statements in these subroutines. One of the ways such a situation (the called processor being busy) can be handled is that the calling processor gets back a negative reply asking it to try again. It could then wait until the called processor is free. This would lead to the calling processor waiting for the other processor. In case some other processor wants to conduct a transfer with the waiting processor, it will have to wait until the waiting processor completes the move. This can easily lead to a deadlocked situation with every processor waiting for another. Another alternative is that the busy node changes the target processor in the message and forwards the message to the next processor. In case the next processor is busy too, it will receive the message in one of the transfer routines, will change the target processor again and will forward the message further. In this way the message is forwarded from one processor to another until it finds a free one. The probability of a message transfer will increase with the proportion of transfer (volume or particle) moves to translational moves. This could happen at lower temperatures, when the number of attempted particle transfers has to be higher to account for the lower probability of successful transfers.

Since the calling processor chooses the other processor randomly, this procedure is termed as the 'random selection multi-processor procedure' (Figure 2.5).

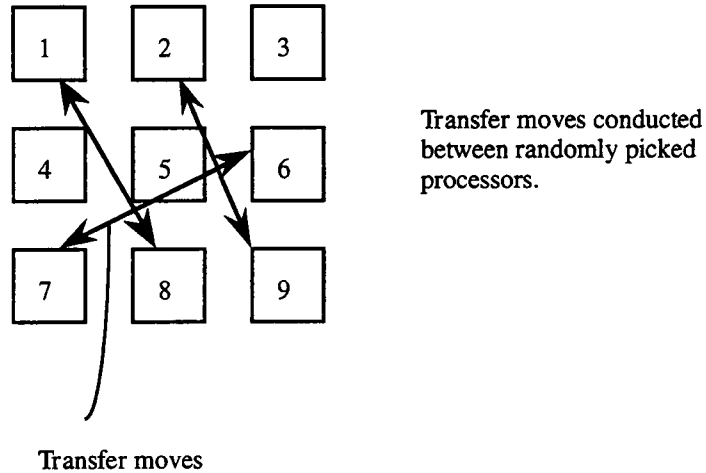


Figure 2.5 Random selection multi-processor procedure

As in the two-node procedure also, the processors may show disparity in the speed of the translational moves due to the difference in the number of particles present on each processor. To bring the simulation to a halt, the same procedure as the two-node code is used. When any processor finishes the allotted number of cycles, it sends out a global message and the variable `node_fin` is incremented on all the processors. The finishing node however continues to conduct moves in a normal fashion. In this way `node_fin` is incremented every time a node finishes the allotted moves. The processors raise a flag when `node_fin` becomes equal to the number of nodes, i.e. all the nodes finish the moves. The flag signals the termination of all transfer move and the winding down of the simulation. However the flag may not be raised on some slower processor in time and it may attempt to conduct a transfer move with some processor which has already terminated transfer moves. In this case the called processor changes the value of the variable `callin` from `callin=0` to `callin=2` and send the message back to the calling processor. The calling processor which is expecting a reply back (message 31) gets back

its own message with `callin=2`, indicating that the other processor has shut down operations and raises the flag itself and terminates its transfer moves.

In the random selection procedure, the number of MC move cycles performed is different for all the boxes. This may be due to the difference in the number of particles. This leads to a difference in the number of transfer moves initiated by the boxes and the number of translational moves conducted in each box. Also the number of transfer moves responded to is different as this processors are chosen randomly. Thus at the end of the simulation, the simulation boxes have conducted different number of translational and transfer moves. The difference in the number of translational moves may not affect the results, as these only lead to the internal equilibration of the boxes. However, the difference in the transfer moves may bring about a difference in the equilibration rates of the different phases. This should not affect the results as the increase in the number of transfer moves is accompanied by a corresponding increase in the translational moves which equilibrate the system.

Intuitively, it can be reasoned that the liquid phase boxes may conduct less number of transfer moves and may show a slower approach to equilibrium. However it seen that the liquid phases show a faster equilibration rate.

Since the selection of the processor called for a transfer move is made randomly, this procedure allows for transfer of volume and particles between processors belonging to the same phase. For optimal efficiency, we should have an equal distribution of the processors in the two phases. Thus if the transfers are made randomly, about 50 percent of the transfers are made between processors belonging to the same phases and are ineffective at bringing about phase equilibrium (though they contribute to the internal

equilibration of the phase). These transfers may not lead to any biasing of the results as they might only contribute to the internal equilibration of the whole phase. Because of the ineffective transfer moves, the equilibration observed may be slower than what is observed in the asynchronous two-node procedure.

The random selection procedure also involves the transfer of messages from a busy processor to a free one. For generating a random chain of configurations, the transfer should be performed with the processor chosen initially. This would mean that the calling processor should wait till the called processor becomes free. But since this would easily lead to a deadlock, the message is transferred. This potentially makes the selection of the processor non-random, biasing the selection procedure. Since this procedure seeks out the processor which is free (or doing translational moves), there is a greater probability of choosing that node, which spends a larger fraction of its time conducting translational moves. Since the particles in the liquid phases have more neighboring particles within the cutoff distance, they are likely to spend a larger fraction of their time conducting translational moves. In this case the liquid phases may have a greater probability of being chosen, thus introducing a bias. This bias may be prominent if both the liquid and the vapor phases attempt the same number of translational moves. However, the number of cycles conducted by a processor is not fixed and the vapor phases conduct more cycles in the course of the simulation. Thus, the vapor phases may spend an equal fraction of time doing translational moves. The fraction of time spend by the different nodes conducting the translational moves may be the main factor in the biasing of the procedure. This is however undetermined as it depends on the number of particles on a processor, and the speed of the processor at that time. The bias introduced can be removed by either precluding the transfer of messages, making the selection of the boxes totally random, or by making the selection of the same phase impossible.

The message transfer can be minimized by increasing the proportion of translational moves to the transfer moves. This will not affect the simulation as long as the number of transfer moves is sufficient for phase separation at the conditions of the simulation. The excess translational moves will serve to equilibrate the systems internally. Also the probability of finding a free node increases because of the decrease in the frequency of the transfer moves. It is also possible to allow transfers between unlike phases only. The message can be transferred until it finds a free node of the other phase. However, this procedure may be very expensive to implement as the ratio of the message passing to computation time will be very large in this case.

For a conventional GEMC simulation, the number of particles in the final states can be determined from the initial densities by a simple mass balance. The initial densities should be chosen in such a way that the final gas phases have enough particles to get good statistical accuracy. A small number of particles in the final states will make the results dependent on the size of the final system and should be avoided if possible. However it may not be possible to start from such an initial density which will give a large number of particles in the gas phase as this density may lie in the metastable region, and phase separation may not be seen at all. For two boxes belonging to different phases, the number of particles at equilibrium and the particle transfers are governed by mass balance considerations. For the random-selection algorithm, the transfer of particles occurs between similar phases also. These transfers are totally random and are not controlled by mass balance considerations. It is thus very difficult to predict the final distribution of particles and volume in any particular box. The random transfers may result in a deficiency of particles in some of the gas phases and the results of these boxes

will depend highly on the system size. Some of these phases may even become empty, leading to a gross load imbalance. The algorithm may thus prove to be unstable.

A possible way to remedy this problem is to allow transfers between boxes belonging to different phases only. As seen earlier, this can be achieved by transferring the message until it reaches a free processor of the other phase. Thus the message will have to travel from processor to processor until it finds the suitable one.

2.5 Perfectly Parallel Multi-Processor Algorithm.

A possible bias might be introduced in the random selection procedure due to the transfer of messages from the busy to the free processor. It is unclear how this biasing affects the results of the simulation and how it might be removed.

A different approach which precludes the need to transfer messages has also been investigated. In the random selection procedure, a processor initiating a transfer has to search for a suitable processor before they can get coupled. If the partner processor is predecided, then the initiating processor will not have to search for a suitable partner. This has two advantages. The first being that the processors do not have to send messages seeking a suitable processor and do not have to probe for the messages that might be coming in. Also, if two processors are coupled together, the transfers will be more ordered and will occur between unlike phases only, thus making the algorithm more efficient as compared to the random selection procedure.

Coupling of the two processors together for the entire length of the simulation is similar to breaking down the basic simulation unit to two processors. Thus the two-

processor codes can be run on multiple pairs of processor. This approach is called the 'perfectly parallel procedure' since it involves a number of simulations running simultaneously without interacting with each other. The perfectly parallel algorithms aim at generating better statistical averages and are analogous to speeding up one particular simulation (which also aims to sample more configuration space in the same wall clock time).

The only modification made to the two-processor asynchronous code to make it run in a perfectly parallel way is the introduction of a subroutine '*distr*', which is responsible for pairing the processors on the basis of the density. The boxes are divided into the liquid or vapor phases by comparing the densities with a threshold value which is set at the start of the simulation. Half of the processor start with densities above the threshold density and represent the liquid phase, whereas the other half start with a value below the threshold density and constitute the vapor phase. Thus the simulation is started with an equal distribution of the processors in the two phases. The subroutine '*distr*' then pairs up two processors belonging to different phases.

The initial densities are usually chosen to be in the unstable region ($\delta P / \delta V > 0$), such that the system rearranges itself to result into two stable phases. It is observed that some of the sub-simulations (2-node unit) equilibrate more slowly than the others. This could be explained by giving the following arguments.

It is possible that during the course of the simulation, some of the sub-systems may get caught in the metastable region and have a slower equilibration rate than the other sub-systems. It is also possible that since the phase separation is the result of random fluctuations of the density, the equilibration rate might be dependent on the

random fluctuations. Also, the system sizes (especially in the vapor boxes) are quite small, which suggests a system size dependence on the density fluctuations. Thus the difference in the equilibration rates might be an artifact of the system size.

The difference in the equilibration rates might be caused by any of these reasons or might be a combination of both these effects. It can be tested if the slower equilibration is due to the metastable region by perturbing the sub-systems in some way, such that they come out of the metastable region. A possible perturbation which could be introduced is the swapping of the processors after a certain number of cycles such that a liquid and a vapor phase are always brought together after the swaps.

If an improvement in the equilibration rates is indeed observed, it can be concluded that the sub-systems get caught in the metastable region. However, if there is no significant increase in the equilibration rates, the difference observed may only be a random occurrence.

The procedure which involves the swapping of boxes after a certain number of cycles is termed as the multi-processor algorithm with swapping (Figure 2.6).

The simulation is now divided into mini-simulations during which a pair of processors is conducting transfers between themselves. At the end of one loop (mini-simulation), the subroutine '*distr*' is called. The processors are again rearranged, forming pairs of unlike phases (Figure 2.6). The swapping of the processors introduces an additional degree of freedom and helps to randomize the procedure further. It may also help in faster equilibration by providing the necessary perturbation to bring the sub-

systems out of metastable regions. The frequency of the swaps can be changed by changing the length of the mini-simulation.

The swapping procedure works fine for lower temperatures when there are no frequent changes in the phase identity of the boxes. At higher temperatures the changes of phase identity may occur frequently and cause an unequal distribution of processors in the two phases. The subroutine '*distr*' requires that there be an equal number of processors above and below the threshold value. This procedure may thus fail at higher temperatures if the swaps are attempted during the identity change of some processors. In this case the number of particles may be used as a criteria for pairing of the processors instead of the densities.

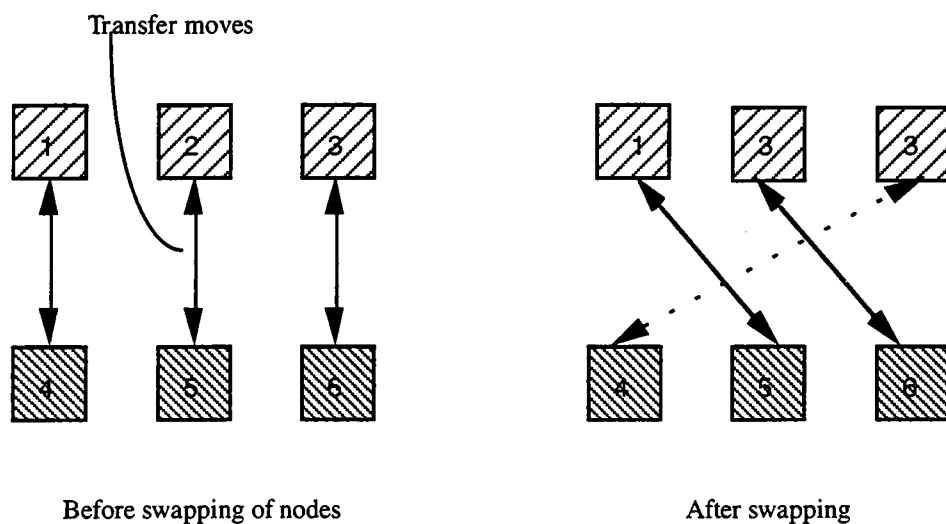


Figure 2.6 Swapping algorithm.

Chapter 3: Calculation of the Isothermal Compressibility

The isothermal compressibility is a quantity defined by the expression

$$\kappa_T = -\frac{1}{V} \left(\frac{\partial V}{\partial P} \right)_T \quad (3.1)$$

where V , P and T are the volume, pressure and the temperature of the system. The value of κ_T diverges with approach to criticality since the critical point is the point of horizontal inflection for P as a function of V . Thus the critical value of any property can be ascertained by plotting the value of the property against the inverse of the isothermal compressibility at different temperatures, approaching the critical point. The value at which the inverse of the isothermal compressibility becomes zero can give a reasonable estimate of the critical value of the property.

The motivation behind the calculation of the isothermal compressibility is to study the fluctuations in a distributed simulation (so called, because it is not restricted to two boxes). For a conventional GEMC simulation conducted in two boxes, the maximum number of particles in a box can become equal to the total number of particles in the system. This happens when one of the simulation box becomes empty. However for the distributed simulations (random selection multi-node procedure and the swapping procedures studied in this work), the maximum number of particles is no longer limited by the total number of particles in the two boxes. Since each box now has access to transfers from all the other boxes, all the particles can theoretically accumulate in one box. Such an occurrence may not be very probable. However, the random transfers between the boxes increases the probability of having a larger number of particles in some processors and a smaller number in others. Thus the extremities are potentially

greater than what might be expected from a normal simulation for the same starting conditions. This may lead to increased fluctuations in number of particles and other properties in these procedures.

For the random-selection procedure, the selection of the processors (for transfer moves) may be biased. Also this procedure involves excessive message passing before the coupling of the processors can take place and is thus not very efficient. An alternative simulation method which has nearly the same features as that of the random selection procedure is the multi-processor procedure with swapping. This method involves the swapping of the nodes in a random manner after a certain number of cycles have been conducted in a perfectly parallel manner. The processors have access to the particles in the other processors due to the random swapping and the fluctuations observed may be larger than those seen in the perfectly parallel procedures. The properties such as the heat capacity and the isothermal compressibility that are calculated from fluctuations may benefit from the swapping procedures. An attempt has been made to calculate the isothermal compressibility for the different procedures (random and non-random) and study the effect of the randomness on the fluctuations.

The expression for the isothermal compressibility in the Gibbs ensemble has not been reported in the literature. An attempt has been made to derive the equation starting from the partition function for the Gibbs ensemble given as [21]:

$$Q_G = \sum_{N_1=0}^N \int dV_1 V_1^{N_1} V_2^{N_2} Q_1 Q_2 \quad (3.2)$$

where V_1 and V_2 denote the volumes of box 1 and box 2, N_1 denotes the number of particles in box 1 and Q_1 and Q_2 denote the canonical partition functions of the two sub-systems given as Equation 3.3.

$$Q = \frac{1}{\Lambda^{3N} N!} \int_V d\mathbf{r}^N \exp[-\beta U(N)] \quad (3.3)$$

where $\mathbf{r}_1^N = \mathbf{r}_1, \dots, \mathbf{r}_N$ are the position of the particles in the sub-system, Λ is the deBroglie wavelength, N is the number of particles in the box, $\beta = 1/k_b T$, k_b is the Boltzmann constant, T is the temperature and U is the intermolecular potential.

Since for the Gibbs ensemble, the whole system (both boxes) is under NVT conditions (constant number of particles, volume and temperature) it can be shown that, for any property f (for the whole system)[Appendix A]

$$\frac{\partial \langle f \rangle}{\partial V} = \beta \left\{ \left\langle f \left(\frac{\mathcal{P}_1 + \mathcal{P}_2}{2} \right) \right\rangle - \langle f \rangle \left\langle \frac{\mathcal{P}_1 + \mathcal{P}_2}{2} \right\rangle \right\} + \left\langle \frac{\partial f}{\partial V} \right\rangle \quad (3.4)$$

where $\langle \dots \rangle$ denotes the average value, $\mathcal{P}_1$ and $\mathcal{P}_2$ are the instantaneous pressures functions in the two sub-systems and V is the volume. The instantaneous pressure function is defined as

$$\mathcal{P} = \rho k_b T + \mathcal{W}/V \quad (3.5)$$

where $\mathcal{W}$ is the internal virial [1]. The values of the pressures in the sub-system are obtained when the quantity $\mathcal{P}_1$ and $\mathcal{P}_2$ are averaged. We can define an instantaneous pressure function for the whole system ($\mathcal{P}$) and it is assumed that $\mathcal{P}_1$ and $\mathcal{P}_2$ contribute equally to $\mathcal{P}$. Thus

$$\mathcal{P} = (\mathcal{P}_1 + \mathcal{P}_2)/2$$

This accounts for the factor 1/2 in Equation (3.4). The quantity $\mathcal{P}$ when averaged should give the pressure of the system P . This pressure for the whole system is analogous to the pressure P for an NVT ensemble obtained as,

$$P = -(\partial A / \partial V)_T$$

where A is the Helmholtz free energy. Since the pressure for the two boxes should be equal in the simulation, we can write $P = \langle \mathcal{P}_1 \rangle = \langle \mathcal{P}_2 \rangle$ where P is the value of the pressure in both the boxes (It should be equal) and

$$\begin{aligned} P = \langle \mathcal{P} \rangle &= \langle (\mathcal{P}_1 + \mathcal{P}_2) / 2 \rangle = \langle \mathcal{P}_1 / 2 \rangle + \langle \mathcal{P}_2 / 2 \rangle \\ &= P_1 / 2 + P_2 / 2 \end{aligned} \quad (3.7)$$

For the pressure in each sub-system, Equation (3.4) reduces to Equation (3.8a) for box 1 and equation (3.8b) for box 2.

$$\frac{\partial \langle \mathcal{P}_1 \rangle}{\partial V} = \beta \left\langle \mathcal{P}_1 \frac{(\mathcal{P}_1 + \mathcal{P}_2)}{2} \right\rangle - \beta \langle \mathcal{P}_1 \rangle \left\langle \frac{\mathcal{P}_1 + \mathcal{P}_2}{2} \right\rangle + \left\langle \frac{\partial \mathcal{P}_1}{\partial V} \right\rangle \quad (3.8a)$$

$$\frac{\partial \langle \mathcal{P}_2 \rangle}{\partial V} = \beta \left\langle \mathcal{P}_2 \frac{(\mathcal{P}_1 + \mathcal{P}_2)}{2} \right\rangle - \beta \langle \mathcal{P}_2 \rangle \left\langle \frac{\mathcal{P}_1 + \mathcal{P}_2}{2} \right\rangle + \left\langle \frac{\partial \mathcal{P}_2}{\partial V} \right\rangle \quad (3.8b)$$

Thus for the individual phases we can write the expression for the inverse of isothermal compressibility as

$$1 / \kappa_{T1} = -V \left(\frac{\partial \langle \mathcal{P}_1 \rangle}{\partial V} \right)_T \quad \text{for phase 1} \quad (3.9a)$$

$$\text{and} \quad 1 / \kappa_{T2} = -V \left(\frac{\partial \langle \mathcal{P}_2 \rangle}{\partial V} \right)_T \quad \text{for phase 2} \quad (3.9b)$$

where V is the total volume of the system.

Starting from the definition of the instantaneous pressure function (Equation 3.5), the expression for $\langle \partial \mathcal{P} / \partial V \rangle$ for each phase is obtained by differentiating $\mathcal{P}$ for each phase with respect to V .

$$\left\langle \frac{\partial \mathcal{P}_1}{\partial V} \right\rangle = -\frac{1}{2} \left[\frac{1}{\beta} \left\langle \frac{N_1}{V_1^2} \right\rangle + \left\langle \frac{\partial^2 U_1}{\partial V_1^2} \right\rangle \right] \quad \text{for phase 1} \quad (3.10a)$$

$$\left\langle \frac{\partial \mathcal{P}_2}{\partial V} \right\rangle = -\frac{1}{2} \left[\frac{1}{\beta} \left\langle \frac{N_2}{V_2^2} \right\rangle + \left\langle \frac{\partial^2 U_2}{\partial V_2^2} \right\rangle \right] \quad \text{for phase 2} \quad (3.10b)$$

From Equations (3.8), (3.9), and (3.10), we get the final expression for the inverse of isothermal compressibility as

$$\frac{1}{\kappa_{T1}} = -V \left\{ \beta \left\langle \mathcal{P}_1 \left(\frac{\mathcal{P}_1 + \mathcal{P}_2}{2} \right) \right\rangle - \beta \langle \mathcal{P}_1 \rangle \left\langle \frac{\mathcal{P}_1 + \mathcal{P}_2}{2} \right\rangle - \frac{1}{2} \left[\frac{1}{\beta} \left\langle \frac{N_1}{V_1^2} \right\rangle + \left\langle \frac{\partial^2 U_1}{\partial V_1^2} \right\rangle \right] \right\} \quad (3.11a)$$

$$\frac{1}{\kappa_{T2}} = -V \left\{ \beta \left\langle \mathcal{P}_2 \left(\frac{\mathcal{P}_1 + \mathcal{P}_2}{2} \right) \right\rangle - \beta \langle \mathcal{P}_2 \rangle \left\langle \frac{\mathcal{P}_1 + \mathcal{P}_2}{2} \right\rangle - \frac{1}{2} \left[\frac{1}{\beta} \left\langle \frac{N_2}{V_2^2} \right\rangle + \left\langle \frac{\partial^2 U_2}{\partial V_2^2} \right\rangle \right] \right\} \quad (3.11b)$$

where V is taken as the total volume of the system.

Chapter 4: Phase Equilibrium Calculations for the BFW Potential

The different algorithms tested in this work have employed the Lennard-Jones potential because of the simplicity to implement and also because it has been widely studied. However, this potential does not model the interactions for the noble gases effectively [17] and is limited in its accuracy. To demonstrate the practical applicability of the GEMC simulations and the new implementations of this technique in the parallel environments, an attempt was made to calculate the phase equilibrium properties of Kr, which is best modeled by the Barker-Fisher-Watts (BFW) potential.

The BFW potential has been derived, taking into consideration a wide range of experimental data and is very accurate for the prediction of noble gas properties. The functional form of this potential is given as

$$u_{BFW}(r) = \varepsilon \left[\sum_{k=0}^5 A_k (x-1)^k \exp(\alpha(1-x)) - \sum_{k=0}^2 \frac{C_{6+2k}}{r^{6+2k} + \delta} + u_1(r) \right] \quad (4.1)$$

where

$$u_1(r) = \begin{cases} P(x-1)^4 + Q(x-1)^5 \exp(\alpha'(1-x)), & x > 1, \\ 0, & x < 1 \end{cases}$$

where ε is the well depth of the potential and $x=r/R_m$ is the normalized internuclear distance, normalized by R_m , which is the position of the potential minimum. The values of the potential parameters for Kr and Ar have been reported by Barker and co-workers [20]. An attempt has been made to calculate the phase coexistence properties of the BFW potential and compare the results for Kr with reported values [19]. Kr was chosen over Ar for the comparison, since the BFW potential is designed to be accurate for Ar only when considering the three-body interactions, which have not been considered here.

Chapter 5: Results and Discussion

5.1 Testing for the Accuracy of the Simulations

The goal of this study is to develop parallel implementations of the conventional GEMC simulation technique. We have concentrated on techniques which involve decoupling of the two boxes and simulating them separately on different processors. As a first step towards a parallel algorithm, it is important to develop a serial code which is accurate.

The values of the pressures, energies and densities obtained from the serial version of the code have been compared with published results [11]. It is important to have a correct serial code because it is the framework for the development of the parallel codes. Any disagreement of results from the parallel algorithm with literature can then be attributed to biases or coding errors in the subsequent procedures tested.

The various codes developed are in the order

- a) Serial code.
- b) Synchronous two node parallel code.
- c) Asynchronous two node parallel code.
- d) Random selection multi-processor code.
- e) Perfectly parallel multi-processor code.
- f) Multi-processor code with swapping.

Each of the procedures has been checked for its correctness before proceeding to the next stage of development. This section presents the results obtained from the various

procedures and the comparison of these results with published values. Section 5.2 deals with the issues of statistical accuracy of the results and the efficiency of the codes.

5.1.1 Results for the Serial Code.

The results obtained from the serial code are presented and compared with published results [11] in Table 5.1. The simulation is divided into a number of blocks for calculating the average values of the properties. The length of each block for the serial code is 75000 configurations. The standard deviation of the mean of the block averages is reported as the uncertainty in the results. As seen, there is a good agreement (within error bars) between the results generated and reported values. A plot of the liquid-vapor coexistence curve has been presented in Figure 5.1 and a graphical comparison of results has been made. It can be concluded from these results that the serial code is accurate and reproduces previous results.

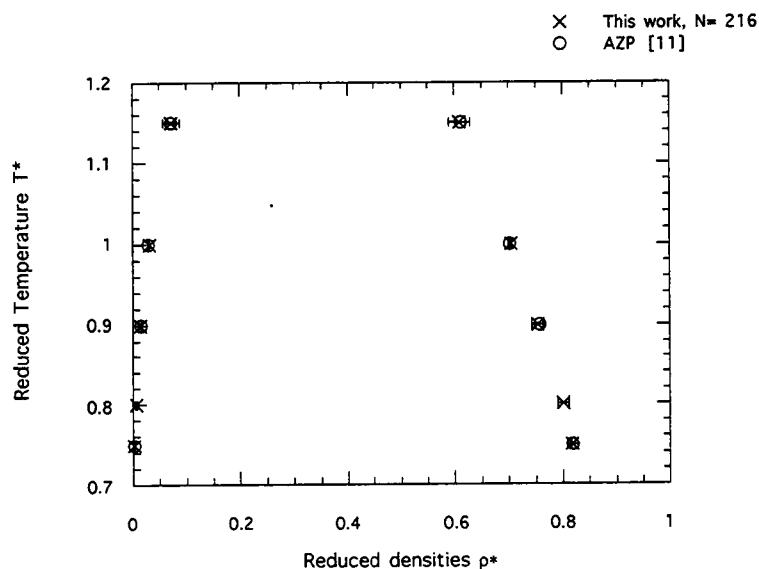


Figure 5.1 Phase coexistence curve for the Lennard-Jones fluid (Serial Code) compared to values reported by Panagiotopolous [11].

Table 5.1 Coexistence properties for Lennard-Jones fluids (Serial code)*

Source	T*	N	ρ_G^*	P_G^*	E_G^*	ρ_L^*	P_L^*	E_L^*
This work	1.15	216	0.072(15)	0.059(7)	-0.58(13)	0.609(19)	0.059(19)	-4.19(13)
A.Z.P. [11]	1.15	500	0.072(9)	0.059(3)	-0.632(9)	0.605(9)	0.059(17)	-4.16(6)
A.Z.P. [11]	1.15	300	0.083(6)	0.064(5)	-0.712(18)	0.612(9)	0.075(24)	-4.20(6)
This work	1.0	216	0.0309(7)	0.0259(5)	-0.278(7)	0.7038(12)	0.0135(23)	-4.920(9)
A.Z.P. [11]	1.0	500	0.0283(6)	0.0241(15)	-0.254(18)	0.703(3)	0.029(22)	-4.91(3)
A.Z.P. [11]	1.0	300	0.0291(6)	0.0246(12)	-0.275(18)	0.702(6)	0.024(27)	-4.90(3)
This work	0.9	216	0.014(2)	0.0115(18)	-0.132(24)	0.754(10)	-0.013(19)	-5.33(8)
A.Z.P. [11]	0.9	300	0.0151(3)	0.0123(6)	-0.145(15)	0.758(9)	0.014(37)	-5.365(6)
This work	0.8	216	0.0066(10)	0.0050(7)	-0.071(15)	0.802(9)	-0.037(16)	-5.75(8)
This work	0.75	216	0.0033(9)	0.0024(7)	-0.037(12)	0.818(8)	-0.052(20)	-5.89(7)
A.Z.P. [11]	0.75	300	0.0031(3)	0.0023(3)	-0.035(3)	0.819(3)	0.019(36)	-5.88(3)

* The temperature, pressure and energies are given in reduced units. The number in parentheses indicate the error in units of last decimal digit: Thus 0.35(2) means 0.35 ± 0.02

5.1.2 Results for the Asynchronous Two-Processor Code

This procedure involves the complete decoupling of the simulation boxes. The translational moves are conducted independently. Though the statistical mechanical considerations remain the same as those of the serial code, there are some changes in the procedural aspects. It is important to check if these procedural changes affect the results in any way. Table 5.2 presents the results obtained from the asynchronous parallel code as compared to the reported values [11]. The uncertainty in the results again represents the standard deviation of the mean of the block averages. The table also presents the values for the chemical potential which was not calculated for the serial code. Since the chemical potentials were obtained from the result of an entire run, the uncertainty estimates could not be calculated. The results of the asynchronous procedure match the literature values within error bars. It is concluded that the asynchronous procedure, i.e. decoupling of the nodes and conducting independent translational moves does not affect the accuracy of the code. The phase coexistence curve calculated from the asynchronous code is presented in Figure 5.2.

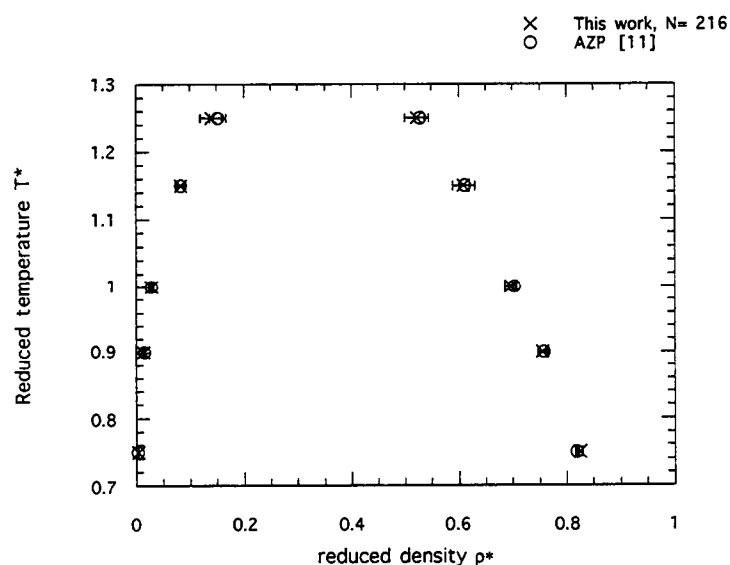


Figure 5.2 Phase coexistence curve for the Lennard-Jones fluid (Asynchronous code).

Table 5.2 Coexistence properties for Lennard-Jones fluids (Asynchronous two processor code)*

Source	T*	N	ρ_G^*	P_G^*	E_G^*	ρ_L^*	P_L^*	E_L^*	μ_G	μ_L
This work	1.25	216	0.14(2)	0.099(6)	-1.09(15)	0.523(22)	0.100(10)	-3.58(14)	-3.575	-3.604
A.Z.P. [11]	1.25	500	0.152(15)	0.101(9)	-1.19(3)	0.529(9)	0.102(16)	-3.61(6)	-3.59	-3.57
A.Z.P. [11]	1.25	300	0.148(3)	0.101(6)	-1.18(3)	0.526(15)	0.108(20)	-3.59(9)	-3.59	-3.56
This work	1.15	216	0.083(8)	0.067(5)	-0.71(8)	0.61(2)	0.065(27)	-4.19(14)	-3.592	-3.652
A.Z.P. [11]	1.15	500	0.072(9)	0.059(3)	-0.632(9)	0.605(9)	0.059(17)	-4.16(6)	-3.70	-3.68
A.Z.P. [11]	1.15	300	0.083(6)	0.064(5)	-0.712(18)	0.612(9)	0.075(24)	-4.20(6)	-3.63	-3.63
This work	1.0	216	0.028(4)	0.027(3)	-0.264(18)	0.697(11)	0.017(9)	-4.87(8)	-3.864	-3.861
A.Z.P. [11]	1.0	500	0.0283(6)	0.0241(15)	-0.254(18)	0.703(3)	0.029(22)	-4.91(3)	-3.88	-3.85
A.Z.P. [11]	1.0	300	0.0291(6)	0.0246(12)	-0.275(18)	0.702(6)	0.024(27)	-4.90(3)	-3.90	-3.85
This work	0.9	216	0.014(2)	0.0126(17)	-0.188(22)	0.755(8)	0.006(11)	-5.33(6)	-3.968	-3.931
A.Z.P. [11]	0.9	300	0.0151(3)	0.0123(6)	-0.145(15)	0.758(9)	0.014(37)	-5.365(6)	-3.95	-3.86
This work	0.75	216	0.0034(4)	0.0025(3)	-0.037(4)	0.825(5)	-0.054(36)	-5.95(4)	-4.024	-4.398
A.Z.P. [11]	0.75	300	0.0031(3)	0.0023(3)	-0.035(3)	0.819(3)	0.019(36)	-5.88(3)	-4.38	-4.35

* The temperature, pressure and energies are given in reduced units. The number in parentheses indicate the error in units of last decimal digit:
Thus 0.35(2) means 0.35 ± 0.02

5.1.3 Results for Multi-Processor Codes.

1) Random-selection multi-processor algorithm.

The random selection procedure in this algorithm necessitates the transfer of messages from a busy processor to a free one indicating that the selection of the processors (for conducting a transfer) is not done in a totally random manner, which might introduce a bias in the simulations. It is unclear how this bias would affect the results. This procedure also allows volume and particle transfers between the boxes of the same phase. Since an equal distribution of the boxes in the two phases is expected, nearly 50 percent of all the transfers may not be contributing towards phase separation (since they occur within the same phase). This suggests that the algorithm is inherently inefficient.

Also the stability of the algorithm is suspect. Because of the transfers between similar phases, the distribution of particles and volumes in the boxes is not governed by mass balance considerations and thus it is not possible to predict these distributions. Some of the vapor phases may become deficient of particles during the course of the simulation and the properties in these boxes may then be dependent on the system size. Also in some of these vapor phases, the boxes may become empty leading to a load imbalance problem. The simulation fails in this case. Theoretically the empty configurations should also be considered within the calculation of the properties, but since the particle transfers are infrequent as compared to the translational moves, it may take the empty processors a considerable number of cycles to gain particles, during which time they may cover a large portion of their allotted cycles. The results for random-selection simulations carried out at $T^*=0.9$ and $T^*=1.0$ have been presented in Table 5.3.

Table 5.3 Coexistence properties for the Lennard Jones fluid (random selection multi-processor code) for $T^*=1.0, 0.9$

	T^*	ρ^*	P^*	E^*
node 0	1.0	0.0293(48)	0.037(7)	-0.33(13)
node 1	1.0	0.0295(27)	0.040(5)	-0.267(43)
node 2	1.0	0.0327(49)	0.044(9)	-0.295(5)
node 3	1.0	0.0304(25)	0.040(6)	-0.284(39)
node 4	1.0	0.0300(28)	0.039(6)	-0.288(29)
Average (5 nodes)		0.0304(14)	0.040(3)	-0.293(2)
A.Z.P.[11] vap	1.0	0.0283(6)	0.0241(15)	-0.254(18)
node 5	1.0	0.703(13)	0.026(10)	-4.91(10)
node 6	1.0	0.704(14)	0.028(20)	-4.92(10)
node 7	1.0	0.707(10)	0.030(15)	-4.95(8)
node 8	1.0	0.702(10)	0.027(19)	-4.90(8)
node 9	1.0	0.705(8)	0.033(17)	-4.92(6)
Average (5 nodes)		0.704(1)	0.0288(24)	-4.92(1)
A.Z.P.[11] liq	1.0	0.703(3)	0.029(22)	-4.91(3)
node 0	0.9	0.758(8)	-0.014(11)	-5.369(73)
node 1	0.9	0.755(8)	-0.011(14)	-5.343(72)
node 2	0.9	0.751(7)	-0.015(15)	-5.317(62)
node 3	0.9	0.754(7)	-0.008(14)	-5.330(60)
node 4	0.9	0.755(9)	-0.011(14)	-5.340(75)
Average (5 nodes)		0.754(2)	-0.0118(24)	-5.339(17)
A.Z.P.[11] liq	0.9	0.758(9)	0.014(37)	-5.365(6)
node 5	0.9	0.0162(16)	0.0194(60)	-0.219(40)
node 6	0.9	0.0115(20)	0.0118(26)	-0.177(55)
node 7	0.9	0.0114(22)	0.0117(26)	-0.133(32)
node 8	0.9	0.0160(24)	0.018(37)	-0.153(32)
node 9	0.9	0.0139(22)	0.0143(22)	-0.154(53)
Average (5 nodes)		0.0138(23)	0.0150(35)	-0.167(29)
A.Z.P.[11] vap	0.9	0.0151(3)	0.0123(6)	-0.145(15)

For both the temperatures, it is observed that there is an equal distribution of the processors in the two phases.

For the state point $T^*=1.0$, the simulation was run on 10 processors for a total of 12000 cycles in two simulations of 8000 cycles and 4000 cycles each. The second simulation was started from the final configurations, saved after the first simulation. The values shown for each processor represent the mean of the block average values over both the simulations (averaged after the equilibration period of the first run). Each cycle consists of 255 translational moves, 4 volume exchanges and 25 particle exchanges or a total of about 3.4 million configurations. For $T^*=1.0$, we see that the results of the liquid phase are fairly consistent with values reported by Panagiotopoulos [11]. The vapor densities of the individual phases and the mean of the processor average (of the density) also agree within error bars. However in spite of the long simulation run, the pressure in the vapor phase does not equilibrate to the correct value and is consistently higher on all the processors. The average value over all the phases is almost twice that reported in literature. These results indicate that a bias is being introduced by the transfer of messages from the busy to the free processors.

For the state point $T^*=0.9$, the total number of cycles was also 12000 (in two simulations of 8000 cycles and 4000 cycles each) with each cycle consisting of 255 translational moves, 7 volume transfers and 35 particle transfers or a total of about 3.5 million configurations. The densities in the liquid phases are higher at $T^*=0.9$, thus decreasing the probability of successful particle transfers. This is the reason for the larger number of particle transfer attempts at $T^*=0.9$. At this state point, the results obtained do not show any significant differences from the reported values. The effect of the bias appears to have been suppressed at $T^*=0.9$.

The running averages of the energies for the random selection procedure have been plotted against the values from the asynchronous code at the same temperatures (Figures 5.3-5.6) to compare the rate of convergence (to equilibrium) for these two procedures. It is seen that the vapor phases in the random selection procedure for $T^*=1.0$ show poor results and a much slower convergence compared to the asynchronous code. For $T^*=0.9$, the results are even worse, and the convergence rate is even slower. The convergence of the liquid phases is almost comparable at both temperatures.

It is also observed that the random selection procedure takes more time than the asynchronous procedure for completing the same number of MC moves. This might be expected since the transfer of messages before a possible coupling can potentially incur a high cost in time.

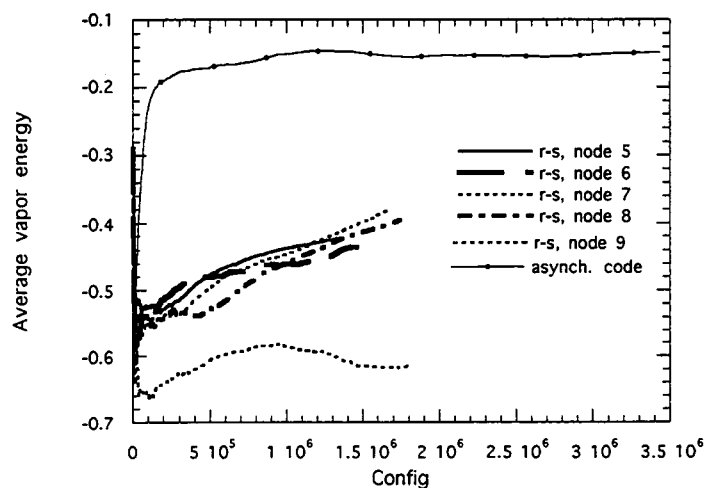


Figure 5.3 Plot of average vapor energies, comparison of random-selection (r-s) and asynchronous codes for $T^*=0.9$.

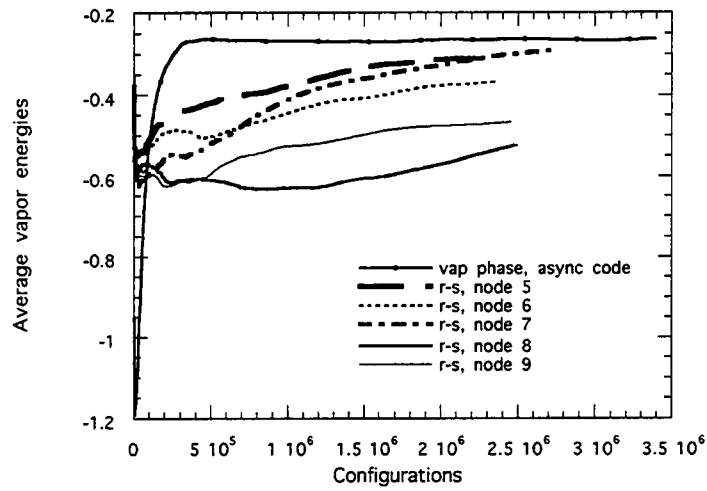


Figure 5.4 Plot of average vapor energies, comparison of random-selection and asynchronous codes for $T^*=1.0$.

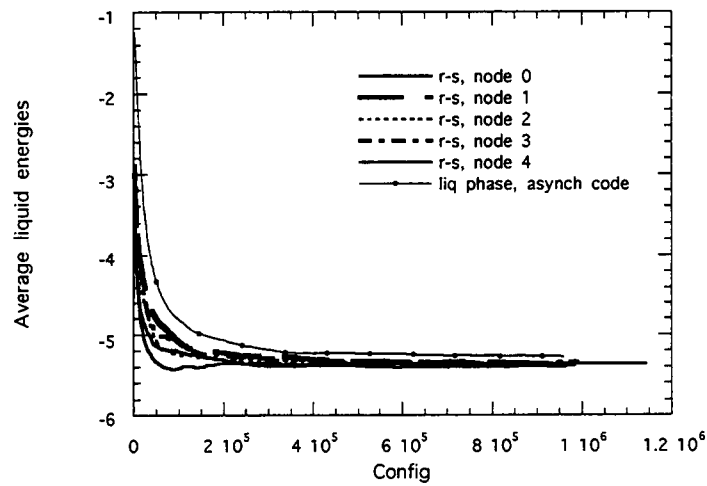


Figure 5.5 Plot of average liquid energies, comparison of random-selection and asynchronous codes for $T^*=0.9$.

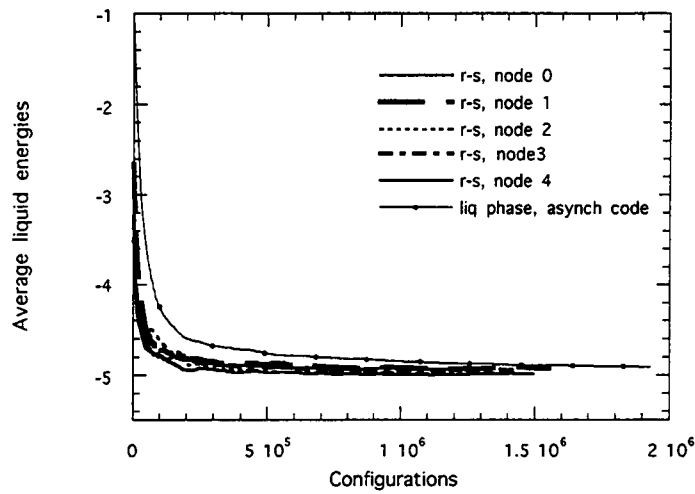


Figure 5.6 Plot of average liquid energies, comparison of random-selection and asynchronous codes for $T^*=1.0$.

From the results obtained from the simulations conducted with the random-selection procedure, it can be concluded that this selection procedure does introduce a bias. Even if we were successful in removing the bias (by precluding the transfer of messages), the procedure would be still unattractive because of the slow convergence rates (possibly due to transfers between similar phases) and the overhead of the excessive message passing.

2) Perfectly parallel procedure

Another implementation of the asynchronous (two processor) code for multiple processors is the perfectly parallel procedure. This method is the replication of the asynchronous 2 node simulation on multiple pairs of processors. The fundamental unit of

the simulation consists of two processors and the asynchronous code can be used without any major modifications. Thus, running a simulation with 20 processors is comparable to running 10 sub-simulations which do not interact. The results for the perfectly parallel simulations using 20 processors have been presented in Table 5.4. The table shows the thermodynamic properties calculated on each processor for the state point $T^*=0.9$.

While calculating the average values, if all the blocks have an equal importance in the averaging, i.e. each block is totally independent of the other, then the standard deviation can be calculated from [14,15]

$$\sigma(\bar{X}) = \sqrt{\frac{\sum_{k=1}^{n_b} (\bar{X}(k) - \langle X \rangle)^2}{n_b(n_b - 1)}} \quad (5.1)$$

where n_b is the number of blocks, $\bar{X}$ is the block average, $\langle X \rangle$ is the mean of block averages, and $\sigma(\bar{X})$ is the standard deviation.

For the perfectly parallel multi-processor code, each processor can be considered as a separate sample which has an equal importance for the calculation of the averages. Thus using equation (5.1) for calculating the variance in the mean of the processor averages, the uncertainty in the results for the simulation taken as a whole should be smaller than that of an individual processor. The indicated errors in the averages (over the whole simulation) represent the statistical uncertainty in the mean of the processor averages. Table 5.5 presents the means of processor averages for all the state points studied. The table also compares the results with those obtained from the serial code and the published values. The results from the parallel code agree with the literature value and

Table 5.4 Results for perfectly parallel multi-processor code for $T^* = 0.9$ showing all 20 processors. The numbers in parentheses represent the standard deviations in the block averages on each processor.

	T	ρ^*	P^*	E^*	state
node 0	0.9	0.763(9)	-0.005(13)	-5.41(8)	liq
node 1	0.9	0.753(8)	-0.017(11)	-5.32(6)	liq
node 2	0.9	0.756(5)	-0.020(9)	-5.36(4)	liq
node 4	0.9	0.758(9)	-0.0002(161)	-5.36(7)	liq
node 5	0.9	0.754(8)	-0.013(14)	-5.34(7)	liq
node 6	0.9	0.755(6)	-0.013(11)	-5.34(5)	liq
node 7	0.9	0.755(8)	-0.011(19)	-5.34(6)	liq
node 8	0.9	0.757(9)	-0.023(16)	-5.36(7)	liq
node 9	0.9	0.750(10)	-0.017(14)	-5.39(8)	liq
node 13	0.9	0.748(4)	-0.018(12)	-5.29(4)	liq
A.Z.P.[11]	0.9	0.758(9)	0.014(37)	-5.365(6)	liq
Average (liq)	0.9	0.755(3)	-0.014(6)	-5.351(3)	liq
node 3	0.9	0.015(3)	0.0121(2)	-0.140(39)	vap
node 10	0.9	0.0131(14)	0.0108(10)	-0.120(14)	vap
node 11	0.9	0.0172(24)	0.0138(17)	-0.164(25)	vap
node 12	0.9	0.0145(25)	0.0118(18)	-0.136(25)	vap
node 14	0.9	0.0154(23)	0.0125(1)	-0.146(23)	vap
node 15	0.9	0.0153(42)	0.0124(3)	-0.144(41)	vap
node 16	0.9	0.0158(25)	0.0128(2)	-0.149(25)	vap
node 17	0.9	0.0145(31)	0.0119(2)	-0.136(30)	vap
node 18	0.9	0.0198(34)	0.0156(2)	-0.190(34)	vap
node 19	0.9	0.0127(29)	0.0105(2)	-0.118(28)	vap
A.Z.P.[11]	0.9	0.0151(3)	0.0123(6)	-0.145(15)	vap
Average (vap)	0.9	0.0153(19)	0.0124(1)	-0.144(19)	vap

Table 5.5 Coexistence properties for Lennard-Jones fluids (Perfectly parallel code). Averages taken over all the processors. The number in the parentheses indicates the standard deviation in the mean of processor averages.*

Source	T*	N	ρ_G^*	P_G^*	E_G^*	ρ_L^*	P_L^*	E_L^*	Number of Processors
This work (S)	1.15	216	0.072(15)	0.059(7)	-0.58(13)	0.609(19)	0.059(19)	-4.19(13)	20
This work (PP)	1.15	--	0.0723(2)	0.059(1)	-0.61(1)	0.608(1)	0.055(1)	-4.18(1)	
A.Z.P. [2]	1.15	500	0.072(9)	0.059(3)	-0.632(9)	0.605(9)	0.059(17)	-4.16(6)	
A.Z.P. [2]	1.15	300	0.083(6)	0.064(5)	-0.712(18)	0.612(9)	0.075(24)	-4.20(6)	
This work (S)	1.0	216	0.0309(7)	0.0259(5)	-0.278(7)	0.7038(12)	0.0135(23)	-4.920(9)	20
This work (PP)	1.0	--	0.0297(6)	0.0250(4)	-0.270(5)	0.703(1)	0.009(1)	-4.91(1)	
A.Z.P. [2]	1.0	500	0.0283(6)	0.0241(15)	-0.254(18)	0.703(3)	0.029(22)	-4.91(3)	
A.Z.P. [2]	1.0	300	0.0291(6)	0.0246(12)	-0.275(18)	0.702(6)	0.024(27)	-4.90(3)	
This work (S)	0.9	216	0.014(2)	0.014(18)	-0.132(24)	0.754(10)	-0.013(19)	-5.33(8)	20
This work (PP)	0.9	--	0.0153(6)	0.01240(3)	-0.144(6)	0.755(1)	-0.014(2)	-5.35(1)	
A.Z.P. [2]	0.9	300	0.0151(3)	0.0123(6)	-0.145(15)	0.758(9)	0.014(37)	-5.365(6)	
This work (S)	0.8	216	0.0066(10)	0.0050(7)	-0.071(15)	0.802(9)	-0.037(16)	-5.75(8)	
This work (PP)	0.8	--	0.0060(3)	0.0046(2)	-0.059(4)	0.799(1)	-0.038(13)	-5.720(7)	20

* The temperature, pressure and energies are given in reduced units. The number in parentheses indicate the error in units of last decimal digit: Thus 0.35(2) means 0.35 ± 0.02 . S stands for the serial code. PP stands for the perfectly parallel procedure.

with those from the serial code within error bars. This indicates that the code is correct. Also the mean of the processor average has a smaller uncertainty than the serial code. This is expected as the parallel codes sample more configurations and have improved statistical accuracy. This issue is discussed in greater detail in Section 5.2.

Figures 5.7 and 5.8 present the plot of the average vapor and liquid energies over all 10 processors of the phases for the state point $T^*=0.9$. It is seen from these plots that the different vapor processors have a different rates of convergence (to equilibrium). The convergence rates for the liquid energies are almost comparable. The difference in the convergence rates for the vapor phases could be due to the smaller fluctuations of the density in some of the phases (which could be attributed to the lower temperature). Some of the vapor phases may thus be equilibrating slowly because the random fluctuations may not be large enough to bring these phases out of the metastable region. The slower phases can be perturbed in some way to enforce faster equilibration. A possible perturbation could be the swapping of processors after a certain number of cycles.

3) Multi-processor procedure with swapping.

The results for the parallel procedure with swapping have been presented in Table 5.6. The reported values are the means of the processor averages, with the values in parentheses representing the standard deviation in the means. The table compares the results for the swapping procedures at different swap frequencies to that from the perfectly parallel procedure, the serial code and the published results [11]. The swap period has been defined as the number of cycles before a swapping of processors occurs and the inverse of the swap period is the swap frequency. The results presented in Table

5.6 suggest that the parallel procedure with swapping generates results comparable to the other procedures and the published values (within error bars).

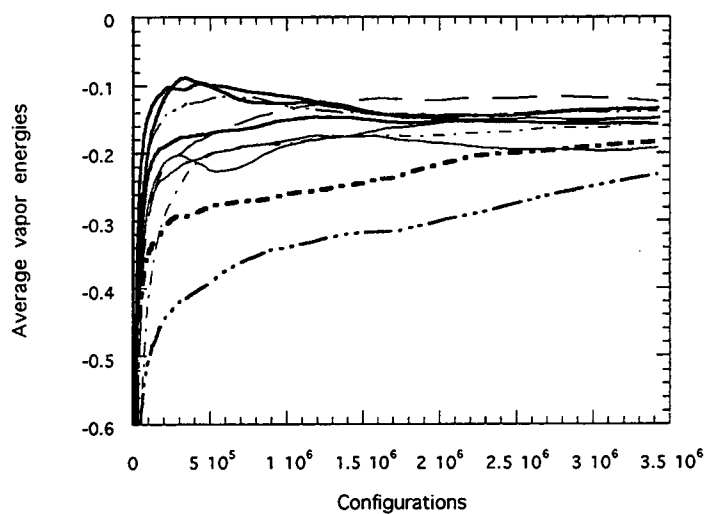


Figure 5.7 Plot of average vapor energies of all the 10 vapor processors for $T^*=0.9$.

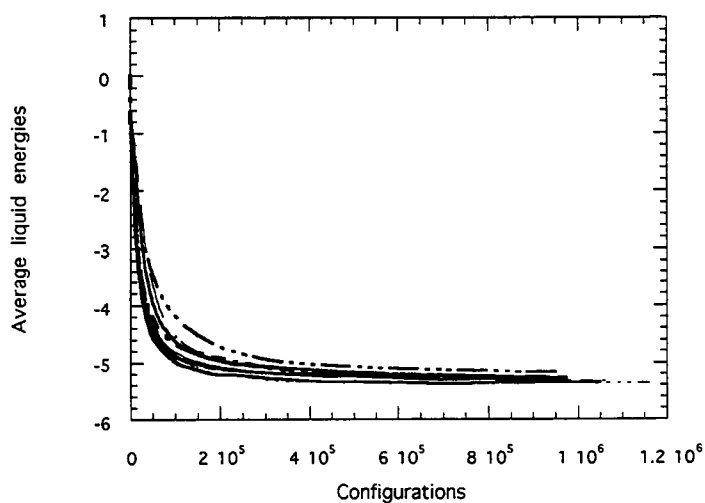


Figure 5.8 Plot of average liquid energies of all 10 liquid processors for $T^*=0.9$.

Table 5.6 Coexistence properties for Lennard-Jones fluids (Parallel multi-processor code with swapping). Averages taken over all the processors. The number in the parentheses indicates the standard deviation in the mean of processor averages.*

Source	T*	N	ρ_G^*	P_G^*	E_G^*	ρ_L^*	P_L^*	E_L^*	Swap frequency (cycles)
This work (S)	1.15	216	0.072(15)	0.059(7)	-0.58(13)	0.609(19)	0.059(19)	-4.19(13)	
This work (PP)	--	--	0.0723(2)	0.059(1)	-0.61(1)	0.608(1)	0.055(1)	-4.18(1)	--
This work (Sw)	--	--	0.0684(9)	0.058(1)	-0.57(1)	0.604(1)	0.052(2)	-4.15(1)	1000
A.Z.P. [11]		500	0.0727(16)	0.059(10)	-0.61(1)	0.606(1)	0.053(1)	-4.170(7)	600
			0.072(9)	0.059(3)	-0.632(9)	0.605(9)	0.059(17)	-4.16(6)	
This work (S)	1.0	216	0.0309(7)	0.0259(5)	-0.278(7)	0.7038(12)	0.0135(23)	-4.920(9)	
This work (PP)	--	--	0.0297(6)	0.0250(4)	-0.270(5)	0.7030(10)	0.0090(10)	-4.910(10)	--
This work (Sw)	--	--	0.0293(4)	0.0248(3)	-0.266(4)	0.7039(7)	0.0101(13)	-4.923(6)	500
			0.0308(3)	0.0259(2)	-0.279(2)	0.7017(1)	0.0090(15)	-4.906(1)	250
A.Z.P. [11]		500	0.0295(5)	0.0249(3)	-0.267(5)	0.7029(7)	0.0078(23)	-4.917(5)	100
			0.0283(6)	0.0241(15)	-0.254(18)	0.703(3)	0.029(22)	-4.91(3)	
This work (S)	0.9	216	0.0140(20)	0.014(18)	-0.132(24)	0.754(10)	-0.013(19)	-5.33(8)	--
This work (PP)	--	--	0.0153(6)	0.01240(3)	-0.144(6)	0.755(1)	-0.0140(20)	-5.350(10)	--
This work (Sw)	--	--	0.0148(4)	0.0121(3)	-0.136(4)	0.752(1)	-0.0120(10)	-5.310(10)	500
			0.0152(2)	0.0123(1)	-0.143(2)	0.756(1)	-0.0170(10)	-5.350(7)	250
			0.0145(5)	0.0118(4)	-0.136(5)	0.753(1)	-0.0190(17)	-5.330(7)	100
A.Z.P. [11]		300	0.0151(3)	0.0123(6)	-0.145(15)	0.758(9)	0.014(37)	-5.365(6)	--
This work (S)	0.8	216	0.0066(10)	0.0050(7)	-0.071(15)	0.802(9)	-0.037(16)	-5.75(8)	
This work (PP)	--	--	0.0060(3)	0.0046(2)	-0.059(4)	0.7990(10)	-0.0380(13)	-5.720(7)	--
This work (Sw)	--	--	0.0052(3)	0.0040(2)	-0.051(3)	0.7990(10)	-0.0410(20)	-5.730(10)	500
			0.0054(2)	0.0042(2)	-0.053(2)	0.7960(13)	-0.0350(43)	-5.700(13)	250
			0.0051(2)	0.0039(2)	-0.049(2)	0.7970(10)	-0.0380(17)	-5.790(10)	100

* The temperature, pressure and energies are given in reduced units. The number in parentheses indicate the error in units of last decimal digit:

Thus 0.35(2) means 0.35 ± 0.02 . S stands for the serial code. PP stands for the perfectly parallel procedure and Sw stands for the parallel procedure with swapping. The swapping frequency is the number of cycles before swapping of the nodes

It is thought that the difference in the equilibration rates of sub-systems (Figure 5.7) in the perfectly parallel procedures may be attributed to these sub-systems getting caught in the metastable region. The swapping of processors aims to alleviate this problem and help the sub-systems equilibrate at a more uniform rate. The equilibration rates for the swapping procedure and the perfectly parallel procedure have been studied in Section 5.2.

This section has mainly dealt with the accuracy of the results obtained by the different procedures developed in this work. The results for the serial code suggest that the code is correct. The asynchronous code has some procedural changes such as the decoupling of the boxes and independent translational moves on each box, which might affect the results. The results suggest that these procedural changes do not affect the accuracy of the code. While extending the asynchronous code to multiple processors with the random-selection procedure, there exists the possibility of introducing a bias in the transfer moves. This may be reflected in the results for $T^*=1.0$ (Table 5.4), though we cannot be certain since the results for $T^*=0.9$ are consistent with prior work. Also as seen from Figures 5.3 and 5.4, the convergence rate for the vapor phases in the random selection procedure is extremely slow. Thus this procedure has not been investigated further. For the implementation of the asynchronous two node code in a perfectly parallel way on multiple processors, the averages taken over all the processors show improved statistical accuracy. This is expected as the parallel simulations generate more configurations and sample configuration space more effectively. Swapping of processors is another modification to the perfectly parallel procedure which aims to help in the uniform equilibration of all the sub-systems. Results (Table 5.6) indicate that the swapping of the processors does not affect the accuracy and correctness of the results.

5.2 Tests for the Efficiency of the Simulations

The previous section compared the results generated by the codes in this work with published values to ascertain the accuracy and correctness of the codes. This section discusses the issue of the statistical accuracy of the codes and compares the efficiency of the parallel procedures.

The different techniques in molecular simulation (Monte Carlo and molecular dynamics simulations) attempt to calculate the macroscopic properties of the system by generating multiple configurations of the same system and then averaging the properties over all the configurations. Each configuration represents a unique point in configuration phase space and the effectiveness of sampling phase space can be a measure of the efficiency of the simulation. MD simulations attempt to sample phase space effectively by running for a longer time thereby following the trajectories of the particles through a longer distance. Thus making an MD algorithm efficient involves speeding up the code and covering more time intervals in the same wall clock time. Also the efficiency depends on the value of the time interval Δt . If Δt is too small then the total time covered would be small and the phase space would not be sampled efficiently.

MC simulations attempt to sample phase space by generating configurations following a fixed distribution and the statistical accuracy of the results increases with the increase in the number of configurations sampled. Similar to an MD simulation, for an MC simulation, it is the value of the maximum displacements Δr (translational displacement) and ΔV (volume displacement) which affect the sampling of the phase space. If these displacements are too small, then the phase space is not sampled

efficiently as the new configurations are not very different from the original ones. On the other hand if the displacements are too large, the probability of acceptance decreases and the moves are not accepted. Thus, there has to be an optimum value of the displacements (which for simple fluids is adjusted to give an acceptance ratio of around 50 percent).

An MC simulation can be considered efficient if it samples more phase space (by generating more configurations using optimum values for the displacements) in the same wall clock time. Generating more configurations for an MC simulation is thus analogous to speeding up the MD simulation. Using this criterion to define the efficiency of a Monte Carlo simulation, it is seen that the parallel procedures may be more efficient than the serial procedures because of the larger number of configurations sampled in the same wall clock time. This is reflected in the results obtained from the parallel procedures (Tables 5.5 and 5.6) compared to those from the serial codes.

5.2.1 Comparison of the Serial and Asynchronous Parallel Procedures

The asynchronous code involves decoupling of the two simulation boxes and conducting independent translational moves. Both the boxes in the asynchronous code initiate transfer moves, thus the total number of prescribed transfers (to be attempted) should be half that of the serial code. For the purpose of comparing the efficiency of the two procedures, simulations have been run at $T^*=0.9$ for 15000 cycles each. The serial code was run with $N_{trans}=270$, $N_{vol}=4$, and $N_{mol}=45$, while the asynchronous code had $N_{trans}=200$, $N_{vol}=2$, and $N_{mol}=25$. Thus the number of moves initiated by each box are approximately half of those for the serial code to maintain the total number of moves attempted equal. The lower temperature was chosen for the comparison of the two codes since the equilibration rates are slower, and the effect of increase in equilibration rates

would be more pronounced. The plots of the average energies for the two runs have been plotted (Figures 5.9 and 5.10). It is seen that the asynchronous code shows a faster approach to equilibrium for the vapor phase.

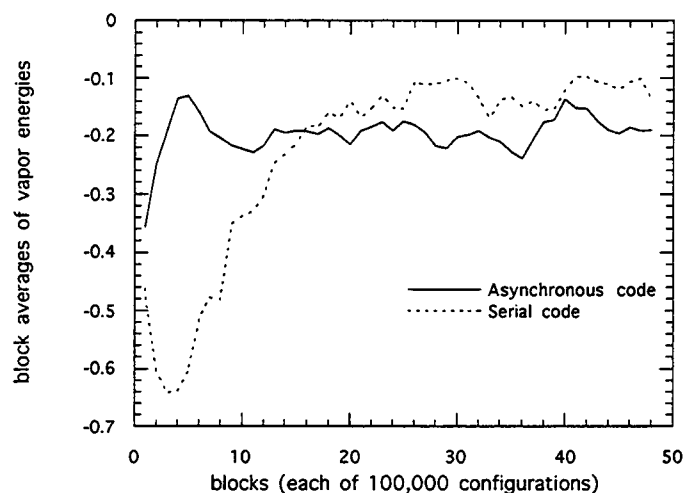


Figure 5.9 Plots of block averages of vapor energies for the asynchronous and the serial code.

Quantitatively, it is seen that the serial code equilibrates in 22 blocks (100,000 configurations each) or a total number of 2.2 million configurations. The asynchronous code equilibrates in 8 blocks or a total of 0.8 million configurations. The results for the vapor phase calculated as the mean of the block averages (after equilibration) are given as

	Asynchronous	Serial	Literature
density*	0.0146(26)	0.0140(22)	0.0151(3)
pressure*	0.0126(17)	0.0115(18)	0.0123(6)
energy*	-0.188(22)	-0.132(24)	-0.145(15)

Though the plot suggests an improvement in the equilibration rates for the vapor phase using the asynchronous code, the results are almost comparable for the two codes. This may be seen because the length of the production run may not be large enough for either, the asynchronous code or the serial code. The plots of the average liquid densities, however, do not show any appreciable difference in the equilibration rates as seen from Figure 5.10.

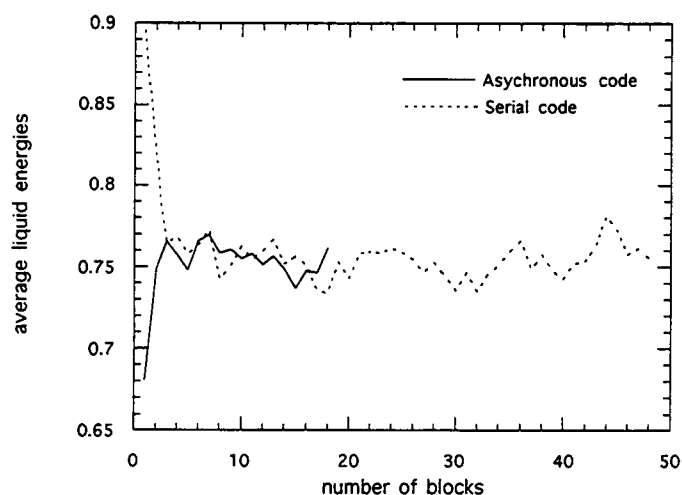


Figure 5.10 Plot of block averages of liquid energies for the serial and asynchronous parallel procedures.

Since the two box GEMC simulation forms the basic unit for the perfectly parallel approach, it will be beneficial to have an efficient two box algorithm as the basic unit. Since the asynchronous code appears to be more efficient than the serial code, it is replicated on multiple pairs of processors.

5.2.2 Comparison of the Perfectly Parallel and Swapping Procedures.

If a simulation is started from an FCC lattice (cold start), it takes some time for the lattice to melt down. This period of the simulation during which the system is being randomized is called the equilibration period. The properties calculated in the simulation are accumulated after the equilibration period (during the production time). For perfectly parallel procedures, it is important that the equilibration period be relatively small compared to the production time since each system running in parallel has to equilibrate independently (and the equilibration time represents time lost). Parallel procedures are thus more efficient if the equilibration time is small, which increases the production time and the statistical accuracy of the results.

For the perfectly parallel multi-processor approach, it is observed that some of the processors equilibrate slower than the others. This increase in the equilibration time means a corresponding decrease in the production time and represents a loss in efficiency of the procedure. The slow equilibration may be caused by a sub-system getting caught in the metastable region or it may be a random occurrence. If some of the sub-systems are indeed getting caught in the metastable region, they could be brought out of the region by introducing a perturbation. Swapping of the processors might provide the necessary perturbation to the slow processors and may lead to the uniform equilibration of all the sub-simulations, increasing the overall efficiency of the algorithm. On the other hand, if the slower equilibration is a random occurrence, the swapping of the nodes should not affect the results to a great extent.

The convergence rate (to equilibrium) from the swapping and the perfectly parallel procedures can be compared by plotting the average energies of the phases

against the number of configurations sampled. For a representative run at $T^*=0.9$ with 10 processors, the average energies of the individual processors have been plotted for the perfectly parallel procedure (without swapping) and for the swapping procedure (swapping at 500 cycles) (Figure 5.11 and 5.12). For the perfectly parallel procedure, it is seen that 2 of the 5 processors in the vapor phase equilibrate slowly. The corresponding plot for the swapping procedure at the same simulation conditions shows a more uniform equilibration rate for all the 5 processors.

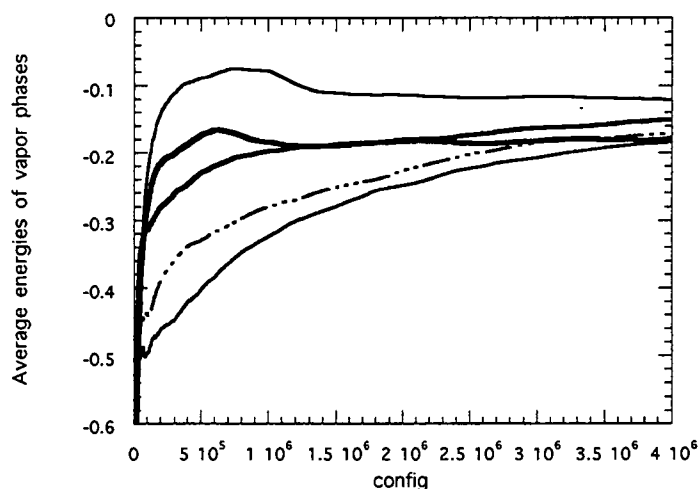


Figure 5.11 Plots of the average energies (for all processors) for runs without swapping at $T^*=0.9$

The visual comparison of the plots in Figure 5.11 and 5.12 suggest that at $T^*=0.9$, the swapping of the processors does help in the uniform equilibration of the sub-systems. At lower temperatures, the fluctuations in the density are smaller than those at higher temperatures. Thus some of the sub-systems may take a longer time to get out of the

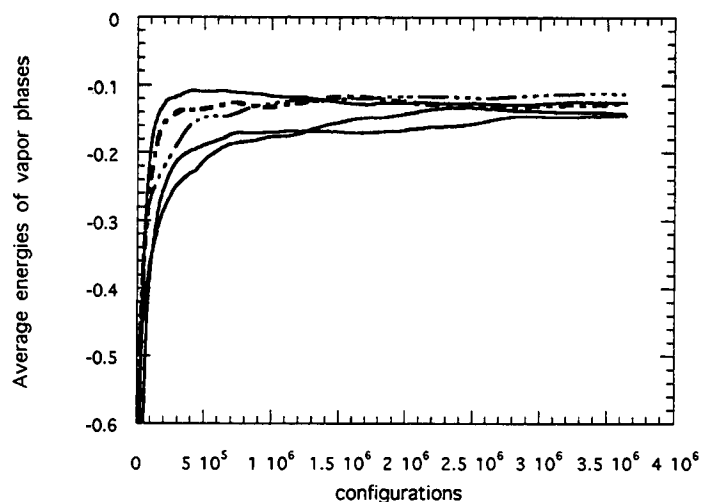


Figure 5.12 Plots of the average energies (for all processors) for runs with swapping at 500 cycles for $T^*=0.9$.

metastable region. The swapping of processors may be providing the perturbation which helps these sub-systems to equilibrate faster. However, there is a possibility that the faster equilibration seen in this simulation (with swapping) may be a random occurrence (especially in the vapor phases, where the system size is very small and fluctuations in the properties are very large).

An attempt to quantify the equilibration rate has been made by plotting the average energies (over all the processors) against the number of configurations for the two simulations. Though the plot of the instantaneous energy may not follow an exponential behavior, it is seen that the plot of the average energy is almost an exponential curve (Figure 5.13). The average energy curve (averaged over all five vapor

processors) can be linearized for the different swap frequencies by fitting it to the empirical equation

$$\frac{E - E_f}{E_o - E_f} = \exp(-\lambda N_{config}) \quad (5.2)$$

where E is the average value of the energy (averaged over all vapor phases), E_f is the final value approached, and E_o is the initial value of the averaged energy.

The slope of the linear plot may give us the quantification of the convergence rate. The linearized plots for the two runs (with swapping at 500 cycles and without swapping) and for other swap frequencies (swapping at 250 and 100 cycles) are compared on the same plot (Figure 5.14). It is seen that the slope of the linear plot for the swapping procedure is more than that for the perfectly parallel procedure, suggesting a higher rate of convergence. The values of λ (slope of the linear plot) have been calculated taking into consideration the middle linear part of the plots. The values obtained for λ are 0.235, 0.384, 0.448 and 0.733 for the procedures without swapping, with 500 swaps, with 250 swaps and with 100 swaps respectively. The extent of error for the λ values for the case without swapping, swaps at 500 and swaps at 100 is likely to be less as the middle portion of the curves is almost linear. However, for the case of swaps at 250, the curve is not linear and the average value of the slope has been reported. In this case the error in the slope may be larger. It is seen that the values for the slopes increase with increase in frequency of swapping, suggesting an improvement of the convergence rate with swapping at the state point $T^*=0.9$.

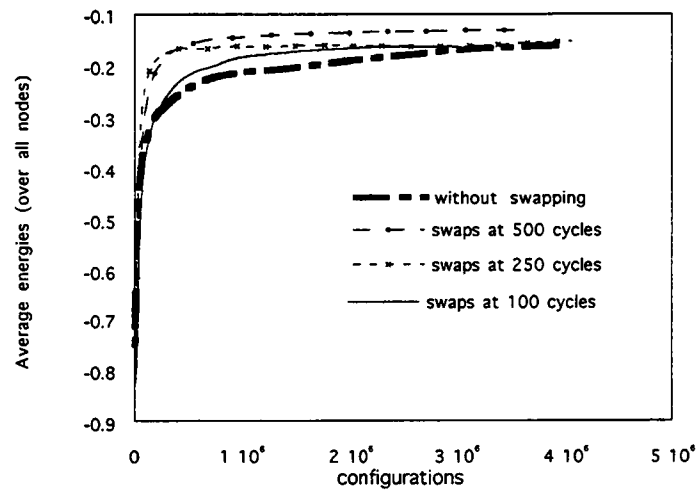


Figure 5.13 Plot of average energies for different swapping frequencies at $T^*=0.9$

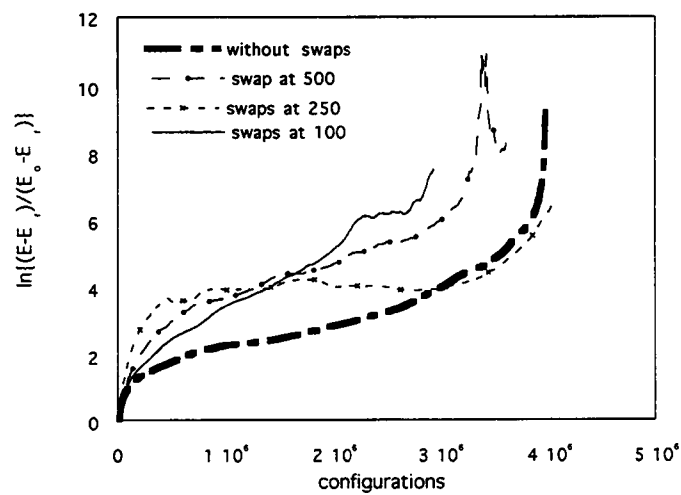


Figure 5.14 Linear plots for different swapping frequencies at $T^*=0.9$

Similar comparisons for $T^*=1.0$ (10 processors) give the value of λ as 0.25 for the perfectly parallel run, and 1.22 for the swapping frequency of 500 cycles. Thus for $T^*=1.0$ also, an increase in the equilibration rate is seen with the swapping procedure. The linearized plots are shown in Figure 5.15.

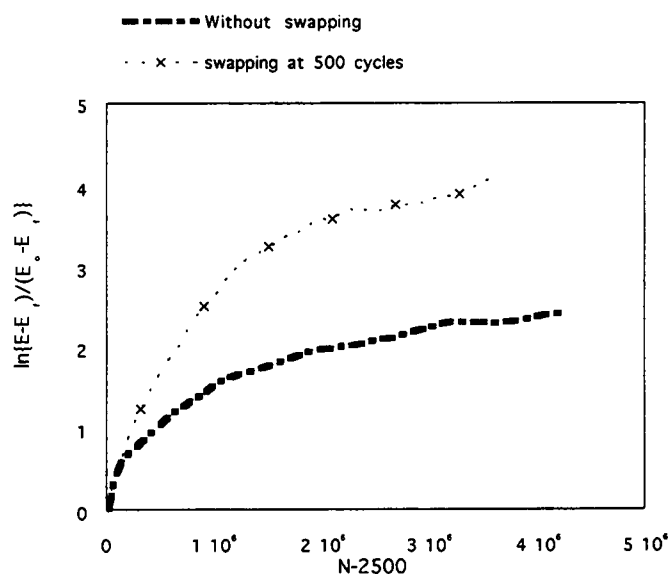


Figure 5.15 Linear plots for the average vapor energies for the case without swapping and for swapping at 500 cycles at $T^*=1.0$ (5 processors in the vapor phase).

The convergence rates can also be studied by calculating the standard deviation of the mean of average energies taken over all the vapor or liquid processors. This standard deviation should give the difference of the individual processor energies from the average energy of the whole phase. This difference can then be used as a measure to compare the convergence rates of the different simulation runs. If some of the processors equilibrate slowly, the energies will be divergent from the average energy of the phase and this will reflect as a larger standard deviation for the mean value. This can then be used as a measure to compare the convergence rates of two simulation runs. The averages over all the nodes and difference from the mean values (the standard deviation) have been plotted

in Figure 5.16-5.19 for simulations at $T^*=0.9$ (20 processors, with $N_{trans}=205$, $N_{vol}=2$, $N_{mol}=30$) for swapping at different frequencies (without swapping and with swapping at 100, 250 and 500 cycles)

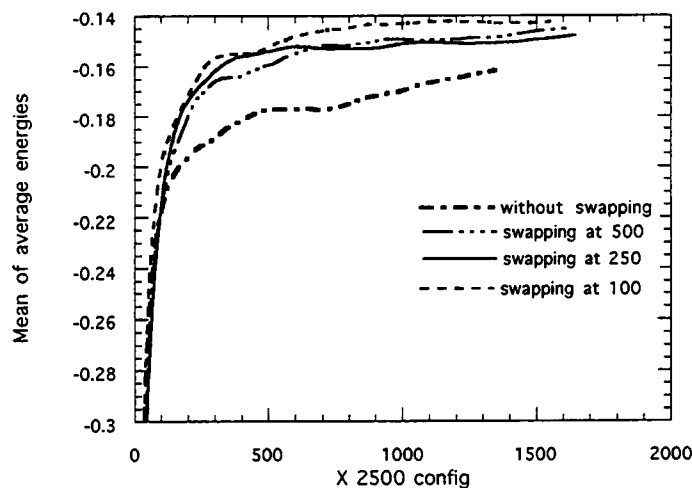


Figure 5.16 Plot of the average vapor energies (over all vapor phases) at $T^*=0.9$.

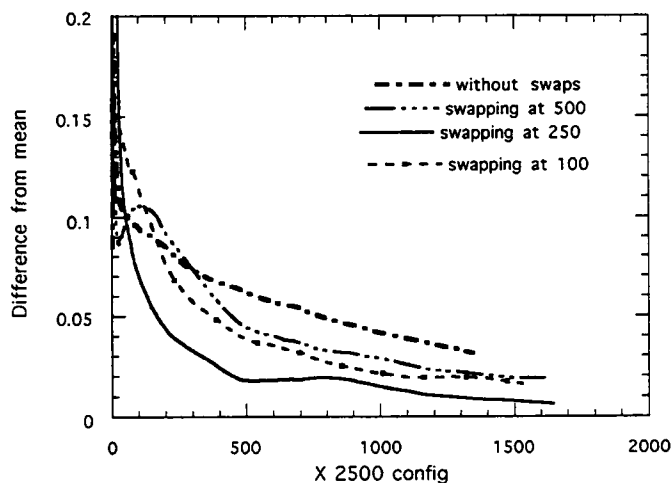


Figure 5.17 Plot of the difference from the mean of the vapor averages at $T^*=0.9$.

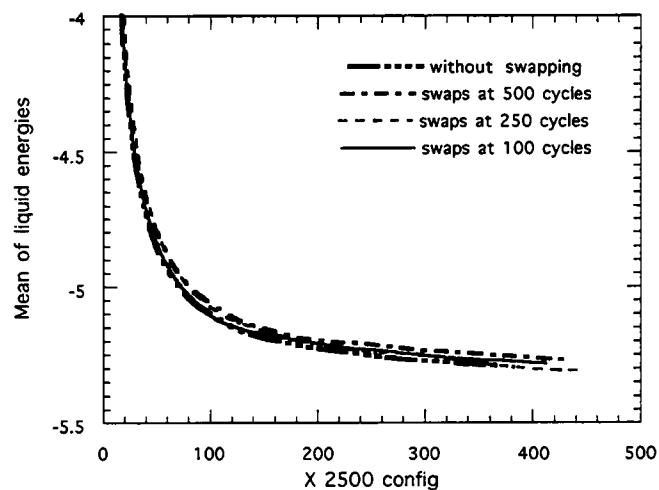


Figure 5.18 Plot of the average liquid energies (over all vapor phases) at $T^*=0.9$.

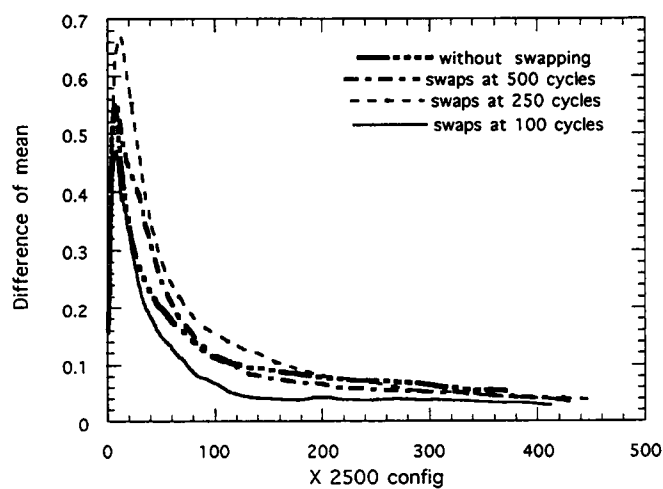


Figure 5.19 Plot of the difference from the mean of average liquid energies at $T^*=0.9$.

As seen from the plots of the average energies and the difference from the mean (of averages values), the average energies for the liquid phase converge in a comparable manner. The difference from mean may decrease slightly with an increase in swapping frequency. However, the difference is very small and could also be a random occurrence. The improvement in the convergence of the liquid energies, if any, is small. For the vapor phases, the swapping clearly increases the rate of convergence of the average energies (Figure 5.16). The differences from the mean for higher swap frequencies (250 and 100) are lower than the swap frequency of 500 and the perfectly parallel simulation. These results suggest that the swapping of nodes results in a better convergence of the vapor phases. The improvement in rate of convergence for the vapor phases may be a result of the perturbations applied to the system by the swapping. Swapping at a higher frequency may be the most beneficial, however, the difference between the curves (Figure 5.16) is too small to make any definitive conclusions.

The plots of the average energies (of all processors) and the difference from the means for $T^*=0.8$, 1.0 and 1.15 are presented below in Figures 5.20-5.25.

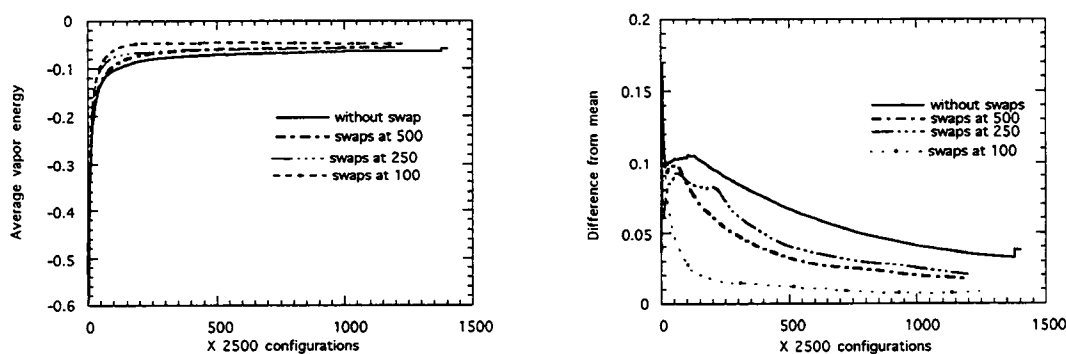


Figure 5.20 Plot of the average vapor energies and difference from the mean values at $T^*=0.8$.

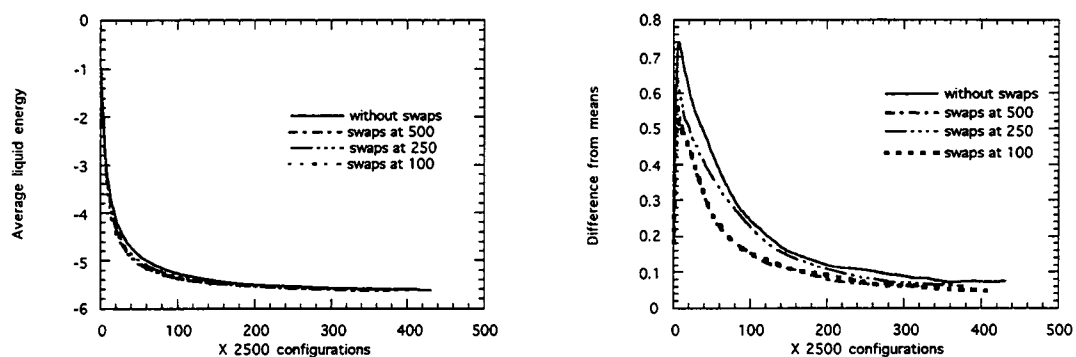


Figure 5.21 Plot of the average liquid energies and difference from the mean values at $T^*=0.8$.

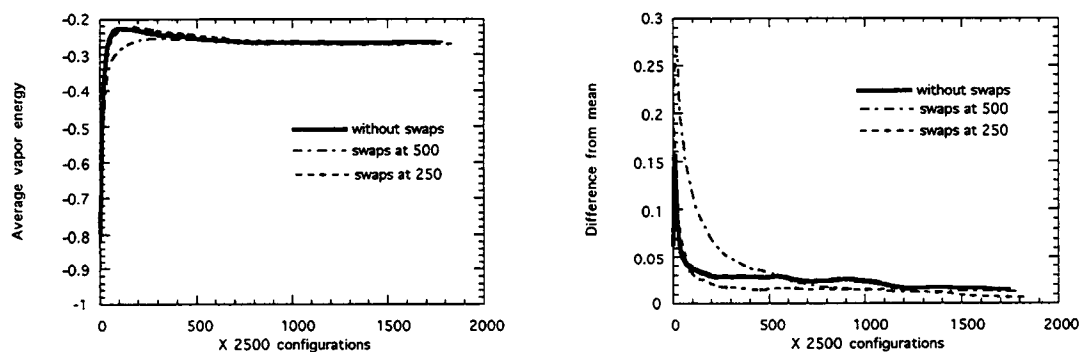


Figure 5.22 Plot of the average vapor energies and difference from the mean values at $T^*=1.0$.

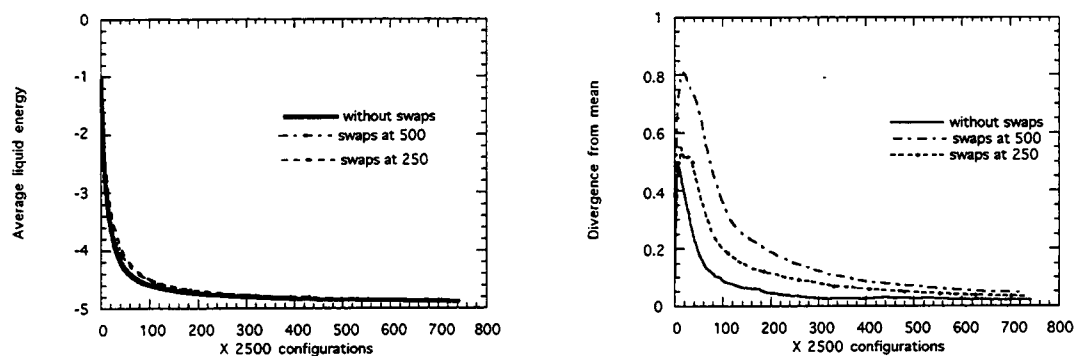


Figure 5.23 Plot of the average liquid energies and difference from the mean values at $T^*=1.0$.

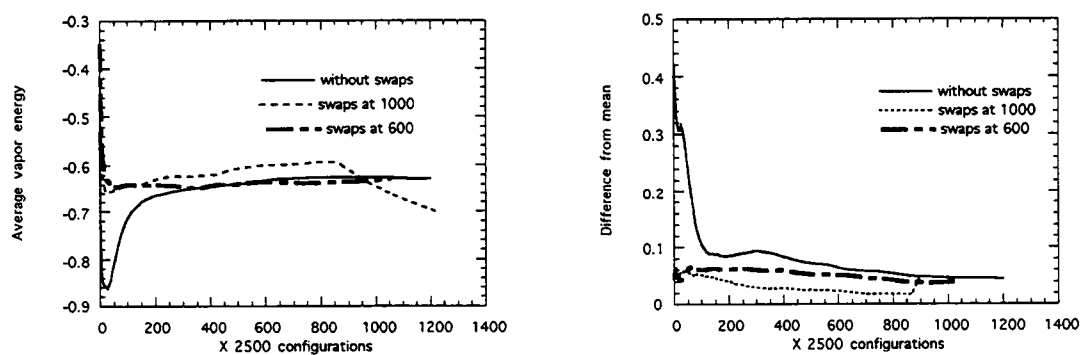


Figure 5.24 Plot of the average vapor energies and difference from the mean values at $T^*=1.15$.

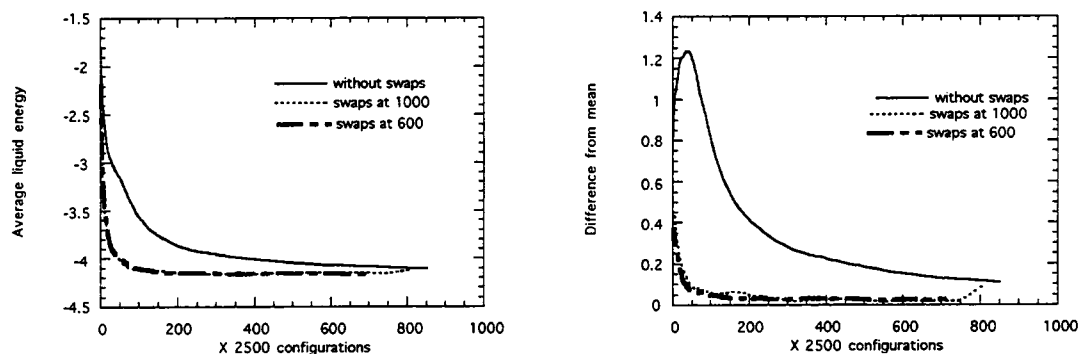


Figure 5.25 Plot of the average liquid energies and difference from the mean values at $T^*=1.15$.

It is seen that, for the state point $T^*=0.8$, the means of the average liquid energies (over all the nodes) converge at almost the same rate for all the different procedures. The swapping of the nodes however, helps to reduce the difference from the mean values, suggesting that it helps in the uniform convergence of the phases. The gain in the convergence rate is small however, and does not make any significant difference in the statistical accuracy of the results as seen in Table 5.6. The vapor phases benefit from the swapping as seen in Figure 5.20 and show a faster rate of convergence with the increase in swap frequencies. Also, the difference from the mean decreases and is the lowest at the highest frequency of 100 cycles/swap.

At higher temperatures ($T^*=1.0$, $T^*=1.15$), no significant difference in the convergence rates of the mean of the average energies is seen. The differences from the mean do not show any fixed patterns and may be random occurrences. For example, at

$T^*=1.0$, the perfectly parallel procedure shows the best convergence rate and the smallest difference from the mean for the liquid phases which is counter-intuitive.

The results suggest that the different sub-simulations have different equilibration rates. This difference could be either attributed to random fluctuations (especially in the vapor phases, where the system sizes are very small) or could be attributed to the sub-systems getting trapped in the metastable regions. It is seen that at lower temperatures ($T^*=0.8$, $T^*=0.9$), the swapping of nodes may improve the convergence rates. The difference in equilibration rates may thus be attributed to the metastability of the sub-systems. This is also supported by the fact that at lower temperatures the fluctuations in density (or other properties) are not as large. However at higher temperatures, the fluctuations are enough for the sub-systems to come out of the metastable region (if they enter them) and the swapping of nodes may be unnecessary. The difference in the convergence rates can then be entirely attributed to random fluctuations within the systems.

5.3 Results for Isothermal Compressibility Calculations

The expression for the isothermal compressibility in the Gibbs Ensemble was presented in Chapter 3 (Equation 3.11). The simulations have been conducted, starting with 108 molecules in each box and have been run on 6 processors. The isothermal compressibility (κ_T) has been calculated from the average values of the fluctuations in each phase and thus one value for κ_T is obtained from each processor. All the simulation have been run for a total of 8000 cycles with the initial 1000 cycles equilibrating the systems. For higher temperatures, the fluctuations may be very large and the results may be affected by the

small system sizes. To study the system size effect, simulations at higher temperatures ($T^*=1.3$) have also been conducted with 256 molecules in each box.

To study the effect of a distributed procedure on the fluctuations in the system, simulations with the swapping procedure have been conducted at $T^*=0.9$ and $T^*=1.0$. The swapping period for each of these simulations is 500 cycles. Results from the simulations have been compared with those from the Lennard-Jones equation of state (Nicolas equation, reparameterised by Johnson and Gubbins) [16]. The simulation results seem to show the right trends (as the eos results) and are numerically comparable with the eos results (within the same order of magnitude). However a lot of scatter is seen in the simulation results and though they seem to be very promising, we cannot comment with certainty about their correctness. The results from the simulations have thus been presented and compared with the eos results in Appendix B.

5.4 Results for the Phase Equilibrium Calculations of the BFW Potential

An attempt has been made to calculate the phase diagram of the Barker-Fisher-Watts (BFW) potential (Equation 4.1) and compare it with published results for Kr. The simulation has been carried out on 108 molecules starting from an FCC lattice. The cutoff distance used is $2.5R_m$ where R_m is the position of the potential minimum. Long range corrections (LRCs) have been applied to compensate for the cutoff. Because of the complexity of the potential, it was not possible to integrate the potential to obtain the expressions for the LRCs. Numerical integration has been applied instead, to obtain the values. The integration procedure was tested with the Lennard-Jones potential and the LRCs obtained were found to agree with the values obtained from the integrated expressions for the Lennard Jones potential. The potential was implemented both in the

serial and the asynchronous parallel codes. The serial code was used for the testing of the phase calculations.

The potential was tested by running the simulation for an NVT ensemble and comparing the values for the energy obtained with reported values for the configurational energy of a krypton-argon mixture at zero strain rate [17]. The value reported for pure krypton is -7.945 kJ/mol at $T=135$ K. The value of the energy obtained from the NVT simulation carried out with the BFW potential at similar conditions for the density and temperature is -8.066 kJ/mol. The difference for the two values is of the order of 1.5 %. Thus implementation of the potential appears to be correct. However the phase calculations yield an expanded phase diagram, with the liquid densities greater and the vapor densities less than the reported values [18,19]. The pressures in the two phases are inconsistent, suggesting an error in the phase calculations. Since the results are not reliable, they have thus been presented in Appendix C.

Chapter 6: Conclusions and Recommendations

The main goal of this study was the development of efficient algorithms for the implementation of the Gibbs ensemble Monte Carlo simulations in a parallel environment. Since the boxes in a GEMC simulation are fairly independent of each other, we have experimented with techniques involving the decoupling of the two boxes and simulating them on different processors.

It is first important to develop a correct serial code as it is the framework for all the parallel codes. Comparison with previous published results suggest that the serial code is correct. The first attempt at a parallel code was the synchronous code for two processors in which the boxes are still coupled to each other. The next code developed is the asynchronous two node code, which involves the decoupling of the two boxes and can be extended to a larger number of processors. The random selection procedure is a multi-processor algorithm which involves the random selection of the boxes for transfer moves. This procedure involves the transfer of messages from a busy node to the free node, introducing a bias in the selection procedure. Results at $T^*=1.0$ confirm this. However, it has not been possible to quantify the bias. The subsequent parallel procedures tested are the perfectly parallel and the multi-node procedure with swapping of nodes. These procedures were developed to preclude the transfer of messages and to avoid the excessive message passing involved in the random-selection procedure. The perfectly parallel procedure involves coupling of two processors for the entire length of the simulation and restricting the volume and particle transfers to between themselves. The simulation is thus divided into sub-simulations, which run in a perfectly parallel manner. Results indicate that some of the sub-simulations show a slower rate of convergence (to equilibrium values) and might have been caught in the metastable region. It is also

possible that the difference might be a random occurrence. A possible perturbation which could be applied to the sub-simulations which show slow convergence is the swapping of the processors. It is seen that for lower temperatures, the swapping of the nodes helps in a uniform convergence of all the sub-simulations. An increase in the frequency of the swapping decreases the divergence of the mean of processor averages. This suggests that at lower temperature, the difference in the convergence rates of the sub-simulations is indeed due to the metastability of the sub-systems. At higher temperatures, it is observed that the convergence rates are unaffected by the swapping. This may be due to larger fluctuations at the higher temperatures, due to which the systems may not get trapped in the metastable regions. The difference between convergence rates for the different swap frequencies do not show a consistent trend, suggesting that the differences may be caused by random fluctuations in the properties.

An attempt has also been made to compare the statistical accuracy of the various algorithms. The parallel algorithms (perfectly parallel or the swapping procedure) generate larger number of configurations in the same wall clock time. Thus these procedures are able to sample configuration space more efficiently, increasing the statistical accuracy of the results. For the perfectly parallel procedures, it is important that the equilibration time of the simulation be small compared to the production time. This is because all the nodes have the same system and have to equilibrate for the same amount of time before the production run. At lower temperatures, the rate of convergence is usually slower due to the smaller fluctuations and the lower probability of successful particle exchanges. Swapping of nodes improves the convergence rate at lower temperatures and may be a useful method for increasing the efficiency of the perfectly parallel code. At higher temperatures, swapping of the processors is not very helpful as the fluctuations are enough to bring the systems out of the metastable regions. The

the fluctuations are enough to bring the systems out of the metastable regions. The swapping of processors is thus potentially useful for the systems with inherently slow convergence rates.

An attempt was made to calculate the isothermal compressibility from simulations in the Gibbs ensemble. The expression for the isothermal compressibility has not been reported for the Gibbs ensemble and was derived starting from basic principles. The inverse of isothermal compressibility is different for the two phases at lower temperature as expected. As the temperature is increased, the inverse value decreases and the values from the two phases become almost equal at temperatures close to the critical point. The results thus show the correct trends. Also the numerical values are comparable with those obtained from the L-J equation of state. Though the simulation results seem to be very promising, they show a lot of scatter and we cannot be absolutely certain about their correctness. These results are presented in Appendix B.

For the phase calculations using the BFW potential, the results obtained were not consistent with literature values. These results have thus been presented in Appendix C.

Recommendations for future work.

This work was another attempt to arrive at an efficient parallel Monte Carlo algorithm. The main emphasis was on the parallelization of the Gibbs Ensemble simulation technique and involved the testing of the distributed techniques involving many simulation boxes.

The swapping of the nodes shows promise at lower temperatures, where the

convergence rates are lower and could be implemented for simulations where the equilibration rate is affected by fluctuations in the system.

The parallel techniques could be implemented using the serial code as a two-box unit instead of the asynchronous code as this would avoid the message passing between the two boxes during the simulation.

The perfectly parallel technique and the swapping procedures can be implemented using the parallel virtual machines (PVM) message passing interface, and can be run on a cluster of workstations. In this case also, the serial code could be used as the two-box simulation unit.

References

References

- 1) M. Allen and D. Tildesley, *Computer Simulations of liquids* (Oxford University Press, Oxford, 1987).
- 2) M. R. S. Pinches D. J. Tildesley and W. Smith, "Large scale molecular dynamics on parallel computers using the link-cell algorithm.", *Molecular Simulation*, **6**, 51, (1991).
- 3) G. S. Heffelfinger and M. E. Lewitt, "A comparison between two massively parallel algorithms for Monte Carlo computer simulation: An investigation in the Grand Canonical ensemble.", *J. Computational Chemistry*, **17** (2), 250, (1996).
- 4) M. Greger and J. J. de Pablo, " The use of time warp protocol in extremely large scale parallel polymer simulation". *Preprint*.
- 5) K. Esselink, L. D. J. C. Loyens and B. Smit, "Parallel Monte Carlo simulations.", *Physical Review E*, **51** (2), 1560, (1995).
- 6) K. E. Gubbins, K. S. Shing and W. B. Streett , "Fluid phase equilibria: Experiment, Computer Simulation and Theory", *J. Phys. Chem.*, **87**, 4573, (1983).
- 7) B. Widom, "Some topics in the theory of fluids", *J. Phys. Chem.* , **39**, 2802, (1963)
- 8) D. J. Adams, "Calculating the low-temperature vapor line by Monte Carlo.", *Molec. Phys.*, **32** (3), 647, (1976).
- 9) D. J. Adams, "Calculating the high-temperature vapor line by Monte Carlo.", *Molec. Phys.*, **37** (1), 211, (1979).
- 10) G. A. Chapela, S. E. Martinez-Casas, and C. Varea, "Square well orthobaric densities via spinodal decomposition.", *J. Chem. Phys.*, **86**(10), 5683, (1987).

- 11) A. Z. Panagiotopoulos, N. Quirke, M. R. Stapelton and D. J. Tildesely, "Phase equilibria by simulation in the Gibbs ensemble.", *Molec. Phys.*, **63**(4), 527, (1988).
- 12) A. Z. Panagiotopoulos, "Direct determination of phase coexistence properties of fluids by Monte Carlo simulations in a new ensemble", *Molec. Phys.*, **61**(4), 813, (1987).
- 13) W. J. Gibbs, Transactions of the Connecticut Academy, Vol III, Reprinted 1961, The Scientific Papers of J. W. Gibbs (Dover), p. 65.
- 14) A. M. Mood, F. A. Graybill and D. C. Boes, "Introduction to the theory of statistics.", *McGraw-Hill, Inc.*, (1963).
- 15) K. Kiyohara, T. Spyriouni, K. E. Gubbins and A. Z. Panagiotopoulos, "A new method for determination of phase coexistence properties of fluids.", *In Print* (1996).
- 16) J. K. Johnson and K. E. Gubbins, "Phase equilibria for associating Lennard-Jones fluids from theory and simulation.", *Molec. Phys.*, **77**, 1033, (1992).
- 17) S. H. Lee and P. T. Cummings, "Shear viscosity of model mixtures by nonequilibrium molecular dynamics. 1. Argon-krypton mixtures.", *J. Chem. Phys.*, **99**(5), 3919, (1993).
- 18) N. B. Vargaftik, "Handbook of physical properties of liquids and gases.", *Hemisphere Publishing Corporation*, (1975).
- 19) R. H. Perry and D. W. Green, Eds., "Perry's Chemical Engineers Handbook, 6th Ed.", *McGraw-Hill Inc.*, (1994).
- 20) J. A. Barker, R. O. Watts, J. K. Lee, T. P. Schafer and Y. T. Lee, "Interatomic potentials for krypton and xenon.", *J. Chem. Phys.*, **61** (8), 3081, (1974).
- 21) B. Smit, 1990, "Simulation of phase coexistence: from atoms to surfactants", Ph.D dissertation, Univ. of Utrecht, The Netherlands.

- 22) B. Smit and D. Frenkel, " Calculation of the chemical potential in the Gibbs Ensemble", *Molec. Phys.*, **68**(4), 951, (1989).
- 23) M. R. Stapelton and A. Z. Panagiotopolous, " Application of excluded volume map sampling to phase equilibrium calculations in the Gibbs ensemble.", *J. Chem. Phys.*, **92**(2), 1285, (1990).
- 24) A. Z. Panagiotopolous and M. R. Stapelton, " The Gibbs Method for Molecular based computer simulations of phase equilibria, *Fluid Phase Equilibria*, **53**, 133, (1988).
- 25) E. N. Rudisill and P. T. Cummings, "Gibbs ensemble simulation of phase equilibrium in the hard core two-Yukawa fluid model for the Lennard-Jones fluid." *Molec. Phys.*, **68**, 629, (1989).
- 26) D. N. Theodorou and U. W. Suter, "Model Systems with periodic boundaries.", *J. Chem. Phys.*, **82** (2), 955, (1985).
- 27) R. L. Rowley, "Statistical mechanics for thermophysical property calculations.", *Prentice Hall*, New Jersey, (1994).
- 28) G. H. Golub and J. M. Ortega, "Scientific computing and differential equations.", *Academic Press, Inc.*, San Diego, (1992).
- 29) P. S. Y. Cheung, "On the calculation of specific heats, thermal pressure coefficients, and compressibilities in molecular dynamics simulations.", *Molec. Phys.*, **33**, 519, (1977).
- 30) A. H. Widmann and U. W. Suter, "Parallelization of a Monte Carlo algorithm for the simulation of polymer melts", *Comp. Phys. Comm.*, **92** , 229 (1995).

Appendices

Appendix A: Derivation of the expression for $\partial\langle f \rangle/\partial V$ for the Gibbs Ensemble

From the definition for an average in the Gibbs Ensemble,

$$\langle f \rangle = \frac{1}{Q_{Gibbs}} \sum_{N_1=0}^N \int_0^V dV_1 V_1^{N_1} V_2^{N_2} f Q_1 Q_2 \quad (1)$$

Taking the derivative with respect to V

$$\begin{aligned} \frac{\partial\langle f \rangle}{\partial V} &= \frac{\partial}{\partial V} \left\{ \frac{1}{Q_{Gibbs}} \sum_{N_1=0}^N \int_0^V dV_1 V_1^{N_1} V_2^{N_2} f Q_1 Q_2 \right\} \\ &= \frac{\partial}{\partial V} \left(\frac{1}{Q_{Gibbs}} \right) \sum_{N_1=0}^N \int_0^V dV_1 V_1^{N_1} V_2^{N_2} f Q_1 Q_2 + \frac{\partial}{\partial V} \left(\sum_{N_1=0}^N \int_0^V dV_1 V_1^{N_1} V_2^{N_2} f Q_1 Q_2 \right) \frac{1}{Q_{Gibbs}} \\ &= -\frac{1}{Q_{Gibbs}^2} \frac{\partial Q_{Gibbs}}{\partial V} \sum_{N_1=0}^N \int_0^V dV_1 V_1^{N_1} V_2^{N_2} f Q_1 Q_2 + \frac{\partial}{\partial V} \left(\sum_{N_1=0}^N \int_0^V dV_1 V_1^{N_1} V_2^{N_2} f Q_1 Q_2 \right) \frac{1}{Q_{Gibbs}} \end{aligned} \quad (2)$$

Now,

$$\begin{aligned} \frac{1}{Q_{Gibbs}} \frac{\partial Q_{Gibbs}}{\partial V} &= \frac{1}{Q_{Gibbs}} \sum_{N_1=0}^N \int_0^V dV_1 V_1^{N_1} V_2^{N_2} Q_1 Q_2 \left\{ \rho_1 \frac{\partial V_1}{\partial V} + \rho_2 \left(1 - \frac{\partial V_1}{\partial V} \right) \right\} \\ &\quad + \frac{\partial \ln Q_1}{\partial V_1} \left(\frac{\partial V_1}{\partial V} \right) + \frac{\partial \ln Q_2}{\partial V_2} \left(\frac{\partial V_2}{\partial V} \right) \end{aligned}$$

Thus

$$\begin{aligned} \frac{1}{Q_{Gibbs}} \frac{\partial Q_{Gibbs}}{\partial V} &= \frac{1}{Q_{Gibbs}} \sum_{N_1=0}^N \int_0^V dV_1 V_1^{N_1} V_2^{N_2} Q_1 Q_2 \left\{ \frac{\partial V_1}{\partial V} \left(\rho_1 - \beta \frac{\partial U_1}{\partial V_1} \right) + \frac{\partial V_2}{\partial V} \left(\rho_2 - \beta \frac{\partial U_2}{\partial V_2} \right) \right\} \\ &= \beta \left\langle \frac{\mathcal{P}_1 + \mathcal{P}_2}{2} \right\rangle \end{aligned} \quad (3)$$

where the $\mathcal{P}$ is defined as the instantaneous pressure function (Equation 3.5).

The expression for $\partial\langle f \rangle / \partial V$ thus becomes

$$\begin{aligned}
 \frac{\partial\langle f \rangle}{\partial V} &= - \left(\frac{1}{\mathcal{Q}_{Gibbs}} \frac{\partial \mathcal{Q}_{Gibbs}}{\partial V} \right) \left(\frac{1}{\mathcal{Q}_{Gibbs}} \sum_{N_1=0}^N \int_0^V (dV_1 V_1^{N_1} V_2^{N_2} f \mathcal{Q}_1 \mathcal{Q}_2) \right) + \\
 &\quad \frac{1}{\mathcal{Q}_{Gibbs}} \sum_{N_1=0}^N \int_0^V dV_1 V_1^{N_1} V_2^{N_2} \left(\frac{\partial f}{\partial V} \right) \mathcal{Q}_1 \mathcal{Q}_2 + \\
 &\quad \frac{1}{\mathcal{Q}_{Gibbs}} \sum_{N_1=0}^N \int_0^V dV_1 V_1^{N_1} V_2^{N_2} \mathcal{Q}_1 \mathcal{Q}_2 \left[\frac{\partial V_1}{\partial V} \left(\rho_1 - \beta \frac{\partial U_1}{\partial V_1} \right) + \frac{\partial V_2}{\partial V} \left(\rho_2 - \beta \frac{\partial U_2}{\partial V_2} \right) \right] \\
 &= -\beta \langle f \rangle \left\langle \frac{\mathcal{P}_1 + \mathcal{P}_2}{2} \right\rangle + \left\langle \frac{\partial f}{\partial V} \right\rangle + \beta \left\langle f \left(\frac{\mathcal{P}_1 + \mathcal{P}_2}{2} \right) \right\rangle
 \end{aligned}$$

Appendix B: Study of Fluctuations (Calculation of the Isothermal Compressibility in the Gibbs Ensemble)

The motivation behind the calculation of the isothermal compressibility was the study of fluctuations in the distributed simulations (as defined in chapter 3). The results obtained from the simulations are encouraging and show the right trends as compared to those from the Lennard-Jones equation of state (eos). However they show a lot of scatter and their correctness cannot be ascertained. The simulation results are thus presented and compared with the eos results in this appendix. Table B.1 presents the results obtained from the simulations and the equation of state. These results have been plotted against the reduced densities in figure B.1, B.2 and B.3. The plot in figure B.2 shows the eos results in the form of a line to clarify the comparison. The points in the higher density region represent the liquid phases and those in the lower density region are the vapor phases.

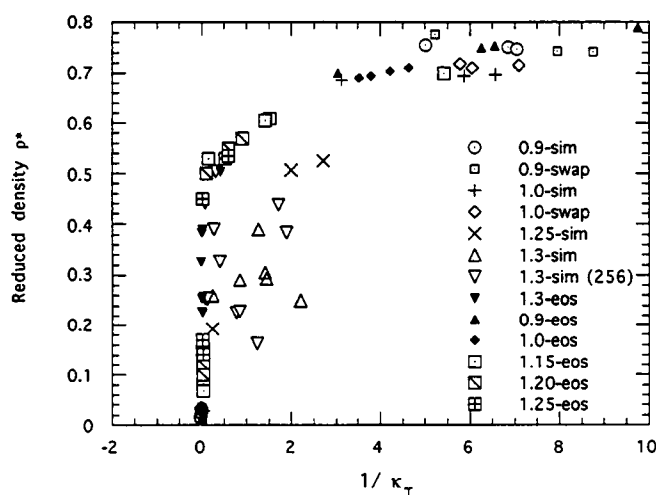


Figure B.1 Plot of $1/\kappa_T$ for simulation results and for the Lennard-Jones equation of state.

Table B.1 Results for isothermal compressibility calculations. The densities and isothermal compressibility for each processor have been reported*.

T*	source	phase	ρ^*	$1/K_T$	source	phase	ρ^*	$1/K_T$	source	phase	ρ^*	$1/K_T$
0.9	sim PP	liq	0.752	6.86	sim swap	liq	0.777	5.21	eos	liq	0.754	6.56
		liq	0.7479	7.092		liq	0.744	7.98		liq	0.750	6.254
		liq	0.7549	5.017		liq	0.741	8.75		liq	0.70	3.035
		vap	0.0197	-0.0246		vap	0.019	0.0164		liq	0.790	9.750
		vap	0.022	-0.01002		vap	0.035	0.0001		vap	0.015	0.011
		vap	0.0141	-0.0195		vap	0.023	-0.02		vap	0.013	0.009
										vap	0.016	0.011
1.0	sim PP	liq	0.6970	6.568	sim swap	liq	0.718	5.78	eos	liq	0.703	4.23
		liq	0.6869	3.1179		liq	0.710	6.06		liq	0.695	3.788
		liq	0.694	5.867		liq	0.716	7.08		liq	0.710	4.64
		vap	0.0290	0.0379		vap	0.03	0.0081		liq	0.690	3.522
		vap	0.0332	-0.0294		vap	0.0268	0.006		vap	0.0309	0.021
		vap	0.033	-0.00957		vap	0.036	-0.015		vap	0.0350	0.022
										vap	0.0270	0.019
1.15	sim PP	liq	0.6016	8.034	eos	liq	0.609	1.524				
		liq	0.6052	9.08		liq	0.605	1.416				
		liq	0.6067	10.27		liq	0.53	0.149				
		vap	0.072	0.029		liq	0.7	5.439				
		vap	0.077	0.065		vap	0.072	0.0376				
		vap	0.077	0.004		vap	0.083	0.0363				
						vap	0.069	0.0377				
1.25	sim PP	liq	0.525	2.706	eos	liq	0.529	0.5292				
		liq	0.507	1.993		liq	0.535	0.6013				
		vap	0.146	0.049		liq	0.45	0.024				
		vap	0.1919	0.243		vap	0.152	0.0282				
						vap	0.142	0.0332				
						vap	0.17	0.01844				
1.3	sim PP (108)	liq	0.39	1.27	sim PP (256)	liq	0.503	0.317	eos	liq	0.503	0.415
		liq	0.29	0.85		liq	0.3906	0.27		liq	0.3906	0.014
		liq	0.305	1.42		liq	0.383	1.89		liq	0.383	0.006
		vap	0.25	2.21		vap	0.226	0.774		vap	0.226	0.010
		vap	0.26	0.24		vap	0.255	0.12		vap	0.255	0.002
		vap	0.2925	1.45		vap	0.327	0.402		vap	0.327	-0.010

* PP stands for perfectly parallel. swap stands for swapping procedure. eos represents equation of state results.

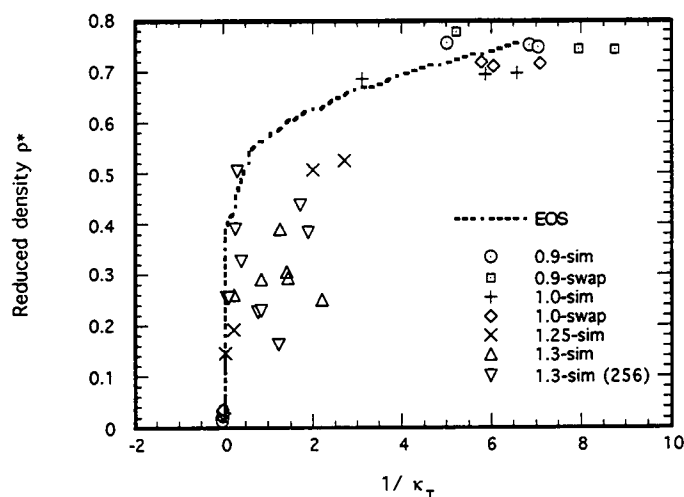


Figure B.2 Plot of $1/\kappa_T$ for simulation results and for the Lennard-Jones equation of state

It is seen that κ_T values from simulations at a lower temperature are quite different for the two phases. The κ_T values for liquid phases are very small at the low temperatures (inverse of κ_T is large). As the temperature is increased, the κ_T values for the liquid phase increase, with a corresponding decrease in the value of $1/\kappa_T$. On the other hand, the vapor values for κ_T are very large at lower temperatures (thus $1/\kappa_T$ values are very small). The vapor values show a small change with the temperature. The results from the eos show a similar trend as that of the simulation results. Also the values of $1/\kappa_T$ are numerically comparable for the lower temperatures. At the higher temperatures there is discrepancy between the values as seen in Figure B.2. This may be attributed to a small system size (108 molecules) in each box. At higher temperatures the fluctuations in the density may be very large and the system size effect may affect the results to a larger extent. An attempt was made to verify this by conducting a simulation at $T^*=1.3$, starting with 256 molecules in each box. The results from this simulation

($T^*=1.3$, 256 molecules) do show some improvement, though the scatter in the results is too large to make any definitive conclusions.

The results obtained from the swapping procedures (at $T^*=0.9, 1.0$) have been compared with the results from the perfectly parallel procedures. The results do not show any appreciable difference. However, the values of the isothermal compressibilities from each phase show a lot of scatter and it is not possible to conclude anything from the comparison of results from the two procedures. Figure B.3 shows the comparison of results from the simulations and the eos for the vapor phases.

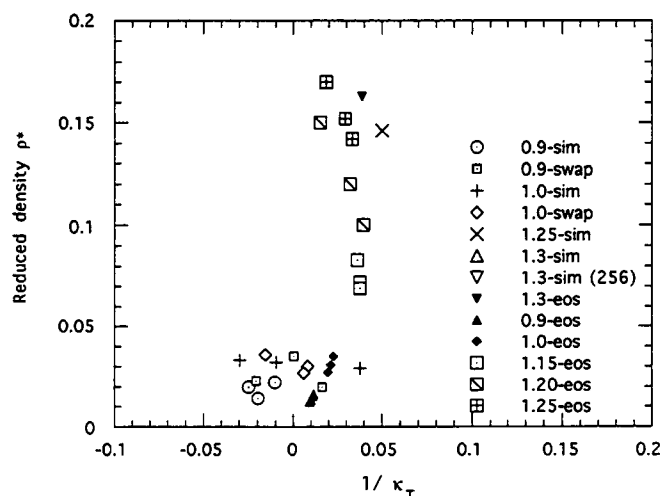


Figure B.3 Plot of $1/\kappa_T$ for simulation results and for the Lennard-Jones equation of state, showing only the vapor phases.

It is seen that the simulation results at $T^*=0.9$ for the vapor phase show a negative κ_T value, suggesting that the vapor phases are in the unstable region. The vapor phases are not equilibrated, thus exhibiting negative κ_T values and densities which are higher than the equilibrium values.

Appendix C: Calculation of the Phase Equilibrium Properties for the BFW Potential

The simulations for the phase equilibrium calculations of the BFW potential were conducted starting with 108 molecules in each phase. A cutoff value of $2.5R_m$ was applied where R_m is the position of the potential minimum. The serial code was used for the testing of the codes. The results obtained have been presented in Table C.1 and graphically compared with literature values (experimental) [18] in Figure C.1. The results yield an expanded phase diagram with the liquid phase densities greater and the vapor phase densities less than the corresponding reported values. The pressures in the two phases are also not consistent, suggesting an error in the calculations. Since the results are not reliable, they have thus been presented in this appendix.

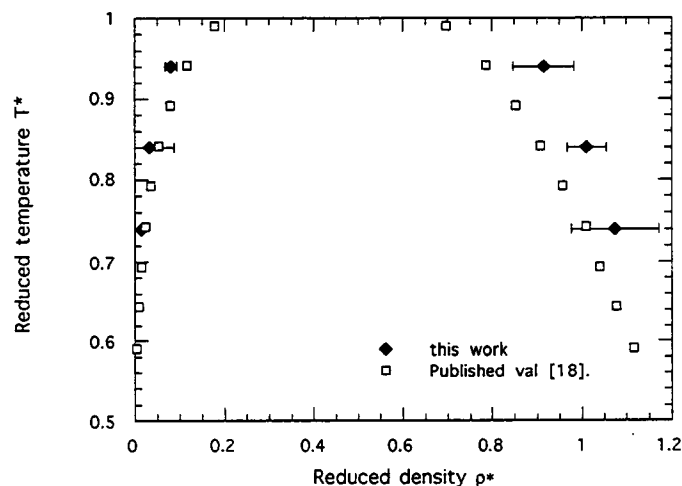


Figure C.1 Phase coexistence curve for Kr. Comparison of the simulation results with reported values [18].

Table C.1 Simulation results for the BFW potential.

Source	T*	ρ^*	P*	E*	State
This work	0.6438	0.0039(7)	0.0025(5)	-0.027(13)	vapor
Literature[18]	0.6438	0.0079			vapor
This work	0.6438	1.1613	0.692(45)	-5.26(17)	liquid
Literature[18]	0.6438	1.077			liquid
This work	0.7429	0.014(4)	0.011(3)	-0.093(37)	vapor
Literature[18]	0.7429	0.023			vapor
This work	0.7429	1.07(9)	0.90(6)	-4.84(31)	liquid
Literature[18]	0.7429	1.0009			liquid
This work	0.8420	0.032(5)	0.030(6)	-0.20(5)	vapor
Literature[18]	0.8420	0.0537			vapor
This work	0.8420	1.01(4)	1.07(4)	-4.42(20)	liquid
Literature[18]	0.8420	0.908			liquid
This work	0.9421	0.080(12)	0.127(28)	-0.47(8)	vapor
Literature[18]	0.9421	0.1155			vapor
This work	0.9421	0.915(67)	2.21(19)	-3.94(26)	liquid
Literature[18]	0.9421	0.786			liquid

VITA

Harpreet Singh Chopra was born on the 17th of May 1973 in Bombay, India. He did his high schooling from Parle College and a B.S. in Chemical Engineering from the University of Bombay, Department of Chemical Technology (UDCT). He came to the University of Tennessee, Knoxville in the fall of 1994 for pursuing higher studies in Chemical Engineering and obtained an M.S. in December 1996.