
Doctoral Dissertations

Student Theses and Dissertations

Fall 2012

A Monte Carlo study of the scaling of nucleation rates in a Lennard-Jones system

Mark Allan Thomason

Follow this and additional works at: https://scholarsmine.mst.edu/doctoral_dissertations



Part of the [Physics Commons](#)

Department: Physics

Recommended Citation

Thomason, Mark Allan, "A Monte Carlo study of the scaling of nucleation rates in a Lennard-Jones system" (2012). *Doctoral Dissertations*. 2025.

https://scholarsmine.mst.edu/doctoral_dissertations/2025

This thesis is brought to you by Scholars' Mine, a service of the Missouri S&T Library and Learning Resources. This work is protected by U. S. Copyright Law. Unauthorized use including reproduction for redistribution requires the permission of the copyright holder. For more information, please contact scholarsmine@mst.edu.

A MONTE CARLO STUDY OF THE SCALING OF NUCLEATION RATES IN
A LENNARD-JONES SYSTEM

by

MARK ALLAN THOMASON

A DISSERTATION

Presented to the Faculty of the Graduate School of the
MISSOURI UNIVERSITY OF SCIENCE & TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

PHYSICS

2012

Approved by

Barbara N. Hale, Advisor
Gerald Wilemski
Paul Parris
Jerry Peacher
Jee-Ching Wang

© 2012

MARK ALLAN THOMASON

All Rights Reserved

ABSTRACT

Temperature scaling of the homogenous vapor-to-liquid nucleation rate, J , is examined in a model Lennard-Jones (LJ) system. The model uses the Bennett Metropolis Monte Carlo technique to determine small cluster growth/decay rate constant ratios, β_{n-1} / α_n , at four temperatures ($T = 40, 50, 60,$ and 83.6K) below the argon Lennard-Jones critical temperature, T_c . The β_{n-1} / α_n for clusters ranging in size from $n = 2$ to $n = 192$ LJ particles are applied to a kinetic steady-state nucleation rate formalism and nucleation rates are determined at the same four temperatures. When these rates are plotted first in the standard way vs. $\ln S$, (where S is the ratio of ambient to coexistence vapor pressure) and then vs. the *scaled supersaturation*, $\ln S / [T/T-I]^{3/2}$, the values of $\log J$ are found to collapse onto a single line. This demonstrates that the nucleation rate is a function of $\ln S / [T/T-I]^{3/2}$ – rather than of the independent variables, S and T . A similar scaling has been observed in the experimental nucleation rate data of water and toluene. The present study is the first simulation based demonstration of vapor to liquid nucleation rate temperature scaling in a model dilute vapor system and provides insight into the “law of mass action” model assumptions which give rise to the scaling.

ACKNOWLEDGMENTS

I would like to thank Dr. Barbara N. Hale, my advisor, for her patience and for giving me this opportunity to do research. Without her, this would have never happened, and for that, I will always be grateful. I would like to thank the other members of my committee for their time and their encouragement at different stages of my growth in Physics. I would also like to thank Jared Gavin for sharing the road with me, through the good times and the bad, we made it. I would also like to send a heartfelt thanks to Jason Alexander and his family, for their love and support, and for Jason who not only listened, but also shared with me his un-biased opinion and knowledge. I would also like to thank my wife, Mary King Thomason, and my daughter, Samantha Xiao Ying Thomason for their unconditional love and support especially at the very end of this whole affair. I also would like to dedicate this work to my parents, Leonard and Peggy Thomason, may they rest in peace; I hope that I made you both proud. I miss and love you both.

TABLE OF CONTENTS

	Page
ABSTRACT	iii
ACKNOWLEDGMENTS	iv
LIST OF FIGURES	vii
LIST OF TABLES	viii
SECTION	
1. INTRODUCTION	1
1.1. OVERVIEW AND MOTIVATION	1
1.2. NUCLEATION AS A PRECURSOR TO FIRST ORDER PHASE TRANSITIONS	1
1.3. NUCLEATION EXPERIMENTS	4
1.3.1. History and General Survey of Data	4
1.3.2. Experimental Argon Nucleation Rates	5
1.4. THEORETICAL PREDICTION OF THE NUCLEATION RATE	6
1.5. THE SCALED NUCLEATION RATE MODEL	9
1.6. COMPUTER SIMULATIONS OF THE NUCLEATION RATE	11
1.7. RESULTS OF PREVIOUS LENNARD-JONES POTENTIAL SIMULATIONS	16
1.8. SECTIONS DESCRIBING THE PRESENT WORK	17
2. BENNETT METROPOLIS MONTE CARLO SIMULATIONS OF SMALL LJ CLUSTERS	19
2.1. MODEL SYSTEM AND STATISTICAL MECHANICAL FORMALISM	19
2.2. THE BENNETT METROPOLIS MONTE CARLO METHOD	22
2.3. THE KINETIC NUCLEATION RATE FORMALISM	23
3. THE MONTE CARLO SIMULATION RESULTS	26
3.1. COMPUTATIONAL DETAILS	26

3.2. METROPOLIS MONTE CARLO METHOD	27
3.3. THE FREE ENERGY DIFFERENCES	28
3.4. THE NUCLEATION RATES	30
4. CONCLUSIONS	44
APPENDICES	
A. STEADY STATE NUCLEATION RATE.....	45
B. DERIVATION OF THE BENNETT TECHNIQUE WITH FERMI FUNCTION WEIGHTING.....	48
C. PHYSICAL REVIEW LETTER	52
D. THE FORTRAN CODE.....	57
BIBLIOGRAPHY.....	150
VITA.....	155

LIST OF FIGURES

Figure	Page
1.1. Homogenous nucleation rate data for toluene	10
3.1. Schematic view of the Metropolis Monte Carlo process	29
3.2. Plot of free energy differences vs. $n^{-1/3}$, at $T=40, 50, 60, 83.6\text{K}$	35
3.3. Plot of $\frac{\delta\Delta F_n}{\left(\frac{T_c}{T}-1\right)}$ vs. $n^{-1/3}$, at $T=40, 50, 60, 83.6\text{K}$	36
3.4. Plot of $\log(J / s^{-1}cm^{-3})$ vs. $\ln S$ at $T^*=40, 50, 60, 83.6\text{K}$	38
3.5. Plot of $\log(J / s^{-1}cm^{-3})$ vs. $\frac{C_0 \ln S}{\left[\frac{T_c}{T}-1\right]^{3/2}}$ at $T^*=40, 50, 60, 83.6\text{K}$	39
3.6. Plot of $-\ln(P / P_c)$ vs. $\frac{T_c}{T}-1$	41
3.7. A snapshot of the B ensemble for an $n=120$ configuration at $T=83.6\text{K}$ after 10M Monte Carlo steps, superimposing each configurations at every 1M steps.....	42
3.8. A snapshot of the B ensemble for an $n=150$ configuration at $T=83.6\text{K}$ after 20M Monte Carlo steps, superimposing each configurations at every 1M steps.....	42
3.9. A snapshot of the B ensemble for an $n = 27$ configuration at $T = 40\text{K}$ after 10M Monte Carlo steps	43
3.10. Snapshot of the B ensemble for an $n = 192$ configuration at $T = 60\text{K}$ after 70M Monte Carlo steps	43

LIST OF TABLES

Table	Page
1.1. Summary of computer simulations results using the LJ potential	17
3.1. Values of configurational free energy differences and number of M_{step} to calibrate each cluster for $n=2-15$ at $T=40, 50, 60, 83.6K$	31
3.2. Values of configurational free energy differences and number of M_{step} to calibrate each cluster for $n=16-30$ at $T=40, 50, 60, 83.6K$	32
3.3. Values of configurational free energy differences and number of M_{step} to calibrate each cluster for $n=32-90$ at $T=40, 50, 60, 83.6K$	33
3.4 Values of configurational free energy differences and number of M_{step} to calibrate each cluster for $n=100-192$ at $T=40, 50, 60, 83.6K$	34

1. INTRODUCTION

1.1. OVERVIEW AND MOTIVATION

The motivation for this work has been to study temperature scaling [1-3] of the homogeneous vapor-to-liquid nucleation rates in a model Lennard-Jones system. The approach uses Bennett [4] Metropolis [5] Monte Carlo computer simulations to determine growth/decay rate constant ratios for small vapor clusters and the kinetic steady state nucleation rate formalism [6] to predict the rates. The scaling of the nucleation rate (essentially a law of corresponding states formulation [7]) provides physical insight into the temperature dependence of the nucleation process and a convenient tool for analyzing nucleation rate data in cases where detailed information about the bulk substance is not available. In this case the scaling is not near the critical point, but rather applies to temperatures far below the critical temperature, T_c , where most of the nucleation data exist [1].

1.2. NUCLEATION AS A PRECURSOR TO FIRST ORDER PHASE TRANSITIONS

Nucleation is the process by which embryos of the new phase (the daughter phase) are formed from the meta-stable or supersaturated parent phase [8, 9]. The rate of formation of these embryos (per unit volume) is the nucleation rate, J . In a vapor at coexistence with the bulk liquid an equilibrium distribution of cluster sizes exists, and the probability of finding an n -atom cluster falls off exponentially with the cluster free

energy of formation. By supersaturating the vapor one reduces the n-cluster free energy of formation, and it becomes possible for one cluster (the so-called "critical cluster") to have equal probabilities of growth and decay. Clusters larger than the critical size grow with decreasing system free energy and become macroscopic in size. For very small supersaturations the critical cluster size is large and the nucleation rate negligible. For large supersaturations the critical cluster size is small and the nucleation rate becomes appreciable.

The phenomenon of nucleation has relevance to many every day processes: formation of water droplets, ice and aerosol particles in the lower atmosphere; formation of solid ternary particles in the stratosphere which act as catalysts for ozone depletion; refinement and production of petroleum products; removal of particulates in smokestacks; freezing of water in biological systems; formation of dust particles in intergalactic space; icing of airplane wings and runways; and a host of industrial processes. In the global climate models, the rate at which ice and water are formed is a crucial input to treatment of cloud formation and radiative forcing calculations [10]. Present day climate models include also the homogeneous nucleation of ice from the vapor in the upper atmosphere.

While the physics of nucleation is understandable at a qualitative level, it is a non-equilibrium process and no fundamental analytical theory exists which describes the "rate" of a first order phase transition. Predictions of the nucleation rate have generally depended on models which estimate the decay of a metastable state (see for example Ref [9]) or which model the system kinetics in the supersaturated system (see for example the molecular dynamics simulations of Refs. [11] and [12]). In most theoretical predictions it

is assumed that the formation of embryos is in a steady state. In the simplest model (classical nucleation theory or CNT) the concentration of critical sized clusters is determined from bulk liquid properties [the bulk liquid surface tension is used to determine the free energy of formation] and the rate is determined by multiplying the critical cluster concentration by the flux of monomers to the cluster. A small correction factor accounts for contributions from clusters near the critical size [13]. In the *kinetic* steady state nucleation rate formalism [6, 14, and 15] rate equations are assumed for each cluster size concentration and it is assumed that the growth/decay rate constant ratios are given by detailed balance at coexistence. These models simplify the approximations while leaving a great deal of work to do in calculating the growth/decay rate constants at equilibrium and the metastable cluster free energies of formation.

In this work, consider only homogenous nucleation in a single substance system containing no impurities. Further details about the homogeneous nucleation rate models are given in Section 2. The real complication in predicting a nucleation rate is best understood in terms of the exponential dependence on the free energy of formation of microscopic clusters which are not well described by bulk properties. This has generated a significant computer simulation effort to calculate small cluster free energies of formation using atom-atom potential interactions. In general it is not necessary to include quantum mechanical effects, but some investigators have considered this complication [16]. Nucleation rates describing the liquid-to-solid phase transition are in general more difficult to model because of diffusion within the parent phase. However, many of the concepts described above are applicable. In the present work, only the vapor-to-liquid phase transition will be considered.

1.3. NUCLEATION EXPERIMENTS

1.3.1. History and General Survey of Data. The earliest experimental techniques used an expansion cloud chamber to study gas to liquid nucleation [17, 18]. These early experiments determined the onset conditions for homogenous water droplet formation in the supersaturated vapor. Diffusion chamber techniques were also used to study homogeneous nucleation of water from the vapor [19, 20] at low nucleation rates close to onset (about $1 \text{ cm}^{-3} \text{ s}^{-1}$). A discussion of these experimental techniques and their shortcomings is found in Mason [21].

In the late 1960's Allen and Kassner [22] developed the "nucleation pulse" technique in which the vapor in an expansion chamber was subjected to a pulse of supersaturation lasting about 1ms. This procedure limited droplet formation, minimized vapor depletion and made possible the decoupling of the nucleation and growth phase. Subsequent experiments using this technique in the 1980s studied water [23], toluene [24], and nonane [25], providing extensive data on the temperature dependence of nucleation rates in the range of $10^3 \text{ cm}^{-3} \text{ s}^{-1}$. Strey and Wagner modified this approach for their fast expansion chambers studying nonane [26]; it was also used to study the n-alcohols [27], water [28], n-butanol [29] and H_2O and D_2O (with rates differing by a factor of 2500) [30], -- all at rates close to $10^7 \text{ cm}^{-3} \text{ s}^{-1}$. The supersonic Laval nozzle [31] (giving rates near $10^{17} \text{ cm}^{-3} \text{ s}^{-1}$) was used to measure homogenous nucleation rates of heavy water (D_2O) and indicated consistency with the lower rates in the fast expansion chamber. Other experiments used the supersonic Laval nozzle [32] to examine the condensation of n-propanol, n-butanol and n-pentanol. A comprehensive comparison of

water nucleation rate data is presented in [33] and indicates that while the magnitude and supersaturation dependence of the water rates are reasonably well predicted by the classical nucleation rate model, the temperature dependence is not.

1.3.2. Experimental Argon Nucleation Rates. In the first cryogenic pulse chamber experimental studies of homogenous nucleation rates for argon [34] nucleation and growth could not be decoupled. Recently, Iland, Wölk, and Strey [35] produced systematic nucleation onset data for argon (about $10^7 \text{ cm}^{-3} \text{ s}^{-1}$) with a temperature range from 42K to 58K and for vapor pressures from 0.3kPa to 10kPa. In cryogenic supersonic nozzle experiments Sinha et al. [36] report estimated argon nucleation rates of $10^{17} \text{ cm}^{-3} \text{ s}^{-1}$ at 34 to 53 K which in a scaling plot are consistent with the data of Iland and Strey. A comparison with the classical model shows a 10^{16} to 10^{26} over-prediction (by CNT) for the Iland and Strey data and a 10^{10} to 10^{13} over-prediction for the Sinha et al data [36]. Models developed by Kaschiev [37] and Kalikmonov [38] report modifications to the classical model which provide significant reduction factors in the CNT rate for argon.

The general state of experimental nucleation rate data at present indicates that the data for water, toluene, nonane and the n- alcohols are reasonably well approximated by the classical nucleation rate predictions, but with unreliable temperature dependence. The experimental data for argon is still a topic of much interest and, in spite of CNT modifications noted above, remains largely inconsistent with Monte Carlo and molecular dynamics simulation predictions, which are closer to the classical model results.

1.4. THEORETICAL PREDICTION OF THE NUCLEATION RATE

Over sixty years ago Volmer and Weber [39, 14], Farkas [15], Becker and Doring [6] and Zeldovich [13] developed a classical steady state nucleation rate formalism which encompassed the basic physical features of the nucleation process [8.9]. The most widely used expression for the classical nucleation rate, J_{CNT} , is that of Becker and Döring [6],

$$J_{CNT} = \left[\frac{P_0}{kT} \right]^2 \left(\frac{2\sigma}{\pi m} \right)^{1/2} \frac{S^2}{\rho_{liq}} \exp \left[\frac{-16\pi}{3} \frac{\left(\frac{\sigma}{kT} \right)^3}{(\rho_{liq} \ln S)^2} \right] \quad (1)$$

where P_0 , σ , S , T , m , ρ_{liq} , and k are the equilibrium vapor pressure, bulk liquid surface tension, supersaturation ratio $\left(\frac{P}{P_0} \right)$, temperature, atomic mass, liquid number density and Boltzmann constant, respectively.

Equation (1) is derived by assuming that the (spherical) liquid embryonic cluster of radius, r , has a free energy of formation (in units of kT) from the vapor given by $\Delta G(n) = 4\pi r^2 \sigma / kT - n \ln S$. The surface tension term represents the excess free energy required to form the surface and the $n \ln S$ term represents the excess free energy available in the supersaturated vapor. By assuming a bulk liquid density for the cluster, $\rho_{liq} = n / (4\pi r^3 / 3)$, one can convert the energy of formation to a simple form, $\Delta G(n) = An^{2/3} - n \ln S$, where $A \equiv (36\pi)^{1/3} \sigma / (\rho_{liq}^{2/3} kT)$. The relative maximum in $\Delta G(n)$ (locating the free energy barrier) occurs at $n^* = [2A / (3 \ln S)]^3$ and gives the classical model critical cluster size, n^* . The free energy of formation of the critical

cluster, $\Delta G(n^*) = (1/2)n^* \ln S = (16\pi/3)(\frac{\sigma}{kT})^3 / (\rho_{liq} \ln S)^2$, appears in the exponent in Eq. (1) and is the essential feature of classical nucleation rate formalism.

The concentration of (metastable) clusters of size n^* is estimated from $[N_{n^*}/V] = [N_1 S/V] \exp[-\Delta G(n^*)]$ with $(P_0/kT)S = (N_1/V)S = \rho_1 S$ in the dilute supersaturated vapor system and the nucleation rate is approximated by multiplying the concentration of n^* -clusters by the flux of monomers, $\rho_1 S (v_{ave}/4) 4\pi r_n^2$. The $v_{av.} = [8kT/\pi m]^{1/2}$.

The classical nucleation rate prediction as found in Eq. (1) is sensitive to the temperature dependence of both the bulk liquid surface tension and the vapor pressure $P_0(T)$ which is used to determine the supersaturation ratio. Since the nucleation rate data is generally taken at low temperatures where experimental data for $\sigma(T)$ and $P_0(T)$ are difficult to determine, extrapolations of assumed formulas for the latter quantities can introduce large errors in J_{CNT} . Over the last 30 years modifications have been made to the classical nucleation rate model in an effort to improve its agreement with the experimental data. The most important of these modifications is imposing the condition that $\Delta G(n) = 0$ for $n = 1$. The latter is accomplished easily by replacing $\Delta G(n)$ with $\Delta G(n-1)$. This also makes the rate expression in Eq. (1) consistent with the law of mass action, which requires that N_n be proportional to S^n rather than S^{n+1} (and J proportional to S^{n+1} rather than S^{n+2}). This modification -- also presented in a model

by Girshick and Chiu [40] -- provides a major correction to the temperature dependence of the classical nucleation rate. A statistical mechanical formulation (which assumes the law of mass action) has no need for this correction.

The classical model was stringently tested in the 1980's by experimental nucleation rate measurements which determined the rate for a range of T and S [23, 41]. By the mid-1990s it was clear the classical model had the wrong temperature dependence [9, 28, 42, 43, 44]. In particular the classical nucleation rate tends to underestimate the rate at low temperatures and overestimate the rate at higher temperatures. The errors for a number of substances range from 1 to 4 orders of magnitude. These observations generated a keen interest in determining the free energy of formation for microscopic clusters and led to an intensive computational effort to calculate free energies of formation for atomic and molecular clusters using effective pair potentials (see Section 1.6). A rather successful empirical modification to the classical model was made by Dillmann and Meier [45], who introduced additional n -dependent terms (including Fisher's $\tau \ln(n)$ term [46]) into the energy of formation. This model significantly improved the predictions for the temperature dependence of the water nucleation rate data. In 1986 a scaled temperature dependence model for the nucleation rate was presented and shown to agree with the data for toluene, nonane and water [1]. This scaled model is discussed in more detail in the next section and it will be the aim of the present work to demonstrate that this scaled temperature dependence follows from simple properties of the dilute vapor system.

1.5. THE SCALED NUCLEATION RATE MODEL

The scaling of the nucleation rate near the critical point was investigated in the late 1970s and 1980s by Fisher [46], Binder and Stauffer [2], Binder [3] and others [47-50]. These scaled rate expressions used critical exponents to expand physical quantities near the critical temperature in powers of $(1 - T/T_c)$. In 1986 Hale [1] proposed a scaled nucleation rate model applicable far below T_c , where most of the nucleation rate data exist. This scaled model for J makes use of the law of corresponding states and a simple linear form for the surface tension $\sigma(T) = \sigma_0' [T_c - T]$. The result for the rate is given by the following expression [51],

$$J_{scaled} = J_{oc} \exp \left[- \frac{16\pi\Omega^3 \left[\frac{T_c}{T} - 1 \right]^3}{3 [\ln S]^2} \right], \quad (2)$$

where Ω is the excess surface entropy per atom (in units of k). That is, Ω is the additional entropy an atom has because it is on the surface – rather than in the bulk. The Ω can be determined from the temperature dependence of the experimental surface tension, and, is about 2 for normal liquids and 1.5 for polar liquids. The prefactor, $J_{oc} = 10^{26} \text{ cm}^{-3} \text{ s}^{-1}$, is to first order independent of S and T and well approximated by one event per *thermal wavelength* cubed per second at the critical temperature, T_c .

As noted in the last section Eq. (2) is consistent with the temperature dependence of the homogeneous nucleation rate data for toluene and nonane [51] and water [52, 53]. Recently, it has been used to analyze experimental rates for n-alcohols [54], n-alkanes

[55], and argon [36]. A plot demonstrating the scaling of the homogeneous nucleation rate data for toluene is shown in Fig. 1.1.

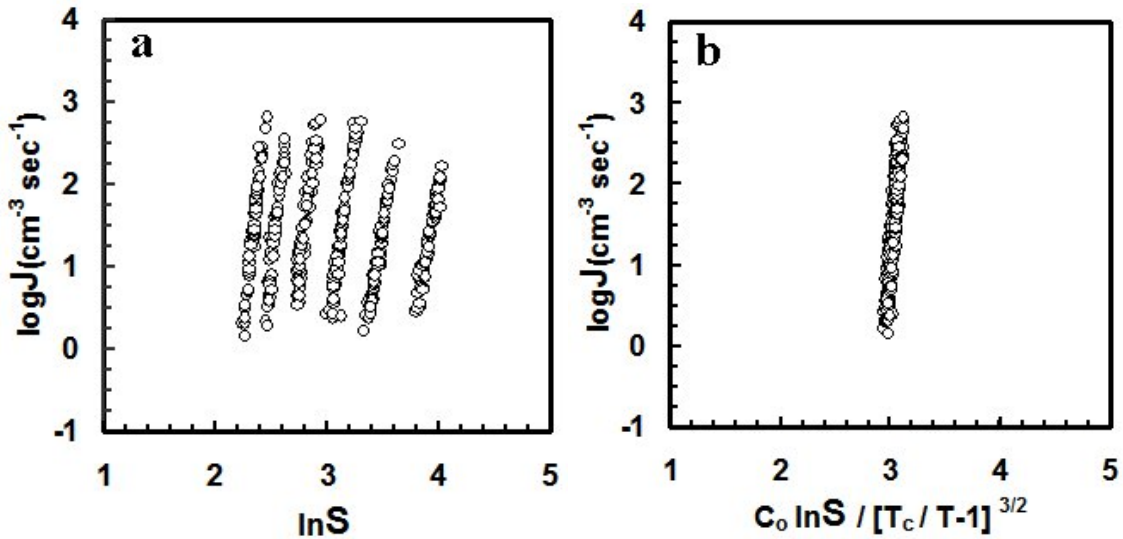


Figure 1.1. Homogenous nucleation rate data for toluene. The homogeneous vapor-to-liquid nucleation rate data of Schmitt et al [24] for toluene at six temperatures plotted vs. a) $\ln S$, and b) $\ln S / [T_c / T - 1]^{3/2}$. The normalization constant, $C_0 = [T_c / 235 - 1]^{3/2}$ and $T_c = 591.8\text{K}$. In part a) the data lines correspond to temperatures 265K, 255K, 245K, 237K, 227K and 216K, left to right.

Besides the temperature scaling of the exponent in Eq. (2) the scaled nucleation rate, J_{scaled} , differs significantly from J_{CNT} in its non-classical pre-factor, J_{oc} . It was shown that the water data of Wölk and Strey [30] fit the temperature dependence of Eq.(2) with $\Omega_{water} = 1.47$ [53]. As indicated by comparing the exponentials in Eqs. (1) and (2), the linear approximation of $\sigma(T)$ and setting $\Omega = \sigma'_0 / (\rho_i^{2/3} k)$ make the

classical and scaled energies of formation formally identical. A scaling law for onset nucleation (with $J = 1 \text{ cm}^{-3} \text{ sec}^{-1}$) can be derived directly from Eq. (2). This scaling law

$$[1], \quad \ln S_{onset} \propto 0.53 \Omega^{3/2} \left[\frac{T_c}{T} - 1 \right]^{3/2},$$

permits simultaneous plots of onset conditions for a wide range of substances and the possibility of estimating onset conditions in the absence of detailed substance bulk properties.

1.6. COMPUTER SIMULATIONS OF THE NUCLEATION RATE

Computer simulations of nucleation rates have primarily focused on three approaches: Monte Carlo (MC), molecular dynamics (MD) and density functional (DF) methods. In MC simulations, particle configurations are sampled in the canonical, Gibbs or grand canonical ensembles. In the molecular dynamics (MD) approach atoms are accelerated via inter-atomic forces along time trajectories. The DF method uses a density functional to calculate the free energy of the system and minimize the free energy, subject to system constraints. These simulations began in the late 1960's and 1970's and were focused on lattice gas systems, argon Lennard-Jones systems, and water systems with a variety of potentials. See references 1-42 of Ref. [53] for an early history of the computer simulation work on microscopic systems of atoms and molecules. A detailed description of these computer methods is given in the book by Allen and Tildesley, *Computer Simulations of Liquids* [56].

In applications of molecular dynamics to the study of nucleation one solves Newton's equations, $m_i \frac{d^2 r_i}{dt^2} = F_i = -\nabla_i \sum_{j \neq i} U(r_j - r_i)$, at small time steps of the order of $10^{-12} s$ to $10^{-15} s$. The first simulation of nucleation in a large periodic system of Lennard-Jones particles was carried out by Yasuoka and Matsumoto [11] in 1998. In this work the system unit cell contained 5000 LJ atoms and 5000 carrier gas atoms interacting with the LJ atoms *via* a LJ potential with no attractive forces. Thermalization of the LJ particles was accomplished *via* collisions with the carrier gas atoms, while the temperature of the carrier gas was controlled with the standard velocity scaling procedure. The system was equilibrated at 149.75K and quenched to 80.3K, after which the system was allowed to develop in time, forming small clusters of LJ argon particles. A steady state nucleation rate of $1.1 \times 10^{29} cm^{-3} s^{-1}$ was reported after about 600 time steps at a supersaturation ratio of 6.8. This result for the rate was about seven orders of magnitude larger than the classical nucleation rate prediction at the same S and T . Yasuoka [12] also simulated the nucleation of water, using a similar technique and TIP4P [57] water-water potentials. Stillinger and Rahman's earlier simulations [58-62] of argon LJ and water by molecular dynamics were not focused on nucleation events but led to the development and testing of the RSL2 water-water potentials which were subsequently applied to atmospheric nucleation simulation studies of water clusters [63] and bulk ice [64, 65].

Monte Carlo simulations of the nucleation rate focus on statistical mechanical calculations of small *n-cluster* free energies of formation from the vapor. Many of these simulated argon via the Lennard-Jones potential. [66-80]]. One of the early Monte Carlo

simulations by Lee, Barker and Abraham [66] determined the free energy difference between n LJ monomers in an ideal gas and a LJ n -cluster constrained to a spherical volume with 0.2 times the density of the bulk liquid at 60K. An integration scheme was used to determine the free energy difference. Garcia [67] used a similar method to determine the free energy differences for a range of temperatures. Some recent simulations of free energies of formation for small clusters include the work of Oh and Zeng [68], Chen et al. [69,70], Kusaka et al. [71], and Lauri et al. [72]. These simulations generally simulated a range of cluster sizes, and determined the critical cluster size and nucleation barrier heights at selected supersaturation ratios at one or two temperatures. A Monte Carlo simulation by ten Wolde and Frenkel [73,74] determined the critical cluster size and nucleation rate for the argon LJ model system at $T = 84K$ and $S = 1.4$. While determined far above the experimental argon nucleation rate temperatures, the MC rate was surprisingly close to that observed experimentally. In the latter simulation a cutoff on the potential was imposed to reduce the computer time. It has been shown that cutoffs on the LJ potential can reduce the surface tension and the critical temperature [82].

In a fixed (N, V, T) canonical ensemble Metropolis Monte Carlo [5] simulation one proceeds as follows. The N atoms in the system volume, V , are initially placed in a random configuration, with potential energy U_i . An atom is chosen at random and subjected to a random displacement, producing a final potential energy, U_j . The potential energy difference, $\Delta U = U_j - U_i$, is used to form a Boltzmann factor,

$\exp\left[\frac{-\Delta U}{kT}\right]$, representing the relative probability of finding the system in the final l state.

In the Metropolis Monte Carlo method [5], the following selection criterion is used. If

$\exp\left[\frac{-\Delta U}{kT}\right] > R$, (where R is a random number satisfying $0 < R < 1$) the move is

accepted; otherwise the move is rejected, the configuration is returned to the initial state

and the process is repeated. After a sufficiently large number, say M , steps, the system

equilibrates to an average potential energy, $\langle U \rangle = \frac{1}{M} \sum_{k=1}^M U_k$. Once the system has

equilibrated, one can calculate a thermodynamic average of the operator, \mathbf{O} , from,

$$\langle O \rangle = \frac{1}{M} \sum_{k=1}^M O_k .$$

The Metropolis Monte Carlo method has been widely used to simulate Lennard-Jones clusters [66-80,]. To calculate free energies in a Metropolis Monte Carlo simulation one must employ techniques which generate a series of free energy differences, with one free energy in the series independently determined. Recent simulations have used a different technique, introduced first by Kusaka et al. [71] In this fixed (μ, V, T) grand canonical ensemble Metropolis method the number of atoms, N , is allowed to vary. Using the grand canonical weighting factor for the accept/reject decision, $\exp[(-\Delta U + \mu\Delta N)/kT]$, and following system equilibration one can count the number of n -clusters, N_n , which appear in the system and generate $\langle N_n / N_1 \rangle$, the

average relative concentration. The expression $\langle N_n / N_1 \rangle = \exp[-\langle \Delta G / kT \rangle]$ is then used to determine the average free energy of formation, $\langle \Delta G_n \rangle = \langle G_n - nG_1 \rangle$.

All of the cluster simulation methods must assume a *cluster definition*, which ranges from n atoms in a predetermined spherical volume, V_n (the cluster definition of Lee, Barker and Abraham [66]), to n "connected" atoms which are all within some distance, r_0 , of at least one other atom in the cluster (the cluster definition of Stillinger [81]). A nucleation rate can be predicted from these MC simulation based free energies of formation using the classical model expression of Eq. (1) and approximations for the monomer flux in the prefactor. Some simulations use the approach developed by Hale and Ward [77] and calculate free energy differences between neighboring sized clusters [52, 63, 72, 77, 78, 79, 80]. One can add up the free energy differences to determine the free energy of formation or use n -cluster growth/decay rate constant ratios from the free energy differences in a kinetic nucleation rate formalism. This is discussed later in Section 3.1.

All results of the Monte Carlo and molecular dynamics simulations depend on the accuracy of the potential model, the definition of the cluster and technical limitations of the simulations method. In particular classical atom-atom interaction potentials often have surface tensions, vapor pressures, and/or critical temperatures which do not agree with experiment. When comparing simulation results to experimental data the most rigorous analysis is achieved by transforming each quantity into a corresponding states representation such as that described by Dunikov [82]. This approach was demonstrated in an evaluation of computer simulation results for water and argon nucleation rates [53].

1.7. RESULTS OF PREVIOUS LENNARD-JONES POTENTIAL SIMULATIONS

The Lennard-Jones potential is given by the following expression where r is separation distance between the two point particles:

$$V_{ij} = 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 \right]. \quad (3)$$

The parameters σ and ϵ reflect the size of the atom and the depth of the attractive well, respectively. The above was developed to model rare gas atom systems, but has been used widely for a range of atoms and molecules because of its less intensive computational requirements. Generally in LJ simulations the lengths are scaled with σ and energies with ϵ . Together with the mass, m , and Boltzmann constant, k , these parameters are used to define time in units of $\tau = \sqrt{m\sigma^2 / \epsilon}$, and pressure in σ^2 / ϵ . The ϵ is often defined in terms of a temperature, $\epsilon = kT_{LJ}$. For the argon LJ system, $T_{LJ} = 119.28K$ and $\sigma = 3.405A$ (in some papers 119.4K and 3.40 A) are used. The latter parameters are usually circumvented by reporting reduced quantities, $T^* = kT / \epsilon$ and $r^* = r/\sigma$. Presently accepted critical point quantities for the full *LJ* potential are $T_c^* = 1.313 \pm 0.003$, and for the critical density, $\rho_c = 0.316 \pm 0.001$ [83]. The negative (positive) terms in Eq.(3) provide attractive (repulsive) interactions between particles. The attractive term is based on dispersion forces and the repulsive term prevents overlap of atoms. Each plays a critical role in generating the thermodynamic properties of the LJ system. Sometimes the powers of the attractive and repulsive parts (6, 12) are altered to (i, j) with the resulting potential function called a LJ i-j type potential. Often when

potential functions are not available for complex molecules the LJ potentials are assumed and the σ and ϵ chosen as the size and average binding energy, respectively. A particular advantage of the LJ potential is that one does not need r , and can avoid the computationally intensive square root operation. For these reasons Lennard-Jones simulations of argon have been the main focus of the majority of simulations. Table 1.1. is a summary of the computer simulation results relevant to nucleation studies for the LJ potential. Where ordinary units are used, the system refers to argon LJ, with $T_{LJ} = 119.28K$.

Table 1.1. Summary of computer simulations results using the LJ potential.

References	N	Simulation Method	T(K)	S	n*	$\Delta G^*/k_B T$	$\rho_{10}(\sigma^{-3})$	$\rho_1(\sigma^{-3})$	$J(\text{cm}^{-3}\text{s}^{-1})$
ten Wolde and Frenkel	864	MD	88.88	1.53	338	59.4±0.6	0.0188	0.766	4.05×10^5
Yasuoka and Matsumoto	5000	MD	80.27	6.8	30-40				1.1×10^{27}
Oh and Zeng	4000	MC	85	5.33	44	16.73	0.02		
Senger and Reiss		MC	85	5					$1 \times 10^{20,22}$
Chen		MC	83.86			62	0.01202		
Kusaka and Oxtoby	$250 \leq N \leq 260$	MC	89.85, 84.77			39.2, 22.5			

1.8. SECTIONS DESCRIBING THE PRESENT WORK

The present work which uses Monte Carlo simulations to generate growth/decay rate constant ratios for clusters in a dilute LJ vapor system at four LJ argon temperatures, 40K, 50K, 60K and 84K, is described in Sections 2 and 3. The model system is

presented in Section 2.1, the computer simulations in Section 2.2, and the results in Section 2.3. Section 3 describes the nucleation rate calculations from the growth/decay rate constant ratios. In Section 3.2 are presented the results and conclusions.

2. BENNETT METROPOLIS MONTE CARLO SIMULATIONS OF SMALL LJ CLUSTERS

2.1. MODEL SYSTEM AND STATISTICAL MECHANICAL FORMALISM

The first two assumptions of the model are as follows: **(I)** The model system is a classical (full potential) Lennard-Jones dilute vapor with volume, V , composed of a non-interacting mixture of ideal gases with each n -atom cluster size constituting an ideal gas of N_n clusters; and **(II)** At equilibrium the n -clusters grow and decay only *via* interactions with monomers so that the following detailed balance condition is maintained: $N_n \alpha_n = \beta_{n-1} N_1 N_{n-1}$. The β_n and α_n are the equilibrium growth and decay rate constants, respectively, for the n -clusters.

The separable classical Hamiltonian gives rise to the following law of mass action for ratios of the number of n-cluster numbers, N_n , in terms of the n-cluster classical translational partition function, Z_T^n , and configurational integral, Q_n [8].

$$\frac{N_n}{N_1^n} = \frac{Z_T^n Q_n}{Z_T^n Q_1^n n!} = \frac{Q_n}{Q_1^n n!}. \quad (4)$$

As long as the system is not quantum mechanical the result is rigorous. Each configurational integral, Q_n , is divided by V^n , so that $Q_1 = 1$. The following form for the law of mass action is used for calculating Helmholtz free energy differences between neighboring sized clusters:

$$\frac{N_n}{N_{n-1} N_1} = \frac{Q_n}{Q_{n-1} Q_1 n}. \quad (5)$$

Finally, one can express the ratio of the rate constants in terms of the canonical configurational partition functions:

$$\frac{Q_n N_1}{Q_{n-1} Q_1^n} = \frac{\beta_{n-1} N_1}{\alpha_n}. \quad (6)$$

At this point one is faced with the task of reducing the macroscopic volume, V , to some computationally tractable n -cluster volume, v_n . It is important to do this in such a way that one preserves the underlying physics of the law of mass action. Further, the relationship of V to v_n has to be carefully chosen in terms of physical properties. In 1982 Hale and Ward proposed the following approach [77] using the cluster definition of Lee, Barker and Abraham: n atoms constrained within a spherical volume, $v_n = \alpha' n / \rho_{liq}$ [66]. The above equation is then rewritten as [52, 63, 77, 78].

$$\frac{Q_n N_1}{Q_{n-1} Q_1^n} = \frac{Q_n (N_1 / V) \alpha'}{Q_{n-1} Q_1 (v_n / V) \rho_{liq}} = \frac{Q_n \alpha' \rho_1}{Q_{n-1} Q_1' \rho_{liq}}. \quad (7)$$

The quantity $Q_1 (v_n / V) = Q_1'$ is the simulation volume accessible to the monomer and the volume scaling information is in the factor (ρ_1 / ρ_{liq}) . Formally, the expression in Eq. (7) is independent of α' . However, volumes too large or too small can place physically unrealistic constraints on the cluster definition. A working range is $5 < \alpha' < 8$. For small clusters with $n < 5$ (and especially for $n = 2$) the values of α' can affect the free energy difference. For the present simulations $\alpha' = 7$ is used. Most simulations struggle with cluster definition and volume definition -- especially in calculating free energy. But Eq. (7) is a ratio of partition functions and there is a definite advantage in the cancellation of

excluded phase space in both the numerator and in the denominator. Finally, the following Helmholtz free energy notation is used,

$$\begin{aligned}
 -\delta\Delta F_n &= \ln \left[\frac{Q_n \alpha' \rho_1}{Q_{n-1} Q'_1 \rho_{lig}} \right] \\
 &= -\delta f_n - \ln \frac{\rho_{liq}}{\rho_1} \\
 &= \ln \left[\frac{\beta_{n-1} N_1}{\alpha_n} \right]
 \end{aligned} \tag{8}$$

As $n \rightarrow \infty$ the $-\delta f_n \rightarrow \ln(\rho_{liq} / \rho_1)$, and $-\delta\Delta F_n \rightarrow 0$. In some work a cluster is defined as n atoms each within an r_0 distance of at least one other atom in the cluster (cluster definition of Stillinger [81]). In this approach all configurations not satisfying the cluster definition are excluded. This can reduce the phase space sampled and hence the entropy. Also, the choice for r_0 affects the result for the free energy difference. Lauri et al. [72] and Merikanto et al. [80] used the formalism presented here [77] and a Stillinger cluster definition, giving Q_n and Q_{n-1} different volumes. It resulted in an additional $\ln(n)$ term in the energy of formation. In the present calculation of $-\delta\Delta F_n$, the volume (v_n) for the n -cluster in Q_n is equal to the volume for the system containing the $(n-1)$ -cluster and the free monomer. Both have $3n$ degrees of freedom and the same number density. This number density is maintained for all n -clusters. In the next section the Bennett Metropolis Monte Carlo technique [4] used to calculate the ratio in Eq.(8) is presented.

2.2. THE BENNETT METROPOLIS MONTE CARLO METHOD

The Bennett Metropolis Monte Carlo method [4], remains after more than thirty years the most efficient technique for determining free energy differences in computer simulations [84]. The method was developed to calculate the free energy difference between two systems which have the same volume and number of degrees of freedom, but different interaction potentials. In the present work it is used to calculate the ratio of the canonical configurational partition functions in Eq. (8). In this application the two systems, A and B , have $3n$ degrees of freedom, volume, V , and a difference in potential energy, $U_B - U_A$. The free energy difference to be calculated is $-\ln(Q_B / Q_A)$. In the Bennett technique one determines the following number:

$$\ln \frac{Q_B}{Q_A} \equiv C. \quad (9)$$

In the **B** ensemble (corresponding to Q_n) all of the n atoms interact via the LJ potential. In the **A** ensemble (corresponding to $Q_{n-1} Q_1$) $n-1$ atoms interact via the LJ potential and the remaining atom (the probe) has its interactions turned off by a multiplicative factor, λ . That is, $U_B = (U + \Delta U) / kT$, and $U_A = (U + \lambda \Delta U) / kT$, where the ΔU contains only interactions with the n th (probe) atom. As λ goes to zero, the A ensemble becomes an $n-1$ atom cluster plus a free monomer.

Assuming that the prime denotes those cluster configuration integrals with volume, v_n ;

$$\lim_{\lambda \rightarrow 0} \left(\ln \frac{Q_B}{Q_A} \right) = \ln \frac{Q'(n)\alpha}{Q'(n-1)Q'(1)}. \quad (10)$$

In the Bennett technique, (Q_B / Q_A) is calculated for a range of constant C values via the following expression [4]:

$$\frac{Q_B}{Q_A} = \frac{\langle f(U_B - U_A + C) \rangle_A}{\langle f(U_A - U_B - C) \rangle_B} e^C. \quad (11)$$

where $f(x)$ is the Fermi function and $\langle \rangle_A$ and $\langle \rangle_B$ are the Metropolis Monte Carlo canonical averages in the A and B ensembles, respectively. (See Appendix B.) Bennett demonstrated that the Fermi function provides the most efficient sampling. One can see from Eq. (11) that when $\langle f(x) \rangle_A = \langle f(x) \rangle_B$ the value of $\ln(Q_B / Q_A) = C$. One can also determine $\ln(Q_B / Q_A)$ from unequal values of the Fermi function averages and use Eq. (11) to determine $\ln(Q_B / Q_A)$. Combining Eqs. (8) – (11) results in having

$$\begin{aligned} \ln \left[\frac{Q_n \alpha'}{Q_{n-1} Q'_1} \right] &= -\delta f_n = C_n \\ -\delta \Delta F_n &= C_n - \ln \frac{\rho_{liq}}{\rho_1} = \ln \left[\frac{\beta_{n-1} N_1}{\alpha_n} \right] \end{aligned} \quad (12)$$

The growth/decay rate constant ratios so determined are used in the kinetic nucleation rate formalism described in the next section.

2.3. THE KINETIC NUCLEATION RATE FORMALISM

The final step in applying the small cluster simulations to a calculation of the nucleation rate is to make use of the kinetic nucleation rate formalism [6, 15, 14]. This

formalism employs the following assumptions for the steady state cluster size concentrations, N^{ss}/V : (1) that clusters grow and decay only *via* interactions with monomers; (2) that the nucleation rate is time independent ($dN_n^{ss}/dt = J$ for all n); (3) that the n -clusters in the steady state size distributions grow and decay with the equilibrium rate constants, β_n and α_n ; and (4) that the concentration of monomers is constant (monomers leaving the system are replaced). With these conditions the steady state nucleation rate (See Appendix A.), J , is given by

$$\frac{1}{J} = \sum_{n=1}^M \frac{1}{J_n} \quad (13)$$

where M is large enough so that the sum converges. The J_n for $n \geq 2$ are given by

$$J_n = \beta_n [\rho_1 S]^2 \prod_{j=2}^n \frac{\beta_{j-1} N_1 S}{\alpha_j} \quad (14)$$

and $J_1 = \beta_1 [\rho_1 S]^2$, with $\rho_1 = N_1/V$ at coexistence. The equilibrium monomer flux on the n -cluster, $\beta_n \rho_1$ is approximated using $\beta_n = (v_{av}/4)\pi r_n^2$ with $[n/\rho_{liq}] = [4/3]\pi r_n^3$ and $v_{av} = [8kT/\pi m]^{1/2}$. In general M slightly larger than the critical cluster size suffices. In this formalism $S = N_1^{ss}/N_1$, is the monomer supersaturation ratio. With Eqs. (12)-(14) the steady state nucleation rate, J , can be determined from the following expression:

$$\frac{1}{J} = \frac{1}{J_1} + \sum_{n=2}^M \left[\beta_n (\rho_1 S)^2 \prod_{j=2}^n \frac{Q_j N_1 S}{Q_{j-1} Q_{1j}} \right]^{-1}. \quad (15)$$

The coexistence monomer concentration ρ_1 must be determined from the properties of the LJ potential and will be discussed in the data analysis. The approach has many

advantages. While calculating the β_n and α_n it is not necessary to impose a supersaturation. Also, J can be calculated for any S using Eq. (15) as long as the vapor remains dilute so that interactions between clusters are negligible. The nucleation rates are based on a well defined model which, while somewhat restrictive with regard to vapor density, can yield physical insights into the nucleation process.

3. THE MONTE CARLO SIMULATION RESULTS

3.1. COMPUTATIONAL DETAILS

The Bennett Metropolis Monte Carlo Method is used to calculate the free energy differences, $\delta f_n = -C_n$, between the n -cluster and $(n-1)$ -cluster plus free monomer for values of n ranging from $n = 2$ to $n = 192$. The value of λ is 10^{-9} except for small cluster sizes $n = 2, 3, 6, 10,$ and 11 , where a value of $\lambda = 10^{-7}$ is used. The efficiency of the Bennett technique relies on an adequate overlap of $(U_B - U_A)$, the potential energy difference between the two ensembles, and the latter quantity is monitored during the run. All averaging is done after equilibration of the ensembles.

The Bennett technique requires only one atom (say, the n th) as a probe in the B ensemble. However, there is no distinction amongst atoms in the B ensemble. So, to carry out a more efficient free energy difference calculation the $\langle f(U_A - U_B - C) \rangle_B$ is determined for each of the i atoms. An average of the contributions from each atom is used to determine the $\langle f(U_A - U_B - C) \rangle_B$ used in Eq. (11):

$$\langle f(U_A - U_B - C) \rangle_B = \frac{1}{n} \sum_{i=1}^n \langle f(U_A - U_B - C) \rangle_{i_B} = \frac{1}{n} \sum_{i=1}^n \langle f(-(1-\lambda)\Delta U - C) \rangle_{i_B} \quad (16)$$

The individual C_i values from the i probes in the B ensemble are also calculated and compared with C_n from the average of the Fermi functions.

3.2. METROPOLIS MONTE CARLO METHOD

The Metropolis Monte Carlo simulations [5] applied to ensembles A and B in the present work assume constant N, V, T (the canonical ensemble) and use the Boltzmann factor to select configurations from the phase space. The N, V and T are n, v_n and T for the n -cluster systems. The resulting Markovian random walk in the configuration space depends only on the previous step and not on the history of steps taken before. All the steps must be reversible. A schematic view of the Metropolis Monte Carlo process is shown in Figure 3.1.

In the present simulations every atom is moved a random distance at every step. This provides an enhancement in computational speed by eliminating repeated do loop operations and facilitates the equilibration process. Thus in the present work each million steps is equivalent to n million steps in a standard Metropolis Monte Carlo. The maximal displacement, however is reduced to maintain an acceptance ratio of 0.5. The acceptance ratio (number of Monte Carlo steps accepted divided by the total number of steps) is optimal (though not mandatory) at 0.5 and can be increased (decreased) by decreasing (increasing) the maximal displacement, d_{\max} . [56].

Because the free energy difference, C_n , includes entropy the C_n values fluctuate substantially at the beginning of the averaging Monte Carlo run. The fluctuations become smaller as the simulation progresses. The n clusters require 2M-10M steps for equilibration, depending on the value of n . Once equilibration is achieved, the averaging run for C_n is initiated. The values for M_{step} for clusters $n = 2$ to 192 are given in Tables (3.1-3.3).

As noted earlier, the cluster of n atoms is constrained within a spherical volume, $v_n^* = \alpha' [n / \rho_{liquid}]$, and the internal cluster number density for all n -clusters, $[\rho_{liquid} / \alpha']$, is a constant. Lee et al. [66] found that the Helmholtz free energy differences are not sensitive to the cluster volume within a typical range of $5 < \alpha' < 8$. In the present work $\alpha' = 7$ is used. Formally, the expressions in Eqs. (8)-(12) are independent of α' . However, volumes too small or too large can place physically unrealistic constraints on the cluster definition. In volumes too small the atoms are subject to a “wall” pressure and in volumes too large, the cluster tends to break up into smaller clusters.

3.3. THE FREE ENERGY DIFFERENCES

This quantity calculated in the present Bennett Monte Carlo simulations is $-\delta f_n$, the magnitude of the Helmholtz free energy difference between an n -atom cluster and an $(n-1)$ -atom cluster plus a free monomer. From these $-\delta f_n$ a sequence of growth/decay rate constant ratios, β_{n-1} / α_n , for n ranging from 2 to 192 are determined. The simulations are carried out at the LJ reduced temperatures, $Tk/\epsilon \equiv T^* = 0.335, 0.419, 0.503, \text{ and } 0.700$ corresponding roughly to experimental argon temperatures of $T = 40, 50, 60, \text{ and } 83.6\text{K}$.

In the analysis the full potential Lennard-Jones system critical temperature, $T_c^* = 1.313 \pm 0.003$, and reduced critical density, $\rho_c^* = 0.316 \pm 0.001$ are used [83]. When the ratio T_c/T is used the reduced temperature notation, T^* , is omitted. The values of the Helmholtz free energy differences, $-\delta f_n$, are given in Tables (3.1-3.4) and Fig. 3.2

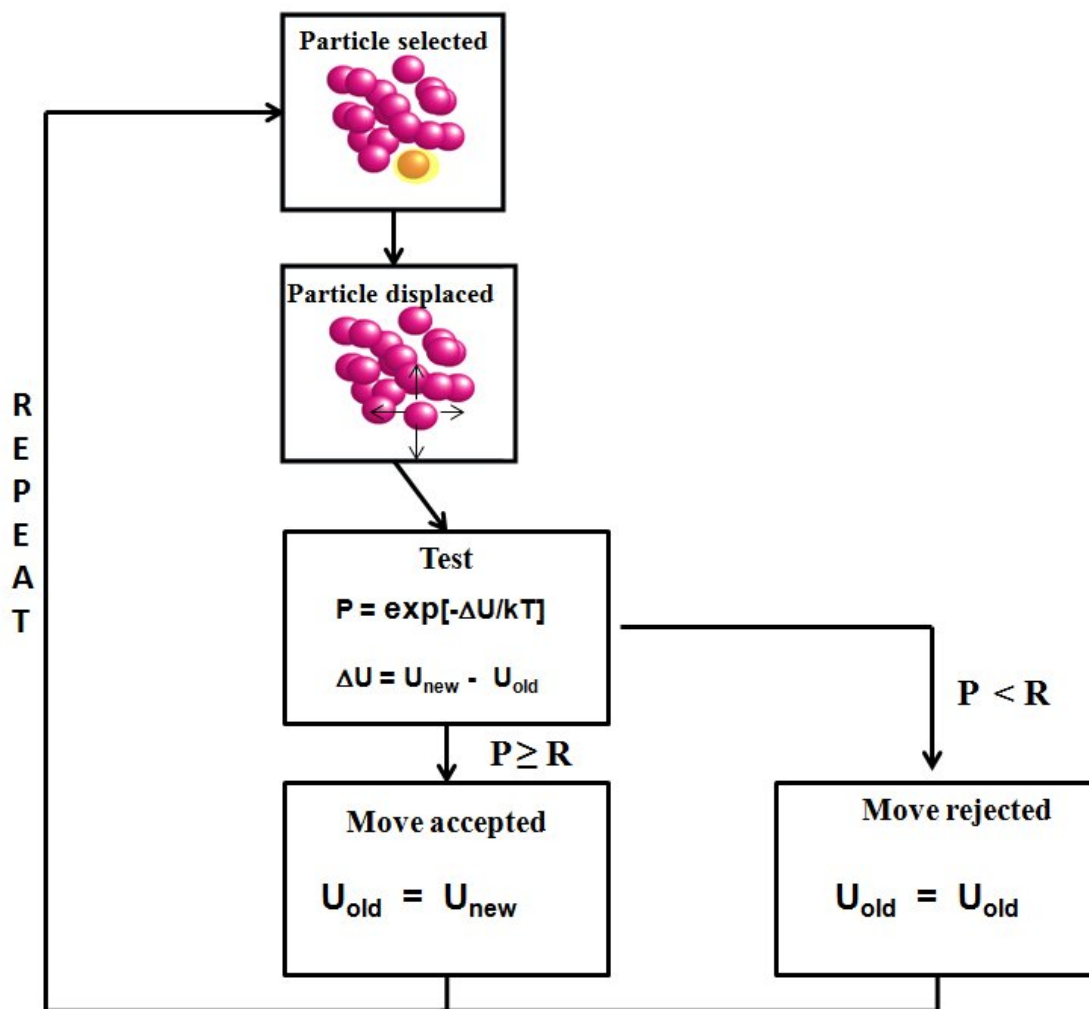


Figure 3.1. Schematic view of the Metropolis Monte Carlo process.

shows the free energy differences, $-\delta f_n$, for the four temperatures. The intercepts, I_0 , at infinite n in Figure 3.2 can be determined from a least square fit to the data for $n > 20$ and are found to be 17.7, 12.9, 9.90 and 5.98 at $T = 40, 50, 60$ and 83.6K , respectively. These intercepts [equal to $\ln(\rho_{\text{liq}} / \rho_1)$] are used to find ρ_1 for the Lennard-Jones

potential model. The abscissa, $n^{-1/3}$ is proportional to $\frac{d}{dn} An^{2/3}$ and the slope of the line is proportional to an effective surface tension.

In Fig.3.3 is a plot of the $\delta\Delta F_n$ divided by $\left[\frac{T_c}{T}-1\right]$. Except in the case of the smallest clusters ($n \leq 6$) the values of $\delta\Delta F_n = I_0 - |\delta f_n|$ in Fig.3.3 are observed to fall on a single line and indicate a scaled temperature dependence. The abscissa, $n^{-1/3}$, is not a good fit to the n -dependence of free energy differences for small clusters, which are increasing the number of bonds as they add another atom. The larger clusters, on the other hand, are increasing their excess surface free energy as they add an atom. A similar scaling was observed [63] for TIP4P [57] and RSL2 potential water clusters and suggests that the scaling is not dependent upon the potential model. The dotted line in Fig. 3.3 represents a classical spherical “drop” Lennard-Jones model with an excess surface entropy per atom of $\Omega = 2.19$ k. The latter is consistent with the experimental surface tension of liquid argon. A major advantage of this scaling is that the scaled $\delta\Delta F_n$ can be used to predict nucleation rates at temperatures other than those for which the simulations are carried out [52].

3.4. THE NUCLEATION RATES

The goal of this study has been to examine the scaling of nucleation rates previously demonstrated in the homogenous vapor-to-liquid nucleation rate data for water and for the toluene data of Schmitt [24] as seen in Fig.1.1. The nucleation rates presented here are calculated from the growth to decay rate constant ratios defined in Eq. (12) and

Table 3.1 Values of configurational free energy differences and number of M_{step} to calibrate each cluster for $n=2-15$ at $T=40, 50, 60, 83.6K$. Values of configurational free energy differences for the smaller clusters, $n=2-15$, at $T = 40, 50, 60, 83.56K$, and $M_{steps} = \text{number of steps}/10^6$ over which the average were determined.

n	$-\delta f_n$ T = 40K	M_{step}	$-\delta f_n$ T = 50K	M_{step}	$-\delta f_n$ T = 60K	M_{step}	$-\delta f_n$ T = 83.58K	M_{step}
15	8.9±.03	20	6.6±.02	20	5.1±.07	20	3.3±.03	20
14	8.7±.08	20	6.4±.06	20	5.0±.05	20	3.2±.07	20
13	8.5±.05	20	6.2±.08	20	4.9±.02	20	3.2±.02	20
12	8.2±.03	20	6.1±.01	20	4.7±.07	20	3.1±.07	20
11	7.9±.04	20	5.9±.03	20	4.6±.02	20	3.1±.02	20
10	7.7±.05	20	5.7±.06	20	4.4±.06	20	3.0±.05	20
9	7.4±.01	20	5.5±.01	20	4.2±.08	20	2.9±.08	20
8	7.0±.02	20	5.1±.07	20	4.0±.04	20	2.9±.01	20
7	6.6±.01	20	4.8±.03	20	3.8±.01	20	2.8±.03	20
6	5.9±.09	20	4.4±.01	20	3.5±.08	20	2.7±.03	20
5	5.3±.02	20	3.9±.07	20	3.2±.09	20	2.6±.02	20
4	4.4±.07	20	3.5±.04	20	3.0±.03	20	2.4±.09	20
3	3.6±.03	20	3.0±.02	20	2.5±.06	20	2.3±.03	20
2	2.3±.06	20	2.3±.02	20	2.2±.01	20	2.0±.09	20

Table 3.2 Values of configurational free energy differences and number of M_{step} to calibrate each cluster for $n=16-30$ at $T=40, 50, 60, 83.6K$. Values of configurational free energy differences for the smaller clusters, $n=16-30$, at $T = 40, 50, 60, 83.56K$, and M_{steps} = number of steps/ 10^6 over which the average were determined.

n	$-\delta f_n$ T = 40K	M_{step}	$-\delta f_n$ T = 50K	M_{step}	$-\delta f_n$ T = 60K	M_{step}	$-\delta f_n$ T = 83.58K	M_{step}
30	10.6±.03	30	7.8±.09	30	6.2±.01	30	3.931	30
29	10.5±.05	20	7.8±.06	15	6.1±.07	10	3.942	20
28	10.4±.04		7.7±.08		6.1±.01			
27	10.4±.02	30	7.7±.04	20	6.0±.06	20	3.8±.01	20
26	10.3±.02	30	7.6±.07	10	6.0±.03	20	3.7±.07	20
25	10.2±.02	20	7.5±.05	20	5.9±.09	20	3.7±.05	20
24	10.1±.02	20	7.4±.09	20	5.9±.02	20	3.7±.01	20
23	10.0±.03	30	7.4±.09	20	5.9±.01	20	3.7±.01	20
22	9.8±.06	20	7.3±.04	20	5.7±.07	20	3.6±.06	20
21	9.7±.07	20	7.2±.04	10	5.6±.08	30	3.5±.09	20
20	9.6±.01	20	7.1±.08	20	5.6±.01	20	3.5±.06	20
19	9.4±.07	20	7.1±.02	20	5.5±.08	20	3.5±.03	20
18	9.3±.01	20	6.9±.06	20	5.4±.07	20	3.4±.07	20
17	9.1±.08	20	6.8±.06	20	5.3±.06	20	3.4±.03	20
16	9.0±.09	20	6.7±.07	20	5.2±.07	20	3.3±.08	20

Table 3.3 Values of configurational free energy differences and number of M_{step} to calibrate each cluster for $n=32-90$ at $T=40, 50, 60, 83.6K$. Values of configurational free energy differences for the smaller clusters, $n=32-90$, at $T = 40, 50, 60, 83.56K$, and M_{steps} = number of steps/ 10^6 over which the average were determined.

n	$-\delta f_n$ T = 40K	M_{step}	$-\delta f_n$ T = 50K	M_{step}	$-\delta f_n$ T = 60K	M_{step}	$-\delta f_n$ T = 83.58K	M_{step}
90	12.6±.02	90	9.3±.05	90	7.3±.05	90	4.6±.09	90
80	12.4±.02	80	9.2±.02	80	7.2±.01	80	4.6±.01	80
70	12.2±.02	70	9.0±.06	70	7.0±.08	70	4.5±.02	70
65	12.0±.07		9.0±.05		7.0±.09			
60	11.9±.04	30	8.8±.09	60	6.9±.07	60	4.4±.03	60
55	11.7±.07	50	8.7±.07	50	6.9±.01	50	4.3±.08	50
50	11.6±.08	50	8.6±.06	50	6.8±.03	50	4.3±.03	50
45	11.4±.02	30	8.4±.09	30	6.6±.06	30	4.2±.05	10
40	11.1±.07	40	8.3±.03	30	6.5±.09	30	4.1±.05	20
35	10.9±.02	40	8.1±.04	40	6.3±.07	30	4.0±.06	40
32	10.7±.05	20	7.9±.07	20	6.3±.02	20	3.9±.08	20

Table 3.4 Values of configurational free energy differences and number of M_{step} to calibrate each cluster for $n=100-192$ at $T=40, 50, 60, 83.6K$. Values of configurational free energy differences for the smaller clusters, $n = 100-192$, at $T = 40, 50, 60, 83.56K$, and $M_{steps} =$ number of steps/ 10^6 over which the average were determined.

n	$-\delta f_n T$ = 40K	M_{step}	$-\delta f_n T =$ 50K	M_{step}	$-\delta f_n T$ = 60K	M_{step}	$-\delta f_n T =$ 83.58K	M_{step}
192	13.8±.07	50	10.2±.01	60	7.9±.05	70	5.0 ±.03	50
180	13.8±.01	70	10.1±.07	20	7.8±.08	20	4.9±.09	20
170	13.6±.06	80	10.1±.04	40	7.9±.02	20	4.9±.07	50
160	13.5±.03	50	9.9±.09	60	7.8±.01	50	4.9±.03	50
150	13.5±.06		9.9±.06		7.8±.02		4.8±.09	
140	13.3±.08	50	9.8±.05	70	7.7±.01	50	4.8±.06	30
130	13.0±.04		9.7±.09		7.6±.07			
120	13.0±.02	30	9.6±.09	10	7.5±.09	20	4.8±.04	10
110	12.8±.07	30	9.6±.02	100	7.5±.04	100	4.7±.07	100
100	12.7±.07	100	9.5±.01	100	7.4±.04	100	4.7±.03	100

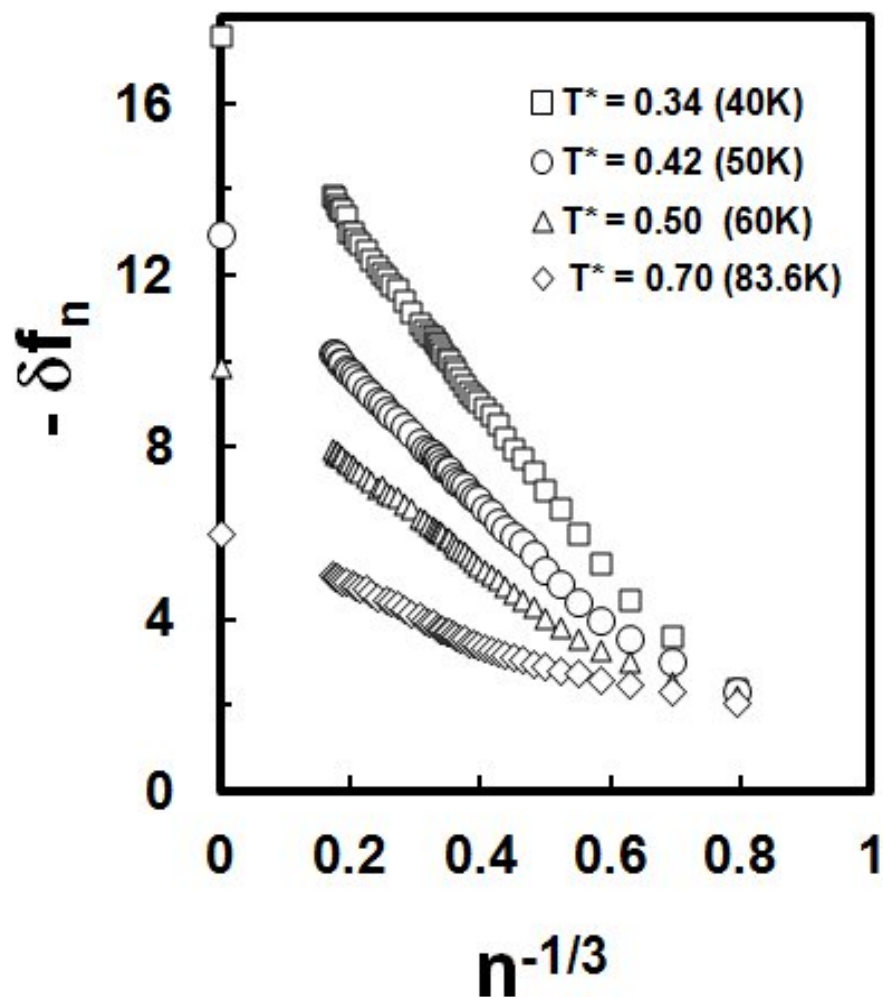


Figure 3.2. Plot of free energy differences vs. $n^{-1/3}$, at $T=40, 50, 60, 83.6\text{K}$. Helmholtz free energy differences (in units of kT), $-\delta f_n$, at $T = 40\text{K}, 50\text{K}, 60\text{K}, 83.6\text{K}$, plotted versus $n^{-1/3}$. The value of n ranges from 2 to 192. The values are given in the following Tables.

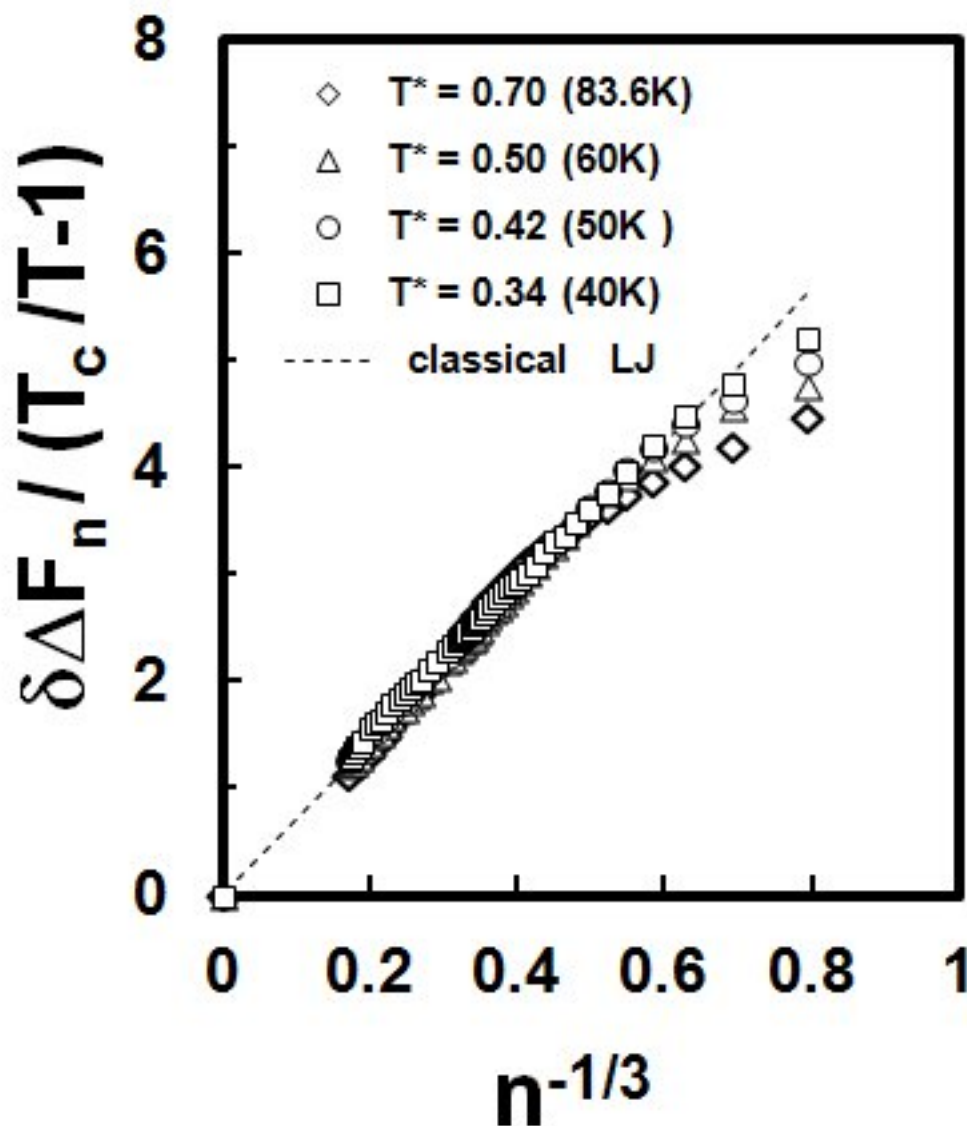


Figure 3.3. Plot of $\frac{\delta\Delta F_n}{\left(\frac{T_c}{T} - 1\right)}$ vs. $n^{-1/3}$, at $T=40, 50, 60, 83.6\text{K}$. The $\delta\Delta F_n$ divided by $\left[\frac{T_c}{T} - 1\right]$. The value of n ranges from 2 to 192. The classical Lennard Jones model is represented by the dotted line with excess surface entropy/ k per atom, $\Omega=2.19$.

are generated by the Bennett Metropolis Monte Carlo simulations of small Lennard-Jones clusters in the dilute vapor system. The primary assumptions are the law of mass action and the kinetic steady state nucleation rate formalism [6, 15, 13].

In Fig. 3.4 is a plot of the nucleation rates versus $\ln(S)$ for the $T = 40, 50, 60,$ and 83.6K (argon) Lennard-Jones system. The values of the supersaturation, S , are chosen so that the nucleation rates range from 10^4 to $10^7 \text{ cm}^{-3} \text{ s}^{-1}$ -- close to the onset nucleation rates observed experimentally for argon by Strey and Iland.[35] In Fig. 3.5 is a plot of the same nucleation rates for the four temperatures plotted vs. the scaled supersaturation,

$\ln S / \left[\frac{T_c}{T} - 1 \right]^{3/2}$. When the nucleation rates are plotted versus the scaled supersaturation,

all four temperature lines collapse onto a single line as shown in Fig. 3.5. That is, the scaling such as that shown in Fig. 1.1 for the experimental toluene data is evidenced in the simulation generated Lennard Jones nucleation rates calculated here.

The scaling of the nucleation rates in Fig. 3.5 suggests that the nucleation rate is not a function of the supersaturation, S , and temperature, T , independently, but a function of the scaled supersaturation, $\ln S / [T_c / T - 1]^{3/2}$. The scaling in the present model comes in part from the discrete summation over the small cluster contributions to the n -cluster energy of formation as well as the corresponding states $[T_c / T - 1]$ temperature dependence of the $\delta\Delta F_n^*$. The discrete summation over the small cluster free energy differences introduces terms proportional to $[T_c / T - 1]$, which cancel the temperature, T , dependence in the monomer flux factor [51].

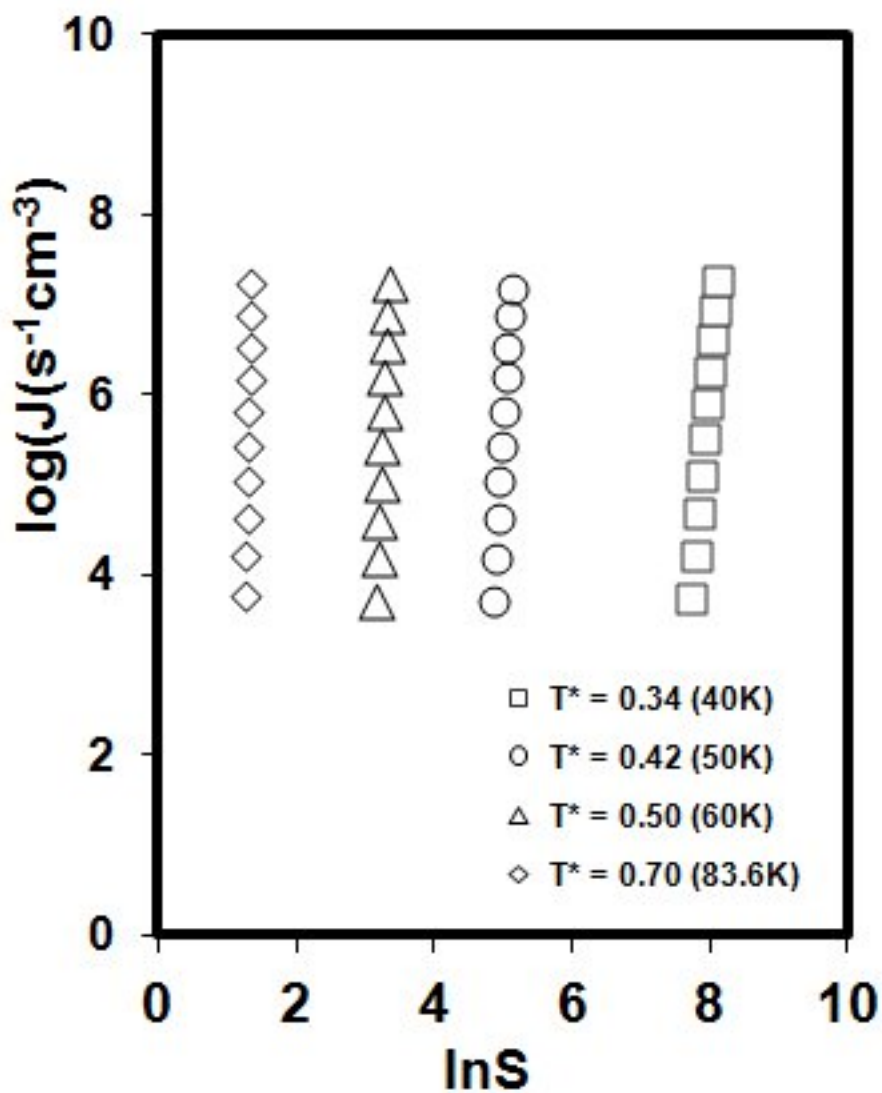


Figure 3.4. Plot of $\log(J/s^{-1}cm^{-3})$ vs. $\ln S$ at $T^*=40, 50, 60, 83.6K$. The nucleation rate, J , vs. $\ln S$ for $T = 40, 50, 60,$ and $83.6K$ calculated from the small LJ cluster growth/decay rate constant ratios generated by the Bennett Metropolis Monte Carlo simulation.

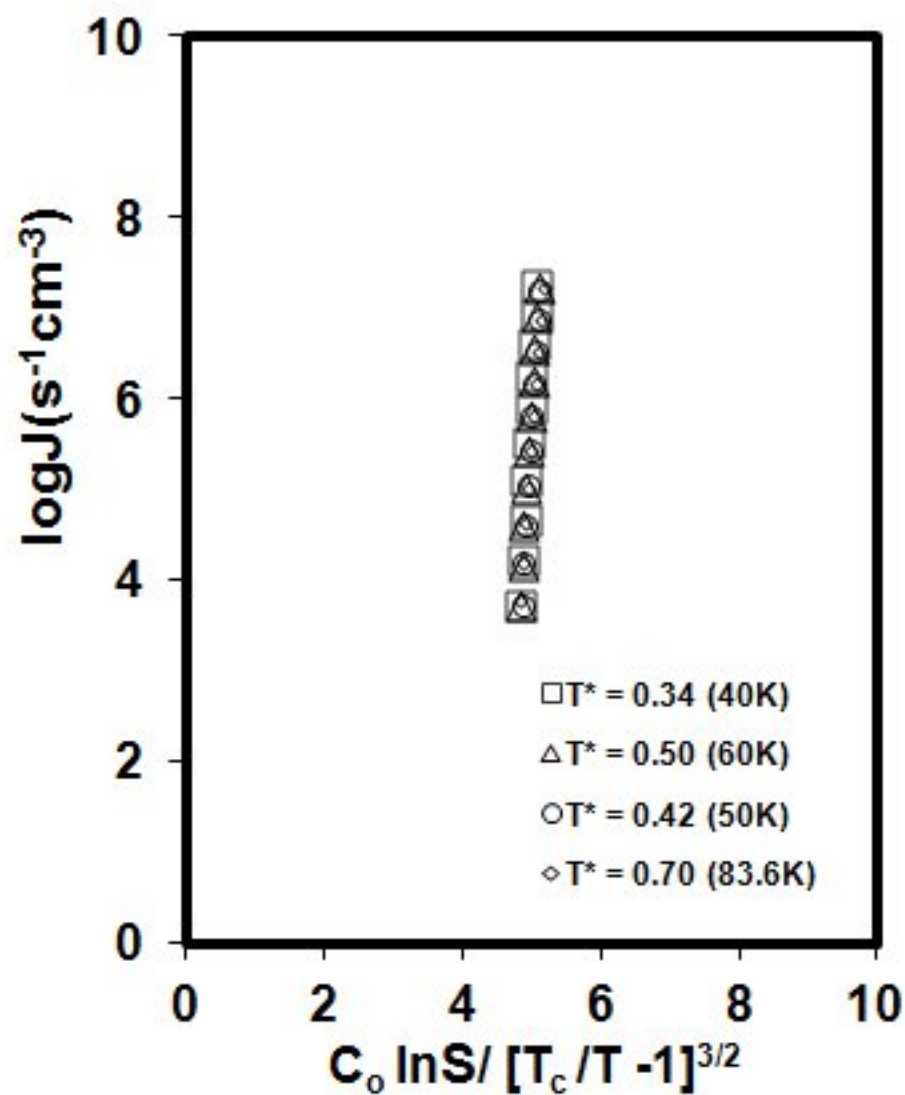


Figure 3.5. Plot of $\log(J / s^{-1} cm^{-3})$ vs. $\frac{C_0 \ln S}{\left[\frac{T_c}{T} - 1\right]^{3/2}}$ at $T^*=40, 50, 60, 83.6K$. The

nucleation rate, J , vs. the scaled supersaturation, $C_0 \ln(S) / [T_c / T - 1]^{3/2}$ for $T = 40, 50, 60$, and $83.6K$. The rates are calculated from the small LJ cluster growth/decay rate constant ratios generated by the Bennett Metropolis Monte Carlo simulation. The $C_0 = [T_c / 50 - 1]^{3/2}$ so that the $T = 50K$ data falls on the same abscissa points as in Figure 3.4.

The majority of the nucleation rate data is at low temperatures, and there is a lack of vapor monomer number densities (and liquid number densities) from computer simulations at these same low temperatures. This lack of vapor monomer number densities makes it difficult to calculate a nucleation rate. In the present case the determination of a vapor pressure, ρ_1 , for the Lennard-Jones full potential from the intercepts, I_0 , in Fig. 3.2 is crucial. The values of ρ_1 are determined using the corresponding states approach of Dunikov [82]. Dunikov demonstrated a quantitative agreement between Lennard-Jones potential model systems (full and cutoff) and the properties of experimental argon using reduced quantities. Using this approach, one can estimate the LJ liquid number densities at low temperatures and determine the ρ_1 , for the Lennard-Jones full potential from the intercepts, I_0 , in Fig. 3.2. As a test of this approach the four vapor pressures corresponding to intercepts, I_0 , in Fig. 3.2 are plotted in Figure 3.6 using the reduced quantities. In this figure, a comparison is made of the vapor pressures determined from the present calculations, the extrapolated argon vapor pressure formula used by Fladerer [34] and Iland et al. [35], the Lennard-Jones Monte Carlo results from Chen et al.[85], and the experimental argon data of Gilgen et al. [86]. It is interesting that the scaling of the nucleation rates fails if the vapor pressures at low temperature deviate markedly from the dashed line – essentially the extrapolated vapor pressure formula of Fladerer [34] and Iland and Strey [35].

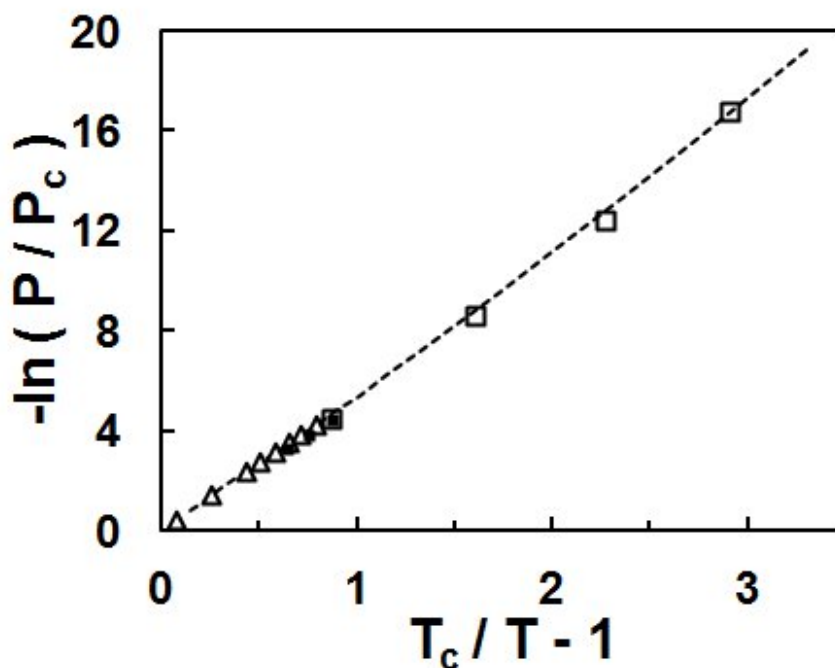


Figure 3.6. Plot of $-\ln(P/P_c)$ vs. $\frac{T_c}{T} - 1$. Corresponding states comparison of $-\ln(P/P_c)$ from the present study (\square), experimental argon vapor pressure data [86] (Δ), and the Lennard-Jones Monte Carlo simulations of Chen et al. [85] (\blacksquare); the dashed line is the argon vapor pressure formula of Iland and Strey [35] and Fladerer [34].

In Figs. 3.7 – 3.10 are snapshots of the ensemble configurations. Figure 3.7 shows the superposition of configurations every million steps for the $n = 120$ particle LJ cluster during a 10M step run at $T = 83.6\text{K}$. Figure 3.8 is a similar superposition of configurations separated by one million steps for the $n = 150$ particle cluster at $T = 83.6\text{K}$ during a 20M step run. In Fig. 3.9 is a snapshot of the $n = 27$ particle LJ cluster at 40K after 10M steps. The last figure, Fig. 3.10, is a snapshot of the $n = 192$ particle LJ cluster at $T = 60\text{K}$ after 70M steps. It was generally necessary to run the cluster simulation about n million steps to obtain a reliable value for the free energy difference.

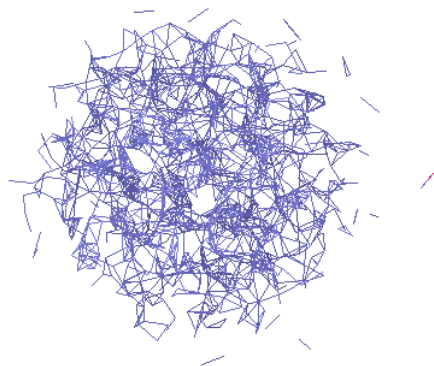


Figure 3.7. A snapshot of the B ensemble for an $n=120$ configuration at $T=83.6\text{K}$ after 10M Monte Carlo steps, superimposing each configurations at every 1M steps.

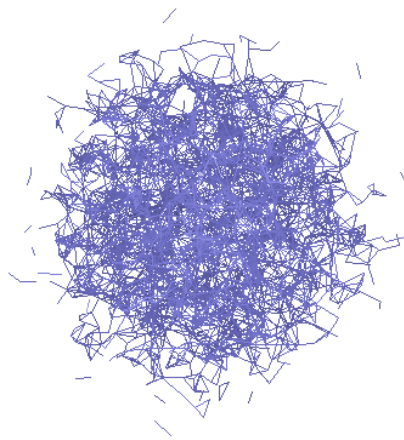


Figure 3.8. A snapshot of the B ensemble for an $n=150$ configuration at $T=83.6\text{K}$ after 20M Monte Carlo steps, superimposing each configurations at every 1M steps.

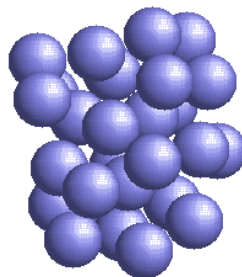


Figure 3.9. A snapshot of the B ensemble for an $n=27$ configuration at $T=40\text{K}$ after 10M Monte Carlo steps.

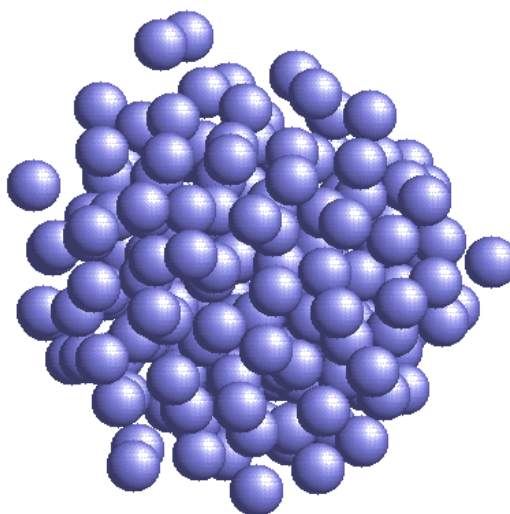


Figure 3.10. Snapshot of the B ensemble for an $n=192$ configuration at $T=60\text{K}$ after 70M Monte Carlo steps.

4. CONCLUSIONS

In this study, Helmholtz configurational free energy differences for neighboring sized n -atom LJ clusters at argon LJ temperatures of $T = 40\text{K}$, 50K , 60K , and 83.6K are determined using a Bennett Monte Carlo simulation and a Lennard-Jones full potential. Small cluster growth/decay rate constant ratios are determined from the free energy differences and are applied to a kinetic steady state nucleation rate calculation of vapor to liquid nucleation rates for the dilute Lennard Jones vapor. The free energy differences scale with temperature as $(T_c/T-1)$ and the nucleation rates for the four temperatures are found to collapse on a single line when plotted vs. the scaled supersaturation, $\ln S/(T_c/T-1)^{3/2}$. This simulation-generated demonstration of temperature scaling for a dilute vapor to liquid model system is similar to that found for experimental water and toluene nucleation rate data. It indicates that the observed experimental scaling should be attributable to the simple assumptions made in the statistical mechanical treatment of the small vapor clusters and the steady state kinetic nucleation formalism. These include the law of mass action, and a discrete summation over small cluster contributions to the energies of formation of the critical sized cluster.

APPENDIX A.

STEADY STATE NUCLEATION RATE

STEADY STATE NUCLEATION RATE

The starting point for deriving the steady state nucleation rate is the following expression for the time derivative of the number of n clusters, N_n^s :

$$\frac{dN_n^s}{dt} = (\beta_{n-1}N_1^sN_{n-1}^s - \alpha_nN_n^s) - (\beta_nN_1^sN_n^s - \alpha_{n+1}N_{n+1}^s).$$

The first assumption is that $\frac{dN_n^s}{dt} = 0$ where the superscript denotes the steady state number of n clusters. The second assumption is that the first condition is satisfied as follows:

$$J = (\beta_{n-1}N_1^sN_{n-1}^s - \alpha_nN_n^s) = (\beta_nN_1^sN_n^s - \alpha_{n+1}N_{n+1}^s)$$

where J is constant.

We start with the above expression for J.

$$J = \beta_{n-1}N_1^sN_{n-1}^s - \alpha_nN_n^s$$

Dividing by $\beta_{n-1}N_1N_{n-1}$, setting $\frac{N_1^s}{N_1} = S$ and using detailed balance which holds for the equilibrium concentrations, $\beta_{n-1}N_1N_{n-1} = \alpha_nN_n^s$, the above expression for J becomes:

$$\frac{J}{\beta_{n-1}N_1N_{n-1}} = S \frac{N_{n-1}^s}{N_{n-1}} - \frac{\alpha_nN_n^s}{\beta_{n-1}N_1N_{n-1}} = S \frac{N_{n-1}^s}{N_{n-1}} - \frac{N_n^s}{N_n}$$

$$\text{Finally, } \frac{J}{S^n \beta_{n-1}N_1N_{n-1}} = \frac{N_{n-1}^s}{S^{n-1}N_{n-1}} - \frac{N_n^s}{S^n N_n}$$

Defining $\frac{N_{n-1}^S}{S^{n-1}N_{n-1}} = F(n-1)$, and $\frac{N_n^S}{S^n N_n} = F(n)$ the following simplification can be made:

$$\frac{J}{S^n \beta_{n-1} N_1 N_{n-1}} = F(n-1) - F(n) = -\delta_n F(n).$$

Taking the sum over both sides and set $\beta_n N_1 N_n S^{n+1} = J_n$

$$\sum_{n=2}^{n^*+m} \frac{J}{J_{n-1}} = \sum_{n=2}^{n^*+m} -\delta_n F(n)$$

$$\sum_{n=2}^{n^*+m} \frac{J}{J_{n-1}} = F(n^* + m) - F(1),$$

The $F(n^* + m) = 0$ since clusters larger than the critical size grow to a macroscopic size and leave the system. The $f(1) = 1$

$$\sum_{n=1}^M \frac{J}{J_n} = 1$$

$$\frac{1}{J} = \sum_{n=1}^M \frac{1}{J_n} \quad \text{where } M > n^*$$

$$= \frac{1}{J_1} + \sum_{n=2}^M \left[\beta_n N_1 S^{n+1} N_n \right]^{-1}$$

$$= \frac{1}{J_1} + \sum_{n=2}^M \left[\beta_n N_1 S^{n+1} \left(\frac{N_n}{N_{n-1}} \right) \left(\frac{N_{n-1}}{N_{n-2}} \right) \dots \left(\frac{N_2}{N_1} \right) N_1 \right]^{-1}$$

$$= \frac{1}{J_1} + \sum_{n=2}^M \left[\beta_n N_1 N_1 S^{n+1} \left(\frac{\beta_{n-1} N_1}{\alpha_n} \right) \left(\frac{\beta_{n-2} N_1}{\alpha_{n-2}} \right) \dots \left(\frac{\beta_1 N_1}{\alpha_2} \right) \right]^{-1}$$

$$\frac{1}{J} = \frac{1}{J_1} + \sum_{n=2}^M \left[\beta_n N_1^2 S^2 \prod_{j=2}^n \frac{\beta_{j-1} N_1 S}{\alpha_j} \right]^{-1}$$

This is the final expression for the steady state nucleation rate.

APPENDIX B.

**DERIVATION OF THE BENNETT TECHNIQUE WITH
FERMI FUNCTION WEIGHTING**

Some Background

We define the canonical “configurational” integral as the following:

$$\begin{aligned} Q_B(n, V) &\equiv \int \dots \int_V \exp(-U_B / kT) d\vec{r}^1 d\vec{r}^2 \dots d\vec{r}^n \\ &= \iiint_{V^n} \exp(-u_B) d\vec{r}^{3n} \end{aligned}$$

$$Q_A(n, V) \equiv \iiint_{V^n} \exp(-u_A) d\vec{r}^{3n}$$

$$\langle u_A \rangle = \frac{1}{Q_A} \iiint_{V^n} u_A \exp(-u_A) d\vec{r}^{3n}$$

$$\langle u_B \rangle = \frac{1}{Q_B} \iiint_{V^n} u_B \exp(-u_B) d\vec{r}^{3n}$$

For a system with n particles in volume, V at temperature, T , the total Helmholtz free energy is given by $F_{total}(n, V) = -kT \ln Z(n, V)$. where $Z(n, V)$ is the canonical partition function (constant n , V , T). For many classical systems (no quantum mechanics) the interaction potential is not a function of the momenta, \vec{p}_j . In this case the Hamiltonian, $H = \sum_j p_j^2 / 2m_j + \sum_{i,j} V(r_i, r_j)$, is separable in \vec{r}_j and \vec{p}_j and the canonical partition function is also separable. That is, the partition function momentum and configuration integrations factor.

$$\begin{aligned} Z(n, V) &= \frac{1}{n! h^{3n}} \iiint \dots \int_V \iiint \dots \int_{\text{all } \vec{p}} \exp(-H_B / kT) d\vec{r}^{3n} d\vec{p}^{3n} \\ &= \left[\frac{1}{h^{3n}} \iiint \dots \int_{\text{all } \vec{p}} \exp(-\sum_j (p_j^2 / 2m_j) / kT) d\vec{p}^{3n} \right] \left[\frac{1}{n!} \iiint \dots \int_{V^n} \exp(-\sum_{i,j} V(r_i, r_j) / kT) d\vec{r}^{3n} \right] \\ &= Z_{trans}(n, V) [Q(n, V) / n!] \\ &= [Z_{trans}(n, V)]^n Q(n, V) / n! \\ Z_{trans}(1, V) &= (2\pi m kT / h^2)^{3/2} \\ &= (\text{thermal wavelength})^{-3} \end{aligned}$$

Note that

$$Z_{trans}(n, V) \approx \frac{1}{(\text{wavelength}^3)^n}$$

$$Q(n, V) / n! \approx V^n$$

so that the partition function is unitless. Recall that $d\vec{r}d\vec{p} / h^3$ "counts" the number of phase space "states" in $d\vec{r}d\vec{p}$.

In this simulation, the following ratio was of interest:

$$\begin{aligned} \frac{Z(n, V)}{Z(n-1, V)Z(1, V)} &= \frac{Q(n, V)}{Q(n-1, V)Q(1, V)n} = \frac{Q_B}{Q_A} \\ &= F(n, V) / kT - F(n-1, V) / kT - F(1, V) / kT \end{aligned}$$

5. Deriving the Bennett formula

We start with an identity (C is a constant):

$$\iiint \frac{1}{e^{u_A - C/2} + e^{u_B + C/2}} d\vec{r}^{3n} = \iiint \frac{1}{e^{u_A - C/2} + e^{u_B + C/2}} d\vec{r}^{3n}$$

Multiply left side under the integral by $e^{-u_A + C/2} / e^{-u_A + C/2}$ and the right side under the integrals by $e^{-u_B - C/2} / e^{-u_B - C/2}$:

$$\iiint \frac{e^{-u_A + C/2}}{1 + e^{u_B - u_A + C}} d\vec{r}^{3n} = \iiint \frac{e^{-u_B - C/2}}{1 + e^{u_A - u_B - C}} d\vec{r}^{3n}$$

Now multiply each side by $e^{C/2}$

$$e^C \iiint \frac{e^{-u_A}}{1 + e^{u_B - u_A + C}} d\vec{r}^{3n} = \iiint \frac{e^{-u_B}}{1 + e^{u_A - u_B - C}} d\vec{r}^{3n}$$

$$e^C = \left[Q_B \left\langle \frac{1}{1 + e^{u_A - u_B - C}} \right\rangle_B \right] / \left[Q_A \left\langle \frac{1}{1 + e^{u_B - u_A + C}} \right\rangle_A \right]$$

$$C = \ln\left(\frac{Q_B}{Q_A}\right) + \ln \left[\frac{\langle f(u_A - u_B - C) \rangle_B}{\langle f(u_B - u_A + C) \rangle_A} \right]$$

$$f(x) = \frac{1}{1 + e^x}$$

If the ratio $\frac{\langle f(u_A - u_B - C) \rangle_B}{\langle f(u_B - u_A + C) \rangle_A} = 1$ then

$$\ln\left(\frac{Q_B}{Q_A}\right) = C$$

In practice one can calculate the ratios $\frac{\langle f(u_A - u_B - C) \rangle_B}{\langle f(u_B - u_A + C) \rangle_A}$ for a range of constant

C values and interpolate to find the value of C for which $\frac{\langle f(u_A - u_B - C) \rangle_B}{\langle f(u_B - u_A + C) \rangle_A} = 1$.

Note that

$$F_B / kT - F_A / kT = -\ln\left(\frac{Q_B}{Q_A}\right) = -C$$

so that the “C” value is the negative of the free energy difference.

It remains only to define the potential energy functions, U_A and U_B :

$$U_B = \frac{1}{2} \sum_{i,j=1,i \neq j}^{n-1} V(r_i, r_j) + \lambda \sum_{j=1, n-1} V(r_n, r_j); \lambda=1; n=\text{probe}$$

$$U_A = \frac{1}{2} \sum_{i,j=1,i \neq j}^{n-1} V(r_i, r_j) + \lambda \sum_{j=1, n-1} V(r_n, r_j); \lambda=10^{-m}; m > 0 \text{ and large}$$

These definitions (and the determination of $\ln\left(\frac{Q_B}{Q_A}\right)$ described above) call for the

Monte Carlo simulation of two systems (ensembles), A and B, each with n particles, and having the same volume. In B the particles all interact normally; in system A, one of the particles, the “probe”, has its interactions with the remaining particles turned off.

APPENDIX C.
PHYSICAL REVIEW LETTER

Scaled Vapor-to-Liquid Nucleation in a Lennard-Jones System

Barbara N. Hale and Mark Thomason

Physics Department, Missouri University of Science & Technology, Rolla, Missouri 65409

(Received 28 February 2010; published 21 July 2010)

Scaling of the homogenous vapor-to-liquid nucleation rate, J , is observed in a model Lennard-Jones (LJ) system. The model uses Monte Carlo simulation-generated small cluster growth to decay rate constant ratios and the kinetic steady-state nucleation rate formalism to determine J at four temperatures below the LJ critical temperature, T_c . When plotted vs the scaled supersaturation, $\ln S/[T_c/T - 1]^{3/2}$, the values of $\log J$ are found to collapse onto a single line. A similar scaling has been observed for the experimental nucleation rate data of water and toluene.

DOI: 10.1103/PhysRevLett.105.046101

PACS numbers: 82.60.Nh, 64.60.qe, 64.70.F-

Nucleation, the process by which embryos of the new phase are formed in a first-order phase transition, occurs in many everyday processes including droplet, aerosol and ice formation in the atmosphere, and in a host of industrial processes which break down complex vapors into their constituents or form alloys and crystalline structures from the liquid melt. Early in the 20th century a simple classical nucleation theory (CNT) [1,2] for predicting the rate of vapor-to-liquid nucleation was developed and used with considerable success to predict onset conditions. In its simplest form this model used the bulk liquid surface tension to determine the free energy of formation of the n -atom cluster from the vapor. However, when it became possible in the 1980s to measure nucleation rates as a function of temperature and for a range of supersaturations [3–8], it was found that the data displayed a temperature dependence which was not consistent with CNT. Several modifications to the classical nucleation theory were proposed in the early 1990s [9–11], the most important correcting the failure of the classical model's cluster free energy of formation to reduce to zero for the monomer (see, for example, Ref. [12]). In the last two decades extensive computer simulations have been undertaken to examine the microscopic nature of the small clusters and the nucleation process [13–40]. In many of these efforts, the temperature dependence of the nucleation rate was greatly improved.

In the late 1980s it was noted that the nucleation rate data exhibited a temperature scaling [41]. An example of this scaling phenomenon for toluene [42] is shown in Fig. 1. Figure 1(a) shows the nucleation rates for six temperatures plotted vs $\ln S$. When plotted in Fig. 1(b) vs $\ln S$ divided by $[T_c/T - 1]^{3/2}$, the temperature lines collapse onto a single line. The experimental supersaturation ratio, S , is given by the ratio of vapor pressures, $P/P_{\text{coexistence}}$, and T_c is the critical temperature for toluene. The collapse of the temperature lines indicates that J is not a function of independent variables, S and T , but of the scaling function, $\ln S_c \equiv \ln S/[T_c/T - 1]^{3/2}$. A similar scaling occurs for homogeneous nucleation rate data of

water [43,44]. The scaling function, $\ln S_c$, is similar to the *scaled supersaturation* used by Binder [45] near the critical point. Two features of the present scaling are different, however: the critical exponents are replaced by the volume to surface dimension ratio, $3/2$, and the temperature, T , of the rate data is far below the critical point, in the range of $T = 0.5T_c$.

It was known that the scaling function, $\ln S_c$, appears in the classical model cluster free energy of formation as $[16\pi\Omega^3/3]/[\ln S_c]^2$ if one assumes a surface tension for the small clusters of the form $\sigma'_0[T_c - T]$, where σ'_0 is a constant [41]. The Ω is the excess surface entropy per atom in units of k , the Boltzmann constant. However, the monomer flux factor in the classical nucleation rate (CNT) has an exponential temperature dependence of the form $P/P_c \approx \exp[-W_o(T_c/T - 1)]$ which destroys the scaling [42,44]. For argon $W_o \approx 5$. Some time ago it was proposed that this P/P_c temperature dependence is canceled by extra terms in the free energy of formation generated as one sums discretely over the smallest cluster free energy differences [42]. Unfortunately, how this arises in a first-

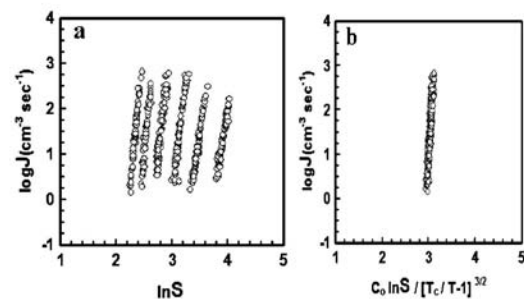


FIG. 1. The homogeneous vapor-to-liquid nucleation rate data of Schmitt *et al.* [4] for toluene at temperatures 265, 255, 245, 235, 225 and 215(± 1) K (left to right) plotted vs (a) $\ln S$, and (b) $\ln S/[T_c/T - 1]^{3/2}$. The normalization constant, C_0 , is $[T_c/235 - 1]^{3/2}$ and $T_c = 591.8$ K.

principles model of the vapor-to-liquid nucleation has remained largely unexplained.

The goal of the present work has been to generate the nucleation rate in a purely statistical mechanical treatment of small cluster partition functions to see if, indeed, one obtains scaling such as that shown in Fig. 1. In the present study we use the Monte Carlo method to generate growth to decay rate constant ratios for small clusters in a dilute Lennard-Jones (LJ) vapor. These rate constant ratios are then applied to a kinetic nucleation rate formalism [1,46]. The model system is a classical (full potential) Lennard-Jones dilute vapor with volume, V , composed of a non-interacting mixture of ideal gases with each n -atom cluster size constituting an ideal gas of N_n clusters. The separable classical Hamiltonian gives rise to the following law of mass action for ratios of cluster numbers in terms of classical canonical configurational integrals, Q_n : [15,20,34].

$$\frac{N_n}{N_{n-1}N_1} = \frac{Q_n}{Q_{n-1}Q_1} = \frac{\beta_{n-1}}{\alpha_n}. \quad (1)$$

At equilibrium in volume, V , detailed balance, $N_n\alpha_n = \beta_{n-1}N_{n-1}N_1$, is maintained. The kinetic nucleation rate formalism (which uses steady-state cluster size numbers, N_n^s) assumes $N_n^s\alpha_n = \beta_{n-1}N_{n-1}^sN_1^s$ with cluster growth and decay taking place *via* monomers with equilibrium rate constants, β_n and α_n . Using $\rho_1 = \frac{N_1}{V}$ at equilibrium and $J_1 = \beta_1[\rho_1 S]^2$, the steady-state nucleation rate, J , is given by

$$\frac{1}{J} = \frac{1}{J_1} + \sum_{n=2}^M \left[\beta_n (\rho_1 S)^2 \prod_{j=2}^n \frac{\beta_{j-1} N_1 S}{\alpha_j} \right]^{-1}, \quad (2)$$

where M is sufficiently large to ensure convergence. In the above expression $S \equiv N_1^s/N_1$ is the monomer supersaturation ratio. The equilibrium monomer flux on the n -cluster, $\beta_n \rho_1$, is approximated using $\beta_n = (v_{av}/4)4\pi r_n^2$ with $[n/\rho_{liq}] = [4/3]\pi r_n^3$, and $v_{av} = [8kT/\pi m]^{1/2}$ [2]. The monomer supersaturation can be put into the calculation of J after the equilibrium ($S = 1$) rate constant ratios are calculated. Bennett [47] Metropolis Monte Carlo simulations are used to determine the canonical partition function ratios in the following expressions: [15]

$$-\delta\Delta F_n \equiv \ln \left[\frac{\beta_{n-1} N_1}{\alpha_n} \right] \quad (3)$$

$$= \ln \left[\frac{Q_n \alpha'}{Q_{n-1} Q_1 (v_n/V) \rho_{liq}} \frac{\rho_1}{\rho_{liq}} \right] \quad (4)$$

$$= -\delta f_n - \ln \frac{\rho_{liq}}{\rho_1}. \quad (5)$$

The Q_n have been normalized with V^n so that $Q_1 = 1$. In these simulations, the assumed cluster definition is n -atoms constrained within a spherical volume, $v_n =$

$\alpha' n / \rho_{liq}$ [13]. The quantity $Q_1(v_n/V)$ is the scaled simulation volume accessible to the monomer. Formally, the expression in Eq. (4) is independent of the constant, α' . However, volumes too large or too small can place physically unrealistic constraints on the cluster definition. A working range is $5 < \alpha' < 8$ and in the present simulations $\alpha' = 7$ was used. Further details are in Refs. [15,20,34].

The law of mass action calls for the Q_n to sample the same configuration space as Q_{n-1} and Q_1 . Equation (4) provides a consistent method of scaling the two volumes, v_n and V , and a means of calculating ρ_1 for the full LJ potential. In the limit of infinite cluster sizes $-\delta f_n$ approaches $\ln \frac{\rho_{liq}}{\rho_1}$. Using the Dunikov [48] corresponding states procedure one can estimate the full LJ liquid number density, ρ_{liq} , at the low temperatures used here and from the intercept of $-\delta f_n$ at infinite n one can predict a vapor monomer number density, ρ_1 . The simulations are carried out at the LJ reduced temperatures, $Tk/\epsilon \equiv T^* = 0.335, 0.419, 0.503, \text{ and } 0.700$ corresponding roughly to experimental argon temperatures of $T = 40, 50, 60, \text{ and } 83.6$ K. The Bennett method [47] is well suited for calculating the free energy differences, δf_n , between the two cluster ensembles: one in which all n -atoms interact normally and a second in which the interaction of one of the atoms is turned off. The simulations produce a sequence of rate constant ratios, $\frac{\beta_{j-1}}{\alpha_j}$, for n ranging from 2 to 192. A reduced LJ critical temperature, $T_c^* = 1.313$ and reduced critical density, $\rho_c^* = 0.317$ are used in the analysis [49]. When the ratio T_c/T is used the reduced temperature notation, T^* , is omitted.

In Fig. 2(a) is a plot of the simulation results for $-\delta f_n$ vs $n^{-1/3}$. The intercepts (17.7, 12.9, 9.90 and 5.98) are extrapolated from a least squares fit to data for $n > 20$. In Fig. 2(b) is shown the $\delta\Delta F_n$ divided by $[T_c/T - 1]$ to

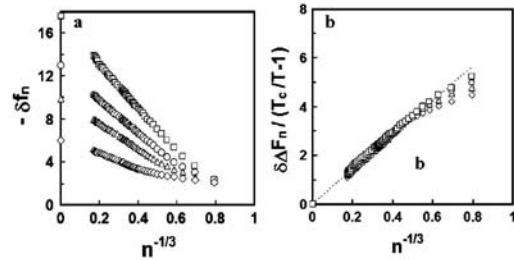


FIG. 2. (a) The $-\delta f_n$ small cluster free energy differences vs $n^{-1/3}$ for the small LJ clusters for $T^* = 0.335$ (\square), 0.419 (\circ), 0.503 (\triangle), and 0.700 (\diamond) corresponding roughly to argon LJ temperatures of 40, 50, 60, and 83.6 K. The n values range from 2 to 192. The intercepts (17.7, 12.9, 9.90, and 5.98) are extrapolated from a least squares fit to data for $n > 20$. (b) The $\delta\Delta F_n$ for the same n values divided by $[T_c/T - 1]$ where $T_c^* = 1.313$. The dotted line corresponds to a classical LJ model with excess surface entropy per atom, $\Omega = 2.19$.

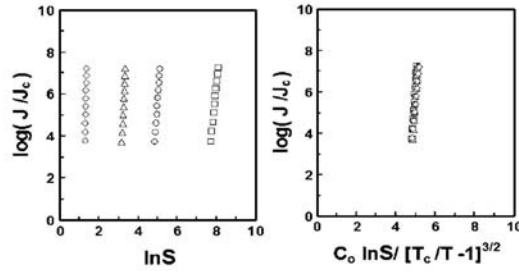


FIG. 3. (a) Nucleation rates, J/J_c , calculated from the Monte Carlo simulation-generated growth to decay rate constant ratios for the model Lennard-Jones system vs $\ln S$ at $T^* = 0.335$ (\square), 0.419 (\circ), 0.503 (\triangle), and 0.700 (\diamond) corresponding roughly to argon LJ temperatures of 40, 50, 60, and 83.6 K. $J_c = \rho_c [40/M]^{1/2} 10^{-22} \text{ s}^{-1}$. (b) Nucleation rates with the abscissa divided by $[T_c/T - 1]^{3/2}$; $C_o = [T_c/0.419 - 1]^{3/2}$.

demonstrate the scaled temperature dependence. An advantage of this analysis is that one can use the scaled quantities to predict nucleation rates at temperatures other than those of the simulation. The slope of the data in Fig. 2(b) is proportional to the effective surface tension of the small clusters [20]. The results are compared with a classical LJ model which corresponds to an excess surface entropy per atom of 2.19 k (the bulk liquid argon value can be estimated from $\sigma'_0/\rho_{\text{liq}}^{2/3} \cong 2.1k$) [41]. Merikanto *et al.* [37,38] have generated similar free energy differences using a modification of the original discrete summation formalism [15] at reduced temperatures of 0.4 and 0.7. However, they do not examine the scaling properties of the free energy differences and their results for $\delta\Delta F_n$ do not scale at small cluster sizes ($n < 20$) because of their cluster definition. Temperatures in the range of 40 K–60 K correspond roughly to those for which argon onset nucleation rates have been measured [50,51].

To demonstrate the scaling of the nucleation rates, values of S are chosen so that the nucleation rates range from 10^4 to $10^7 \text{ cm}^{-3} \text{ s}^{-1}$ for the LJ argon system. To preserve the corresponding states representation, J/J_c is plotted in Fig. 3 where $J_c = \rho_c [40/M]^{1/2} 10^{-22} \text{ s}^{-1}$, ρ_c is the LJ critical number density and M is the relative atomic mass. Figure 3(a) shows the calculated nucleation rates for the four temperatures plotted vs $\ln S$. When plotted in Fig. 3(b) vs the scaled supersaturation, $\ln S/[T_c/T - 1]^{3/2}$, the temperature lines collapse onto a single line. That is, scaling such as that shown in Fig. 1 for the experimental toluene data is observed in the LJ system modeled here. [52,53]

The source of the scaling in the present model is twofold: (1) the corresponding states $[T_c/T - 1]$ temperature dependence of the $\delta\Delta F_n$ [see Fig. 2(b)]; and (2) the discrete summation over the small cluster growth to decay rate constant ratio contributions prescribed by Eq. (2). The latter summation introduces terms (not present in the classical model) which cancel the temperature dependence of

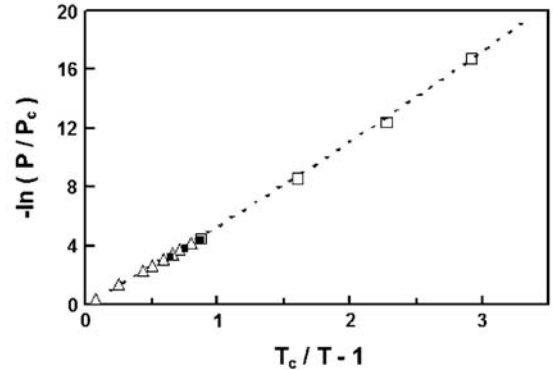


FIG. 4. Corresponding states comparison of $-\ln(P/P_c)$ from the LJ Bennett MC calculations of the present work (\square), the argon vapor pressure formula of Fladerer [50] and Iland and Strey [51] (dashed line), the argon experimental vapor pressure data [54] (\triangle), and the LJ Monte Carlo simulations of Chen *et al.* [31] (\blacksquare).

the monomer flux factor [42]. One notes that $1/J = \sum_{n=1}^M [1/J_n]$, where $J_n = \beta_n \rho_1^2 S^{n+1} \exp[-\sum_{j=2}^n \delta\Delta F_j]$.

In obtaining the scaling from the Monte Carlo results, a reliable ρ_1 for the full LJ potential from the intercepts of Fig. 2(a) is essential. Vapor pressures corresponding to these intercepts for the full LJ potential (assuming the vapor consists of monomers only) are shown in Fig. 4. In this figure a Dunikov [48] corresponding states comparison is made with the extrapolated argon vapor pressure formula used by Fladerer [50] and Iland *et al.* [51], experimental data for argon vapor pressure at higher temperatures [54] and the LJ Monte Carlo results of Chen *et al.* [31]. One of the challenges in predicting nucleation rates from potential models is obtaining reliable monomer vapor number densities at low temperatures, where most of the nucleation rate data exist. Equally troublesome for experimentalists can be extrapolating measured vapor pressure data far below the freezing point. A plot such as Fig. 4 can be helpful in evaluating the validity of both approximations.

As far as we are aware, this is the first purely simulation based demonstration of scaling in a model dilute vapor-to-liquid nucleation process. Future applications of this scaling to atmospheric nucleation via molecular-sized heterogeneous sites should be of particular importance [55]. In this case, the scaled supersaturation dependence can survive in the larger growing clusters while the molecularly sized heterogeneity produces an effective reduced excess molecular or atomic entropy at the cluster surface (Ω parameter). This effect is well known in the nucleation of water (and ice) on surfaces, where the contact angle plays the role of a reduced Ω [56]. Applications of the scaling to homogeneous binary vapor-to-liquid nucleation [57] and to ice formation on AgI impurities in supercooled water [58] suggest that this scaling has a wider application. Toward

this end, our hope is that the present simulation results will motivate a more fundamental analysis of scaling in nucleation phenomena, beyond the scope of classical model concepts.

The authors acknowledge G. Wilemski for helpful discussions, J. Kiefer for work done on some of the original LJ cluster simulations and J. Kassner for focusing interest on the scaling approach.

-
- [1] R. Becker and W. Döring, *Ann. Phys. (Leipzig)* **24**, 719 (1935) [online version **416**, 719 (2006)].
- [2] M. Volmer and A. Weber, *Z. Phys. Chem.* **119**, 277 (1926).
- [3] R. C. Miller, R. J. Anderson, J. L. Kassner, and D. E. Hagen, *J. Chem. Phys.* **78**, 3204 (1983).
- [4] J. L. Schmitt, R. A. Zalabsky, and G. W. Adams, *J. Chem. Phys.* **79**, 4496 (1983).
- [5] P. E. Wagner and R. Strey, *J. Chem. Phys.* **80**, 5266 (1984).
- [6] G. W. Adams, J. L. Schmitt, and R. A. Zalabsky, *J. Chem. Phys.* **81**, 5074 (1984).
- [7] R. Strey, P. E. Wagner, and T. Schmeling, *J. Chem. Phys.* **84**, 2325 (1986).
- [8] C. Hung, M. J. Krasnopolcer, and J. L. Katz, *J. Chem. Phys.* **90**, 1856 (1989).
- [9] G. Wilemski, *J. Chem. Phys.* **103**, 1119 (1995) gives a comprehensive discussion of the early developments of vapor phase nucleation theory.
- [10] D. W. Oxtoby, *J. Phys. Condens. Matter* **4**, 7627 (1992).
- [11] A. Dillmann, Ph.D. thesis, Göttingen, 1989; A. Dillmann and G. E. Meier, *J. Chem. Phys.* **94**, 3872 (1991).
- [12] S. L. Girshick and C. Chiu, *J. Chem. Phys.* **93**, 1273 (1990).
- [13] J. K. Lee, J. A. Barker, and F. F. Abraham, *J. Chem. Phys.* **58**, 3166 (1973).
- [14] N. Garcia and J. M. S. Torroja, *Phys. Rev. Lett.* **47**, 186 (1981).
- [15] B. N. Hale and R. C. Ward, *J. Stat. Phys.* **28**, 487 (1982).
- [16] X. C. Zeng and D. W. Oxtoby, *J. Chem. Phys.* **95**, 5940 (1991).
- [17] X. C. Zeng and D. W. Oxtoby, *J. Chem. Phys.* **94**, 4472 (1991).
- [18] C. Weakliem and H. Reiss, *J. Chem. Phys.* **99**, 5374 (1993).
- [19] D. I. Zhukhovitskii, *J. Chem. Phys.* **103**, 9401 (1995).
- [20] B. Hale, *Aust. J. Phys.* **49**, 425 (1996).
- [21] I. Kusaka, Z. G. Wang, and J. H. Seinfeld, *J. Chem. Phys.* **108**, 3416 (1998).
- [22] K. Yasuoka and M. Matsumoto, *J. Chem. Phys.* **109**, 8451 (1998).
- [23] K. Yasuoka and M. Matsumoto, *J. Chem. Phys.* **109**, 8463 (1998).
- [24] P. R. ten Wolde and D. Frenkel, *J. Chem. Phys.* **109**, 9901 (1998).
- [25] P. R. ten Wolde, M. J. Ruiz-Montero and D. Frenkel, *J. Chem. Phys.* **110**, 1591 (1999).
- [26] K. J. Oh and X. C. Zeng, *J. Chem. Phys.* **110**, 4471 (1999); *ibid.* **114**, 2681 (2001).
- [27] B. Senger, P. Schaaf, D. S. Corti, R. Bowles, D. Pointu, J. C. Voegel, and H. Reiss, *J. Chem. Phys.* **110**, 6438 (1999).
- [28] S. M. Kathmann, G. K. Schenter, and B. C. Garrett, *J. Chem. Phys.* **111**, 4688 (1999).
- [29] S. Wonzak, R. Strey, and D. Stauffer, *J. Chem. Phys.* **113**, 1976 (2000).
- [30] K. J. Oh and X. C. Zeng, *J. Chem. Phys.* **112**, 294 (2000).
- [31] B. Chen, J. I. Siepmann, K. J. Oh, and M. L. Klein, *J. Chem. Phys.* **115**, 10903 (2001).
- [32] B. N. Hale and S. M. Kathmann, *J. Phys. Chem. B* **105**, 11 719 (2001).
- [33] I. Kusaka, *J. Chem. Phys.* **119**, 3820 (2003).
- [34] B. N. Hale and D. J. DiMatteo, *J. Phys. Chem. B* **108**, 19 780 (2004).
- [35] J. Merikanto, E. Zapadinsky, and H. Vehkamäki, *J. Chem. Phys.* **121**, 914 (2004).
- [36] J. Wedekind, J. Wölk, D. Reguera, and R. Strey, *J. Chem. Phys.* **127**, 154515 (2007).
- [37] A. Lauri, J. Merikanto, E. Zapadinsky, and H. Vehkamäki, *Atmos. Res.* **82**, 489 (2006).
- [38] J. Merikanto, E. Zapadinsky, A. Lauri, and H. Vehkamäki, *Phys. Rev. Lett.* **98**, 145702 (2007).
- [39] M. Horsch, J. Vrabec, and H. Hasse, *Phys. Rev. E* **78**, 011603 (2008).
- [40] M. Schrader, P. Virnau, and K. Binder, *Phys. Rev. E* **79**, 061104 (2009).
- [41] B. N. Hale, *Phys. Rev. A* **33**, 4156 (1986).
- [42] B. N. Hale, *Metall. Trans. A* **23**, 1863 (1992).
- [43] J. Wölk and R. Strey, *J. Phys. Chem. B* **105**, 11683 (2001).
- [44] B. N. Hale, *J. Chem. Phys.* **122**, 204509 (2005).
- [45] K. Binder, *J. Phys. (Paris), Colloq.* **41**, C4-51 (1980); K. Binder and D. Stauffer, *Adv. Phys.* **25**, 343 (1976).
- [46] J. L. Katz and F. Spaepen, *Philos. Mag. B* **37**, 137 (1978).
- [47] C. H. Bennett, *J. Comput. Phys.* **22**, 245 (1976).
- [48] D. O. Dunikov, S. P. Mallyshenko, and V. V. Zhakhovskii, *J. Chem. Phys.* **115**, 6623 (2001).
- [49] J. Perez-Pellitero, P. Ungerer, G. Orkoulas, and A. D. Mackie, *J. Chem. Phys.* **125**, 054515 (2006).
- [50] A. Fladerer and R. Strey, *J. Chem. Phys.* **124**, 164710 (2006).
- [51] K. Iland, J. Wölk, and R. Strey, *J. Chem. Phys.* **127**, 154506 (2007).
- [52] The experimental argon “onset nucleation rates” of Ref. [51] also scale; the S , however, are a factor of 5 smaller than those in Fig. 3 and the excess surface entropy corresponds to $\Omega = 1.5$. See Fig. 6 in Ref. [53].
- [53] S. Sinha, A. Bhabhe, H. Laksmono, J. Wölk, R. Strey, and B. Wyslouzil, *J. Chem. Phys.* **132**, 064304 (2010).
- [54] R. Gilgen, R. Kleinrahm, and W. Wagner, *J. Chem. Thermodyn.* **26**, 399 (1994).
- [55] P. M. Winkler, G. Steiner, A. Vrtala, H. Vehkamäki, M. Noppel, K. E. J. Lehtinen, G. P. Reischl, P. E. Wagner, and M. Kumala, *Science* **319**, 1374 (2008).
- [56] N. H. Fletcher, *The Physics of Rainclouds* (Cambridge, London, 1969), p. 52; *The Chemical Physics of Ice* (Cambridge, London, 1970), p. 98.
- [57] B. N. Hale and G. Wilemski, *Chem. Phys. Lett.* **305**, 263 (1999).
- [58] B. Hale, *Lect. Notes Phys.* **309**, 321 (1988).

APPENDIX D.
THE FORTRAN CODE

PROGRAM mainprg

```

c*****
c***** written January 1997 by B. Hale to check Shawn's program **
c***** 1. pwsuab written and checked 1-27-97 (full sum) **
c***** 2. moveab (2-7-97) **
c***** 3. program finished; including calculation of C (2-12-97) **
c***** 4. added kall = 1,0 (2-19-97) **
c***** 5. added polarization from SO4(3-9-97);from H+ (3-11-97) **
c***** added falph to account for istop < 7 (3-24-97) **
c***** (checked with Shawn's, istop=5,C2a2wp) (3-24-97) **
c***** 6. added qf to reduce charge and vary C **
c***** 7. made atomic array parameter: MAT=#atoms 4-17-97 **
c***** MAT=50 takes only 2.8MB memory **
c***** 8. added Rsqa calculation in program 4-20-97 **
c***** 9. added subroutine to generate Rasmol coords. 5-2-97 **
c***** 10. added It_max input, call rasmolh each print 5-10-97 **
c***** 11. added Iprobe(i): tag probes 5-11-97 **
c***** 12. revised pwsuab to use delU Iprobe switches 5-12-97 **
c***** 13. added cfg_out subroutine to write configs. 6-02-97 **
c***** 14. change print UB_del to ave. over cluster 6-04-97 **
c***** added qw factor to scale H2O charges **
c***** 15. modified for continuation runs 6-10-97 **
c***** (format read/write; no pwsuab calls) 6-12-97 **
c***** converted seeds to integers; 6-13-97 **
c***** must call cmass(iflag=0) for mstart>0 **
c***** 16. infout corrected for mm = 1 case 6-18-97 **
c***** 17. increased number of probes in B to 4*m 6-25-97 **
c***** 18. angle subrout: checks/fixes HOH angles 6-30-97 **
c***** fixed ra(i1,j), format iacc_sum 7-14-97 **
c***** 19. add nxang toggle to implement angle 7-15-97 **
c***** 20. add movAhh toggle: reset H in H2SO4 A probe 7-16-97 **
c***** 21. let A ensemble probes have large disp, beta 9-16-97 **
c***** (this started with binMC_MPxangR.f program) **
c***** method: create dis(na) = 1 or 10 **
c***** 22. added <FAint>,<FBint>,<Uint>,R in Rasmol 10-15-97 **
c***** 23. simulation volume parameter, alp added 10-15-97 **
c***** 24. increased # probes in B: jup 12-19-97 **
c***** 25. added T, lamb and jup to last line of print 03-28-98 **
c***** The run file is called "... disp6c" 03-28-98 **
c***** 26. revised to run all probes but calculate 05-29-98 **
c***** <FB> only every MB steps **
c***** 27. added binning of dUA and dUB 06-04-98 **
c***** 28. make lambda =lmb variable for B ensemble 07-07-98 **
c***** 29. adjusted C range for C1 calculation 07-09-98 **
c*****

```

```

c***** convert to run water clusters only **
c***** 1. if nac = 0, k = 1, m = number of waters 9-12-97 **
c***** 2. fixed probe tags for water only case 9-17-97 **
c*****
c*****
c***** convert to run water clusters only TIP4P **
c***** 1. started 10-24-98 **
c***** 2. corrected 11-11-98 (alp) **
c***** 3. added test using LJ argon potentials 11-12-98 **
c***** ( read in LJ= (0,1) for (TIP4P,LJ) **
c***** ( read in densLJ in g/cm**3) **
c***** ( cmas determined from Oxygen (Ar) only) **
c***** 4. corrected cmas subr for LJ case: **
c***** reject only when O (the LJ atom) is out of sphere **
c***** 11-18-06 LJ a, LJ b were setting all irsa, irsa =0! **
c*****
c*****
c***** 01-30-07: convert from exp. to LJ potential properties **
c***** 1. print out more digits in Cplt **
c***** 2. convert liquid density to LJ pot, with T dep. **
c***** 3. corrected M in densLJ, increase MAT MAB 3-29-07 **
c***** 4. corrected format Cf; added densLJ print 5-4-07 **
c***** 5. stream line pwsun for LJ simulations 5-28-07 **
c***** 6. checked cmas calc for LJ: OK (8-1-07) **
c***** 7. fixed atom out: only if "O atoms" out 8-1-07 **
c***** 8. fixed format 1009 (3 digits imol) 10-21-07 **
c***** 9. calculate c range 11-1-07 **
c***** 10. increase MAT to 600 for n up to 200 **
c***** 11. print out irsa, irsb for MOLECULE OUT 12-2-07 **
c*****
c*****
implicit real*8 (a-h,o-z)
real*8 lamb,lmb,kb,mass
character*5 cxaxw
parameter(MAT=600)
parameter(MAM=600)
parameter(MAB=600)
c character*4 runx
c character*1 S
c character*2 O1
c character*2 O2
c character*2 H1
c character*1 O
c character*1 H
c*****7**0***5***0***5***0***5***0***5***0***5***0***5***0***5***0***5***0***5***0**

```

```

COMMON/R/ ra(MAT,3),rb(MAT,3),ia(MAT),ib(MAT)
1      ,imola(MAT),imolb(MAT),rcma(3),rcmb(3)
COMMON/seeds/iseeda(7),iseedb(7)
c COMMON/CONST/ kb,coul,pi,mi,ic,kall,alpha,istop,qf,qw,alp
COMMON/CONST/ kb,coul,pi,alpha,qf,qw,alp,ic,kall,istop,mi
COMMON/STEP/mstart,mmax,mprint,MBdo,MBstep
c COMMON/km/k,m,na,nac,nwa,lamb,lmb,disp,beta,dd,Rsqa,jup,disa(MAT)
c 1 ,jlo,jhi,jxtot,LJ
COMMON/km/lamb,lmb,disp,beta,dd,Rsqa,disa(MAT),k,m,na,nac,nwa,jup,
1 jlo,jhi,jxtot,LJ
COMMON/SIG/is(MAT),sig(8),eps(8),q(8),mass(8),Iprobe(MAM,MAT)

COMMON/UAUB/ UA_aa,UB_aa,UAdel_aa,UBdel_aa,
1      UA_aw,UB_aw,UAdel_aw,UBdel_aw,
2      UA_ww,UB_ww,UAdel_ww,UBdel_ww,
6      UAp_aa,UAp_aw,UBp_aa,UBp_aw,UA_px,UB_px,
3      UA, UB, UAdel, UBdel,UBdelf(MAM),
4      UAL,UBL,UAdelL,UBdelL,UBdelLf(MAM),
5      UA_pol,UB_pol,UAdel_p,UBdel_p
6      ,binUA(MAB),binUB(MAB)

COMMON/UAUBuff/ bA_aa,bB_aa,bAdel_aa,bBdel_aa,
1      bA_aw,bB_aw,bAdel_aw,bBdel_aw,
2      bA_ww,bB_ww,bAdel_ww,bBdel_ww,
6      bAp_aa,bAp_aw,bBp_aa,bBp_aw,bA_px,bB_px,
3      bA, bB, bAdel, bBdel, bBdelf(MAM),
4      bAL,bBL,bAdelL,bBdelL,bBdelLf(MAM),
5      bApol,bBpol,bAdelp,bBdelp

COMMON/FMASS/fmass(MAT),fmasstot,fmasstota

COMMON/COUNTS/ icounta(MAT),icountb(MAT),It_max
COMMON/SUMS/ UAL_sum,UB_sum,UAdL_sum,UBdL_sum,UBdf_sum
COMMON/FAFB/ FA(20),FB(MAM,20),c(20),
1      FA_sum(20),FB_sum(MAM,20),
2      FA_av(20),FB_av(MAM,20),MB
COMMON/DATA/cfin,cplot,cav,cp_ave,UAL_av,UB_av,UAdel_av,UBdel_av
1 ,UBdf_av
c-----
open(55,file='nkm.name',status='old')
read(55,1111) cxaxw
1111 format(A)
open(60,file=cxaxw/'_MCkp.cfg',status='unknown',access='append')
open(10,file=cxaxw/'_MCkp.run',status='unknown',access='append')

```

```

open(20,file=cxaxw//'_MCkp.inp',status='unknown')
open(30,file=cxaxw//'_MCkp.out',status='unknown',access='append')
open(40,file=cxaxw//'_A.pdb',status='unknown',access='append')
open(50,file=cxaxw//'_B.pdb',status='unknown',access='append')

c  open(10,file='c2a2w.cfg',status='unknown')
c  open(20,file='c2a2w.inp',status='unknown')
c  open(30,file='c2a2w.out',status='unknown',access='append')
c  open(40,file='c2a2w.sav',status='unknown',access='append')
c*****7**0***5****0****5****0****5****0****5****0****5****0****5****0****5****0**
c  s = secnds(0.0)

WRITE(30,1003)
write(30,*) 'Program: bin3MC_98a.f ver6 July 9 1998'
write(30,*) 'Program: bin_wat_ver6.f  October 24, 1998'
write(30,*) 'Program: bin_wat_TIP4P.f  October 24, 1998'
write(30,*) 'Program: bin_wat_TIP4P.f  mod. November 11, 1998'
write(30,*) 'Program: bin_argon.f  mod. November 12, 1998'
write(30,*) 'Program: bin_argon.f  mod. November 18, 2006'
write(30,*) 'Program: bin_argon.f  mod. March 23, 2007'
write(30,*) 'Program: bin_argon4.f  mod. May 5, 2007'
write(30,*) 'Program: bin_argon4.f  mod. May 6, 2007'
write(30,*) 'Program: bin_argon4.f  mod. pwsun May 28, 2007'
write(30,*) 'Program: bin_argon6e.f pwsun;atm out Aug 1, 2007'
write(30,*) 'Program: bin_argon6f.pwsun.f Oct 21, 2007'
write(30,*) 'Program: bin_argon6g.pwsun.f Nov 1, 2007'
write(30,*) 'Program: bin_argon6h.pwsun.f Nov 6, 2007'
write(30,*) 'Program: bin_argon6h2.pwsun.f Dec 2, 2007'
c  write(30,*) '  minor mod.  March  18, 1998'
write(30,*) 'modified version of binMC_disp.f'

c*****
kb= 1.987905D-03
pi=3.141592653589793D0
mi=1686754213
ic=453806245
coul= 331.5793968d0
fNa=6.0221367d23

sigLJ = 3.400000d0
epsLJ = 119.40000d0

mass(1)= 32.066d0
mass(2)= 15.9994d0
mass(3)= mass(2)

```

```

mass(4)= 1.00794d0
mass(5)= mass(2)
mass(6)= mass(4)
mass(7)= mass(2)
mass(8)= mass(4)

if(LJ.ne.1) call test4P(v,0)
if(LJ.ne.1) write(30,*)'test of TIP4P dimer: -6.24 = ',v
c*****
c*****7**0***5*****0***5*****0***5*****0***5*****0***5*****0***5*****0***5*****0***5*****0**
c*****
c  m * 7 acid atoms listed first
c  km * 3 water atoms listed last
c  last acid and last k water atoms are probes
c***** ia, ib atom codes for charges, sig, eps *****
c  1 ==> S      in H2SO4
c  2,3 ==> O1, O2  in H2SO4
c  4 ==> H      in H2SO4
c  5 ==> O on water when interacting with H2SO4
c  6 ==> H on water when interacting with H2SO4
c  7,8 ==> O ,H on water when interacting with water
c*****7**0***5*****0***5*****0***5*****0***5*****0***5*****0***5*****0***5*****0**
  nxang=0
  read(20,*) m,k,nac,nwa,nkind,nxang,movAhh,qS,alp,LJ,densLJ
  read(20,*) T,mprint,mmax,mstart,It_max,dispaa,jup,MB
  read(20,*) disp,beta,lamb,lmb,dd,kall,alpha,istop,qf,qw
  read(20,*) sig
  read(20,*) eps
  read(20,*) q

if(LJ.eq.1)mass(4)=0.0d0
if(LJ.eq.1)mass(2)=39.948d0

c  read(20,*) (iseeda(i), i=1,5)
c  read(20,*) (iseeda(i), i=6,7)
c  read(20,*) (iseedb(i), i=1,5)
c  read(20,*) (iseedb(i), i=6,7)
c***** check array sizes *****

MATCHK = nac*m + 3*k*m
if(MAT.lt.MATCHK) then
write(30,*)'STOP ARRAY SIZE WRONG!!'
go to 900
else
endif

```

```

If(nac.eq.0)write(30,*)'NUMBER OF ACID ATOMS = NAC = 0'
if(nac.eq.0)write(30,*)'m = # waters = ',m, 'k = ',k

c***** set jlo and jhi to m if lmb < 1.0 *****
c***** set MB = 1          if lmb < 1.0 *****

      if(lmb.ne.1.0d0) then
        jlo=m
        jhi=m
        MB=1

      else
        jlo=1
        jhi=jup
      endif

      jxtot=jhi-jlo+1

c***** determine the radius of constraint *****

      if(k.eq.1) Rsqaold=6.42213586884774d0**2
      if(k.eq.3) Rsqaold=7.29914179591958d0**2

      ratio = dfloat(k+1)/( dfloat(k)*18.d0+ 98.1d0 )
c    ratio = dfloat(k+1)/( dfloat(k)*18.1d0+ 98.1d0 )
      ratio = ratio*fNa*1.0d-24
      conc = 100.d0*dfloat(m)*98.1d0
1    /( dfloat(m)*98.1d0 +dfloat(k*m)*18.0d0)
      dens = 1.d0/alp*(9.278d-03*conc+0.9552d0)*ratio
      volume = dfloat(m+k*m)/dens

c-----correct density for pure water clusters -----
      if(nac.eq.0) then
c    dens = 1.0d0
c-----more accurate water density 10-98 -----
c----- expression from Strey (Dillmann's) -----
c----- use dens = 1.0d0 to check program -----
c-----

      if(LJ.ne.1) then
        dens =.99984d0+(8.6d-5)*(T-273.15)-(1.08d-5)*(T-273.15)**2
        dens = 1.0d0

```



```

dens = 1.0d0/alp*dens/18.0d0*fna/1.0d24

else
c-----
c----- old liquid density: dens = 1.34
c----- change liquid density from experimental value for argon
c----- to LJ potential result: densLJ
c----- read densLJ from data file          ---
c-----
c----- changed 3-12-07 to use LJ full potential liqui density ---
c----- given by Dunikov JCP 115, 6623 (2001) fig 5          ---
c----- in reduced units          ---
c----- nc = 0.310, Tc = 1.31 for full LJ potential          ---
c-----
c----- below: set densLj = 1.57d0 to get old densLj results -----
c-----   deleted old density option 6-20-07          -----
c-----
-----
fmassargon=39.948

dred=fmassargon/(fna*(sigLJ*1.0d-8)**3)
Tred=T/epsLJ
densLJ=0.310*dred*(1.75d0+2.1d0*(1.0d0-(Tred/1.31d0)))

TcLJ=1.31d0*epsLJ

dens = 1.0d0/alp*densLJ/fmassargon*fna/1.0d24
endif

volume=dfloat(k*m)/dens
else
endif

Rmax = (3.0d0*volume/(4.0d0*pi))**(1.0d0/3.0d0)
Rmaxold = 0d0
c   if(k.eq.3) Rmaxold=dsqrt(Rsqaold)
c   if(k.eq.1) Rmaxold=dsqrt(Rsqaold)
Rsqa = Rmax*Rmax

c*****

write(30,*) 'Rmax = ',Rmax,' Angstroms'

```

```

c  write(30,*) 'Rmaxold, Rmax = ',Rmaxold,Rmax,' Angstroms'
write(30,*) 'ratio,conc,density = ',ratio,conc,dens
write(30,*) '*****'
write(30,*) 'sigLJ = ',sigLJ,'epsLJ = ',epsLJ
write(30,*) 'densLJ full pot = ', densLJ, 'g/cm**3'
write(30,*) 'Tc LJ full pot = ', TcLJ
write(30,*) 'cluster volume (A**3) =',volume,' alp = ',alp
write(30,*) 'PROBE NUMBER: jup = ',jup,' and lmb = ',lmb
write(30,*) 'FB calculated every ',MB,' steps'
c***** check MB value *****

      MBchk= dint( mod( dfloat(mprint),dfloat(MB) ) )
      if(MBchk.ne.0) then
        write(30,*) 'STOP: MB value wrong! MBchk =',MBchk
        go to 900
      else
      endif
c*****
      WRITE(30,1003)

      na = k*m*nwa+m*nac

c
c  jup = m*m
c  if(m.eq.1) jup = 1

c  if(m.eq.2) then
c  if(k.eq.1) jup = 4
c  else
c  endif

      if(k.eq.1) jupx=m*m
      if(k.gt.1) jupx=m*k*m
      if(jupx.gt.MAM)jupx=MAM
      if(jup.gt.jupx) jup=jupx

      write(30,*) 'PROBE NUMBER: jup = ',jup
c  if(m.eq.3) then

c  if(k.eq.1) jup = 9
c  else
c  endif

      if(jup.gt.MAM) then
        write(30,*) 'JUP greater than array size, MAM!'
        go to 900

```

```

else
endif
mstop = mstart+mmax
write(30,1003)

if(nac.ne.0)then
If(mstart.eq.0) write(30,1000)
if(mstart.gt.1) write(30,1006)
else
if(LJ.ne.1) write(30,*)'WATER CLUSTER SIMULATION ONLY'
if(LJ.eq.1) write(30,*)'ARGON CLUSTER SIMULATION ONLY'
endif

write(30,1005)
WRITE(30,1003)
write(30,1001)m,k,na,T,lamb,disp,beta,dd,kall,alpha,istop
write(30,*)'lambda_b = lmb = ',lmb
write(30,*) 'Acid Q factor: qf = ',qf,' Water Q fact: qw = ',qw
write(30,*) 'nxang = ',nxang,' movAhh = ',movAhh
write(30,*) 'jup = ',jup,' = Number of Probe Combinations '
write(30,1005)
if(LJ.eq.1) write(30,*)' Argon density = ',densLJ,' gm/cm**3'
write(30,1013) mstart,mstop
WRITE(30,1003)
c*****
c***** Create disa(MAT) 9-16-97 *****
c*****
write(30,*)'program allows large disp for A probes'
write(30,*)'dispaa = ',dispaa
write(30,*) 'disa array:'

do i=1,MAT
disa(i)=1.0d0
end do

c.....
if(nac.ne.0) then

c..... below if nac.ne.0 .....

do im=1,m
do ii=1,nac
i = (im-1)*nac+ii
if(im.eq.m)disa(i)=1.0d0*dispaa
end do
end do

```

```

do im = 1,m
do ii=1,3*k
i = m*nac + (im-1)*3*k +ii
if(im.eq.m)disa(i)=1.0d0*dispaa
end do
end do
c..... else for nac.ne.0 condition .....
  else
c.....special part for pure water .....

do ii=1,3
i = (m-1)*3 + ii
disa(i)=1.0d0*dispaa
end do
c.....
endif
c..... endif for nac.ne.0 condition .....

if(nac.ne.0) then
do im=1,m
i1=(im-1)*nac+1
i2=(im-1)*nac+nac
write(30,1018)(disa(i),i=i1,i2)
end do
else
endif

do im=1,m
i1= m*nac + (im-1)*k*3+1
i2= m*nac + im*k*3
write(30,1018)(disa(i),i=i1,i2)
end do
WRITE(30,1003)

c*****
c***** Tag the probes 5-10-97 *****
c*****
c*****7**0**5**0**5**0**5**0**5**0**5**0**5**0**5**0**5**0**

do i=1,na
do jx=1,MAM
Iprobe(jx,i) =0
end do

```

```

end do

if(nac.ne.0)then

do im = 1,m

iim = (im-1)*nac
do i = iim+1, iim+nac
c-----
c----- create 4*m arrays:
c----- 1: first 7 acid atoms are probes
c----- 2: 2nd 7 acid atoms "
c----- 3: 3rd 7 acid atoms
c----- ..
c----- m: mth 7 acid atoms are probes
c-----
c----- REPEAT 4 times
c----- revised to m repeats 12-17-97
c-----
c Iprobe(im,i)=1
c ims1=im+m
c Iprobe(ims1,i)=1
c ims2=im+2*m
c Iprobe(ims2,i)=1
c ims3=im+3*m
c Iprobe(ims3,i)=1
c----- increase to m times
do mir=1,m
Iprobe(im+(mir-1)*m,i) = 1
end do

end do

end do

write(30,*)'Acid Iprobe values: m,k = ',m,k

c imup = m*2
imup = jup

do im=1,jup
jx=mod(im,m)
jy=im/m+ 1
if(jx.eq.1)then
write(30,*)** next set ('jy,') of m probes *****'

```

```

else
continue
endif
write(30,*) (Iprobe(im,iy), iy=1,m*nac)
end do
c..... else and endif for nac.ne.0 condition .....
else
endif

c***** Tag the water probes *****
c-----
c---- 1. tag first k waters
c---- 2. tag 2nd k waters
c---- ..
c---- m. tag mth k waters
c-----
c---- 1. shift set 1. by k=3
c---- 2. shift set 2. by k=3
c---- ..
c---- m. shift set m by k=3
c-----
c---- 1. shift set 1 by 2*k=6
c---- 2. shift set 2 by 2*k=6
c---- ..
c---- m. shift set m by 2*k=6
c-----
c---- REPEAT, shifting by 3*k=9
c-----
      nwater=3*k*m

      do im=1,m

c----- the following runs over k time 3 atoms
c----- 3k water atoms are designated

      do iik= 1,3*k

c----- move the 3k atom sets by 3k for each im value

      i = (im-1)*3*k +iik

c-----
c----- new routine to generate tags -----
c----- generates m*m probe sets -----

```

```

c----- jm denotes the probe set index -----
c----- picks 1 of m*m acid choices -----
c----- jmir displaces water indices by -----
c----- one water so that starting -----
c----- point of subsequent sets of -----
c----- 3k water atom probes -----
c----- are different -----
c----- ji keeps atom index -----
c----- to be in jm probe set -----
c-----
c----- corrected for pure water case -----
c----- generate only one set when -----
c----- nac = 0 .... no acids. -----
c-----

```

```

I2=m
if(nac.eq.0) I2=1

do mir=1,I2
jm=im+(mir-1)*m
jmir= i+ 3*(mir-1)
c   jii=jmod(jmir,nwater)
   jii=mod(jmir,nwater)

if(jii.eq.0) jii=nwater
ji = m*nac + jii
Iprobe(jm,ji)=1
end do

```

```

c-----old routine to generate tags -----
c   Iprobe(im,i)=1
c   j = i + 3
c   ji=j
c   if(j.gt.na) ji = m*7+ j-na
c   jm=im+m
c   Iprobe(jm,ji)=1
c
c   j = i + 2*3
c   ji=j
c   if(j.gt.na) ji = m*7+ j-na
c   jm=im+m+m
c   Iprobe(jm,ji)=1
c   j = i + 3*3
c   ji=j
c   if(j.gt.na) ji = m*7+ j-na

```

```

c   jm=im+m+m+m
c   Iprobe(jm,ji)=1
c-----
  end do

  end do

  write(30,*)'Water Iprobe values: m,k = ',m,k

  do jm=1,jup

    iy1=nac*m+1
    iy2=nac*m+m*k*3
c   jx=jmod(jm,m)
    jx=mod(jm,m)

    jy=jm/m+ 1
    if(jx.eq.1)then
      write(30,*)** next set ('jy,') of m probes **
    else
      continue
    endif

    write(30,*) (Iprobe(jm,iy), iy=iy1,iy2)
  end do

  WRITE(30,1003)

  write(30,*)' All Iprobe values: m,k = ',m,k,'jup = ',jup
  do im=1,jup

c   jx=jmod(im,m)
    jx=mod(im,m)

    jy=im/m+ 1
    if(jx.eq.1)then
      write(30,*)** next set ('jy,') of m probes **
    else
      continue
    endif
    write(30,*) (Iprobe(im,iy), iy=1,na)
  end do

  WRITE(30,1003)

```



```

c*****7**0***5****0****5****0****5****0****5****0****5****0****5****0****5****0**

c*****
c***** FORMATS *****
c*****
1000 format('MONTE CARLO FOR WATER-SULFURIC ACID (initial run)',/)
1001 format(1x,'m =',I3,' k =',I3,' na =',I5,' T =',f5.0/,
& ' lambda =',d22.16,' disp =',f7.4,' beta =',f7.4,
& ' dd =',f7.4,' kall =',I1,' alpha =',f6.3,' istop =',I2)

1002 format(1x,'sig(8): ',8(F5.2,1x))
1003 format('*****
1 *****')
1004 format(1x,'eps(8): ',8(F5.2,1x))
1005 format('ORDER: m*7 acid atoms; km*3 water atoms')
1006 format('MONTE CARLO FOR WATER-SULFURIC ACID,(continued run'//)
1007 format(1x,' q(8): ',8(F11.8,1x))
1008 format(1x,'seeds(7): ',8(I10,1x))
c1009 format(3(d22.16,1x),2(I2,1x))
1009 format(3(d24.16,1x),I1,1x,I3)
c in main
1010 format(1x,'cm =',3(d24.16,1x))
c1011 format below
c1012 format below
1013 format(1x,'mstart =',i10,' mstop =',i10)
c1014 see format below
1015 format(5(d22.16,1x))
1016 format(5(f12.1,1x))
1017 format(5(i10,1x))
1018 format(12(f6.1,1x))
c*****

write(30,1002) sig
write(30,1004) eps
write(30,1007) q
c----- redistribute charge on S and H+, H+ -----
qSsav=q(1)
qHsav=q(4)

q(1)=qS
q(4)=(qSsav-qS)/2.0d0+qhsav
write(30,*) 'Reduce qS to:',q(1)
write(30,*) 'Increase qH to:',q(4)

```

```

do i=1,4
q(i) = q(i)*qf
end do
do i=5,8
q(i) = q(i)*qw
end do

write(30,*) 'scaled charges: qf, qw =',qf,qw
write(30,1007) q

c*****
c**** Set the C values; decrease C range for C1 calc. *****
c*****
c(1) = (dfloat(m)*0.1d0 +2.0d0)*dfloat(k+1)
c(20)= (16.0d0 + dfloat(m)*0.5d0)*dfloat(k+1)

if(m.le.4) then
c(1)= dfloat(m-1)*dfloat(k+1)
c(20)= (10.0d0+dfloat(m-1)*3.0d0)*dfloat(k+1)
else
endif

c***** for C1 calc. decrease C range *****

if(lmb.ne.1.0d0) then
c(1) = 1.0d0
c(20)= c(20)/3.0d0
else
endif

c*****

if(nac.eq.0) then
c(1) = 0.0d0
c(20)= 14.0d0
c if(LJ.eq.1) c(20)= 10.0d0

c*****added below to set LJ range of c *****
if(LJ.eq.1) then
cest1=(6.0-2.0d0/3.0d0*2.1*(36.0d0*pi)**(1/3)*m**(-1/3))
cest = cest1*(150.0d0/T-1.0d0)-dlog(alp)
c(1)= cest-3.0d0
c(20)=cest+3.0d0

```

```

    else
    endif
c*****
    else
    endif

    delC = (c(20)-c(1))/20.0d0
    do icc=2,20
    c(icc)= c(icc-1) + delC
    end do

    if (mstart.eq.0) then

c*****
*
c***** mstart = 0: *****
c***** 1. READ ra, rb seeds file 20 *****
c***** 2. call pwsun, infout *****
c***** 3. initialize sums to 0.0d0 *****
c***** *****
c***** NOTE: seeda,seedb *****
c***** should be integers *****
c***** to avoid roundoff *****
c***** *****
*

    read(20,*) (iseeda(i), i=1,5)
    read(20,*) (iseeda(i), i=6,7)
    read(20,*) (iseedb(i), i=1,5)
    read(20,*) (iseedb(i), i=6,7)

    WRITE(30,1003)
    write(30,1008)(iseeda(i), i=1,5)
    write(30,1008)(iseeda(i), i=6,7)
    write(30,1008)(iseedb(i), i=1,5)
    write(30,1008)(iseedb(i), i=6,7)

    do i=1,na
    read(20,*) (ra(i,j), j = 1,3),ia(i),imola(i)
    write(30,1009) (ra(i,j), j = 1,3),ia(i),imola(i)
    end do

    do i=1,na

```

```

read(20,*) (rb(i,j), j = 1,3),ib(i),imolb(i)
write(30,1009) (rb(i,j), j = 1,3),ib(i),imolb(i)
end do

c..... RSL2 angles .....
roh=0.9584d0
anghoh=104.45D0

c..... TIP4P angles .....
roh=0.9572d0
anghoh=104.52D0

if(LJ.eq.0)nxang=0
if(nxang.eq.1) call angle(anghoh,roh)

c-----
c----- move two A probe H to monomer position ----
c----- OR reset H on acids next to SO4 ----
c-----
if(nac.ne.0) then

if(movAhh.gt.0) then
itop=m

if(movAhh.eq.1) ilow=m
if(movAhh.eq.2) ilow=1
if(movAhh.eq.3) ilow=1

do iacid =ilow,itop
isulf=nac*iacid-6
ih1=isulf + 5
ih2=ih1+1

write(30,*)'Before movAhh:'
write(30,*)'*****'
write(30,*)'atom: ',isulf,' ra: ',(ra(isulf,j),j=1,3)
write(30,*)'atom: ',ih1,' ra: ',(ra(ih1,j),j=1,3)
write(30,*)'atom: ',ih2,' ra: ',(ra(ih2,j),j=1,3)

if(movAhh.eq.3) then
write(30,*)'*****'
write(30,*)'atom: ',isulf,' rb: ',(rb(isulf,j),j=1,3)
write(30,*)'atom: ',ih1,' rb: ',(rb(ih1,j),j=1,3)
write(30,*)'atom: ',ih2,' rb: ',(rb(ih2,j),j=1,3)

```

```

write(30,*)'*****'
else
endif

do j=1,3

ra(ih1,j)=ra(isulf,j)+1.5
ra(ih2,j)=ra(isulf,j)-1.5

if(movAhh.eq.3) then
rb(ih1,j)=rb(isulf,j)+1.5
rb(ih2,j)=rb(isulf,j)-1.5
else
endif

end do

write(30,*)'After movAhh:'
write(30,*)'atom: ',isulf,' ra: ',(ra(isulf,j),j=1,3)
write(30,*)'atom: ',ih1,' ra: ',(ra(ih1,j),j=1,3)
write(30,*)'atom: ',ih2,' ra: ',(ra(ih2,j),j=1,3)

if(movAhh.eq.3) then
write(30,*)'*****'
write(30,*)'atom: ',isulf,' rb: ',(rb(isulf,j),j=1,3)
write(30,*)'atom: ',ih1,' rb: ',(rb(ih1,j),j=1,3)
write(30,*)'atom: ',ih2,' rb: ',(rb(ih2,j),j=1,3)
else
endif

end do
else
endif
c----- end if(nac.ne.0) -----
else
endif
c-----
c----- check potential sum -----
c----- and initialize (mstart = 0) -----
c-----
write(30,1003)
ichekk = 0

UAL_sum = 0.0d0
UB_sum = 0.0d0

```

```

UAdL_sum = 0.0d0
UBdL_sum = 0.0d0
UBdf_sum = 0.0d0
UbdF_av = 0.0d0

do icc=1,20
FA_sum(icc) =0.0d0

do jx=1,jup
FB_sum(jx,icc) =0.0d0
FB_av(jx,icc) = 0.0d0
FB(jx,icc)=0.0d0
UBdelf(jx)=0.0d0
UBdelLf(jx)=0.0d0
end do
end do

iacc_sum= 0
ibcc_sum= 0
iacc = 0
ibcc = 0
xacc=0d0
xbcc=0d0

xacc = dfloat(iacc_sum)/dfloat(mstart +1)
xbcc = dfloat(ibcc_sum)/dfloat(mstart +1)

do i=1,na
icounta(i) = 0
icountb(i) = 0
end do

MBdo=0
call pwsum(ra,rb,0)

write(30,*) 'UAL, UB, UAdelL, UbdelL: '
write(30,*) UAL,UB,UAdelL,UBdelL
write(30,*) 'UBdelf(jx): '
write(30,*) (UBdelf(jx),jx=1,jup)

do jx=jlo,jhi
UBdf_sum = UBdf_sum + UBdelf(jx)/dfloat(jxtot)
end do

do icc = 1,20

```

```

FA(icc) = 1.0d0/(1.0d0+ dexp( UAdelL/(kb*T) +c(icc) ))
FA_sum(icc) = FA_sum(icc) +FA(icc)

do jx=jlo,jhi
FB(jx,icc) = 1.0d0/(1.0d0+
1 dexp( -UBdelLf(jx)/(kb*T) -c(icc) ))
FB_sum(jx,icc) = FB_sum(jx,icc) +FB(jx,icc)
end do

end do

c  write(30,*)'initial: UBdelf(jx), jx=1,jup; UBdel'
c  write(30,*)(UBdelf(jx),do jx=1,jup),UBdel
c  write(30,*)'initial: UBdelLf(jx), do jx=1,jup; UBdelL'
c  write(30,*)(UBdelLf(jx),jx=1,jup),UBdelL

call infout(mstart,xacc,xbcc)
c----- move center of mass to origin -----

irsa=0
irsb=0

call cmas(ra,rb,rcma,rcmb,0,irsa,irsb)

write(30,1003)
if(irsa.gt.0) write(30,1011) irsa
if(irsb.gt.0) write(30,1012) irsb

1011 format('atom in A ',i4,' out of constraining sphere
1 initial step of run')
1012 format('atom in B ',i4,' out of constraining sphere
1 initial step of run')

write(30,1010) ( rcma(j),j=1,3 )
write(30,1010) ( rcmb(j),j=1,3 )

do i=1,na
write(30,1009) (ra(i,j), j = 1,3),ia(i),imola(i)
end do

do i=1,na
write(30,1009) (rb(i,j), j = 1,3),ib(i),imolb(i)
end do

```

```

write(30,1003)
call cmas(ra,rb,rcma,rcmb,0,irsa,irsb)
write(30,1010) ( rcma(j),j=1,3 )
write(30,1010) ( rcmb(j),j=1,3 )
write(30,1003)

c*****
c  call test
c*****

MBdo=0
call pwsun(ra,rb,mstart)
call infout(mstart,xacc,xbcc)

write(30,1003)

c*****
c***** Re-initialize sums mstart = 0 *****
c***** (infout needed sums for initial *****
c***** print out of info. for run ) *****
c*****

UAL_sum = 0.0d0
UB_sum = 0.0d0
UAdL_sum = 0.0d0
UBdL_sum = 0.0d0
UBdf_sum = 0.0d0

iacc_sum= 0
ibcc_sum= 0

do icc=1,20
FA_sum(icc)=0.0d0
do jx=1,jup
FB_sum(jx,icc)=0.0d0
c  FB(jx,icc)=0.0d0
FB_av(jx,icc)=0.0d0
c  UBdelf(jx)=0.0d0
c  UBdelLf(jx)=0.0d0
c  bBdelf(jx)=0.0d0
c  bBdelLf(jx)=0.0d0
end do

```



```

end do

else

c*****
**
c***** mstart > 0: *****
c***** 1. READ ra, rb seeds file 10 *****
c***** 2. call pwsun, infout *****
c***** 3. calc. FA FB *****
c*****
**

c----- check parameters in file 10 -----
rewind(10)

read(10,*) mmstep,k_x,m_x,T_x,alph_x,istop_x,qf_x,qw_x,alp_x

write(30,*) 'read mstep,k,m,T,alpha,istop,qf,qw,alp'
write(30,*) mmstep,k_x,m_x,T_x,alph_x,istop_x,qf_x,qw_x,alp_x

mstpcek=0
mcek = mmstep-mstart
kcek = k_x - k
mcek = m_x - m
Tcek = T_x - T
achek = alph_x- alpha
istpek = istop_x - istop
qfcek = qf_x - qf
qwcek = qw_x - qw
alpchk = alp_x- alp
icheckk = mstpcek+kcek+mcek+achek+istpek
1      +qfcek + qwcek+ alpchk

if(Tcek.ne.0.000d0) then
write(30,*) 'Temperature does not check in cfg. file'
icheck =1
else
endif

if(achek.ne.0.000d0) then
write(30,*) 'Alpha does not check in cfg. file'

```

```

ichekk = 1
else
endif

if (ichekk.ne.0) then
write(30,*) 'PARAMETERS DO not check in run, file'
go to 900
else
endif

c----- read seeds, ra, rb -----

read(10,*) (iseeda(i), i=1,5)
read(10,*) (iseeda(i), i=6,7)
read(10,*) (iseedb(i), i=1,5)
read(10,*) (iseedb(i), i=6,7)

do i=1,na
read(10,1009) (ra(i,j), j = 1,3),ia(i),imola(i)
write(30,1009) (ra(i,j), j = 1,3),ia(i),imola(i)
end do

do i=1,na
read(10,1009) (rb(i,j), j = 1,3),ib(i),imolb(i)
write(30,1009) (rb(i,j), j = 1,3),ib(i),imolb(i)
end do

c----- read stored data from last run -----

read(10,1015) UA_aa,UB_aa,UAdel_aa,UBdel_aa,
1 UA_aw,UB_aw,UAdel_aw,UBdel_aw,
2 UA_ww,UB_ww,UAdel_ww,UBdel_ww,
6 UAp_aa,UAp_aw,UBp_aa,UBp_aw,UA_px,UB_px,
3 UA, UB, UAdel, UBdel,UBdelf,
4 UAL,UBL,UAdelL,UBdelL,UBdelLf,
5 UA_pol,UB_pol,UAdel_p,UBdel_p
6 ,binUA,binUB

read(10,1015) UAL_sum, UB_sum,UAdL_sum,UBdL_sum,UBdf_sum
read(10,1017) iacc_sum, ibcc_sum
read(10,1015) (c(icc),icc=1,20)

read(10,1015) (FA(icc),icc=1,20)
do jx=1,jup

```

```

read(10,1015) (FB(jx,icc),icc=1,20)
end do

read(10,1015) (FA_sum(icc),icc=1,20)
do jx=1,jup
read(10,1015) (FB_sum(jx,icc),icc=1,20)
end do

c-----
c----- calculate xacc, xbcc and call infout -----
c----- call cmas to calculate fmass(i) -----
c-----

xacc = dfloat(iacc_sum)/dfloat(mstart)
xbcc = dfloat(ibcc_sum)/dfloat(mstart)

call cmas(ra,rb,rcma,rcmb,0,irsa,irsb)

c   call pwsun(ra,rb,mstart)

Mbstep=mstart/MB
call infout(mstart,xacc,xbcc)

c----- endif for mstart = 0, mstart >0 -----

endif

c*****
c***** MOVE MOVE MOVE *****
c*****

c*****
c***** BEGIN DO LOOP *****
if(mstart.eq.0)then
call rasmolh(0)
do iMAB=1,MAB
binUA(iMAB)=0.0d0
binUB(iMAB)=0.0d0
end do
else
endif

```

```

MBstep=mstart/MB

DO mstepp=1,mmax
mstep= mstart + mstepp

MBdo =dint( mod( dfloat(mstep),dfloat(MB) ) )

call moveab(T,mstep,iacc,ibcc)
if(iacc.lt.0) go to 900
iacc_sum = iacc_sum + iacc
ibcc_sum = ibcc_sum + ibcc
c  write(30,*)mstep,iacc,ibcc

nprint=dint( mod( dfloat(mstep),dfloat(mprint) ) )
xacc=dfloat(iacc_sum)/dfloat(mstep)
xbcc=dfloat(ibcc_sum)/dfloat(mstep)

UAL_sum = UAL_sum + UAL
UB_sum = UB_sum + UB
UBdL_sum= UBdL_sum + UBdelL
UAdL_sum= UAdL_sum + UAdelL

c***** Bin the delUA *****

UAbin=UAdel + 100.0d0
nbinA=dint(UAbin)
if(nbinA.gt.MAB) nbinA=MAB
if(nbinA.gt.0) then
binUA(nbinA)=binUA(nbinA) +1.0d0
else
write(30,*)'UAdelL= ',UAdel,' nbinA lt. 0'
endif

c*****

if(MBdo.eq.0)then
MBstep = MBstep+1
if(mstep.eq.MB) UBdf_sum=0.0d0

do jx=jlo,jhi
UBdf_sum = UBdf_sum + UBdelf(jx)/dfloat(jxtot)

UBbin=UBdelf(jx)+100.d0

```



```

c  call infout(mstep,xacc,xbcc)
  write(30,1003)
c***** write out molecules out counts *****

  write(30,*) ' atom out:  key # times in A    in B'
  do i=1,na
  write(30,1014) i, ia(i),icounta(i),icountb(i)
  end do
c***** write out UAbin and UB bins *****
  write(30,1003)
  write(30,*) '#bin dU      binA      binB  '
  do iMAB=1,MAB
  xdU=dfloat(iMAB)-100.d0
  write(30,5000) iMAB,xdU,binUA(iMAB),binUB(iMAB)
  end do
  MBtot=MBstep*jxtot
  write(30,*) 'ntotA, ntotB',mstep,MBtot
  write(30,1003)
5000 format(I6,4x,f8.2,5x,f12.2,5x,f12.2)
c***** write out DATA for TABLE *****
  fmstpp=mstep/1.0d6
  nwai=k*m
  write(30,1003)
  flam=dlog(lamb)/dlog(10.0d0)
  write(30,*)' m km  jup  lamb  alph  densLJ  Cf  Cplt  UA  UB  UAdel
1 UBdel  stp#  qw  qs  alp  lmb  T'
  write(30,4000) m,nwai,jup, flam,alpha,densLJ, Cfin,Cplot,
1 UAL_av,UB_av,UAdel_av,UBdf_av, fmstpp, qw,qs, alp,lmb,T
  write(30,4000) m,nwai,jup, flam,alpha,densLJ,cav,cp_ave,
c  1 UAL_av,UB_av,UAdel_av,UBdf_av, fmstpp, qw,qs,alp,lmb,T
  1 UAL_av,UB_av,UAdel_av,UBdel_av, fmstpp, qw,qs,alp,lmb,T
c*****7**0***5****0****5****0****5****0****5****0****5****0****5****0****5****0****5****0**
4000 format(I3,I3,1x,I3,1x, f4.1,1x,f4.1,1x,f5.2, 2(1x,f6.3),
  1 2(1x,f6.2),1x, f10.1, 1x,f6.1,1x, f6.2,'M', 2(1x,f5.2),1x,
  2 f3.1,1x,f3.1,1x,f5.1)

c*****
c*****
c----- write info for next run -----
c----- to cxaxw.run -----
c*****
c*****
c  mstep = mstart+mmax
  rewind (10)

```

```
write(10,*) mstop,k,m,T,'d0 ',alpha,'d0 ',istop,
1 qf,'d0 ',qw,'d0 ',alp,'d0'
```

```
write(10,*) (iseeda(i), i=1,5)
write(10,*) (iseeda(i), i=6,7)
write(10,*) (iseedb(i), i=1,5)
write(10,*) (iseedb(i), i=6,7)
```

```
do i=1,na
write(10,1009) (ra(i,j), j = 1,3),ia(i),imola(i)
end do
```

```
do i=1,na
write(10,1009) (rb(i,j), j = 1,3),ib(i),imolb(i)
end do
```

```
write(10,1015) UA_aa,UB_aa,UAdel_aa,UBdel_aa,
1 UA_aw,UB_aw,UAdel_aw,UBdel_aw,
2 UA_ww,UB_ww,UAdel_ww,UBdel_ww,
6 UAp_aa,UAp_aw,UBp_aa,UBp_aw,UA_px,UB_px,
3 UA, UB, UAdel, UBdel, UBdelf,
4 UAL,UBL,UAdelL,UBdelL,UBdelLf,
5 UA_pol,UB_pol,UAdel_p,UBdel_p
6 ,binUA,binUB
```

```
write(10,1015) UAL_sum, UB_sum,UAdL_sum,UBdL_sum,UBdf_sum
write(10,1017) iacc_sum, ibcc_sum
```

```
write(10,1015) (c(icc),icc=1,20)
```

```
write(10,1015) (FA(icc),icc=1,20)
do jx=1,jup
write(10,1015) (FB(jx,icc),icc=1,20)
end do
```

```
write(10,1015) (FA_sum(icc),icc=1,20)
do jx=1,jup
write(10,1015) (FB_sum(jx,icc),icc=1,20)
end do
```

```
c***** contingency end point *****
```

```
900 continue
```

```

if(ichekk.ne.0) write(30,*) 'INPUT not consistent'
if(iacc.lt.0) write(30,*) 'RUN TERMINATED: MOLECULEE OUT!'

1014 format(i4,9x,i3,2(i10,2x))
c   s = secnds(s)
c   write(30,*)'Run took ',s,' seconds for ',mmax,' steps'
   stop
   end

c*****
c*****
c*****

SUBROUTINE moveab(Tx,mstep,iacc,ibcc)
implicit real*8 (a-h,o-z)
real*8 lamb,lmb,kb,mass
parameter(MAT=600)
parameter(MAM=600)
parameter(MAB=600)
COMMON/R/ ra(MAT,3),rb(MAT,3),ia(MAT),ib(MAT)
1   ,imola(MAT),imolb(MAT),rcma(3),rcmb(3)
COMMON/seeds/iseeda(7),iseedb(7)
c   COMMON/CONST/ kb,coul,pi,mi,ic,kall,alpha,istop,qf,qw,alp
COMMON/CONST/ kb,coul,pi,alpha,qf,qw,alp,ic,kall,istop,mi
c   COMMON/km/k,m,na,nac,nwa,lamb,lmb,disp,beta,dd,Rsqa,jup,disa(MAT)
c   1 ,jlo,jhi,jxtot,LJ
COMMON/km/lamb,lmb,disp,beta,dd,Rsqa,disa(MAT),k,m,na,nac,nwa,jup,
1 jlo,jhi,jxtot,LJ

COMMON/SIG/is(MAT),sig(8),eps(8),q(8),mass(8),Iprobe(MAM,MAT)
COMMON/UAUB/ UA_aa,UB_aa,UAdel_aa,UBdel_aa,
1   UA_aw,UB_aw,UAdel_aw,UBdel_aw,
2   UA_ww,UB_ww,UAdel_ww,UBdel_ww,
6   UAp_aa,UAp_aw,UBp_aa,UBp_aw,UA_px,UB_px,
3   UA, UB, UAdel, UBdel,UBdelf(MAM),
4   UAL,UBL,UAdelL,UBdelL,UBdelLf(MAM),
5   UA_pol,UB_pol,UAdel_p,UBdel_p
6   ,binUA(MAB),binUB(MAB)

COMMON/FMASS/fmass(MAT),fmasstot,fmasstota
COMMON/COUNTS/ icounta(MAT),icountb(MAT),It_max

double precision rna(7),rnb(7),delta(3),deltb(3)

```



```

double precision da,db,dxa(3),dxb(3),dxap(3),dxbp(3)
double precision ra_buff(MAT,3),rb_buff(MAT,3)

c*****7**0***5****0****5****0****5****0****5****0****5****0****5****0****5****0**
c-----
c----- SEEDS: (1,2,3)=>translate; 4=> axis; 5=> angle; 6 not used; --
c-----      7==> accept, not accept.          --
c-----
      T=Tx
      do iseed=7,7
      call rn( iseeda(iseed),mi,ic, rna(iseed))
      call rn( iseedb(iseed),mi,ic, rnb(iseed))
      end do

      if(mstep.le.2)then
      write(30,1000)
      write(30,1001) mstep,k,m,dd,disp,beta
      write(30,1018) (disa(i), i=1,m*nac)
      write(30,1019) (disa(i), i=m*nac+1,na)
      else
      endif

1000 format('*****')
      1*****')
1001 format('in moveab',3(I3,1x),3(d24.16,1x))
1018 format(7(f6.1,1x))
1019 format(12(f6.1,1x))
c----- move all the acid molecules -----
      I_tries = 0

100 continue
      Iaaa=1
      Ibbb=2
      if(nac.ne.0) then

      DO mac = 1,m

c-----
c----- translate SO4-- unit -----
c----- nmov = 1,8,15,22,29 -----
c-----      the S atoms -----

```

```

c-----

      nmov = nac*(mac-1)+1

      do i=1,3
      call rn(iseeda(i),mi,ic,rna(i))
      call rn(iseedb(i),mi,ic,rnb(i))
      delta(i)=2.0d0*disp*(rna(i)-0.5d0)/dd*disa(nmov)
      deltb(i)=2.0d0*disp*(rnb(i)-0.5d0)/dd
      ra_buff(nmov,i)=ra(nmov,i)+delta(i)
      rb_buff(nmov,i)=rb(nmov,i)+deltb(i)

      do iso = 1,4
      natm=nmov+iso
c    write(30,*) nmov,natm
      ra_buff(natm,i)=ra(natm,i)+delta(i)
      rb_buff(natm,i)=rb(natm,i)+deltb(i)
      end do

      end do

c-----
c----- rotate the SO4 unit -----
c-----
c----- ix =1,2,3 ==> x,y,z axis
c-----

      call rn(iseeda(4),mi,ic,rna(4))
      call rn(iseedb(4),mi,ic,rnb(4))
      third=1.0d0/3.0d0
      ixa=dint(rna(4)/third) + 1
      ixb=dint(rnb(4)/third) + 1

      call rn(iseeda(5),mi,ic,rna(5))
      call rn(iseedb(5),mi,ic,rnb(5))
      da = 2.0d0*beta*(rna(5)-0.5d0)*disa(nmov)
      db = 2.0d0*beta*(rnb(5)-0.5d0)

      das=dsin(da )
      dac=dcos(da )
      dbs=dsin(db )
      dbc=dcos(db )

```

```

do iso4 = 1,4
natm=nmov+iso

do ii=1,3
dxa(ii) = ra_buff(natm,ii)-ra_buff(nmov,ii)
dxb(ii) = rb_buff(natm,ii)-rb_buff(nmov,ii)
dxap(ii)=0.0d0
dxbp(ii)=0.0d0
end do

call rotate(ixa,dxa,dxap,dac,das)
call rotate(ixb,dxb,dxbp,dbc,dbs)

do ii=1,3
ra_buff(natm,ii) = ra_buff(nmov,ii) + dxap(ii)
rb_buff(natm,ii) = rb_buff(nmov,ii) + dxbp(ii)
end do

end do

c----- move the H+ -----

do ihp = 5,6

do i=1,3
call rn(iseeda(i),mi,ic,rna(i))
call rn(iseedb(i),mi,ic,rnb(i))
delta(i)=2.0d0*disp*(rna(i)-0.5d0)/dd*disa(nmov)
deltb(i)=2.0d0*disp*(rnb(i)-0.5d0)/dd
ra_buff(nmov+ihp,i)=ra(nmov+ihp,i)+delta(i)
rb_buff(nmov+ihp,i)=rb(nmov+ihp,i)+deltb(i)
end do

end do
c----- end do on mac = 1,m -----
END DO

c----- end if(nac.ne.0) -----
else
endif

c----- move the waters -----

DO kk=1,k*m

```

```

nmov = nac*m + 3*(kk-1) + 1

do i=1,3
call rn(iseeda(i),mi,ic,rna(i))
call rn(iseedb(i),mi,ic,rnb(i))
delta(i)=2.0d0*disp*(rna(i)-0.5d0)/dd*disa(nmov)
deltb(i)=2.0d0*disp*(rnb(i)-0.5d0)/dd
ra_buff(nmov,i)=ra(nmov,i)+delta(i)
rb_buff(nmov,i)=rb(nmov,i)+deltb(i)

do iso = 1,2
natm=nmov+iso
ra_buff(natm,i)=ra(natm,i)+delta(i)
rb_buff(natm,i)=rb(natm,i)+deltb(i)
end do

end do

c-----
c----- rotate the H2O unit -----
c-----
c----- ix =1,2,3 ==> x,y,z axis
c-----

call rn(iseeda(4),mi,ic,rna(4))
call rn(iseedb(4),mi,ic,rnb(4))
third=1.0d0/3.0d0
ixa=dint(rna(4)/third) + 1
ixb=dint(rnb(4)/third) + 1

call rn(iseeda(5),mi,ic,rna(5))
call rn(iseedb(5),mi,ic,rnb(5))
da = 2.0d0*beta*(rna(5)-0.5d0)*disa(nmov)
db = 2.0d0*beta*(rnb(5)-0.5d0)

das=dsin(da )
dac=dcos(da )
dbs=dsin(db )
dbc=dcos(db )

do ih2 = 1,2
natm=nmov+ih2

do ii=1,3

```

```

dxa(ii) = ra_buff(natm,ii)-ra_buff(nmov,ii)
dxb(ii) = rb_buff(natm,ii)-rb_buff(nmov,ii)
end do

```

```

call rotate(ixa,dxa,dxap,dac,das)
call rotate(ixb,dxb,dxbp,dbc,dbs)

```

```

do ii=1,3
ra_buff(natm,ii) = ra_buff(nmov,ii) + dxap(ii)
rb_buff(natm,ii) = rb_buff(nmov,ii) + dxbp(ii)
end do

```

```

end do

```

```

END DO

```

```

do i=1,na

```

```

rda=0.0d0
rbr=0.0d0
do j=1,3
rda=rda+ra_buff(i,j)**2
rdb=rdb+rb_buff(i,j)**2
end do

```

```

end do

```

```

c-----
c----- Check: all atoms inside radius? -----
c----- if irsa > 0 atom irsa is out in A -----
c----- if irsb > 0 atom irsb is out in B -----
c----- ... irsa, irsb found in cmas -----
c----- ... cmas also puts cm at (0,0,0) -----
c----- for ra_buff or rb_buff coords -----
c-----
c----- AFTER cmas is called: -----
c----- IF kall = 0 -----
c----- .. irsa set = 1 ONLY IF -----
c----- S,H+, or O in -----
c----- water is out of sphere -----
c----- .. irsa set = 0 otherwise -----
c----- (H in water or O in SO4 -----

```

```

c----- can be out of sphere) -----
c----- -----
c----- IF kall = 1 then any atom out -----
c----- sends routine back -----
c----- to find another move -----
c----- .. used this to match Shawn's -----
c----- constraint condition. -----
c----- .. kall = 0 ==> use 1,4,5 check -----
c-----
c
  irsa=0
  irsb=0
  irsav = irsa
  irsbv = irsb

c  kall=1

  call cmas(ra_buff,rb_buff,rcma,rcmb,1,irsa,irsb)
  irsav = irsa
  irsbv = irsb

  if(irsa.gt.0)then
    icounta(irsa) = icounta(irsa) + 1
    irsa=1

c    if(ia(irsa).eq.1) irsa=1
c    if(ia(irsa).eq.4) irsa=1
c    if(ia(irsa).eq.5) irsa=1

c    if(kall.ne.1) write(30,*)'kall = ',kall
c    if(kall.eq.1) irsa = 1

c    if(irsa.gt.1) irsa=0
c    if(irsa.eq.1) write(30,1014) irsa,ia(irsa)

  else
  endif

  if(irsb.gt.0) then
    icountb(irsb) = icountb(irsb) + 1
    irsb=1

c    if(ib(irsb).eq.1) irsb=1
c    if(ib(irsb).eq.4) irsb=1
c    if(ib(irsb).eq.5) irsb=1

```

```

c   if(kall.ne.1) write(30,*)'kall = ',kall
c   if(kall.eq.1) irsb = 1

c   if(irsb.gt.1) irsb=0
c   if (irsb.eq.1) write(30,1015) irsb,ib(irsb)

      else
      endif

      if((1-irsa).lt.0) write(30,*)'irsa greater than 1!'
      if((1-irsb).lt.0) write(30,*)'irsb greater than 1!'

c-----
c----- reject both moves if one fails -----
c----- failure: S,H+ or O-water out -----
c-----

      if((irsa+irsb).eq.0) go to 500
c   It_max = 10000
      I_tries = I_tries + 1

c   if(I_tries.eq.1000)
c   1 write(30,*) 'Move repeated',I_tries, ' times:
c   2 step= ',mstep,irsa,irsb,' kall = ',kall

      if(I_tries.ge.It_max) then

          write(30,*) 'Exhausted ',It_max,' tries at step ',mstep
          write(30,*) 'MOLECULE OUT ! in A, B =',irsav,irsbv
          iacc=-1
          ibcc=-1
          return
      else
          go to 100
      endif

500 continue
c   write(30,*)'I_tries: ',I_tries
      I_tries = 0
c   if((irsa+irsb).ne.0) write(30,*) 'MOLECULE OUT !',mstep
c   if((irsav+irsbv).ne.0) write(30,*) 'MOLECULE OUT, KALL wrong!'
c-----

c1009 format(1x,I4,3(d24.16,1x),5x,2(I2,3x),d24.16)

```

```

c in moveab
1009 format(3(d24.16,1x),I1,1x,I3)
1010 format('atom',i4,' in A (key= ',i4,') out sphere, reload old?')
1011 format('atom',i4,' in B (key= ',i4,') out sphere, reload old?')
1012 format(' irsa set = ',i4,' in A (key= ',i4,') do not reload old')
1013 format(' irsb set = ',i4,' in B (key= ',i4,') do not reload old')
1014 format(' irsa set = ',i4,' in A (key= ',i4,') reload old')
1015 format(' irsb set = ',i4,' in B (key= ',i4,') reload old')

1016 format(1x,3(d24.16,1x),5x,2(I2,3x),d24.16)
c-----
c----- accept/reject coordinates -----
c----- if S, H, H or O (in H2O) are out -----
c----- key= 1, 4, 4 or 5 then -----
c----- move is rejected, irsa set = 1 -----
c----- ... otherwise accept, irsa = 0 -----
c-----
  IAAA=1
  IBBB=1
  iacc=0
  ibcc=0
  UALold = UAL
  UBold = UB

  call bufferE(na,m)

  call pwsum(ra_buff,rb_buff,1)

c----- calculate Boltzmann factors -----

  boltzA =dexp( -(UAL- UALold)/(kb*T) )
  boltzB =dexp( -(UB - UBold)/(kb*T) )

c  if(mstep.eq.23) then
c  write(30,*)'UAL,UALold: ',UAL,UALold
c  write(30,*)'irsa,irsb: ',irsav,irsbv
c  write(30,*)'boltzA, rna(7): ',boltzA,rna(7)
c  do iz= 1,3
c  write(30,*)iz, (ra_buff(iz,kk),kk=1,3)
c  end do
c  write(30,*)'rna: ',(rna(kk),kk=1,7)
c  else
c  endif

```


c----- Metropolis decision for ensemble A -----

```
if (boltzA.ge.rna(7)) then
  iacc = 1
```

```
  do i=1,na
    do j=1,3
      ra(i,j)=ra_buff(i,j)
    end do
  end do
```

```
else
  call loadA(0)
endif
```

c----- Metropolis decision for ensemble B -----

```
if (boltzB.ge.rnb(7)) then
  ibcc=1
```

```
  do i=1,na
    do j=1,3
      rb(i,j)=rb_buff(i,j)
    end do
  end do
```

```
else
  call loadB(na,m)
endif
```

c-----

```
c  write(30,*) 'iacc, ibcc at end of moveab step= ',iacc,ibcc,mstep
  return
end
```

```
c*****7*****
c*****7*****
```

```
SUBROUTINE bufferE(nap,mp)
  implicit real*8 (a-h,o-z)
  real*8 lamb,lmb
```

c-----

```
  parameter(MAT=600)
  parameter(MAM=600)
  parameter(MAB=600)
  COMMON/UAUB/ UA_aa,UB_aa,UAdel_aa,UBdel_aa,
1      UA_aw,UB_aw,UAdel_aw,UBdel_aw,
```

```

2      UA_ww,UB_ww,UAdel_ww,UBdel_ww,
6      UAp_aa,UAp_aw,UBp_aa,UBp_aw,UA_px,UB_px,
3      UA, UB, UAdel, UBdel,UBdelf(MAM),
4      UAL,UBL,UAdelL,UBdelL,UBdelLf(MAM),
5      UA_pol,UB_pol,UAdel_p,UBdel_p
6      ,binUA(MAB),binUB(MAB)

```

c COMMON/km/k,m,na,nac,nwa,lamb,lmb,disp,beta,dd,Rsqa,jup,disa(MAT)

c 1 ,jlo,jhi,jxtot,LJ

COMMON/km/lamb,lmb,disp,beta,dd,Rsqa,disa(MAT),k,m,na,nac,nwa,jup,

1 jlo,jhi,jxtot,LJ

COMMON/UAUBuff/ bA_aa,bB_aa,bAdel_aa,bBdel_aa,

```

1      bA_aw,bB_aw,bAdel_aw,bBdel_aw,
2      bA_ww,bB_ww,bAdel_ww,bBdel_ww,
6      bAp_aa,bAp_aw,bBp_aa,bBp_aw,bA_px,bB_px,
3      bA, bB, bAdel, bBdel, bBdelf(MAM),
4      bAL,bBL,bAdelL,bBdelL,bBdelLf(MAM),
5      bApol,bBpol,bAdelp,bBdelp

```

c-----

```

bA_aa = UA_aa
bB_aa = UB_aa
bAdel_aa = UAdel_aa
bBdel_aa = UBdel_aa

```

```

bA_aw = UA_aw
bB_aw = UB_aw
bAdel_aw = UAdel_aw
bBdel_aw = UBdel_aw

```

```

bA_ww = UA_ww
bB_ww = UB_ww
bAdel_ww = UAdel_ww
bBdel_ww = UBdel_ww

```

```

bAp_aa = UAp_aa
bAp_aw = UAp_aw
bBp_aa = UBp_aa
bBp_aw = UBp_aw
bA_px = UA_px
bB_px = UB_px

```

```

bA = UA
bB = UB
bAdel = UAdel

```

```

bBdel  = UBdel

bAL    = UAL
bBL    = UBL
bAdelL = UAdelL
bBdelL = UBdelL

bApol  = UA_pol
bBpol  = UB_pol
bAdelp = UAdel_p
bBdelp = UBdel_p

do jx=jlo,jhi
  bBdelf(jx)=UBdelf(jx)
  bBdelLf(jx)=UBdelLf(jx)
end do

return
end
c*****
SUBROUTINE loadA(nap)
  implicit real*8 (a-h,o-z)
  real*8 lamb,lmb
c-----
  parameter(MAT=600)
  parameter(MAM=600)
  parameter(MAB=600)
  COMMON/UAUB/ UA_aa,UB_aa,UAdel_aa,UBdel_aa,
1    UA_aw,UB_aw,UAdel_aw,UBdel_aw,
2    UA_ww,UB_ww,UAdel_ww,UBdel_ww,
6    UAp_aa,UAp_aw,UBp_aa,UBp_aw,UA_px,UB_px,
3    UA, UB, UAdel, UBdel,UBdelf(MAM),
4    UAL,UBL,UAdelL,UBdelL,UBdelLf(MAM),
5    UA_pol,UB_pol,UAdel_p,UBdel_p
6    ,binUA(MAB),binUB(MAB)
c  COMMON/km/k,m,na,nac,nwa,lamb,lmb,disp,beta,dd,Rsqa,jup,disa(MAT)
c  1 ,jlo,jhi,jxtot,LJ
  COMMON/km/lamb,lmb,disp,beta,dd,Rsqa,disa(MAT),k,m,na,nac,nwa,jup,
1  jlo,jhi,jxtot,LJ

  COMMON/UAUBuff/ bA_aa,bB_aa,bAdel_aa,bBdel_aa,
1    bA_aw,bB_aw,bAdel_aw,bBdel_aw,
2    bA_ww,bB_ww,bAdel_ww,bBdel_ww,

```

```

6      bAp_aa,bAp_aw,bBp_aa,bBp_aw,bA_px,bB_px,
3      bA, bB, bAdel, bBdel, bBdelf(MAM),
4      bAL,bBL,bAdelL,bBdelL,bBdelLf(MAM),
5      bApol,bBpol,bAdelp,bBdelp

```

c-----

```

UA_aa  = bA_aa
UAdel_aa = bAdel_aa

```

```

UA_aw  = bA_aw
UAdel_aw = bAdel_aw

```

```

UA_ww  = bA_ww
UAdel_ww = bAdel_ww

```

```

UAp_aa = bAp_aa
UAp_aw = bAp_aw
UA_px  = bA_px

```

```

UA     = bA
UAdel = bAdel

```

```

UAL     = bAL
UAdelL = bAdelL

```

```

UA_pol = bApol
UAdel_p = bAdelp

```

```

return
end

```

c*****

```

SUBROUTINE loadB(nap,mp)
implicit real*8 (a-h,o-z)
real*8 lamb,lmb

```

c-----

```

parameter(MAT=600)
parameter(MAM=600)
parameter(MAB=600)
COMMON/UAUB/ UA_aa,UB_aa,UAdel_aa,UBdel_aa,
1      UA_aw,UB_aw,UAdel_aw,UBdel_aw,
2      UA_ww,UB_ww,UAdel_ww,UBdel_ww,

```

```

6      UAp_aa,UAp_aw,UBp_aa,UBp_aw,UA_px,UB_px,
3      UA, UB, UAdel, UBdel,UBdelf(MAM),
4      UAL,UBL,UAdelL,UBdelL,UBdelLf(MAM),
5      UA_pol,UB_pol,UAdel_p,UBdel_p
6      ,binUA(MAB),binUB(MAB)

```

```

c      COMMON/km/k,m,na,nac,nwa,lamb,lmb,disp,beta,dd,Rsqa,jup,disa(MAT)
c      1 ,jlo,jhi,jxtot,LJ
COMMON/km/lamb,lmb,disp,beta,dd,Rsqa,disa(MAT),k,m,na,nac,nwa,jup,
1 jlo,jhi,jxtot,LJ

```

```

COMMON/UAUBuff/ bA_aa,bB_aa,bAdel_aa,bBdel_aa,
1      bA_aw,bB_aw,bAdel_aw,bBdel_aw,
2      bA_ww,bB_ww,bAdel_ww,bBdel_ww,
6      bAp_aa,bAp_aw,bBp_aa,bBp_aw,bA_px,bB_px,
3      bA, bB, bAdel, bBdel, bBdelf(MAM),
4      bAL,bBL,bAdelL,bBdelL,bBdelLf(MAM),
5      bApol,bBpol,bAdelp,bBdelp

```

c-----

```

UB_aa = bB_aa
UBdel_aa = bBdel_aa

```

```

UB_aw = bB_aw
UBdel_aw = bBdel_aw

```

```

UB_ww = bB_ww
UBdel_ww = bBdel_ww

```

```

UBp_aa = bBp_aa
UBp_aw = bBp_aw
UB_px = bB_px

```

```

UB = bB
UBdel = bBdel

```

```

UBL = bBL
UBdelL = bBdelL

```

```

UB_pol = bBpol
UBdel_p = bBdelp

```

```

do jx=jlo,jhi

```

```

UBdelf(jx)=bBdelf(jx)
UBdelLf(jx)=bBdelLf(jx)
end do

return
end
c*****
SUBROUTINE rotate(ixa,dxx,dxxp,dac,das)
implicit real*8 (a-h,o-z)
double precision dxx(3),dxxp(3),ix(3,3)
c-----
c----- ixa is axis about which rotation takes place -----
c----- Euler angle rotations: Goldstein convention -----
c----- postive angles are clockwise -----
c----- rotated frame moves with rigid molecule -----
c----- .... so dxxp(i) = R**(-1)ij dxx(j) -----
c----- .... "primed frame" on molecule -----
c----- .... dxpp(i) are the new fixed frame -----
c----- coordinates of rotated atoms -----
c-----
fcos=dac
fsin=das

ix(1,1)=1
ix(1,2)=2
ix(1,3)=3

ix(2,1)=2
ix(2,2)=3
ix(2,3)=1

ix(3,1)=3
ix(3,2)=1
ix(3,3)=2

i1=ixa
i2=ix(ixa,2)
i3=ix(ixa,3)

d1 = dxx(i1)
d2 = dxx(i2)
d3 = dxx(i3)

dxxp(i1) = d1

```

```

dxxp(i2) = d2*fcos - d3*fsin
dxxp(i3) = d3*fcos + d2*fsin

return
end

c*****7**0***5****0****5****0****5****0****5****0****5****0****5****0****5****0**

SUBROUTINE pwsun(rar,rbr,iflag)
implicit real*8 (a-h,o-z)
real*8 lamb,lmb,kb,mas
parameter(MAT=600)
parameter(MAM=600)
parameter(MAB=600)
COMMON/R/ ra(MAT,3),rb(MAT,3),ia(MAT),ib(MAT)
1      ,imola(MAT),imolb(MAT),rcma(3),rcmb(3)
COMMON/seeds/iseeda(7),iseedb(7)
c COMMON/CONST/ kb,coul,pi,mi,ic,kall,alpha,istop,qf,qw,alp
COMMON/CONST/ kb,coul,pi,alpha,qf,qw,alp,ic,kall,istop,mi
COMMON/STEP/mstart,mmax,mprint,MBdo,MBstep
c COMMON/km/k,m,na,nac,nwa,lamb,lmb,disp,beta,dd,Rsqa,jup,disa(MAT)
c 1 jlo,jhi,jxtot,LJ
COMMON/km/lamb,lmb,disp,beta,dd,Rsqa,disa(MAT),k,m,na,nac,nwa,jup,
1 jlo,jhi,jxtot,LJ

COMMON/SIG/is(MAT),sig(8),eps(8),q(8),mass(8),Iprobe(MAM,MAT)
COMMON/UAUB/ UA_aa,UB_aa,UAdel_aa,UBdel_aa,
1      UA_aw,UB_aw,UAdel_aw,UBdel_aw,
2      UA_ww,UB_ww,UAdel_ww,UBdel_ww,
6      UAp_aa,UAp_aw,UBp_aa,UBp_aw,UA_px,UB_px,
3      UA, UB, UAdel, UBdel,UBdelf(MAM),
4      UAL,UBL,UAdelL,UBdelL,UBdelLf(MAM),
5      UA_pol,UB_pol,UAdel_p,UBdel_p
6      ,binUA(MAB),binUB(MAB)

double precision sigg(8,8),epss(8,8),rra(MAT,MAT),rrb(MAT,MAT)
double precision rar(MAT,3),rbr(MAT,3),qq(8),falp(10)
double precision rma(MAT,3),rmb(MAT,3)

dimension j1(700)
c*****

```

```

c   vlj(xx,epss)= 4.d0*epss*(xx*2-xx)

c   vcoul(rx,ix,jx)= coul*q(ix)*q(jx)/rx
c.....
c..... The RSL2 interaction potential .....
c.....
   voo(rx) = (144.538d0/rx) + 26758.2d0/rx**8.8591d0
1         -0.25d0*dexp(-4.0d0*(rx-3.4d0)**2)
2         -0.25d0*dexp(-1.5d0*(rx-4.5d0)**2)
   voh(rx) = -72.269d0/rx + 6.23403d0/rx**9.19912d0
1         -10.0d0/( 1.0d0+dexp(40.0d0*(rx-1.05d0)) )
2         -4.0d0 / ( 1.0d0+dexp(5.49305d0*(rx-2.2d0)) )
   vhh(rx) = 36.1345d0/rx +
1         18.0d0/( 1.0d0+dexp(40.0d0*(rx-2.05d0)) ) +
2         -17.0d0*dexp(-7.62177d0*(rx-1.45251d0)**2)
c.....
c..... The TIP4P interaction potential .....
c..... note rx6 = rx**6 .....
c..... HOH angle = 104.52 .....
c..... Roh = 0.9572 A .....
c.....

c   voo4P(rx6) = 600.0d3/rx6**2 - 610.0d0/rx6
   voo4P(rx6) = Aoo/rx6**2 - Coorx6

   vmm4P(rx) = flagLJ*(0.52d0/0.32983d0)**2*(144.538d0/rx)
   vmh4P(rx) = flagLJ*(0.52d0/0.32983d0)**2*(-72.269d0/rx)
   vhh4P(rx) = flagLJ*(0.52d0/0.32983d0)**2*(36.1345d0/rx)

   Q4P = 0.52d0
   QRS= 0.32983d0
   scale = (Q4P/QRS)**2

c*****
c   do i=1,na
c   write(30,*) 'in BEGINNING OF pwsun'
c   write(30,*) 'rar(i,j) then same for B ensemble'
c   write(30,*) 'A ensemble',(rar(i,jj),jj=1,3)
c   write(30,*) 'B ensemble',(rbr(i,jj),jj=1,3)
c   write(30,*)'rbr(i,j)-rb(i,j),then same for A ensemble'
c   write(30,*) ((rbr(i,jj)-rb(i,jj)),jj=1,3)
c   write(30,*) ((rar(i,jj)-ra(i,jj)),jj=1,3)
c   end do
c   write(30,*) 'in BEGINNING OF pwsun'
c   write(30,*) 'UAL= ',UAL,'UB= ',UB

```



```

c  if(iflag.ne.0)
c-----
  nca=nac
  nwa=3
  nc=m*nca
  nw=k*m*nwa

  Aoo=600.0d3
  Coo=610.0d0

  flagLJ=1.0d0

  if(LJ.eq.1)then
c.....
c.....  Lennard-Jones parameters  .....
c.....
  flagLJ=0.0d0

  epsilon=4.0D+00*119.4D+00*1.987905D-03
  sigr=3.4D+00*3.4D+00

  Aoo= epsilon*sigr**6
  Coo= epsilon*sigr**3
c.....

  else
  endif

  do i=1,8
  do j=1,8
  sigg(i,j)= (sig(i)+sig(j))/2.0d0
  epss(i,j)= dsqrt(dabs( eps(i) )*dabs( eps(j) ) )
  end do

  qq(i) = q(i)

  end do

c***** moved this to main program *****
  go to 888
  if(qf.ne.0.0d0) then
  do i = 1,4

```

```

qq(i)= q(i)*qf
end do

else
do i=1,4
qq(i) = q(i)
end do

endif

if(qw.ne.0.0d0) then
do i = 5,8
qq(i)=q(i)*qw
end do

else
do i = 5,8
qq(i)=q(i)
end do

endif
888 continue

c***** acid - acid interactions *****

UB_aa=0.0d0
UA_aa=0.0d0
UBdel_aa=0.0d0
UAdel_aa=0.0d0
Uaaint=0.0d0

if(MBdo.eq.0)then
do jx=jlo,jhi
UBdelf(jx)=0.0d0
end do
else
endif

c----- acid-acid -----
c--- algorithm to omit internal SO4 interactions ----
c-----
c-----
c----- if(nac.ne.0) -----

c***** IF NAC not 0 *****

```

```

if(nac.ne.0)then

do ii=1,m

do i= ii +(ii-1)*6,nac*ii-2
j1(i) = nac*ii-1
end do
j1(nac*ii-1) =nac*ii
j1(nac*ii) =nac*ii+1
end do

c-----
c----- j1 = 6,6,6,6,6, 7, 8, 13,13,13,13,13, 14 ---
c----- i = 1,2,3,4,5, 6, 7, 8, 9,10,11,12, 13 ---
c-----

do i=1,nc-1
do j=j1(i),nc

ibi=ib(i)

ibj=ib(j)
iai=ia(i)
iaj=ia(j)

dx=rar(i,1)-rar(j,1)
dy=rar(i,2)-rar(j,2)
dz=rar(i,3)-rar(j,3)
rrs=(dx**2+dy**2+dz**2)
xxa = sigg(ia(i),ia(j))**2/rrs
xxa = xxa*xxa*xxa
rda = dsqrt(rrs)

dx=rbr(i,1)-rbr(j,1)
dy=rbr(i,2)-rbr(j,2)
dz=rbr(i,3)-rbr(j,3)
rrs=(dx**2+dy**2+dz**2)
xxb = sigg(ib(i),ib(j))**2/rrs
xxb = xxb*xxb*xxb
rdb = dsqrt(rrs)

vb= 4.d0* epss( ibi,ibj )*(xxb*xxb-xxb)
vb= vb + qq( ibi )*qq( ibj )*coul/rdb

va= 4.d0* epss( iai,iaj )*(xxa*xxa-xxa)
va= va + qq( iai )*qq( iaj )*coul/rda

```

```

UB_aa = UB_aa + vb
UA_aa = UA_aa + va

c*****
c***** In calculating Ixij probe switch *****
c***** need 2 factor in last term so that *****
c***** the H+ of probe acid interactions *****
c***** with its own SO4-- are NOT counted *****
c***** as a probe interaction and are NOT *****
c***** turned off!! *****
c***** *****
c***** So, if both i,j are probe atoms *****
c***** in acid-acid interaction case *****
c***** switch must = 0! *****
c***** This will always work as long as *****
c***** there is only one acid probe. *****
c***** *****
c***** NOTE: Iprobe = (1,0) *****
c***** ==> (turned off, normal) *****
c***** ==> (in UAdel , not in UAdel) *****
c***** *****
c***** H2O-H2O has no internal interact- *****
c***** ions in this calculation. *****
c*****
c*****7**0***5*****0***5*****0***5*****0***5*****0***5*****0***5*****0***
c
c  Ixij=Iprobe(1,i)+Iprobe(1,j)-Iprobe(1,i)*Iprobe(1,j)*2
c  fIxij = dble(Ixij)

c***** find UBdel(jx) for alternate probes *****
  if(Mbdo.eq.0) then

    do jx=jlo,jhi
      IBij=Iprobe(jx,i)+Iprobe(jx,j)-
1      Iprobe(jx,i)*Iprobe(jx,j)*2
      fIBij = dfloat(IBij)
      UBdelf(jx) = UBdelf(jx) + fIBij*vb
c    if(jx.eq.m) fIxij = dfloat(IBij)
    end do

```

```

else
endif

c  IBijm=Iprobe(m,i)+Iprobe(m,j)-
c  1      Iprobe(m,i)*Iprobe(m,j)*2
c  flxij = dfloat(IBijm)

c*****
  jx=m
  IBij=Iprobe(jx,i)+Iprobe(jx,j)-
  1      Iprobe(jx,i)*Iprobe(jx,j)*2
  flxij = dfloat(IBij)

  UAdel_aa=UAdel_aa + flxij*va
  UBdel_aa=UBdel_aa + flxij*vb

end do
end do

c*****
c***** acid - water *****
c*****

  UB_aw=0.0d0
  UA_aw=0.0d0
  UBdel_aw=0.0d0
  UAdel_aw=0.0d0

  do i=1,nc
  do j=nc+1,na

  ibi=ib(i)
  ibj=ib(j)
  iai=ia(i)
  iaj=ia(j)

  dx=rar(i,1)-rar(j,1)
  dy=rar(i,2)-rar(j,2)
  dz=rar(i,3)-rar(j,3)
  rrs=(dx**2+dy**2+dz**2)
  xxa = sigg(ia(i),ia(j))**2/rrs
  xxa = xxa*xxa*xxa
  rda = dsqrt(rrs)

```

```

dx=rbr(i,1)-rbr(j,1)
dy=rbr(i,2)-rbr(j,2)
dz=rbr(i,3)-rbr(j,3)
rrs=(dx**2+dy**2+dz**2)
xxb = sigg(ib(i),ib(j))**2/rrs
xxb = xxb*xxb*xxb
rdb = dsqrt(rrs)

vb= 4.d0* epss( ibi,ibj )*(xxb*xxb-xxb)
vb= vb + qq( ibi )*qq( ibj )*coul/rdb

va= 4.d0* epss( iai,iaj )*(xxa*xxa-xxa)
va= va + qq( iai )*qq( iaj )*coul/rda

UB_aw=UB_aw + vb
UA_aw=UA_aw + va

c*****
c***** find UBdel(jx) for alternate probes *****
c***** to make water probes the original *****
c***** probes, set (jxw,j) = (m,j) *****
c*****

if(MBdo.eq.0) then

do jx=jlo,jhi
jxw=jx
IBij=Iprobe(jx,i)+Iprobe(jxw,j)-
1 Iprobe(jx,i)*Iprobe(jxw,j)
fIBij = dfloat(IBij)
UBdelf(jx) = UBdelf(jx) + fIBij*vb
c If(jx.eq.m) fIxij = dfloat(IBij)
end do

else
endif

c IBijm=Iprobe(m,i)+Iprobe(m,j)-
c 1 Iprobe(m,i)*Iprobe(m,j)*2
c fIxij = dfloat(IBijm)
c*****
jx=m

```

```

jxw=m
IBij=Iprobe(jx,i)+Iprobe(jxw,j)-
1 Iprobe(jx,i)*Iprobe(jxw,j)
flxij = dfloat(Ibij)

UBdel_aw=UBdel_aw + flxij*vb
UAdel_aw=UAdel_aw + flxij*va

end do
end do

c ***** end if on nac = 0 *****
else
endif

c*****
c***** all water- all water *****
c*****
c***** water atoms: nc + : 123 456 789 10,11,12 ***
c*****

UB_ww=0.0d0
UA_ww=0.0d0
UBdel_ww=0.0d0
UAdel_ww=0.0d0

c.....
c..... TIP4P additions 10-98 B.Hale .....
c..... pointm returns: .....
c..... m point ==> rma(water_atom,ii) .....
c..... H point ==> rma(water_atom+1,ii) .....
c..... H point ==> rma(water_atom+2,ii) .....
c..... (same for rmb) .....
c.....

call pointm(rar,rbr,rma,rmb)

c----- note sums over H2O MOLECULES! -----
do iw=1,k*m-1
do jw=iw+1,k*m
c-----
c***** io - nc = 1, 4, 7, 10 .... 3*(km-1)-2 ---
c***** iw = 1, 2, 3, 4 .... km-1 ---
c-----
io = m*nac + iw*3 -2

```

```

      ih1 = io+1
      ih2 = io+2
c***** jo-nc = 4, 7, 10, 13, ....
      jo = nac*m + jw*3 -2
      jh1 = jo+1
      jh2 = jo+2

c   write(30,9000) io,ih1,ih2,jo,jh1,jh2
c 9000 format('io,ih1,ih2, jo,jh1,jh2',6I3)

      dx =rar(io,1)-rar(jo,1)
      dy =rar(io,2)-rar(jo,2)
      dz =rar(io,3)-rar(jo,3)
      rooa =(dx**2+dy**2+dz**2)**3

      dx =rbr(io,1)-rbr(jo,1)
      dy =rbr(io,2)-rbr(jo,2)
      dz =rbr(io,3)-rbr(jo,3)
      roob =(dx**2+dy**2+dz**2)**3

c*****
c***** if LJ =1 calculate potential ***
c*****
      if(LJ.eq.1)then
      vb= voo4P( roob )
      va= voo4P( rooa )
      else
      endif

      if (LJ.eq.1) go to 4444

c*****
c***** skip if Lennard-Jones *****
c*****
c*****
      do ix =io,io+2
      do jx =jo,jo+2
      dx=rma(ix,1)-rma(jx,1)
      dy=rma(ix,2)-rma(jx,2)
      dz=rma(ix,3)-rma(jx,3)
      rra(ix,jx)=dsqrt(dx**2+dy**2+dz**2)

      dx=rmb(ix,1)-rmb(jx,1)
      dy=rmb(ix,2)-rmb(jx,2)
      dz=rmb(ix,3)-rmb(jx,3)

```



```

rrb(ixx,jxx)=dsqrt(dx**2+dy**2+dz**2)

end do
end do

```

```

c  vb = voo(rrb(io,jo))
c  write(30,9004) io,jo ,vb
c 9004 format('io,jo, vb',2I3, 2d24.16)

```

```

c..... skip rsl2 potentials .....
go to 3333

```

```

vb= voo( rrb( io,jo ) ) +
1  vhh( rrb( ih1,jh1 ) ) +
2  vhh( rrb( ih1,jh2 ) ) +
3  vhh( rrb( ih2,jh1 ) ) +
4  vhh( rrb( ih2,jh2 ) ) +
5  voh( rrb( io,jh1 ) ) +
6  voh( rrb( io,jh2 ) ) +
7  voh( rrb( ih1, jo ) ) +
8  voh( rrb( ih2, jo ) )

```

```

va= voo( rra( io,jo ) ) +
1  vhh( rra( ih1,jh1 ) ) +
2  vhh( rra( ih1,jh2 ) ) +
3  vhh( rra( ih2,jh1 ) ) +
4  vhh( rra( ih2,jh2 ) ) +
5  voh( rra( io,jh1 ) ) +
6  voh( rra( io,jh2 ) ) +
7  voh( rra( ih1, jo ) ) +
8  voh( rra( ih2, jo ) )

```

```

3333 continue

```

```

vb= voo4P( roob ) +
1  vhh4P( rrb( ih1,jh1 ) ) +
2  vhh4P( rrb( ih1,jh2 ) ) +
3  vhh4P( rrb( ih2,jh1 ) ) +
4  vhh4P( rrb( ih2,jh2 ) ) +
5  vmh4P( rrb( io,jh1 ) ) +
6  vmh4P( rrb( io,jh2 ) ) +
7  vmh4P( rrb( ih1, jo ) ) +
8  vmh4P( rrb( ih2, jo ) ) +

```

```

9  vmm4P( rrb( io, jo ) )

   va= voo4P( rooa ) +
1  vhh4P( rra( ih1,jh1 ) ) +
2  vhh4P( rra( ih1,jh2 ) ) +
3  vhh4P( rra( ih2,jh1 ) ) +
4  vhh4P( rra( ih2,jh2 ) ) +
5  vmh4P( rra( io,jh1 ) ) +
6  vmh4P( rra( io,jh2 ) ) +
7  vmh4P( rra( ih1, jo ) ) +
8  vmh4P( rra( ih2, jo ) ) +
9  vmm4P( rra( io, jo ) )
c*****
c*****
4444 continue
   UB_ww=UB_ww + vb
   UA_ww=UA_ww + va

c*****
c  Ixij=Iprobe(io)+Iprobe(jo)-Iprobe(io)*Iprobe(jo)
c  flxij = dble(Ixij)

c***** find UBdel(jx) for alternate probes *****

   if(MBdo.eq.0)then

   do jx=jlo,jhi

   IBij=Iprobe(jx,io)+Iprobe(jx,jo)-
1     Iprobe(jx,io)*Iprobe(jx,jo)
   flBij = dfloat(IBij)
   UBdelf(jx) = UBdelf(jx) + flBij*vb
c   if(jx.eq.m) flxij = dfloat(Ibij)
   end do
   else
   endif

c*****
c  Ixij=Iprobe(io)+Iprobe(jo)-Iprobe(io)*Iprobe(jo)
c  flxij = dble(Ixij)

   jx=m
   IBij=Iprobe(jx,io)+Iprobe(jx,jo)-
1     Iprobe(jx,io)*Iprobe(jx,jo)

```

```

flxij = dfloat(Ibij)

UBdel_ww=UBdel_ww + flxij*vb
UAdel_ww=UAdel_ww + flxij*va

end do
end do

c*****7**0***5****0****5****0****5****0****5****0****5****0****
c-----
c---*****
c---
c---
c--- Polarization interactions ---
c---
c---*****
c-----
c
  if(nac.ne.0)then

    do j=1,nac
      falph(j)=0.5d0
    end do

    if (istop.lt.nac) then

      do j=istop+1,nac
        falph(j)=1.0d0
      end do

    else
      endif

c--- end if(nac.ne.0) -----
    else
      endif

    UA_pol = 0.0d0
    UB_pol = 0.0d0

    UAp_aa = 0.0d0
    UAp_aw = 0.0d0
    UBp_aa = 0.0d0
    UBp_aw = 0.0d0

```

```

      UAdel_p= 0.0d0
      UBdel_p= 0.0d0
c*****
      if(nac.ne.0) then

          if(alpha.ne.0.0d0) then

c-----
c----- kk ==> O atom on the waters -----
c----- only these O atoms are polarized -----
c-----
          do kk=nac*m+1,na-2,3

c-----
c----- i ==> acid atoms: S, O O O O -----
c----- these SO4-- atoms do the polarizing -----
c-----

          do ii= 1,m

              do i = (ii-1)*nac+1, (ii-1)*nac + istop

                  dxkia=rar(kk,1)-rar(i,1)
                  dykia=rar(kk,2)-rar(i,2)
                  dzkia=rar(kk,3)-rar(i,3)

                  rrkia=(dxkia**2+dykia**2+dzkia**2)
                  rkia = dsqrt(rrkia)

                  dxkib=rbr(kk,1)-rbr(i,1)
                  dykib=rbr(kk,2)-rbr(i,2)
                  dzkib=rbr(kk,3)-rbr(i,3)

                  rrkib=(dxkib**2+dykib**2+dzkib**2)
                  rkib = dsqrt(rrkib)

                  iai = ia(i)
                  ibi = ib(i)

c----- put in E field due to acid atoms -----
c*****7**0***5*****0*****5*****0*****5*****0*****5*****0*****5*****0***

```

```

do jj= 1,m
do j= (jj-1)*nac+1, (jj-1)*nac +nac

    iaj = ia(j)
    ibj = ib(j)
c----- A -----
    dxkj=rar(kk,1)-rar(j,1)
    dykj=rar(kk,2)-rar(j,2)
    dzkj=rar(kk,3)-rar(j,3)

    rrkj=(dxkj**2+dykj**2+dzkj**2)
    rkj = dsqrt(rrkj)

    nf = j-(jj-1)*nac
    vap = -alpha*qq(iai)*qq(iaj)*coul*
2    (dxkia*dxkj+dykia*dykj+dzkia*dzkj)/
3    (rkia**3*rkj**3)*falph(nf)

    UAp_aa = UAp_aa + vap
c    write(30,*) kk,i,j,rkia,rkj,'vap,UAp_aa=',vap,UAp_aa

c----- B -----
    dxkj=rbr(kk,1)-rbr(j,1)
    dykj=rbr(kk,2)-rbr(j,2)
    dzkj=rbr(kk,3)-rbr(j,3)

    rrkj=(dxkj**2+dykj**2+dzkj**2)
    rkj = dsqrt(rrkj)

    vbp = -alpha*qq(ibi)*qq(ibj)*coul*
2    (dxkib*dxkj+dykib*dykj+dzkib*dzkj)/
3    (rkib**3*rkj**3)*falph(nf)

    UBp_aa = UBp_aa + vbp

c*****
    if(MBdo.eq.0) then

    do jx=jlo,jhi

    IBij=Iprobe(jx,i)+Iprobe(jx,j)+Iprobe(jx,kk)

```

```

1   -Iprobe(jx,i)*Iprobe(jx,j)
2   -Iprobe(jx,i)*Iprobe(jx,kk)
3   -Iprobe(jx,j)*Iprobe(jx,kk)
4   + Iprobe(jx,i)*Iprobe(jx,j)*Iprobe(jx,kk)

  flxij = dfloat(IBij)
  UBdelf(jx) = UBdelf(jx) + flxij*vbp
c   if(jx.eq.m) flxijk = dfloat(IBij)
  end do

  else
  endif

c*****
  jx=m
  IBij=Iprobe(jx,i)+Iprobe(jx,j)+Iprobe(jx,kk)
1   -Iprobe(jx,i)*Iprobe(jx,j)
2   -Iprobe(jx,i)*Iprobe(jx,kk)
3   -Iprobe(jx,j)*Iprobe(jx,kk)
4   + Iprobe(jx,i)*Iprobe(jx,j)*Iprobe(jx,kk)
  flxijk = dfloat(IBij)

  UAdel_p= UAdel_p + vap*flxijk
  UBdel_p= UBdel_p + vbp*flxijk

  end do
  end do

c-----
c----- E_kj field from other H2O -----
c----- NOTE: no factor of 1/2 here! -----
c-----
  do jj = nac*m+1,na-2,3
  do j = jj, jj+2
  if(jj.ne.kk) then
  iaj = ia(j)
  ibj = ib(j)
c----- A -----
  dxkj=rar(kk,1)-rar(j,1)
  dykj=rar(kk,2)-rar(j,2)
  dzkj=rar(kk,3)-rar(j,3)

  rrkj=(dxkj**2+dykj**2+dzkj**2)
  rkj = dsqrt(rrkj)

```

```

vap = -alpha*qq(iai)*qq(iaj)*coul*
2   (dxkia*dxkj+dykia*dykj+dzkia*dzkj)/
3   (rkia**3*rkj**3)

```

```

UAp_aw = UAp_aw + vap

```

```

c   write(30,*) kk,i,j,rkia,rkj,'vap,UAp_aw=',vap,UAp_aw

```

```

c----- B -----

```

```

dxkj=rbr(kk,1)-rbr(j,1)
dykj=rbr(kk,2)-rbr(j,2)
dzkj=rbr(kk,3)-rbr(j,3)

```

```

rrkj=(dxkj**2+dykj**2+dzkj**2)
rkj = dsqrt(rrkj)

```

```

vbp = -alpha*qq(ibi)*qq(ibj)*coul*
2   (dxkib*dxkj+dykib*dykj+dzkib*dzkj)/
3   (rkib**3*rkj**3)

```

```

UBp_aw = UBp_aw + vbp

```

```

c*****

```

```

if(MBdo.eq.0) then

```

```

do jx=jlo,jhi

```

```

IBij=Iprobe(jx,i)+Iprobe(jx,j)+Iprobe(jx,kk)

```

```

1   -Iprobe(jx,i)*Iprobe(jx,j)
2   -Iprobe(jx,i)*Iprobe(jx,kk)
3   -Iprobe(jx,j)*Iprobe(jx,kk)
4   + Iprobe(jx,i)*Iprobe(jx,j)*Iprobe(jx,kk)

```

```

flxij = dfloat(IBij)

```

```

UBdelf(jx) = UBdelf(jx) + flxij*vbp

```

```

c   if(jx.eq.m) flxijk = dfloat(IBij)
end do

```

```

else
endif

```

```

c*****
  jx=m
  IBij=Iprobe(jx,i)+Iprobe(jx,j)+Iprobe(jx,kk)
  1  -Iprobe(jx,i)*Iprobe(jx,j)
  2  -Iprobe(jx,i)*Iprobe(jx,kk)
  3  -Iprobe(jx,j)*Iprobe(jx,kk)
  4  + Iprobe(jx,i)*Iprobe(jx,j)*Iprobe(jx,kk)
  flxijk = dfloat(IBij)

  UAdel_p= UAdel_p + vap*flxijk
  UBdel_p= UBdel_p + vbp*flxijk

  else
  endif
  end do
  end do
c*****7**0***5****0****5****0****5****0****5****0****5****0****
c-----
c----- end do on ii, i -----
  end do
  end do

c-----
c---- The following is for print out only.---
c-----

  UA_pol = UAp_aa + UAp_aw
  UB_pol = UBp_aa + UBp_aw

c  write(30,*) kk,'/UB_pol=',UB_pol
c  write(30,*) kk,'/UA_pol=',UA_pol

c-----
c-----  END DO on kk (O in waters) -----
c-----

  end do

  UA_pol = UAp_aa + UAp_aw
  UB_pol = UBp_aa + UBp_aw

```



```

UA_px = UA_pol - UAdel_p
UB_px = UB_pol - UBdel_p

```

```

c-----
go to 3000

write(30,*) 'check on UAdel_p, UBdel_p'
write(30,*) '*****'
write(30,*) 'UAp_aa      = ', UAp_aa
write(30,*) 'UAp_aw      = ', UAp_aw
write(30,*) 'UA_pol total = ', UA_pol
write(30,*) 'UA_px non probe = ', UA_px
write(30,*) 'UAdel_p     = ', UAdel_p

write(30,*) 'UBp_aa      = ', UBp_aa
write(30,*) 'UBp_aw      = ', UBp_aw
write(30,*) 'UB_pol total = ', UB_pol
write(30,*) 'UB_px non probe = ', UB_px
write(30,*) 'UBdel_p     = ', UBdel_p
write(30,*) '*****'
3000 continue
c-----
c ----- end condition that alpha ne. 0.0 -----

else
endif

c----- end if(nac.ne.0) -----
else
endif
c*****
c*****

c-----
c----- end of polarization interactions -----
c-----
c----- Final values for interactions -----
c----- note: in B ensemble if lmb ne. 1 -----
c----- UB = UB-(1-lmb)UBdel -----
c-----
UB = UB_aa + UB_aw + UB_ww + UB_pol
UBdel = (UBdel_aa + UBdel_aw + UBdel_ww + UBdel_p)
c UBdelL= (1.0d0-lamb)*UBdel
UBdelL= (lmb-lamb)*UBdel

```

```

    if(MBdo.eq.0) then
    do jx=jlo,jhi
c   UBdelLf(jx) = (1.0d0-lamb)*UBdelf(jx)
   UBdelLf(jx) = (lmb-lamb)*UBdelf(jx)
    end do
    else
    endif

c   if(MBstep.le.10) then
c   ddUBx=UBdel-UBdelf(m)
c   if(ddUBx.ne.0.0d0)write(30,*)'UBdel-UBdelf(m) = ',ddUBx

c   else
c   endif

    if(lmb.ne.1.0d0)then
    UB = UB-(1.0d0 -lmb)*Ubdelf(m)
    UBL = UB-UBdelLf(m)
    else
    UBL = UB-UBdelL
    endif

    UA = UA_aa + UA_aw + UA_ww + UA_pol
    UAdel = UAdel_aa + UAdel_aw + UAdel_ww + UAdel_p
c   UAdelL= (1.0d0-lamb)*UAdel
    UAdelL= (lmb-lamb)*UAdel
c.....
c..... UAL must be the weighting function for A ensemble ....
c..... so UAL = UA -(1-lamb)UAdel          ....
c.....
c   UAL = UA-UAdelL
    UAL = UA- (1.0d0-lamb)*UAdel

c   write(30,*)'pwsom UBdelf(jx), jx=1,jup; UBdel'
c   write(30,*)(UBdelf(jx),jx=1,jup),UBdel
c   write(30,*)'pwsom UBdelLf(jx), do jx=1,jup; UBdelL'
c   write(30,*)(UBdelLf(jx),jx=1,jup),UBdelL

c   write(30,*) 'END OF pwsom'
c   write(30,*) 'UAL= ',UAL,'   UB= ',UB
    return
    end

```

C*****

```

SUBROUTINE infout(mstep,xxacc,xxbcc)
implicit real*8 (a-h,o-z)
real*8 lamb,lmb,lb,mb,mas
parameter(MAT=600)
parameter(MAM=600)
parameter(MAB=600)
COMMON/R/ ra(MAT,3),rb(MAT,3),ia(MAT),ib(MAT)
1      ,imola(MAT),imolb(MAT),rcma(3),rcmb(3)
COMMON/seeds/iseeda(7),iseedb(7)
c COMMON/CONST/ kb,coul,pi,mi,ic,kall,alpha,istop,qf,qw,alp
COMMON/CONST/ kb,coul,pi,alpha,qf,qw,alp,ic,kall,istop,mi
COMMON/STEP/mstart,mmax,mprint,MBdo,MBstep
c COMMON/km/k,m,na,nac,nwa,lamb,lmb,disp,beta,dd,Rsqa,jup,disa(MAT)
c 1 ,jlo,jhi,jxtot,LJ
COMMON/km/lamb,lmb,disp,beta,dd,Rsqa,disa(MAT),k,m,na,nac,nwa,jup,
1 jlo,jhi,jxtot,LJ

COMMON/SIG/is(MAT),sig(8),eps(8),q(8),mass(8),Iprobe(MAM,MAT)
COMMON/UAUB/ UA_aa,UB_aa,UAdel_aa,UBdel_aa,
1      UA_aw,UB_aw,UAdel_aw,UBdel_aw,
2      UA_ww,UB_ww,UAdel_ww,UBdel_ww,
6      UAp_aa,UAp_aw,UBp_aa,UBp_aw,UA_px,UB_px,
3      UA, UB, UAdel, UBdel,UBdelf(MAM),
4      UAL,UBL,UAdelL,UBdelL,UBdelLf(MAM),
5      UA_pol,UB_pol,UAdel_p,UBdel_p
6      ,binUA(MAB),binUB(MAB)

COMMON/SUMS/ UAL_sum,UB_sum,UAdL_sum,UBdL_sum,UBdf_sum
COMMON/FAFB/ FA(20),FB(MAM,20),c(20),
1      FA_sum(20),FB_sum(MAM,20),
2      FA_av(20) ,FB_av(MAM,20),MB

COMMON/DATA/cfin,cplot,cav,cp_ave,UAL_av,UB_av,UAdel_av,UBdel_av
1 ,UBdf_av
double precision xxacc,xxbcc,FA_ln(20),FB_ln(MAM,20),
1 FB_pav(20),FB_lnv(20),cf_m(MAM)
c*****
write(30,8009)
if(mstep.eq.mstart) write(30,8000) mstep
if(mstep.gt.mstart) write(30,8008) mstep,MBstep
write(30,8010) xxacc,xxbcc
write(30,8001)
write(30,8002) UA_aa,UB_aa,UAdel_aa,UBdel_aa
write(30,8003) UA_aw,UB_aw,UAdel_aw,UBdel_aw

```

```

write(30,8004) UA_ww,UB_ww,UAdel_ww,UBdel_ww
write(30,8014) UA_pol,UB_pol,UAdel_p,UBdel_p
write(30,8005) UA, UB, UAdel, UBdel
write(30,8006) UAL, UBL, UAdelL, UBdelL

```

```

c*****
c***** DATA for checking with Shawn's results ***
c***** Shawn multiplies all his energies in the A ensemble ***
c***** by (lambda-1). Not so in B ensemble. ***
c*****

```

```

if(k.eq.1) then

```

```

c-----below for A revised 3-26-97 -----

```

```

UBpolLx= -0.158225333082609*(1.0d0-lamb)
UApolLx= +0.7047565889487516d0*(1.0d0-lamb)

```

```

UALx = -56.4759290709947d0+UApolLx
UBx = -92.6723678915287d0

```

```

UAdelLx= 0.546819939230155d0
UBdelLx= -44.33645745022018d0

```

```

else

```

```

UALx = -88.2000794744995d0
UBx = -165.077772224778d0
UAdelLx= 102.889999183813d0
UBdelLx= -116.4891217254737d0
endif

```

```

chkUAL = UAL - UALx
chkUB = UB - UBx
cUBdelL= UBdelL - UBdelLx
cUAdelL= UAdelL - UAdelLx

```

```

c***** use below if mstep > 0 *****
if(MBdo.ne.0) then
write(30,*) 'STOP: MB and mprint not compatible'
else
endif

```

```

if(mstep.gt.1)then
  MBBstep=MBstep
else
  MBBstep=1
endif

if(mstep.ne.0) then

  UAL_av  = UAL_sum/dfloat(mstep)
  UB_av   = UB_sum/dfloat(mstep)
  UAdeL_av = UAdL_sum/dfloat(mstep)
  UBdeL_av = UBdL_sum/dfloat(mstep)
  UBdf_av = UBdf_sum/MBBstep

  do icc=1,20
    FA_av(icc) = FA_sum(icc)/dfloat(mstep)
    FA_ln(icc) = dlog(FA_av(icc))

    if(MBdo.eq.0) then

      FB_pav(icc) = 0.0d0
      FB_lnv(icc) = 0.0d0

      do jx=jlo,jhi

        FB_av(jx,icc) = FB_sum(jx,icc)/dfloat(MBBstep)
        FB_ln(jx,icc) = dlog(FB_av(jx,icc))
        FB_pav(icc) = FB_pav(icc) + FB_av(jx,icc)
      end do
    else
    endif

    FB_pav(icc) = FB_pav(icc)/dfloat(jxtot)
    FB_lnv(icc)=dlog(FB_pav(icc))

  end do
c*****7**0***5****0****5****0****5****0****5****0****5****0****5****0***

  cav=0.0d0

```

```

do jx=jlo,jhi
mp=jx
call cfind_m(mstep,cfindm,mp,jmi)
cf_m(jx)=cfindm
cav= cav + cfindm/dfloat(jxtot)
end do

```

```

mp=m
call cfind(mstep,cfinal,mp,jfi)

```

```

cfin = cfinal
cfindm=cf_m(mp)

```

```

c***** below used if mstep = 0 *****

```

```

else

```

```

UAL_av = UAL
UB_av = UB
UAdeL_av = UAdeL
UBdeL_av = UBdeL
UBdf_av = UBdf_sum
UbdF = UBdf_av

```

```

do icc = 1,20
FA_av(icc) = FA(icc)
FA_ln(icc) = 0.0d0
FB_pav(icc)=0.0d0
FB_lnv(icc) = 0.0d0

```

```

do jx=jlo,jhi
FB_av(jx,icc) = FB(jx,icc)
FB_ln(jx,icc) = dlog( FB_av(jx,icc))
FB_pav(icc) = FB_pav(icc) + FB_av(jx,icc)/dfloat(jxtot)
end do

```

```

FB_lnv(icc)= dlog(FB_pav(icc))
end do

```

```

cfinal = 0.0d0
cfindm = 0.0d0

```

```

endif

```

```

c***** below carried out for all mstep values *****

UAxx = 0.0d0
UBxx = 0.0d0
UAdxx = 0.0d0

write(30,8015) UAxx, UBxx, UAdxx, UBdf_av
c  write(30,*)'***** <UB_delf> above *****'

c  write(30,8007) chkUAL, chkUB, cUAdelL, cUBdelL
write(30,8011) UAL_av, UB_av, UAdel_av,UBdel_av
write(30,8009)

write(30,*) 'check on UAdel_p, UBdel_p'
write(30,*) '*****'
write(30,*) 'UAp_aa      =', UAp_aa
write(30,*) 'UAp_aw      =', UAp_aw
write(30,*) 'UA_pol total =', UA_pol
write(30,*) 'UA_px non probe =', UA_px
write(30,*) 'UAdel_p     =', UAdel_p

write(30,*) 'UBp_aa      =', UBp_aa
write(30,*) 'UBp_aw      =', UBp_aw
write(30,*) 'UB_pol total =', UB_pol
write(30,*) 'UB_px non probe =', UB_px
write(30,*) 'UBdel_p     =', UBdel_p
write(30,*) '*****'

write(30,8009)
c*****7**0***5*****0***5*****0***5*****0***5*****0***5*****0***5*****0***5*****0**
do icc=1,20
fblv=FB_lnv(icc)
faln=FA_ln(icc)
fbln=FB_ln(m,icc)

write(30,8012)c(icc),FA_av(icc),FB_pav(icc),FA_ln(icc),fblv

end do
write(30,8009)

zk=dfloat(k)
zm=dfloat(m)

if(m.eq.1) then

```

```

zeta = zk*dlog(zk*zm) +
1 - zk*zm*(dlog(zk*zm)-1.0d0)
else

zeta = zk*dlog(zk*zm) +
1 zk*(zm-1.0d0)*( dlog( zk*(zm-1.0d0) ) -1.0d0 )
1 - zk*zm*(dlog(zk*zm)-1.0d0)
endif

if(nac.eq.0) zeta = 0.0d0
if(nac.eq.0) zk=0.0d0

if(k.eq.1) zeta=0.0d0

cplot = ( cfinal + zeta)/(zk+1.0d0) +dlog(alp)
cplotm = ( cfindm + zeta)/(zk+1.0d0) +dlog(alp)
cp_ave = ( cav + zeta)/(zk+1.0d0) +dlog(alp)
c-----
write(30,*) '***** C from Average of FB functions *****'
write(30,8013) cfinal,cplot,mstep
write(30,*) '***** Old C value (last atom probes): *****'
write(30,8013) cfindm,cplotm,mstep
write(30,*) '***** Average of C, <lnFB(jx)>, values *****'
write(30,8013) cav,cp_ave,mstep

c write(30,*)'FB_ln(jx,icc=ji), jx=1,jup ', ' ji = ',jfi
c write(30,*) (FB_ln(jx,jfi),jx=1,jup),' FA_ln= ',FA_ln(jfi)
write(30,8009)
write(30,*)'cf_m(jx) = ',(cf_m(jx),jx=jlo,jhi),' Cave = ',cav,
1 ' cav_plot = ',cp_ave
c-----
8000 format('STEP NUMBER = ',I10,' INITIAL SUMS FOR A AND B')
8001 format(10x,'UA--',21x,'UB--',21x,'UAdel--',20x,'UBdel--')
8002 format(' aa:',4(1x,D24.16) )
8003 format(' aw:',4(1x,D24.16) )
8004 format(' ww:',4(1x,D24.16) )
8014 format(' pol:',4(1x,D24.16),/)
8015 format('UBf>:',4(1x,D24.16) )
8005 format('tot :',4(1x,D24.16) )
8006 format('totL:',4(1x,D24.16),/)
8007 format('chek:',4(1x,D24.16) )
8011 format('<U> :',4(1x,D24.16) )
8012 format('<FA>:',f5.1,4(1x,D24.16))
8013 format('Cfin:',d24.16,' Cplot: ',d24.16,' [ step = ',I10,' ],/')
8008 format('STEP =',i10,' MBstep = ',i10)

```



```

8010 format(' Xacc= ',d24.16,' Xbcc = ',d24.16,' SUMS FOR A and B')
8009 format('*****')
1 *****')
return
end
c*****7**0***5****0****5****0****5****0****5****0****5****0****5****0****5****0****5****0**
SUBROUTINE cmas(rxa,rxb,rcma,rcmb,iflag,irsa,irsb)
implicit real*8 (a-h,o-z)
real*8 lamb,lmb,mass
parameter(MAT=600)
parameter(MAM=600)
COMMON/R/ ra(MAT,3),rb(MAT,3),ia(MAT),ib(MAT)
1 ,imola(MAT),imolb(MAT),rcma(3),rcmb(3)
c COMMON/km/k,m,na,nac,nwa,lamb,lmb,disp,beta,dd,Rsqa,jup,disa(MAT)
c 1 ,jlo,jhi,jxtot,LJ
COMMON/km/lamb,lmb,disp,beta,dd,Rsqa,disa(MAT),k,m,na,nac,nwa,jup,
1 jlo,jhi,jxtot,LJ

COMMON/STEP/mstart,mmax,mprint,MBdo,MBstep
COMMON/SIG/is(MAT),sig(8),eps(8),q(8),mass(8),Iprobe(MAM,MAT)
COMMON/FMASS/fmass(MAT),fmasstot,fmasstota
double precision rxa(MAT,3),rxb(MAT,3)
double precision rcma(3),rcmb(3)
c-----
c----- mass(1) = S, mass(2) = O, mass(4) = H -----
c----- assign masses to the atoms -----
c-----
ffc=1.0d0

if(iflag.ne.0) go to 100
c-----
if(nac.ne.0) then

do mi=1,m
msulf = nac*(mi-1)+1
fmass(msulf) = mass(1)

do j=1,4
fmass(msulf + j) = mass(2)
end do
do j = 1,2
fmass(msulf + 4 + j) = mass(4)
end do

end do

```

```

c----- end if(nac.ne.0) -----
  else
  ffc=0.0d0
  endif

  do kk=1,k*m
  mox = nac*m+ 3*(kk-1)+1
  fmass(mox)= mass(2)
  fmass(mox +1)= mass(4)
  fmass(mox +2)= mass(4)
  end do

  fmasstota = m*ffc*( mass(1) + 4.0d0*mass(2)
1      +      2.0d0*mass(4) )
2      + k*m*( mass(2) + 2.0d0*mass(4) )

  fmasstot = 0.0d0
  do iatm=1,na
  fmasstot = fmasstot + fmass(iatm)
  end do

  write(30,1000)fmasstota, fmasstot
1000 format( 'check on total mass',2(d24.16,1x))

  if(mstart.gt.0) return
c-----
c----- initialize sums to 0.0 -----

100 continue
  do i=1,3
  rccma(i) = 0.0d0
  rccmb(i) = 0.0d0
  end do

c----- calculate center of mass -----
  irsa=0
  irsb=0
  If(nac.eq.0) fmasstota=fmasstot
  do iatm = 1,na

  do j = 1,3

```

```

rccma(j) = rccma(j) + rxa(iatm,j)*fmass(iatm)/fmasstota
rccmb(j) = rccmb(j) + rxb(iatm,j)*fmass(iatm)/fmasstota
end do

end do

c----- translate center of mass to origin -----
c----- and flag with irsa, irsb if atom is out -----
c----- O needs tag if system is LJ -----

do iatm = 1,na
  rsa=0d0
  rsb=0d0

  fLJa=0.0d0
  fLJb=0.0d0
  if(LJ.eq.1) fLJa= dfloat(6-ia(iatm))
  if(LJ.eq.1) fLJb= dfloat(6-ib(iatm))

c*****
c***** (LJa,LJb) = (1,0) for ia = (5,6) = (O,H) ***
c***** sets rsquared = 0 for hydrogens; no rejection for being out ***
c*****

do j = 1,3
  rxa(iatm,j) = -rccma(j) + rxa(iatm,j)
  rxb(iatm,j) = -rccmb(j) + rxb(iatm,j)

  rsa=rsa+fLJa*rx(iatm,j)**2
  rsb=rsb+fLJb*rx(iatm,j)**2
end do

if(rsa.gt.Rsqa) irsa=iatm
if(rsb.gt.Rsqa) irsb=iatm

end do

return
end
C*****

```

```

SUBROUTINE cfind(mmstep,cfinal,mp,ji)
implicit real*8 (a-h,o-z)
real*8 lamb,lmb
parameter(MAT=600)
parameter(MAM=600)
COMMON/FAFB/ FA(20),FB(MAM,20),c(20),
1      FA_sum(20),FB_sum(MAM,20),
2      FA_av(20), FB_av(MAM,20),MB
c  COMMON/km/k,m,na,nac,nwa,lamb,lmb,disp,beta,dd,Rsqa,jup,disa(MAT)
c  1 ,jlo,jhi,jxtot,LJ
COMMON/km/lamb,lmb,disp,beta,dd,Rsqa,disa(MAT),k,m,na,nac,nwa,jup,
1 jlo,jhi,jxtot,LJ

double precision xA(20),xB(20),dAB(20),rat(20)
double precision FB_xsum(20),FB_avx(20)

if(mmstep.le.1) then
  cfinal = 0.0d0
  return
else

  ji = 1
  do j=1,20

    FB_xsum(j) =0.0d0

    do jx=jlo,jhi
      FB_xsum(j) =FB_xsum(j) + FB_av(jx,j)
    end do

    FB_avx(j) =FB_xsum(j) /dfloat(jxtot)

    xA(j) = dlog( FA_av(j) )*(-1.0d0)
c   xB(j) = dlog( FB_av(j) )*(-1.0d0)
    xB(j) = dlog( FB_avx(j) )*(-1.0d0)
    dAB(j) =(xB(j)-xA(j))

    if (j.gt.1) then
      rat(j) = dAB(j)/dAB(j-1)
      jsave=ji
      if (rat(j).lt.0.0d0) ji = j
c   write(30,*) 'j,ji',j,ji,'dAB(j)',dAB(j)
    else
    endif
  end do

```

```

end do

if(ji.gt.20)then
write(30,*)'c(ji) index OUT OF BOUNDS! '
write(30,*) jup,' = jup'
write(30,*) 'j,jj',j,jj,'dAB(j)',dAB(j)
ji = 10
else
continue
endif

if(ji.eq.1) ji = 20
j0=ji-1

deltac = c(ji)-c(j0)
slopea = ( xA(ji)-xA(j0))/deltac
slopeb = ( xB(ji)-xB(j0))/deltac
delC = (xB(j0) -xA(j0))/(slopea-slopeb)

cfinal = c(j0) + delC

endif
if(ji.eq.0)write(30,*) 'ERROR: ji = 0'

return
end
C*****
SUBROUTINE cfind_m(mmstep,cfinal,mp,ji)
implicit real*8 (a-h,o-z)
real*8 lamb,lmb
parameter(MAT=600)
parameter(MAM=600)
COMMON/FAFB/ FA(20),FB(MAM,20),c(20),
1          FA_sum(20),FB_sum(MAM,20),
2          FA_av(20), FB_av(MAM,20),MB
c  COMMON/km/k,m,na,nac,nwa,lamb,lmb,disp,beta,dd,Rsqa,jup,disa(MAT)
c  1 jlo,jhi,jxtot,LJ
COMMON/km/lamb,lmb,disp,beta,dd,Rsqa,disa(MAT),k,m,na,nac,nwa,jup,
1 jlo,jhi,jxtot,LJ

double precision xA(20),xB(20),dAB(20),rat(20)
double precision FB_xsum(20),FB_avx(20)

if(mmstep.le.1) then

```

```

cfinal = 0.0d0
return
else

ji = 1
do j=1,20

FB_xsum(j) =0.0d0

mmx=0
do jx=mp,mp
FB_xsum(j) =FB_xsum(j) + FB_av(jx,j)
mmx=mmx+1
end do

FB_avx(j) =FB_xsum(j) /dfloat(mmx)

xA(j) = dlog( FA_av(j) )*(-1.0d0)
c  xB(j) = dlog( FB_av(j) )*(-1.0d0)
xB(j) = dlog( FB_avx(j) )*(-1.0d0)
dAB(j) =(xB(j)-xA(j))

if (j.gt.1) then
rat(j) = dAB(j)/dAB(j-1)
jsave=ji
if (rat(j).lt.0.0d0) ji = j
c  write(30,*) 'j,ji',j,ji,'dAB(j)',dAB(j)
else
endif

end do

if(ji.eq.1) ji = 20
j0=ji-1

deltac = c(ji)-c(j0)
slopea = ( xA(ji)-xA(j0))/deltac
slopeb = ( xB(ji)-xB(j0))/deltac
delC = (xB(j0) -xA(j0))/(slopea-slopeb)

cfinal = c(j0) + delC

endif

return

```

end

```

C*****
SUBROUTINE test
implicit real*8 (a-h,o-z)
double precision x(3),dx(3),dxcalc(3)
x(1) = 1d0
x(2) = 2d0
x(3) = 3d0
ixa = 2
theta = 3.141592653589793D0/6.0d0
dac = dcos(theta)
das = dsin(theta)
call rotate(ixa,x,dx,dac,das)
dxcalc(1) = 1.0d0*dac+3.0d0*das
dxcalc(2) = 2d0
dxcalc(3) = -1.0d0*das + 3.0d0*dac
write(30,1000) (dxcalc(i),i=1,3)
write(30,1001) (dx(i),i=1,3)
1000 format('calculated: ',3(d24.16,1x))
1001 format('rotated: ',3(d24.16,1x))
return
end
C*****
c*****7**0***5*****0****5*****0****5*****0****5*****0****5*****0****5*****0****5*****0**
SUBROUTINE rasmolh(nstep)

implicit real*8 (a-h,o-z)
real*8 lamb,lmb,kb
parameter(MAT=600)
parameter(MAM=600)

COMMON/R/ ra(MAT,3),rb(MAT,3),ia(MAT),ib(MAT)
1 ,imola(MAT),imolb(MAT),rcma(3),rcmb(3)
c COMMON/CONST/ kb,coul,pi,mi,ic,kall,alpha,istop,qf,qw,alp
COMMON/CONST/ kb,coul,pi,alpha,qf,qw,alp,ic,kall,istop,mi
COMMON/STEP/mstart,mmax,mprint,MBdo,MBstep
c COMMON/km/k,m,na,nac,nwa,lamb,lmb,disp,beta,dd,Rsqa,jup,disa(MAT)
c 1 ,jlo,jhi,jxtot,LJ
COMMON/km/lamb,lmb,disp,beta,dd,Rsqa,disa(MAT),k,m,na,nac,nwa,jup,
1 jlo,jhi,jxtot,LJ

nwat=k*m
nacid=m

```

```

if(nac.eq.0) nacid=0

write(40,*) 'HEADER SULFURIC ACID AND WATER: ',nstep,' STEPS'
write(40,*) ' COMPND:',nacid,'-H2SO4 +',nwat,'-H2O',' Qf = ',qf,
1 ' alpha = ',alpha,' qw = ',qw
write(50,*) 'HEADER SULFURIC ACID AND WATER: ',nstep,' STEPS'
write(50,*) ' COMPND:',nacid,'-H2SO4 +',nwat,'-H2O',' Qf = ',qf,
1 ' alpha = ',alpha,' qw = ',qw

write(30,*)'in rasmolh',k,' = k ',m,' = m'

nat = m*nac+k*m*3

if(nac.ne.0)then
c-----
  do im = 1,m
  do ii=1,nac
  i = (im-1)*nac +ii

  ix=i
c   if(i.gt.99)ix=i-100
c   if(i.gt.199)ix=i-200

  if(ii.eq.1) then
  write(40,100) ix,(ra(i,kx),kx=1,3),ix
  write(50,100) ix,(rb(i,kx),kx=1,3),ix
  else

  if(ii.le.5) then
  write(40,200) ix,(ra(i,kx),kx=1,3),ix
  write(50,200) ix,(rb(i,kx),kx=1,3),ix
  else
  write(40,300) ix,(ra(i,kx),kx=1,3),ix
  write(50,300) ix,(rb(i,kx),kx=1,3),ix
  endif

  endif

  end do
  end do

c----- end if(nac.ne.0)-----

  else
  endif

```



```

do ik = 1,k*m
do ii = 1,3
i = nac*m + (ik-1)*3 +ii

ix=i
c  if(i.gt.99)ix=i-100
c  if(i.gt.199)ix=i-200

if(ii.eq.1) then
write(40,400) ix,(ra(i,kx),kx=1,3),ix
write(50,400) ix,(rb(i,kx),kx=1,3),ix
else
c----- If LJ case, don't write H coordinates to rasmol file -----
if(LJ.eq.0)then
write(40,500) ix,(ra(i,kx),kx=1,3),ix
write(50,500) ix,(rb(i,kx),kx=1,3),ix
else
endif

endif

end do
end do

ix=nat
zze=0.0d0
rsp=dsqrt(Rsqa)

do iy=1,2
fact=(-1.0d0)**iy
rrr=rsp*fact
ix=nat+iy
write(40,600) ix,rrr,zze,zze
write(50,600) ix,rrr,zze,zze
ix = nat+1+iy
write(40,600) ix,zze,rrr,zze
write(50,600) ix,zze,rrr,zze
ix= nat +2 +iy
write(40,600) ix,zze,zze,rrr
write(50,600) ix,zze,zze,rrr
end do

```

```

write(40,*)'END'
write(50,*)'END'

100  format('ATOM  ',i3,2x,'S  ACD  1',5x,3(f7.3,1x),
&  ' 1.00  .00 ',i3)
200  format('ATOM  ',i3,2x,'O  ACD  1',5x,3(f7.3,1x),
&  ' 1.00  .00 ',i3)
300  format('ATOM  ',i3,2x,'BI ACD  2',5x,3(f7.3,1x),
&  ' 1.00  .00 ',i3)
400  format('ATOM  ',i3,2x,'NA WAT  3',5x,3(f7.3,1x),
&  ' 1.00  .00 ',i3)
500  format('ATOM  ',i3,2x,'H  WAT  3',5x,3(f7.3,1x),
&  ' 1.00  .00 ',i3)
600  format('ATOM  ',i3,2x,'AG SPH  3',5x,3(f7.3,1x),
&  ' 0.10  .00 ',i3)

write(30,*) 'Wrote Rasmol coordinate files'
return
end
c*****
SUBROUTINE cfg_out(Tx,mstep)
implicit real*8 (a-h,o-z)
real*8 lamb,lmb,kb,mass
parameter(MAT=600)
parameter(MAM=600)
parameter(MAB=600)
COMMON/R/ ra(MAT,3),rb(MAT,3),ia(MAT),ib(MAT)
1  ,imola(MAT),imolb(MAT),rcma(3),rcmb(3)
COMMON/seeds/iseeda(7),iseedb(7)
c  COMMON/CONST/ kb,coul,pi,mi,ic,kall,alpha,istop,qf,qw,alp
COMMON/CONST/ kb,coul,pi,alpha,qf,qw,alp,ic,kall,istop,mi
COMMON/STEP/mstart,mmax,mprint,MBdo,MBstep
c  COMMON/km/k,m,na,nac,nwa,lamb,lmb,disp,beta,dd,Rsqa,jup,disa(MAT)
c  1  ,jlo,jhi,jxtot,LJ
COMMON/km/lamb,lmb,disp,beta,dd,Rsqa,disa(MAT),k,m,na,nac,nwa,jup,
1  jlo,jhi,jxtot,LJ

COMMON/SIG/is(MAT),sig(8),eps(8),q(8),mass(8),Iprobe(MAM,MAT)

COMMON/UAUB/ UA_aa,UB_aa,UAdel_aa,UBdel_aa,
1  UA_aw,UB_aw,UAdel_aw,UBdel_aw,
2  UA_ww,UB_ww,UAdel_ww,UBdel_ww,
6  UAp_aa,UAp_aw,UBp_aa,UBp_aw,UA_px,UB_px,
3  UA, UB, UAdel, UBdel,UBdelf(MAM),

```

```

4      UAL,UBL,UAdelL,UBdelL,UBdelLf(MAM),
5      UA_pol,UB_pol,UAdel_p,UBdel_p
6      ,binUA(MAB),binUB(MAB)

COMMON/COUNTS/ icounta(MAT),icountb(MAT),It_max
COMMON/SUMS/  UAL_sum,UB_sum,UAdL_sum,UBdL_sum,UBdf_sum
COMMON/FAFB/  FA(20),FB(MAM,20),c(20),
1      FA_sum(20),FB_sum(MAM,20),
2      FA_av(20) ,FB_av(MAM,20),MB
COMMON/DATA/cfin,cplot,cav,cp_ave,UAL_av,UB_av,UAdel_av,UBdel_av
1 ,UBdf_av
c*****7**0***5****0****5****0****5****0****5****0****5****0****5****0****5****0**
1015 format(5(d24.16,1x))
c1009 format(1x,3(d24.16,1x),2x,2(I2,2x))
c in cfg.out
1009 format(1x,3(d24.16,1x),I1,1x,I3)
1016 format(5(f12.1,1x))
1017 format(5(i10,1x))
      write(60,*) mstep,k,m,Tx,'d0 ',alpha,'d0 ',istop,
1 qf,'d0 ',qw,'d0 ',alp,'d0',' C: ',cplot,cp_ave

c   write(60,*) mstep,k,m,Tx,alpha,istop,qf,qw,alp

write(60,*) (iseeda(i), i=1,5)
write(60,*) (iseeda(i), i=6,7)
write(60,*) (iseedb(i), i=1,5)
write(60,*) (iseedb(i), i=6,7)

do i=1,na
write(60,1009)  (ra(i,j), j = 1,3),ia(i),imola(i)
end do

do i=1,na
write(60,1009)  (rb(i,j), j = 1,3),ib(i),imolb(i)
end do

c   write(60,1015) UA_aa,UB_aa,UAdel_aa,UBdel_aa,
c   1      UA_aw,UB_aw,UAdel_aw,UBdel_aw,
c   2      UA_ww,UB_ww,UAdel_ww,UBdel_ww,
c   6      UAp_aa,UAp_aw,UBp_aa,UBp_aw,UA_px,UB_px,
c   3      UA, UB, UAdel, UBdel, UBdelf,
c   4      UAL,UBL,UAdelL,UBdelL,UBdelLf,
c   5      UA_pol,UB_pol,UAdel_p,UBdel_p

```

```

c 6      ,binUA,binUB

write(60,1015) UAL_sum, UB_sum,UAdL_sum,UBdL_sum,UBdf_sum
write(60,1017) iacc_sum, ibcc_sum
write(60,1015) (c(icc),icc=1,20)

write(60,1015) (FA(icc),icc=1,20)
do jx=1,jup
write(60,1015) (FB(jx,icc),icc=1,20)
end do

write(60,1015) (FA_sum(icc),icc=1,20)
do jx=1,jup
write(60,1015) (FB_sum(jx,icc),icc=1,20)
end do
return
end

C*****
SUBROUTINE angle(anghoh,roh)
c*****
c***** this program restores HOH angles to anghoh and roh to 0.9584 ***
c***** 8-4-95 B. Hale
c***** 6-27-97 converted for BinMC_MPx.f ***
c*****
implicit real*8 (a-h,o-z)
real*8 lamb,lmb,kb
parameter(MAT=600)
parameter(MAM=600)

COMMON/R/ ra(MAT,3),rb(MAT,3),ia(MAT),ib(MAT)
1 ,imola(MAT),imolb(MAT),rcma(3),rcmb(3)
c COMMON/CONST/ kb,coul,pi,mi,ic,kall,alpha,istop,qf,qw,alp
COMMON/CONST/ kb,coul,pi,alpha,qf,qw,alp,ic,kall,istop,mi
COMMON/STEP/mstart,mmax,mprint,MBdo,MBstep
c COMMON/km/k,m,na,nac,nwa,lamb,lmb,disp,beta,dd,Rsqa,jup,disa(MAT)
c 1 ,jlo,jhi,jxtot,LJ
COMMON/km/lamb,lmb,disp,beta,dd,Rsqa,disa(MAT),k,m,na,nac,nwa,jup,
1 jlo,jhi,jxtot,LJ

dimension ha1(MAT,3),ha2(MAT,3),hb1(MAT,3),hb2(MAT,3)

c roh=0.9584d0
c anghoh=104.45D0
angsav=dacos(-.5d0)/pi*180.D0
ranghoh= anghoh/180.D0*pi

```

```

write(6,5003)
5003 format('this program changes roh, hoh angles on step 0')
write(6,5002) roh,anghoh
5002 format(' roh = ',D24.16,'anghoh = ',D24.16)

c***** DO LOOP OVER ALL WATERS *****

do iw=1,k*m

r1a = 0.0D0
r1b = 0.0D0
r2a = 0.0D0
r2b = 0.0D0

c-----
c----- i is the atom index for the O atom in H2O -----
c----- i1 is first H atom, i2 is second H atom -----
c----- These are atom indices. -----
c-----
i = nac*m +(iw-1)*3+1
i1=i+1
i2=i+2
iwm=k*m
if(iw.eq.iwm) write(30,*)'iw, i, i1, i2: ',iw,i,i1,i2
if(iw.eq.iwm) write(30,*)'ra(i,j): ',(ra(i,j),j=1,3)
if(iw.eq.iwm) write(30,*)'ra(i1,j): ',(ra(i1,j),j=1,3)
if(iw.eq.iwm) write(30,*)'ra(i2,j): ',(ra(i2,j),j=1,3)

if(iw.eq.iwm) write(30,*)'rb(i,j): ',(rb(i,j),j=1,3)
if(iw.eq.iwm) write(30,*)'rb(i1,j): ',(rb(i1,j),j=1,3)
if(iw.eq.iwm) write(30,*)'rb(i2,j): ',(rb(i2,j),j=1,3)

c***** find old roh**2 *****

do j=1,3
c r1a = r1a + chaa(j,i)*chaa(j,i)
c r1b = r1b + chba(j,i)*chba(j,i)
c r2a = r2a + chab(j,i)*chab(j,i)
c r2b = r2b + chbb(j,i)*chbb(j,i)

ha1(iw,j) = ra(i1,j)-ra(i,j)
ha2(iw,j) = ra(i2,j)-ra(i,j)
hb1(iw,j) = rb(i1,j)-rb(i,j)

```

```

hb2(iw,j) = rb(i2,j)-rb(i,j)

r1a = r1a + ha1(iw,j)*ha1(iw,j)
r2a = r2a + ha2(iw,j)*ha2(iw,j)
r1b = r1b + hb1(iw,j)*hb1(iw,j)
r2b = r2b + hb2(iw,j)*hb2(iw,j)
end do

c-----
c***** change roh to desired value *****
c----- and calculate the dot products of the oh vectors ---
c-----
      dota = 0.0d0
      dotb = 0.0d0

do j=1,3

c   chaa(j,i)=chaa(j,i)*roh/dsqrt(r1a)
c   chba(j,i)=chba(j,i)*roh/dsqrt(r1b)
c   chab(j,i)=chab(j,i)*roh/dsqrt(r2a)
c   chbb(j,i)=chbb(j,i)*roh/dsqrt(r2b)

      ha1(iw,j) = ha1(iw,j)*roh/dsqrt(r1a)
      ha2(iw,j) = ha2(iw,j)*roh/dsqrt(r2a)

      hb1(iw,j) = hb1(iw,j)*roh/dsqrt(r1b)
      hb2(iw,j) = hb2(iw,j)*roh/dsqrt(r2b)

      dota = dota + ha1(iw,j)*ha2(iw,j)
      dotb = dotb + hb1(iw,j)*hb2(iw,j)

end do

c-----
c***** calculate old HOH angles for each molecule *****
c-----
c   anga =dacos( (chaa(1,i)*chba(1,i)+chaa(2,i)*chba(2,i)
c   1
c   angb =dacos( (chab(1,i)*chbb(1,i)+chab(2,i)*chbb(2,i)
c   1      + chab(3,i)*chbb(3,i) ) / roh**2 )

      anga = dacos( dota/roh**2 )
      angb = dacos( dotb/roh**2 )
      angad = anga/(2.0D0*pi)*360.0D0
      angbd = angb/(2.0D0*pi)*360.0D0

```

```

r1a=dsqrt(r1a)
r2a=dsqrt(r2a)

r2b=dsqrt(r2b)
r1b=dsqrt(r1b)

write(30,5000) iw,r1a,r2a,angad
write(30,5001) iw,r1b,r2b,angbd

a1 = dcos(ranghoh/2.0d0) / dcos(anga/2.0d0)
b1 = dsin(ranghoh/2.0d0) / dsin(anga/2.0d0)
a2 = dcos(ranghoh/2.0d0) / dcos(angb/2.0d0)
b2 = dsin(ranghoh/2.0d0) / dsin(angb/2.0d0)

c***** reconstruct correct angles *****

dota=0.0d0
dotb=0.0d0

do j=1,3
r1 = ha1(iw,j)
r2 = ha2(iw,j)
ha1(iw,j) = (a1+b1)/2*r1 + (a1-b1)/2*r2
ha2(iw,j) = (a1-b1)/2*r1 + (a1+b1)/2*r2
r1 = hb1(iw,j)
r2 = hb2(iw,j)
hb1(iw,j) = (a2+b2)/2*r1 + (a2-b2)/2*r2
hb2(iw,j) = (a2-b2)/2*r1 + (a2+b2)/2*r2

dota = dota + ha1(iw,j)*ha2(iw,j)
dotb = dotb + hb1(iw,j)*hb2(iw,j)

end do

c***** check to see if angles are correct *****

anga = dacos( dota/roh**2 )
angb = dacos( dotb/roh**2 )
angad = anga/(2.0D0*pi)*360.0D0
angbd = angb/(2.0D0*pi)*360.0D0

c***** find new roh**2 *****

```

```

r1a = 0.0d0
r1b = 0.0d0
r2a = 0.0d0
r2b = 0.0d0

do j=1,3
  r1a = r1a + ha1(iw,j)*ha1(iw,j)
  r2a = r2a + ha2(iw,j)*ha2(iw,j)
  r1b = r1b + hb1(iw,j)*hb1(iw,j)
  r2b = r2b + hb2(iw,j)*hb2(iw,j)

c  r1a = r1a + chaa(j,i)*chaa(j,i)
c  r1b = r1b + chba(j,i)*chba(j,i)
c  r2a = r2a + chab(j,i)*chab(j,i)
c  r2b = r2b + chbb(j,i)*chbb(j,i)
end do

r1a=dsqrt(r1a)
r1b=dsqrt(r1b)
r2a=dsqrt(r2a)
r2b=dsqrt(r2b)

write(30,5000) iw,r1a,r1b,angad
write(30,5001) iw,r2a,r2b,angbd

5000 format('A: iw=',I3,'roh=',2D24.16,'anga=',d24.16)
5001 format('B: iw=',I3,'roh=',2D24.16,'angb=',d24.16)
c*****
c***** reconstruct the lab coordinates *****
c***** drr is used to scale the O position vectors up or down **
c***** this expands the cluster if some molecules overlap *****
c***** B. Hale 9-26-95 *****
c*****

do j=1,3
  ra(i1,j) = ha1(iw,j)+ra(i,j)
  ra(i2,j) = ha2(iw,j)+ra(i,j)
  rb(i1,j) = hb1(iw,j)+rb(i,j)
  rb(i2,j) = hb2(iw,j)+rb(i,j)
end do
c*****
if(iw.eq.iwm) write(30,*)'iw, i, i1, i2: ',iw,i,i1,i2
if(iw.eq.iwm) write(30,*)'ra(i,j): ',(ra(i,j),j=1,3)
if(iw.eq.iwm) write(30,*)'ra(i1,j): ',(ra(i1,j),j=1,3)

```



```

if(iw.eq.iwm) write(30,*)'ra(i2,j): ',(ra(i2,j),j=1,3)

if(iw.eq.iwm) write(30,*)'rb(i,j): ',(rb(i,j),j=1,3)
if(iw.eq.iwm) write(30,*)'rb(i1,j): ',(rb(i1,j),j=1,3)
if(iw.eq.iwm) write(30,*)'rb(i2,j): ',(rb(i2,j),j=1,3)

end do

return
end
C*****
SUBROUTINE pointm(rar,rbr,rma,rmb)
implicit real*8 (a-h,o-z)
real*8 lamb,lmb,kb
parameter(MAT=600)
parameter(MAM=600)

COMMON/R/ ra(MAT,3),rb(MAT,3),ia(MAT),ib(MAT)
1      ,imola(MAT),imolb(MAT),rcma(3),rcmb(3)
COMMON/seeds/iseeda(7),iseedb(7)
c COMMON/CONST/ kb,coul,pi,mi,ic,kall,alpha,istop,qf,qw,alp
COMMON/CONST/ kb,coul,pi,alpha,qf,qw,alp,ic,kall,istop,mi

c COMMON/km/k,m,na,nac,nwa,lamb,lmb,disp,beta,dd,Rsqa,jup,disa(MAT)
c 1 ,jlo,jhi,jxtot,LJ
COMMON/km/lamb,lmb,disp,beta,dd,Rsqa,disa(MAT),k,m,na,nac,nwa,jup,
1 jlo,jhi,jxtot,LJ

double precision rar(MAT,3),rbr(MAT,3)
double precision rma(MAT,3),rmb(MAT,3)
double precision dxa(3),dxb(3)
c.....
rpoint = 0.15d0

do iw=1,k*m

io= m*nac + iw*3-2
ih1=io+1
ih2=io+2
dra2 = 0.0d0
drb2 = 0.0d0

do ii=1,3
dxa(ii) =-2.d0*rar(io,ii)+rar(ih1,ii)+rar(ih2,ii)
dxb(ii) =-2.d0*rbr(io,ii)+rbr(ih1,ii)+rbr(ih2,ii)

```

```

dra2= dra2 + dxa(ii)**2
drb2= drb2 + dxb(ii)**2
end do

do ii=1,3
rma(io,ii) = rar(io,ii) + dxa(ii)*rpoint/dsqrt(dra2)
rmb(io,ii) = rbr(io,ii) + dxb(ii)*rpoint/dsqrt(drb2)

rma(ih1,ii) = rar(ih1,ii)
rma(ih2,ii) = rar(ih2,ii)
rmb(ih1,ii) = rbr(ih1,ii)
rmb(ih2,ii) = rbr(ih2,ii)

end do

end do
return
end
C*****
SUBROUTINE test4P(v,iflag)
implicit real*8 (a-h,o-z)
real*8 lamb,lmb,kb
parameter(MAT=600)
parameter(MAM=600)

COMMON/R/ ra(MAT,3),rb(MAT,3),ia(MAT),ib(MAT)
1 ,imola(MAT),imolb(MAT),rcma(3),rcmb(3)
COMMON/seeds/iseeda(7),iseedb(7)
c COMMON/CONST/ kb,coul,pi,mi,ic,kall,alpha,istop,qf,qw,alp
COMMON/CONST/ kb,coul,pi,alpha,qf,qw,alp,ic,kall,istop,mi

c COMMON/km/k,m,na,nac,nwa,lamb,lmb,disp,beta,dd,Rsqa,jup,disa(MAT)
c 1 ,jlo,jhi,jxtot,LJ
COMMON/km/lamb,lmb,disp,beta,dd,Rsqa,disa(MAT),k,m,na,nac,nwa,jup,
1 jlo,jhi,jxtot,LJ

double precision rar(MAT,3),rbr(MAT,3)
double precision rma(MAT,3),rmb(MAT,3)
double precision rra(MAT,MAT),rrb(MAT,MAT)
c.....
c..... The TIP4P interaction potential .....
c..... note rx6 = rx**6 .....
c..... HOH angle = 104.52 .....
c..... Roh = 0.9572 A .....

```

c.....

$$\text{voo4P}(\text{rx6}) = 600.0\text{d}3/\text{rx6}^{**2} - 610.0\text{d}0/\text{rx6}$$

$$\text{vmm4P}(\text{rx}) = (0.52\text{d}0/0.32983\text{d}0)^{**2}*(144.538\text{d}0/\text{rx})$$

$$\text{vmh4P}(\text{rx}) = (0.52\text{d}0/0.32983\text{d}0)^{**2}*(-72.269\text{d}0/\text{rx})$$

$$\text{vhh4P}(\text{rx}) = (0.52\text{d}0/0.32983\text{d}0)^{**2}*(36.1345\text{d}0/\text{rx})$$

c.....

c..... constants for the linear dimer with maximal H bond

c..... from Jorgensen et al. JCP 79 926 (1983) (TIP4P

c.....

$$\text{Q4P} = 0.52\text{d}0$$

$$\text{QRS} = 0.32983\text{d}0$$

$$\text{scale} = (\text{Q4P}/\text{QRS})^{**2}$$

$$\text{roh} = 0.9572\text{d}0$$

$$\text{roo} = 2.75\text{d}0$$

$$\text{theta} = 104.52\text{d}0$$

$$\text{rad} = 360.\text{d}0/2.0\text{d}0/\text{pi}$$

$$\text{th2} = \text{theta}/\text{rad}/2.0\text{d}0$$

$$\text{ang} = (\text{theta} - 90.0\text{d}0)/\text{rad}$$

$$\text{phi} = 46.0\text{d}0/\text{rad}$$

c.....

do i=1,MAT

do j=1,3

rar(i,j)=dfloat(i+j)

rbr(i,j)=dfloat(i+j)**2

rma(i,j)=0.0d0

rmb(i,j)=0.0d0

end do

end do

if(iflag.eq.0)then

rar(1,1)=0.0d0

rar(1,2)=0.0d0

rar(1,3)=0.0d0

```

rar(2,1)=0.0d0
rar(2,2)=-roh*dsin(ang)
rar(2,3)= roh*dcos(ang)

rar(3,1)=0.0d0
rar(3,2)=roh
rar(3,3)=0.0d0

rar(4,1)=0.0d0
rar(4,2)=roo
rar(4,3)=0.0d0

rar(5,1)= roh*dsin(th2)
rar(5,2)= roo + roh*dcos(th2)* dcos(phi)
rar(5,3)=  -roh*dcos(th2)* dsin(phi)

rar(6,1)= -roh*dsin(th2)
rar(6,2)= roo + roh*dcos(th2)* dcos(phi)
rar(6,3)=  -roh*dcos(th2)* dsin(phi)

else
endif

write(30,*)'*****'
write(30,*)'test of TIP4P potential'
write(30,*)'the atomic positions: '

write(30,*)'O atom: ',(rar(1,j),j=1,3)
write(30,*)'Ha atom: ',(rar(2,j),j=1,3)
write(30,*)'Hb atom: ',(rar(3,j),j=1,3)
write(30,*)'O atom: ',(rar(4,j),j=1,3)
write(30,*)'Ha atom: ',(rar(5,j),j=1,3)
write(30,*)'Ha atom: ',(rar(6,j),j=1,3)
write(30,*)'*****'
call pointm(rar,rbr,rma,rmb)

io=1
ih1=2
ih2=3

jo=4
jh1=5
jh2=6

dx =rar(io,1)-rar(jo,1)

```

```

dy =rar(io,2)-rar(jo,2)
dz =rar(io,3)-rar(jo,3)
rooa =(dx**2+dy**2+dz**2)**3

do ixx =io,io+2
do jxx =jo,jo+2
dx=rma(ixx,1)-rma(jxx,1)
dy=rma(ixx,2)-rma(jxx,2)
dz=rma(ixx,3)-rma(jxx,3)
rra(ixx,jxx)=dsqrt(dx**2+dy**2+dz**2)

end do
end do

```

```

v= voo4P( rooa ) +
1  vhh4P( rra( ih1,jh1 ) ) +
2  vhh4P( rra( ih1,jh2 ) ) +
3  vhh4P( rra( ih2,jh1 ) ) +
4  vhh4P( rra( ih2,jh2 ) ) +
5  vmh4P( rra( io,jh1 ) ) +
6  vmh4P( rra( io,jh2 ) ) +
7  vmh4P( rra( ih1, jo ) ) +
8  vmh4P( rra( ih2, jo ) ) +
9  vmm4P( rra( io, jo ) )

```

```

return
end

```

C*****

```

SUBROUTINE RN(iSEED,MI,IC,RNNUM)
implicit real*8 (a-h,o-z)
integer iseed
c-----
c---- NOTE: seeds should be integers -----
c---- to avoid roundoff errors -----
c---- 0 < iseed <(or=) 2147483648 -----
c---- since they are integers -----
c---- Best seeds: 8-10 integers -----
c-----

```

```
c----- old version -----
c  DOUBLE PRECISION A,B,SEED,RNNUM
C  WRITE(6,1)
C 1  FORMAT(' RN')
c  A=7.D00**5*SEED
c  B=2.0D00**31-1.D0
c  SEED=DMOD(A,B)
c  RNNUM=SEED/2.0D00**3
c-----
c-----
  A=7.D00**5*iSEED
  B=2.0D00**31-1.D0
  iSEED=idnint(DMOD(A,B))
  RNNUM=iSEED/2.0D00**31
c-----
  RETURN
  END
```

BIBLIOGRAPHY

1. B. N. Hale, Phys. Rev. A. **33**, 4156 (1986).
2. K. Binder and D. Stauffer, Adv. Physics. **25**, 343 (1976).
3. K. Binder, J. Phys. C **4**, 51 (1980).
4. C. H. Bennett, J. Computat. Phys. **22**, 245 (1976).
5. M. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, J. Chem. Phys. **21**, 1087 (1953).
6. R. Becker and W. Döring, Ann. Phys. (Leipzig) **24**, 719 (1935).
7. H. E. Stanley, Introduction to Phase Transitions and Critical Phenomena, Oxford University Press, New York, 1971, p. 9.
8. F. Abraham, Homogeneous Nucleation Theory, Academic Press, New York, 1974.
9. D. Oxtoby, J. Phys: Condens. Matter, **4**, 7627 (1992).
10. William D. Collins, Maurice L. Blackmon, Gordon B. Bonan, James J. Hack, Thomas B. Henderson, Jeffrey T. Kiehl, William G. Large, and Daniel S. McKenna, Cecilia M. Bitz and Christopher S. Bretherton, James A. Carton, Ping Chang Scott C. Doney, Benjamin D. Santer, Richard D. Smith, J. of Climate **19**, 2122 (2006).
11. K. Yasuoka and M. Matsumoto, J. Chem. Phys. **109**, 8451 (1998).
12. K. Yasuoka, M. Matsumoto, J. Chem. Phys. **109**, 8463 (1998).
13. Y. B. Zeldovich, Acta Physicochim. **18**, 1 (1943).
14. M. Volmer, Kinetik der Phasenbildung, Steinkopff, Dresden(1939).
15. L. Farkas, Z. Phys. Chem. (Leipzig) **A125**, 236 (1927).
16. A. B. Nadykto, A. Al Natsheh, F. Yu, K.V. Mikkelsen, and J. Herb, Computational quantum chemistry: A new approach to atmospheric nucleation, Advances in Quantum Chemistry **55**, 449-478 (2008).
17. C. R. T. Wilson, Philos. Trans. R. Soc. London, Ser. A **189**, 871 (1897); **A193**, 289.
18. M. Volmer, and H. Flood, Z. Phys. Chem. **A119**, 277 (1934).

19. R. H. Heist, H. Reiss, *J. Chem. Phys.* **54**, 665 (1973).
20. J. L. Katz and B. J. Ostermier, *J. Chem. Phys.* **47**, 478 (1967).
21. B. J. Mason, *The Physics of Clouds*, Oxford University Press, London (1957).
22. L. B. Allen and J. L. Kassner, *J. Coll. and Interface Sci.* **30**, 81 (1969).
23. R. C. Miller, R. J. Anderson, J. L. Kassner, and D. E. Hagen, *J. Chem. Phys.* **78**, 3204 (1983).
24. J. L. Schmitt, R. A. Zalabsky, and G. W. Adams, *J. Chem. Phys.* **79**, 4496 (1983).
25. G. W. Adams, J. L. Schmitt, and R. A. Zalabsky, *J. Chem. Phys.* **81**, 5074 (1984).
26. P. E. Wagner and R. Strey, *J. Chem. Phys.* **80**, 5266 (1984).
27. R. Strey, P. E. Wagner and T. Schmeling, *J. Chem. Phys.* **84**, 2325 (1986).
28. Y. Viisanen, R. Strey and H. Reiss, *J. Chem. Phys.* **99**, 4680 (1993); **112**, 8205 (2000).
29. Y. Viisanen and R. Strey, *J. Chem. Phys.* **101**, 7835 (1994).
30. J. Wölk and R. Strey, *J. Phys. Chem.* **105**, 11683 (2001).
31. A. Khan, C. H. Heath, U. Diergswiler, B. E. Wyslouzil and R. Strey, *J. Chem. Phys.* **119**, 3138 (2003).
32. M. Garibeh, Y. Kim, U. Diergswiler, B. E. Wyslouzil, D. Ghosh and R. Strey, *J. Chem. Phys.* **122**, 094512 (2005).
33. J. Wölk, R. Strey, C. H. Heath and B. E. Wyslouzil, *J. Chem. Phys.* **117**, 4954 (2003).
34. A. Fladerer and R. Strey, *J. Chem. Phys.* **124**, 164710 (2006).
35. K. Lland, J. Wölk and R. Strey, *J. Chem. Phys.* **127**, 154506 (2007).
36. S. Sinha, A. Bhabhe, H. Laksmono, J. Wölk, R. Strey and B. Wyslouzil, *J. Chem. Phys.* **132**, 1 (2010).
37. V. I. Kalikmanov, *J. Chem. Phys.* **124**, 124505 (2006).
38. D. Kashchiev, *J. Chem. Phys.* **118**, 1837 (2003).
39. M. Volmer and A. Weber, *Z. Phys. Chem.* **119**, 227 (1926).

40. S. L. Girshick and C. Chiu, *J. Chem. Phys.* **93**, 1273 (1990).
41. C Hung, M. J. Krasnopoler, and J. L. Katz, *J. L. J. Chem. Phys.* **90**, 1856 (1989).
42. B. Hale, "Scaled Models for Nucleation," in *Lecture Notes in Physics* **309**, 323 (1988).
43. J. L. Katz, ; J. A. Fisk, ; M. M. Rudek, *Nucleation and Atmospheric Aerosols*; M. Kulmala, P.E. Wagner, Eds.; Pergamon Press: New York, 1996, p 1.
44. B. Hale,; K. Han, In *Nucleation and Atmospheric Aerosols*; N. Fukuta, P. Wagner, E., Eds.; A. Deepak: Hampton, VA, 1992; p 133.
45. A. Dillmann, Ph.D. thesis, Göttingen, (1989); A. Dillmann and G. E. Meier, *J. Chem. Phys.* **94**, 3872 (1990).
46. M. E. Fisher, *Physics* **3**, 255 (1967).
47. J. S. Langer and L. A. Turski, *Phys. Rev. A* **8**, 3230 (1973); **22**, 2189 (1980).
48. J. S. Langer and A. J. Schwartz, *Phys. Rev. A* **21**, 948 (1980).
49. J. D. Gunton, "The Dynamics of First Order Phase Transitions," in *Phase Transitions and Critical Phenomena*, Ed. by C. Domb and J. L. Lebowitz, Vol. **8**, Academic Press, New York, (1983).
50. B.J.C.Wu, P. P. Wegener, and G.D. Stein, *J. Chem. Phys.* **68**, 308 (1978).
51. B. N. Hale, *Metallurg. Trans.* **23A**, 1863 (1992).
52. B. N. Hale and D. J. DiMaggio, *J. Phys. Chem.* **B 108**, 19780 (2004).
53. B. N. Hale, *Nucleation and Atmospheric Aerosols* (M. Kasahara and M. Kulmala, Kyoto University Press, Kyoto Japan)(2004) 3-14.
54. M. Gharibeh, Y. Kim, U. Dierregswiler and B. Wyslouzil *J. Chem. Phys.* **122**, 094512 (2005); D. Ghosh, A. Manka, R. Strey, S. Seifert, R. Winans and B. Wyslouzil, *J. Chem. Phys.* **129**, 124302 (2008).
55. D. Ghosh, D. Bergmann, R. Schwering, J. Wölk, R. Strey, S. Tanimura and B. Wyslouzil, *J. Chem. Phys.* **132**, 024307 (2010).
56. M. P. Allen and D. J. Tildesley, *Computer Simulations of Liquids* (Oxford University Press, Oxford, 1987).
57. W. L. Jorgensen, J. Chandrasekhar and J. D. Madura, *J. Chem. Phys.* **79**, 926, (1983).

58. A. Rahman and F. H. Stillinger, *J. Chem. Phys.* **55**, 3336-59 (1971).
59. F. H. Stillinger, A. Rahman. *J. Chem. Phys.* **60**, 1545-57 (1974).
60. H. L. Lemberg, F. H. Stillenger., *J. Chem. Phys.* **62**, 1677-90 (1975).
61. A. Rahman, F. H. Stillenger, H. L. Lemberg. *J. Chem. Phys.* **63**, 5223-30 (1975).
62. F. H. Stillenger, A. Rahman. *J.Chem. Phys.* **68**, 666-70 (1978).
63. B. N. Hale, *Aust. J. Phys.* **49**, 425 (1996).
64. R. C. Ward, B. N. Hale, S. J. Terrazas, *Chem. Phys.* **78**, 420-3 (1983).
65. P. Deutsch ,B. Hale, R. Ward, D. Reago Jr, *J. Chem. Phys.* **78**, 5103-7 (1983).
66. J.K. Lee, J.A. Barker, F.F. Abraham, *J. Chem. Phys.* **58**, 3166-80 (1973).
67. N. Garcia, J.M. Torroja, *Phys. Rev. Letters* **47**, 186-190, (1980).
68. K. J. Oh, X. C. Zeng, and H. Reiss, *J. Chem. Phys.* **107**, 1242, (1997).
69. B. Chen B. and J. I. Siepmann, *J. Phys. Chem. B* **104**, 8725, (2000).
70. B. Chen and J. I. Siepmann, *J. Phys. Chem. B* **105**, 11275, (2001).
71. I. Kusaka, G.Wang, and J. H. Seinfeld, *J. Chem. Phys.* **108**, 3416, (1998).
72. A. Lauri, J. Merikanto, E. Zapadinsky, and H. Vehkamaki, *Atm. Res.* **82**, 489 (2006).
73. P.R. ten Wolde, D. Frenke, *J. Chem. Phys.* **109**, 9901-18,(1998).
74. R. Gilgen, R. Kleinrahm and W. Wagner, *J. Chem. Therm.* **22**, 339 (1994).
75. H. M. Ellerby, C. L. Weakliem, and H. Reiss, *J. Chem. Phys.* **95**, 9209 (1991).
76. C.Weakliem and H. Reiss, *J. Chem. Phys.* **99**, 5374, (1993).
77. B. Hale, R. Ward, *J. Stat. Phys.* **28**, 487-95, (1982).
78. B. N. Hale and D. M. DiMattio, "A Monte Carlo Discrete Sum (MCDS) Nucleation Rate Model for Water". in *Nucleation and Atmospheric Aerosols 2000*, Ed. by B. N. Hale and M. Kulmala, AIP Press, Melville NY (2000) p 31.
79. J. Merikanto , E. Zapadinsky , and H.VehkamSki , *J. Chem. Phys.* **121**, 914, (2004).
80. J. Merikanto, E. Zapadinsky, A. Lauri, and H. Vekhamaki, *Phys. Rev. Letters* **98**, 145702 (2007).

81. F. H. Stillinger, J. Chem. Phys. **38**, 1486, (1963).
82. D. O. Dunikov, S. P. Mallyshenko, and V. V. Zhakhovskii, J. Chem. Phys. **115**, 6623 (2001).
83. J. Perez-Pellitero, P. Ungerer, G. Orkoulas and A. D. Mackie, J. Chem. Phys. **125**, 054515 (2006).
84. M. R. Shirts and V. S. Pande, J. Chem. Phys. **122**, 144107 (2005).
85. B. Chen, J. I. Siepmann, K. I. Oh, and M. I. Klein, J. Chem. Phys. **115**, 10903 (2001).
86. R. Gilgen, R. Kleinrahm and W. Wagner, J. Chem. Therm. **22**, 339 (1994).

VITA

Mark Allan Thomason was born in Fordyce, Arkansas. He received his secondary education from Fordyce High School in Fordyce, AR in 1991. He graduated from University of Arkansas at Monticello with a B.S. in Mathematics in 2002. He earned an M.S. in Physics at the University of Missouri Rolla in 2006.