The Role of Three-Body Interactions in the Adsorption of Argon in Silicalite-1

Volume I

Third Year Physical Chemistry Research Report

by
Félix Fernández-Alonso

under the supervision of Dr. David Nicholson and Mr. Roland J-M Pellenq

Imperial College of Science Technology and Medicine
University of London
June 1993

Abstract

We report a study of the role of three-body, non-additive interactions in the adsorption of argon in silicalite-1. This work has been fostered by striking disagreement between simulation and experiment in the shapes of the isotherm and isosteric heat curve at 77.4 K and also by the fact that adsorbate-adsorbate three-body terms are not included in the most recent potential functions for the adsorption of noble gases in silicalite-1.

We have adopted an intermolecular potential function comprised of the empirical two-body Barker-Fisher-Watts potential and three-body terms up to the fourth-order, triple-dipole term. In assessing the importance of three-body interactions we take the effective Lennard-Jones potential as a reference since it is known that this function reproduces the properties of solid and liquid argon quite accurately. We have found that the effective Lennard-Jones potential is inadequate in describing the adsorbate-adsorbate potential. Of the three types of three-body interactions, the argon-argon-oxygen is dominant being around eighty percent of the total three-body energy. Furthermore, unlike the solid or liquid argon phase, three-body terms beyond the Axilrod-Teller-Muto do not show mutual cancellation and at least the dipole-dipole-dipole and dipole-dipole-quadrupole need to be considered. Moreover, three-body effects do not seem to be local, the same conclusions being applicable to different regions within the unit cell of silicalite-1

Finally, the effect of three-body forces was incorporated into a full-scale molecular simulation in the form of a coverage-dependent effective Lennard-Jones potential. Preliminary results show that the theoretical isosteric heat curve can be refined to match the experimental if proper account of three-body effects is taken.

Table of Contents

Chapter 1. Introduction.

1.1 Zeolites and simulation.

Zeolites (from the Greek 'zeo', to boil, and 'lithos', stone) are microporous solids with voids sufficiently large to allow the diffusion and adsorption of molecules. Approximately thirty-four zeolites can be found in nature, the first one discovered by Baron Axel Cronsted back in 1756 [1]. Significant progress in the field did not arrive until the 1940s when Barrer synthesized pure-phase zeolites for the first time [2]. Barrer's steps were followed in the next decades by Union Carbide and later by the Mobil Research Division when it was discovered that the cracking of crude oil could be enhanced by the use of zeolite X [3]. Today, about sixty-five zeolitic structures are known with a wide range of compositions and applications. The latter largely depend on the main properties of most zeolites, that is, their selectivity, high adsorption capacities and ion exchange capabilities. Some of the present applications include: ion exchange in detergents or in the handling of radioactive waste, catalysis in alkylation, isomerisation, hydrogenation and polymerisation reactions as it is the case of the aforementioned hydrocarbon cracking process and purification, drying and liquid or gas separation [4].

From the previous general overview it is apparent that these materials are of great economic and industrial importance. Similarly, they are also of great academic interest from both an experimental and theoretical viewpoint. The microporous framework exhibited by zeolites can be used to study the fundamentals of adsorption and diffusion as well as the nature of the intermolecular forces between the adsorbate and the zeolitic framework. Furthermore, some of the applications mentioned above ultimately depend on the adsorption mechanisms that occur inside the pores and therefore it seems compelling to study these systems in detail if we are to understand and probably one day tailor the properties of these materials.

A very exciting area which has acquired tremendous momentum in the last few years comprises the computer simulation of structure, adsorption and diffusion processes in zeolites. The enormous computational power available today has allowed the testing of theories and models of adsorption and intermolecular forces which had been laid down decades ago [5]. In a relatively short period of time significant advances have been achieved in this field of computer-aided simulation as, for example, in the modelling of zeolite structures [6-7], location of non-framework ions [8-9], simulation of the dynamics of adsorbed molecules [10-12], the calculation of diffusion coefficients [13-15] or the prediction of the electronic structure of adsorbed molecules [16-17].

1.2 Intermolecular forces.

The theory of intermolecular forces not only tells us about the specific, quantum-mechanical nature of the physical interactions between molecules but also provides us with explicit functions to evaluate the potential as a function of variables such as distance and angle. The subject reviewed briefly in this section is too vast to allow a complete account of the origin and derivation of the expressions to be presented here [18-20].

1.2.1 The rigorous treatment of intermolecular forces.

Intermolecular forces are entirely of coulombic nature, that is, they arise from the interactions of the charges contained within molecules. The theory presented here holds as long as the interaction is of a physical type as opposed to chemical, the latter characterised by electron transfer and bond formation.

It is customary to divide the intermolecular potential into a long-range and a short-range part, the word 'range' adopting in this context the meaning of separation of the molecules under consideration.

1.2.1.1 The long-range part of the potential.

At large separations, it is possible to expand the coulombic potential for the interaction between two molecules as a series of terms each of which describes the interactions between different permanent multipoles. This expression is known as the multipolar expansion (i.e. see ref. 18). For example, for two molecules (A and B) containing permanent dipoles (μ) the potential could be described by:

$$U_{\mu_A\mu_B} = \frac{-2\mu_A\mu_B}{(4\pi\epsilon_o)R^3}\,[\,f(\alpha,\beta,\xi)\,] \qquad (1)$$

where μ_a and μ_b are the permanent dipoles of A and B respectively, r is the distance from the centres of mass and f is a function of the angles that define the relative orientation of the dipoles with respect to one another. There are expressions for a large host of multipole-multipole interactions but it goes beyond the scope of this paper to reproduce all of them (i.e see ref. 19).

Molecules do not only exhibit interactions of the type explained above. In a system comprised of many molecules there is always an electric field surrounding each molecule which causes redistribution of charge, a phenomenon called polarization. The induction of a multipole moment is then possible and it turns out to be proportional to the ability of the electrons to reorganise within the molecule. One would then expect the interaction to be proportional to the polarizability of the molecule. For example, in the case of an induced-dipole interaction for an spherical distribution of charge (i.e. a noble gas), the average energy can be written as:

$$\langle U\rangle = \frac{-\alpha\mu^2}{(4\pi\epsilon_o)^2R^6} \qquad (2)$$

where α is the polarizability of the molecule.

So far we have been dealing with two phenomena, namely electrostatic and induced interactions, which can be described in classical terms quite satisfactorily. The third and last type that we will cover in this section is of a purely quantum-mechanical nature and comes from the interaction of instantaneous multipoles. They are referred to as 'dispersion' forces, a term coined by London in the 1930s [21] and they are often the most important part of the intermolecular potential. One can extract the most important features of the dispersion force by using the semiclassical model of Drude which considers the electrons as classical oscillators subjected to the Schrödinger equation (see Hirshfelder et al. in ref. 19). From the Drude model the leading term in the dispersion energy is proportional to r^{-6}, the same type of distance dependence found for induction. The complete pair dispersion energy has the form:

$$U_{disp} = -\frac{C_6}{R^6} - \frac{C_8}{R_8} - \frac{C_{10}}{R^{10}} - \ldots \qquad (3)$$

where the first term accounts for the interaction between instantaneous dipoles and, similarly, higher order terms correspond to the coupling of instantaneous dipole-quadrupole, quadrupole-quadrupole, etc. In the same way as there was a polarization dependence for the induction forces one might expect in this case a dependence on the product of polarizabilities of the two molecules. For example, the C_6 term can be written as:

$$C_6^{AB} = \left(\frac{3}{\pi}\right)\int_0^\infty \alpha_A(i\omega)\,\alpha_B(i\omega)\,d\omega \qquad (4)$$

where $\alpha(i\omega)$ is the frequency-dependent dipole polarizability.

Even though so far we have been using a classical or, at most, a semiclassical approach to find the main features of long-range forces, it is possible to put them altogether in the framework of quantum mechanics and more specifically of perturbation theory. If the intermolecular potential between two molecules is regarded as a very small perturbation of the system, it is possible to write the energy as a series of terms, the so-called Rayleigh-Schrödinger expansion. The first two terms account for the long-range electrostatic, and the induction and dispersion energies respectively. However, the expansion does not truncate after the second term and continues with third, fourth ... terms. This is equivalent to saying that the intermolecular potential for an ensemble of molecules is not only composed of the pair interaction energies between the particles. The simplest case of this many-body problem is that of three molecules A, B and C. If we were to calculate the total interaction potential we would find that it does not correspond to the pair energies AB, AC and BC. In this case the potential is said to be non-additive because the presence of a third molecule perturbs the interaction of the other two. Electrostatic interactions are strictly pair-additive while induction and dispersion are non-additive and many-body interactions are a part of the total interaction potential.

As we have said, non-additivity occurs for induction but even more for dispersion forces. The leading correction of the three-body dispersion energy for spherical particles, a dipole-dipole-dipole term, was first evaluated by Axilrod and Teller [22] and by Muto [23] independently from the former two. It reads as shown below.

$$U_{ddd} = \upsilon_{ABC} R_{12}^{-3} R_{23}^{-3} R_{31}^{-3} \ (1 + 3 \cos \phi_1 \cos \phi_2 \cos \phi_3) \qquad (5)$$

where ν_{ABC} is a constant whose value depends on the nature of the molecules interacting. The three angles correspond to the internal angles of the triangle defined by the three particles. It is worthwhile noting the limiting behaviour of this expression. In the case of the equilateral and right triangles, the interaction is positive (repulsive) but not for the arrangement of the three atoms on a line for which the interaction is negative (attractive).

1.2.1.2 The short-range part.

The theory presented above fails in the case when two molecules are so close to each other that their electron clouds overlap. As a consequence of the Pauli Exclusion principle the overlap area is depleted of electron density and the nuclei experience mutual repulsion. The rigorous treatment of these forces poses a great theoretical challenge even today. Amongst other things it needs to account for effects such as electron exchange, that is, the mixing of wavefunctions as the molecules come very close. From self-consistent field (SCF) quantum-mechanical calculations it is possible to obtain both the long-range electrostatic and repulsive parts of the intermolecular potential. For repulsion, these computations suggest a decaying exponential form known as the Born-Mayer repulsion function [24] which reads as follows:

$$U_{rep} = A \exp(-br) \qquad (6)$$

The parameters A and b can be obtained from quantum-mechanical calculation yet it is common practice to regard them as adjustable parameters to fit experiment (i.e. molecular-beam scattering data).

1.2.2 The importance and use of effective potentials.

The previous section showed the non-trivial character of a rigorous treatment of the forces between molecules. Using this 'brute-force' approach, the problem becomes unmanageable beyond a few, simple molecules. What can we do then when we are interested in modelling a zeolite which is comprised of hundreds and maybe thousands of molecules? The usual procedure in this extremely complicated problem is to combine the effects of all the interactions into one simple function called an effective potential. This function will hopefully contain all the necessary ingredients to account for the intermolecular energy (induction, dispersion, etc). A very common effective potential is the 12-6 Lennard-Jones function with two adjustable parameters: the well depth (ε) and the intermolecular separation at zero energy (σ).

$$U_{effective} = 4\epsilon \left(\left(\frac{\sigma}{R}\right)^{12} - \left(\frac{\sigma}{R}\right)^{6} \right) \tag{7}$$

An illustrative example in close connection with the aims of this paper is the effective potential of argon (ε = 120 K; σ = 3,405 Å). This potential is known to reproduce quite well the properties of liquid argon [25]. It is devised to account for two-body and three-body forces, the latter being almost entirely of a dipole-dipole-dipole character.

The main problem with effective potentials is one of transferability. The sole fact that it works for a uniform system does not guarantee at all a satisfactory performance in more complicated cases (i.e. non-uniform adsorption systems) and requires testing its validity and in many cases readjusting the parameters to accommodate to the characteristics of the new system under study.

1.3 The study of sorption in zeolites.

Adsorption in zeolites can be physical or chemical. We will focus on the former, characterised by low heats of sorption, of the same order of magnitude as the latent heat of liquefaction, and the lack of electronic transfer between the adsorbate and the adsorbent [26].

It is quite convenient in the study of adsorption to split the intermolecular potential into two parts: the adsorbate-adsorbent and the adsorbate-adsorbate. Each of these parts is subsequently subdivided into electrostatic, induction, dispersion and repulsion according to the properties of the atomic components of both zeolite and adsorbate.

Obviously the choice of the potential function is both a delicate and essential task. As the next chapter will show in more detail, it is often possible to obtain an accurate adsorbate-adsorbent potential due to the static character of the zeolite. The adsorbate-adsorbate interactions pose a severe technical problem and in most cases workers describe it in terms of an effective Lennard-Jones potential of the type shown before.

With a complete intermolecular potential it is possible to simulate adsorption and calculate thermodynamic quantities such as for example the heat of adsorption. At zero coverage (i.e. one adsorbate molecule per zeolite unit cell), the heat of adsorption can be evaluated by [27] :

$$q^{\circ} = \langle U \rangle - RT \ , \tag{8}$$

$$\langle U \rangle = \frac{1}{Z} \int U \exp\left(-\frac{U}{RT}\right) d\boldsymbol{r} \ , \tag{9}$$

$$Z = \int \exp\left(-\frac{U}{RT}\right) d\boldsymbol{r} \tag{10}$$

where Z is the configurational integral, <U> is the internal energy and the U in the integrand is the adsorbate-adsorbent energy of interaction. From these two quantities, it is possible to then calculate the isosteric heat q^{iso} at zero coverage. At higher coverage it is necessary to include the potential between all the adsorbate molecules and the zeolite wall as well as between the adsorbate particles and the integral shown above becomes more tedious to evaluate.

Simulations also allow the determination of Henry Law constants for the low coverage region and sorption isotherms, the latter specially useful when comparing simulation with experiment. There are several methodologies to carry out a simulation. For adsorption systems the Grand Canonical Ensemble (constant chemical potential, volume and temperature) is more common than others (i.e the Canonical Ensemble with a constant number of particles, volume and temperature) [28]. The Grand Canonical scheme is a very convenient one for the study of adsorption systems because as the number of adsorbate particles is allowed to fluctuate, isotherms and heats of adsorption are easily calculated.

Chapter 2. The MFI-type zeolite silicalite-1.

2.1 General features of silicalite-1.

Silicalite-1 [29] is the pure siliceous form of ZSM-5 (Zeolite Socony No. 5) [30]. Both of these zeolites and all the polymorphs formed by substitution of silicon by aluminium or other elements such as boron, iron, gallium, etc receive the same name: MFI (Mobil Five) zeolite. They all have the unit cell formula [31]:

$$\mathbf{mI[(SiO_2)_{96-n}(TO_2)_n]yH_2O} \qquad \textbf{(11)}$$

Where m is the number of compensatory cations I that balance the negative charges introduced by the TO_2 groups in the framework. In the case of silicalite-1 the unit cell structure is composed of pure silicon oxide units and has the formula shown by equation 12 [32]:

$$\mathbf{[(SiO_2)_{96}]yH_2O} \qquad \textbf{(12)}$$

As numerous experimental studies employing mainly ^{29}Si and ^{27}Al NMR techniques have shown [33], the structure of any MFI-type zeolite, including silicalite-1, can be described as an assembly of 5-1 complexes joined together to give the so-called pentasil unit. Several pentasil units then yield the pentasil chain as it is shown in figure 1 [34].

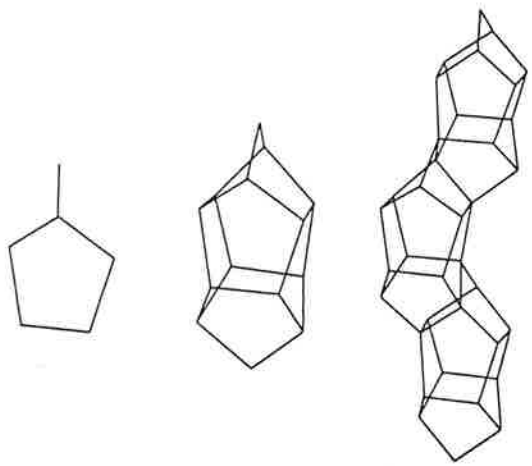


Fig. 1 Structural constituents of the MFI-type zeolites. From left to right: 5-1 complex, pentasil unit and pentasil chain.

The structure of the unit cell is shown in figure 2 [37]. From this picture, three different regions are distinguishable: region I, made up of sinusoidal channels running in a zig-zag fashion along the a-axis; region II along the b-axis, the so-called straight channel; and region

III, the intersection between sinusoidal and straight channels.

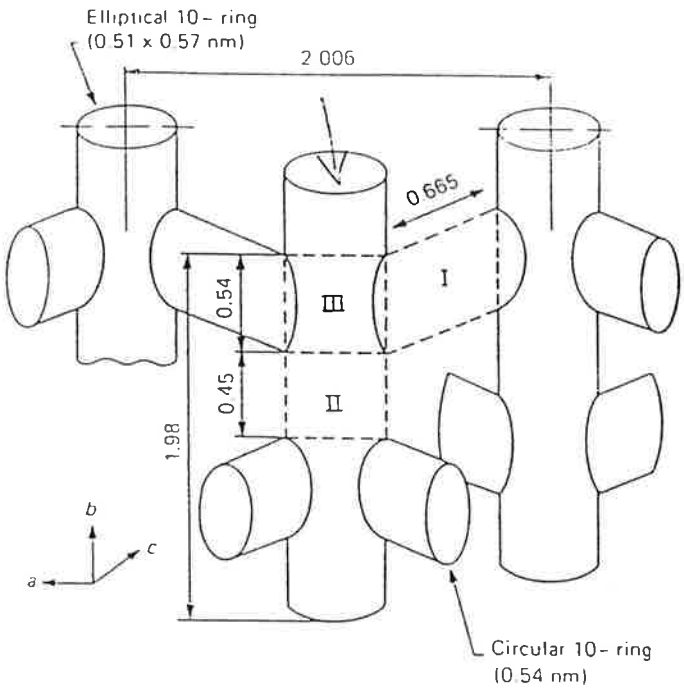


Fig. 2 Structure of the unit cell for silicalite-1. Note three distinct regions: sinusoidal channels (I), straight channels (II) and intersections (III) (reprint from ref. 37).

X-ray diffraction experiments [35] have shown two different crystal structures for silicalite-I: an orthorhombic form at high temperatures and a monoclinic at low temperatures. This temperature-dependent phase transition, which occurs at 340 K [39], can also be induced by the adsorption of certain organic molecules, i.e. p-xylene [40], and is generally reversible upon desorption.

The orthorhombic form is well described by the group Pnma which in terms of the crystallographic parameters is defined as: $a \neq b \neq c$ and $\alpha=\beta=\tau=\pi/2$, that is to say, a unit cell composed of different crystallographic parameters but same angles between the different axis. As shown in figure 2, there are two types of channels, both of them consisting of 10-oxygen rings. The straight channels in the orthorhombic phase have diameters of 5.2Åx5.75Å. The sinusoidal channels are of a more circular shape with diameters of 5.3Åx5.6Å [38].

The monoclinic form belongs to the $P2_1/N.1.1$ space group which is defined as: $a \neq b \neq c$ and $\alpha \neq \pi/2$, $\beta=\tau=\pi/2$. The straight channels in this structure are 5.2Åx5.8Å and the sinusoidal 5.0-5.3Åx5.8-5.9Å, the latter being more elliptical than in the orthorhombic form.

Both structures are quite similar, the only difference being a small change in the α

angle from 90° (orthorhombic) to 90.67° (monoclinic). The transition induced by the sorption of small amounts of organic molecules is generally called the 'para' form. This new phase involves distortion of the lattice structure and is characteristic of the adsorbate used [41]. Adsorbates known to induce such transitions have sizes comparable to that of the zeolite pore although size is not the only prerequisite; for example cyclohexane is similar in size to p-xylene yet it does not induce a transition. The induction of a structural transition seems also connected to the ability of the adsorbate to rearrange in the sinusoidal channels [42].

An important point to address in this section is in regard to the nature of the Si-O bond in silicalite-1. Adopting a purely ionic view of the bond one would assign a +4 to Si and a -2 to the O. However, both Pauling's electronegativity rules [43] and, more recently, the electronegativity equalisation model [44] show that the charge should be around +2 for Si and -1 for O, coinciding with those found for quartz and thus corroborating the idea that silicalite-1 could be regarded as a polymorph of quartz. These results also agree with ab-initio Hartree-Fock calculations of the partial charges in quartz, though no studies have been performed in the silicalite-I system yet [45].

2.2 The sorption of noble gas probes in silicalite-I.

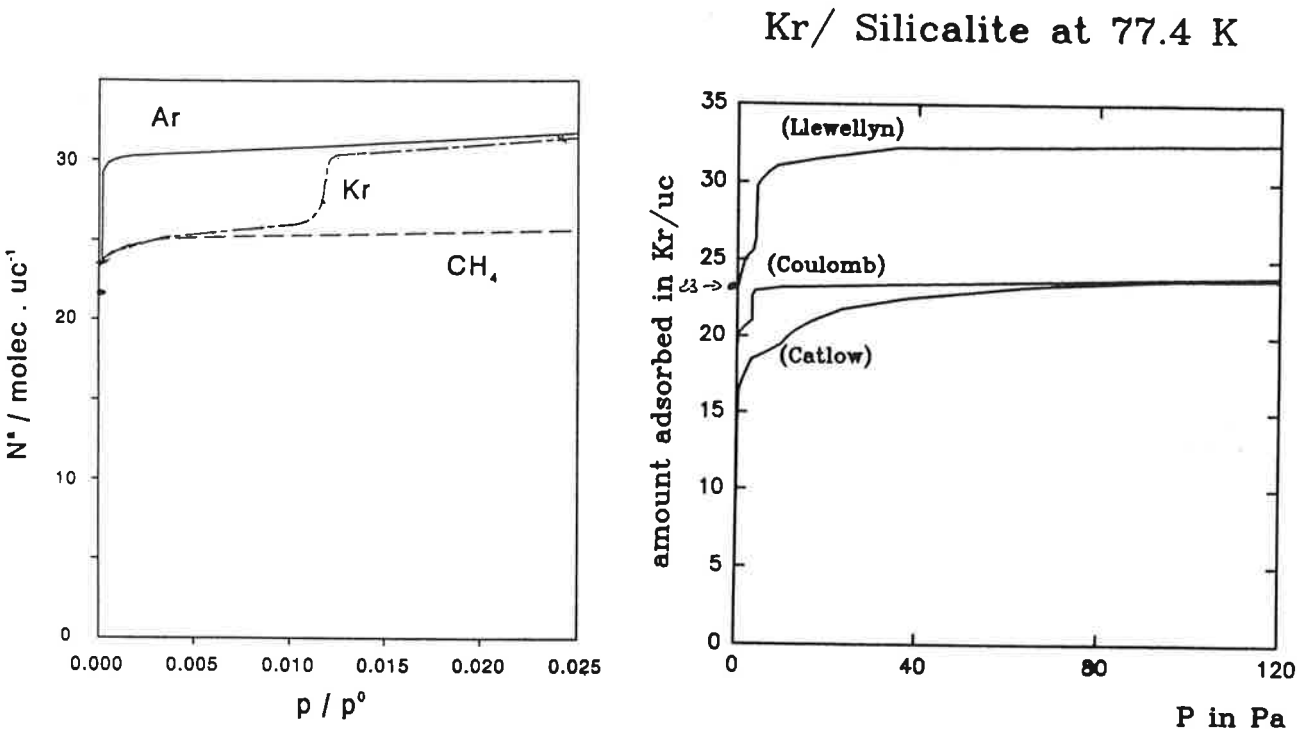
2.2.1 Experiment.

In this section we outline the isotherms and isosteric heat curves for the adsorption of probe molecules including argon and krypton that have been obtained more recently [37,46,57]. This section will focus on the experimental data available for noble gas probes with special emphasis on argon.

At 77.4 K the argon isotherm shows a very peculiar behaviour with a substep at a relative pressure of $2x10^{-4}$ from an uptake of twenty-three to thirty-one molecules per unit cell (see fig. 3).

The net enthalpy curve gives a fairly constant exothermic signal around -7.5 kJ/mol up to an uptake of about twenty molecules per unit cell. A distinct increase in the enthalpy of sorption occurs simultaneously with the substep and reaches a value of -9.8 kJ/mol. After this increase the heat decreases rapidly to the enthalpy of liquefaction in correspondence to the final plateau region of the isotherm (see fig. 5).

It is interesting to contrast the data for argon with that for krypton. At the same temperature, the latter also exhibits a substep although the relative pressure is much higher, at around $1.1x10^{-2}$. The initial plateau region in the isotherm occurs at about twenty-four molecules per unit cell. The maximum uptake was thought to be thirty-one molecules per unit cell, as in the argon case, but recent data (see fig. 4) show that in fact the final plateau occurs at a lower coverage of about twenty-four atoms per unit cell [46], in agreement with previous studies by Catlow et al. [57].



Figs. 3, 4 Adsorption isotherms for Ar, Kr and CH_4 at 77.4 K. Left: data due to Llevellyn (reprint from ref. 37). Right: more recent experimental data (Coulomb and Catlow) showing a maximum uptake for Kr of twenty-four atoms per unit cell (reprints from refs. 46, 57)

The net enthalpy of adsorption shows an initial horizontal line, at -6.9 kJ/mol, up to an uptake of twenty molecules and then decreases to a value of -4.5 kJ/mol.

In the case of argon the most recent experimental findings agree quite well with previous studies [47,48]. The substep was attributed to a densification of the adsorbate phase yet this conjecture has not been confirmed by a structural study of the adsorbate phase. The heat curve remains constant up to twenty-two molecules per unit cell and between twenty-four and thirty the heat is clearly more exothermic suggesting an adsorbate phase transition. This has been supported by neutron scattering data which shows an enhancement of peaks correlated to the crystalline organisation of the argon adsorbate phase in the zeolite cavities.

The behaviour of krypton is very similar to that of argon. However, in this case the heat becomes increasingly endothermic during the substep. This has been attributed to confinement effects within the microporous network [49]. Intense neutron diffraction peaks have been observed in the same region as argon and suggests a disordered to crystalline 'solid-like' transition.

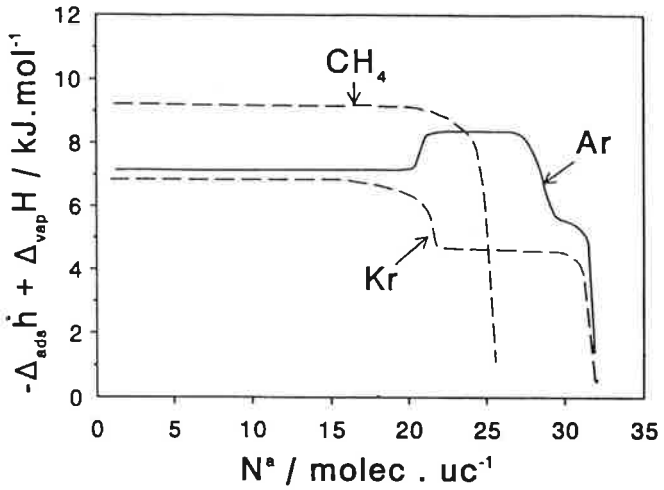


Fig. 5 Net enthalphies of adsorption for Ar, Kr and CH_4 at 77.4 K. The net enthalpy is defined as $-\Delta H_{ads}+\Delta H_{vap}$ where $-\Delta H_{ads}$ is the isosteric heat and ΔH_{vap} is the enthalpy of vaporization (reprint from ref. 37).

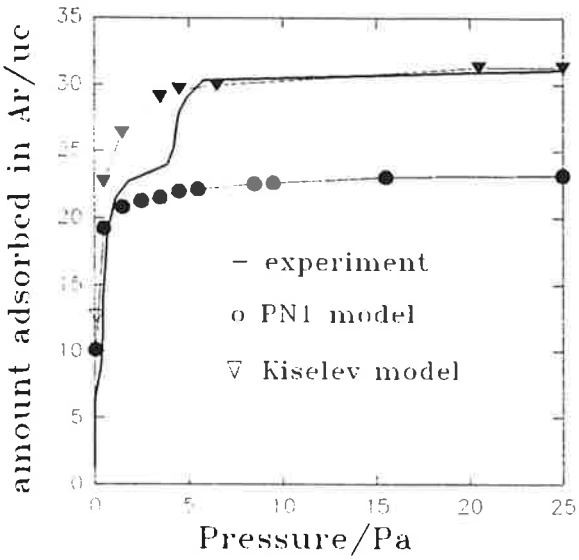
2.2.2 Theory and simulation.

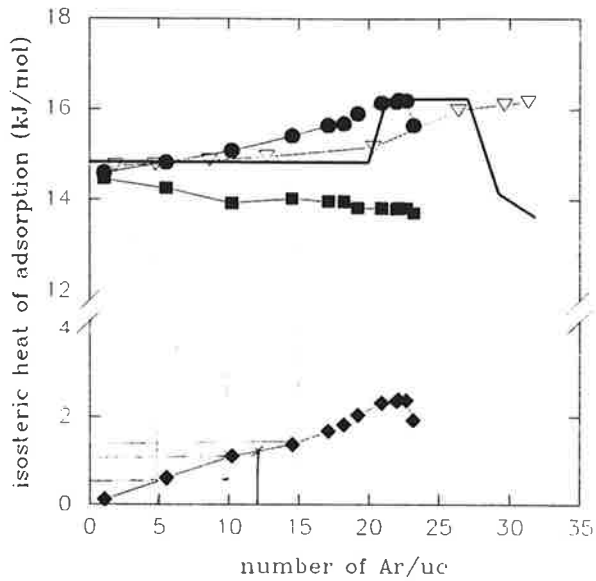
Kiselev et al. [50] were the first to perform a simulation of the adsorption of noble gases in silicalite-1, even before there was experimental data available. The choice in the potential function, a step which the first chapter has emphasized to be crucial, was rather arbitrary. A potential form of Lennard-Jones type was chosen and quite drastic approximations made; these included, for instance, ignoring the interaction energy with the silicon atoms in the framework. From the simulation it was found a correlation between the isosteric heats and the mean molecular polarizabilities of the adsorbate as well as a linear relationship between the entropy of adsorption and adsorbate size. The main problem at the time was the lack of experimental data to contrast against this results.

More recently, the Kiselev model has been challenged by a more elaborate one: the PN1 (Pellenq-Nicholson-1) potential [51]. A complete description of the potential has been given elsewhere [52]. The PN1 model emphasizes the long-range attractive part of the intermolecular energy between a neutral species and the charged framework using dispersion terms up to C_{10} and three-body interactions up to dipole-quadrupole and fourth-order terms of the type argon-oxygen-oxygen and argon-silicon-silicon. The polarizabilities, necessary to determine the dispersion coefficients, have been obtained from Auger spectroscopy [53]. The repulsive part of the dispersion potential has a Born-Mayer form (see chapter 1). Finally, it ignores induction interactions as it has been shown that the induced energy in the physisorption of spherical atoms is always a small quantity compared to the dispersion energy [56].

The PN1 model seems to be an appropriate model as suggested by simulations of

argon adsorption at low coverages, in good agreement with experimental data. It predicts pores 0.06nm narrower than the Kiselev model although the spatial distribution of sites remains more or less unchanged [54].





Figs. 6,7 Comparison between simulation and experiment: adsorption isotherms (left) and isosteric heat curves (right) at 77.4 K. In the latter, the line represents the experimental curve and the symbols the simulation results. Diamonds represent the molecule-molecule and squares the molecule-wall parts of the isosteric heat as predicted by the PN1 model. The Kiselev isosteric heat prediction is shown with inverted triangles and the PN1 with black dots (reprint from ref. 51).

In a simulation, the adsorbate-adsorbate interactions are accounted in the form of an effective Lennard-Jones potential. Neither the Kiselev not the PN1 model are capable, however, of reproducing the experimental substep. The former reaches a plateau at an uptake of thirty-one molecules and the latter at the coverage just before the phase transition (around twenty-three argons per unit cell). The agreement between the Kiselev model and experiment for the maximum uptake could be regarded as fortuitous since it is known that the model predicts both pores far too wide and uses an incorrect value for the polarizability of oxygen.

Recently it has been proposed that the substep might be a consequence of an adsorbate-induced phase transition as it happens for other molecules such as p-xylene. This hypothesis seems to be confirmed by neutron scattering data which shows peaks originating from the adsorbent phase during the transition [55].

The isosteric heat curves predicted by the PN1 and Kiselev models follow the experimental trend but at the same time are not completely correct since both predict an almost linear rise of the isosteric heat up to the transition uptake (see fig. 7).

From the previous paragraphs it is possible to infer an improvement of the PN1 model. While it is quite rigorous in the treatment of the adsorbate-adsorbent part of the potential, it needs to rely on an effective Lennard-Jones function for the adsorbate-adsorbate interactions at high coverages. Disagreement between simulation and experiment might very well arise from this poor treatment of the adsorbate potential.

The purpose of the work presented in this paper is to study in more detail the argon two- and three-body interactions in the hope that it might give us more insight about the present discrepancies between simulation and experiment.

The proposed transition to a solid-like phase at high coverages could be triggered by the intermolecular interactions between argon atoms inside the cavity in a way not predicted by a simple effective potential. In the same vein, a more rigorous approach to the adsorbate-adsorbate potential could also help in refining the shape of the theoretical isosteric heat curve. Three-body forces, generally repulsive, might become important at intermediate coverages causing a decrease in the argon-argon potential and therefore a decrease in the isosteric heat. That would ideally trigger a flattening of the theoretical curve for the isosteric heat and therefore a better agreement with experiment.

The ultimate goal would be to be able to incorporate three-body effects into a full-scale molecular simulation, something impossible to do if we were to use the explicit functions for the two- and three-body potentials. One has to go back now to the idea of an effective potential discussed in the first chapter. Such an effective potential would be dependent on coverage since, as we shall show, three-body effects depend on this variable.

The chapters to follow are organised at follows:

-discussion of the possible potential functions that could be used in the accurate description of the intermolecular potential for argon up to three-body terms.

-choosing the right combination of potential functions by performing a preliminary calculation on a close-packed, centred-cubic argon lattice.

-evaluation of three-body effects of the type argon-argon-argon, argon-argon-oxygen and argon-argon-silicon in the adsorption of argon in silicalite-1 .

-attempting to incorporate three-body effects into a full-scale Grand Canonical Monte Carlo (GCMC) simulation via an effective Lennard-Jones potential which varies as function of coverage. And finally, application of this effective potential to the refinement of the isosteric heat curve.

Chapter 3. An accurate adsorbate-adsorbate potential for argon in silicalite-I.

As it has been pointed out before, the PN1 potential models quite accurately the adsorbate-adsorbent part of the interaction energy. This has been corroborated recently by Extended-Hückel calculations in silicalite-1 [58]. It is interesting now to focus on the adsorbate-adsorbate part and depart from the far too simplistic form of the effective Lennard-Jones potential.

Initially all the potential functions investigated will be presented and then tested in an argon lattice. This will ensure a proper choice of the explicit functions that will be used throughout the study of the silicalite system. We have used both atomic (a.u.) and SI units throughout this section (1 a.u. energy = 1 Hartree = $3.1578e^{+5}$ K and 1 a.u. distance = 1 Bohr = 0.52917 Å).

3.1 Available potential functions for argon.

3.1.1 Effective Lennard-Jones potential.

A lot of effort has gone into the development of an accurate potential for argon. In the early days most research was focused on finding the right parameters for a Lennard-Jones potential that could reproduce the properties of condensed argon [82-85]. The final fruition of this idea came at the end of the 1940s [86] with the parameters which are even today used in the effective Lennard-Jones potential for argon (ε = 120 K; σ = 3.405 Å).

3.1.2 Two-body potentials.

New experimental data in the 1950s and 1960s showed the inadequacy of this effective potential to account for some of the physical properties of argon. Early attempts did not meet with success [87] due in part to the systematic error found later in gas transport coefficient data. It soon became apparent that in order to have good agreement with experiment it was required to use an accurate two-body potential and to include three-body terms. The problem of the two-body potential for argon can be considered resolved after Barker, Fisher and Watts [63,25]. Subsequent work on argon has gone in two different directions. On the one hand, those attempting to find a simpler potential form for the potential of Barker-Fisher and Watts [88,89]. On the other hand, there are those such as Tang and Toennies and more recently Aziz et al. who have departed from a purely empirical approach and attempted to create semiempirical models of dispersion interactions [65,68,70,72,73]. In this section we will introduce three different two-body potentials. Two of them, the two-body Lennard-Jones and the Barker-Fisher-Watts are purely empirical. The semiempirical one is due to the work of mainly Tang and Toennies

3.1.2.1 The true two-body Lennard-Jones potential.

Its mathematical form has been discussed previously (see chapter 1). It consists of two adjustable parameters set at ε = 142.1 K and σ = 3.347 Å. This function is supposed to reproduce the two-body interaction between two argon atoms quite satisfactorily [62].

3.1.2.2. The empirical Barker-Fisher-Watts potential.

During the late 1960s and early 1970s a great deal of effort went into creating high-quality two-body potentials for the noble gases. Barker et al., using a vast amount of experimental data including molecular-beam scattering and second virial coefficients, found what is even today known as the most reliable potential for argon [25,63]. The potential has the following form:

$$U(r) = \epsilon(\exp[\alpha(1-r)]\sum_{i=0,5} A*_i(r-1)^i - \sum_{i=0,2} \frac{C*_{2i+6}}{(\delta + r^{2i+6})} \qquad (13)$$

In this expression $r = R/R_m$ and R_m is the separation at the minimum. ε is the well depth as usual. Other parameters which might not be familiar include α, the steepness of the repulsive wall and δ, a constant term inserted to prevent the occurrence of a spurious maximum at small separations. The table below displays the parameters for argon. Similar high-quality potentials have been developed for krypton and xenon [64].

Table 1. Barker-Fisher-Watts coefficients in reduced units (ref. 56)

C_6*	-1.10727	A_0*	0.27783
C_8*	-0.16971	A_1*	-4.50431
C_{10}*	-0.0361	A_2*	-8.33122
α	12.5	A_3 *	-25.2696
δ	0.01	A_4*	-102.0195
ε	142.1 K	A_5*	-113.2500
R_m	3.761 Å		

3.1.2.3 The Tang and Toennies semi-empirical potential.

Despite the high accuracy of the Barker-Fisher-Watts function it still remains a purely empirical potential and provides little insight into the nature of the dispersion interaction in argon. In recent years there have been significant advances in the development of a semi-empirical theory of intermolecular potentials. The model introduced by Tang and Toennies [65] is probably the most popular and has proven to be successful in systems such as H_2 and He_2. The PN1 model uses this form to account for the dispersion energy between the zeolite and the adsorbate.

The explicit form of the potential is given below:

$$U_{ij}(r) = A\exp(-br) - \sum_{n\geq 3} f_{2n}\frac{C_{2n}}{r^{2n}} \qquad (14)$$

The first term is the celebrated Born-Mayer expression for the repulsive part of the potential. The constants A and b are, to a first approximation, parameters fitted to the self-consistent field potential. The second term corresponds to the long-range attractive part and is composed of the dispersion terms C_{2n}/r^{2n} multiplied by a damping function f_{2n}.

$$f_{2n}(R) = 1 - \left(\sum_{k=0}^{2n} \frac{(bR)^k}{k!}\right) \exp(-bR) \qquad (15)$$

The damping function ensures that the attractive part vanishes at sufficiently small separations, where the multipolar expansion is proned to diverge. It is a function between zero and one that switches the long-range potential on and off [66-68]. In the Tang and Toennies potential the b parameter in the damping function is the same as that in the repulsion term. Some workers, however, treat this parameter in the damping function as an extra adjustable constant [69].

There are other semi-empirical potentials proposed for argon such as that of Aziz et al [70]. However, these are generally of a more complicated form, i.e the Aziz potential includes a second-order polynomial expansion of the Born-Mayer exponent. We preferred, however, to retain the relative simple form of the Tang and Toennies that seems to take account of all the factors predicted by the theory (i.e. the Born-Mayer form of the potential).

Table 2 shows the values for the dispersion and Born-Mayer coefficients for the two-body potential of argon. The last column contains the best dispersion coefficients up to C_{10}.

Table 2. Dispersion coefficients and repulsive parameters for the Tang-Toennies potential.

Coefficients[a]	Tang-Toennies[b]	Ab-initio[c]
C_6	67.2	68.30
C_8	1610	1142
C_{10}	42700	38559
A	385.08	328.85[d]
b	1.917	1.918[d]

a. all coefficients in atomic units.
b. ref. 65
c. refs. 106, 107
d. obtained from Ar_2 SCF potentials, ref. 92

It is worthwhile at this point to briefly outline how the dispersion coefficients have been estimated. A recent and comprehensive description of the procedure can be found in ref.

The dispersion coefficients account for the interactions between different multipole moments. For example, C_6 describes the dipole-dipole interaction while C_8 corresponds to the dipole-quadrupole and C_{10} takes care of the dipole-octupole and quadrupole-quadrupole. For the general case of two spherical atoms A and B they might be written as:

$$\mathbf{C_6 = C_{AB}(1,1)} \tag{16}$$

$$\mathbf{C_8 = C_{AB}(1,2) + C_{AB}(2,1)} \tag{17}$$

$$\mathbf{C_{10} = C_{AB}(1,3) + C_{AB}(2,2) + C_{AB}(3,1)} \tag{18}$$

where the numbers within brackets indicate the 2^1, 2^2, 2^3 moments of the atom. The coefficients $C_{AB}(l_1, l_2)$ are given by [72-74]:

$$C_{AB}(l_1, l_2) = \frac{(2l_1 + 2l_2)!}{4(2l_1)!(2l_2)!}\left(\frac{2}{\pi}\right)\int_0^\infty \alpha_{l_1}{}^A(i\omega)\,\alpha_{l_2}{}^B(i\omega)\,d\omega \tag{19}$$

where $\alpha_{l1}{}^A$ and $\alpha_{l2}{}^B$ are the 2^{l1}- and 2^{l2}-pole dynamic polarizabilities at a frequency ω. The frequency-dependent polarizabilities are then defined as:

$$\alpha_l(\omega) = \sum \frac{f_n^l}{(\delta_n^l)^2 - \omega^2} \tag{20}$$

that is, the sum of all the f_n^l oscillator strengths for the 2^l-th pole transition from the ground state to the n-th state, with a transition energy δ_n^l. The static polarizabilities are obtained at zero frequency (i.e. $\omega = 0$). Since the polarizability is a decaying function of frequency Tang [72] has shown that it can be approximated with small error using a one-term Padé approximant which reads as follows:

$$\overline{\alpha_l}(i\omega) = \frac{\alpha_l(0)\cdot\eta_l^2}{\eta_l^2 + \omega^2} \tag{21}$$

where η_l is the average energy parameter for the 2^l-pole transition which may be defined as:

$$\eta_l = \sqrt{\frac{S_l(0)}{\alpha_l(0)}} \quad where \quad S_l(k=0) = \sum_n f_n^l (E^l)^{(k=0)} \tag{22}$$

According to the expression shown above, $S_l(0)$ is reduced to a sum of oscillator

strengths. When l is unity it can be identified with the number of effective electrons, that is, those electrons which contribute to the polarization of the atom more vigorously. It usually corresponds to the number of electrons in the outer subshell. Thus, the $C_{AB}(l_1, l_2)$ terms can be written as:

$$C_6^{AB}(l_1, l_2) = \frac{(2l_1 + 2l_2)!}{4(2l_1)!(2l_2)!} \frac{\eta_{l_1}{}^A \eta_{l_2}{}^B}{\eta_{l_1}{}^A + \eta_{l_2}{}^B} \alpha_{l_1}{}^A(0) \alpha_{l_2}{}^B(0) \qquad \textbf{(23)}$$

Calculation of the C_6, C_8 and C_{10} terms depends on the knowledge of the static dipole, quadrupole and octupole polarizabilities and also of the $S_1(0)$, $S_2(0)$, $S_3(0)$ terms. An empirical recursion relation to evaluate higher coefficients has been proposed by Tang and Toennies [65]. It reads as shown:

$$C_{2n+4} = \left(\frac{C_{2n+2}}{C_{2n}}\right)^3 C_{2n-2} \qquad \textbf{(24)}$$

The number of effective electrons for spherical and isolated atoms can be estimated from accurate Coupled Hartree-Fock calculations (CHF) [75]. For argon, the element of interest to us, this figure is 6.106 as it is expected from Tang's theory. In the case of the oxygen and silicon in the framework it is necessary to add a -1 and a +2 to the effective number in the neutral form respectively to take account of their charges. For oxygen it is 4.656 and for silicon 1.525.

The static polarizabilities are readily available for the noble gases [76,81] and the zeolitic oxygen and silicon polarizabilities can be obtained from Auger Electron spectroscopy [53]. All the parameters required for the argon-argon, argon-oxygen, argon-silicon, oxygen-oxygen, oxygen-silicon and silicon-silicon two-body interaction have been discussed in detail elsewhere [52].

3.1.3 Third-order three-body potentials.

Even though most of the energy for an ensemble of molecules in condensed phases comes from the pairwise potential, contributions from three-body terms are not negligible, especially at high densities [79]. For argon, it was Barker et al. [63] who first pointed out the deficiencies of not including three-body terms. Using the Barker-Fisher-Watts two body potential and the Axilrod-Teller-Muto triple-dipole term led to good agreement with experimental data of liquid argon [25]. Similar results can be obtained if the effective Lennard-Jones described above is used instead. In the case of adsorption, Kim and Cole [59] and Nicholson [60,61] have shown that the three-body energy can be twenty percent of the total energy and therefore it is by no means negligible.

The non-additive, third-order, three-body, long-range dispersion interactions arise from the third term in a perturbative treatment of the intermolecular potential. As the equation below shows, they can be expressed as the product of a geometrical factor $W^{(3)ABC}$, dependent solely on the positions of the atomic nuclei, and an interaction constant $Z^{(3)ABC}$ which depends on the atomic species involved in the interaction [77].

$$E^{(3)\,ABC} = \sum_{l_1} \sum_{l_2} \sum_{l_3} Z^{(3)\,ABC}(l_1, l_2, l_3)\, W^{(3)\,ABC}(l_1, l_2, l_3) \tag{25}$$

There is a total of eleven third-order, three-body dispersion terms which account for all the possible interactions between the instantaneous dipoles, quadrupoles and octupoles of three particles. These are: dipole-dipole-dipole (1), dipole-dipole-quadrupole (3), dipole-quadrupole-quadrupole (3), quadrupole-quadrupole-quadrupole (1) and dipole-dipole-octupole (3).

3.1.3.1 The Geometric Functions.

We display in this section the explicit form of the geometric functions for three spherical particles forming a triangle of sides R_{12}, R_{13}, R_{23} and interior angles ϕ_1, ϕ_2, ϕ_3 [77,78].

-dipole-dipole-dipole (DDD):

$$W^{(3)}(DDD) = 3 R_{12}^{-3} R_{23}^{-3} R_{31}^{-3} \; (1 + 3 \cos\phi_1 \cos\phi_2 \cos\phi_3) \tag{26}$$

-dipole-dipole-quadrupole, including a total of three permutations (DDQ, DQD, QDD):

$$W^{(3)}(DDQ) = \frac{3}{16} R_{12}^{-3} R_{23}^{-4} R_{31}^{-4} \; [\,(9 \cos\phi_3 - 25 \cos 3\phi_3) + 6\,[\cos(\phi_1 - \phi_2)]\,[3 + 5 \cos 2\phi_3]\,] \tag{27}$$

-quadrupole-quadrupole-dipole, including a total of three permutations as well (DQQ,QDQ, QQD):

$$W^{(3)}(QQD) = \frac{15}{64} R_{12}^{-5} R_{23}^{-4} R_{31}^{-4} [3(\cos\phi_3 + 5\cos 3\phi_3) + 20\cos(\phi_1 - \phi_2)(1 - 3\cos(2\phi_3) + 70\cos 2(\phi_1 - \phi_2)\cos\phi_3] \quad (28)$$

-quadrupole-quadrupole-quadrupole (QQQ):

$$W^{(3)}(QQQ) = \frac{15}{128} R_{12}^{-5} R_{23}^{-5} R_{31}^{-5} [-27 + 220\cos\phi_1\cos\phi_2\cos\phi_3 + 490\cos 2\phi_1\cos 2\phi_2\cos 2\phi_3 + 175(\cos 2(\phi_1 - \phi_2) + \cos 2(\phi_2 - \phi_3) + \cos 2(\phi_3 - \phi_1))] \quad (29)$$

-dipole-dipole-octupole (DDO, DOD, DDO):

$$W^{(3)}(DDO) = \frac{5}{32} R_{12}^{-3} R_{23}^{-5} R_{31}^{-5} [9 + 8\cos 2\phi_3 - 49\cos 4\phi_3 + 6\cos(\phi_1 - \phi_2)(9\cos\phi_3 + 7\cos 3\phi_3)] \quad (30)$$

3.1.3.2 The Electronic Functions.

Tang [73] has extended the scheme described before to determine the dispersion coefficients of the three-body dispersion interactions. These coefficients are of the form shown below:

$$Z^{(3)ABC}(l_1, l_2, l_3) = \frac{1}{\pi}\int_0^\infty \alpha_{l_1}{}^A(i\omega)\,\alpha_{l_2}{}^B(i\omega)\,\alpha_{l_3}{}^C(i\omega)\, d\omega \quad (31)$$

As we saw before, the approximate value of this function can be obtained from a knowledge of the static polarizabilities and the number of effective electrons if the Padé approximant is used. The electronic term then becomes:

$$Z^{(3)ABC}(l_1, l_2, l_3) = \frac{1}{2}\alpha_{l_1}{}^A(0)\,\alpha_{l_2}{}^B(0)\,\alpha_{l_3}{}^C(0)\;\eta_{l_1}{}^A\,\eta_{l_2}{}^B\,\eta_{l_3}{}^C \times \frac{\eta_{l_1}{}^A + \eta_{l_2}{}^B + \eta_{l_3}{}^C}{(\eta_{l_1}{}^A + \eta_{l_2}{}^B)\;(\eta_{l_2}{}^B + \eta_{l_3}{}^C)\;(\eta_{l_3}{}^C + \eta_{l_1}{}^A)} \quad (32)$$

3.1.4 The fourth-order term.

We have included in the calculation a dipole-dipole-dipole term that arises from the fourth term in the perturbation expansion [79]. The explicit form of the geometric function is shown below:

$$
\begin{aligned}
W^{(4)}(DDD) &= Z_1^{(4)} \frac{5}{4} R_{12}^{-6} R_{23}^{-6} \left[1 + \cos^2\phi_3\right] \\
&+ Z_2^{(4)} \frac{5}{4} R_{23}^{-6} R_{13}^{-6} \left[1 + \cos^2\phi_1\right] \\
&+ Z_3^{(4)} \frac{5}{4} R_{13}^{-6} R_{12}^{-6} \left[1 + \cos^2\phi_2\right]
\end{aligned}
\qquad (33)
$$

The electronic functions $Z^{(4)}(DDD)$ depend on the identity of the interacting particles and can be calculated as shown by MacRaury et al. [80].

The table displayed below shows the values for the electronic functions in the three cases of relevance to us, namely, argon-argon-argon, argon-argon-oxygen and argon-argon-silicon.

Table 3. Three-body electronic functions in atomic units.

Electronic functions	ArAr Ar	ArArO	ArArSi
$(DDD)_3$	189.1	138.3	41.31
$(QDD)_3$	687.7	501.8	149.3
$(DDQ)_3$	687.7	511.1	64.74
$(DQD)_3$	687.7	501.8	149.3
$(QQD)_3$	2616	1906	565.7
$(QDQ)_3$	2616	2131	218.2
$(DQQ)_3$	2616	2131	218.2
$(QQQ)_3$	10295	9022	787.3
$(DDD)_4(1)$	-7854	-5730	-1705
$(DDD)_4(2)$	-7854	-5730	-1705
$(DDD)_4(3)$	-7854	-4181	-370.2

3.2 Testing the potentials in a close-packed, face-centred cubic argon lattice.

From the last section we have three candidates for the two-body potential function. There is not a similar choice for the three-body terms described above since they are the most accurate expressions available at the present time.

The three potentials were tested in a close-packed, face-centred cubic argon lattice with a van der Waals radius for argon of 1.88 Å [94]. This system has been known relatively well for many years. Calculations have shown that the magnitude of the non-additive contributions to the lattice energy are quite large and cannot be neglected [91]. Overall, the non-additive contribution is positive and most of it comes from the third-order, triple-dipole term. The contribution of the fourth-order, triple-dipole term was found to be negative, almost exactly cancelling the total contribution of third-order dipole-dipole-quadrupole, dipole-quadrupole-quadrupole and quadrupole-quadrupole-quadrupole terms.

Appendix I describes the package of programs used to construct the lattice and evaluate the energies for the different potential models. Figure 8 below shows the results in terms of deviations from the effective Lennard-Jones potential, known to reproduce both the solid and liquid properties of argon quite well [90]. The reported deviations have been determined as shown by equation 34:

$$\%\,deviation = \frac{(TrueEnergy - EffectiveEnergy)}{Effectiveenergy} \qquad (34)$$

where 'true' denotes the quantity obtained by using a high-quality two-body potential and the three-body functions previously discussed. This quantity is a measure of how far the two-body plus three-body energy is from the one obtained with an effective Lennard-Jones.

As expected, the Barker-Fisher-Watts potential works very satisfactorily and the deviation from the effective potential is negligible as soon as the number of atoms in the lattice becomes sufficiently large. It is a well-known phenomenon that an accurate two-body potential like this one together with at least the Axilrod-Teller-Muto three-body term leads to accurate predictions of the properties of argon in gaseous, liquid and solid state as well as for other noble gases like neon, krypton and xenon in the solid state [95-97].

The two-body Lennard-Jones potential gives a poorer performance manifested as a larger positive deviation from the effective potential around ten percent. The failure of this function can be attributed to its wrong shape, wider in the well than the Barker-Fisher-Watts thus predicting a more stable lattice.

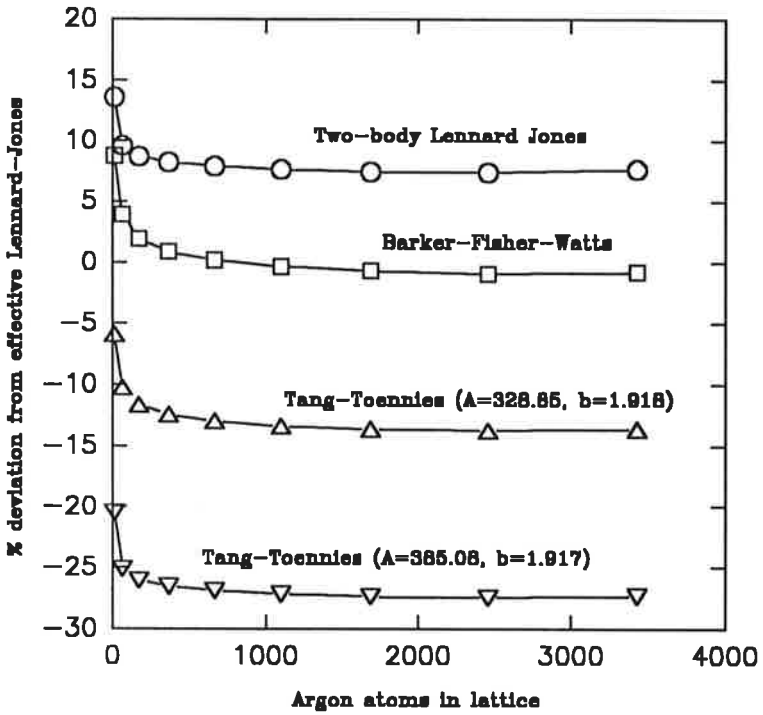


Fig. 8 Percent deviations for four two-body potential functions plus three-body terms from the effective Lennard-Jones in the argon lattice. For the Tang-Toennies two different sets of repulsive parameters have been used (see Table 2).

The striking fact in figure 8 is the extremely poor performance of the Tang-Toennies potential which is much more repulsive than the Barker-Fisher-Watts for the two sets of repulsive parameters. Several factors are strong candidates for the failure of this potential function. First only three terms are included in the expansion of the potential, namely C_6, C_8 and C_{10} and second is the use of the wrong parameters in the Born-Mayer repulsive part of the potential. While the dispersion coefficients were calculated following the procedure described in the previous section, the repulsive parameters chosen were those from ab-initio calculation found in the literature [92]. Thus these values might not be consistent with our dispersion coefficients. Figure 9 shows the Barker-Fisher-Watts potential plotted along with Tang-Toennies potentials with different sets of repulsive parameters. Even in the cases when consistent repulsive and attractive parameters taken from the Tang-Toennies theory and ab-initio calculation are used, the disagreement is still around 20 K at the equilibrium distance. Both the Tang-Toennies and ab-initio parameters do equally badly with respect to the Barker-

Fisher-Watts potential. Consequently, the disagreement between the Barker-Fisher-Watts and the Tang-Toennies potential is likely to arise from a neglect of dispersion terms higher than C_{10}. To show the effect of mixing the parameters from two different theoretical treatments, we have included the potential that results from having the Tang-Toennies dispersion coefficients and the ab-initio Born-Mayer parameters. The result is disastrous, with a shift in the epsilon of roughly 40 K.

There are several problems if we try to include more dispersion terms in the Tang-Toennies. Firstly, there are no accurate values for polarizabilities beyond the octupole available. Tang and Toennies have proposed a semiempirical recursion formula to calculate higher order terms [65]. This, however, is not a very reliable procedure and the accuracy of higher order terms would not be by any means as good as that of the first three terms. Secondly, the damping functions, as recent investigations have shown [93], are ill-defined beyond the fourth dispersion term.

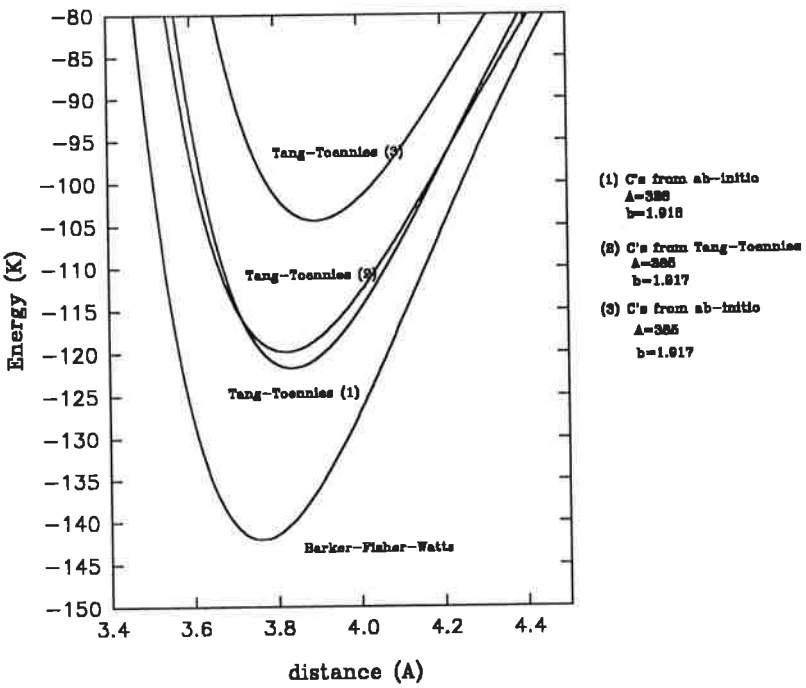


Fig.9 Barker-Fisher and Watts potential vs. Tang-Toennies (1-3). The different sets of repulsive parameters were used for the Tang-Toennies as indicated above.

Recapitulating, the long-range, attractive part of the Tang-Toennies potential is known quite accurately up to the third dispersion term. It seems necessary at this point to find the Born-Mayer parameters that give the correct behaviour, namely the potential minimum at 142.1 K. Appendix II contains the programs used in the finding of the repulsive parameters. The first approach used was that proposed by Tang-Toennies [65]. According to this procedure, if the dispersion potential is written in reduced units ($x = R/R_m$, $A^* = A/\varepsilon$, $C_{2n}^* = C_{2n}/\varepsilon R_m^{2n}$) then at the minimum (reduced distance equal to one) the reduced potential and its derivative are:

$$U^*(1) = -1 \tag{35}$$

$$\left[\frac{dU^*(x)}{dx} \right]_{x=1} = 0 \tag{36}$$

This is a system of two equations with A and b, the repulsive parameters, as unknowns. The program written to implement this idea was unable to reach convergence to reasonable values of the potential (see Appendix II).

The second method used was a variation of the well-known Newton-Raphson algorithm to find the roots of a equation. In this method, the potential and its derivative at the equilibrium distance, both in reduced units, can be regarded as functions of A* and b*, the reduced repulsive parameters. From this it follows that:

$$U^*(A^*, b^*, x=1) = -1 + \frac{dU^*}{dA^*} dA^* + \frac{dU^*}{db^*} db^* \tag{37}$$

$$\frac{dU^*(A^*, b^*, x=1)}{dx} = \frac{d}{dA^*}\left(\frac{dU^*}{dx}\right) dA^* + \frac{d}{db^*}\left(\frac{dU^*}{dx}\right) db^* \tag{38}$$

Starting with the values for A and b given in the literature it is possible, at least in principle, to solve these two equations iteratively until convergence. Unfortunately, the procedure did not work either. One could justify the failure of these two rigorous attempts to find the repulsive coefficients as being probably due to the discontinuous behaviour of the partial derivatives of the potential function with respect to A and b.

From a more qualitative viewpoint, a look at the figure displaying the Barker-Fisher-Watts and the Tang-Toennies potentials (fig. 9) will reveal that the shapes of both potentials are quite similar despite the fact that one is roughly 20 K above the other. Also, from the functional form of the Born-Mayer potential it is clear that the position of the Tang and Toennies potential is mostly dependent on the b parameter, the one in the exponential argument. Adjusting this parameter so that it brings the potential down to the right well depth shows good agreement with the Barker-Fisher-Watts potential (see fig. 10).

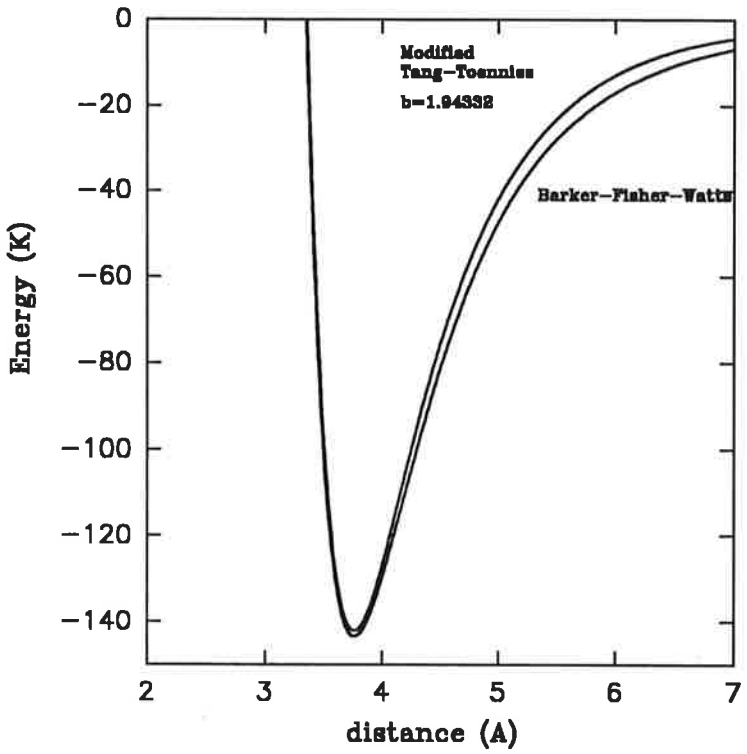


Fig. 10 Barker-Fisher-Watts potential and modified Tang-Toennies. Note the good agreement in the vicinity of the potential minimum and the slight disagreement at longer distances.

The Tang-Toennies potential with the new repulsion parameters has become at this point an almost empirical potential. The Born-Mayer parameters are valid as long as the series is truncated after the third term and the dispersion coefficients are those that were calculated according to the procedure described in the previous section. In many respects, this potential function has lost most of the features that led us to consider it in our preliminary studies of the argon potential functions. We have therefore chosen the Barker-Fisher-Watts potential as the 'true' two-body potential in future calculations. Excellent results of this function for the argon lattice secure reliable results in the novel silicalite-1 system.

Chapter 4. Three-Body interactions in the adsorption of argon in silicalite-1.

4.1 Brief description of the calculation.

A description of the specifics of the calculation has been provided in Appendix III. In very broad terms, the role of three-body forces was assessed as follows. The system was comprised of twenty-seven unit cells of the orthorhombic form of silicalite-1 filled with argon at different coverages. The argon configurations were obtained from previous simulations using a Grand-Canonical Monte-Carlo (GCMC) simulation (program msimu8.f) at 77.4 K and covered a range from one to approximately thirty argon atoms per unit cell. Starting with these configurations, the GCMC simulation was used to move the argon atoms in the cavities but this time not allowing for either creation nor destruction of adsorbate particles, in essence a canonical (NVT) simulation. Every one hundred-thousand adsorbate configurations or so in the simulation, the snapshot was subjected to a full calculation of the two-body and three-body energy for the argon atoms located in the central unit cell (program shcalc4.f). This calculation would then determine not only the absolute values of the energies but also the deviations of the Barker-Fisher-Watts and two-body Lennard-Jones plus three-body terms from the effective potential. The procedure was repeated for a total of twelve runs at each coverage and the mean energies and deviations estimated from them.

The calculation was extended later to examine the behaviour of the different parts of the unit cell, that is, the straight channels, intersections and sinusoidal channels by performing the same type of calculation described above for those atoms occupying these regions. From these calculations it was possible to determine:

-the distribution of argon atoms in the unit cell as a function of coverage.
-the deviations of the most accurate potential function (the Barker-Fisher-Watts plus three-body terms) from the effective Lennard-Jones function as a function of coverage.
-the contribution of three-body interactions to the overall potential.
-the relative importance of argon-argon-argon, argon-argon-oxygen and argon-argon-silicon three-body interactions.
-the contribution of different three-body terms (i.e.dipole-dipole-dipole,dipole-dipole-quadrupole, etc) to the overall three-body energy.

The detailed discussion of all these topics is the subject of most of this chapter. But before doing so, it is necessary to discuss some of the details of the calculation.

4.2 An analysis of the calculation parameters: periodic boundaries and long-range cutoffs.

As it has been pointed out previously, the zeolite system was composed of a total of twenty-seven unit cells. However, it is questionable whether this fragment of the zeolite is sufficiently large to avoid spurious effects in the calculation. It is common practice in a simulation to replicate the simulation box several times to wash out the effect of having a 'vacuum' surrounding the system. The programs have been written so that the number of replicas of the initial twenty-seven unit cells can be easily modified.

Another important question is in regard to the long-range cutoffs in the potentials. In a calculation of the sort described in this work it is necessary to disregard the interactions beyond a certain distance from the particle of interest, the choice justified by the need to speed up the calculation. Especially in the case of the three-body interactions, inclusion of all the interactions would be quite inefficient from a computational viewpoint. In making the decision as to which should be the cut-off distance one has to make sure that the contribution of all these interactions beyond the cut-off is negligible.

The table shown below shows the results from an analysis of the effect of periodic boundaries and different long-range cut-offs in the final results. We have chosen the Lennard-Jones two-body potential as our reference. Three-body forces have a stronger distance dependence and tend to disappear long before the two-body interactions become negligible. We have also chosen high-pressure configurations which will enable us to probe different distance ranges more accurately since they are not subject to as large a fluctuation as in the number of adsorbate particles as it would be the case of a low-coverage configuration.

Table 4. Analysis of periodic boundaries and long-range cutoffs.

<Ar> / unit cell	Number of replicas	Cut-off dist. (Bohr)	Two-body Lennard-Jones energy argon (K)
20	0	20	-267.8
20	0	40	-278.7
20	0	60	-279.5
20	0	80	-279.5
20	1	20	-267.8
20	1	40	-278.8
20	1	60	-279.9
20	1	80	-280.1

From the results presented in this table, it is clear that the Lennard-Jones two-body energies are calculated with negligible error with no replicas around the original twenty-seven unit cell system and with a cut-off of 40 Bohr. These results are quite significant in terms of the use of computer resources since the choice of longer cutoff and the inclusion of periodic boundaries would slow down the calculation significantly.

4.3 Results from the calculation.

4.3.1 Distribution of argon atoms in the unit cell.

Figure 11 shows the average number of argon atoms in each region of the unit cell as

a function of coverage.

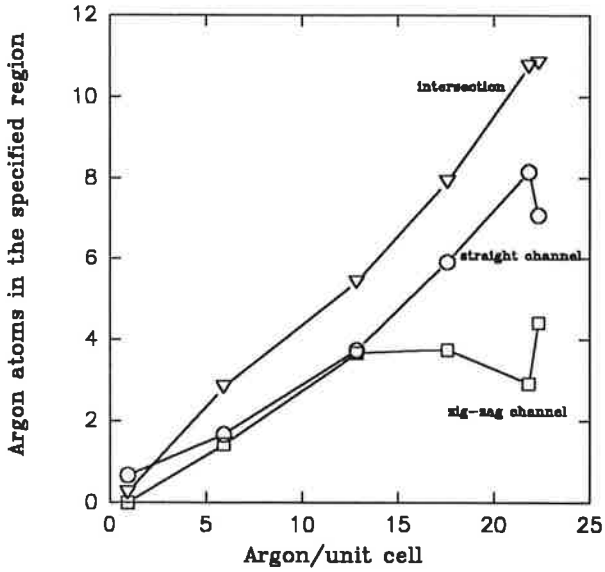


Fig. 11 Distribution of argon atoms in the three different regions of the unit cell.

These results agree quite well with a previous 'snapshot' analysis of the sorption of argon in the zeolite [98]. From the plot the following generalisations are possible:

-at low loading, circa two argon atoms per unit cell, the strongest sites are located in the centre of the straight channels.
-at around eight molecules per unit cell the argon atoms start moving into the sinusoidal channels as well. Consequently, the straight and sinusoidal channels become equally loaded.
-at higher coverages, approximately sixteen molecules per unit cell, adsorption occurs into most of the straight pores yet the intersection becomes the most preferred site due to a more favourable adsorbate-adsorbate interaction.
-finally the depleted regions are occupied with a maximum loading of twenty-three argon atoms per unit cell.

This picture agrees with the idea of non-localised adsorption. The location of the most favourable sites not only depends on the characteristics of the pore but also in the number of adsorbate atoms in the cavity. Preferred sites are a sensitive function of coverage.

4.3.2 The importance of three-body forces.

The table below shows the contributions of three-body terms to the total adsorbate energy for different coverages. The percentage increases slightly for high coverages but in broad terms it stays around seventeen and twenty percent, a non-negligible amount similar to that found for other sorption systems [59-61] and larger than what is typical for the argon solid and liquid phases.

Table 5. Non-additive contribution to the total energy.

<Argon atoms in unit cell>	% total energy from three-body terms
5.9	18.0
12.8	17.7
17.6	17.3
21.8	18.6
22.3	20.2

These results are also quite different from previous work on three-body interactions in zeolites. H. Kono and A. Takasaka [105] found that in the case of argon adsorption in zeolite 4A the contribution of adsorbate dispersion and adsorbate-adsorbent induction interactions was negligible. This is definitely not the case for the argon-silicalite system.

4.3.3 Deviations from the effective Lennard-Jones potential.

Probably the most important piece of information from these calculations is the deviation of the best estimate for the energy of the system (Barker-Fisher-Watts plus three-body terms) from the effective Lennard-Jones energy (fig. 12). The reported deviations have been determined as already explained in section 3.2.

From fig. 12 it is clear that argon in silicalite-1 behaves quite differently from solid argon. The figure not only includes the 'true' energy computed with the Barker-Fisher-Watts but also with the two-body Lennard-Jones potential for comparison. In both cases the computed deviation is negative, indicating a more repulsive energy than that predicted with the effective Lennard-Jones. The fact the two-body Lennard-Jones always yields a less repulsive energy due to a wider potential well with respect to the Barker-Fisher-Watts potential coincides with the findings for the argon lattice discussed in previous sections.

As the calculations in the argon lattice showed, the Barker-Fisher-Watts deviation is for all practical purposes very close to zero for solid argon. Thus any differences between the true and effective energies in silicalite-1 can be attributed to the role of three-body forces in

the adsorbate-adsorbate potential inside the cavity. To show this we can compare the deviations in fig. 12 with those obtained for the argon lattice. The two-body Lennard-Jones plus three-body is in both cases about ten percent more attractive. Thus, in order to correct the deviations for the two-body Lennard-Jones we could add ten percent repulsion energy to it, bringing it to agreement with the Barker-Fisher-Watts. Thus, to a first-order approximation, the deviations of the Barker-Fisher-Watts from the effective Lennard-Jones can be attributed to mostly three-body, non-additive effects.

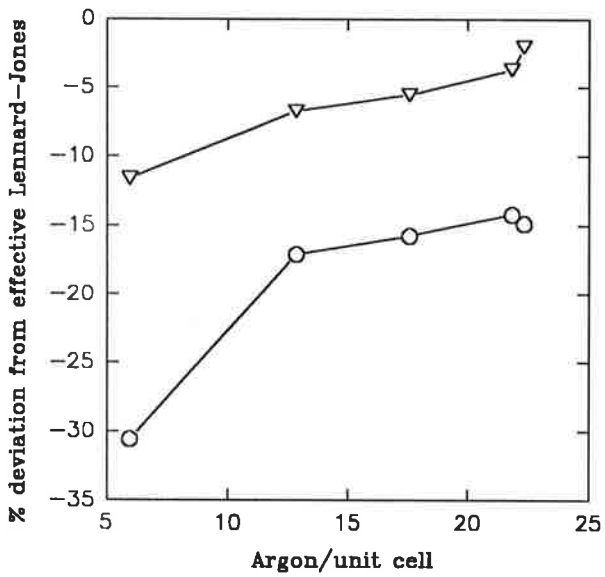


Fig. 12 Deviations from effective Lennard-Jones potential with coverage. Barker-Fisher-Watts (circles) and two-body Lennard-Jones (triangles).

Fig. 12 also shows larger deviations for the low coverage region, the deviation then acquiring a monotonically decreasing shape with coverage. It should be mentioned, however, that the results at very low coverages are subjected to quite large standard deviations from the mean value obtained from the total of twelve runs. This effect can be attributed to the very small value of the adsorbate energy at such coverages as well as to very large fluctuations in the number of molecules in the unit cell where the calculation was performed. For example, in the case of loadings of two or three argon atoms per unit cell a fluctuation in the number of molecules by one or two argon atoms can cause dramatic changes in the total energy and translate into large standard deviations from the computed mean value.

Beyond coverages of approximately twenty-three argon atoms per unit cell (not included in the figure) there were also great fluctuations in the final results. The source of this anomalous behaviour is, however, of a very different nature from the one previously

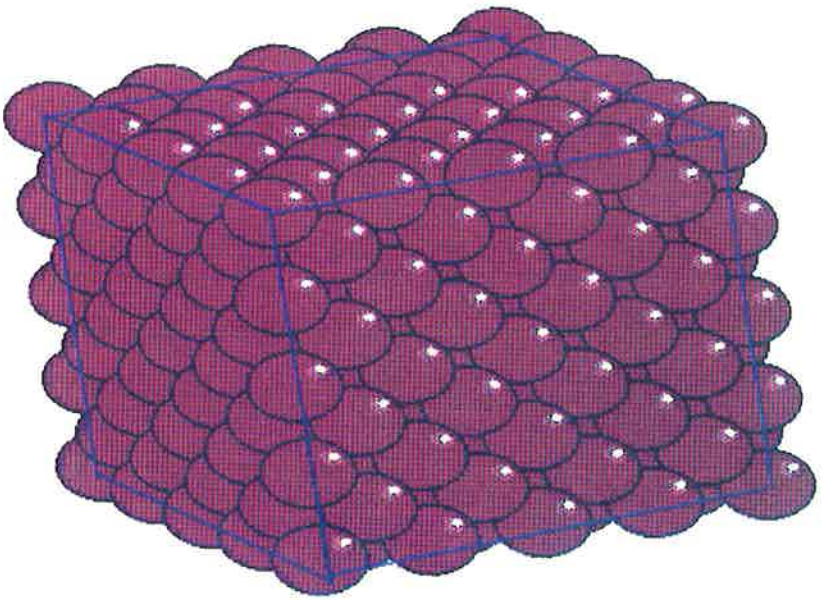
discussed. In this case the zeolite is extremely loaded and no major changes in the number of atoms in the unit cell are possible. It should be remembered at this point how the analytical expressions for the three-body terms have been obtained. They arise from a third- and fourth-order perturbation treatment of the long-range part of the potential and, strictly speaking, they are only valid in the limit of large separations. The perturbation theory used in the calculation of these non-additive terms is probably faulty when the pair interactions are near to the equilibrium distance since charge-overlap effects start becoming significant. Unfortunately, quantum mechanical calculations do not lead to convenient analytical results and it is common practice to employ expressions such as the Axilrod-Teller-Muto for all ranges of pair interactions [99]. The justification given to this is based upon a relative success in simple systems [100]. The subject still remains obscure today and has caused a fair amount of confusion and controversy in the literature. Meath et al. had already examined the validity of the triple-dipole term interaction at short intermolecular distances in the 1970s [101]. The subject has been reviewed more recently as quantum-mechanical calculations indicated that there were three-body exchange forces which were comparable in magnitude with the Axilrod-Teller-Muto interaction but of opposite sign (see ref. 102 but also see 103). P. Loubeyre [104] has shown the importance of three-body exchange effects in dense rare-gas solids. He employed an analytical expression for the repulsive part of the potential similar to the Born-Mayer function with the constants fitted to the best available exchange calculations. This is the first attempt to account in a more systematic form the problem of three-body exchange forces.

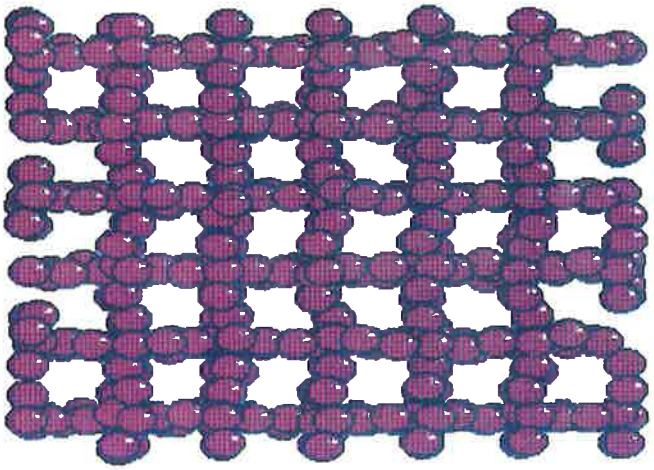
Including charge-overlap effects in our calculation in the form of a repulsive potential as the one discussed above is beyond the scope of this project but could be considered in further refinements of the potential functions available at the present time. One of the reasons for rejecting the inclusion of three-body effects stems from a snapshot analysis of configurations beyond the twenty-three argon per unit cell coverage. In these configurations there was an unrealistic overlap of argon atoms leading to unexpectedly large attractive and repulsive three-body energies which is probably incorrect. Thus this study also points out, although indirectly, that with the given crystallographic structure of silicalite-1 and the given model for the zeolite-argon interaction, the maximum loading occurs at roughly twenty-three argon atoms.

The deviations observed in different areas of the unit cell are the same as those found for the whole unit cell. No stark differences were found for either the intersections or the channels yet one would expect initially different trends for particles located in, for example, the straight channels (linear or almost linear configuration) and the intersections (aggregate-like structure). This homogeneous trend for the overall unit cell can be understood more easily if we take a less local picture of the zeolite cavity. Even though adsorption of argon in different regions of the unit cell leads to different configurations, these are not the only ones that contribute to the overall three-body energy.

The really interesting part of the plot of the deviations occurs at intermediate coverages, between ten and twenty-three argons per unit cell. The deviation of the true energy is between twenty and fifteen percent, showing a mild decreasing trend with coverage. This decrease could be attributed to the formation of a solid-like argon cluster at high coverages which resembles more the solid phase of argon thus having a smaller deviation. Nevertheless, the departure of the true from the effective is still very significant, highlighting the substantial structural differences between an argon lattice and the aggregate formed inside the zeolitic

cavity (figs. 13,14).





Figs. 13, 14 Comparison of the close-packed, centred-cubic argon lattice (top) with the condensed argon phase in the zeolite (bottom).

4.3.4 Relative importance of ArArAr, ArArO and ArArSi three-body interactions.

Figure 15 shows the percent composition of the total three-body energy as a function of coverage. ArArO contributes at least eighty percent of the total three-body energy at all coverages. This cannot be attributed to a stronger three-body ArArO since the electronic functions for ArArAr and ArArO are very similar yet ArArAr only makes up a maximum of ten percent in the high coverage limit. The predominance of ArArO is caused by the large number of oxygen atoms present in the zeolite compared to the number of argon atoms.

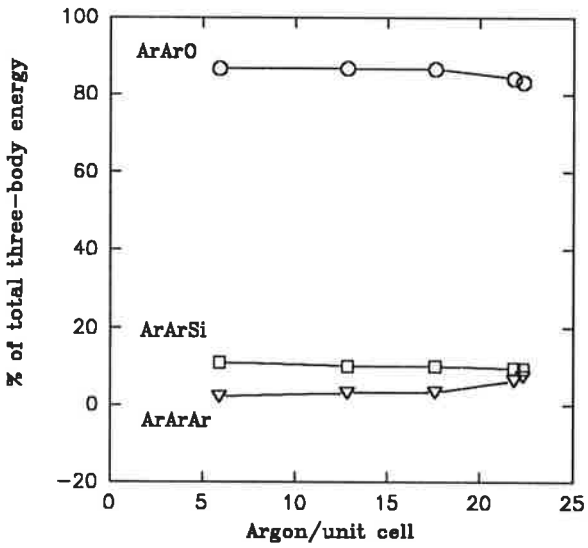
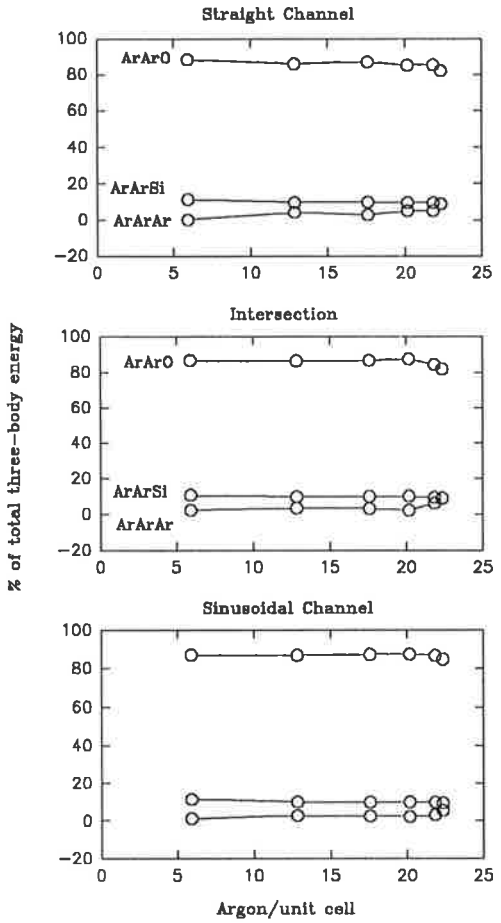


Fig. 15 Relative contribution of ArArAr, ArArO and ArArSi to the total three-body energy.

From the plot it is also clear that there is a weak interplay between ArArAr and ArArO and a fairly constant contribution of ArArSi to the total three-body energy. ArArAr increases with coverage and this rise is off-set by a decrease by roughly an equal amount of the ArArO contribution to the total three-body energy.

As observed before, different regions within the unit cell showed a markedly similar behaviour. The three figures provided below certify this statement.



Figs. 16,17,18 Relative contributions of ArArAr, ArArO and ArArSi to the total three-body energy in different regions of the unit cell.

4.3.5 Relative importance of dipole-dipole-dipole (DDD) and dipole-dipole-quadrupole (DDQ) terms.

In all cases, only the DDD and DDQ terms make up at least ninety-five percent of the total three-body energy, the DDD term being generally eighty percent. Other terms make a negligible contribution to the overall energy. The third-order ones are always positive while the fourth-order triple dipole is always minute and negative. This is an interesting finding not only from a computational viewpoint (it speeds up the calculation), but also from a physical one. These results are not comparable at all with those found for the solid and liquid argon. Firstly, it seems necessary to include both the DDD and DDQ third-order terms to account for almost all the three-body energy in contrast to only the DDD for liquid and solid argon. Secondly, the fourth-order triple-dipole is very small and no cancellation occurs between this term and the third-order ones beyond the DDD.

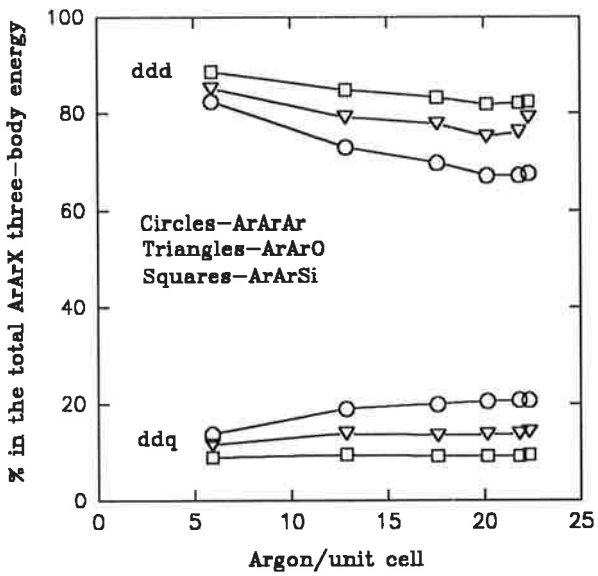


Fig. 19 Relative importance of DDD and DDQ terms in the ArArX three-body energy (X=Ar,O, Si).

It is interesting to note as well the interplay between DDD and DDQ as a function of coverage, DDD decreasing and DDQ increasing. This also denotes the particular nature of the aggregate of argon atoms formed in the cavities at high coverages, markedly different from a conventional solid argon lattice.

Chapter 5. Incorporating three-body effects into a full-scale molecular simulation.

The previous section has provided a better understanding of three-body forces in zeolites. However, the calculations necessary to reach those results were quite lengthy and were also limited to the computation of adsorbate energies for given 'snapshot' configurations. Even though from this information it was possible to infer many aspects of the adsorption of argon in the zeolite such as the impossibility of packing more than twenty-three molecules in a unit cell or the marked different character of the zeolite argon with respect the solid phase of argon, it would be very desirable to incorporate into a molecular simulation the effects of three-body forces. This would allow the determination of thermodynamic quantities such as the isosteric heat of adsorption which require the averaging over millions of configurations.

Unfortunately, it is not viable to use the rigorous approach taken so far and we need to resort again to the concept of an effective potential that mimics the effects observed for a very reduced number of configurations. The first part of this section will outline the procedure which would allow the inclusion of three-body effects in a simulation via a coverage-dependent effective Lennard-Jones potential. The second part will show how this procedure can be applied to a concrete problem, specifically that of refining the theoretical isosteric heat curve.

5.1 Generating a coverage-dependent effective Lennard-Jones potential.

The last chapter showed that the deviation of the so-called 'true' from the effective was a function of coverage. This means that three-body effects are also coverage-dependent, the deviation, or repulsiveness, of the true energy becoming less prominent with increasing argon uptake by the zeolite. One could make use of an iterative procedure in order to find the right parameters of the effective Lennard-Jones potential that could reproduce the deviation previously observed. Keeping things simple, we could keep the σ constant and let the ε vary until we found an effective Lennard-Jones potential for that given coverage. The effective potential would then have a coverage-dependent well-depth. Calculating the effective epsilon at different coverages would provide us with an empirical correlation between the epsilon and the number of particles in the unit cell. This would not add an extra degree of difficulty to the simulation since we have kept the same potential form the PN1 model is using presently to model the zeolite-argon potential..

The following procedure using a simulation is proposed:

(1) Run a canonical (NVT) simulation and obtain several snapshots. This time find for each snapshot the effective epsilon that would yield the true energy. Obtain an average epsilon from this set of calculations. This is our initial guess.

(2) Now, use this epsilon in a full-scale GCMC simulation. During the simulation the number of molecules will fluctuate but by a very small amount if we care to use configurations that have been previously equilibrated.

(3) With the configuration obtained in the simulation go back to the first step

and obtain a new value for the effective epsilon.

(4) Repeat steps (1)-(3) until the value of epsilon has converged to a given steady value.

As we pointed out in the previous chapter, this procedure is liable to fail at the very low coverage range due to very large fluctuations in the number of particles during the simulation. However, one would expect to find a quite steady effective epsilon for medium coverages. Appendix IV gives a detailed account of the format of the calculation

5.2 Refining the isosteric heat curve.

There is an immediate application of the procedure outlined above and that is the refining of the theoretical heat curve for the adsorption of argon in silicalite-1. As chapter 2 showed there is disagreement between theory and experiment on the shape of this curve. Experiment shows a constant heat of sorption up to the substep in the isotherm while simulation predicts a steady rise with coverage.

The isosteric heat can be split into adsorbate-adsorbent and adsorbate-adsorbate contributions. We will assume that the adsorbate-adsorbent contribution to the isosteric heat predicted by the PN1 model is correct. Then, since three-body forces are repulsive and non-negligible at all coverages, it might be necessary to include them in the calculation of isosteric heats, a fluctuation quantity that requires a complete molecular simulation. Hopefully, three-body forces will yield, at least at intermediate coverages, a lower adsorbate-adsorbate isosteric heat and thus a more satisfactory agreement with experimental data.

Testing this hypothesis is a good case study to implement the procedure described before in section 5.1. The emphasis will be put on the low and intermediate coverage regions, where the disagreement between simulation and experiment is largest. The effective epsilon for four different coverages are given in the table below:

Table 6. Effective epsilon at different coverages.

Number of argon atoms in unit cell.	Effective epsilon (K)
0.9	64.9±13.9
5.9	71.0±29.9
9.3	86.3±25.3
16.3	98.1±2.6

As expected the epsilon shows an increasing trend with increasing coverage and the deviations also show an overall decrease with coverage, the same basic traits found in our

study of the three-body forces described in the previous chapter.

The convergence to a steady epsilon is likely to be very slow for low coverages and it is even possible that it will never reach a constant value. Probably, the use of a canonical simulation (NVT) in which the number of molecules does not fluctuate would be a better procedure for the low coverage region, the only problem being that the isosteric heat becomes more difficult to evaluate.

As a preliminary and encouraging result in the calculation of isosteric heats, we have calculated the isosteric heat for the coverage with an epsilon of 71.02 K (see table 6) and found that the adsorbate-adsorbate contribution to the isosteric heat decreases significantly. For a simulation using an effective 120 K the adsorbate contribution to the isosteric heat is around 0.6 kJ/mol for a coverage of six argons per unit cell. We have found that for the corrected epsilon, the argon contribution to the isosteric heat goes down to 0.33 kJ/mol, almost half of the initially-predicted value.

However, these results are no more than preliminary and a full assessment of the procedure used throughout this section could only be guaranteed if:

-a canonical ensemble is used in the simulation, that is, with a constant number of adsorbate particles so that the effective epsilon calculated corresponds to a unique coverage.

-an effective epsilon is found for coverages ranging from zero to twenty-three argon atoms per unit cell and the performance of this coverage-dependent epsilon is tested using the experimental isosteric heat curve.

Concluding remarks and future aims.

From our analysis of the two- and three-body potential functions for argon we can conclude:

-the Barker-Fisher-Watts empirical potential still remains as the most reliable potential function for argon.

-the semiempirical potential of Tang-Toennies still needs further revision. It gives quite poor results if the best dispersion coefficients known today are used, probably due to the lack of reliable data on the repulsive part of the potential.

The study of non-additive interactions in the adsorption of argon in silicalite-1 has shown that:

-in contrast with previous work in zeolite 4A the role of three-body adsorbate interactions is not negligible, contributing up to twenty percent of the total energy.

-three-body effects are not local, that is, different regions within the unit cell exhibit similar trends in regard to three-body, non-additive terms. The geometric and topological peculiarities of each region that one would expect to lead to markedly different three-body effects are washed out by the proximity of adjacent channels and intersections within the zeolite.

-the breakdown of the validity of the long-range, three-body potentials utilized in this work at coverages over twenty-three argon atoms per unit cell added to the fact that significant overlap occurs at these high coverages suggest the impossibility of packing more than the aforementioned coverage in the silicalite-1 unit cell. This implies that given the zeolitic structure used in the simulations it is not possible to see the experimental substep. One is forced to postulate an adsorbate-induced structural phase transition, a phenomenon common in zeolites. This hypothesis has been partially corroborated by recent neutron scattering experiments although it is still premature to reach final conclusions.

-most of the three-body energy comes from ArArO interactions, its importance decreasing slightly as more argon atoms are put in the cavity. In contrast, ArArAr and ArArSi interactions are only ten percent each of the total three-body energy. There is a weak coupling of ArArO and ArArAr contributions to the total three-body energy, the latter increasing at the expense of the former as more argon atoms enter the zeolite cavity.

-significant deviations of the true potentials (Barker-Fisher-Watts plus three-body terms) from the effective Lennard-Jones are present even at the highest coverages when the adsorbate phase reaches its maximum density. This solid-like argon phase exhibits very peculiar properties certainly induced by the presence of the zeolite cavity. As opposed to the argon solid phase, three-body effects are dominated by third-order, triple-dipole (80%) and dipole-dipole-quadrupole (15%) terms. Thus any incorporation of three-body terms into a calculation in this system should at least

include these two terms while for the argon solid lattice only a third-order triple dipole term is sufficient.

-preliminary attempts to refine the isosteric heat curve are encouraging. The total isosteric heat as predicted before by simulation was a monotonic increasing function of coverage. The inclusion of adsorbate three-body interactions has the effect of decreasing the contribution of the adsorbate phase to the isosteric heat by a significant amount and thus brings simulation closer to experiment.

There are some immediate and obvious extensions of this work. Amongst them we could emphasize:

-the study of three-body effects in the sorption of other spherical or pseudo-spherical atoms and molecules for which experimental data is already available. This would include krypton, xenon and probably methane although in this case it would be more appropriate to analyze the atom-atom interactions (i.e. hydrogen-hydrogen, carbon-carbon, etc) instead of treating methane as a unique entity. The programs used for argon can be readily used for the other noble gas atoms with no modification of the code. Only the dispersion and three-body coefficients need to be modified.

-the modification of the three-body functions to include electron exchange effects. This is obviously a great theoretical endeavour and it would probably require a longer period of time until sufficient knowledge is acquired in more simple systems.

-the development of an accurate effective potential that mimics three-body effects in the zeolite and its implementation on other systems such as those proposed above (noble gases and simple pseudo-spherical molecules).

Acknowledgements.

My most sincere gratitude to Dr. Nicholson and R. J-M. Pellenq for their invaluable patience and support in the course of the project. Nothing achieved in this work would have been possible without their continuous encouragement and scientific insight.

I would also like to thank Y. C. Kong for sharing with me his computer expertise throughout this work.

Appendices I-IV.

See volume II.

References.

1.A. Cronsted, *Akad. Handl. Stockholm*, **17** , 1756: 1201
2. R. M. Barrer, *J. Chem. Soc.*, 1948: 2158
3. J. N. Maile, N. Y. Chen and P. B. Weisz, *J. Catalysis*, **6**, 1966: 278
4. A. P. Bolton, 'Zeolite Chemistry and Catalysis', J. A. Rabo Ed., A.C.S, Washington (1968)
5. C. R. A. Catlow, 'Modelling of Structure and Reactivity in Zeolites', Academic Press, London, 1992
6. R. G. Gordon and Y.S Kim, *J. Chem. Phys.*, **56**, 1976: 3122
7. R. G. Bell, R. A. Jackson and C. R. A. Catlow, *J. Chem. Soc. Chem. Comm.*, 1990: 782
8. M J. Sanders, C. R. A. Catlow and J. V. Smith, *J. Phys. Chem.*, **88**, 1984: 2796
9.C. J. J. den Ouden, R. A. Jackson, C. R. A. Catlow and M. F. M. Post, *J. Phys. Chem.*, **94**, 1990: 5286
10. S. Yashonath, J. M. Thomas, A. K. Nowak and A. K. Cheethan, *Nature*, **331**, 1988: 601
11. B. Smit and C. J.J. den Ouden, *J. Phys. Chem.*, **92**, 1988: 7169
12. G. B. Woods and J. S. Rowlinson, *J. Chem. Soc. Faraday Trans. 2*, **85**, 1989: 765
13. J. M. Skin, K. T. No and M.S. Jhon, *J. Phys. Chem.*, **92**, 1988: 4533
14. P. Demontis, S. Yashonath and M. L. Klein, *J. Phys. Chem.*, **93**, 1989: 5016
15. S. D. Pickett et al., *J. Phys. Chem.*, **94**, 1990: 1233
16. E. E. Dimukhambetov, V. I. Lygin and E. G. Chadiarov, *Zh. Fiz. Khim.*, **62**, 1988:226
17. N. U. Zhanpeisov, et al., *Kinet. Katal.*, **28**, 1987: 194
18. For an introduction to the field of intermolecular forces see: M. Rigby, E. B. Smith, W. A. Wakeham and G. C. Maitland, 'The Forces between Molecules', Oxford Science Publications, Clarendon Press, Oxford, 1986.
19. For a deeper coverage of the subject see: H. Margenau and N. R. Kestner, 'Theory of Intermolecular Forces', Int. Series of Monographs in Natural Philosophy, Vol. 18, Pergamon Press, Oxford, 1969; J. O. Hirshfelder, C. F. Curtiss adn R. B. Bird, 'Molecular Theory of Gases, Liquids and Solids', John Wiley & Sons, inc., New York, 1964; B. Pullmann, 'Intermolecular Forces: from Diatomics to Biopolymers', John Wiley & Sons, New York, 1978; G. C. Maitland, 'Intermolecular Forces', Oxford Science Publications, Clarendon Press, Oxford, 1981.
20. For recent reviews in the literature see: *Chem. Rev.*, **88**, 1988; A. J. Stone, S. L. Price, *J. Phys. Chem.*, **92**, 1988: 3325
21. R. Eisenschitz and F. London, *Z. Physik*, **60**, 1930: 491
22. B. M. Axilrod, *J. Chem. Phys.*, **11**, 1943: 299
23. Y. Muto, *Proc. Phys. Math. Soc. Japan*, **17**, 1943: 629
24. J. P. Toennies, *Chem. Phys. Lett.*, **1**, 1967: 325
25. J. A. Barker, R. A. Fisher and R. O. Watts, *Mol. Phys.*, **21**, 1971: 657
26. Barrer, 'Zeolites and Clay Minerals as Sorbents and Molecular Sieves', Academic Press, London, 1978
27. Y. S. Kim, R. G. Gordon, *J. Chem. Phys.*, **56**, 1972: 3172
28. D. Nicholson and N. Parsonage, 'Statistical Mechanics and the Simulation of Adsorption', Academic Press, 1982.
29. R. W. Grose and E. M. Flanigen, U.S. Patent, No. 4,061,724 (1977)
30. Argauer and Landolt, U.S. Patent, No. 3,832,8806 (1972)
31. W. M. Meier and D. H. Olson, 'Atlas of Zeolite Structure Types', 3rd ed., Butherworth-Heinemann, London, 1992.

32. H. Lermer, M. Drager, J. Steffen, K. K. Unger, *Zeolites*, **6**, 1986: 10
33. C. A. Fyfe, G. C. Gobbi, J. Klinowski, J. M. Thomas, S. Ramdas, *Nature*, **286**, 1982: 530
34. W. M. Meier, 'Molecular Sieves', Soc. Chem. Ind., London, 1968: 10
35. H. Van Koningsveld, J. C. Hansen, H. Van Bekkum, *Zeolites*, **7**, 1987: 73
36. R. M. Pellenq, Ph.D. Thesis, Imperial College of Science, Technology and Medicine, London, 1992
37. P. L. LLewellyn, 'The MFI-type Zeolites: An Examination of the Phenomena Observed on Adsorption of Simple Probe Molecules, Ph.D Thesis, Brunel University, 1992
38. H. Van Koningsveld, H. Van Bekkum and J. C. Hansen, *Acta Cryst.*, **B46**, 1987: 127
39. H. Van Koningsveld, J. C. Hansen, H. Van Bekkum, *Zeolites*, **10**, 1990: 235
40. H. Van Koningsveld, F. Tuinstra, H. Van Bekkum, J. C. Hansen, *Acta Cryst.*, **B45**, 1989: 423
41. C. A. Fyfe, G. J. Kennedy, C. T. De Schutter, G. T. Kokotailo, *J. Chem. Soc., Chem. Comm.*, 1984: 541
42. J. A. Muller, W. C. Conner, *J. Phys. Chem.*, **97**, 1993: 1451
43. L. Pauling, 'The Nature of the Chemical Bond', 3rd ed., Cornell University Press, Ithaca, 1960
44. K. A. Van Genechten and W. Mortier, *Zeolites*, **8**, 1988: 273
45. R. Nada, C.R.A Catlow, R. Dovesi, C. Pisani, *J. Phys. Chem.*, **17**, 1990: 353
46. J. P. Coulomb and Y. Grillet, et al., *Langmuir*, in press
47. U. Muller, K. K. Unger, D. Pan, A. Mersmann, Y. Grillet, F. Rouquerol, J. Rouquerol, 'Zeolites as Catalysts, Sorbents and Detergent Builders', H.G. Karge and J. Weitkamp eds., Elsevier, Amsterdam, 1989: 625
48. U. Muller, H. Reichert, E. Robens, K.K. Unger, Y. Grillet, F. Rouquerol, J. Rouquerol, D. Pan and A. Mersmann, *Frezenius Z. Anal. Chem.*, **333**, 1989: 433
49. I. Derycke, J. P. Vigneron, P. Lambin, A. Lucas, E. G. Derouanne, *J. Chem. Phys.* **94**, 1991: 4620
50. A. V. Kiselev, A. A. Lopatkin and A. A. Shulga, *Zeolites*, **5**, 1985: 261
51. R. J-M. Pellenq and D. Nicholson, 4th International Conference on Fundamentals of Adsorption', Kyoto, 1992
52. R. J-M. Pellenq and D. Nicholson, 'Potential function for rare gases interacting with Silicalite-1 zeolite', *J. Phys. Chem.*, in preparation.
53. R. J-M. Pellenq and D. Nicholson,'In-framework ion dipole polarizabilities in porous and non-porous silicates and aluminosilicates determined from Auger Electron Spectroscopy Data', *J. Chem. Soc. Faraday Trans.*, in press
54. R. J-M. Pellenq, personal communication, 1993
55. Pellenq, R. J-M., LLewellyn, P.L., Pellenq, N. J.M, Coulomb, J-P. and Nicholson, D., 'Phase transitions in silicalites containing physically adsorbed molecules', in preparation
56. G. C. Maitland, M. Rigby, E. B. Smith, W. A. Wakeham, 'Intermolecular Forces', Clarendon Press, Oxford, 1981
57. A. T. J. Hope, C. A. Leng and C. R. A. Catlow, *Proc. R. Soc. Lond.*, **A 424**, 1989: 57
58. R. J-M. Pellenq, A. Pellegatti, C. Minot, D. Nicholson, *J. Phys. Chem.*, in preparation
59. H. Y. Kim, M. W. Cole, *Phys. Rev.*, **B35**, 1987: 3990
60. D. Nicholson, *Surf. Sci.*, **184**, 1987: 255
61. D. Nicholson, 'Fundamentals of Adsorption', Mersmann A. B. and Scholl S. E. eds., United Engineering Trustee Incorporation, 1990
62. G. C. Maitland et al., 'Intermolecular Forces', Oxford Science Publications, Clarendon Press, Oxford, 1981: 497

63. J. A. Barker, A. Pompe, *Aust. J. Chem.*, **21**, 1968: 1683
64. J. A. Barker, R. O. Watts, J.K. Lee, T. P. Schafer and Y. T. Lee, *J. Chem. Phys.*, **61**, 1974: 3081
65. K. T. Tang and J. P. Toennies, *J. Chem. Phys.*, **80**, 1984: 3726
66. R. Feltgen, *J. Chem. Phys.*, **74**, 1981: 1186
67. R. Feltgen, H, Kirst, K. A. Kohler, H. Pauli, F. Torello, *J. Chem. Phys.*, **76**, 1982: 2360
68. K. T. Tang and J. P. Toennies, *J. Chem. Phys.*, **66**, 1977: 1496
69. P. W. Fowler, J. M. Hutson, A. D. Buckingham, *Chem. Rev.*, **88**, 1988: 963
70. R. A. Aziz and M. J. Slaman, *J. Chem. Phys.*, **92**, 1990: 1030
71. P. W. Fowler, *Mol. Sim.*, **4**, 1990: 313
72. K. T. Tang, *Phys. Rev.*, **177**, 1969: 108
73. K. T. Tang, J. M. Norbeck, P. R. Certain, *J. Chem. Phys.*, **64**, 1976: 3063
74. A. Dalargno, *Adv. Chem. Phys.*, **12**, 1967: 143
75. P. W. Fowler and N. C. Pyper, *Mol. Phys.*, **59**, 1990: 317
76. J. M. Standard, P. R. Certain, *J. Chem. Phys.*, **83**, 1985: 3002
77. R. J. Bell, *J. Phys. B: Atom, Mol. Phys.*, **3**, 1970: 751
78. W. L. Bade, *J. Chem. Phys.*, **27**, 1957: 1280
79. M. B. Doran and I. J. Zucker, *J. Phys. C: Solid St. Phys.*, **4**, 1971: 307
80. T. B. MacRaury, B. Linder, *J. Chem. Phys.*, **54**, 1970: 2056
81. T. M. Miller, CRC Handbook of Physics and Chemistry, 1991
82. J. E. Lennard-Jones, *Proc. Roy. Soc.*, **A106**, 1924: 463
83. T. Kihara, M. Kotani, *M. Proc. Phys. Math. Soc. Jpn.*, **24**, 1942: 76
84. J. O. Hirshfelder, R. B. Bird, E. L. Spotz, *J. Chem. Phys.*, **16**, 1948: 968
85. K. E. Grew, *J. Chem. Phys.*, **18**, 1950: 149
86. A. Michels, H. Wijker, H. K. Wijker, *Physica*, **15**, 1949: 1949
87. E. A. Guggenheim, M. L. McGlasham, *Proc. Roy. Soc.*, **A255**, 1960: 446
88. M. Klein, H.J.M. Hanley, *J. Chem. Phys.*, **53**, 1977: 4722
89. R. A. Aziz and H. H. Chen, *J. Chem. Phys.*, **67**, 1977: 5719
90. J. A. Barker, D. Henderson, *Rev. Mod. Phys.*, **48**, 1976: 587
91. J. O. Hirshfelder (Ed.), 'Intermolecular Forces', *Adv. Chem. Phys.*, **12**, 1967
92. W. R. Wadt, *J. Chem. Phys.*, **68**, 1978: 402
93. R. J-M. Pellenq, personal communication, 1993
94. C. Kittel, 'Physique de l'Etat Solide', 5th ed., Dunod Université, 1983: 76
95. M. V. Bobetic, J. A. Barker, *Phys. Rev.* **B2**, 1970: 4169
96. J. A. Barker, M. L. Klein, *Phys. Rev.* **B7**, 1973: 4707
97. J. A. Barker, 'Rare Gas Solids', edited by M. L. Klein and J. A. Venables, Academic Press, New York, 1976, vol. 1, chapter 4
98. R. J-M. Pellenq, personal communication, 1993
99. D. R. Williams, L. J. Schaad, J. N. Murrell, *J. Chem. Phys.*, **47**, 1967: 4916
100. A. E. Sherwood, A. G. DeRocco, E. A. Mason, *J. Chem. Phys.*, **44**, 1966: 2984
101. S. F. O'Shea, W. J. Meath, *Mol. Phys.*, **31**, 1976: 515
102. W. J. Meath, R. A. Aziz, *Mol. Phys.*, **52**, 1984: 225
103. J. A. Barker, *Phys. Rev. Lett.*, **57**, 1986: 230
104. P. Loubeyre, *Phys. Rev. B*, **37**, 1988: 5432
105. H. Kono, A. Takasaka, *J. Phys. Chem.*, **91**, 1987: 4044
106. P. W. Fowler, N. C. Pyper, *Mol. Phys.*, **59**, 1986: 317
107. A. J. C. Varandas, J. D. da Silva, *J. Chem. Soc. Faraday Trans. 2*, **82**, 1986: 593

The Role of Three-Body Interactions in the Adsorption of Argon in Silicalite-1

Volume II: Appendices I-IV

Third Year Physical Chemistry Research Report

by
Félix Fernández-Alonso

under the supervision of Dr. David Nicholson and Mr. Roland J-M Pellenq

Imperial College of Science Technology and Medicine
University of London
June 1993

Appendix I

Calculation on a close-packed, face-centred cubic argon lattice

Programs:

(1) **arlat.f** : generates a close-packed, centred-cubic argon lattice.

-input: file 'arlat.par' with the coordinates of the four atoms needed to define the lattice.
-output: file 'arpos.dat' with the coordinates of the lattice in reduced units

(2) **ar2b3b.f**: calculates the two-body and three-body energies.

-input: file 'arpos.dat'
-output: file 'results.tem', which includes the two- and three-body energies and the deviations from the effective Lennard-Jones potential.

Calculation:

arlat.f generates several lattices with sizes ranging from a few to several thousand atoms. The lattice energies are computed for each lattice with ar2b3b.f. The potential functions used were:

- effective Lennard-Jones potential

- two-body potentials:
    - Lennard-Jones
    - Barker-Fisher-Watts
    - Tang-Toennies

- three-body potentials.

The deviations of the energies obtained between the two-body plus three-body and the effective potential were plotted for the three distinct two-body potentials as a function of lattice size.

```
      PROGRAM arlat

c     This program generates the coordinates of a close-packed
c     cubic-centred lattice.
c
c     Note: coordinates are given in reduced units.
c
c     Input files:
c          -'arlat.par': all parameters needed in calculation
c     Output files:
c          -'arpos.dat': argon (x,y,z) coordinates in reduced
c          coordinates
c
c
c     Felix Fernandez Alonso.
c     Imperial College.
c     February 1993
c
c     ***************************************************************
c
c     main program
c
      common/argpos/xar(10000),yar(10000),zar(10000)
      common/params/ntemplates,nar,nstps

      print*,'... generating argon lattice'
      call readparams
      call makelattice
      call writelattice
      print*,'...argon lattice succesfully constructed'
      print*,''
      end
c
c     ***************************************************************
c
      subroutine readparams

      common/argpos/xar(10000),yar(10000),zar(10000)
      common/params/ntemplates,nar,nstps

      open(unit=1,file='arlat.par')

c     'ntemplates' is the number of atoms necessary to build
c     the lattice.
c     'nar' denotes the number of template molecules (four).
c     'nstps'is the variable that determines the number of
c     atoms in the lattice.

      read(1,*) ntemplates
      read(1,*) nar
      read(1,*) nstps
c
c     read the coordinates of the template atoms.
c
      read(1,*) xar(1)
      read(1,*) yar(1)
      read(1,*) zar(1)
```

```
      read(1,*) xar(2)
      read(1,*) yar(2)
      read(1,*) zar(2)

      read(1,*) xar(3)
      read(1,*) yar(3)
      read(1,*) zar(3)

      read(1,*) xar(4)
      read(1,*) yar(4)
      read(1,*) zar(4)

      print*,' '
      print*,'  parameters read successfully'

      return
      end
c
c     ***************************************************************
c
      subroutine makelattice

      common/argpos/xar(10000),yar(10000),zar(10000)
      common/params/ntemplates,nar,nstps
c
c     Loops that replicate in three-dimensions the four template
c     atoms.
c
      do 5 i=1,ntemplates

      if (i.eq.1) then
      arcubes=nstps+1
      else
      arcubes=nstps
      endif

      xtemplate=xar(i)
      ytemplate=yar(i)
      ztemplate=zar(i)

      do 10 j=1,arcubes
      do 20 k=1,arcubes
      do 30 l=1,arcubes

c     the following three loops avoid repeating the coordinates
c     of the template atoms

      if ((j.eq.1).and.(k.eq.1).and.(l.eq.1)) then
      goto 30
      else

      nar=nar+1

      xar(nar)=(xtemplate-1)+j
      yar(nar)=(ytemplate-1)+k
      zar(nar)=(ztemplate-1)+l

      if ((i.eq.2).and.(j.eq.arcubes)) then
```

```
      nar=nar+1
      xar(nar)=(xtemplate-1)+j+1
      yar(nar)=(ytemplate-1)+k
      zar(nar)=(ztemplate-1)+l
      endif

      if ((i.eq.3).and.(k.eq.arcubes)) then
      nar=nar+1
      xar(nar)=(xtemplate-1)+j
      yar(nar)=(ytemplate-1)+k+1
      zar(nar)=(ztemplate-1)+l
      endif

      if ((i.eq.4).and.(l.eq.arcubes)) then
      nar=nar+1
      xar(nar)=(xtemplate-1)+j
      yar(nar)=(ytemplate-1)+k
      zar(nar)=(ztemplate-1)+l+1
      endif

      endif

30    continue
20    continue
10    continue

5     continue

      print*, ' lattice created'
      print*, ' total of ',nar,' atoms'
      return
      end
c
c     ****************************************************************
c
      subroutine writelattice

      common/argpos/xar(10000),yar(10000),zar(10000)
      common/params/ntemplates,nar,nstps

      open(unit=2,file='arpos.dat')
c
c     Write the coordinates of the atoms in file 'arpos.dat'
c
      do 10 i=1,nar
      write(2,*)xar(i),yar(i),zar(i)
10    continue
      print*,'  parameters written to arpos.dat'
      print*,''
      return
      end
```

```
            PROGRAM  ar2b3b

c       Two-body and Three-body Potential Energy Calculation
c
c       This program calculates two-body and three-body terms
c        of the dispersion potential for a given configuration:
c
c               - 2b arar dispersion potential (Barker-Fisher-Watts
c                 Potential,Tang and Toennies, true 2b Lennard-Jones
c                 and effective Lennard-Jones)
c               - 3b ararar.
c
c       It also gives the % deviation from the effective Lennard-Jones
c       potential.
c
c       This program can be used for any spherical molecule or atom.
c       It reads:
c               - 'parama.dat', with all the required parameters
c               - 'arpos.dat', the reduced coordinates of the
c                 configuration.
c
c       Results are displayed in the file 'results.tem'
c
c
c       Felix Fernandez-Alonso, Imperial College, February, March, April, 1993
c
c*******************************************************************
c
      common /param6e/ c6arar,c8arar,c10arar
      common /param6ebfw/ c6bfw,c8bfw,c10bfw
      common /bfwparams/ a0arar,a1arar,a2arar,a3arar,a4arar,a5arar,
     +                   aarar,deltarar,epsilbfw,radbfw
      common /LJones/epsilar,sigmar
      common /LJones2b/epsilar2b,sigmar2b
      common /param3bararar/addd,addq,adqd,aqdd,adqq,aqdq,aqqd,
     +                   aqqq,ad41,ad42,ad43
      common /cutoff/ hcut
      common /repulsy/ areparar,breparar
      common /arpos/ nar, xar(10000),yar(10000),zar(10000)
      common /values/ auc,buc,cuc,rbohr,energy
c
c     Main Program.
c
      print*,''
      print*,'...calculation starts.'
      call parameters
      call geometry
      call set3bararar
      call potcalcul
      print*, '... calculation ends.'
      print*,' '
      end
c
c*********************************************************************
c
      subroutine parameters

      common /param6e/ c6arar,c8arar,c10arar
```

```
      common /param6ebfw/ c6bfw,c8bfw,c10bfw
      common /bfwparams/ a0arar,a1arar,a2arar,a3arar,a4arar,a5arar,
     +                   aarar,deltarar,epsilbfw,radbfw
      common /LJones/epsilar,sigmar
      common /LJones2b/epsilar2b,sigmar2b
      common /param3barararar/ addd, addq,adqd,aqdd,adqq,aqdq,aqqd,
     +                   aqqq,ad41,ad42,ad43
      common /cutoff/ hcut
      common /repulsy/ areparar,breparar


c
      open(unit=20,file='paramar.dat')
c
      read(20,*) hcut


c     BFW Two-body terms for ArAr.

      read(20,*) c6bfw
      read(20,*) c8bfw
      read(20,*) c10bfw

      read(20,*) a0arar
      read(20,*) a1arar
      read(20,*) a2arar
      read(20,*) a3arar
      read(20,*) a4arar
      read(20,*) a5arar
      read(20,*) aarar
      read(20,*) deltarar
      read(20,*) epsilbfw
      read(20,*) radbfw

c     Tang and Toennies parameters

      read(20,*) c6arar
      read(20,*) c8arar
      read(20,*) c10arar
      read(20,*) areparar
      read(20,*) breparar

c     Lennard-Jones parameters, epsilon (in kelvin) and sigma (A),
c     for argon

      read(20,*) epsilar2b
      read(20,*) sigmar2b
      read(20,*) epsilar
      read(20,*) sigmar


c     Three-body terms for ArArAr

      read(20,*) addd
      read(20,*) aqdd
      read(20,*) addq
      read(20,*) adqd
      read(20,*) aqqd
```

```
      read(20,*) aqdq
      read(20,*) adqq
      read(20,*) aqqq
      read(20,*) ad41
      read(20,*) ad42
      read(20,*) ad43


c     print*, 'all parameters in atomic units'
c     print*, 'exceptions: ar lennard-Jones parameters'

      return
      end

c
c*********************************************************************
c
      subroutine geometry
c
      common /arpos/ nar, xar(10000),yar(10000),zar(10000)
      common /values/ auc,buc,cuc,rbohr,energy
c

      open(unit=4,file='arpos.dat')
c
c     The van der Waals radius for argon is 1.88 A. For a close-packed
c     centered-cubic lattice the crystallographic parameter is
c     2*sqrt(2)*1.88 A
c

      auc=5.317443
      buc=5.317443
      cuc=5.317443
      rbohr=0.52917
c
c     read argon coordinates.
c
      nar=0
 55    nar=nar+1
      read(4,*,end=56) x3,y3,z3
      xar(nar)=x3
      yar(nar)=y3
      zar(nar)=z3
      goto 55
 56    nar=nar-1
c      print*,nar,' argon coordinates read in'
c     convert to bohr units
      do 57 j=1,nar
      xar(j)=auc*xar(j)/rbohr
      yar(j)=buc*yar(j)/rbohr
      zar(j)=cuc*zar(j)/rbohr
 57    continue
c      print*,'argon coordinates converted to Bohr units'
c
      return
      end
c
c*********************************************************************
```

```
c

c
      subroutine set3bararar

      common /param3bararar/ addd, addq,adqd,aqdd,adqq,aqdq,aqqd,
     +                 aqqq,ad41,ad42,ad43

c
c     convert electronic functions of three-body potentials to k
c
      factk=3.158e5
      addd=addd*factk
c
      aqdd=aqdd*factk
      addq=addq*factk
      adqd=adqd*factk
c
      aqqd=aqqd*factk
      aqdq=aqdq*factk
      adqq=adqq*factk
c
      aqqq=aqqq*factk

c
      addd41=ad41*factk
      addd42=ad42*factk
      addd43=ad43*factk
c
      return
      end
c
c*********************************************************************
c
      subroutine potcalcul
c


      common /param3bararar/ addd, addq,adqd,aqdd,adqq,aqdq,aqqd,
     +                 aqqq,ad41,ad42,ad43

      common /cutoff/ hcut

      common /repulsy/ areparar,breparar

      common /arpos/ nar,xar(10000),yar(10000),zar(10000)

      common /values/ auc,buc,cuc,rboh,energy

c
c     potential calculation
c
c     calls all the potential function subroutines.
c
c     initialise variables

      factk=3.158e5
      pot3ararar=0.0
```

```
      potarar=0.0
      potTT=0.0
      potLJarar=0.0
      potbfw=0.0

c
c     call two-body interaction subroutines
c
      call pot2barar(potarar)
      call pot2bTT(potTT)
      call potLJParar(potLJarar)
      call pot2bfw(potbfw)
      call convertok(potbfw)
      potTT=potTT*factk

c     call three-body interaction subroutines

      call pot3bararar(pot3ararar)

c     evaluate and display final results

      pot3ararar=pot3ararar/nar
      potarar=potarar/nar
      potTT=potTT/nar
      potLJarar=potLJarar/nar
      potbfw=potbfw/nar

c     determine deviations

      devLJ=100*(potarar+pot3ararar-potLJarar)/potLJarar
      devTT=100*(potTT+pot3ararar-potLJarar)/potLJarar
      devBFW=100*(potbfw+pot3ararar-potLJarar)/potLJarar

      open(unit=17,file='results.tem')

      print *,'NORMALISED RESULTS (in units of kelvin/argon atom)'
      print *,' '
      print *,' total number of argon atoms in latttice   ',nar
      print *,' effective Lennard-Jones energy            ',potLJarar
      print *,' two-body energies:'
      print *,'        true Lennard-Jones                ',potarar
      print *,'        Tang and Toennies                 ',potTT
      print *,'        Barker-Fisher-Watts               ',potbfw
      print *,' total 3-body energy ar-ar-ar             ',pot3ararar
      print *,' '
      print *,' deviations from effective Lennard-Jones:'
      print *,'        true Lennard-Jones                ',devLJ
      print *,'        Tang and Toennies                 ',devTT
      print *,'        Barker-Fisher-Watts               ',devBFW
      print *,' '


      write(17,*)nar
      write(17,*)potLJarar
      write(17,*)pot3ararar
      write(17,*)potarar,devLJ
      write(17,*)potTT,devTT
      write(17,*)potbfw,devBFW
```

```
      write(17,*)

      return
      end
c
c*********************************************************************
c
      subroutine pot2barar(sumarar)

      common /cutoff/ hcut
      common /arpos/ nar, xar(10000),yar(10000),zar(10000)
      common /values/ auc,buc,cuc,rbohr,energy
      common /LJones2b/epsilar2b,sigmar2b

c     Evaluate the two-body Lennard-Jones energy

      sumarar=0.0
      n2b=0

      rbohr=0.52917
      sigmar2bB=sigmar2b/rbohr


      do 10 n=1,nar-1

      x1=xar(n)
      y1=yar(n)
      z1=zar(n)

      do 20 j=n+1,nar

      xarx2=(xar(j)-x1)**2
      yary2=(yar(j)-y1)**2
      zarz2=(zar(j)-z1)**2
      xarx1=xar(j)-x1
      yary1=yar(j)-y1
      zarz1=zar(j)-z1
      r2arar=xarx2+yary2+zarz2
      r1arar=sqrt(r2arar)

      arar2b=0.0

      redr6=(r1arar/sigmar2bB)**6
      redr12=(r1arar/sigmar2bB)**12
      arar2b=4*epsilar2b*((1/redr12)-(1/redr6))
c
      sumarar=arar2b+sumarar


      n2b=n2b+1
20     continue
10     continue
      print*,'number of 2b arar  ',n2b
      return
      end

c*********************************************************************
```

```
      subroutine pot2bTT(sumTT)

      common /param6e/ c6arar,c8arar,c10arar
      common /repulsy/ areparar,breparar
      common /arpos/ nar, xar(10000),yar(10000),zar(10000)
      common /values/ auc,buc,cuc,rbohr,energy

c     Evaluate the pair energy with the Tang-Toennies potential

      sumTT=0.0

      do 45 n=1,nar-1

      x1=xar(n)
      y1=yar(n)
      z1=zar(n)

      do 55 j=n+1,nar

      xar2=(xar(j)-x1)**2
      yar2=(yar(j)-y1)**2
      zar2=(zar(j)-z1)**2
      xar1=xar(j)-x1
      yar1=yar(j)-y1
      zar1=zar(j)-z1
      r2arar=xar2+yar2+zar2
      r1arar=sqrt(r2arar)

      disp6arar=c6arar/(r2arar**3)
      disp8arar=c8arar/(r2arar**4)
      disp10arar=c10arar/(r2arar**5)
      exponarar=exp(-breparar*r1arar)
c
      it=1
      g6arar=1.0
      g8arar=0.0
      g10arar=0.0
c
      do 11 il=1,6
      g6arar=(g6arar+(breparar*r1arar)**il)/it
      it=(il+1)*it
11     continue
      g8arar=g6arar
      do 12 im=7,8
      g8arar=g8arar+((breparar*r1arar)**im)/it
      it=(im+1)*it
12     continue
      g10arar=g8arar
      do 13 in=9,10
      g10arar=g10arar+((breparar*r1arar)**in)/it
      it=(in+1)*it
13     continue
c
      f6arar=1-g6arar*exponarar
      f8arar=1-g8arar*exponarar
      f10arar=1-g10arar*exponarar
      att6arar=disp6arar*f6arar
      att8arar=disp8arar*f8arar
```

```
      att10arar=disp10arar*f10arar
      attarar=att6arar+att8arar+att10arar
      reparar=areparar*exponarar
c
      sumTT=-attarar+reparar+sumTT

55     continue
      print*,sumTT
45     continue

      return
      end
c
c*********************************************************************
c
      subroutine potLJParar(sumLJarar)


      common /cutoff/ hcut
      common /arpos/ nar, xar(10000),yar(10000),zar(10000)
      common /values/ auc,buc,cuc,rbohr,energy
      common /LJones/epsilar,sigmar
c
c     Evaluate the energy with the effective Lennard-Jones potential.
c
      rbohr=0.52917
      sigmarB=sigmar/rbohr
      sumLJarar=0.0
      nLJ=0


      do 60 n=1,nar-1

      x1=xar(n)
      y1=yar(n)
      z1=zar(n)

      do 70 j=n+1,nar

      xarx2=(xar(j)-x1)**2
      yary2=(yar(j)-y1)**2
      zarz2=(zar(j)-z1)**2

      xarx1=xar(j)-x1
      yary1=yar(j)-y1
      zarz1=zar(j)-z1

      r2arar=xarx2+yary2+zarz2
      r1arar=sqrt(r2arar)
      ararLJ=0.0

      redr6=(r1arar/sigmarB)**6
      redr12=(r1arar/sigmarB)**12
      ararLJ=4*epsilar*((1/redr12)-(1/redr6))
c
      sumLJarar=ararLJ+sumLJarar
      nLJ=nLJ+1
70     continue
```

```
      print*,sumLJarar
60     continue
      print*,'Lennard-Jones 2b interact.   ',nLJ
      return
      end

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

      subroutine pot2bfw(sumbfw)

      common /param6ebfw/ c6bfw,c8bfw,c10bfw
      common /bfwparams/ a0arar,a1arar,a2arar,a3arar,a4arar,a5arar,
     +                   aarar,deltarar,epsilbfw,radbfw
      common /arpos/ nar, xar(10000),yar(10000),zar(10000)
      common /values/ auc,buc,cuc,rbohr,energy

c
c     Evaluate the energy with the Barker-Fisher-Watts potential.
c
      sumbfw=0.0
      n2b=0

      rbohr=0.52917
      radbfwB=radbfw/rbohr


      do 10 n=1,nar-1

      x1=xar(n)
      y1=yar(n)
      z1=zar(n)

      do 20 j=n+1,nar

      xarx2=(xar(j)-x1)**2
      yary2=(yar(j)-y1)**2
      zarz2=(zar(j)-z1)**2
      xarx1=xar(j)-x1
      yary1=yar(j)-y1
      zarz1=zar(j)-z1
      r2arar=xarx2+yary2+zarz2
      r1arar=sqrt(r2arar)
      redr1=r1arar/radbfwB

      ararbfw=0.0

      expterm=exp(aarar*(1-redr1))

      a0term=a0arar
      a1term=a1arar*((redr1-1))
      a2term=a2arar*((redr1-1)**2)
      a3term=a3arar*((redr1-1)**3)
      a4term=a4arar*((redr1-1)**4)
      a5term=a5arar*((redr1-1)**5)

      asum=a0term+a1term+a2term+a3term+a4term+a5term

      term1=expterm*asum
```

```
      c6term=c6bfw/(deltarar+(redr1**6))
      c8term=c8bfw/(deltarar+(redr1**8))
      c10term=c10bfw/(deltarar+(redr1**10))

      term2=c6term+c8term+c10term

      ararbfw=term1+term2

      sumbfw=ararbfw+sumbfw

      n2b=n2b+1

20    continue
10    continue
      print*,'number of bfw 2b arar  ',n2b
      return
      end

c
c*********************************************************************
c
      subroutine pot3barararar(sumararar)

      common /param3barararar/addd,addq,adqd,aqdd,adqq,aqdq,aqqd,
     +                aqqq,ad41,ad42,ad43

      common /arpos/ nar, xar(10000),yar(10000),zar(10000)

      common /cutoff/ hcut
c
c     Evaluate the three-body energy up to the fourth-order, triple-
c     dipole term.
c
      sumararar=0.0
      n3b=0
      totararar1=0.0
      totararar2=0.0
      totararar3=0.0
      totararar4=0.0
      totararar5=0.0

c
      do 40 il=1,nar-2

      xar0=xar(il)
      yar0=yar(il)
      zar0=zar(il)

      totalddd=0.00
      totalddq=0.00
      totaldqd=0.00
      totalqdd=0.00
      totaldqq=0.00
      totalqdq=0.00
      totalqqd=0.00
      totalqqq=0.00
      totald4=0.000
```

```
      do 50 im=il+1,nar-1
c
      xar1=xar(im)
      yar1=yar(im)
      zar1=zar(im)
      dif1x=xar1-xar0
      dif1y=yar1-yar0
      dif1z=zar1-zar0
      rar0ar1=sqrt(dif1x*dif1x+dif1y*dif1y+dif1z*dif1z)
c


      if(rar0ar1.gt.hcut) then
      go to 50
      endif
c
      do 60 in=im+1,nar
c

      xar2=xar(in)
      yar2=yar(in)
      zar2=zar(in)
      dif2x=xar2-xar0
      dif2y=yar2-yar0
      dif2z=zar2-zar0
c
      dif3x=xar2-xar1
      dif3y=yar2-yar1
      dif3z=zar2-zar1
      rar0ar2=sqrt(dif2x*dif2x+dif2y*dif2y+dif2z*dif2z)
      rar1ar2=sqrt(dif3x*dif3x+dif3y*dif3y+dif3z*dif3z)
c
      if(rar0ar2.gt.hcut) then
      go to 60
      endif
c
      cos1=(dif1x*dif2x+dif1y*dif2y+dif1z*dif2z)/(rar0ar1*rar0ar2)
      cos2=(-dif1x*dif3x-dif1y*dif3y-dif1z*dif3z)/(rar0ar1*rar1ar2)
      cos3=(dif2x*dif3x+dif2y*dif3y+dif2z*dif3z)/(rar0ar2*rar1ar2)
c
      cosangledif1=cos1*cos2+sqrt(abs((1-cos1**2)*(1-cos2**2)))
      cosangledif2=cos2*cos3+sqrt(abs((1-cos2**2)*(1-cos3**2)))
      cosangledif3=cos3*cos1+sqrt(abs((1-cos3**2)*(1-cos1**2)))
c
      cos3fi1=-3*cos1+4*cos1**3
      cos3fi2=-3*cos2+4*cos2**3
      cos3fi3=-3*cos3+4*cos3**3
c
      cos2fi1=2*cos1**2-1
      cos2fi2=2*cos2**2-1
      cos2fi3=2*cos3**2-1
c
      cos2angledif1=2*(cosangledif1)**2-1
      cos2angledif2=2*(cosangledif2)**2-1
      cos2angledif3=2*(cosangledif3)**2-1
c
      cos4fi1=8*cos1**3*sqrt(abs(1-cos1**2))-4*cos1*sqrt(abs(1-cos1**2))
```

```
      cos4fi2=8*cos2**3*sqrt(abs(1-cos2**2))-4*cos2*sqrt(abs(1-cos2**2))
      cos4fi3=8*cos3**3*sqrt(abs(1-cos3**2))-4*cos3*sqrt(abs(1-cos3**2))

c     in all following functions, the order ar0,ar1,ar2 is considered
c
c     function three body dipole-dipole-dipole
c
      fddd=3.0*addd*(1.0+3.0*cos1*cos2*cos3)*
     +(rar0ar1*rar0ar2*rar1ar2)**-3.0
c
c     function three body dipole-dipole-quadrupole
c
      fddq=addq*3.0/(16.0*(rar0ar1**3)*(rar0ar2*rar1ar2)**4)*
     +((9.0*cos3-25.0*cos3fi3)+6.0*cosangledif1*(3.0+5.0*cos2fi3))
c
c     function three body dipole-quadrupole-dipole
c
      fdqd=adqd*3.0/(16.0*(rar0ar2**3)*(rar1ar2*rar0ar1)**4)*
     +((9.0*cos2-25.0*cos3fi2)+6.0*cosangledif3*(3.0+5.0*cos2fi2))
c
c     function three body quadrupole-dipole-dipole
c
      fqdd=aqdd*3.0/(16.0*(rar1ar2**3)*(rar0ar2*rar0ar1)**4)*
     +((9.0*cos1-25.0*cos3fi1)+6.0*cosangledif2*(3.0+5.0*cos2fi1))
c
c     function three body dipole-dipole-dipole fourth order
c
      fd4=ad41*(1.0/(rar0ar1*rar1ar2)**6)*(1.0+cos2*cos2)+
     +ad42*1.0/((rar1ar2*rar0ar2)**6)*(1.0+cos3*cos3)+
     +ad43*1.0/((rar0ar2*rar0ar1)**6)*(1.0+cos1*cos1)
c
c     function three body dipole-quadrupole-quadrupole
c
      fdqq=adqq*15.0/(64.0*(rar1ar2**5)*(rar0ar1*rar0ar2)**4)*
     +(3.0*(cos1+5.0*cos3fi1)+20.0*cosangledif2*(1.0-3.0*cos2fi1)+
     +70.0*cos2angledif2*cos1)
c
c     function three body quadrupole-dipole-quadrupole
c
      fqdq=aqdq*15.0/(64.0*(rar0ar2**5)*(rar1ar2*rar0ar1)**4)*
     +(3.0*(cos2+5.0*cos3fi2)+20.0*cosangledif3*(1.0-3.0*cos2fi2)+
     +70.0*cos2angledif3*cos2)
c
c     function three body quadrupole-quadrupole-dipole
c
      fqqd=aqqd*15.0/(64.0*(rar0ar1**5)*(rar1ar2*rar0ar2)**4)*
     +(3.0*(cos3+5.0*cos3fi3)+20.0*cosangledif1*(1.0-3.0*cos2fi3)+
     +70.0*cos2angledif1*cos3)
c
c     function three body quadrupole-quadrupole-quadrupole
c
      fqqq=aqqq*15.0/(128.0*(rar0ar1*rar1ar2*rar0ar2)**5)*
     +(-27.0+220.0*cos1*cos2*cos3+490.0*cos2fi1*cos2fi2*cos2fi3+
     +175.0*(cos2angledif1+cos2angledif2+cos2angledif3))

c
c     make the sum at each step for each sub-term
c
```

```
      totalddd=totalddd+fddd
c
      totalddq=totalddq+fddq
      totaldqd=totaldqd+fdqd
      totalqdd=totalqdd+fqdd
c
      totald4=totald4+fd4
c
      totaldqq=totaldqq+fdqq
      totalqdq=totalqdq+fqdq
      totalqqd=totalqqd+fqqd
c
      totalqqq=totalqqq+fqqq

      n3b=n3b+1

60    continue
50    continue
c
      totararar1=totalddd
      totararar2=totalddq+totaldqd+totalqdd
      totararar3=totalqdq+totalqqd+totaldqq
      totararar4=totalqqq
      totararar5=totald4
      sumararar= sumararar+totararar1+totararar2+totararar3+
     +totararar4+totararar5
c      print*,sumararar
40    continue
      print*,'number of ararar interactions  ',n3b
      return
      end

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

      subroutine convertok(thing)

      common /bfwparams/ a0arar,a1arar,a2arar,a3arar,a4arar,a5arar,
     +                   aarar,deltarar,epsilbfw,radbfw
c
c     Subroutine that convert from atomic units to Kelvin
c

      thing=thing*epsilbfw

      return
      end
```

Appendix II

Calculation of the Born-Mayer parameters for the Tang-Toennies potential

Programs:

(1) **repara.f**: calculates the A and b coefficients using the Tang-Toennies procedure (see chapter three of the report).

- input: 'params.dat'

- output: 'results.dat'

(2) **abcalcul.f**: uses the Newton-Raphson technique explained in chapter three of the report to determine the Born-Mayer parameters simultaneously.

(3) **bar.f**: it fits the Tang-Toennies potential to the well-depth of the Barker-Fisher-Watts potential (142.1 K) by adjusting the b repulsive parameter.

```
      PROGRAM repara

c     This program determines the parameters for the repulsive part
c     of the van der Waals pot. proposed by Tang et al.
c     (J. Chem. Phys 80(8), 1984:3726)
c
c     Uses the Newton-Raphson method to determine the two parameters via
c     an iterative process until self-consistency is reached.
c
c
c     Written by Felix Fernandez-Alonso
c     Department of Chemistry, Imperial College, London
c     February 1993
c
      common/param/bguess,cyclesml,cycleslb,cycleslab,contolml,contolb,
     +contolab,c6,c8,c10
      common/reparam/b1,b2,a1

c     Initialise variables

      OK=0
      b1=0.0
      b2=0.0
      a1=0.0

c     Read from datafile all the variables needed in the program

      call readparam

c     Convert c6,c8,c10 to reduced units

      call redunits

c     Open a data file where errors and/or results are displayed

c      open(unit=2,file='results.dat')

c     Start the main loop

      do 10 n=1,cyclesml

      if (n.eq.1) then
      b1=bguess
      endif

      call newtonb(b1)
      a1=calculatea(b1)
      call newtonab(b2)

      if ((abs(b1-b2).gt.1.0e-5).and.((b1.ne.0).or.(b2.ne.0))) then
      b1=b2
      else
      n=cycles
      OK=1
      endif

10    continue
```

```
      if (OK.eq.1) then
c      write(2,*) a1,b1,b2
      print*, a1,b1,b2
      else
c      write(2,*) a1,b1,b2
c      write(2,*) 'exceeded max of ',cyclesml,' in main loop.'
      print*,'a1   b1     b2'
      print*,a1,b1,b2
      endif
      print*,'end of program'
      end

      subroutine readparam
      common/wellparam/epsilon,epsrad
      common/param/bguess,cyclesml,cycleslb,cycleslab,contolml,contolb,
     +contolab,c6,c8,c10

      open(unit=1,file='params.dat')

      read(1,*) epsilon
      read(1,*) epsrad
      read(1,*) bguess
      read(1,*) cyclesml
      read(1,*) cycleslb
      read(1,*) cycleslab
      read(1,*) contolml
      read(1,*) contolb
      read(1,*) contolab
      read(1,*) c6
      read(1,*) c8
      read(1,*) c10

      return
      end

      subroutine redunits

      common/wellparam/epsilon,epsrad
      common/param/bguess,cyclesml,cycleslb,cycleslab,contolml,contolb,
     + contolab,c6,c8,c10

      c6=(c6/(epsilon*(epsrad**6)))
      c8=(c8/(epsilon*(epsrad**8)))
      c10=(c10/(epsilon*(epsrad**10)))

      return
      end


      function potentialb(x1)

      common/param/bguess,cyclesml,cycleslb,cycleslab,contolml,contolb,
     + contolab,c6,c8,c10

      a6=0.0
      a8=0.0
      a10=0.0
      d6=0.0
```

```
      d8=0.0
      d10=0.0

      do 10 il=1,6
      a6 = a6 + ((x1**il)/it)
      if (il.eq.6) then
      d6 = (x1**il)/it
      endif
      it = (il+1)*it
10     continue
      a8 = a6+a8
      do 20 im=7,8
      a8 = a8 + ((x1**im)/it)
      if (im.eq.8) then
      d8 = (x1**im)/it
      endif
      it = (il+1)*it
20     continue
      a10 = a8+a10
      do 30 in=9,10
      a10 = a10 + ((x1**in)/it)
      if (in.eq.10) then
      d10 = (x1**in)*it
      endif
      it = (in+1)*it
30     continue

      b6=(1-(a6*exp(-x1)))
      b8=(1-(a8*exp(-x1)))
      b10=(1-(a10*exp(-x1)))

      g6=c6*b6*((12/x1)-1)
      g8=c8*b8*((18/x1)-1)
      g10=c10*b10*((20/x1)-1)

      e6=c6*d6*exp(-x1)
      e8=c8*d8*exp(-x1)
      e10=c10*d10*exp(-x1)

      f6=g6-e6
      f8=g8-e8
      f10=g10-e10

      potentialb=f6+f8+f10+1

      return
      end

      function dpotentialb(x1)

      common/param/bguess,cyclesml,cycleslb,cycleslab,contolml,contolb,
     + contolab,c6,c8,c10

      a6=0.0
      a8=0.0
      a10=0.0

      do 10 il=1,6
```

```
      a6 = a6 + ((x1**il)/it)
      it = (il+1)*it
10     continue
      a8 = a6+a8
      do 20 im=7,8
      a8 = a8 + ((x1**im)/it)
      it = (il+1)*it
20     continue
      a10 = a8+a10
      do 30 in=9,10
      a10 = a10 + ((x1**in)/it)
      it = (in+1)*it
30     continue

      b6=(1-(a6*exp(-x1)))
      b8=(1-(a8*exp(-x1)))
      b10=(1-(a10*exp(-x1)))

      d6=-b6*(12/(x1**2))
      d8=-b8*(18/(x1**2))
      d10=-b10*(20/(x1**2))

      e6=c6*d6
      e8=c8*d8
      e10=c10*d10

      dpotentialb=e6+e8+e10

      return
      end

      function dpotentialab(adum,x1)

      common/param/bguess.cyclesml,cycleslb,cycleslab,contolml,contolb,
     +contolab,c6,c8,c10
      common/reparam/b1,b2,a1

      repterm=0.0
      d6=0.0
      d8=0.0
      d10=0.0

      do 10 il=1,6
      if (il.eq.6) then
      d6 = (x1**il)/it
      endif
      it = (il+1)*it
10     continue
      do 20 im=7,8
      if (im.eq.8) then
      d8 = (x1**im)/it
      endif
      it = (il+1)*it
20     continue
      do 30 in=9,10
      if (in.eq.10) then
      d10 = (x1**in)*it
      endif
```

```
      it = (in+1)*it
30     continue

      b6=(c6*d6*exp(-x1))
      b8=(c8*d8*exp(-x1))
      b10=(c10*d10*exp(-x1))

      repterm=-(adum*exp(-x1))

      dpotentialab=repterm-(b6+b8+b10)

      return
      end

      function potentialab(adum,x1)

      common/param/bguess,cyclesml,cycleslb,cycleslab,contolml,contolb,
     +contolab,c6,c8,c10
      common/reparam/b1,b2,a1

      repterm=0.0
      a6=0.0
      a8=0.0
      a10=0.0

      do 10 il=1,6
      a6 = a6 + ((x1**il)/it)
      it = (il+1)*it
10     continue
      a8 = a6+a8
      do 20 im=7,8
      a8 = a8 + ((x1**im)/it)
      it = (il+1)*it
20     continue
      a10 = a8+a10
      do 30 in=9,10
      a10 = a10 + ((x1**in)/it)
      it = (in+1)*it
30     continue

      b6=(1-(a6*exp(-x1)))
      b8=(1-(a8*exp(-x1)))
      b10=(1-(a10*exp(-x1)))

      g6=c6*b6
      g8=c8*b8
      g10=c10*b10

      repterm=adum*(exp(-x1))

      potentialab=repterm-(g6+g8+g10)+1

      return
      end

      subroutine newtonb(bb)

      common/param/bguess,cyclesml,cycleslb,cycleslab,contolml,contolb,
```

```
     +contolab,c6,c8,c10

      x=bb
      i=0
      funct=0.0
      derivative=0.0

1     funct= potentialb(x)
      derivative= dpotentialb(x)

c     Newton-Raphson algorithm breaks down if the derivative is zero.

      if ((abs(derivative).eq.0).or.(abs(derivative).lt.1.0e-5)) then
      print*,'NEWTONb FAILED TO CONVERGE. DERIV=0'
      bb=0.0
      goto 10
      endif

      dx=-(funct)/(derivative)
      x=x+dx
      i=i+1

      if (i.eq.cyclesb) then
      print*,'CONVERGENCE NOT REACHED after ',cyclesb,' cycles'
      bb=0.0
      goto 10
      else if (abs(dx).lt.contolb) then
      print*,'CONVERGENCE REACHED after ', i,' steps'
      print*,'The value of b* is ',x
      bb=x
      goto 10
      else
      goto 1
      endif
10    return
      end

      subroutine newtonab(ab)

      common/param/bguess,cyclesml,cycleslb,cycleslab,contolml,contolb,
     +contolab,c6,c8,c10

      x=ab
      i=0
      funct=0.0
      derivative=0.0

1     funct= potentialab(x)
      derivative= dpotentialab(x)

c     Newton-Raphson algorithm breaks down if the derivative is zero.

      if ((abs(derivative).eq.0).or.(abs(derivative).lt.1.0e-5)) then
      print*,'NEWTONb FAILED TO CONVERGE. DERIV=0'
      ab=0.0
      goto 10
      endif
```

```
      dx=-(funct)/(derivative)
      x=x+dx
      i=i+1

      if (i.eq.cyclesab) then
      print*,'CONVERGENCE NOT REACHED after ',cyclesab,' cycles'
      ab=0.0
      goto 10
      else if (abs(dx).lt.contolab) then
      print*,'CONVERGENCE REACHED after ', i,' steps'
      print*,'The value of b* is ',x
      ab=x
      goto 10
      else
      goto 1
      endif
10    return
      end

      function calculatea(x1)

      common/param/bguess,cyclesml,cycleslb,cycleslab,contolml,contolb,
     +contolab,c6,c8,c10

      a6=0.0
      a8=0.0
      a10=0.0
      d6=0.0
      d8=0.0
      d10=0.0

      do 10 il=1,6
      a6 = a6 + ((x1**il)/it)
      if (il.eq.6) then
      d6 = (x1**il)/it
      endif
      it = (il+1)*it
10    continue
      a8 = a6+a8
      do 20 im=7,8
      a8 = a8 + ((x1**im)/it)
      if (im.eq.8) then
      d8 = (x1**im)/it
      endif
      it = (il+1)*it
20    continue
      a10 = a8+a10
      do 30 in=9,10
      a10 = a10 + ((x1**in)/it)
      if (in.eq.10) then
      d10 = (x1**in)*it
      endif
      it = (in+1)*it
30    continue

      b6=(exp(x1))-a6
      b8=(exp(x1))-a8
      b10=(exp(x1))-a10
```

```
g6=c6*b6*((12/x1))
g8=c8*b8*((18/x1))
g10=c10*b10*((20/x1))

e6=c6*d6
e8=c8*d8
e10=c10*d10

f6=g6-e6
f8=g8-e8
f10=g10-e10

calculatea=f6+f8+f10

return
end
```

```
      PROGRAM abcalcul.f

c     This program determines the parameters for the repulsive part
c     of the van der Waals potential proposed by Tang et al.
c     (J. Chem. Phys. 80(8), 1984:3726)
c
c     Uses the Newton-Raphson method to determine the two parameters via
c     an iterative process.
c     Essentially, the potential at the equilibrium distance is expanded
c     as a Taylor series in terms of A and b. The program will iterate
c     until convergence.
c
c
c     Written by Felix Fernandez-Alonso
c     Department of Chemistry, Imperial College, London
c     February 23, 1993
c

      implicit double precision (a-h,o-z)

      common/wellparam/epsilon,epsrad
      common/param/aguess,bguess,mcycles,contol,c6,c8,c10
      common/reparam/a1,b1

c     Read from datafile all the variables needed in the program

      call readparam
      print*,'parameters read in ...'
c     Convert c6,c8,c10 to reduced units
      call redunits
      print*,'convert to reduced units ...'
c     call subroutine which calculates a and b
      call abcalcul(a1,b1)
c     call subroutine that converts A and b to real units
      call realunits(a1,b1)
      print*,'a=  ',a1
      print*,'b=  ',b1
      print*,'end of program'
      end

      subroutine readparam

      implicit double precision (a-h,o-z)

      common/wellparam/epsilon,epsrad
      common/param/aguess,bguess,mcycles,contol,c6,c8,c10

      open(unit=1,file='nrpar.dat')

      read(1,*) epsilon
      read(1,*) epsrad
      read(1,*) aguess
      read(1,*) bguess
      read(1,*) mcycles
      read(1,*) contol
      read(1,*) c6
      read(1,*) c8
      read(1,*) c10
```

```
      print*, epsilon
      print*, epsrad
      print*, aguess
      print*, bguess
      print*, mcycles
      print*, contol
      print*, c6
      print*, c8
      print*, c10

      return
      end

      subroutine redunits

      implicit double precision (a-h,o-z)

      common/wellparam/epsilon,epsrad
      common/param/aguess,bguess,mcycles,contol,c6,c8,c10

      c6=(c6/(epsilon*(epsrad**6)))
      c8=(c8/(epsilon*(epsrad**8)))
      c10=(c10/(epsilon*(epsrad**10)))
      aguess=aguess/epsilon
      bguess=bguess*epsrad

      return
      end

      function dbpotab(adum,x1)

      implicit double precision (a-h,o-z)

      common/param/aguess,bguess,mcycles,contol,c6,c8,c10

      it=1.0
      repterm=0.0
      d6=0.0
      d8=0.0
      d10=0.0

      do 10 il=1,6
      if (il.eq.6) then
      d6 = (x1**il)/it
      endif
      it = (il+1)*it
10     continue

      do 20 im=7,8
      if (im.eq.8) then
      d8 = (x1**im)/it
      endif
      it = (im+1)*it
20     continue

      do 30 in=9,10
      if (in.eq.10) then
      d10 = (x1**in)/it
```

```
      endif
      it = (in+1)*it
30    continue

      b6=(c6*d6*exp(-x1))
      b8=(c8*d8*exp(-x1))
      b10=(c10*d10*exp(-x1))

      repterm=-(adum*exp(-x1))

      dbpotab=repterm-(b6+b8+b10)

      return
      end

      function dapotab(x1)

      implicit double precision (a-h,o-z)

      common/param/aguess,bguess,mcycles,contol,c6,c8,c10

      repterm=0.0
      repterm=exp(-x1)
      dapotab=repterm

      return
      end


      function potab(adum,x1)

      implicit double precision (a-h,o-z)

      common/param/aguess,bguess,mcycles,contol,c6,c8,c10

      it=1.0
      repterm=0.0
      a6=1.0
      a8=0.0
      a10=0.0

      do 10 il=1,6
      a6 = a6 + ((x1**il)/it)
      it = (il+1)*it
10    continue

      a8=a6

      do 20 im=7,8
      a8 = a8 + ((x1**im)/it)
      it = (im+1)*it
20    continue

      a10=a8

      do 30 in=9,10
      a10 = a10 + ((x1**in)/it)
      it = (in+1)*it
```

```
30    continue

      b6=(1-(a6*exp(-x1)))
      b8=(1-(a8*exp(-x1)))
      b10=(1-(a10*exp(-x1)))

      g6=c6*b6
      g8=c8*b8
      g10=c10*b10

      repterm=adum*(exp(-x1))

      potab=repterm-(g6+g8+g10)

      return
      end

      function dpotab(adum,x1)

      implicit double precision (a-h,o-z)

      common/param/aguess,bguess,mcycles,contol,c6,c8,c10

      it=1.0
      repterm=0.0
      a6=1.0
      a8=0.0
      a10=0.0

      do 10 il=1,6
      a6 = a6 + ((x1**il)/it)
      if (il.eq.6) then
      d6 = ((x1)**il)/it
      endif
      it = (il+1)*it
10    continue

      a8=a6

      do 20 im=7,8
      a8=a8+((x1**im)/it)
      if (im.eq.8) then
      d8=((x1)**im)/it
      endif
      it=(im+1)*it
20    continue

      a10=a8

      do 30 in=9,10
      a10=a10+((x1**in)/it)
      if (in.eq.10) then
      d10=((x1)**in)/it
      endif
      it=(in+1)*it
30    continue

      b6=(1-(a6*exp(-x1)))
```

```
      b8=(1-(a8*exp(-x1)))
      b10=(1-(a10*exp(-x1)))
      g6=12*c6*b6
      g8=16*c8*b8
      g10=20*c10*b10

      e6=(x1)*(exp(-x1))*d6
      e8=(x1)*(exp(-x1))*d8
      e10=(x1)*(exp(-x1))*d10

      repterm=(-x1)*(adum)*(exp(-x1))

      dpotab=repterm-(e6+e8+e10)+(g6+g8+g10)

      return
      end

      function dadpotab(x1)

      implicit double precision (a-h,o-z)

      common/param/aguess,bguess,mcycles,contol,c6,c8,c10

      repterm=0.0
      repterm=(-x1)*(exp(-x1))
      dadpotab=repterm

      return
      end

      function dbdpotab(adum,x1)

      implicit double precision (a-h,o-z)

      common/param/aguess,bguess,mcycles,contol,c6,c8,c10

      it=1.0
      repterm=0.0
      a6=1.0
      a8=0.0
      a10=0.0

      do 10 il=1,6
      if (il.eq.6) then
      d6 = ((x1)**il)/it
      endif
      it = (il+1)*it
10     continue

      a8=a6

      do 20 im=7,8
      if (im.eq.8) then
      d8 = ((x1)**im)/it
      endif
      it = (im+1)*it
20     continue
```

```
      a10=a8

      do 30 in=9,10
      if (in.eq.10) then
      d10 = ((x1)**in)/it
      endif
      it = (in+1)*it
30     continue

      b6=a6*c6
      b8=a8*c8
      b10=a10*c10

      dbdpotab=(x1-1)*(exp(-x1))*(adum+b6+b8+b10)

      return
      end

      subroutine abcalcul(a,b)

      implicit double precision (a-h,o-z)

      common/param/aguess,bguess,mcycles,contol,c6,c8,c10

      open(unit=10,file='results.dat')

      do 10 i=1,mcycles

      if (i.eq.1) then
      a=aguess
      b=bguess
      print*,'initial A* ',a
      print*,'initial b* ',b
      endif
c     calculate all necessary quantities

      pot=potab(a,b)
      dapot=dapotab(b)
      dbpot=dbpotab(a,b)

      dpot=dpotab(a,b)
      dadpot=dadpotab(b)
      dbdpot=dbdpotab(a,b)

      pot=pot+1

c     calculate dA and db using Cramer's rule.
c     note: program crashes at this point if det1=0.

      det1=(dapot*dbdpot)-(dbpot*dadpot)
      det2=(pot*dbdpot)-(dpot*dbpot)
      det3=(dapot*dpot)-(pot*dadpot)

      write(10,*)'cycle ',i
      write(10,*)'det1  ',det1
      write(10,*)'det2  ',det2
      write(10,*)'det3  ',det3
```

```
      if (abs(det1).lt.1.0D-30) then
      print*,'Cramers rule fails ... END OF CALCULATION.'
      goto 20
      endif

      da= det2/det1
      db=det3/det1

      if ((abs(da).lt.contol).and.(abs(db).lt.contol)) then
      print*,'convergence reached after ',i,' steps'
      goto 20
      endif
      a=a+da
      b=b+db
c     write to output file

      write(10,*)'A*     ',a
      write(10,*)'b*,    ',b

10    continue
      print*,'Convergence not reached.'
20    return
      end

      subroutine realunits(a,b)

      common/wellparam/epsilon,epsrad

      a=a*epsilon
      b=b/epsrad

      return
      end
```

```
      PROGRAM bar.f

c     This program fits the Tang and Toennies potential for argon
c     to the minimum given of the Barker-Fisher and Watts potential.
c     For further reference see:
c     J. A. Barker and A. Pompe, Aust. J. Chem.,1968,21,1683
c     K.T. Tang and J. P. Toennies, J. Chem. Phys., 80(8), 1984: 3726

      common/variables/r,entang,b

      call parameters
      call potcalcul
      print*,'end of program'
      end


      subroutine parameters

      common/param/arep,bguess,c6,c8,c10,eptang,rmtang

      eptang=4.54e-4
      rmtang=7.10
      arep=328.85
      bguess=1.918
      c6=68.305440
      c8=1141.964
      c10=38559 74

      print*,'parameters in ...'
      return
      end

      function tangpot(x1,b1)

      common/param/arep,bguess,c6.c8,c10,eptang,rmtang

      it=1.0
      repterm=0.0
      a6=1.0
      a8=0.0
      a10=0.0

      do 10 il=1,6
      a6 = a6 + (((x1*b1)**il)/it)
      it = (il+1)*it
10     continue

      a8=a6

      do 20 im=7,8
      a8 = a8 + (((x1*b1)**im)/it)
      it = (im+1)*it
20     continue

      a10=a8

      do 30 in=9,10
      a10 = a10 + (((x1*b1)**in)/it)
```

```
      it = (in+1)*it
30     continue

      b6=(1-(a6*exp(-x1*b1)))
      b8=(1-(a8*exp(-x1*b1)))
      b10=(1-(a10*exp(-x1*b1)))

      g6=c6*b6/(x1**6)
      g8=c8*b8/(x1**8)
      g10=c10*b10/(x1**10)

      repterm=arep*(exp(-x1*b1))

      tangpot=repterm-(g6+g8+g10)

      return
      end

      subroutine realunits(en)

      en=en*3.1578e+5


      return
      end

      subroutine potcalcul

      common/variables/r,entang,b
      common/param/arep,bguess,c6,c8,c10,eptang,rmtang

      r=7.10
      enmin=-140.750
      endiff=0.0
      b=bguess
c      open(unit=4,file='bparam.dat')

      do 10 i=1,80000

      entang=tangpot(r,b)
      call realunits(entang)

      endiff=enmin-entang

      if (endiff.gt.0.000) then
      print*,'convergence reached'
      print*,'b= ',b
      goto 20
      endif
      print*,b,entang
      b=bguess+0.00001*i
10     continue
      print*,'convergence not reached'
      print*,'b = ',b
20     return
      end
```

Appendix III

Three-body calculation in silicalite-1

Programs:

(1) **msimu6.f**: an NVT molecular simulation in silicalite-1. Adsorbate molecules are neither created not destroyed.

-input: grid data ('zsmgrd3.dat', 'gauss1.dat', 'gauss2.dat', 'gauss3.dat'), argon coordinates ('conit5.bak'), starting variables such as temperature, pressure, etc ('vanit5.dat') and Lennard-Jones parameters ('ljpotsim.dat').

-output: for the purposes of this calculation it ouputs a new configuration (in atomic units) of the sorbate in the zeolite in the file 'conit5.dat'.

(2) **argcor.f**: writes the coordinates of the adsorbate molecules in reduced coordinates in the file 'arpos.dat'

(3) **shcalc4.f**:

-input: dispersion parameters,etc ('parama5.dat') and coordinates of the oxygens ('zsmox.dat'), silicon ('zsmsi.dat') and argon ('arpos.dat').

-output: in file 'result.tem', includes all the raw data from the two-body and three-body computation for the argon atoms in the central unit cell of the twenty-seven unit cell system. See source code for more detailed information on the calculation.

(4) **shanal.f**: analyses the raw data from a single calculation.

-input: 'result.tem' from shcalc4.f

-output: 'analysis1.dat', the final results for a single calculation.

(5) **runanal.f**: analyses the data from a series of runs at a given coverage and gives the averages and standard deviations for two- and three-body energies.

-input: 'rundata.tem', file that contains all the analyzed data for all the calculations at a given coverage.

-output: 'runres.dat', the final results for a given coverage, including mean values and standard deviations.

(6) **anal3b.f**: it reads a file containing the coordinates (in atomic units) of specific three-body configurations (i.e. those that exceed one K). It then calculates the

distances and angles of the atoms of each configuration.

-input: 'x3b.dat', containing the coordinates of the argon, oxygen and silicon atoms of interest.

-output: 'x3bo.dat', containing the distances (in atomic units) and angles for a given configuration.

Calculation:

msimu6.f moves the adsorbate particles inside the zeolite cavity. Every one hundred thousand adsorbate moves, the configuration is written in 'arpos.dat'. Then, the program shcalc4.f calculates the two- and three- body energies for that given configuration.

This procedure is repeated several times, usually twelve, and runanal.f calculates the mean values and standard deviations for each of the computed quantities (i.e. energies, deviations, ArArAr, ArArO, ArArSi energies, DDD, DDQ terms, etc).

```
      program msimu6
      implicit double precision(a-h,o-z)

c     Ar/Silicalite-1
c     London january-february-march-may-June-July-september 1992
c     Program using simulation method Monte-Carlo GCMC.
c     The isosteric heat of adsorption, the Ar-Ar contribution and the
c     the Ar-O contribution to the internal energy are calculated.
c     The simulation box is a cube of 3*3*3 unit cells.
c     The adsorption potential model uses an interpolated grid
c     calculated previously point grid.
c
      call startf
      call lectur
      call gridy
      call simula
      call result
      end
c
c
c
c
      subroutine startf
      implicit double precision(a-h,o-z)

c
c
c
c
      common /con/ looche,loosum,lootot,rumvol
      common /pot/ siArAr2,epArAr,epArAr4
      common /ran/ ix1,ix2,ix3,r(97)
      common /sta/ cutp01,pressr,sqenet,sqdent
      common /cb1/ xpopos(1000),ypopos(1000),zpopos(1000),
     +ixdist(1000),iydist(1000),izdist(1000),cucell,haspac,
     +aucell,bucell,hasidx,hasidy,hasidz,
     +volpor,recvol,cutp02,inumpo,inumm1,inump1,temper,zedvee,
     +steple,nwaren,nwarmo,aucell,bucell,cucell,numsid
      common /cb2/ nwarlo,nwarhi,ipasno,ipasnn,ipasto,iflamp,
     +iactra,iacdes,iaccre,loopas,runnen,eneave,eruave,
     +denave,druave,widsidx,widsidy,widsidz,twoxpi
      common /cb3/ bolcon,rarega,testen,heatad,
     +enenum,numave,sqnuve,ebenum,tubave,sqbuve
      common /cb4/ runaro,runarar,aroave,ararave,broave,brarave,
     +enaronum,enararnum,ebaronum,ebararnum,heataro,heatArAr,eninte,
     +enarointe,enararinte
      common /cb5/ grid3d(104,28,68),w1(104),w2(28),w3(68),
     +idim1,idim2,idim3
c
c
c **** Initialise data.
c
c
c Open files.
      open(11,file='vanit5.dat')
      open(12,file='conit5.bak')
      open(2,file='ljpotsim.dat')
c ** Data for potential functions.
```

```
c Ar-Ar dispersion repulsion LJ potential constants.
      read(2,*) siArAr
      siArAr2=siArAr*siArAr
      read(2,*) epArAr
      epArAr4=4*epArAr
c ** Data Avogadro number
      avogad=6.022045D23
c ** Data Boltzmann constant
      bolcon=1.380658D-23
c ** Data-rarega is the rare gas constant
      rarega=8.31451D0
c ** Data for simulation.
c temper-temperature of the system in K.
      read(11,*) temper
c pressr-specified pressure in Pa.
      read(11,*) pressr
c cutp01-spherical cutoff distance beyond which no interactions
c between molecules are considered.
      read(11,*) cutp01
      cutp02=cutp01*cutp01
c twoxpi-two times pi.
      pi=3.14159265359D0
      twoxpi=2*pi
c testen-energy test which determines if the molecule is in side or
c outside the cavity
      read(11,*) testen
c lootot-number of passes per run.
      read(11,*) lootot
c loopas-number of times through the loop per pass.
      read(11,*) loopas
c loosum-number of passes before partial sums are output.
      read(11,*) loosum
c looche-number of passes before energy drift is checked.
      read(11,*) looche
c presentation chart
c     print*,'Simulation Ar/Silicalite-1 Chart'
c     print*,' Pellenq-Nicholson model'
c Define initial unit cell dimensions
      aucell=20.07D0
      bucell=19.92D0
      cucell=13.42D0
c iactra-number of translation acceptances.
c iacdes-number of destruction acceptances.
c iaccre-number of creation acceptances.
      iactra=0
      iacdes=0
      iaccre=0
c ** Read restart data.
c iresfl-flag which resets averages, number of passes and the
c distribution functions when raised.
      read(12,*) iresfl
c aucell-length of the box side along x direction.
      read(12,*) aucell
c hasidx-half of the box side along the x direction
      hasidx=aucell/2.
c bucell-length of the box side along y direction.
      read(12,*) bucell
c hasidy-half of the box side along the y direction
```

```
      hasidy=bucell/2.
c cucell-length of the box side along z direction.
      read(12,*) cucell
c hasidz-half of the box side along the z direction
      hasidz=cucell/2.
c
      widsidx=aucell/1000.0D0
      widsidy=bucell/1000.0D0
      widsidz=cucell/1000.0D0
c volpor-volume of the box space.
      volpor=aucell*bucell*cucell
c numsid-number of slices used in distribution function in each direction.
      numsid=1000
c zedvee-constant used in creation and destruction decision-making.
      zedvee=pressr*volpor*7.2435D-8/temper
c steple-molecule move translation step-length.
      read(12,*) steple
c eneave-sum of total energy terms.
c aroave-sum of Ar-wall energy terms.
c ararave-sum of Ar-Ar energy terms.
c enenum-sum of total energy*number of particles products.
c enaronum-sum of Ar-O energy*number of particle products.
c enararnum-sum of Ar-Ar energy*number of particle products.
c numave-sum of particles number
c sqnuve-sum of particles number squares.
c sqenet-sum of energy squares.
c denave-sum of density terms.
c sqdent-sum of density squares.
c ipasno-number of passes since reset.
c ipasto-total number of passes.
      read(12,*) eneave
      read(12,*) aroave
      read(12,*) ararave
      read(12,*) enenum
      read(12,*) enaronum
      read(12,*) enararnum
      read(12,*) numave
      read(12,*) sqnuve
      read(12,*) sqenet
      read(12,*) denave
      read(12,*) sqdent
      read(12,*) ipasno
      read(12,*) ipasto
c
      if (iresfl.eq.1) then
      eneave=0.
      aroave=0
      ararave=0
      enenum=0.
      enaronum=0.
      enararnum=0.
      numave=0.
      sqnuve=0.
      sqenet=0.
      denave=0.
      sqdent=0.
      ipasno=0.
      endif
```

```
      ipasnn=0
c inumpo-number of argon atoms in pore.
      read(12,*) inumpo
      inump1=inumpo+1
      inumm1=inumpo-1
c rumvol-constant used in density calculation.
      rumvol=DBLE(inumpo)/volpor
c Set warning flag.
      iflche=0
c List xpopos-x-coordinates of the pore molecules.
c List ypopos-y-coordinates of the pore molecules.
c List zpopos-z-coordinates of the pore molecules.
      do 30 loop1=1,inumpo
      read(12,*) xtar01
      read(12,*) ytar01
      read(12,*) ztar01
      xpopos(loop1)=xtar01
      ypopos(loop1)=ytar01
      zpopos(loop1)=ztar01
c Raise flag if molecule is outside pore.
      iflche=iflche+IDINT(DABS(xtar01)/hasidx)+
     +IDINT(DABS(ytar01)/hasidy)+IDINT(DABS(ztar01)/hasidz)
30    continue
c If the flag has been raised, molecules have inappropriate coordinates.
      if (iflche.gt.0) then
      write(14,2000)
2000  format(' ** Warning! **')
      write(14,2010)
2010  format(' Molecules initially outside pore.')
      write(14,*)
      endif
c List ixdist-bins for the singlet distribution function in x.
c List iydist-bins for the singlet distribution function in y.
c List izdist-bins for the singlet distribution function in z.
      do 40 loop1=1,1000
      read(12,*) ixdist(loop1)
      read(12,*) iydist(loop1)
      read(12,*) izdist(loop1)
40    continue
      if (iresfl.eq.1) then
      do 70 loop1=1,1000
      ixdist(loop1)=0
      iydist(loop1)=0
      izdist(loop1)=0
70    continue
      endif
c ix1 to 3-seeds for ran1.
c List r(97)-table for ran1.
      read(12,*) ix1
      read(12,*) ix2
      read(12,*) ix3
      do 100 loop1=1,97
      read(12,*) r(loop1)
100   continue
c ** Check that there are no atoms closer to each other than the
c hard-sphere distance.
c Set warning flag.
      iflast=0
```

```
c Run a loop over all molecules in the pore but the last.
      do 110 loop1=1,inumm1
c Set molecule number flag.
      loopp1=loop1+1
c Store target molecule coordinates.
      xtar01=xpopos(loop1)
      ytar01=ypopos(loop1)
      ztar01=zpopos(loop1)
c Run a loop over all higher-numbered molecules.
      do 120 loop2=loopp1,inumpo
c Store partner molecule coordinates.
      xpar01=xpopos(loop2)
      ypar01=ypopos(loop2)
      zpar01=zpopos(loop2)
c Calculate differences in coordinates of target and partner.
      xdifcc=xtar01-xpar01
      ydifcc=ytar01-ypar01
      zdifcc=ztar01-zpar01
c ** Correct x, y and z for periodic boundary conditions.
c Raise flag if partner is not a minimum image.
c Reset partner molecule coordinates.
      xpar01=xpar01-DSIGN(aucell,xpar01)*
     +(ISIGN(1,IDINT(DABS(xdifcc)/hasidx)-1)+1)/2
      ypar01=ypar01-DSIGN(bucell,ypar01)*
     +(ISIGN(1,IDINT(DABS(ydifcc)/hasidy)-1)+1)/2
      zpar01=zpar01-DSIGN(cucell,zpar01)*
     +(ISIGN(1,IDINT(DABS(zdifcc)/hasidz)-1)+1)/2
c Recalculate differences.
      xdifcc=xtar01-xpar01
      ydifcc=ytar01-ypar01
      zdifcc=ztar01-zpar01
c ** Calculate distance terms.
      dist02=xdifcc*xdifcc+ydifcc*ydifcc+zdifcc*zdifcc
      if(dist02.lt.cutp02) then
      iflag=1
      endif
120   continue
110   continue
c If the flag has been raised, initial configuration is unsuitable.
      if (iflast.gt.0) then
      write(14,2000)
      write(14,2020)
2020  format(' Hard-sphere overlap in starting configuration.')
      write(14,*)
      write(8,*) ' Hard-sphere overlap in starting configuration.'
      endif
      return
      end
c
c
      subroutine lectur
      implicit double precision(a-h,o-z)
c
      dimension en1(200000)
c
      common /cb5/ grid3d(104,28,68),w1(104),w2(28),w3(68),
     +idim1,idim2,idim3
c
```

```
      open(1,file='zsmgrd3.dat')
c Fill the energy grid which is used in the framework-adsorbate
c interaction calculation
      n=0
31     n=n+1
      read(1,*,END=32) en1(n)
      goto 31
32     n=n-1
c     print*,n,' values in big file1'
      idim1=104
      idim2=28
      idim3=68
c     read-in bigfile1
      l=1
      do 177 i=1,104
      do 178 j=1,28
      do 179 k=1,68
      grid3d(i,j,k)=en1(l)
      l=l+1
179    continue
178    continue
177    continue
      return
      end
c
c
c
      subroutine gridy
c
      implicit double precision(a-h,o-z)
      dimension t1(52),t2(14),t3(34)
c
      common /cb5/ grid3d(104,28,68),w1(104),w2(28),w3(68),
     +idim1,idim2,idim3

c
      open(unit=3,file='gauss1.dat')
      open(unit=4,file='gauss2.dat')
      open(unit=7,file='gauss3.dat')
c     open(unit=3,file='gauss96.dat')
c     open(unit=4,file='gauss24.dat')
c     open(unit=7,file='gauss64.dat')
c
      l1=52
      n1=53
      k1=0
45     k1=k1+1
      read(3,*,end=46) t1(k1)
      w1(l1+k1)=t1(k1)/2.0
      w1(n1-k1)=-t1(k1)/2.0
      go to 45
46     k1=k1-1
c     print*,2*k1,' abscissa in gauss1'
      l2=14
      n2=15
      k2=0
47     k2=k2+1
      read(4,*,end=48) t2(k2)
```

```
      w2(l2+k2)=t2(k2)/2.0
      w2(n2-k2)=-t2(k2)/2.0
      go to 47
48     k2=k2-1
c     print*,2*k2,' abscissa in gauss2'
      l3=34
      n3=35
      k3=0
49     k3=k3+1
      read(7,*,end=50) t3(k3)
      w3(l3+k3)=t3(k3)/2.0
      w3(n3-k3)=-t3(k3)/2.0
      go to 49
50     k3=k3-1
c     print*,2*k3,' abscissa in gauss3'
      do 60 i=1,28
      w2(i)=w2(i)*0.25+0.375
60     continue
      return
      end
c
c
c
      subroutine simula
      implicit double precision(a-h,o-z)
c
c
c
c
c
      common /con/ looche,loosum,lootot,rumvol
      common /pot/ siArAr2,epArAr,epArAr4
      common /ran/ ix1,ix2,ix3,r(97)
      common /sta/ cutp01,pressr,sqenet,sqdent
      common /cb1/ xpopos(1000),ypopos(1000),zpopos(1000),
     +ixdist(1000),iydist(1000),izdist(1000),cucell,haspac,
     +aucell,bucell,hasidx,hasidy,hasidz,
     +volpor,recvol,cutp02,inumpo,inumm1,inump1,temper,zedvee,
     +steple,nwaren,nwarmo,aucell,bucell,cucell,numsid
      common /cb2/ nwarlo,nwarhi,ipasno,ipasnn,ipasto,iflamp,
     +iactra,iacdes,iaccre,loopas,runnen,eneave,eruave,
     +denave,druave,widsidx,widsidy,widsidz,twoxpi
      common /cb3/ bolcon,rarega,testen,heatad,
     +enenum,numave,sqnuve,ebenum,tubave,sqbuve
      common /cb4/ runaro,runarar,aroave,ararave,broave,brarave,
     +enaronum,enararnum,ebaronum,ebararnum,heataro,heatArAr,eninte,
     +enarointe,enararinte
      common /cb5/ grid3d(104,28,68),w1(104),w2(28),w3(68),
     +idim1,idim2,idim3
c
c
c **** Main simulation loops.
c
c
      open(unit=15,file='ennit5.tem')
      open(unit=16,file='qsnit5.tem')
      open(unit=8,file='bin.dat')
c Set loop count values.
```

```
      nipasn=ipasno
      nipast=ipasto
c Set warning indicators to zero.
      nwaren=0
      nwarmo=0
      nwarlo=0
      nwarhi=0
c Initialise runnen by calculating the total potential energy.
      call enetot(runnen,runaro,runarar)
c ** Main body of simulation.
      do 10 loop1=1,lootot
c ** Moves.
      do 20 loop2=1,loopas
c ** Molecule translation.
c Select a target molecule.
      random=ran1()
      iflamp=IDINT(random*inumpo)+1
c Call the molecule translation subroutine.
      call patran
c ** Molecule interchange.
c Select creation or destruction.
      random=ran1()
      if(random.lt.0.5D0) then
c Select a target molecule.
      random=ran1()
      iflamp=IDINT(random*inumpo)+1
c If allowed, then call the destruction subroutine.
            if (inumpo.gt.1) then
c           call destry
            else
            nwarlo=nwarlo+1
            endif
      else
c If allowed, then call the creation subroutine.
            if (inumpo.lt.1000) then
c           call create
            else
            nwarhi=nwarhi+1
            endif
      endif
20    continue
c ** Calculations at the end of a pass.
c Calculate the total number of data for averaging.
      nipasn=nipasn+1
      nipast=nipast+1
c Update the singlet distribution functions and the orientation
c functions.
      do 30 loop2=1,inumpo
c
      iflbin=IDINT((xpopos(loop2)+hasidx)/widsidx)+1
      if((iflbin.gt.1000).or.(iflbin.lt.1)) then
      iti=1
      write(8,*) iti,iflbin,loop2,loop1,hasidx,widsidx
      endif
      ixdist(iflbin)=ixdist(iflbin)+1
c
      iflbin=IDINT((ypopos(loop2)+hasidy)/widsidy)+1
      if((iflbin.gt.1000).or.(iflbin.lt.1)) then
```

```
      iti=2
      write(8,*) iti,iflbin,loop2,loop1,hasidy,widsidy
      endif
      iydist(iflbin)=iydist(iflbin)+1
c
      iflbin=IDINT((zpopos(loop2)+hasidz)/widsidz)+1
      if((iflbin.gt.1000).or.(iflbin.lt.1)) then
      iti=3
      write(8,*)'iti,iflbin,loop2,lootot'
      write(8,*) iti,iflbin,loop2,loop1,hasidz,widsidz
      endif
      izdist(iflbin)=izdist(iflbin)+1
c
30    continue
c
c ** Calculate the running averages.
c Energy terms.
c
      eneave=eneave+runnen
      aroave=aroave+runaro
      ararave=ararave+runarar
c
      enenum=enenum+inumpo*runnen
      enaronum=enaronum+inumpo*runaro
      enararnum=enararnum+inumpo*runarar
c
      numave=numave+inumpo
c
      sqnuve=sqnuve+inumpo*inumpo
c
      sqenet=sqenet+runnen*runnen
c
      eruave=eneave/nipasn
      broave=aroave/nipasn
      brarave=ararave/nipasn
c
      ebenum=enenum/nipasn
      ebaronum=enaronum/nipasn
      ebararnum=enararnum/nipasn
c
      tubave=numave*1./nipasn*1.
      sqbuve=sqnuve/nipasn
c Density terms.
      redden=DBLE(inumpo)/volpor
      denave=denave+redden
      sqdent=sqdent+redden*redden
      druave=denave/nipasn
c If the right number of passes has occurred, then write out the
c energy and density terms and their running averages.
      if (MOD(nipast,loosum).eq.0) then
      write(15,2000) nipast,runnen,eruave,tubave,inumpo
      endif
2000  format(i6,1x,f16.4,1x,f16.4,1x,f12.7,1x,i3)
c If the right number of passes has occurrred, then calculate the
c total potential of the system and compare it to the running
c total. If the two are grossly different, add to warnings. Reset
c the running total to the value calculated.
      if (MOD(nipast,looche).eq.0) then
```

```
      call enetot(energy,tenaro,tenarar)
c     print*,'energy from enetot ',energy
c      print*,'runnen              ',runnen
      difrnc=DABS(energy/1.D6)
      absdif=DABS(energy-runnen)
      absdifaro=DABS(tenaro-runaro)
      absdifarar=DABS(tenarar-runarar)
      absdic=absdifaro+absdifarar
c     print*,'absolute drift Ar-O ',absdifaro
c     print*,'absolute drift Ar-Ar',absdifarar
      nwaren=nwaren+IDINT((DSIGN(1.0D0,IDINT(absdif/difrnc)*
     +DABS(difrnc)-1)+1))/2
      runnen=energy
c     eninte=(ebenum-eruave*tubave)/(sqbuve-tubave*tubave)
c     enarointe=(ebaronum-broave*tubave)/(sqbuve-tubave*tubave)
c     enararinte=(ebararnum-brarave*tubave)/(sqbuve-tubave*tubave)
c     heatad=(-eninte+temper)*rarega*1D-3
c     heataro=(-enarointe+temper)*rarega*1D-3
c     heatArAr=(-enararinte)*rarega*1D-3
c     control=heataro+heatArAr
c     print*,'isosteric heat        in Kj/mol            ',heatad
c     print*,'isosteric heat from the sum Ar-Ar and Ar-O',control
c     print*,'value of lootot                            ',loop1
c     print*,'average number of particle in the box      ',tubave
c     print*,'exact   number of particle in the box      ',inumpo
      write(16,2002) nipast,heataro,heatArAr,heatad,tubave,inumpo
2002  format(i6,1x,f16.4,1x,f12.4,1x,f12.4,1x,f12.7,1x,i3)
      endif
10    continue
      return
      end
c
c
c
c
      subroutine result
      implicit double precision(a-h,o-z)
c
c
c
      common /con/ looche,loosum,lootot,rumvol
      common /pot/ siArAr2,epArAr,epArAr4
      common /ran/ ix1,ix2,ix3,r(97)
      common /sta/ cutp01,pressr,sqenet,sqdent
      common /cb1/ xpopos(1000),ypopos(1000),zpopos(1000),
     +ixdist(1000),iydist(1000),izdist(1000),cucell,haspac,
     +aucell,bucell,hasidx,hasidy,hasidz,
     +volpor,recvol,cutp02,inumpo,inumm1,inump1,temper,zedvee,
     +steple,nwaren,nwarmo,aucell,bucell,cucell,numsid
      common /cb2/ nwarlo,nwarhi,ipasno,ipasnn,ipasto,iflamp,
     +iactra,iacdes,iaccre,loopas,runnen,eneave,eruave,
     +denave,druave,widsidx,widsidy,widsidz,twoxpi
      common /cb3/ bolcon,rarega,testen,heatad,
     +enenum,numave,sqnuve,ebenum,tubave,sqbuve
      common /cb4/ runaro,runarar,aroave,ararave,broave,brarave,
     +enaronum,enararnum,ebaronum,ebararnum,heataro,heatArAr,eninte,
     +enarointe,enararinte
      common /cb5/ grid3d(104,28,68),w1(104),w2(28),w3(68),
```

```
     +idim1,idim2,idim3
c
c
c
c
c
c
c **** Write out results.
c
c
c
      open(unit=13,file='conit5.dat')
      open(unit=14,file='annit5.res')
      open(unit=17,file='sinit5.res')
c Write output data.
      if (nwaren.gt.0) then
      write(14,2000)
2000  format(' ** Warning! **')
      write(14,2010)
2010  format(' Recalculated energy disagrees with running energy.')
      write(14,*)
      endif
      if (nwarmo.gt.0) then
      write(14,2000)
      write(14,2020)
2020  format(' Program caught in move loop.')
      write(14,*)
      endif
      if (nwarlo.gt.0) then
      write(14,2000)
      write(14,2030)
2030  format(' Program tries to destroy particles at N=1.')
      write(14,*)
      endif
      if (nwarhi.gt.0) then
      write(14,2000)
      write(14,2040)
2040  format(' Program tries to create particles at N=1000.')
      write(14,*)
      endif
c
      write(14,2050)
2050  format('*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*')
      write(14,2060)
2060  format('*-*-*-* ARGON1  *-*-*-* MAIN RESULTS  *-*-*-*-*-*-*-*')
      write(14,2050)
c
      write(14,2070) nwaren
2070  format(' Number of energy mismatch warnings',12x,i5)
      write(14,2080) nwarmo
2080  format(' Number of loop warnings',23x,i5)
      write(14,2090) nwarlo
2090  format(' Number of destruction warnings',16x,i5)
      write(14,2100) nwarhi
2100  format(' Number of creation warnings',19x,i5)
      write(14,2050)
c
      write(14,2110) temper
```

```
2110  format(' Temperature of the system (K)',17x,f5.1)
      write(14,2050)
c
      write(14,2120) pressr
2120  format(' Specified vapour pressure (Pa)',15x,f5.1)
      write(14,2050)
      write(14,2130) aucell
2130  format(' x unit cell dimensions',24x,f6.3)
      write(14,2135) bucell
2135  format(' y unit cell dimensions',24x,f6.3)
      write(14,2140) cucell
2140  format(' z unit cell dimensions',24x,f6.3)
      write(14,2160) inumpo
2160  format(' Number of argon atoms',25x,i4)
      write(14,2050)
c
      newsno=ipasno+lootot
      write(14,2170) newsno
2170  format(' Number of passes for averages',16x,i6)
      newsto=ipasto+lootot
      write(14,2190) newsto
2190  format(' Number of passes in total',20x,i6)
      write(14,2180) lootot
2180  format(' Number of passes this run',20x,i6)
      write(14,*)
      write(14,2200) loopas
2200  format(' Number of moves of each type per pass',10x,i4)
      write(14,2050)
c
      rederu=eruave/temper
      write(14,2210) rederu
2210  format(' The calculated average pore energy',5x,f10.2)
      ipasqm=newsno*newsno*(newsno-1)
      sdenet=(newsno*sqenet-(eneave*eneave))/ipasqm
      redsde=dsqrt(sdenet)/temper
c
      write(14,2220) heatad
2220  format(' The calculated heat of adsorption in Kj/mol',1x,f7.2)
      write(14,2221) heataro
2221  format('The Argon-Oxygen heat of adsorption in Kj/mol',1x,f7.2)
      write(14,2222) heatArAr
2222  format('The Argon-Argon  heat of adsorption in Kj/mol',1x,f7.2)
      write(14,2240) redsde
2240  format(' Estimated error for the averaged pore energy',1x,f8.3)
      write(14,*)
      write(14,2250) cutp01
2250  format(' The spherical potential Ar-Ar cutoff',9x,f5.2)
      write(14,2060)
c
      write(14,2260) iactra
2260  format(' Number of translation acceptances',12x,i6)
      ipaloo=lootot*loopas
      tranac=DBLE(iactra)/ipaloo
      write(14,2270) tranac
2270  format(' Translation move acceptance rate',12x,f7.5)
      write(14,*)
      write(14,2290) iacdes
2290  format(' Number of destruction acceptances',12x,i6)
```

```
      destac=DBLE(iacdes)/ipaloo
      write(14,2300) destac
2300  format(' Destruction acceptance rate',17x,f7.5)
      write(14,*)
      write(14,2310) iaccre
2310  format(' Number of creation acceptances',15x,i6)
      cretac=DBLE(iaccre)/ipaloo
      write(14,2320) cretac
2320  format(' Creation acceptance rate',20x,f7.5)
      write(14,2050)
c
      redden=DBLE(inumpo)/volpor
      write(14,2330) redden
2330  format(' Current density',29x,f7.5)
      write(14,2340) druave
2340  format(' Average density in the pore',17x,f7.5)
      sdedet=(newsno*sqdent-(denave*denave))/ipasqm
      sdedet=dsqrt(sdedet)
      write(14,2350) sdedet
2350  format(' Estimated error',29x,f7.5)
      write(14,2050)
      write(14,2060)
      write(14,2050)
c Write restart record.
      write(13,*) 0
      write(13,*) aucell
      write(13,*) bucell
      write(13,*) cucell
      write(13,*) steple
      write(13,*) eneave
      write(13,*) aroave
      write(13,*) ararave
      write(13,*) enenum
      write(13,*) enaronum
      write(13,*) enararnum
      write(13,*) numave
      write(13,*) sqnuve
      write(13,*) sqenet
      write(13,*) denave
      write(13,*) sqdent
      write(13,*) newsno
      write(13,*) newsto
      write(13,*) inumpo
      do 10 loop1=1,inumpo
      write(13,*) xpopos(loop1)
      write(13,*) ypopos(loop1)
      write(13,*) zpopos(loop1)
10    continue
      do 20 loop1=1,1000
      write(13,*) ixdist(loop1)
      write(13,*) iydist(loop1)
      write(13,*) izdist(loop1)
20    continue
      write(13,*) ix1
      write(13,*) ix2
      write(13,*) ix3
      do 50 loop1=1,97
      write(13,*) r(loop1)
```

```
50    continue
c Write singlet distribution functions.
      ipavol=newsno*volpor
      numsid=1000
      fousidx=DBLE(numsid)/ipavol
      fousidy=DBLE(numsid)/ipavol
      fousidz=DBLE(numsid)/ipavol
      addsidx=DBLE(numsid+1)/2
      addsidy=DBLE(numsid+1)/2
      addsidz=DBLE(numsid+1)/2
      do 60 loop1=1,1000
      disisx=fousidx*ixdist(loop1)
      disisy=fousidy*iydist(loop1)
      disisz=fousidz*izdist(loop1)
      sigsidx=(ISIGN(1,(numsid/loop1)-1)+1)/2*(loop1-addsidx)*widsidx
      sigsidy=(ISIGN(1,(numsid/loop1)-1)+1)/2*(loop1-addsidy)*widsidy
      sigsidz=(ISIGN(1,(numsid/loop1)-1)+1)/2*(loop1-addsidz)*widsidz
      write(17,2360) sigsidx,sigsidy,sigsidz,disisx,disisy,disisz
60    continue
2360  format(f14.11,1x,f14.11,1x,f14.11,1x,f14.11,1x,f14.11,1x,f14.11)
c     write(*,*)'normal end of msimu6.f'
      return
      end
c
c
c
c
      subroutine enetot(poten1,tesaro,tesarar)
      implicit double precision(a-h,o-z)
c
c
c
c
c
      common /con/ looche,loosum,lootot,rumvol
      common /pot/ siArAr2,epArAr,epArAr4
      common /ran/ ix1,ix2,ix3,r(97)
      common /sta/ cutp01,pressr,sqenet,sqdent
      common /cb1/ xpopos(1000),ypopos(1000),zpopos(1000),
     +ixdist(1000),iydist(1000),izdist(1000),cucell,haspac,
     +aucell,bucell,hasidx,hasidy,hasidz,
     +volpor,recvol,cutp02,inumpo,inumm1,inump1,temper,zedvee,
     +steple,nwaren,nwarmo,aucell,bucell,cucell,numsid
      common /cb2/ nwarlo,nwarhi,ipasno,ipasnn,ipasto,iflamp,
     +iactra,iacdes,iaccre,loopas,runnen,eneave,eruave,
     +denave,druave,widsidx,widsidy,widsidz,twoxpi
      common /cb3/ bolcon,rarega,testen,heatad,
     +enenum,numave,sqnuve,ebenum,tubave,sqbuve
      common /cb4/ runaro,runarar,aroave,ararave,broave,brarave,
     +enaronum,enararnum,ebaronum,ebararnum,heataro,heatArAr,eninte,
     +enarointe,enararinte
      common /cb5/ grid3d(104,28,68),w1(104),w2(28),w3(68),
     +idim1,idim2,idim3
c
c
c
c
c **** Calculate total interaction.
```

```
c
c
c
c Reset energy total to zero.
      poten1=0.0D0
      tesaro=0.0D0
      tesarar=0.0D0
c Run a loop over all molecules in the pore but the last.
      do 10 loop1=1,inumm1
c Set molecule number flag.
      loopp1=loop1+1
c Store target molecule coordinates.
      xtar01=xpopos(loop1)
      ytar01=ypopos(loop1)
      ztar01=zpopos(loop1)
c Run a loop over all higher-numbered molecules.
      do 20 loop2=loopp1,inumpo
c Store partner molecule coordinates.
      xpar01=xpopos(loop2)
      ypar01=ypopos(loop2)
      zpar01=zpopos(loop2)
c Calculate differences in coordinates of target and partner.
      xdifcc=xtar01-xpar01
      ydifcc=ytar01-ypar01
      zdifcc=ztar01-zpar01
c ** Correct x, y and z for periodic boundary conditions.
c Raise flag if partner is not a minimum image.
c Reset partner molecule coordinates.
      xpar01=xpar01-DSIGN(aucell,xpar01)*
     +(ISIGN(1,IDINT(DABS(xdifcc)/hasidx)-1)+1)/2
      ypar01=ypar01-DSIGN(bucell,ypar01)*
     +(ISIGN(1,IDINT(DABS(ydifcc)/hasidy)-1)+1)/2
      zpar01=zpar01-DSIGN(cucell,zpar01)*
     +(ISIGN(1,IDINT(DABS(zdifcc)/hasidz)-1)+1)/2
c Recalculate differences.
      xdifcc=xtar01-xpar01
      ydifcc=ytar01-ypar01
      zdifcc=ztar01-zpar01
c Calculate distance between molecule centres.
      dist02=xdifcc*xdifcc+ydifcc*ydifcc+zdifcc*zdifcc
c Raise flag if target and partner are within the potential cutoff.
      iflacu=(ISIGN(1,IDINT(cutp02/dist02)-1)+1)/2
c ** Calculate the particle-particle dispersion repulsion interaction.
      atArAr=(siArAr2/dist02)
      atArAr3=atArAr*atArAr*atArAr
      reArAr=atArAr3*atArAr3
      energ1=iflacu*epArAr4*(reArAr-atArAr3)
      tesarar=tesarar+energ1
20    continue
c ** Calculate the wall-particle dispersion repulsion interaction.
      xredar=xtar01/aucell
      yredar=ytar01/bucell
      zredar=ztar01/cucell
      if((DABS(xredar).le.1.5D0).and.(DABS(xredar).gt.0.5D0)) then
          if(xredar.gt.0.0D0) then
          xredar=xredar-1.0D0
          else
          xredar=xredar+1.0D0
```

```
        endif
      endif
      if((DABS(yredar).le.1.5D0).and.(DABS(yredar).gt.0.5D0)) then
        if(yredar.gt.0.0D0) then
        yredar=yredar-1.0D0
        else
        yredar=yredar+1.0D0
        endif
      endif
      if((DABS(zredar).le.1.5D0).and.(DABS(zredar).gt.0.5D0)) then
        if(zredar.gt.0.0D0) then
        zredar=zredar-1.0D0
        else
        zredar=zredar+1.0D0
        endif
      endif
      if((yredar.ge.-0.5D0).and.(yredar.lt.-0.25D0)) then
        xredar=-xredar
        yredar=-yredar
        zredar=-zredar
      endif
      if((yredar.ge.-0.25D0).and.(yredar.lt.0.0D0)) then
        xredar=-xredar
        yredar=yredar+0.5D0
        zredar=-zredar
      endif
      if((yredar.ge.0.0D0).and.(yredar.lt.0.25D0)) then
         xredar=xredar
         yredar=-yredar+0.5D0
         zredar=zredar
      endif
      call lininter (xredar,yredar,zredar,pesaro)
      tesaro=tesaro+pesaro
10    continue
c Store last target molecule coordinates.
      xtar01=xpopos(inumpo)
      ytar01=ypopos(inumpo)
      ztar01=zpopos(inumpo)
c ** Calculate the wall-particle dispersion interaction for the
c last particle.
      xredar=xtar01/aucell
      yredar=ytar01/bucell
      zredar=ztar01/cucell
      if((DABS(xredar).le.1.5D0).and.(DABS(xredar).gt.0.5D0)) then
        if(xredar.gt.0.0D0) then
        xredar=xredar-1.0D0
        else
        xredar=xredar+1.0D0
        endif
      endif
      if((DABS(yredar).le.1.5D0).and.(DABS(yredar).gt.0.5D0)) then
        if(yredar.gt.0.0D0) then
        yredar=yredar-1.0D0
        else
        yredar=yredar+1.0D0
        endif
      endif
      if((DABS(zredar).le.1.5D0).and.(DABS(zredar).gt.0.5D0)) then
```

```
      if(zredar.gt.0.0D0) then
      zredar=zredar-1.0D0
      else
      zredar=zredar+1.0D0
      endif
    endif
    if((yredar.ge.-0.5D0).and.(yredar.lt.-0.25D0)) then
      xredar=-xredar
      yredar=-yredar
      zredar=-zredar
    endif
    if((yredar.ge.-0.25D0).and.(yredar.lt.0.0D0)) then
      xredar=-xredar
      yredar=yredar+0.5D0
      zredar=-zredar
    endif
    if((yredar.ge.0.0D0).and.(yredar.lt.0.25D0)) then
       xredar=xredar
       yredar=-yredar+0.5D0
       zredar=zredar
    endif
    call lininter (xredar,yredar,zredar,energy)
    tesaro=tesaro+energy
    poten1=tesarar+tesaro
    return
    end
c
c
c
c
    subroutine patran
    implicit double precision(a-h,o-z)
c
c
c
c
    dimension temden(10)
c
c
    common /con/ looche,loosum,lootot,rumvol
    common /pot/ siArAr2,epArAr,epArAr4
    common /ran/ ix1,ix2,ix3,r(97)
    common /sta/ cutp01,pressr,sqenet,sqdent
    common /cb1/ xpopos(1000),ypopos(1000),zpopos(1000),
   +ixdist(1000),iydist(1000),izdist(1000),cucell,haspac,
   +aucell,bucell,hasidx,hasidy,hasidz,
   +volpor,recvol,cutp02,inumpo,inumm1,inump1,temper,zedvee,
   +steple,nwaren,nwarmo,aucell,bucell,cucell,numsid
    common /cb2/ nwarlo,nwarhi,ipasno,ipasnn,ipasto,iflamp,
   +iactra,iacdes,iaccre,loopas,runnen,eneave,eruave,
   +denave,druave,widsidx,widsidy,widsidz,twoxpi
    common /cb3/ bolcon,rarega,testen,heatad,
   +enenum,numave,sqnuve,ebenum,tubave,sqbuve
    common /cb4/ runaro,runarar,aroave,ararave,broave,brarave,
   +enaronum,enararnum,ebaronum,ebararnum,heataro,heatArAr,eninte,
   +enarointe,enararinte
    common /cb5/ grid3d(104,28,68),w1(104),w2(28),w3(68),
   +idim1,idim2,idim3
```

```
c
c
c
c
c
c **** Calculation for molecule translations.
c
c
c
c Reset energy terms.
      poten1=0.0D0
      enaro1=0.0D0
      enarar1=0.0D0
      enaro2=0.0D0
      enarar2=0.0D0
      poten2=0.0D0
c ** Calculations before move.
c Store target molecule coordinates.
      xtar01=xpopos(iflamp)
      ytar01=ypopos(iflamp)
      ztar01=zpopos(iflamp)
c ** Calculate the wall-particle dispersion repulsion interaction.
      xredar=xtar01/aucell
      yredar=ytar01/bucell
      zredar=ztar01/cucell
      if((DABS(xredar).le.1.5D0).and.(DABS(xredar).gt.0.5D0)) then
          if(xredar.gt.0.0D0) then
          xredar=xredar-1.0D0
          else
          xredar=xredar+1.0D0
          endif
      endif
      if((DABS(yredar).le.1.5D0).and.(DABS(yredar).gt.0.5D0)) then
          if(yredar.gt.0.0D0) then
          yredar=yredar-1.0D0
          else
          yredar=yredar+1.0D0
          endif
      endif
      if((DABS(zredar).le.1.5D0).and.(DABS(zredar).gt.0.5D0)) then
          if(zredar.gt.0.0D0) then
          zredar=zredar-1.0D0
          else
          zredar=zredar+1.0D0
          endif
      endif
      if((yredar.ge.-0.5D0).and.(yredar.lt.-0.25D0)) then
          xredar=-xredar
          yredar=-yredar
          zredar=-zredar
      endif
      if((yredar.ge.-0.25D0).and.(yredar.lt.0.0D0)) then
          xredar=-xredar
          yredar=yredar+0.5D0
          zredar=-zredar
      endif
      if((yredar.ge.0.0D0).and.(yredar.lt.0.25D0)) then
           xredar=xredar
```

```
          yredar=-yredar+0.5D0
          zredar=zredar
      endif
      call lininter (xredar,yredar,zredar,poten1)
      enaro1=poten1
c ** Find neighbour distances of target molecule.
c Run a loop over all molecules in the pore.
      do 15 loop1=1,inumpo
c Raise flag if target and partner are different molecules.
      ifladi=(ISIGN(1,ABS(loop1-iflamp)-1)+1)/2
      samfla=(1-ifladi)*cutp02
c Store partner molecule coordinates.
      xpar01=xpopos(loop1)+samfla
      ypar01=ypopos(loop1)+samfla
      zpar01=zpopos(loop1)+samfla
c Calculate differences in coordinates of target and partner.
      xdifcc=xtar01-xpar01
      ydifcc=ytar01-ypar01
      zdifcc=ztar01-zpar01
c ** Correct x, y and for periodic boundary conditions.
c Raise flag if partner is not a minimum image.
c Reset partner molecule coordinates
      xpar01=xpar01-DSIGN(aucell,xpar01)*
     +(ISIGN(1,IDINT(DABS(xdifcc)/hasidx)-1)+1)/2
      ypar01=ypar01-DSIGN(bucell,ypar01)*
     +(ISIGN(1,IDINT(DABS(ydifcc)/hasidy)-1)+1)/2
      zpar01=zpar01-DSIGN(cucell,zpar01)*
     +(ISIGN(1,IDINT(DABS(zdifcc)/hasidz)-1)+1)/2
c Recalculate differences.
      xdifcc=xtar01-xpar01
      ydifcc=ytar01-ypar01
      zdifcc=ztar01-zpar01
c Calculate distance between molecule centres.
      dist02=xdifcc*xdifcc+ydifcc*ydifcc+zdifcc*zdifcc
c Raise flag if target and partner are within the potential cutoff
      iflacu=(ISIGN(1,IDINT(cutp02/dist02)-1)+1)/2
c ** Calculate distance terms for dispersion repulsion energy.
      atArAr=(siArAr2/dist02)
      atArAr3=atArAr*atArAr*atArAr
      reArAr=atArAr3*atArAr3
      enarar1=enarar1+iflacu*ifladi*epArAr4*(reArAr-atArAr3)
      poten1=poten1+iflacu*ifladi*epArAr4*(reArAr-atArAr3)
15    continue
c ** Calculations for move.
c Select coordinates of proposed site.
      random=ran1()
      xtar01=xpopos(iflamp)+(random-0.5)*steple
      random=ran1()
      ytar01=ypopos(iflamp)+(random-0.5)*steple
      random=ran1()
      ztar01=zpopos(iflamp)+(random-0.5)*steple
c ** Correct x, y and z for periodic boundary conditions.
      xtar01=xtar01-DSIGN(aucell,xtar01)*
     +(ISIGN(1,IDINT(DABS(xtar01)/hasidx)-1)+1)/2
      ytar01=ytar01-DSIGN(bucell,ytar01)*
     +(ISIGN(1,IDINT(DABS(ytar01)/hasidy)-1)+1)/2
      ztar01=ztar01-DSIGN(cucell,ztar01)*
     +(ISIGN(1,IDINT(DABS(ztar01)/hasidz)-1)+1)/2
```

```
c ** Calculations after move.
c ** Calculate the wall-particle dispersion interaction.
      xredar=xtar01/aucell
      yredar=ytar01/bucell
      zredar=ztar01/cucell
      if((DABS(xredar).le.1.5D0).and.(DABS(xredar).gt.0.5D0)) then
          if(xredar.gt.0.0D0) then
          xredar=xredar-1.0D0
          else
          xredar=xredar+1.0D0
          endif
      endif
      if((DABS(yredar).le.1.5D0).and.(DABS(yredar).gt.0.5D0)) then
          if(yredar.gt.0.0D0) then
          yredar=yredar-1.0D0
          else
          yredar=yredar+1.0D0
          endif
      endif
      if((DABS(zredar).le.1.5D0).and.(DABS(zredar).gt.0.5D0)) then
          if(zredar.gt.0.0D0) then
          zredar=zredar-1.0D0
          else
          zredar=zredar+1.0D0
          endif
      endif
      if((yredar.ge.-0.5D0).and.(yredar.lt.-0.25D0)) then
          xredar=-xredar
          yredar=-yredar
          zredar=-zredar
      endif
      if((yredar.ge.-0.25D0).and.(yredar.lt.0.0D0)) then
          xredar=-xredar
          yredar=yredar+0.5D0
          zredar=-zredar
      endif
      if((yredar.ge.0.0D0).and.(yredar.lt.0.25D0)) then
           xredar=xredar
           yredar=-yredar+0.5D0
           zredar=zredar
      endif
      call lininter (xredar,yredar,zredar,poten2)
      enaro2=poten2
      if(poten2.gt.testen) then
      nwarmo=nwarmo+1
      return
      endif
c Run a loop over all molecules in the pore.
      do 35 loop1=1,inumpo
c Raise flag if target and partner are different molecules.
      ifladi=(ISIGN(1,ABS(loop1-iflamp)-1)+1)/2
      samfla=(1-ifladi)*cutp02
c Store partner molecule coordinates.
      xpar01=xpopos(loop1)+samfla
      ypar01=ypopos(loop1)+samfla
      zpar01=zpopos(loop1)+samfla
c Calculate differences in coordinates of target and partner.
      xdifcc=xtar01-xpar01
```

```
      ydifcc=ytar01-ypar01
      zdifcc=ztar01-zpar01
c ** Correct x, y and for periodic boundary conditions.
c Raise flag if partner is not a minimum image.
c Reset partner molecule coordinates.
      xpar01=xpar01-DSIGN(aucell,xpar01)*
     +(ISIGN(1,IDINT(DABS(xdifcc)/hasidx)-1)+1)/2
      ypar01=ypar01-DSIGN(bucell,ypar01)*
     +(ISIGN(1,IDINT(DABS(ydifcc)/hasidy)-1)+1)/2
      zpar01=zpar01-DSIGN(cucell,zpar01)*
     +(ISIGN(1,IDINT(DABS(zdifcc)/hasidz)-1)+1)/2
c Recalculate differences.
      xdifcc=xtar01-xpar01
      ydifcc=ytar01-ypar01
      zdifcc=ztar01-zpar01
c Calculate distance between molecule centres.
      dist02=xdifcc*xdifcc+ydifcc*ydifcc+zdifcc*zdifcc
c Raise flag if target and partner are within the potential cutoff.
      iflacu=(ISIGN(1,IDINT(cutp02/dist02)-1)+1)/2
c ** Calculate distance terms for dispersion repulsion energy.
      atArAr=(siArAr2/dist02)
      atArAr3=atArAr*atArAr*atArAr
      reArAr=atArAr3*atArAr3
      enarar2=enarar2+iflacu*ifladi*epArAr4*(reArAr-atArAr3)
      poten2=poten2+iflacu*ifladi*epArAr4*(reArAr-atArAr3)
35    continue
c Calculate the energy change.
      enerch=poten2-poten1
      arocha=enaro2-enaro1
      ararcha=enarar2-enarar1
c Calculate the energy change scaled to kt.
      scench=(enerch)/temper
c Calculate the probability of success.
      if (scench.gt.13.0D0) then
      probab=0
      elseif (scench.le.0.0D0) then
      probab=14.0D0
      else
      probab=dexp(-scench)
      endif
c If the move is successful, update the coordinates, running energy
c and number of acceptances.
      random=ran1()
      if (random.lt.probab) then
      xpopos(iflamp)=xtar01
      ypopos(iflamp)=ytar01
      zpopos(iflamp)=ztar01
      runnen=runnen+enerch
      runaro=runaro+arocha
      runarar=runarar+ararcha
      iactra=iactra+1
      endif
      return
      end
c
c
c
c
```

```
      subroutine destry
      implicit double precision(a-h,o-z)
c
c
c
      common /con/ looche,loosum,lootot,rumvol
      common /pot/ siArAr2,epArAr,epArAr4
      common /ran/ ix1,ix2,ix3,r(97)
      common /sta/ cutp01,pressr,sqenet,sqdent
      common /cb1/ xpopos(1000),ypopos(1000),zpopos(1000),
     +ixdist(1000),iydist(1000),izdist(1000),cucell,haspac,
     +aucell,bucell,hasidx,hasidy,hasidz,
     +volpor,recvol,cutp02,inumpo,inumm1,inump1,temper,zedvee,
     +steple,nwaren,nwarmo,aucell,bucell,cucell,numsid
      common /cb2/ nwarlo,nwarhi,ipasno,ipasnn,ipasto,iflamp,
     +iactra,iacdes,iaccre,loopas,runnen,eneave,eruave,
     +denave,druave,widsidx,widsidy,widsidz,twoxpi
      common /cb3/ bolcon,rarega,testen,heatad,
     +enenum,numave,sqnuve,ebenum,tubave,sqbuve
      common /cb4/ runaro,runarar,aroave,ararave,broave,brarave,
     +enaronum,enararnum,ebaronum,ebararnum,heataro,heatArAr,eninte,
     +enarointe,enararinte
      common /cb5/ grid3d(104,28,68),w1(104),w2(28),w3(68),
     +idim1,idim2,idim3
c
c
c
c
c
c **** Calculation for molecule destruction.
c
c
c
c Reset energy terms.
      poten1=0.0D0
      potaro=0.0D0
      potarar=0.0D0
c ** Calculations before move.
c Store target molecule coordinates.
      xtar01=xpopos(iflamp)
      ytar01=ypopos(iflamp)
      ztar01=zpopos(iflamp)
c ** Find neighbour distances of target molecule.
c Run a loop over all molecules in the pore.
      do 10 loop1=1,inumpo
c Raise flag if target and partner are different molecules.
      ifladi=(ISIGN(1,ABS(loop1-iflamp)-1)+1)/2
      samfla=(1-ifladi)*cutp02
c Store partner molecule coordinates.
      xpar01=xpopos(loop1)+samfla
      ypar01=ypopos(loop1)+samfla
      zpar01=zpopos(loop1)+samfla
c Calculate differences in coordinates of target and partner.
      xdifcc=xtar01-xpar01
      ydifcc=ytar01-ypar01
      zdifcc=ztar01-zpar01
c ** Correct x, y and for periodic boundary conditions.
c Raise flag if partner is not a minimum image.
```

```
c Reset partner molecule coordinates.
      xpar01=xpar01-DSIGN(aucell,xpar01)*
     +(ISIGN(1,IDINT(DABS(xdifcc)/hasidx)-1)+1)/2
      ypar01=ypar01-DSIGN(bucell,ypar01)*
     +(ISIGN(1,IDINT(DABS(ydifcc)/hasidy)-1)+1)/2
      zpar01=zpar01-DSIGN(cucell,zpar01)*
     +(ISIGN(1,IDINT(DABS(zdifcc)/hasidz)-1)+1)/2
c Recalculate differences.
      xdifcc=xtar01-xpar01
      ydifcc=ytar01-ypar01
      zdifcc=ztar01-zpar01
c Calculate distance between molecule centres.
      dist02=xdifcc*xdifcc+ydifcc*ydifcc+zdifcc*zdifcc
c Raise flag if target and partner are within the potential cutoff.
      iflacu=(ISIGN(1,IDINT(cutp02/dist02)-1)+1)/2
c ** Calculate distance terms for dispersion repulsion energy.
      atArAr=(siArAr2/dist02)
      atArAr3=atArAr*atArAr*atArAr
      reArAr=atArAr3*atArAr3
      quanta=iflacu*ifladi*epArAr4*(reArAr-atArAr3)
      potarar=potarar+quanta
10    continue
c ** Calculate the wall-particle dispersion repulsion interaction.
      xredar=xtar01/aucell
      yredar=ytar01/bucell
      zredar=ztar01/cucell
      if((DABS(xredar).le.1.5D0).and.(DABS(xredar).gt.0.5D0)) then
          if(xredar.gt.0.0D0) then
          xredar=xredar-1.0D0
          else
          xredar=xredar+1.0D0
          endif
      endif
      if((DABS(yredar).le.1.5D0).and.(DABS(yredar).gt.0.5D0)) then
          if(yredar.gt.0.0D0) then
          yredar=yredar-1.0D0
          else
          yredar=yredar+1.0D0
          endif
      endif
      if((DABS(zredar).le.1.5D0).and.(DABS(zredar).gt.0.5D0)) then
          if(zredar.gt.0.0D0) then
          zredar=zredar-1.0D0
          else
          zredar=zredar+1.0D0
          endif
      endif
      if((yredar.ge.-0.5D0).and.(yredar.lt.-0.25D0)) then
          xredar=-xredar
          yredar=-yredar
          zredar=-zredar
      endif
      if((yredar.ge.-0.25D0).and.(yredar.lt.0.0D0)) then
          xredar=-xredar
          yredar=yredar+0.5D0
          zredar=-zredar
      endif
      if((yredar.ge.0.0D0).and.(yredar.lt.0.25D0)) then
```

```
          xredar=xredar
          yredar=-yredar+0.5D0
          zredar=zredar
      endif
      call lininter (xredar,yredar,zredar,potaro)
c Calculate the energy change.
      poten1=potarar+potaro
      enerch=-poten1
      encharo=-potaro
      encharar=-potarar
c Calculate the energy change scaled to kt.
      scench=enerch/temper
c Calculate the chemical potential term.
      cpterm=DBLE(inumpo)/zedvee
c Calculate the probability of success.
      expluc=dlog(cpterm)-scench
      if (expluc.lt.-100.0D0) then
      probab=0
      else if (expluc.gt.0.0D0) then
      probab=1.0D0
      else
      probab=dexp(expluc)
      endif
c If the move is successful, update the coordinates, running energy
c and number of acceptances.
      random=ran1()
      if (random.lt.probab) then
      do 30 loop1=iflamp,inumm1
      loopp1=loop1+1
      xpopos(loop1)=xpopos(loopp1)
      ypopos(loop1)=ypopos(loopp1)
      zpopos(loop1)=zpopos(loopp1)
30    continue
      inump1=inumpo
      inumpo=inumm1
      inumm1=inumm1-1
      runnen=runnen+enerch
      runaro=runaro+encharo
      runarar=runarar+encharar
      iacdes=iacdes+1
      endif
      return
      end
c
c
c
c
      subroutine create
      implicit double precision(a-h,o-z)
c
c
c
c
      dimension iflama(9)
c
      common /con/ looche,loosum,lootot,rumvol
      common /pot/ siArAr2,epArAr,epArAr4
      common /ran/ ix1,ix2,ix3,r(97)
```

```
      common /sta/ cutp01,pressr,sqenet,sqdent
      common /cb1/ xpopos(1000),ypopos(1000),zpopos(1000),
     +ixdist(1000),iydist(1000),izdist(1000),cucell,haspac,
     +aucell,bucell,hasidx,hasidy,hasidz,
     +volpor,recvol,cutp02,inumpo,inumm1,inump1,temper,zedvee,
     +steple,nwaren,nwarmo,aucell,bucell,cucell,numsid
      common /cb2/ nwarlo,nwarhi,ipasno,ipasnn,ipasto,iflamp,
     +iactra,iacdes,iaccre,loopas,runnen,eneave,eruave,
     +denave,druave,widsidx,widsidy,widsidz,twoxpi
      common /cb3/ bolcon,rarega,testen,heatad,
     +enenum,numave,sqnuve,ebenum,tubave,sqbuve
      common /cb4/ runaro,runarar,aroave,ararave,broave,brarave,
     +enaronum,enararnum,ebaronum,ebararnum,heataro,heatArAr,eninte,
     +enarointe,enararinte
      common /cb5/ grid3d(104,28,68),w1(104),w2(28),w3(68),
     +idim1,idim2,idim3
c
c
c
c
c
c **** Calculation for molecule creation.
c
c
c
c Reset energy term.
      poten2=0.0D0
      enaro2=0.0D0
      enarar2=0.0D0
c ** Select coordinates.
c Select coordinates.
      random=ran1()
      xtar01=(random-0.5D0)*aucell
      random=ran1()
      ytar01=(random-0.5D0)*bucell
      random=ran1()
      ztar01=(random-0.5D0)*cucell
c ** Calculations after move.
c ** Calculate the wall-particle dispersion interaction.
      xredar=xtar01/aucell
      yredar=ytar01/bucell
      zredar=ztar01/cucell
      if((DABS(xredar).le.1.5D0).and.(DABS(xredar).gt.0.5D0)) then
          if(xredar.gt.0.0D0) then
          xredar=xredar-1.0D0
          else
          xredar=xredar+1.0D0
          endif
      endif
      if((DABS(yredar).le.1.5D0).and.(DABS(yredar).gt.0.5D0)) then
          if(yredar.gt.0.0D0) then
          yredar=yredar-1.0D0
          else
          yredar=yredar+1.0D0
          endif
      endif
      if((DABS(zredar).le.1.5D0).and.(DABS(zredar).gt.0.5D0)) then
          if(zredar.gt.0.0D0) then
```

```
        zredar=zredar-1.0D0
        else
        zredar=zredar+1.0D0
        endif
    endif
    if((yredar.ge.-0.5D0).and.(yredar.lt.-0.25D0)) then
        xredar=-xredar
        yredar=-yredar
        zredar=-zredar
    endif
    if((yredar.ge.-0.25D0).and.(yredar.lt.0.0D0)) then
        xredar=-xredar
        yredar=yredar+0.5D0
        zredar=-zredar
    endif
    if((yredar.ge.0.0D0).and.(yredar.lt.0.25D0)) then
         xredar=xredar
         yredar=-yredar+0.5D0
         zredar=zredar
    endif
    call lininter (xredar,yredar,zredar,poten2)
    enaro2=poten2
c Run a loop over all molecules in the pore.
    do 30 loop1=1,inumpo
c Store partner molecule coordinates.
    xpar01=xpopos(loop1)
    ypar01=ypopos(loop1)
    zpar01=zpopos(loop1)
c Calculate differences in coordinates of target and partner.
    xdifcc=xtar01-xpar01
    ydifcc=ytar01-ypar01
    zdifcc=ztar01-zpar01
c ** Correct x, y and z for periodic boundary conditions.
c Raise flag if partner is not a minimum image.
c Reset partner molecule coordinates.
    xpar01=xpar01-DSIGN(aucell,xpar01)*
   +(ISIGN(1,IDINT(DABS(xdifcc)/hasidx)-1)+1)/2
    ypar01=ypar01-DSIGN(bucell,ypar01)*
   +(ISIGN(1,IDINT(DABS(ydifcc)/hasidy)-1)+1)/2
    zpar01=zpar01-DSIGN(cucell,zpar01)*
   +(ISIGN(1,IDINT(DABS(zdifcc)/hasidz)-1)+1)/2
c Recalculate differences.
    xdifcc=xtar01-xpar01
    ydifcc=ytar01-ypar01
    zdifcc=ztar01-zpar01
c Calculate distance between molecule centres.
    dist02=xdifcc*xdifcc+ydifcc*ydifcc+zdifcc*zdifcc
c Raise flag if target and partner are within the potential cutoff.
    iflacu=(ISIGN(1,IDINT(cutp02/dist02)-1)+1)/2
c ** Calculate distance terms for dispersion energy.
    atArAr=(siArAr2/dist02)
    atArAr3=atArAr*atArAr*atArAr
    reArAr=atArAr3*atArAr3
    poten2=poten2+iflacu*epArAr4*(reArAr-atArAr3)
    enarar2=enarar2+iflacu*epArAr4*(reArAr-atArAr3)
30   continue
c Calculate the energy change.
    enerch=poten2
```

```
      encharo=enaro2
      encharar=enarar2
c Calculate the energy change scaled to kt .
      scench=(enerch)/temper
c Calculate the chemical potential term.
      cpterm=zedvee/DBLE(inump1)
c Calculate the probability of success.
      expluc=dlog(cpterm)-scench
      if (expluc.lt.-100.0D0) then
      probab=0.0D0
      else if (expluc.gt.0D0) then
      probab=1.0D0
      else
      probab=dexp(expluc)
      endif
c if the move is successful, update the coordinates, running energy
c and number of acceptances.
      random=ran1()
      if (random.lt.probab) then
      xpopos(inump1)=xtar01
      ypopos(inump1)=ytar01
      zpopos(inump1)=ztar01
      if((xtar01.gt.hasidx).or.(xtar01.lt.-hasidx)) then
      it=1
      write(8,*) 'in create',it,xtar01,inump1
      endif
      if((ytar01.gt.hasidy).or.(ytar01.lt.-hasidy)) then
      it=2
      write(8,*) 'in create',it,ytar01,inump1
      endif
      if((ztar01.gt.hasidz).or.(ztar01.lt.-hasidz)) then
      it=3
      write(8,*) 'in create',it,ztar01,inump1
      endif
      inumm1=inumpo
      inumpo=inump1
      inump1=inump1+1
      runnen=runnen+enerch
      runaro=runaro+encharo
      runarar=runarar+encharar
      iaccre=iaccre+1
      iflamp=inumpo
      endif
      return
      end
c
c
c
c
      function ran1()
c
      implicit double precision(a-h,o-z)
c
c
c
      parameter (m1=259200,ia1=7141,ic1=54773,rm1=3.8580247D-6)
      parameter (m2=134456,ia2=8121,ic2=28411,rm2=7.4373773D-6)
      parameter (m3=243000,ia3=4561,ic3=51349)
```

```
      common /ran/ ix1,ix2,ix3,r(97)
c
c
c
c **** Function from Numerical Recipes

      ix1=MOD(ia1*ix1+ic1,m1)
      ix2=MOD(ia2*ix2+ic2,m2)
      ix3=MOD(ia3*ix3+ic3,m3)
      j=1+(97*ix3)/m3
      if((j.gt.97).or.(j.lt.1)) then
      endif
      ran1=r(j)
      r(j)=(dfloat(ix1)+dfloat(ix2)*rm2)*rm1
      return
      end
c
c
c
c
c
      subroutine lininter(x,y,z,value)
      implicit double precision(a-h,o-z)
c
          dimension xneigh(8),yneigh(8),zneigh(8),eneigh(8)

      common /con/ looche,loosum,lootot,rumvol
      common /pot/ siArAr2,epArAr,epArAr4
      common /ran/ ix1,ix2,ix3,r(97)
      common /sta/ cutp01,pressr,sqenet,sqdent
      common /cb1/ xpopos(1000),ypopos(1000),zpopos(1000),
     +ixdist(1000),iydist(1000),izdist(1000),cucell,haspac,
     +aucell,bucell,hasidx,hasidy,hasidz,
     +volpor,recvol,cutp02,inumpo,inumm1,inump1,temper,zedvee,
     +steple,nwaren,nwarmo,aucell,bucell,cucell,numsid
      common /cb2/ nwarlo,nwarhi,ipasno,ipasnn,ipasto,iflamp,
     +iactra,iacdes,iaccre,loopas,runnen,eneave,eruave,
     +denave,druave,widsidx,widsidy,widsidz,twoxpi
      common /cb3/ bolcon,rarega,testen,heatad,
     +enenum,numave,sqnuve,ebenum,tubave,sqbuve
      common /cb4/ runaro,runarar,aroave,ararave,broave,brarave,
     +enaronum,enararnum,ebaronum,ebararnum,heataro,heatArAr,eninte,
     +enarointe,enararinte
      common /cb5/ grid3d(104,28,68),w1(104),w2(28),w3(68),
     +idim1,idim2,idim3
c
c
c
c
c
          value=555555.5D0
          if((x.lt.-0.5D0).or.(x.gt.0.5D0)) then
          write(8,*) 'error: x out of range'
          endif
          if((y.lt.0.25D0).or.(y.gt.0.5D0)) then
          write(8,*) 'error: y out of range'
          endif
          if((z.lt.-0.5D0).or.(z.gt.0.5D0)) then
```

```
      write(8,*) 'error: z out of range'
      endif
c
c     find 8 nearest neighbours
      xmin=0.0D0
      xmax=0.0D0
      ymin=0.0D0
      ymax=0.0D0
      zmin=0.0D0
      zmax=0.0D0
      l=0
13    continue
      l=l+1
      if(l.eq.104) then
      xmin=w1(l-1)
      xmax=w1(l)
      i1min=l-1
      i1max=l
      l=0
      go to 14
      endif
      if(x.ge.w1(l)) then
      xmin=w1(l)
      i1min=l
      else
      xmax=w1(l)
      i1max=l
      l=0
      go to 14
      endif
      go to 13
c
14    continue
      l=l+1
      if(l.eq.28) then
      ymin=w2(l-1)
      ymax=w2(l)
      i2min=l-1
      i2max=l
      l=0
      go to 15
      endif
      if(y.ge.w2(l)) then
      ymin=w2(l)
      i2min=l
      else
      ymax=w2(l)
      i2max=l
      l=0
      go to 15
      endif
      go to 14
c
15    continue
      l=l+1
      if(l.eq.68) then
      zmin=w3(l-1)
      zmax=w3(l)
```

```
      i3min=l-1
      i3max=l
      l=0
      go to 16
      endif
      if(z.ge.w3(l)) then
      zmin=w3(l)
      i3min=l
      else
      zmax=w3(l)
      i3max=l
      go to 16
      endif
      go to 15
c
16    continue
      xneigh(1)=xmin
      yneigh(1)=ymin
      zneigh(1)=zmin
      eneigh(1)=grid3d(i1min,i2min,i3min)
c
      xneigh(2)=xmax
      yneigh(2)=ymin
      zneigh(2)=zmin
      eneigh(2)=grid3d(i1max,i2min,i3min)
c
      xneigh(3)=xmin
      yneigh(3)=ymax
      zneigh(3)=zmin
      eneigh(3)=grid3d(i1min,i2max,i3min)
c
      xneigh(4)=xmax
      yneigh(4)=ymax
      zneigh(4)=zmin
      eneigh(4)=grid3d(i1max,i2max,i3min)
c
      xneigh(5)=xmin
      yneigh(5)=ymin
      zneigh(5)=zmax
      eneigh(5)=grid3d(i1min,i2min,i3max)
c
      xneigh(6)=xmax
      yneigh(6)=ymin
      zneigh(6)=zmax
      eneigh(6)=grid3d(i1max,i2min,i3max)
c
      xneigh(7)=xmin
      yneigh(7)=ymax
      zneigh(7)=zmax
      eneigh(7)=grid3d(i1min,i2max,i3max)
c
      xneigh(8)=xmax
      yneigh(8)=ymax
      zneigh(8)=zmax
      eneigh(8)=grid3d(i1max,i2max,i3max)
c
c     calculate fractional coordinates along x-direction
      xfract1=(x-xneigh(1))/DABS(xneigh(1)-xneigh(2))
```

```
      xfract2=(x-xneigh(3))/DABS(xneigh(3)-xneigh(4))
      xfract3=(x-xneigh(5))/DABS(xneigh(5)-xneigh(6))
      xfract4=(x-xneigh(7))/DABS(xneigh(7)-xneigh(8))
c
c     interpolate energy along x-direction
      valuex1=(1-xfract1)*eneigh(1)+xfract1*eneigh(2)
      valuex2=(1-xfract2)*eneigh(3)+xfract2*eneigh(4)
      valuex3=(1-xfract3)*eneigh(5)+xfract3*eneigh(6)
      valuex4=(1-xfract4)*eneigh(7)+xfract4*eneigh(8)
c
c     calculate fractional coordinates along y-direction
      yfract1=(y-yneigh(1))/DABS(yneigh(1)-yneigh(3))
      yfract2=(y-yneigh(5))/DABS(yneigh(5)-yneigh(7))
c
c     interpolate energy along y direction
      valuey1=(1-yfract1)*valuex1+yfract1*valuex2
      valuey2=(1-yfract2)*valuex3+yfract2*valuex4
c
c     calculate fractional coordinate along z-direction
      zfract=(z-zneigh(1))/DABS(zneigh(1)-zneigh(5))
c
c     interpolate along z-direction
      valuez=(1-zfract)*valuey1+zfract*valuey2
c
      value=valuez
c
      return
      end
c
```

```
      program argcor
c
c     This program converts coordinates from atomic units
c     to reduced units in a silicate-1 framework of 27
c     unit cells
c
      common /cb1/ xpopos(1000),ypopos(1000),zpopos(1000),
     +ixdist(1000),iydist(1000),izdist(1000),cucell,haspac,
     +aucell,bucell,hasidx,hasidy,hasidz,
     +volpor,recvol,cutp02,inumpo,inumm1,inump1,temper,zedvee,
     +steple,nwaren,nwarmo,aucell,bucell,cucell,numsid
      common /cb2/ xpopos1(1000),ypopos1(1000),zpopos1(1000)
      common /cb3/ alimx,alimy,alimz,blimx,blimy,blimz
c
c
c **** Writes out a file for zsmdtmm3.dat 1*3 uc
c
c
c
c
      open(unit=13,file='arpos.dat')
c
      call argon
      do 100 loop2=1,inumpo
      write(13,*) xpopos1(loop2),ypopos1(loop2),zpopos1(loop2)
100   continue
      stop
      end
c
c
c
      subroutine argon
c
      dimension r(97)
      common /cb1/ xpopos(1000),ypopos(1000),zpopos(1000),
     +ixdist(1000),iydist(1000),izdist(1000),cucell,haspac,
     +aucell,bucell,hasidx,hasidy,hasidz,
     +volpor,recvol,cutp02,inumpo,inumm1,inump1,temper,zedvee,
     +steple,nwaren,nwarmo,aucell,bucell,cucell,numsid
      common /cb2/ xpopos1(1000),ypopos1(1000),zpopos1(1000)
      common /cb3/ alimx,alimy,alimz,blimx,blimy,blimz
c
c
        aucell=20.07
        bucell=19.92
        cucell=13.42
c Open files.
      open(unit=12,file='conit5.dat')
      read(12,*) iresfl
c aucell-length of the box side along x direction.
      read(12,*) aucell
c hasidx-half of the box side along the x direction
      hasidx=aucell/2.
c bucell-length of the box side along y direction.
      read(12,*) bucell
c hasidy-half of the box side along the y direction
      hasidy=bucell/2.
c cucell-length of the box side along z direction.
```

```
      read(12,*) cucell
c hasidz-half of the box side along the z direction
      hasidz=cucell/2.
c steple-molecule move translation step-length.
      read(12,*) steple
      read(12,*) eneave
      read(12,*) aroave
      read(12,*) ararave
      read(12,*) enenum
      read(12,*) enaronum
      read(12,*) enararnum
      read(12,*) numave
      read(12,*) sqnuve
      read(12,*) sqenet
      read(12,*) denave
      read(12,*) sqdent
      read(12,*) ipasno
      read(12,*) ipasto
c inumpo-number of argon atoms in pore.
      read(12,*) inumpo
c       print*,'total argon number in system ',inumpo
      inump1=inumpo+1
      inumm1=inumpo-1
c List xpopos-x-coordinates of the pore molecules.
c List ypopos-y-coordinates of the pore molecules.
c List zpopos-z-coordinates of the pore molecules.
      do 30 loop1=1,inumpo
      read(12,*) xtar01
      read(12,*) ytar01
      read(12,*) ztar01
      xpopos(loop1)=xtar01
      ypopos(loop1)=ytar01
      zpopos(loop1)=ztar01
30    continue
c List ixdist-bins for the singlet distribution function in x.
c List iydist-bins for the singlet distribution function in y.
c List izdist-bins for the singlet distribution function in z.
      do 40 loop1=1,1000
      read(12,*) ixdist(loop1)
      read(12,*) iydist(loop1)
      read(12,*) izdist(loop1)
40    continue
c List r(97)-table for ran1.
      read(12,*) ix1
      read(12,*) ix2
      read(12,*) ix3
      do 100 loop1=1,97
      read(12,*) r(loop1)
100   continue
      DO 105 J=1,inumpo
      xpopos(j)=xpopos(j)/aucell
      ypopos(j)=ypopos(j)/bucell
      zpopos(j)=zpopos(j)/cucell
105   CONTINUE
      do 122 k=1,3
      l=1
      do 110 i=1,inumpo
      if((xpopos(i).le.1.5).and.(xpopos(i).ge.-1.5)) then
```

```
      if((ypopos(i).le.1.5).and.(ypopos(i).ge.-1.5)) then
      if((zpopos(i).le.1.5).and.(zpopos(i).ge.-1.5)) then
      xpopos1(l)=xpopos(i)
      ypopos1(l)=ypopos(i)
      zpopos1(l)=zpopos(i)
      l=l+1
      endif
      endif
      endif
110    continue
122    continue
c      print*,'argons in unit cell',l-1
      return
      end
```

The Role of Three-Body Interactions in the Adsorption of Argon in Silicalite-1

Volume II: Appendices I-IV (continued)

Third Year Physical Chemistry Research Report

by
Félix Fernández-Alonso

under the supervision of Dr. David Nicholson and Mr. Roland J-M Pellenq

Imperial College of Science Technology and Medicine
University of London
June 1993

```
      PROGRAM  shcalc4

c
c     ******************************************************************
c
c     Two-body and Three-body Potential Energy Calculation
c
c     This program calculates two-body and three-body terms
c     of the dispersion potential for argon. It includes:
c       - Two-body arar dispersion potential (Barker-Fisher-Watts
c         Potential, true 2b Lennard-Jones and effective
c         Lennard-Jones).
c       - Three-body ararar, ararsi and araro for the central unit
c         cell.
c       - The calculation has been corrected for periodic
c         boundaries by replicating in 3D the 27 original unit cells
c         a number of times specified by the variable nreplica.
c
c     It reads the following data files:
c       - parama.dat, parameter file with all the necessary constants
c         such as dispersion coefficients, long range cutoffs, etc
c       - zsmox.dat and zsmsi.dat, the silicon and oxygen coordinates
c         of zeolite ZSM5 silicalite.
c       - arpos.dat, the argon coordinates in the zeolite.
c
c     Discriminates between particles in different parts of the unit
c     cell, namely:
c       - Straight channel.
c       - Intersection.
c       - Zig-zag (sinusoidal) channels.
c
c     And calculates the energy contributions from each region in the
c     central unit cell.
c
c     Final results are displayed in file results.tem in standard
c     format.
c
c     Program analysis.f creates a second file with the data analysis
c
c
c     ******************************************************************
c
c     The explicit form of the two-body and three-body potentials can
c     be found in the following references:
c
c       -Two-body dispersion potential for argon:
c        J.A. Barker and A. Pompe, Aust. J. of Chem.,1968,21:1683-94
c        J.A. Barker, R.A. Fisher and R.O. Watts, Mol. Phys.,1971,
c        21: 657
c       -Two-body dispersion potential for krypton and xenon:
c        J.A. Barker, R.O. Watts, et al., J. Chem. Phys.,61(8),1974:
c        3081
c       -Three-body dispersion potentials:
c        B.M. Axilrod and E. Teller, J. Chem. Phys.,11(6),1943:299
c        R. J. Bell, J. Phys. B: Atom. Mol. Phys.,1970,3:751
c        K.T. Tang et al., J. Chem. Phys. 64(7),1976:3063
c        M.B. Doran et al., J. Phys. C: Solid St. Phys.,1971,4:307
c
```

```
c     ******************************************************************
c
c       Written by:
c       Felix Fernandez-Alonso, Imperial College.
c       February, March, April, May 1993
c
c     ******************************************************************

      implicit double precision(a-h,o-z)

      common /param6e/ c6arar,c8arar,c10arar,c6aro,c8aro,c10aro,
     +                 c6arsi,c8arsi,c10arsi
      common /bfwparams/ a0arar,a1arar,a2arar,a3arar,a4arar,a5arar,
     +                 aarar,deltarar,epsilbfw,radbfw
      common /LJones/epsilar,sigmar
      common /LJones2b/epsilar2b,sigmar2b
      common /param3bararar/addd,addq,adqd,aqdd,adqq,aqdq,aqqd,
     +                  aqqq,ad41,ad42,ad43
      common /param3bararo/ oddd,oddq,odqd,oqdd,odqq,oqdq,oqqd,
     +                  oqqq,od41,od42,od43
      common /param3bararsi/ sddd,sddq,sdqd,sqdd,sdqq,sqdq,sqqd,
     +                  sqqq,sd41,sd42,sd43
      common /cut2b3b/ hcut
      common /repulsy/ areparar,areparo,areparsi,breparar,
     +                breparo,breparsi
      common /oxpos/ nox,xox(5184),yox(5184),zox(5184)
      common /sipos/ nsi,xsi(2592),ysi(2592),zsi(2592)
      common /arpos/ nar,naruc,xar(1000),yar(1000),zar(1000)
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
      common/check/natlj,nrtlj,nabfw,nrbfw,naelj,nrelj,na3ar,nr3ar,
     +na3o,nr3o,na3si,nr3si
      common/intera/ntlj,nelj,nbfw,n3ar,n3o,n3si

      print*,'program POTENTIAL CALCULATION starts'

      call readparams
      call geometry
      call set3bararar
      call set3bararo
      call set3bararsi
      call setbound
      calcflag=0
      call potcalculuc
      call potcalculsch
      call potcalculint
      call potcalculzzc
      end

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

      subroutine readparams

      implicit double precision(a-h,o-z)

      common /param6e/ c6arar,c8arar,c10arar,c6aro,c8aro,c10aro,
     +                 c6arsi,c8arsi,c10arsi
      common /bfwparams/ a0arar,a1arar,a2arar,a3arar,a4arar,a5arar,
     +                 aarar,deltarar,epsilbfw,radbfw
```

```
      common /LJones/epsilar,sigmar
      common /LJones2b/epsilar2b,sigmar2b
      common /param3bararar/ addd, addq,adqd,aqdd,adqq,aqdq,aqqd,
     +                aqqq,ad41,ad42,ad43
      common /param3bararo/ oddd, oddq,odqd,oqdd,odqq,oqdq,oqqd,
     +                oqqq,od41,od42,od43
      common /param3bararsi/ sddd, sddq,sdqd,sqdd,sdqq,sqdq,sqqd,
     +                sqqq,sd41,sd42,sd43
      common /cut2b3b/ hcut
      common /highenergy/ en3ar,en3o,en3si
      common /repulsy/ areparar,areparo,areparsi,breparar,
     +               breparo,breparsi
      common/bounds/xlow,xhigh,ylow,yhigh,zlow,zhigh
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
      common /ucsegments/ xuc1,xuc2,yuc1,yuc2

      open(unit=20,file='parama3.dat')

c     Long-range cutoff for 2b and 3b interactions.

      read(20,*) hcut

c     Number of replicas of the original 27 unit cells.

      read(20,*) nreplica

c     Lower and upper bounds to which the calculation is restricted
c     (in reduced units).

      read(20,*) xlow
      read(20,*) xhigh
      read(20,*) ylow
      read(20,*) yhigh
      read(20,*) zlow
      read(20,*) zhigh

c     BFW Two-body terms for ArAr, ArO and ArSi interactions.

      read(20,*) c6arar
      read(20,*) c8arar
      read(20,*) c10arar

      read(20,*) a0arar
      read(20,*) a1arar
      read(20,*) a2arar
      read(20,*) a3arar
      read(20,*) a4arar
      read(20,*) a5arar
      read(20,*) aarar
      read(20,*) deltarar
      read(20,*) epsilbfw
      read(20,*) radbfw

c     Lennard-Jones parameters, epsilon (in kelvin) and sigma (A),
c     for argon

      read(20,*) epsilar
      read(20,*) sigmar
```

```
      read(20,*) epsilar2b
      read(20,*) sigmar2b

c     Three-body terms for ArArAr (a), ArArO (o)
c     and ArArSi (s) interactions.

      read(20,*) addd
      read(20,*) aqdd
      read(20,*) addq
      read(20,*) adqd
      read(20,*) aqqd
      read(20,*) aqdq
      read(20,*) adqq
      read(20,*) aqqq
      read(20,*) ad41
      read(20,*) ad42
      read(20,*) ad43

      read(20,*) oddd
      read(20,*) oqdd
      read(20,*) oddq
      read(20,*) odqd
      read(20,*) oqqd
      read(20,*) oqdq
      read(20,*) odqq
      read(20,*) oqqq
      read(20,*) od41
      read(20,*) od42
      read(20,*) od43

      read(20,*) sddd
      read(20,*) sqdd
      read(20,*) sddq
      read(20,*) sdqd
      read(20,*) sqqd
      read(20,*) sqdq
      read(20,*) sdqq
      read(20,*) sqqq
      read(20,*) sd41
      read(20,*) sd42
      read(20,*) sd43

c     energy threshold for 3-body interactions.

      read(20,*) en3ar
      read(20,*) en3o
      read(20,*) en3si

c     Reduced coordinates of the straight channel,
c     intersection and zig-zag channel in the unit cell.

      read(20,*) xuc1
      read(20,*) xuc2
      read(20,*) yuc1
      read(20,*) yuc2


      print*, 'all parameters read in ...'
```

```
      return
      end

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

      subroutine geometry

      implicit double precision(a-h,o-z)

      common /oxpos/ nox,xox(5184),yox(5184),zox(5184)
      common /sipos/ nsi,xsi(2592),ysi(2592),zsi(2592)
      common /arpos/ nar,naruc, xar(1000),yar(1000),zar(1000)
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
      common/bounds/xlow,xhigh,ylow,yhigh,zlow,zhigh
      common /ucsegments/ xuc1,xuc2,yuc1,yuc2
      common /arposuc/ xaruc(50),yaruc(50),zaruc(50)
      common /arpossch/ xarsch(50),yarsch(50),zarsch(50),narsch
      common /arposint/ xarint(50),yarint(50),zarint(50),narint
      common /arposzzc/ xarzzc(50),yarzzc(50),zarzzc(50),narzzc

c     O and Si coords in zsmox and zsmsi are in fractional u.c. units.
c     27 unit cells with origin at the centre.

      open(unit=2,file='zsmox.dat')
      open(unit=3,file='zsmsi.dat')
      open(unit=4,file='arpos.dat')

      auc=20.07
      buc=19.92
      cuc=13.42
      rbohr=0.52917

c     read oxygen coordinates.

      nox=0
21     nox=nox+1
      read(2,*,end=23) x1,y1,z1
      xox(nox)=x1
      yox(nox)=y1
      zox(nox)=z1
      goto 21
23     nox=nox-1

      print*,nox,' oxygen coordinates read in'

c     convert to bohr units
      do 10 j=1,nox
      xox(j)=auc*xox(j)/rbohr
      yox(j)=buc*yox(j)/rbohr
      zox(j)=cuc*zox(j)/rbohr
10     continue
      print*,'oxygen coordinates converted to Bohr units'

c     read silicon coordinates.

      nsi=0
20     nsi=nsi+1
      read(3,*,end=30) x2,y2,z2
```

```
      xsi(nsi)=x2
      ysi(nsi)=y2
      zsi(nsi)=z2
      goto 20
30     nsi=nsi-1
      print*,nsi,' silicon coordinates read in'

c     convert to bohr units
      do 40 j=1,nsi
      xsi(j)=auc*xsi(j)/rbohr
      ysi(j)=buc*ysi(j)/rbohr
      zsi(j)=cuc*zsi(j)/rbohr
40     continue
      print*,'silicon coordinates converted to Bohr units'

c     read argon coordinates.
c     count and store how many atoms are located in central
c     unit cell.

      nar=0
      naruc=0
      narsch=0
      narint=0
      narzzc=0

55     nar=nar+1
      read(4,*,end=56) x3,y3,z3
      xar(nar)=x3
      yar(nar)=y3
      zar(nar)=z3

      if ((xar(nar).gt.xlow).and.(xar(nar).lt.xhigh)
     +.and.(yar(nar).gt.ylow).and.(yar(nar).lt.yhigh)
     +.and.(zar(nar).lt.zhigh).and.(zar(nar).gt.zlow)) then

      naruc=naruc+1
      xaruc(naruc)=xar(nar)
      yaruc(naruc)=yar(nar)
      zaruc(naruc)=zar(nar)

      endif

      goto 55

56     nar=nar-1

      narsch=0
      narint=0
      narzzc=0

      do 61 i=1,naruc

      if (abs(xaruc(i)).lt.xuc1) then

      if ((abs(yaruc(i)).lt.yuc1).or.(abs(yaruc(i)).gt.yuc2)) then

      narsch=narsch+1
      xarsch(narsch)=xaruc(i)
```

```
      yarsch(narsch)=yaruc(i)
      zarsch(narsch)=zaruc(i)

      else

      narint=narint+1
      xarint(narint)=xaruc(i)
      yarint(narint)=yaruc(i)
      zarint(narint)=zar(i)

      endif
      endif

      if ((abs(xaruc(i)).gt.xuc1).and.(abs(xaruc(i)).lt.xuc2)) then

      narzzc=narzzc+1
      xarzzc(narzzc)=xaruc(i)
      yarzzc(narzzc)=yaruc(i)
      zarzzc(narzzc)=zaruc(i)

      endif

      if (abs(xaruc(i)).gt.xuc2) then

      if ((abs(yaruc(i)).lt.yuc1).or.(abs(yaruc(i)).gt.yuc2)) then

      narsch=narsch+1
      xarsch(narsch)=xaruc(i)
      yarsch(narsch)=yaruc(i)
      zarsch(narsch)=zaruc(i)

      else

      narint=narint+1
      xarint(narint)=xaruc(i)
      yarint(narint)=yaruc(i)
      zarint(narint)=zaruc(i)

      endif
      endif


61    continue

      print*,nar,' argon coordinates read in'
      print*,naruc, ' argon atoms in selected unit cell'
      print*,narsch,' argon atoms in straight channel'
      print*,narint,' argon atoms in intersection'
      print*,narzzc,' argon atoms in zig-zag channel'

c     convert to bohr units

      do 57 j=1,nar
      xar(j)=auc*xar(j)/rbohr
      yar(j)=buc*yar(j)/rbohr
      zar(j)=cuc*zar(j)/rbohr

57    continue
```

```
      do 58 j=1,narsch
      xarsch(j)=auc*xarsch(j)/rbohr
      yarsch(j)=buc*yarsch(j)/rbohr
      zarsch(j)=cuc*zarsch(j)/rbohr

58    continue

      do 59 j=1,narint
      xarint(j)=auc*xarint(j)/rbohr
      yarint(j)=buc*yarint(j)/rbohr
      zarint(j)=cuc*zarint(j)/rbohr

59    continue

      do 60 j=1,narzzc
      xarzzc(j)=auc*xarzzc(j)/rbohr
      yarzzc(j)=buc*yarzzc(j)/rbohr
      zarzzc(j)=cuc*zarzzc(j)/rbohr

60    continue

      do 65 j=1,naruc
      xaruc(j)=auc*xaruc(j)/rbohr
      yaruc(j)=buc*yaruc(j)/rbohr
      zaruc(j)=cuc*zaruc(j)/rbohr

65    continue

      print*,'argon coordinates converted to Bohr units'

      return
      end

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

      subroutine set3bararar

      implicit double precision(a-h,o-z)

      common /param3bararar/ addd, addq,adqd,aqdd,adqq,aqdq,aqqd,
     +                       aqqq,ad41,ad42,ad43


c     convert energy to k

      factk=3.158e5
      addd=addd*factk

      aqdd=aqdd*factk
      addq=addq*factk
      adqd=adqd*factk

      aqqd=aqqd*factk
      aqdq=aqdq*factk
      adqq=adqq*factk

      aqqq=aqqq*factk
```

```
      addd41=ad41*factk
      addd42=ad42*factk
      addd43=ad43*factk

      return
      end

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

      subroutine set3bararo

      implicit double precision(a-h,o-z)

      common /param3bararo/ oddd, oddq,odqd,oqdd,odqq,oqdq,oqqd,
     +                oqqq,od41,od42,od43

c     convert energy to k

      factk=3.158e5
      oddd=oddd*factk

      oqdd=oqdd*factk
      oddq=oddq*factk
      odqd=odqd*factk

      oqqd=oqqd*factk
      oqdq=oqdq*factk
      odqq=odqq*factk

      oqqq=oqqq*factk

      oddd41=od41*factk
      oddd42=od42*factk
      oddd43=od43*factk

      return
      end

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

      subroutine set3bararsi

      implicit double precision(a-h,o-z)

      common /param3bararsi/ sddd, sddq,sdqd,sqdd,sdqq,sqdq,sqqd,
     +                sqqq,sd41,sd42,sd43

c     convert energy to k

      factk=3.158e5
      sddd=sddd*factk

      sqdd=sqdd*factk
      sddq=sddq*factk
      sdqd=sdqd*factk

      sqqd=sqqd*factk
```

```
      sqdq=sqdq*factk
      sdqq=sdqq*factk

      sqqq=sqqq*factk

      sddd41=sd41*factk
      sddd42=sd42*factk
      sddd43=sd43*factk

      return
      end

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

      subroutine setbound

      implicit double precision(a-h,o-z)

      common/bounds/xlow,xhigh,ylow,yhigh,zlow,zhigh
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
      common /ucsegments/ xuc1,xuc2,yuc1,yuc2

c     Converts to atomic units the lower and upper bounds of the unit
c     cell where the calculation is performed.

      xlow=xlow*auc/rbohr
      xhigh=xhigh*auc/rbohr
      ylow=ylow*buc/rbohr
      yhigh=yhigh*buc/rbohr
      zlow=zlow*cuc/rbohr
      zhigh=zhigh*cuc/rbohr

      return
      end

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

      subroutine potcalculuc

      implicit double precision(a-h,o-z)

      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arpos/ nar,naruc, xar(1000),yar(1000),zar(1000)
      common /arposuc/ xaruc(50),yaruc(50),zaruc(50)


      do 10 i=1,naruc

      xardum(i)=xaruc(i)
      yardum(i)=yaruc(i)
      zardum(i)=zaruc(i)

10    continue

      nardum=naruc

      call shiftarray
      call shcalcul
```

```
      return
      end

      subroutine potcalculsch

      implicit double precision(a-h,o-z)

      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arpossch/ xarsch(50),yarsch(50),zarsch(50),narsch


      do 10 i=1,narsch

      xardum(i)=xarsch(i)
      yardum(i)=yarsch(i)
      zardum(i)=zarsch(i)

10    continue

      nardum=narsch

      call shiftarray
      call shcalcul

      return
      end

      subroutine potcalculint

      implicit double precision(a-h,o-z)

      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arposint/ xarint(50),yarint(50),zarint(50),narint


      do 10 i=1,narint

      xardum(i)=xarint(i)
      yardum(i)=yarint(i)
      zardum(i)=zarint(i)

10    continue

      nardum=narint

      call shiftarray
      call shcalcul

      return
      end

      subroutine potcalculzzc

      implicit double precision(a-h,o-z)

      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arposzzc/ xarzzc(50),yarzzc(50),zarzzc(50),narzzc
```

```
      do 10 i=1,narzzc

      xardum(i)=xarzzc(i)
      yardum(i)=yarzzc(i)
      zardum(i)=zarzzc(i)

10     continue

      nardum=narzzc

      call shiftarray
      call shcalcul

      return
      end

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

      subroutine shiftarray

      implicit double precision(a-h,o-z)

      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arpos/ nar,naruc, xar(1000),yar(1000),zar(1000)

      print*,'in subroutine shift array'

      pos=1

      do 10 i=1,nar
      do 20 j=1,nardum

      if ((xar(i).eq.xardum(j)).and.(yar(i).eq.yardum(j)).and.
     +(zar(i).eq.zardum(j))) then

      xtem=xar(i)
      ytem=yar(i)
      ztem=zar(i)

      xar(i)=xar(pos)
      yar(i)=yar(pos)
      zar(i)=zar(pos)

      xar(pos)=xtem
      yar(pos)=ytem
      zar(pos)=ztem

      pos=pos+1

      goto 10

      endif

20     continue
10     continue
```

```
      return
      end

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

      subroutine shcalcul

      implicit double precision(a-h,o-z)

      common /param6e/ c6arar,c8arar,c10arar,c6aro,c8aro,c10aro,
     +              c6arsi,c8arsi,c10arsi
      common /param3bararar/ addd, addq,adqd,aqdd,adqq,aqdq,aqqd,
     +               aqqq,ad41,ad42,ad43
      common /param3bararo/ oddd, oddq,odqd,oqdd,odqq,oqdq,oqqd,
     +               oqqq,od41,od42,od43
      common /param3bararsi/ sddd, sddq,sdqd,sqdd,sdqq,sqdq,sqqd,
     +               sqqq,sd41,sd42,sd43
      common /cut2b3b/ hcut
      common /repulsy/ areparar,areparo,areparsi,breparar,
     +               breparo,breparsi
      common /oxpos/ nox,xox(5184),yox(5184),zox(5184)
      common /sipos/ nsi,xsi(2592),ysi(2592),zsi(2592)
      common /arpos/ nar,naruc,xar(1000),yar(1000),zar(1000)
      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
      common/results/potarar,potLJarar,potbfw,pot3ararar,pot3araro,
     +pot3ararsi,pot3ar1,pot3ar2,pot3ar3,pot3ar4,pot3ar5,pot3o1,
     +pot3o2,pot3o3,pot3o4,pot3o5,pot3si1,pot3si2,pot3si3,
     +pot3si4,pot3si5


c     potential calculation

c     initialise variables

      potarar=0.0
      potLJarar=0.0
      potbfw=0.0

      pot3ararar=0.0
      pot3ar1=0.0
      pot3ar2=0.0
      pot3ar3=0.0
      pot3ar4=0.0
      pot3ar5=0.0

      pot3araro=0.0
      pot3o1=0.0
      pot3o2=0.0
      pot3o3=0.0
      pot3o4=0.0
      pot3o5=0.0

      pot3ararsi=0.0
      pot3si1=0.0
      pot3si2=0.0
      pot3si3=0.0
      pot3si4=0.0
```

```
      pot3si5=0.0

c     call two-body interaction subroutines

      print*,'calculating potential of interaction ...'

      call pot2barar(potarar)
      call potLJParar(potLJarar)
      call pot2bfw(potbfw)
      call convertok(potbfw)

c     call three-body interaction subroutines

      call pot3barararar(pot3arararar,pot3ar1,pot3ar2,pot3ar3,pot3ar4,
     +pot3ar5)
      call pot3bararo(pot3araro,pot3o1,pot3o2,pot3o3,pot3o4,
     +pot3o5)
      call pot3bararsi(pot3ararsi,pot3si1,pot3si2,pot3si3,pot3si4,
     +pot3si5)

      calcflag=calcflag+1

      call finresul

      return
      end

      subroutine finresul

      implicit double precision(a-h,o-z)

      common /arpos/ nar,naruc,xar(1000),yar(1000),zar(1000)
      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common/results/potarar,potLJarar,potbfw,pot3arararar,pot3araro,
     +pot3ararsi,pot3ar1,pot3ar2,pot3ar3,pot3ar4,pot3ar5,pot3o1,
     +pot3o2,pot3o3,pot3o4,pot3o5,pot3si1,pot3si2,pot3si3,
     +pot3si4,pot3si5
      common/check/natlj,nrtlj,nabfw,nrbfw,naelj,nrelj,na3ar,nr3ar,
     +na3o,nr3o,na3si,nr3si
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag


      open(unit=17,file='result.tem')

      if (calcflag.eq.1) then
      write(17,*) nar
      endif
      write(17,*) calcflag
      write(17,*) nardum
      write(17,*) potarar
      write(17,*) potLJarar
      write(17,*) potbfw
      write(17,*) pot3arararar
      write(17,*) pot3ar1
      write(17,*) pot3ar2
      write(17,*) pot3ar3
      write(17,*) pot3ar4
      write(17,*) pot3ar5
```

```
      write(17,*) pot3araro
      write(17,*) pot3o1
      write(17,*) pot3o2
      write(17,*) pot3o3
      write(17,*) pot3o4
      write(17,*) pot3o5
      write(17,*) pot3ararsi
      write(17,*) pot3si1
      write(17,*) pot3si2
      write(17,*) pot3si3
      write(17,*) pot3si4
      write(17,*) pot3si5
      write(17,*) natlj
      write(17,*) nrtlj
      write(17,*) naelj
      write(17,*) nrelj
      write(17,*) nabfw
      write(17,*) nrbfw
      write(17,*) na3ar
      write(17,*) nr3ar
      write(17,*) na3o
      write(17,*) nr3o
      write(17,*) na3si
      write(17,*) nr3si

      return
      end

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

      subroutine pot2barar(sumarar)

      implicit double precision(a-h,o-z)

      common /param6e/ c6arar,c8arar,c10arar,c6aro,c8aro,c10aro,
     +                 c6arsi,c8arsi,c10arsi
      common /cut2b3b/ hcut
      common /repulsy/ areparar,areparo,areparsi,breparar,
     +                 breparo,breparsi
      common /arpos/ nar,naruc, xar(1000),yar(1000),zar(1000)
      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arpostem/xartem,yartem,zartem
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
      common/bounds/xlow,xhigh,ylow,yhigh,zlow,zhigh
      common/LJones2b/epsilar2b,sigmar2b
      common/check/natlj,nrtlj,nabfw,nrbfw,naelj,nrelj,na3ar,nr3ar,
     +na3o,nr3o,na3si,nr3si

      sumarar=0.0
      natlj=0
      nrtlj=0

      rbohr=0.52917
      sigmar2bB=sigmar2b/rbohr

      do 10 m=1,nardum

      x1=xar(m)
```

```
      y1=yar(m)
      z1=zar(m)

      do 20 n=m+1,nar

      do 30 i=-nreplica,nreplica
      do 40 j=-nreplica,nreplica
      do 50 k=-nreplica,nreplica

c     Convert argon coordinates to reduced units

      xar(n)=xar(n)*(rbohr/(3*auc))
      yar(n)=yar(n)*(rbohr/(3*buc))
      zar(n)=zar(n)*(rbohr/(3*cuc))

c     Store coordinates of replicas of original simulation box.

      xartem=(3*auc/rbohr)*(xar(n)+i)
      yartem=(3*buc/rbohr)*(yar(n)+j)
      zartem=(3*cuc/rbohr)*(zar(n)+k)

      xar(n)=(3*auc/rbohr)*xar(n)
      yar(n)=(3*buc/rbohr)*yar(n)
      zar(n)=(3*cuc/rbohr)*zar(n)

      xarx2=(xartem-x1)**2
      yary2=(yartem-y1)**2
      zarz2=(zartem-z1)**2
      xarx1=xartem-x1
      yary1=yartem-y1
      zarz1=zartem-z1

      r2arar=xarx2+yary2+zarz2
      r1arar=sqrt(r2arar)

      if (r1arar.gt.hcut) then
      nrtlj=nrtlj+1
      goto 50
      endif

      arar2b=0.0

      redr6=(r1arar/sigmar2bB)**6
      redr12=(r1arar/sigmar2bB)**12
      arar2b=4*epsilar2b*((1/redr12)-(1/redr6))
c
      sumarar=arar2b+sumarar
      natlj=natlj+1

50    continue
40    continue
30    continue
20    continue
10    continue

      return
      end
```

```
c     ******************************************************************

      subroutine potLJParar(sumLJarar)

      implicit double precision(a-h,o-z)

      common /cut2b3b/ hcut
      common /arpos/ nar,naruc, xar(1000),yar(1000),zar(1000)
      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arpostem/xartem,yartem,zartem
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
      common/bounds/xlow,xhigh,ylow,yhigh,zlow,zhigh
      common /LJones/epsilar,sigmar
      common/check/natlj,nrtlj,nabfw,nrbfw,naelj,nrelj,na3ar,nr3ar,
     +na3o,nr3o,na3si,nr3si

      rbohr=0.52917
      sigmarB=sigmar/rbohr
      sumLJarar=0.0
      naelj=0
      nrelj=0

      do 10 m=1,nardum

      x1=xar(m)
      y1=yar(m)
      z1=zar(m)

      do 20 n=m+1,nar

      do 30 i=-nreplica,nreplica
      do 40 j=-nreplica,nreplica
      do 50 k=-nreplica,nreplica

c     Convert argon coordinates to reduced units

      xar(n)=xar(n)*(rbohr/(3*auc))
      yar(n)=yar(n)*(rbohr/(3*buc))
      zar(n)=zar(n)*(rbohr/(3*cuc))

c     Store coordinates of replicas of original simulation box.

      xartem=(3*auc/rbohr)*(xar(n)+i)
      yartem=(3*buc/rbohr)*(yar(n)+j)
      zartem=(3*cuc/rbohr)*(zar(n)+k)

      xar(n)=(3*auc/rbohr)*xar(n)
      yar(n)=(3*buc/rbohr)*yar(n)
      zar(n)=(3*cuc/rbohr)*zar(n)

      xarx2=(xartem-x1)**2
      yary2=(yartem-y1)**2
      zarz2=(zartem-z1)**2
      xarx1=xartem-x1
      yary1=yartem-y1
      zarz1=zartem-z1
      r2arar=xarx2+yary2+zarz2
      r1arar=sqrt(r2arar)
```

```
      ararLJ=0.0

      if (r1arar.gt.hcut) then
      nrelj=nrelj+1
      goto 50
      endif

      redr6=(r1arar/sigmarB)**6
      redr12=(r1arar/sigmarB)**12
      ararLJ=4*epsilar*((1/redr12)-(1/redr6))

      sumLJarar=ararLJ+sumLJarar
      naelj=naelj+1

50    continue
40    continue
30    continue
20    continue
10    continue

      return
      end

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

      subroutine pot2bfw(sumbfw)

      implicit double precision(a-h,o-z)

      common /param6e/ c6arar,c8arar,c10arar,c6aro,c8aro,c10aro,
     +                 c6arsi,c8arsi,c10arsi
      common /bfwparams/ a0arar,a1arar,a2arar,a3arar,a4arar,a5arar,
     +                 aarar,deltarar,epsilbfw,radbfw
      common /cut2b3b/ hcut
      common /arpos/ nar,naruc, xar(1000),yar(1000),zar(1000)
      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arpostem/xartem,yartem,zartem
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
      common/bounds/xlow,xhigh,ylow,yhigh,zlow,zhigh
      common/check/natlj,nrtlj,nabfw,nrbfw,naelj,nrelj,na3ar,nr3ar,
     +na3o,nr3o,na3si,nr3si


      sumbfw=0.0
      nabfw=0
      nrbfw=0

      rbohr=0.52917
      radbfwB=radbfw/rbohr

      do 10 m=1,nardum

      x1=xar(m)
      y1=yar(m)
      z1=zar(m)

      do 20 n=m+1,nar
```

```
      do 30 i=-nreplica,nreplica
      do 40 j=-nreplica,nreplica
      do 50 k=-nreplica,nreplica

c     Convert argon coordinates to reduced units

      xar(n)=xar(n)*(rbohr/(3*auc))
      yar(n)=yar(n)*(rbohr/(3*buc))
      zar(n)=zar(n)*(rbohr/(3*cuc))

c     Store coordinates of replicas of original simulation box.

      xartem=(3*auc/rbohr)*(xar(n)+i)
      yartem=(3*buc/rbohr)*(yar(n)+j)
      zartem=(3*cuc/rbohr)*(zar(n)+k)

      xar(n)=(3*auc/rbohr)*xar(n)
      yar(n)=(3*buc/rbohr)*yar(n)
      zar(n)=(3*cuc/rbohr)*zar(n)

      xarx2=(xartem-x1)**2
      yary2=(yartem-y1)**2
      zarz2=(zartem-z1)**2
      xarx1=xartem-x1
      yary1=yartem-y1
      zarz1=zartem-z1
      r2arar=xarx2+yary2+zarz2
      r1arar=sqrt(r2arar)

      if (r1arar.gt.hcut) then
      nrbfw=nrbfw+1
      goto 50
      endif

c     Evaluate the Barker-Fisher-Watts Potential.

      redr1=r1arar/radbfwB
      ararbfw=0.0
      expterm=exp(aarar*(1-redr1))

      a0term=a0arar
      a1term=a1arar*((redr1-1))
      a2term=a2arar*((redr1-1)**2)
      a3term=a3arar*((redr1-1)**3)
      a4term=a4arar*((redr1-1)**4)
      a5term=a5arar*((redr1-1)**5)

      asum=a0term+a1term+a2term+a3term+a4term+a5term

      term1=expterm*asum

      c6term=c6arar/(deltarar+(redr1**6))
      c8term=c8arar/(deltarar+(redr1**8))
      c10term=c10arar/(deltarar+(redr1**10))

      term2=c6term+c8term+c10term

      ararbfw=term1+term2
```

```
      sumbfw=ararbfw+sumbfw

      nabfw=nabfw+1

50    continue
40    continue
30    continue
20    continue
10    continue

      return
      end

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

      subroutine pot3bararar(sumararar,totararar1,totararar2,
     +totararar3,totararar4,totararar5)

      implicit double precision(a-h,o-z)

      common /param3bararar/addd,addq,adqd,aqdd,adqq,aqdq,aqqd,
     +                aqqq,ad41,ad42,ad43
      common /arpos/ nar,naruc, xar(1000),yar(1000),zar(1000)
      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arpostem/xartem,yartem,zartem
      common /cut2b3b/ hcut
      common /highenergy/ en3ar,en3o,en3si
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
      common/bounds/xlow,xhigh,ylow,yhigh,zlow,zhigh
      common/check/natlj,nrtlj,nabfw,nrbfw,naelj,nrelj,na3ar,nr3ar,
     +na3o,nr3o,na3si,nr3si

      sumararar=0.0
      na3ar=0
      nr3ar=0
      totararar1=0.0
      totararar2=0.0
      totararar3=0.0
      totararar4=0.0
      totararar5=0.0

      do 10 il=1,nardum

      xar0=xar(il)
      yar0=yar(il)
      zar0=zar(il)

      totalddd=0.00
      totalddq=0.00
      totaldqd=0.00
      totalqdd=0.00
      totaldqq=0.00
      totalqdq=0.00
      totalqqd=0.00
      totalqqq=0.00
      totald4=0.000

      do 20 im=(il+1),(nar-1)
```

```
      do 40 i=-nreplica,nreplica
      do 50 j=-nreplica,nreplica
      do 60 k=-nreplica,nreplica

c     Convert argon coordinates to reduced units

      xar(im)=xar(im)*(rbohr/(3*auc))
      yar(im)=yar(im)*(rbohr/(3*buc))
      zar(im)=zar(im)*(rbohr/(3*cuc))

c     Store coordinates of replicas of original simulation box.

      xartem=(3*auc/rbohr)*(xar(im)+i)
      yartem=(3*buc/rbohr)*(yar(im)+j)
      zartem=(3*cuc/rbohr)*(zar(im)+k)

      xar(im)=(3*auc/rbohr)*xar(im)
      yar(im)=(3*buc/rbohr)*yar(im)
      zar(im)=(3*cuc/rbohr)*zar(im)

      xar1=xartem
      yar1=yartem
      zar1=zartem
      dif1x=xar1-xar0
      dif1y=yar1-yar0
      dif1z=zar1-zar0
      rar0ar1=sqrt(dif1x*dif1x+dif1y*dif1y+dif1z*dif1z)

      if(rar0ar1.gt.hcut) then
      nr3ar=nr3ar+1
      go to 60
      endif

      do 30 in=im+1,nar

      do 70 l=-nreplica,nreplica
      do 80 m=-nreplica,nreplica
      do 90 n=-nreplica,nreplica

c     Convert argon coordinates to reduced units

      xar(in)=xar(in)*(rbohr/(3*auc))
      yar(in)=yar(in)*(rbohr/(3*buc))
      zar(in)=zar(in)*(rbohr/(3*cuc))

c     Store coordinates of replicas of original simulation box.

      xartem=(3*auc/rbohr)*(xar(in)+l)
      yartem=(3*buc/rbohr)*(yar(in)+m)
      zartem=(3*cuc/rbohr)*(zar(in)+n)

      xar(in)=(3*auc/rbohr)*xar(in)
      yar(in)=(3*buc/rbohr)*yar(in)
      zar(in)=(3*cuc/rbohr)*zar(in)

      xar2=xartem
      yar2=yartem
      zar2=zartem
```

```
      dif2x=xar2-xar0
      dif2y=yar2-yar0
      dif2z=zar2-zar0
c
      dif3x=xar2-xar1
      dif3y=yar2-yar1
      dif3z=zar2-zar1
      rar0ar2=sqrt(dif2x*dif2x+dif2y*dif2y+dif2z*dif2z)
      rar1ar2=sqrt(dif3x*dif3x+dif3y*dif3y+dif3z*dif3z)
c
      if(rar0ar2.gt.hcut) then
      nr3ar=nr3ar+1
      go to 90
      endif
c
      cos1=(dif1x*dif2x+dif1y*dif2y+dif1z*dif2z)/(rar0ar1*rar0ar2)
      cos2=(-dif1x*dif3x-dif1y*dif3y-dif1z*dif3z)/(rar0ar1*rar1ar2)
      cos3=(dif2x*dif3x+dif2y*dif3y+dif2z*dif3z)/(rar0ar2*rar1ar2)
c
      cosangledif1=cos1*cos2+sqrt(abs((1-cos1**2)*(1-cos2**2)))
      cosangledif2=cos2*cos3+sqrt(abs((1-cos2**2)*(1-cos3**2)))
      cosangledif3=cos3*cos1+sqrt(abs((1-cos3**2)*(1-cos1**2)))
c
      cos3fi1=-3*cos1+4*cos1**3
      cos3fi2=-3*cos2+4*cos2**3
      cos3fi3=-3*cos3+4*cos3**3
c
      cos2fi1=2*cos1**2-1
      cos2fi2=2*cos2**2-1
      cos2fi3=2*cos3**2-1
c
      cos2angledif1=2*(cosangledif1)**2-1
      cos2angledif2=2*(cosangledif2)**2-1
      cos2angledif3=2*(cosangledif3)**2-1
c
      cos4fi1=8*cos1**3*sqrt(abs(1-cos1**2))-4*cos1*sqrt(abs(1-cos1**2))
      cos4fi2=8*cos2**3*sqrt(abs(1-cos2**2))-4*cos2*sqrt(abs(1-cos2**2))
      cos4fi3=8*cos3**3*sqrt(abs(1-cos3**2))-4*cos3*sqrt(abs(1-cos3**2))

c     in all following functions, the order ar0,ar1,ar2 is considered
c
c     function three body dipole-dipole-dipole
c
      fddd=3.0*addd*(1.0+3.0*cos1*cos2*cos3)*
     +(rar0ar1*rar0ar2*rar1ar2)**(-3.0)
c
c     function three body dipole-dipole-quadrupole
c
      fddq=addq*3.0/(16.0*(rar0ar1**3)*(rar0ar2*rar1ar2)**4)*
     +((9.0*cos3-25.0*cos3fi3)+6.0*cosangledif1*(3.0+5.0*cos2fi3))
c
c     function three body dipole-quadrupole-dipole
c
      fdqd=adqd*3.0/(16.0*(rar0ar2**3)*(rar1ar2*rar0ar1)**4)*
     +((9.0*cos2-25.0*cos3fi2)+6.0*cosangledif3*(3.0+5.0*cos2fi2))
c
c     function three body quadrupole-dipole-dipole
c
```

```
      fqdd=aqdd*3.0/(16.0*(rar1ar2**3)*(rar0ar2*rar0ar1)**4)*
     +((9.0*cos1-25.0*cos3fi1)+6.0*cosangledif2*(3.0+5.0*cos2fi1))
c
c     function three body dipole-dipole-dipole fourth order
c
      fd4=ad41*(1.0/(rar0ar1*rar1ar2)**6)*(1.0+cos2*cos2)+
     +ad42*1.0/((rar1ar2*rar0ar2)**6)*(1.0+cos3*cos3)+
     +ad43*1.0/((rar0ar2*rar0ar1)**6)*(1.0+cos1*cos1)
c
c     function three body dipole-quadrupole-quadrupole
c
      fdqq=adqq*15.0/(64.0*(rar1ar2**5)*(rar0ar1*rar0ar2)**4)*
     +(3.0*(cos1+5.0*cos3fi1)+20.0*cosangledif2*(1.0-3.0*cos2fi1)+
     +70.0*cos2angledif2*cos1)
c
c     function three body quadrupole-dipole-quadrupole
c
      fqdq=aqdq*15.0/(64.0*(rar0ar2**5)*(rar1ar2*rar0ar1)**4)*
     +(3.0*(cos2+5.0*cos3fi2)+20.0*cosangledif3*(1.0-3.0*cos2fi2)+
     +70.0*cos2angledif3*cos2)
c
c     function three body quadrupole-quadrupole-dipole
c
      fqqd=aqqd*15.0/(64.0*(rar0ar1**5)*(rar1ar2*rar0ar2)**4)*
     +(3.0*(cos3+5.0*cos3fi3)+20.0*cosangledif1*(1.0-3.0*cos2fi3)+
     +70.0*cos2angledif1*cos3)
c
c     function three body quadrupole-quadrupole-quadrupole
c
      fqqq=aqqq*15.0/(128.0*(rar0ar1*rar1ar2*rar0ar2)**5)*
     +(-27.0+220.0*cos1*cos2*cos3+490.0*cos2fi1*cos2fi2*cos2fi3+
     +175.0*(cos2angledif1+cos2angledif2+cos2angledif3))

c
c     make the sum at each step for each sub-term
c
c     variable test3b checks for high-energy 3b interactions
c
      test3b=fddd+fddq+fdqd+fqdd+fd4+fdqq+fqdq+fqqd+fqqq

c
      totalddd=totalddd+fddd
c
      totalddq=totalddq+fddq
      totaldqd=totaldqd+fdqd
      totalqdd=totalqdd+fqdd
c
      totald4=totald4+fd4
c
      totaldqq=totaldqq+fdqq
      totalqdq=totalqdq+fqdq
      totalqqd=totalqqd+fqqd
c
      totalqqq=totalqqq+fqqq

      na3ar=na3ar+1

90    continue
```

```
80    continue
70    continue
30    continue
60    continue
50    continue
40    continue
20    continue

      totararar1=totararar1+totalddd
      totararar2=totararar2+totalddq+totaldqd+totalqdd
      totararar3=totararar3+totalqdq+totalqqd+totaldqq
      totararar4=totararar4+totalqqq
      totararar5=totararar5+totald4
      sumararar= sumararar+totalddd+totalddq+totaldqd+
     +totalqdd+totalqdq+totalqqd+totaldqq+totalqqq+
     +totald4

10    continue

c      call resul3b(sumararar,totararar1,totararar2,
c    +totararar3,totararar4,totararar5)

      return
      end

      subroutine resul3b(sum,total1,total2,
     +total3,total4,total5)

      implicit double precision(a-h,o-z)

      open(unit=41,file='3bresult.tem')

      write(41,*)'total ddd ',total1
      write(41,*)'total ddq ',total2
      write(41,*)'total dqq ',total3
      write(41,*)'total qqq ',total4
      write(41,*)'total d4  ',total5
      write(41,*)'total 3b  ',sum

      print*,'total ddd ',total1
      print*,'total ddq ',total2
      print*,'total dqq ',total3
      print*,'total qqq ',total4
      print*,'total d4  ',total5
      print*,'total 3b  ',sum

      return
      end

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

      subroutine pot3bararo(sumararo,totararo1,totararo2,
     +totararo3,totararo4,totararo5)

      implicit double precision(a-h,o-z)

      common /param3bararo/oddd,oddq,odqd,oqdd,odqq,oqdq,oqqd,
     +                oqqq,od41,od42,od43
```

```
      common /arpos/ nar,naruc, xar(1000),yar(1000),zar(1000)
      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arpostem/xartem,yartem,zartem
      common /oxpos/ nox,xox(5184),yox(5184),zox(5184)
      common /oxpostem/xoxtem,yoxtem,zoxtem
      common /cut2b3b/ hcut
      common /highenergy/ en3ar,en3o,en3si
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
      common/bounds/xlow,xhigh,ylow,yhigh,zlow,zhigh
      common/check/natlj,nrtlj,nabfw,nrbfw,naelj,nrelj,na3ar,nr3ar,
     +na3o,nr3o,na3si,nr3si

      na3o=0
      nr3o=0
      sumararo=0.0
      totararo1=0.0
      totararo2=0.0
      totararo3=0.0
      totararo4=0.0
      totararo5=0.0

      do 10 il=1,nardum

      xar0=xar(il)
      yar0=yar(il)
      zar0=zar(il)

      totalddd=0.00
      totalddq=0.00
      totaldqd=0.00
      totalqdd=0.00
      totaldqq=0.00
      totalqdq=0.00
      totalqqd=0.00
      totalqqq=0.00
      totald4=0.000

      do 20 im=il+1,nar

      do 40 i=-nreplica,nreplica
      do 50 j=-nreplica,nreplica
      do 60 k=-nreplica,nreplica

c     Convert argon coordinates to reduced units

      xar(im)=xar(im)*(rbohr/(3*auc))
      yar(im)=yar(im)*(rbohr/(3*buc))
      zar(im)=zar(im)*(rbohr/(3*cuc))

c     Store coordinates of replicas of original simulation box.

      xartem=(3*auc/rbohr)*(xar(im)+i)
      yartem=(3*buc/rbohr)*(yar(im)+j)
      zartem=(3*cuc/rbohr)*(zar(im)+k)

      xar(im)=(3*auc/rbohr)*xar(im)
      yar(im)=(3*buc/rbohr)*yar(im)
      zar(im)=(3*cuc/rbohr)*zar(im)
```

```
c
      xar1=xartem
      yar1=yartem
      zar1=zartem
      dif1x=xar1-xar0
      dif1y=yar1-yar0
      dif1z=zar1-zar0
      rar0ar1=sqrt(dif1x*dif1x+dif1y*dif1y+dif1z*dif1z)
c
      if(rar0ar1.gt.hcut) then
      nr3o=nr3o+1
      go to 60
      endif
c
      do 30 in=1,nox

      do 70 l=-nreplica,nreplica
      do 80 m=-nreplica,nreplica
      do 90 n=-nreplica,nreplica

c     Convert argon coordinates to reduced units

      xox(in)=xox(in)*(rbohr/(3*auc))
      yox(in)=yox(in)*(rbohr/(3*buc))
      zox(in)=zox(in)*(rbohr/(3*cuc))

c     Store coordinates of replicas of original simulation box.

      xoxtem=(3*auc/rbohr)*(xox(in)+l)
      yoxtem=(3*buc/rbohr)*(yox(in)+m)
      zoxtem=(3*cuc/rbohr)*(zox(in)+n)

      xox(in)=(3*auc/rbohr)*xox(in)
      yox(in)=(3*buc/rbohr)*yox(in)
      zox(in)=(3*cuc/rbohr)*zox(in)
c
      xo2=xoxtem
      yo2=yoxtem
      zo2=zoxtem
      dif2x=xo2-xar0
      dif2y=yo2-yar0
      dif2z=zo2-zar0
c
      dif3x=xo2-xar1
      dif3y=yo2-yar1
      dif3z=zo2-zar1
      rar0o2=sqrt(dif2x*dif2x+dif2y*dif2y+dif2z*dif2z)
      rar1o2=sqrt(dif3x*dif3x+dif3y*dif3y+dif3z*dif3z)
c
      if(rar0o2.gt.hcut) then
      nr3o=nr3o+1
      go to 90
      endif
c
      cos1=(dif1x*dif2x+dif1y*dif2y+dif1z*dif2z)/(rar0ar1*rar0o2)
      cos2=(-dif1x*dif3x-dif1y*dif3y-dif1z*dif3z)/(rar0ar1*rar1o2)
      cos3=(dif2x*dif3x+dif2y*dif3y+dif2z*dif3z)/(rar0o2*rar1o2)
c
```

```
      cosangledif1=cos1*cos2+sqrt(abs((1-cos1**2)*(1-cos2**2)))
      cosangledif2=cos2*cos3+sqrt(abs((1-cos2**2)*(1-cos3**2)))
      cosangledif3=cos3*cos1+sqrt(abs((1-cos3**2)*(1-cos1**2)))
c
      cos3fi1=-3*cos1+4*cos1**3
      cos3fi2=-3*cos2+4*cos2**3
      cos3fi3=-3*cos3+4*cos3**3
c
      cos2fi1=2*cos1**2-1
      cos2fi2=2*cos2**2-1
      cos2fi3=2*cos3**2-1
c
      cos2angledif1=2*(cosangledif1)**2-1
      cos2angledif2=2*(cosangledif2)**2-1
      cos2angledif3=2*(cosangledif3)**2-1
c
      cos4fi1=8*cos1**3*sqrt(abs(1-cos1**2))-4*cos1*sqrt(abs(1-cos1**2))
      cos4fi2=8*cos2**3*sqrt(abs(1-cos2**2))-4*cos2*sqrt(abs(1-cos2**2))
      cos4fi3=8*cos3**3*sqrt(abs(1-cos3**2))-4*cos3*sqrt(abs(1-cos3**2))

c     in all following functions, the order ar0,ar1,o2 is considered
c
c     function three body dipole-dipole-dipole
c
      fddd=3.0*oddd*(1.0+3.0*cos1*cos2*cos3)*
     +(rar0ar1*rar0o2*rar1o2)**(-3.0)
c
c     function three body dipole-dipole-quadrupole
c
      fddq=oddq*3.0/(16.0*(rar0ar1**3)*(rar0o2*rar1o2)**4)*
     +((9.0*cos3-25.0*cos3fi3)+6.0*cosangledif1*(3.0+5.0*cos2fi3))
c
c     function three body dipole-quadrupole-dipole
c
      fdqd=odqd*3.0/(16.0*(rar0o2**3)*(rar1o2*rar0ar1)**4)*
     +((9.0*cos2-25.0*cos3fi2)+6.0*cosangledif3*(3.0+5.0*cos2fi2))
c
c     function three body quadrupole-dipole-dipole
c
      fqdd=oqdd*3.0/(16.0*(rar1o2**3)*(rar0o2*rar0ar1)**4)*
     +((9.0*cos1-25.0*cos3fi1)+6.0*cosangledif2*(3.0+5.0*cos2fi1))
c
c     function three body dipole-dipole-dipole fourth order
c
      fd4=od41*(1.0/(rar0ar1*rar1o2)**6)*(1.0+cos2*cos2)+
     +od42*1.0/((rar1o2*rar0o2)**6)*(1.0+cos3*cos3)+
     +od43*1.0/((rar0o2*rar0ar1)**6)*(1.0+cos1*cos1)
c
c     function three body dipole-quadrupole-quadrupole
c
      fdqq=odqq*15.0/(64.0*(rar1o2**5)*(rar0ar1*rar0o2)**4)*
     +(3.0*(cos1+5.0*cos3fi1)+20.0*cosangledif2*(1.0-3.0*cos2fi1)+
     +70.0*cos2angledif2*cos1)
c
c     function three body quadrupole-dipole-quadrupole
c
      fqdq=oqdq*15.0/(64.0*(rar0o2**5)*(rar1o2*rar0ar1)**4)*
     +(3.0*(cos2+5.0*cos3fi2)+20.0*cosangledif3*(1.0-3.0*cos2fi2)+
```

```
     +70.0*cos2angledif3*cos2)
c
c     function three body quadrupole-quadrupole-dipole
c
      fqqd=oqqd*15.0/(64.0*(rar0ar1**5)*(rar1o2*rar0o2)**4)*
     +(3.0*(cos3+5.0*cos3fi3)+20.0*cosangledif1*(1.0-3.0*cos2fi3)+
     +70.0*cos2angledif1*cos3)
c
c     function three body quadrupole-quadrupole-quadrupole
c
      fqqq=oqqq*15.0/(128.0*(rar0ar1*rar1o2*rar0o2)**5)*
     +(-27.0+220.0*cos1*cos2*cos3+490.0*cos2fi1*cos2fi2*cos2fi3+
     +175.0*(cos2angledif1+cos2angledif2+cos2angledif3))

c     variable test3b checks for high-energy 3b interactions

      test3b=fddd+fddq+fdqd+fqdd+fd4+fdqq+fqdq+fqqd+fqqq


c     make the sum at each step for each sub-term
c
      totalddd=totalddd+fddd
c
      totalddq=totalddq+fddq
      totaldqd=totaldqd+fdqd
      totalqdd=totalqdd+fqdd
c
      totald4=totald4+fd4
c
      totaldqq=totaldqq+fdqq
      totalqdq=totalqdq+fqdq
      totalqqd=totalqqd+fqqd
c
      totalqqq=totalqqq+fqqq
c
      na3o=na3o+1

90    continue
80    continue
70    continue
30    continue
60    continue
50    continue
40    continue
20    continue

      totararo1=totararo1+totalddd
      totararo2=totararo2+totalddq+totaldqd+totalqdd
      totararo3=totararo3+totalqdq+totalqqd+totaldqq
      totararo4=totararo4+totalqqq
      totararo5=totararo5+totald4
      sumararo= sumararo+totalddd+totalddq+totaldqd+
     +totalqdd+totalqdq+totalqqd+totaldqq+totalqqq+
     +totald4

10    continue

c     call resul3b(sumararo,totararo1,totararo2,
```

```
c     +totararo3,totararo4,totararo5)

      return
      end

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

      subroutine pot3bararsi(sumararsi,totararsi1,totararsi2,
     +totararsi3,totararsi4,totararsi5)

      implicit double precision(a-h,o-z)

      common /param3bararsi/ sddd,sddq,sdqd,sqdd,sdqq,sqdq,sqqd,
     +                 sqqq,sd41,sd42,sd43
      common /cut2b3b/ hcut
      common /highenergy/ en3ar,en3o,en3si
      common/bounds/xlow,xhigh,ylow,yhigh,zlow,zhigh
      common /sipos/ nsi,xsi(2592),ysi(2592),zsi(2592)
      common /sipostem/ xsitem,ysitem,zsitem
      common /arpos/ nar,naruc, xar(1000),yar(1000),zar(1000)
      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arpostem/xartem,yartem,zartem
      common/check/natlj,nrtlj,nabfw,nrbfw,naelj,nrelj,na3ar,nr3ar,
     +na3o,nr3o,na3si,nr3si
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag

      na3si=0
      nr3si=0
      sumararsi=0.0
      totararsi1=0.0
      totararsi2=0.0
      totararsi3=0.0
      totararsi4=0.0
      totararsi5=0.0

      do 10 il=1,nardum

      xar0=xar(il)
      yar0=yar(il)
      zar0=zar(il)

      totalddd=0.00
      totalddq=0.00
      totaldqd=0.00
      totalqdd=0.00
      totaldqq=0.00
      totalqdq=0.00
      totalqqd=0.00
      totalqqq=0.00
      totald4=0.000

      do 20 im=(il+1),nar

      do 40 i=-nreplica,nreplica
      do 50 j=-nreplica,nreplica
      do 60 k=-nreplica,nreplica

c     Convert argon coordinates to reduced units
```

```
      xar(im)=xar(im)*(rbohr/(3*auc))
      yar(im)=yar(im)*(rbohr/(3*buc))
      zar(im)=zar(im)*(rbohr/(3*cuc))

c     Store coordinates of replicas of original simulation box.

      xartem=(3*auc/rbohr)*(xar(im)+i)
      yartem=(3*buc/rbohr)*(yar(im)+j)
      zartem=(3*cuc/rbohr)*(zar(im)+k)

      xar(im)=(3*auc/rbohr)*xar(im)
      yar(im)=(3*buc/rbohr)*yar(im)
      zar(im)=(3*cuc/rbohr)*zar(im)

      xar1=xartem
      yar1=yartem
      zar1=zartem
      dif1x=xar1-xar0
      dif1y=yar1-yar0
      dif1z=zar1-zar0
      rar0ar1=sqrt(dif1x*dif1x+dif1y*dif1y+dif1z*dif1z)
c
      if(rar0ar1.gt.hcut) then
      nr3si=nr3si+1
      go to 60
      endif
c
      do 30 in=1,nsi

      do 70 l=-nreplica,nreplica
      do 80 m=-nreplica,nreplica
      do 90 n=-nreplica,nreplica

c     Convert silicon coordinates to reduced units

      xsi(in)=xsi(in)*(rbohr/(3*auc))
      ysi(in)=ysi(in)*(rbohr/(3*buc))
      zsi(in)=zsi(in)*(rbohr/(3*cuc))

c     Store coordinates of replicas of original simulation box.

      xsitem=(3*auc/rbohr)*(xsi(in)+l)
      ysitem=(3*buc/rbohr)*(ysi(in)+m)
      zsitem=(3*cuc/rbohr)*(zsi(in)+n)

      xsi(in)=(3*auc/rbohr)*xsi(in)
      ysi(in)=(3*buc/rbohr)*ysi(in)
      zsi(in)=(3*cuc/rbohr)*zsi(in)

      xsi2=xsitem
      ysi2=ysitem
      zsi2=zsitem
      dif2x=xsi2-xar0
      dif2y=ysi2-yar0
      dif2z=zsi2-zar0
c
      dif3x=xsi2-xar1
      dif3y=ysi2-yar1
```

```
      dif3z=zsi2-zar1
      rar0si2=sqrt(dif2x*dif2x+dif2y*dif2y+dif2z*dif2z)
      rar1si2=sqrt(dif3x*dif3x+dif3y*dif3y+dif3z*dif3z)
c
      if(rar0si2.gt.hcut) then
      nr3si=nr3si+1
      go to 90
      endif
c
      cos1=(dif1x*dif2x+dif1y*dif2y+dif1z*dif2z)/(rar0ar1*rar0si2)
      cos2=(-dif1x*dif3x-dif1y*dif3y-dif1z*dif3z)/(rar0ar1*rar1si2)
      cos3=(dif2x*dif3x+dif2y*dif3y+dif2z*dif3z)/(rar0si2*rar1si2)
c
      cosangledif1=cos1*cos2+sqrt(abs((1-cos1**2)*(1-cos2**2)))
      cosangledif2=cos2*cos3+sqrt(abs((1-cos2**2)*(1-cos3**2)))
      cosangledif3=cos3*cos1+sqrt(abs((1-cos3**2)*(1-cos1**2)))
c
      cos3fi1=-3*cos1+4*cos1**3
      cos3fi2=-3*cos2+4*cos2**3
      cos3fi3=-3*cos3+4*cos3**3
c
      cos2fi1=2*cos1**2-1
      cos2fi2=2*cos2**2-1
      cos2fi3=2*cos3**2-1
c
      cos2angledif1=2*(cosangledif1)**2-1
      cos2angledif2=2*(cosangledif2)**2-1
      cos2angledif3=2*(cosangledif3)**2-1
c
      cos4fi1=8*cos1**3*sqrt(abs(1-cos1**2))-4*cos1*sqrt(abs(1-cos1**2))
      cos4fi2=8*cos2**3*sqrt(abs(1-cos2**2))-4*cos2*sqrt(abs(1-cos2**2))
      cos4fi3=8*cos3**3*sqrt(abs(1-cos3**2))-4*cos3*sqrt(abs(1-cos3**2))

c     in all following functions, the order ar0,ar1,o2 is considered
c
c     function three body dipole-dipole-dipole
c
      fddd=3.0*sddd*(1.0+3.0*cos1*cos2*cos3)*
     +(rar0ar1*rar0si2*rar1si2)**(-3.0)
c
c     function three body dipole-dipole-quadrupole
c
      fddq=sddq*3.0/(16.0*(rar0ar1**3)*(rar0si2*rar1si2)**4)*
     +((9.0*cos3-25.0*cos3fi3)+6.0*cosangledif1*(3.0+5.0*cos2fi3))
c
c     function three body dipole-quadrupole-dipole
c
      fdqd=sdqd*3.0/(16.0*(rar0si2**3)*(rar1si2*rar0ar1)**4)*
     +((9.0*cos2-25.0*cos3fi2)+6.0*cosangledif3*(3.0+5.0*cos2fi2))
c
c     function three body quadrupole-dipole-dipole
c
      fqdd=sqdd*3.0/(16.0*(rar1si2**3)*(rar0si2*rar0ar1)**4)*
     +((9.0*cos1-25.0*cos3fi1)+6.0*cosangledif2*(3.0+5.0*cos2fi1))
c
c     function three body dipole-dipole-dipole fourth order
c
      fd4=sd41*(1.0/(rar0ar1*rar1si2)**6)*(1.0+cos2*cos2)+
```

```
     +sd42*1.0/((rar1si2*rar0si2)**6)*(1.0+cos3*cos3)+
     +sd43*1.0/((rar0si2*rar0ar1)**6)*(1.0+cos1*cos1)
c
c     function three body dipole-quadrupole-quadrupole
c
      fdqq=sdqq*15.0/(64.0*(rar1si2**5)*(rar0ar1*rar0si2)**4)*
     +(3.0*(cos1+5.0*cos3fi1)+20.0*cosangledif2*(1.0-3.0*cos2fi1)+
     +70.0*cos2angledif2*cos1)
c
c     function three body quadrupole-dipole-quadrupole
c
      fqdq=sqdq*15.0/(64.0*(rar0si2**5)*(rar1si2*rar0ar1)**4)*
     +(3.0*(cos2+5.0*cos3fi2)+20.0*cosangledif3*(1.0-3.0*cos2fi2)+
     +70.0*cos2angledif3*cos2)
c
c     function three body quadrupole-quadrupole-dipole
c
      fqqd=sqqd*15.0/(64.0*(rar0ar1**5)*(rar1si2*rar0si2)**4)*
     +(3.0*(cos3+5.0*cos3fi3)+20.0*cosangledif1*(1.0-3.0*cos2fi3)+
     +70.0*cos2angledif1*cos3)
c
c     function three body quadrupole-quadrupole-quadrupole
c
      fqqq=sqqq*15.0/(128.0*(rar0ar1*rar1si2*rar0si2)**5)*
     +(-27.0+220.0*cos1*cos2*cos3+490.0*cos2fi1*cos2fi2*cos2fi3+
     +175.0*(cos2angledif1+cos2angledif2+cos2angledif3))

c
c     variable test3b checks for high-energy 3b interactions

      test3b=fddd+fddq+fdqd+fqdd+fd4+fdqq+fqdq+fqqd+fqqq

c     make the sum at each step for each sub-term
c
      totalddd=totalddd+fddd
c
      totalddq=totalddq+fddq
      totaldqd=totaldqd+fdqd
      totalqdd=totalqdd+fqdd
c
      totald4=totald4+fd4
c
      totaldqq=totaldqq+fdqq
      totalqdq=totalqdq+fqdq
      totalqqd=totalqqd+fqqd
c
      totalqqq=totalqqq+fqqq
      na3si=na3si+1

90    continue
80    continue
70    continue
30    continue
60    continue
50    continue
40    continue
20    continue
```

```
      totararsi1=totararsi1+totalddd
      totararsi2=totararsi2+totalddq+totaldqd+totalqdd
      totararsi3=totararsi3+totalqdq+totalqqd+totaldqq
      totararsi4=totararsi4+totalqqq
      totararsi5=totararsi5+totald4
      sumararsi= sumararsi+totalddd+totalddq+totaldqd+
     +totalqdd+totalqdq+totalqqd+totaldqq+totalqqq+
     +totald4

10    continue

c      call resul3b(sumararsi,totararsi1,totararsi2,
c     +totararsi3,totararsi4,totararsi5)

      return
      end

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

      subroutine convertok(something)
      implicit double precision(a-h,o-z)
      common /bfwparams/ a0arar,a1arar,a2arar,a3arar,a4arar,a5arar,
     +                   aarar,deltarar,epsilbfw,radbfw

      something=something*epsilbfw

      return
      end
```

```
      PROGRAM runanal

c     This program reads the output from a series of runs
c     and writes out a file called runanal.dat
c
c     Felix Fernandez-Alonso.
c     Imperial College
c     March 1993
c


      implicit double precision(a-h,o-z)

      common/params/nruns,ncalc
      common/values/nar,nardum(12,4)
      common/input/potLJarar(12,4),potarar(12,4),
     +potbfw(12,4),pot3ararar(12,4),
     +pot3araro(12,4),pot3ararsi(12,4),
     +totpot3b(12,4),totpotLJ(12,4),
     +totpotbfw(12,4),devLJ(12,4),
     +devbfw(12,4),p3bar(12,4),p3bsi(12,4),
     +p3bo(12,4),pdddar(12,4),pddqar(12,4),
     +pdqqar(12,4),pqqqar(12,4),pd40ar(12,4),
     +pdddo(12,4),pddqo(12,4),
     +pdqqo(12,4),pqqqo(12,4),pd40o(12,4),
     +pdddsi(12,4),pddqsi(12,4),pdqqsi(12,4),
     +pqqqsi(12,4),pd40si(12,4)

      nruns=12
      ncalc=4
      call readparams
      call initialise
      call mean
      call standardev
      call display
      end

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

      subroutine readparams

      implicit double precision(a-h,o-z)

      common/params/nruns,ncalc
      common/values/nar,nardum(12,4)
      common/input/potLJarar(12,4),potarar(12,4),
     +potbfw(12,4),pot3ararar(12,4),
     +pot3araro(12,4),pot3ararsi(12,4),
     +totpot3b(12,4),totpotLJ(12,4),
     +totpotbfw(12,4),devLJ(12,4),
     +devbfw(12,4),p3bar(12,4),p3bsi(12,4),
     +p3bo(12,4),pdddar(12,4),pddqar(12,4),
     +pdqqar(12,4),pqqqar(12,4),pd40ar(12,4),
     +pdddo(12,4),pddqo(12,4),
     +pdqqo(12,4),pqqqo(12,4),pd40o(12,4),
     +pdddsi(12,4),pddqsi(12,4),pdqqsi(12,4),
     +pqqqsi(12,4),pd40si(12,4)
```

```
      PROGRAM shanal

c     This program reads the output from the shell calculation
c     and writes out a file called analysis1.dat
c
c     Felix Fernandez-Alonso.
c     Imperial College.
c     March 1993
c


      implicit double precision(a-h,o-z)

      common/values/nar,ncalc,nardum(4)
      common/input/potarar(4),potLJarar(4),potbfw(4),pot3arararar(4),
     +pot3araro(4),pot3ararsi(4),pot3ar1(4),pot3ar2(4),pot3ar3(4),
     +pot3ar4(4),pot3ar5(4),pot3o1(4),pot3o2(4),pot3o3(4),pot3o4(4),
     +pot3o5(4),pot3si1(4),pot3si2(4),pot3si3(4),
     +pot3si4(4),pot3si5(4)
      common/output/totpot3b(4),totpotLJ(4),totpotbfw(4),devLJ(4),
     +devbfw(4),p3bar(4),p3bsi(4),p3bo(4),pdddar(4),
     +pddqar(4),pdqqar(4),pqqqar(4),pd40ar(4),pdddo(4),pddqo(4),
     +pdqqo(4),pqqqo(4),pd40o(4),pdddsi(4),pddqsi(4),pdqqsi(4),
     +pqqqsi(4),pd40si(4)

      ncalc=4
      call readparams
      call analyse
      call display
      end

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

      subroutine readparams

      implicit double precision(a-h,o-z)

      common/values/nar,ncalc,nardum(4)
      common/input/potarar(4),potLJarar(4),potbfw(4),pot3arararar(4),
     +pot3araro(4),pot3ararsi(4),pot3ar1(4),pot3ar2(4),pot3ar3(4),
     +pot3ar4(4),pot3ar5(4),pot3o1(4),pot3o2(4),pot3o3(4),pot3o4(4),
     +pot3o5(4),pot3si1(4),pot3si2(4),pot3si3(4),
     +pot3si4(4),pot3si5(4)


      open(unit=17,file='result.tem')


      read(17,*) nar

      do 10 i=1,ncalc

      read(17,*)
      read(17,*) nardum(i)
      read(17,*) potarar(i)
      read(17,*) potLJarar(i)
      read(17,*) potbfw(i)
      read(17,*) pot3arararar(i)
```

```
      read(17,*) pot3ar1(i)
      read(17,*) pot3ar2(i)
      read(17,*) pot3ar3(i)
      read(17,*) pot3ar4(i)
      read(17,*) pot3ar5(i)
      read(17,*) pot3araro(i)
      read(17,*) pot3o1(i)
      read(17,*) pot3o2(i)
      read(17,*) pot3o3(i)
      read(17,*) pot3o4(i)
      read(17,*) pot3o5(i)
      read(17,*) pot3ararsi(i)
      read(17,*) pot3si1(i)
      read(17,*) pot3si2(i)
      read(17,*) pot3si3(i)
      read(17,*) pot3si4(i)
      read(17,*) pot3si5(i)
      do 20 j=1,12
      read(17,*)
20    continue

10    continue

      return
      end


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

      subroutine analyse

      implicit double precision(a-h,o-z)

      common/values/nar,ncalc,nardum(4)
      common/input/potarar(4),potLJarar(4),potbfw(4),pot3ararar(4),
     +pot3araro(4),pot3ararsi(4),pot3ar1(4),pot3ar2(4),pot3ar3(4),
     +pot3ar4(4),pot3ar5(4),pot3o1(4),pot3o2(4),pot3o3(4),pot3o4(4),
     +pot3o5(4),pot3si1(4),pot3si2(4),pot3si3(4),
     +pot3si4(4),pot3si5(4)
      common/output/totpot3b(4),totpotLJ(4),totpotbfw(4),devLJ(4),
     +devbfw(4),p3bar(4),p3bsi(4),p3bo(4),pdddar(4),
     +pddqar(4),pdqqar(4),pqqqar(4),pd40ar(4),pdddo(4),pddqo(4),
     +pdqqo(4),pqqqo(4),pd40o(4),pdddsi(4),pddqsi(4),pdqqsi(4),
     +pqqqsi(4),pd40si(4)


c     calculates deviations from Lennard-Jones potential and
c     contributions from different regions in the unit cell.

c     also determines relative importance of three-body terms

c     evaluate total energies per argon atom

      do 10 i=1,ncalc


      potarar(i)=potarar(i)/nardum(i)
      potbfw(i)=potbfw(i)/nardum(i)
```

```
      potLJarar(i)=potLJarar(i)/nardum(i)
      pot3ararar(i)=pot3ararar(i)/nardum(i)
      pot3araro(i)=pot3araro(i)/nardum(i)
      pot3ararsi(i)=pot3ararsi(i)/nardum(i)

      totpot3b(i)=(pot3ararar(i)+pot3araro(i)+pot3ararsi(i))
      totpotLJ(i)=(potarar(i)+totpot3b(i))
      totpotbfw(i)=(potbfw(i)+totpot3b(i))

      devLJ(i)=100*(totpotLJ(i)-potLJarar(i))/potLJarar(i)
      devbfw(i)=100*(totpotbfw(i)-potLJarar(i))/potLJarar(i)

      p3bar(i)=100*pot3ararar(i)/totpot3b(i)
      p3bsi(i)=100*pot3ararsi(i)/totpot3b(i)
      p3bo(i)=100*pot3araro(i)/totpot3b(i)

      pdddar(i)=(100/nardum(i))*pot3ar1(i)/pot3ararar(i)
      pddqar(i)=(100/nardum(i))*pot3ar2(i)/pot3ararar(i)
      pdqqar(i)=(100/nardum(i))*pot3ar3(i)/pot3ararar(i)
      pqqqar(i)=(100/nardum(i))*pot3ar4(i)/pot3ararar(i)
      pd40ar(i)=(100/nardum(i))*pot3ar5(i)/pot3ararar(i)

      pdddo(i)=(100/nardum(i))*pot3o1(i)/pot3araro(i)
      pddqo(i)=(100/nardum(i))*pot3o2(i)/pot3araro(i)
      pdqqo(i)=(100/nardum(i))*pot3o3(i)/pot3araro(i)
      pqqqo(i)=(100/nardum(i))*pot3o4(i)/pot3araro(i)
      pd40o(i)=(100/nardum(i))*pot3o5(i)/pot3araro(i)

      pdddsi(i)=(100/nardum(i))*pot3si1(i)/pot3ararsi(i)
      pddqsi(i)=(100/nardum(i))*pot3si2(i)/pot3ararsi(i)
      pdqqsi(i)=(100/nardum(i))*pot3si3(i)/pot3ararsi(i)
      pqqqsi(i)=(100/nardum(i))*pot3si4(i)/pot3ararsi(i)
      pd40si(i)=(100/nardum(i))*pot3si5(i)/pot3ararsi(i)

10    continue

      return
      end

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

      subroutine display

      implicit double precision(a-h,o-z)

      common/values/nar,ncalc,nardum(4)
      common/input/potarar(4),potLJarar(4),potbfw(4),pot3ararar(4),
     +pot3araro(4),pot3ararsi(4),pot3ar1(4),pot3ar2(4),pot3ar3(4),
     +pot3ar4(4),pot3ar5(4),pot3o1(4),pot3o2(4),pot3o3(4),pot3o4(4),
     +pot3o5(4),pot3si1(4),pot3si2(4),pot3si3(4),
     +pot3si4(4),pot3si5(4)
      common/output/totpot3b(4),totpotLJ(4),totpotbfw(4),devLJ(4),
     +devbfw(4),p3bar(4),p3bsi(4),p3bo(4),pdddar(4),
     +pddqar(4),pdqqar(4),pqqqar(4),pd40ar(4),pdddo(4),pddqo(4),
     +pdqqo(4),pqqqo(4),pd40o(4),pdddsi(4),pddqsi(4),pdqqsi(4),
     +pqqqsi(4),pd40si(4)
```

```
      open(unit=18,file='anal.tem')


      write(18,*) nar

      do 10 i=1,ncalc

      write(18,*)
      write(18,*) i
      write(18,*) 'number of argon atoms in region.'
      write(18,*) nardum(i)
      write(18,*) 'energies: potLJ,potarar,potbfw,pot3ar,pot3o,pot3si.'
      write(18,*) potLJarar(i)
      write(18,*) potarar(i)
      write(18,*) potbfw(i)
      write(18,*) pot3arararar(i)
      write(18,*) pot3araro(i)
      write(18,*) pot3ararsi(i)
      write(18,*) 'deviations from Lennard-Jones: LJarar and BFW.'
      write(18,*) devLJ(i)
      write(18,*) devbfw(i)
      write(18,*) '% three-body forces: ar,o,si.'
      write(18,*) p3bar(i)
      write(18,*) p3bo(i)
      write(18,*) p3bsi(i)
      write(18,*) '% three-body terms:ddd,ddq,dqq,qqq,d4 / ar,o,si'
      write(18,*) pdddar(i)
      write(18,*) pddqar(i)
      write(18,*) pdqqar(i)
      write(18,*) pqqqar(i)
      write(18,*) pd40ar(i)
      write(18,*)
      write(18,*) pdddo(i)
      write(18,*) pddqo(i)
      write(18,*) pdqqo(i)
      write(18,*) pqqqo(i)
      write(18,*) pd40o(i)
      write(18,*)
      write(18,*) pdddsi(i)
      write(18,*) pddqsi(i)
      write(18,*) pdqqsi(i)
      write(18,*) pqqqsi(i)
      write(18,*) pd40si(i)

10    continue

      return
      end
```

```
      open(unit=17,file='rundata.tem')

      do 10 i=1,nruns

      read(17,*),nar

      do 20 j=1,ncalc

      read(17,*)
      read(17,*)
      read(17,*)
      read(17,*) nardum(i,j)
      read(17,*)
      read(17,*) potLJarar(i,j)
      read(17,*) potarar(i,j)
      read(17,*) potbfw(i,j)
      read(17,*) pot3ararar(i,j)
      read(17,*) pot3araro(i,j)
      read(17,*) pot3ararsi(i,j)
      read(17,*)
      read(17,*) devLJ(i,j)
      read(17,*) devbfw(i,j)
      read(17,*)
      read(17,*) p3bar(i,j)
      read(17,*) p3bo(i,j)
      read(17,*) p3bsi(i,j)
      read(17,*)
      read(17,*) pdddar(i,j)
      read(17,*) pddqar(i,j)
      read(17,*) pdqqar(i,j)
      read(17,*) pqqqar(i,j)
      read(17,*) pd40ar(i,j)
      read(17,*)
      read(17,*) pdddo(i,j)
      read(17,*) pddqo(i,j)
      read(17,*) pdqqo(i,j)
      read(17,*) pqqqo(i,j)
      read(17,*) pd40o(i,j)
      read(17,*)
      read(17,*) pdddsi(i,j)
      read(17,*) pddqsi(i,j)
      read(17,*) pdqqsi(i,j)
      read(17,*) pqqqsi(i,j)
      read(17,*) pd40si(i,j)

20    continue
10    continue

      return
      end

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

      subroutine mean
```

```
      implicit double precision(a-h,o-z)

      common/params/nruns,ncalc
      common/values/nar,nardum(12,4)
      common/input/potLJarar(12,4),potarar(12,4),
     +potbfw(12,4),pot3arararar(12,4),
     +pot3araro(12,4),pot3ararsi(12,4),
     +totpot3b(12,4),totpotLJ(12,4),
     +totpotbfw(12,4),devLJ(12,4),
     +devbfw(12,4),p3bar(12,4),p3bsi(12,4),
     +p3bo(12,4),pdddar(12,4),pddqar(12,4),
     +pdqqar(12,4),pqqqar(12,4),pd40ar(12,4),
     +pdddo(12,4),pddqo(12,4),
     +pdqqo(12,4),pqqqo(12,4),pd40o(12,4),
     +pdddsi(12,4),pddqsi(12,4),pdqqsi(12,4),
     +pqqqsi(12,4),pd40si(12,4)
      common/ave/ anardum(4),apotLJarar(4),apotarar(4),
     +apotbfw(4),apot3arararar(4),
     +apot3araro(4),apot3ararsi(4),
     +atotpot3b(4),atotpotLJ(4),
     +atotpotbfw(4),adevLJ(4),
     +adevbfw(4),ap3bar(4),ap3bsi(4),
     +ap3bo(4),apdddar(4),apddqar(4),
     +apdqqar(4),apqqqar(4),apd40ar(4),
     +apdddo(4),apddqo(4),
     +apdqqo(4),apqqqo(4),apd40o(4),
     +apdddsi(4),apddqsi(4),apdqqsi(4),
     +apqqqsi(4),apd40si(4)
      common/dev/ dnardum(4),dpotLJarar(4),dpotarar(4),
     +dpotbfw(4),dpot3arararar(4),
     +dpot3araro(4),dpot3ararsi(4),
     +dtotpot3b(4),dtotpotLJ(4),
     +dtotpotbfw(4),ddevLJ(4),
     +ddevbfw(4),dp3bar(4),dp3bsi(4),
     +dp3bo(4),dpdddar(4),dpddqar(4),
     +dpdqqar(4),dpqqqar(4),dpd40ar(4),
     +dpdddo(4),dpddqo(4),
     +dpdqqo(4),dpqqqo(4),dpd40o(4),
     +dpdddsi(4),dpddqsi(4),dpdqqsi(4),
     +dpqqqsi(4),dpd40si(4)


c     calculates means for all runs

      do 10 i=1,ncalc
      do 20 j=1,nruns

      anardum(i)=anardum(i)+nardum(j,i)
      apotLJarar(i)=apotLJarar(i)+potLJarar(j,i)
      apotarar(i)=apotarar(i)+potarar(j,i)
      apotbfw(i)=apotbfw(i)+potbfw(j,i)
      apot3arararar(i)=apot3arararar(i)+pot3arararar(j,i)
      apot3araro(i)=apot3araro(i)+pot3araro(j,i)
      apot3ararsi(i)=apot3ararsi(i)+pot3ararsi(j,i)
      adevLJ(i)=adevLJ(i)+devLJ(j,i)
      adevbfw(i)=adevbfw(i)+devbfw(j,i)
      ap3bar(i)=ap3bar(i)+p3bar(j,i)
      ap3bo(i)=ap3bo(i)+p3bo(j,i)
```

```
      ap3bsi(i)=ap3bsi(i)+p3bsi(j,i)
      apdddar(i)=apdddar(i)+pdddar(j,i)
      apddqar(i)=apddqar(i)+pddqar(j,i)
      apdqqar(i)=apdqqar(i)+pdqqar(j,i)
      apqqqar(i)=apqqqar(i)+pqqqar(j,i)
      apd40ar(i)=apd40ar(i)+pd40ar(j,i)
      apdddo(i)=apdddo(i)+pdddo(j,i)
      apddqo(i)=apddqo(i)+pddqo(j,i)
      apdqqo(i)=apdqqo(i)+pdqqo(j,i)
      apqqqo(i)=apqqqo(i)+pqqqo(j,i)
      apd40o(i)=apd40o(i)+pd40o(j,i)
      apdddsi(i)=apdddsi(i)+pdddsi(j,i)
      apddqsi(i)=apddqsi(i)+pddqsi(j,i)
      apdqqsi(i)=apdqqsi(i)+pdqqsi(j,i)
      apqqqsi(i)=apqqqsi(i)+pqqqsi(j,i)
      apd40si(i)=apd40si(i)+pd40si(j,i)

20    continue
10    continue

      do 30 i=1,ncalc

      anardum(i)=anardum(i)/nruns
      apotLJarar(i)=apotLJarar(i)/nruns
      apotarar(i)=apotarar(i)/nruns
      apotbfw(i)=apotbfw(i)/nruns
      apot3ararar(i)=apot3ararar(i)/nruns
      apot3araro(i)=apot3araro(i)/nruns
      apot3ararsi(i)=apot3ararsi(i)/nruns
      adevLJ(i)=adevLJ(i)/nruns
      adevbfw(i)=adevbfw(i)/nruns
      ap3bar(i)=ap3bar(i)/nruns
      ap3bo(i)=ap3bo(i)/nruns
      ap3bsi(i)=ap3bsi(i)/nruns
      apdddar(i)=apdddar(i)/nruns
      apddqar(i)=apddqar(i)/nruns
      apdqqar(i)=apdqqar(i)/nruns
      apqqqar(i)=apqqqar(i)/nruns
      apd40ar(i)=apd40ar(i)/nruns
      apdddo(i)=apdddo(i)/nruns
      apddqo(i)=apddqo(i)/nruns
      apdqqo(i)=apdqqo(i)/nruns
      apqqqo(i)=apqqqo(i)/nruns
      apd40o(i)=apd40o(i)/nruns
      apdddsi(i)=apdddsi(i)/nruns
      apddqsi(i)=apddqsi(i)/nruns
      apdqqsi(i)=apdqqsi(i)/nruns
      apqqqsi(i)=apqqqsi(i)/nruns
      apd40si(i)=apd40si(i)/nruns

30    continue

      return
      end

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

      subroutine standardev
```

```
      implicit double precision(a-h,o-z)

      common/params/nruns,ncalc
      common/values/nar,nardum(12,4)
      common/input/potLJarar(12,4),potarar(12,4),
     +potbfw(12,4),pot3ararar(12,4),
     +pot3araro(12,4),pot3ararsi(12,4),
     +totpot3b(12,4),totpotLJ(12,4),
     +totpotbfw(12,4),devLJ(12,4),
     +devbfw(12,4),p3bar(12,4),p3bsi(12,4),
     +p3bo(12,4),pdddar(12,4),pddqar(12,4),
     +pdqqar(12,4),pqqqar(12,4),pd40ar(12,4),
     +pdddo(12,4),pddqo(12,4),
     +pdqqo(12,4),pqqqo(12,4),pd40o(12,4),
     +pdddsi(12,4),pddqsi(12,4),pdqqsi(12,4),
     +pqqqsi(12,4),pd40si(12,4)
      common/ave/ anardum(4),apotLJarar(4),apotarar(4),
     +apotbfw(4),apot3ararar(4),
     +apot3araro(4),apot3ararsi(4),
     +atotpot3b(4),atotpotLJ(4),
     +atotpotbfw(4),adevLJ(4),
     +adevbfw(4),ap3bar(4),ap3bsi(4),
     +ap3bo(4),apdddar(4),apddqar(4),
     +apdqqar(4),apqqqar(4),apd40ar(4),
     +apdddo(4),apddqo(4),
     +apdqqo(4),apqqqo(4),apd40o(4),
     +apdddsi(4),apddqsi(4),apdqqsi(4),
     +apqqqsi(4),apd40si(4)
      common/dev/ dnardum(4),dpotLJarar(4),dpotarar(4),
     +dpotbfw(4),dpot3ararar(4),
     +dpot3araro(4),dpot3ararsi(4),
     +dtotpot3b(4),dtotpotLJ(4),
     +dtotpotbfw(4),ddevLJ(4),
     +ddevbfw(4),dp3bar(4),dp3bsi(4),
     +dp3bo(4),dpdddar(4),dpddqar(4),
     +dpdqqar(4),dpqqqar(4),dpd40ar(4),
     +dpdddo(4),dpddqo(4),
     +dpdqqo(4),dpqqqo(4),dpd40o(4),
     +dpdddsi(4),dpddqsi(4),dpdqqsi(4),
     +dpqqqsi(4),dpd40si(4)


c     calculates standard deviations for all runs

      do 10 i=1,ncalc
      do 20 j=1,nruns

      dnardum(i)=dnardum(i)+(anardum(i)-nardum(j,i))**2
      dpotLJarar(i)=dpotLJarar(i)+(apotLJarar(i)-potLJarar(j,i))**2
      dpotarar(i)=dpotarar(i)+(apotarar(i)-potarar(j,i))**2
      dpotbfw(i)=dpotbfw(i)+(apotbfw(i)-potbfw(j,i))**2
      dpot3ararar(i)=dpot3ararar(i)+(apot3ararar(i)-pot3ararar(j,i))**2
      dpot3araro(i)=dpot3araro(i)+(apot3araro(i)-pot3araro(j,i))**2
      dpot3ararsi(i)=dpot3ararsi(i)+(apot3ararsi(i)-pot3ararsi(j,i))**2
      ddevLJ(i)=ddevLJ(i)+(adevLJ(i)-devLJ(j,i))**2
      ddevbfw(i)=ddevbfw(i)+(adevbfw(i)-devbfw(j,i))**2
      dp3bar(i)=dp3bar(i)+(ap3bar(i)-p3bar(j,i))**2
```

```
      dp3bo(i)=dp3bo(i)+(ap3bo(i)-p3bo(j,i))**2
      dp3bsi(i)=dp3bsi(i)+(ap3bsi(i)-p3bsi(j,i))**2
      dpdddar(i)=dpdddar(i)+(apdddar(i)-pdddar(j,i))**2
      dpddqar(i)=dpddqar(i)+(apddqar(i)-pddqar(j,i))**2
      dpdqqar(i)=dpdqqar(i)+(apdqqar(i)-pdqqar(j,i))**2
      dpqqqar(i)=dpqqqar(i)+(apqqqar(i)-pqqqar(j,i))**2
      dpd40ar(i)=dpd40ar(i)+(apd40ar(i)-pd40ar(j,i))**2
      dpdddo(i)=dpdddo(i)+(apdddo(i)-pdddo(j,i))**2
      dpddqo(i)=dpddqo(i)+(apddqo(i)-pddqo(j,i))**2
      dpdqqo(i)=dpdqqo(i)+(apdqqo(i)-pdqqo(j,i))**2
      dpqqqo(i)=dpqqqo(i)+(apqqqo(i)-pqqqo(j,i))**2
      dpd40o(i)=dpd40o(i)+(apd40o(i)-pd40o(j,i))**2
      dpdddsi(i)=dpdddsi(i)+(apdddsi(i)-pdddsi(j,i))**2
      dpddqsi(i)=dpddqsi(i)+(apddqsi(i)-pddqsi(j,i))**2
      dpdqqsi(i)=dpdqqsi(i)+(apdqqsi(i)-pdqqsi(j,i))**2
      dpqqqsi(i)=dpqqqsi(i)+(apqqqsi(i)-pqqqsi(j,i))**2
      dpd40si(i)=dpd40si(i)+(apd40si(i)-pd40si(j,i))**2

20    continue
10    continue

      do 30 i=1,ncalc

      dnardum(i)=(dnardum(i)/(nruns-1))**0.5
      dpotLJarar(i)=(dpotLJarar(i)/(nruns-1))**0.5
      dpotarar(i)=(dpotarar(i)/(nruns-1))**0.5
      dpotbfw(i)=(dpotbfw(i)/(nruns-1))**0.5
      dpot3ararar(i)=(dpot3ararar(i)/(nruns-1))**0.5
      dpot3araro(i)=(dpot3araro(i)/(nruns-1))**0.5
      dpot3ararsi(i)=(dpot3ararsi(i)/(nruns-1))**0.5
      ddevLJ(i)=(ddevLJ(i)/(nruns-1))**0.5
      ddevbfw(i)=(ddevbfw(i)/(nruns-1))**0.5
      dp3bar(i)=(dp3bar(i)/(nruns-1))**0.5
      dp3bo(i)=(dp3bo(i)/(nruns-1))**0.5
      dp3bsi(i)=(dp3bsi(i)/(nruns-1))**0.5
      dpdddar(i)=(dpdddar(i)/(nruns-1))**0.5
      dpddqar(i)=(dpddqar(i)/(nruns-1))**0.5
      dpdqqar(i)=(dpdqqar(i)/(nruns-1))**0.5
      dpqqqar(i)=(dpqqqar(i)/(nruns-1))**0.5
      dpd40ar(i)=(dpd40ar(i)/(nruns-1))**0.5
      dpdddo(i)=(dpdddo(i)/(nruns-1))**0.5
      dpddqo(i)=(dpddqo(i)/(nruns-1))**0.5
      dpdqqo(i)=(dpdqqo(i)/(nruns-1))**0.5
      dpqqqo(i)=(dpqqqo(i)/(nruns-1))**0.5
      dpd40o(i)=(dpd40o(i)/(nruns-1))**0.5
      dpdddsi(i)=(dpdddsi(i)/(nruns-1))**0.5
      dpddqsi(i)=(dpddqsi(i)/(nruns-1))**0.5
      dpdqqsi(i)=(dpdqqsi(i)/(nruns-1))**0.5
      dpqqqsi(i)=(dpqqqsi(i)/(nruns-1))**0.5
      dpd40si(i)=(dpd40si(i)/(nruns-1))**0.5

30    continue

      return
      end

c     ***********************************************************************
```

```
subroutine display

implicit double precision(a-h,o-z)

common/params/nruns,ncalc
common/values/nar,nardum(12,4)
common/ave/ anardum(4),apotLJarar(4),apotarar(4),
+apotbfw(4),apot3ararar(4),
+apot3araro(4),apot3ararsi(4),
+atotpot3b(4),atotpotLJ(4),
+atotpotbfw(4),adevLJ(4),
+adevbfw(4),ap3bar(4),ap3bsi(4),
+ap3bo(4),apdddar(4),apddqar(4),
+apdqqar(4),apqqqar(4),apd40ar(4),
+apdddo(4),apddqo(4),
+apdqqo(4),apqqqo(4),apd40o(4),
+apdddsi(4),apddqsi(4),apdqqsi(4),
+apqqqsi(4),apd40si(4)
common/dev/ dnardum(4),dpotLJarar(4),dpotarar(4),
+dpotbfw(4),dpot3ararar(4),
+dpot3araro(4),dpot3ararsi(4),
+dtotpot3b(4),dtotpotLJ(4),
+dtotpotbfw(4),ddevLJ(4),
+ddevbfw(4),dp3bar(4),dp3bsi(4),
+dp3bo(4),dpdddar(4),dpddqar(4),
+dpdqqar(4),dpqqqar(4),dpd40ar(4),
+dpdddo(4),dpddqo(4),
+dpdqqo(4),dpqqqo(4),dpd40o(4),
+dpdddsi(4),dpddqsi(4),dpdqqsi(4),
+dpqqqsi(4),dpd40si(4)


open(unit=18,file='runres.dat')


write(18,*) 'ANALYSIS OF RESULTS'
write(18,*)
write(18,*) 'total number of argon atoms : ',nar
write(18,*) 'total number of calculations: ',ncalc
write(18,*) '    1. central unit cell.'
write(18,*) '    2. straight channels.'
write(18,*) '    3. intersections.'
write(18,*) '    4. zig-zag channels'
write(18,*) 'total number of runs        : ',nruns
write(18,*)
write(18,*) 'MEAN        STANDARD DEVIATION'

do 10 i=1,ncalc

write(18,*)
write(18,*) 'Calculation number: ',i
write(18,*) 'number of argon atoms in region.'
write(18,*)  anardum(i)
write(18,*) 'mean energies: potLJ,potarar,potbfw,'
write(18,*) 'pot3ar,pot3o,pot3si.'
write(18,*) apotLJarar(i),dpotLJarar(i)
write(18,*) apotarar(i),dpotarar(i)
write(18,*) apotbfw(i),dpotbfw(i)
```

```
      write(18,*) apot3ararar(i),dpot3ararar(i)
      write(18,*) apot3araro(i),dpot3araro(i)
      write(18,*) apot3ararsi(i),dpot3ararsi(i)
      write(18,*)
      write(18,*) 'deviations from Lennard-Jones: LJarar and BFW.'
      write(18,*) adevLJ(i),ddevLJ(i)
      write(18,*) adevbfw(i),ddevbfw(i)
      write(18,*)
      write(18,*) '% three-body forces: ar,o,si.'
      write(18,*) ap3bar(i),dp3bar(i)
      write(18,*) ap3bo(i),dp3bo(i)
      write(18,*) ap3bsi(i),dp3bsi(i)
      write(18,*)
      write(18,*) '% three-body terms:ddd,ddq,dqq,qqq,d4 / ar,o,si'
      write(18,*) apdddar(i),dpdddar(i)
      write(18,*) apddqar(i),dpddqar(i)
      write(18,*) apdqqar(i),dpddqar(i)
      write(18,*) apqqqar(i),dpqqqar(i)
      write(18,*) apd40ar(i),dpd40ar(i)
      write(18,*)
      write(18,*) apdddo(i),dpdddo(i)
      write(18,*) apddqo(i),dpddqo(i)
      write(18,*) apdqqo(i),dpdqqo(i)
      write(18,*) apqqqo(i),dpqqqo(i)
      write(18,*) apd40o(i),dpd40o(i)
      write(18,*)
      write(18,*) apdddsi(i),dpdddsi(i)
      write(18,*) apddqsi(i),dpddqsi(i)
      write(18,*) apdqqsi(i),dpdqqsi(i)
      write(18,*) apqqqsi(i),dpqqqsi(i)
      write(18,*) apd40si(i),dpd40si(i)

10    continue

      return
      end


      subroutine initialise

      implicit double precision(a-h,o-z)

      common/params/nruns,ncalc
      common/values/nar,nardum(12,4)
      common/ave/ anardum(4),apotLJarar(4),apotarar(4),
     +apotbfw(4),apot3ararar(4),
     +apot3araro(4),apot3ararsi(4),
     +atotpot3b(4),atotpotLJ(4),
     +atotpotbfw(4),adevLJ(4),
     +adevbfw(4),ap3bar(4),ap3bsi(4),
     +ap3bo(4),apdddar(4),apddqar(4),
     +apdqqar(4),apqqqar(4),apd40ar(4),
     +apdddo(4),apddqo(4),
     +apdqqo(4),apqqqo(4),apd40o(4),
     +apdddsi(4),apddqsi(4),apdqqsi(4),
     +apqqqsi(4),apd40si(4)
      common/dev/ dnardum(4),dpotLJarar(4),dpotarar(4),
     +dpotbfw(4),dpot3ararar(4),
```

```
+dpot3araro(4),dpot3ararsi(4),
+dtotpot3b(4),dtotpotLJ(4),
+dtotpotbfw(4),ddevLJ(4),
+ddevbfw(4),dp3bar(4),dp3bsi(4),
+dp3bo(4),dpdddar(4),dpddqar(4),
+dpdqqar(4),dpqqqar(4),dpd40ar(4),
+dpdddo(4),dpddqo(4),
+dpdqqo(4),dpqqqo(4),dpd40o(4),
+dpdddsi(4),dpddqsi(4),dpdqqsi(4),
+dpqqqsi(4),dpd40si(4)


 do 10 i=1,ncalc

 anardum(i)=0.0
 apotLJarar(i)=0.0
 apotarar(i)=0.0
 apotbfw(i)=0.0
 apot3ararar(i)=0.0
 apot3araro(i)=0.0
 apot3ararsi(i)=0.0
 adevLJ(i)=0.0
 adevbfw(i)=0.0
 ap3bar(i)=0.0
 ap3bo(i)=0.0
 ap3bsi(i)=0.0
 apdddar(i)=0.0
 apddqar(i)=0.0
 apdqqar(i)=0.0
 apqqqar(i)=0.0
 apd40ar(i)=0.0
 apdddo(i)=0.0
 apddqo(i)=0.0
 apdqqo(i)=0.0
 apqqqo(i)=0.0
 apd40o(i)=0.0
 apdddsi(i)=0.0
 apddqsi(i)=0.0
 apdqqsi(i)=0.0
 apqqqsi(i)=0.0
 apd40si(i)=0.0
 dnardum(i)=0.0
 dpotLJarar(i)=0.0
 dpotarar(i)=0.0
 dpotbfw(i)=0.0
 dpot3ararar(i)=0.0
 dpot3araro(i)=0.0
 dpot3ararsi(i)=0.0
 ddevLJ(i)=0.0
 ddevbfw(i)=0.0
 dp3bar(i)=0.0
 dp3bo(i)=0.0
 dp3bsi(i)=0.0
 dpdddar(i)=0.0
 dpddqar(i)=0.0
 dpdqqar(i)=0.0
 dpqqqar(i)=0.0
 dpd40ar(i)=0.0
```

```
      dpdddo(i)=0.0
      dpddqo(i)=0.0
      dpdqqo(i)=0.0
      dpqqqo(i)=0.0
      dpd40o(i)=0.0
      dpdddsi(i)=0.0
      dpddqsi(i)=0.0
      dpdqqsi(i)=0.0
      dpqqqsi(i)=0.0
      dpd40si(i)=0.0

10    continue

      return
      end
```

```
      PROGRAM anal3b
c
c     Program that analyses the three-body configurations selected
c     from a calculation
c     It calculates the distances and angles for a given configuration
c     of three spherical atoms.
c
c     Felix Fernandez-Alonso.
c     Imperial College.
c     March, 1993
c

      character*80 input,output
      common/filename/input,output

      call inputfile
      call readata
      call rancalcul
      call finresult
      end

      subroutine inputfile

      character*80 input,output
      common/filename/input,output
      character*80 x,y

      x='x3b.dat'
      y='x3bo.dat'
      input=x
      output=y
1     FORMAT(A)
      return
      end

      subroutine readata

      character*80 input,output
      common/filename/input,output
      common /values/ auc,buc,euc,rbohr
      common/data/x0(5000),y0(5000),z0(5000),x1(5000),
     + y1(5000),z1(5000),x2(5000),y2(5000),z2(5000),energy(5000),
     + nintera



      open(unit=30,file=input)

      read(30,*)
      read(30,*)
      read(30,*)
      read(30,*)

      i=1
      nintera=0
10     read(30,*,end=200) energy(i)
      read(30,*,end=200) x0(i),y0(i),z0(i)
      read(30,*,end=200) x1(i),y1(i),z1(i)
```

```
      read(30,*,end=20) x2(i),y2(i),z2(i)
      read(30,*,end=20)
      nintera=nintera+1
      i=i+1
      goto 10

20    return
      end


      subroutine rancalcul

      common/data/x0(5000),y0(5000),z0(5000),x1(5000),
     + y1(5000),z1(5000),x2(5000),y2(5000),z2(5000),energy(5000),
     + nintera
      common/result/r01(5000),r02(5000),r12(5000),cos1(5000),cos2(5000),
     + cos3(5000),ang1(5000),ang2(5000),ang3(5000)

      do 10 i=1,nintera

      dif1x=x1(i)-x0(i)
      dif1y=y1(i)-y0(i)
      dif1z=z1(i)-z0(i)
      r01(i)=sqrt(dif1x*dif1x+dif1y*dif1y+dif1z*dif1z)

      dif2x=x2(i)-x0(i)
      dif2y=y2(i)-y0(i)
      dif2z=z2(i)-z0(i)
c
      dif3x=x2(i)-x1(i)
      dif3y=y2(i)-y1(i)
      dif3z=z2(i)-z1(i)
      r02(i)=sqrt(dif2x*dif2x+dif2y*dif2y+dif2z*dif2z)
      r12(i)=sqrt(dif3x*dif3x+dif3y*dif3y+dif3z*dif3z)
c

      cos1(i)=(dif1x*dif2x+dif1y*dif2y+dif1z*dif2z)/
     +(r01(i)*r02(i))
      cos2(i)=(-dif1x*dif3x-dif1y*dif3y-dif1z*dif3z)/
     +(r01(i)*r12(i))
      cos3(i)=(dif2x*dif3x+dif2y*dif3y+dif2z*dif3z)/
     +(r02(i)*r12(i))

10    continue

      return
      end


      subroutine finresult

      character*80 input,output
      common/filename/input,output
      common/data/x0(5000),y0(5000),z0(5000),x1(5000),
     + y1(5000),z1(5000),x2(5000),y2(5000),z2(5000),energy(5000),
     + nintera
      common/result/r01(5000),r02(5000),r12(5000),cos1(5000),cos2(5000),
     + cos3(5000),ang1(5000),ang2(5000),ang3(5000)
```

```
      open(unit=40,file=output)

      write(40,*) 'NEW DATA SET STARTS'
      write(40,*)
      write(40,*) 'filename: ',output
      write(40,*)
      write(40,*)'Distances and angles for three-body'
      write(40,*)'configurations exceeding 1K'
      write(40,*)
      write(40,*)'source file: ',input
      write(40,*)
      write(40,*)'     '.          atom1     ','    atom2     ',
     + '     atom3'
      do 10 i=1,nintera
      write(40,*)'  '
      write(40,*)'Energy(K) ',energy(i)
      write(40,*)'Distances ',r01(i),r02(i),r12(i)
      write(40,*)'Cosines     ',cos1(i),cos2(i),cos3(i)

10    continue

      write(40,*)
      write(40,*)'END OF DATA SET'
      write(40,*)

      return
      end
```

Appendix IV

Incorporating three-body effects into a molecular simulation

Programs:

(1) **msimu6.f**: see appendix III

(2) **msimu8.f**: a slight modification of the code given for msimu6.f, this time allowing for creation and destruction of adsorbate particles.

(3) **shcalc6.f**: similarly, it is a modification of shcalc4.f. This time, an extra subroutine ('adjepsilon') finds the epsilon for the Lennard-Jones potential that gives the energy given by the Barker-fisher-Watts potential plus the three-body energy.

Calculation:

The procedure followed in this calculation could be split into two parts. Initially, one would start by calculating an initial guess for the epsilon following the procedure outlined in Appendix III, that is, running msimu8.f (no destruction or creation of adsorbate particles) and finding an effective epsilon for each configuration. This procedure was repeated a few times. The final result would be an average effective epsilon.

The second step was to include the second guess into a full scale GCMC simulation (msimu6.f). The only important piece of information was the isosteric heat, displayed in the data file 'annit5.res' and the final configurations, in the data file 'conit5.dat'. Simulations were of at least twenty million moves to ensure a reliable value for the isosteric heat.

The new configuration from the simulation was then used to first calculate an average effective epsilon and to then incorporate this new value into a new simulation. The procedure would be then repeated until the epsilon remained more or less constant.

```
      PROGRAM shcalc6
c
c     ******************************************************************
c
c       Two-body and Three-body Potential Energy Calculation
c
c       This program calculates two-body and three-body terms
c       of the dispersion potential for argon. It includes:
c         - Two-body arar dispersion potential (Barker-Fisher-Watts
c           Potential, true 2b Lennard-Jones and effective
c           Lennard-Jones).
c         - Three-body ararar, ararsi and araro for the central unit
c           cell.
c         - The calculation has been corrected for periodic
c           boundaries by replicating in 3D the 27 original unit cells
c           a number of times specified by the variable 'nreplica'.
c
c       It also adjusts the epsilon in the Lennard-Jones effective
c       potential so that it reproduces the sum of the Barker-Fisher-
c       two-body potential and the three-body terms.
c
c       It reads the following data files:
c         - 'parama.dat', parameter file with all the necessary constants
c           such as dispersion coefficients, long range cutoffs, etc
c         - zsmox.dat and zsmsi.dat, the silicon and oxygen coordinates
c           of zeolite silicalite-1.
c         - 'arpos.dat', the argon coordinates in the zeolite.
c
c       Final results are displayed in file 'results.tem' in standard
c       format.
c
c       The new epsilon for the effective Lennard-Jones is written in the
c       file ljpotsim.dat, ready to be used by the simulation.
c
c       Program analysis.f creates a second file with the data analysis
c
c
c     ******************************************************************
c
c       The explicit form of the two-body and three-body potentials can
c       be found in the following references:
c
c         -Two-body dispersion potential for argon:
c          J.A. Barker and A. Pompe, Aust. J. of Chem.,1968,21:1683-94
c          J.A. Barker, R.A. Fisher and R.O. Watts, Mol. Phys.,1971,
c          21: 657
c
c         -Two-body dispersion potential for krypton and xenon:
c          J.A. Barker, R.O. Watts, et al., J. Chem. Phys.,61(8),1974:
c          3081
c
c         -Three-body dispersion potentials:
c          B.M. Axilrod and E. Teller, J. Chem. Phys.,11(6),1943:299
c          R. J. Bell, J. Phys. B: Atom. Mol. Phys.,1970,3:751
c          K.T. Tang et al., J. Chem. Phys. 64(7),1976:3063
c          M.B. Doran et al., J. Phys. C: Solid St. Phys.,1971,4:307
c
```

```
c     ******************************************************************
c
c       Written by:
c       Felix Fernandez-Alonso, Imperial College.
c       February, March, April, May 1993
c
c     ******************************************************************



      common /param6e/ c6arar,c8arar,c10arar,c6aro,c8aro,c10aro,
     +                 c6arsi,c8arsi,c10arsi
      common /bfwparams/ a0arar,a1arar,a2arar,a3arar,a4arar,a5arar,
     +                 aarar,deltarar,epsilbfw,radbfw
      common /LJones/epsilar,sigmar
      common /adjust/contol,nmax,step
      common /LJones2b/epsilar2b,sigmar2b
      common /param3bararar/addd,addq,adqd,aqdd,adqq,aqdq,aqqd,
     +                  aqqq,ad41,ad42,ad43
      common /param3bararo/ oddd,oddq,odqd,oqdd,odqq,oqdq,oqqd,
     +                  oqqq,od41,od42,od43
      common /param3bararsi/ sddd,sddq,sdqd,sqdd,sdqq,sqdq,sqqd,
     +                  sqqq,sd41,sd42,sd43
      common /cut2b3b/ hcut
      common /repulsy/ areparar,areparo,areparsi,breparar,
     +                 breparo,breparsi
      common /oxpos/ nox,xox(5184),yox(5184),zox(5184)
      common /sipos/ nsi,xsi(2592),ysi(2592),zsi(2592)
      common /arpos/ nar,naruc,xar(1000),yar(1000),zar(1000)
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
      common/check/natlj,nrtlj,nabfw,nrbfw,naelj,nrelj,na3ar,nr3ar,
     +na3o,nr3o,na3si,nr3si
      common/intera/ntlj,nelj,nbfw,n3ar,n3o,n3si

      print*,'program POTENTIAL CALCULATION starts'

      call readparams
      call readLJparams
      call geometry
      call set3bararar
      call set3bararo
      call set3bararsi
      calcflag=0
      call potcalculuc
      call adjepsilon
      call writeLJparams
      print*,'end of calculation.'
      end

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

      subroutine readparams



      common /param6e/ c6arar,c8arar,c10arar,c6aro,c8aro,c10aro,
     +                 c6arsi,c8arsi,c10arsi
      common /bfwparams/ a0arar,a1arar,a2arar,a3arar,a4arar,a5arar,
```

```
+                aarar,deltarar,epsilbfw,radbfw
 common /LJones/epsilar,sigmar
 common /LJones2b/epsilar2b,sigmar2b
 common /param3bararar/ addd, addq,adqd,aqdd,adqq,aqdq,aqqd,
+                aqqq,ad41,ad42,ad43
 common /param3bararo/ oddd, oddq,odqd,oqdd,odqq,oqdq,oqqd,
+                oqqq,od41,od42,od43
 common /param3bararsi/ sddd, sddq,sdqd,sqdd,sdqq,sqdq,sqqd,
+                sqqq,sd41,sd42,sd43
 common /cut2b3b/ hcut
 common /highenergy/ en3ar,en3o,en3si
 common /repulsy/ areparar,areparo,areparsi,breparar,
+               breparo,breparsi
 common/bounds/xlow,xhigh,ylow,yhigh,zlow,zhigh
 common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
 common /ucsegments/ xuc1,xuc2,yuc1,yuc2
 common /adjust/contol,nmax,step

 open(unit=20,file='parama5.dat')

c   Long-range cutoff for 2b and 3b interactions.

 read(20,*) hcut

c   Number of replicas of the original 27 unit cells.

 read(20,*) nreplica

c   Lower and upper bounds to which the calculation is restricted
c   (in reduced units)

 read(20,*) xlow
 read(20,*) xhigh
 read(20,*) ylow
 read(20,*) yhigh
 read(20,*) zlow
 read(20,*) zhigh

c   BFW Two-body terms for ArAr, ArO and ArSi interactions.

 read(20,*) c6arar
 read(20,*) c8arar
 read(20,*) c10arar

 read(20,*) a0arar
 read(20,*) a1arar
 read(20,*) a2arar
 read(20,*) a3arar
 read(20,*) a4arar
 read(20,*) a5arar
 read(20,*) aarar
 read(20,*) deltarar
 read(20,*) epsilbfw
 read(20,*) radbfw

c   Lennard-Jones parameters, epsilon (in kelvin) and sigma (A),
c   for argon
```

```
c      read(20,*) epsilar
c      read(20,*) sigmar
       read(20,*) epsilar2b
       read(20,*) sigmar2b

c      Three-body terms for ArArAr (a), ArArO (o)
c      and ArArSi (s) interactions.

       read(20,*) addd
       read(20,*) aqdd
       read(20,*) addq
       read(20,*) adqd
       read(20,*) aqqd
       read(20,*) aqdq
       read(20,*) adqq
       read(20,*) aqqq
       read(20,*) ad41
       read(20,*) ad42
       read(20,*) ad43

       read(20,*) oddd
       read(20,*) oqdd
       read(20,*) oddq
       read(20,*) odqd
       read(20,*) oqqd
       read(20,*) oqdq
       read(20,*) odqq
       read(20,*) oqqq
       read(20,*) od41
       read(20,*) od42
       read(20,*) od43

       read(20,*) sddd
       read(20,*) sqdd
       read(20,*) sddq
       read(20,*) sdqd
       read(20,*) sqqd
       read(20,*) sqdq
       read(20,*) sdqq
       read(20,*) sqqq
       read(20,*) sd41
       read(20,*) sd42
       read(20,*) sd43

c      energy threshold for 3-body interactions.

       read(20,*) en3ar
       read(20,*) en3o
       read(20,*) en3si

c      Reduced coordinates of the straight channel,
c      intersection and zig-zag channel in the unit cell.

       read(20,*) xuc1
       read(20,*) xuc2
       read(20,*) yuc1
       read(20,*) yuc2
```

```
c     convergence tolerance for the epsilon adjustment.

      read(20,*) contol
      read(20,*) step
      read(20,*) nmax

c      print*,contol
c      print*,step
c      print*,nmax

c      print*, 'all parameters read in ...'

      return
      end

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

      subroutine readLJparams

      common /LJones/epsilar,sigmar

      open(unit=23,file='ljpotsim.dat')

      read(23,*) sigmar
      read(23,*) epsilar

      return
      end

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

      subroutine geometry



      common /oxpos/ nox,xox(5184),yox(5184),zox(5184)
      common /sipos/ nsi,xsi(2592),ysi(2592),zsi(2592)
      common /arpos/ nar,naruc, xar(1000),yar(1000),zar(1000)
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
      common/bounds/xlow,xhigh,ylow,yhigh,zlow,zhigh
      common /ucsegments/ xuc1,xuc2,yuc1,yuc2
      common /arposuc/ xaruc(50),yaruc(50),zaruc(50)

c     O and Si coords in zsmox and zsmsi are in fractional u.c. units.
c     27 unit cells with origin at the centre.

      open(unit=2,file='zsmox.dat')
      open(unit=3,file='zsmsi.dat')
      open(unit=4,file='arpos.dat')

      auc=20.07
      buc=19.92
      cuc=13.42
      rbohr=0.52917

c     read oxygen coordinates.

      nox=0
```

```
21    nox=nox+1
      read(2,*,end=23) x1,y1,z1
      xox(nox)=x1
      yox(nox)=y1
      zox(nox)=z1
      goto 21
23    nox=nox-1

c      print*,nox,' oxygen coordinates read in'

c     convert to bohr units
      do 10 j=1,nox
      xox(j)=auc*xox(j)/rbohr
      yox(j)=buc*yox(j)/rbohr
      zox(j)=cuc*zox(j)/rbohr
10    continue
c      print*,'oxygen coordinates converted to Bohr units'

c     read silicon coordinates.

      nsi=0
20    nsi=nsi+1
      read(3,*,end=30) x2,y2,z2
      xsi(nsi)=x2
      ysi(nsi)=y2
      zsi(nsi)=z2
      goto 20
30    nsi=nsi-1
c      print*,nsi,' silicon coordinates read in'

c     convert to bohr units
      do 40 j=1,nsi
      xsi(j)=auc*xsi(j)/rbohr
      ysi(j)=buc*ysi(j)/rbohr
      zsi(j)=cuc*zsi(j)/rbohr
40    continue
c      print*,'silicon coordinates converted to Bohr units'

c     read argon coordinates.
c     count and store how many atoms are located in central
c     unit cell.

      nar=0
      naruc=0
      narsch=0
      narint=0
      narzzc=0

55    nar=nar+1
      read(4,*,end=56) x3,y3,z3
      xar(nar)=x3
      yar(nar)=y3
      zar(nar)=z3

      if ((xar(nar).gt.xlow).and.(xar(nar).lt.xhigh)
     +.and.(yar(nar).gt.ylow).and.(yar(nar).lt.yhigh)
     +.and.(zar(nar).lt.zhigh).and.(zar(nar).gt.zlow)) then
```

```
      naruc=naruc+1
      xaruc(naruc)=xar(nar)
      yaruc(naruc)=yar(nar)
      zaruc(naruc)=zar(nar)

      endif

      goto 55

56    nar=nar-1


c     convert to bohr units

      do 57 j=1,nar
      xar(j)=auc*xar(j)/rbohr
      yar(j)=buc*yar(j)/rbohr
      zar(j)=cuc*zar(j)/rbohr

57    continue

      do 65 j=1,naruc
      xaruc(j)=auc*xaruc(j)/rbohr
      yaruc(j)=buc*yaruc(j)/rbohr
      zaruc(j)=cuc*zaruc(j)/rbohr

65    continue

c     print*,'argon coordinates converted to Bohr units'

      return
      end

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

      subroutine set3bararar



      common /param3bararar/ adddd, addq,adqd,aqdd,adqq,aqdq,aqqd,
     +                 aqqq,ad41,ad42,ad43


c     convert energy to k

      factk=3.158e5
      adddd=adddd*factk

      aqdd=aqdd*factk
      addq=addq*factk
      adqd=adqd*factk

      aqqd=aqqd*factk
      aqdq=aqdq*factk
      adqq=adqq*factk

      aqqq=aqqq*factk
```

```
      addd41=ad41*factk
      addd42=ad42*factk
      addd43=ad43*factk

      return
      end

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

      subroutine set3bararo



      common /param3bararo/ oddd, oddq,odqd,oqdd,odqq,oqdq,oqqd,
     +                oqqq,od41,od42,od43

c     convert energy to k

      factk=3.158e5
      oddd=oddd*factk

      oqdd=oqdd*factk
      oddq=oddq*factk
      odqd=odqd*factk

      oqqd=oqqd*factk
      oqdq=oqdq*factk
      odqq=odqq*factk

      oqqq=oqqq*factk

      oddd41=od41*factk
      oddd42=od42*factk
      oddd43=od43*factk

      return
      end

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

      subroutine set3bararsi



      common /param3bararsi/ sddd, sddq,sdqd,sqdd,sdqq,sqdq,sqqd,
     +                sqqq,sd41,sd42,sd43

c     convert energy to k

      factk=3.158e5
      sddd=sddd*factk

      sqdd=sqdd*factk
      sddq=sddq*factk
      sdqd=sdqd*factk

      sqqd=sqqd*factk
      sqdq=sqdq*factk
```

```
      sdqq=sdqq*factk

      sqqq=sqqq*factk

      sddd41=sd41*factk
      sddd42=sd42*factk
      sddd43=sd43*factk

      return
      end

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

      subroutine potcalculuc


      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arpos/ nar,naruc, xar(1000),yar(1000),zar(1000)
      common /arposuc/ xaruc(50),yaruc(50),zaruc(50)


      do 10 i=1,naruc

      xardum(i)=xaruc(i)
      yardum(i)=yaruc(i)
      zardum(i)=zaruc(i)

10     continue

      nardum=naruc

      call shiftarray
      call shcalcul

      return
      end

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

      subroutine shiftarray


      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arpos/ nar,naruc, xar(1000),yar(1000),zar(1000)
c
c     subroutine that rearranges the array of argon coordinates
c     to ensure proper counting of two-body and three-body
c     interactions.
c
c      print*,'in subroutine shift array'

      pos=1

      do 10 i=1,nar
      do 20 j=1,nardum
```

```
      if ((xar(i).eq.xardum(j)).and.(yar(i).eq.yardum(j)).and.
     +(zar(i).eq.zardum(j))) then

      xtem=xar(i)
      ytem=yar(i)
      ztem=zar(i)

      xar(i)=xar(pos)
      yar(i)=yar(pos)
      zar(i)=zar(pos)

      xar(pos)=xtem
      yar(pos)=ytem
      zar(pos)=ztem

      pos=pos+1

      goto 10

      endif

20    continue
10    continue

      return
      end

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

      subroutine adjepsilon

      common /adjust/contol,nmax,step
      common /LJones/epsilar,sigmar
      common/results/potarar,potLJarar,potbfw,pot3ararar,pot3araro,
     +pot3ararsi,pot3ar1,pot3ar2,pot3ar3,pot3ar4,pot3ar5,pot3o1,
     +pot3o2,pot3o3,pot3o4,pot3o5,pot3si1,pot3si2,pot3si3,
     +pot3si4,pot3si5

c
c     subroutine that find the effective epsilon which reproduces
c     the true energy (Barker-Fisher-Watts plus three-body).
c
      toten=potbfw+pot3ararar+pot3araro+pot3ararsi
      nloop=0

c      open(unit=37,file='epsres.tem')
c      write(37,*)'ITER      TOTAL ENERGY      LJ ENERGY      EPSILON'
c      write(37,*)
c      print*,'adjusting the epsilon'

      do 10 i=1,nmax

c      print*,'ITER: ',i,toten,potLJarar,epsilar
c      write(37,*)i,toten,potLJarar,epsilar
      if (abs(toten-potLJarar).lt.contol) then
      goto 20
      endif
      if ((toten-potLJarar).gt.0.0) then
```

```
      epsilar=epsilar-step
      else
      epsilar=epsilar+step
      endif

      potLJarar=0.0
      call potLJParar(potLJarar)

      nloop=nloop+1

10    continue

20    if (nloop.eq.nmax) then
c     print*,'convergence not reached'
      endif

      return
      end

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

      subroutine shcalcul



      common /param6c/ c6arar,c8arar,c10arar,c6aro,c8aro,c10aro,
     +              c6arsi,c8arsi,c10arsi
      common /param3bararar/ addd, addq,adqd,aqdd,adqq,aqdq,aqqd,
     +               aqqq,ad41,ad42,ad43
      common /param3bararo/ oddd, oddq,odqd,oqdd,odqq,oqdq,oqqd,
     +               oqqq,od41,od42,od43
      common /param3bararsi/ sddd, sddq,sdqd,sqdd,sdqq,sqdq,sqqd,
     +               sqqq,sd41,sd42,sd43
      common /cut2b3b/ hcut
      common /repulsy/ areparar,areparo,areparsi,breparar,
     +              breparo,breparsi
      common /oxpos/ nox,xox(5184),yox(5184),zox(5184)
      common /sipos/ nsi,xsi(2592),ysi(2592),zsi(2592)
      common /arpos/ nar,naruc,xar(1000),yar(1000),zar(1000)
      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
      common/results/potarar,potLJarar,potbfw,pot3ararar,pot3araro,
     +pot3ararsi,pot3ar1,pot3ar2,pot3ar3,pot3ar4,pot3ar5,pot3o1,
     +pot3o2,pot3o3,pot3o4,pot3o5,pot3si1,pot3si2,pot3si3,
     +pot3si4,pot3si5


c     potential calculation

c     initialise variables

      potarar=0.0
      potLJarar=0.0
      potbfw=0.0

      pot3ararar=0.0
      pot3ar1=0.0
      pot3ar2=0.0
```

```
      pot3ar3=0.0
      pot3ar4=0.0
      pot3ar5=0.0

      pot3araro=0.0
      pot3o1=0.0
      pot3o2=0.0
      pot3o3=0.0
      pot3o4=0.0
      pot3o5=0.0

      pot3ararsi=0.0
      pot3si1=0.0
      pot3si2=0.0
      pot3si3=0.0
      pot3si4=0.0
      pot3si5=0.0

c     call two-body interaction subroutines

c      print*,'calculating potential of interaction ...'

      call pot2barar(potarar)
      call potLJParar(potLJarar)
      call pot2bfw(potbfw)
      call convertok(potbfw)

c     call three-body interaction subroutines

      call pot3barararar(pot3ararar,pot3ar1,pot3ar2,pot3ar3,pot3ar4,
     +pot3ar5)
      call pot3bararo(pot3araro,pot3o1,pot3o2,pot3o3,pot3o4,
     +pot3o5)
      call pot3bararsi(pot3ararsi,pot3si1,pot3si2,pot3si3,pot3si4,
     +pot3si5)

      calcflag=calcflag+1

c      call finresul

      return
      end

      subroutine finresul


      common /arpos/ nar,naruc,xar(1000),yar(1000),zar(1000)
      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common/results/potarar,potLJarar,potbfw,pot3ararar,pot3araro,
     +pot3ararsi,pot3ar1,pot3ar2,pot3ar3,pot3ar4,pot3ar5,pot3o1,
     +pot3o2,pot3o3,pot3o4,pot3o5,pot3si1,pot3si2,pot3si3,
     +pot3si4,pot3si5
      common/check/natlj,nrtlj,nabfw,nrbfw,naelj,nrelj,na3ar,nr3ar,
     +na3o,nr3o,na3si,nr3si
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
```

```
      open(unit=17,file='result.tem')

      write(17,*) nar
      write(17,*) nardum
      write(17,*) potarar
      write(17,*) potbfw
      write(17,*) pot3ararar
      write(17,*) pot3araro
      write(17,*) pot3ararsi
      write(17,*)


      return
      end

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

      subroutine writeLJparams

      common /LJones/epsilar,sigmar
      common/results/potarar,potLJarar,potbfw,pot3ararar,pot3araro,
     +pot3ararsi,pot3ar1,pot3ar2,pot3ar3,pot3ar4,pot3ar5,pot3o1,
     +pot3o2,pot3o3,pot3o4,pot3o5,pot3si1,pot3si2,pot3si3,
     +pot3si4,pot3si5
c
c     writes the new effective Lennard-Jones parameters.
c
      toten=potbfw+pot3ararar+pot3ararsi+pot3araro
      open(unit=41,file='ljpot.tem')
      write(41,*) toten,potLJarar,epsilar

      return
      end

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

      subroutine pot2barar(sumarar)



      common /param6c/ c6arar,c8arar,c10arar,c6aro,c8aro,c10aro,
     +                 c6arsi,c8arsi,c10arsi
      common /cut2b3b/ heut
      common /repulsy/ areparar,areparo,areparsi,breparar,
     +                 breparo,breparsi
      common /arpos/ nar,naruc, xar(1000),yar(1000),zar(1000)
      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arpostem/xartem,yartem,zartem
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
      common/bounds/xlow,xhigh,ylow,yhigh,zlow,zhigh
      common/LJones2b/epsilar2b,sigmar2b
      common/check/natlj,nrtlj,nabfw,nrbfw,naelj,nrelj,na3ar,nr3ar,
     +na3o,nr3o,na3si,nr3si

      sumarar=0.0
      natlj=0
      nrtlj=0
```

```
      rbohr=0.52917
      sigmar2bB=sigmar2b/rbohr

      do 10 m=1,nardum

      x1=xar(m)
      y1=yar(m)
      z1=zar(m)

      do 20 n=m+1,nar

      do 30 i=-nreplica,nreplica
      do 40 j=-nreplica,nreplica
      do 50 k=-nreplica,nreplica

c     Convert argon coordinates to reduced units

      xar(n)=xar(n)*(rbohr/(3*auc))
      yar(n)=yar(n)*(rbohr/(3*buc))
      zar(n)=zar(n)*(rbohr/(3*cuc))

c     Store coordinates of replicas of original simulation box.

      xartem=(3*auc/rbohr)*(xar(n)+i)
      yartem=(3*buc/rbohr)*(yar(n)+j)
      zartem=(3*cuc/rbohr)*(zar(n)+k)

      xar(n)=(3*auc/rbohr)*xar(n)
      yar(n)=(3*buc/rbohr)*yar(n)
      zar(n)=(3*cuc/rbohr)*zar(n)

      xarx2=(xartem-x1)**2
      yary2=(yartem-y1)**2
      zarz2=(zartem-z1)**2
      xarx1=xartem-x1
      yary1=yartem-y1
      zarz1=zartem-z1

      r2arar=xarx2+yary2+zarz2
      r1arar=sqrt(r2arar)

      if (r1arar.gt.hcut) then
      nrtlj=nrtlj+1
      goto 50
      endif

      arar2b=0.0

      redr6=(r1arar/sigmar2bB)**6
      redr12=(r1arar/sigmar2bB)**12
      arar2b=4*epsilar2b*((1/redr12)-(1/redr6))
c
      sumarar=arar2b+sumarar
      natlj=natlj+1

50    continue
40    continue
30    continue
```

```
20    continue
10    continue

      return
      end

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

      subroutine potLJParar(sumLJarar)



      common /cut2b3b/ hcut
      common /arpos/ nar,naruc, xar(1000),yar(1000),zar(1000)
      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arpostem/xartem,yartem,zartem
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
      common/bounds/xlow,xhigh,ylow,yhigh,zlow,zhigh
      common /LJones/epsilar,sigmar
      common/check/natlj,nrtlj,nabfw,nrbfw,naelj,nrelj,na3ar,nr3ar,
     +na3o,nr3o,na3si,nr3si

      rbohr=0.52917
      sigmarB=sigmar/rbohr
      sumLJarar=0.0
      naelj=0
      nrelj=0

      do 10 m=1,nardum

      x1=xar(m)
      y1=yar(m)
      z1=zar(m)

      do 20 n=m+1,nar

      do 30 i=-nreplica,nreplica
      do 40 j=-nreplica,nreplica
      do 50 k=-nreplica,nreplica

c     Convert argon coordinates to reduced units

      xar(n)=xar(n)*(rbohr/(3*auc))
      yar(n)=yar(n)*(rbohr/(3*buc))
      zar(n)=zar(n)*(rbohr/(3*cuc))

c     Store coordinates of replicas of original simulation box.

      xartem=(3*auc/rbohr)*(xar(n)+i)
      yartem=(3*buc/rbohr)*(yar(n)+j)
      zartem=(3*cuc/rbohr)*(zar(n)+k)

      xar(n)=(3*auc/rbohr)*xar(n)
      yar(n)=(3*buc/rbohr)*yar(n)
      zar(n)=(3*cuc/rbohr)*zar(n)

      xarx2=(xartem-x1)**2
      yary2=(yartem-y1)**2
```

```
      zarz2=(zartem-z1)**2
      xarx1=xartem-x1
      yary1=yartem-y1
      zarz1=zartem-z1
      r2arar=xarx2+yary2+zarz2
      r1arar=sqrt(r2arar)
      ararLJ=0.0

      if (r1arar.gt.hcut) then
      nrelj=nrelj+1
      goto 50
      endif

      redr6=(r1arar/sigmarB)**6
      redr12=(r1arar/sigmarB)**12
      ararLJ=4*epsilar*((1/redr12)-(1/redr6))

      sumLJarar=ararLJ+sumLJarar
      naelj=naelj+1

50    continue
40    continue
30    continue
20    continue
10    continue

      return
      end

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

      subroutine pot2bfw(sumbfw)



      common /param6e/ c6arar,c8arar,c10arar,c6aro,c8aro,c10aro,
     +                 c6arsi,c8arsi,c10arsi
      common /bfwparams/ a0arar,a1arar,a2arar,a3arar,a4arar,a5arar,
     +                 aarar,deltarar,epsilbfw,radbfw
      common /cut2b3b/ hcut
      common /arpos/ nar,naruc, xar(1000),yar(1000),zar(1000)
      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arpostem/xartem,yartem,zartem
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
      common/bounds/xlow,xhigh,ylow,yhigh,zlow,zhigh
      common/check/natlj,nrtlj,nabfw,nrbfw,naelj,nrelj,na3ar,nr3ar,
     +na3o,nr3o,na3si,nr3si


      sumbfw=0.0
      nabfw=0
      nrbfw=0

      rbohr=0.52917
      radbfwB=radbfw/rbohr

      do 10 m=1,nardum
```

```
      x1=xar(m)
      y1=yar(m)
      z1=zar(m)

      do 20 n=m+1,nar

      do 30 i=-nreplica,nreplica
      do 40 j=-nreplica,nreplica
      do 50 k=-nreplica,nreplica

c     Convert argon coordinates to reduced units

      xar(n)=xar(n)*(rbohr/(3*auc))
      yar(n)=yar(n)*(rbohr/(3*buc))
      zar(n)=zar(n)*(rbohr/(3*cuc))

c     Store coordinates of replicas of original simulation box.

      xartem=(3*auc/rbohr)*(xar(n)+i)
      yartem=(3*buc/rbohr)*(yar(n)+j)
      zartem=(3*cuc/rbohr)*(zar(n)+k)

      xar(n)=(3*auc/rbohr)*xar(n)
      yar(n)=(3*buc/rbohr)*yar(n)
      zar(n)=(3*cuc/rbohr)*zar(n)

      xarx2=(xartem-x1)**2
      yary2=(yartem-y1)**2
      zarz2=(zartem-z1)**2
      xarx1=xartem-x1
      yary1=yartem-y1
      zarz1=zartem-z1
      r2arar=xarx2+yary2+zarz2
      r1arar=sqrt(r2arar)

      if (r1arar.gt.heut) then
      nrbfw=nrbfw+1
      goto 50
      endif

c     Evaluate the Barker-Fisher-Watts Potential.

      redr1=r1arar/radbfwB
      ararbfw=0.0
      expterm=exp(aarar*(1-redr1))

      a0term=a0arar
      a1term=a1arar*((redr1-1))
      a2term=a2arar*((redr1-1)**2)
      a3term=a3arar*((redr1-1)**3)
      a4term=a4arar*((redr1-1)**4)
      a5term=a5arar*((redr1-1)**5)

      asum=a0term+a1term+a2term+a3term+a4term+a5term

      term1=expterm*asum

      c6term=c6arar/(deltarar+(redr1**6))
```

```
      c8term=c8arar/(deltarar+(redr1**8))
      c10term=c10arar/(deltarar+(redr1**10))

      term2=c6term+c8term+c10term

      ararbfw=term1+term2

      sumbfw=ararbfw+sumbfw

      nabfw=nabfw+1

50    continue
40    continue
30    continue
20    continue
10    continue

      return
      end

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

      subroutine pot3bararar(sumararar,totararar1,totararar2,
     +totararar3,totararar4,totararar5)



      common /param3bararar/addd,addq,adqd,aqdd,adqq,aqdq,aqqd,
     +                aqqq,ad41,ad42,ad43
      common /arpos/ nar,naruc, xar(1000),yar(1000),zar(1000)
      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arpostem/xartem,yartem,zartem
      common /cut2b3b/ hcut
      common /highenergy/ en3ar,en3o,en3si
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
      common/bounds/xlow,xhigh,ylow,yhigh,zlow,zhigh
      common/check/natlj,nrtlj,nabfw,nrbfw,naelj,nrelj,na3ar,nr3ar,
     +na3o,nr3o,na3si,nr3si

      sumararar=0.0
      na3ar=0
      nr3ar=0
      totararar1=0.0
      totararar2=0.0
      totararar3=0.0
      totararar4=0.0
      totararar5=0.0

      do 10 il=1,nardum

      xar0=xar(il)
      yar0=yar(il)
      zar0=zar(il)

      totalddd=0.00
      totalddq=0.00
      totaldqd=0.00
      totalqdd=0.00
```

```
      totaldqq=0.00
      totalqdq=0.00
      totalqqd=0.00
      totalqqq=0.00
      totald4=0.000

      do 20 im=(il+1),(nar-1)

      do 40 i=-nreplica,nreplica
      do 50 j=-nreplica,nreplica
      do 60 k=-nreplica,nreplica

c     Convert argon coordinates to reduced units

      xar(im)=xar(im)*(rbohr/(3*auc))
      yar(im)=yar(im)*(rbohr/(3*buc))
      zar(im)=zar(im)*(rbohr/(3*cuc))

c     Store coordinates of replicas of original simulation box.

      xartem=(3*auc/rbohr)*(xar(im)+i)
      yartem=(3*buc/rbohr)*(yar(im)+j)
      zartem=(3*cuc/rbohr)*(zar(im)+k)

      xar(im)=(3*auc/rbohr)*xar(im)
      yar(im)=(3*buc/rbohr)*yar(im)
      zar(im)=(3*cuc/rbohr)*zar(im)

      xar1=xartem
      yar1=yartem
      zar1=zartem
      dif1x=xar1-xar0
      dif1y=yar1-yar0
      dif1z=zar1-zar0
      rar0ar1=sqrt(dif1x*dif1x+dif1y*dif1y+dif1z*dif1z)

      if(rar0ar1.gt.hcut) then
      nr3ar=nr3ar+1
      go to 60
      endif

      do 30 in=im+1,nar

      do 70 l=-nreplica,nreplica
      do 80 m=-nreplica,nreplica
      do 90 n=-nreplica,nreplica

c     Convert argon coordinates to reduced units

      xar(in)=xar(in)*(rbohr/(3*auc))
      yar(in)=yar(in)*(rbohr/(3*buc))
      zar(in)=zar(in)*(rbohr/(3*cuc))

c     Store coordinates of replicas of original simulation box.

      xartem=(3*auc/rbohr)*(xar(in)+l)
      yartem=(3*buc/rbohr)*(yar(in)+m)
      zartem=(3*cuc/rbohr)*(zar(in)+n)
```

```
      xar(in)=(3*auc/rbohr)*xar(in)
      yar(in)=(3*buc/rbohr)*yar(in)
      zar(in)=(3*cuc/rbohr)*zar(in)

      xar2=xartem
      yar2=yartem
      zar2=zartem
      dif2x=xar2-xar0
      dif2y=yar2-yar0
      dif2z=zar2-zar0
c
      dif3x=xar2-xar1
      dif3y=yar2-yar1
      dif3z=zar2-zar1
      rar0ar2=sqrt(dif2x*dif2x+dif2y*dif2y+dif2z*dif2z)
      rar1ar2=sqrt(dif3x*dif3x+dif3y*dif3y+dif3z*dif3z)
c
      if(rar0ar2.gt.hcut) then
      nr3ar=nr3ar+1
      go to 90
      endif
c
      cos1=(dif1x*dif2x+dif1y*dif2y+dif1z*dif2z)/(rar0ar1*rar0ar2)
      cos2=(-dif1x*dif3x-dif1y*dif3y-dif1z*dif3z)/(rar0ar1*rar1ar2)
      cos3=(dif2x*dif3x+dif2y*dif3y+dif2z*dif3z)/(rar0ar2*rar1ar2)
c
      cosangledif1=cos1*cos2+sqrt(abs((1-cos1**2)*(1-cos2**2)))
      cosangledif2=cos2*cos3+sqrt(abs((1-cos2**2)*(1-cos3**2)))
      cosangledif3=cos3*cos1+sqrt(abs((1-cos3**2)*(1-cos1**2)))
c
      cos3fi1=-3*cos1+4*cos1**3
      cos3fi2=-3*cos2+4*cos2**3
      cos3fi3=-3*cos3+4*cos3**3
c
      cos2fi1=2*cos1**2-1
      cos2fi2=2*cos2**2-1
      cos2fi3=2*cos3**2-1
c
      cos2angledif1=2*(cosangledif1)**2-1
      cos2angledif2=2*(cosangledif2)**2-1
      cos2angledif3=2*(cosangledif3)**2-1
c
      cos4fi1=8*cos1**3*sqrt(abs(1-cos1**2))-4*cos1*sqrt(abs(1-cos1**2))
      cos4fi2=8*cos2**3*sqrt(abs(1-cos2**2))-4*cos2*sqrt(abs(1-cos2**2))
      cos4fi3=8*cos3**3*sqrt(abs(1-cos3**2))-4*cos3*sqrt(abs(1-cos3**2))

c     in all following functions, the order ar0,ar1,ar2 is considered
c
c     function three body dipole-dipole-dipole
c
      fddd=3.0*addd*(1.0+3.0*cos1*cos2*cos3)*
     +(rar0ar1*rar0ar2*rar1ar2)**(-3.0)
c
c     function three body dipole-dipole-quadrupole
c
      fddq=addq*3.0/(16.0*(rar0ar1**3)*(rar0ar2*rar1ar2)**4)*
     +((9.0*cos3-25.0*cos3fi3)+6.0*cosangledif1*(3.0+5.0*cos2fi3))
c
```

```
c     function three body dipole-quadrupole-dipole
c
      fdqd=adqd*3.0/(16.0*(rar0ar2**3)*(rar1ar2*rar0ar1)**4)*
     +((9.0*cos2-25.0*cos3fi2)+6.0*cosangledif3*(3.0+5.0*cos2fi2))
c
c     function three body quadrupole-dipole-dipole
c
      fqdd=aqdd*3.0/(16.0*(rar1ar2**3)*(rar0ar2*rar0ar1)**4)*
     +((9.0*cos1-25.0*cos3fi1)+6.0*cosangledif2*(3.0+5.0*cos2fi1))
c
c     function three body dipole-dipole-dipole fourth order
c
      fd4=ad41*(1.0/(rar0ar1*rar1ar2)**6)*(1.0+cos2*cos2)+
     +ad42*1.0/((rar1ar2*rar0ar2)**6)*(1.0+cos3*cos3)+
     +ad43*1.0/((rar0ar2*rar0ar1)**6)*(1.0+cos1*cos1)
c
c     function three body dipole-quadrupole-quadrupole
c
      fdqq=adqq*15.0/(64.0*(rar1ar2**5)*(rar0ar1*rar0ar2)**4)*
     +(3.0*(cos1+5.0*cos3fi1)+20.0*cosangledif2*(1.0-3.0*cos2fi1)+
     +70.0*cos2angledif2*cos1)
c
c     function three body quadrupole-dipole-quadrupole
c
      fqdq=aqdq*15.0/(64.0*(rar0ar2**5)*(rar1ar2*rar0ar1)**4)*
     +(3.0*(cos2+5.0*cos3fi2)+20.0*cosangledif3*(1.0-3.0*cos2fi2)+
     +70.0*cos2angledif3*cos2)
c
c     function three body quadrupole-quadrupole-dipole
c
      fqqd=aqqd*15.0/(64.0*(rar0ar1**5)*(rar1ar2*rar0ar2)**4)*
     +(3.0*(cos3+5.0*cos3fi3)+20.0*cosangledif1*(1.0-3.0*cos2fi3)+
     +70.0*cos2angledif1*cos3)
c
c     function three body quadrupole-quadrupole-quadrupole
c
      fqqq=aqqq*15.0/(128.0*(rar0ar1*rar1ar2*rar0ar2)**5)*
     +(-27.0+220.0*cos1*cos2*cos3+490.0*cos2fi1*cos2fi2*cos2fi3+
     +175.0*(cos2angledif1+cos2angledif2+cos2angledif3))

c
c     make the sum at each step for each sub-term
c
c     variable test3b checks for high-energy 3b interactions
c
      test3b=fddd+fddq+fdqd+fqdd+fd4+fdqq+fqdq+fqqd+fqqq

c
      totalddd=totalddd+fddd
c
      totalddq=totalddq+fddq
      totaldqd=totaldqd+fdqd
      totalqdd=totalqdd+fqdd
c
      totald4=totald4+fd4
c
      totaldqq=totaldqq+fdqq
      totalqdq=totalqdq+fqdq
```

```
      totalqqd=totalqqd+fqqd
c
      totalqqq=totalqqq+fqqq

      na3ar=na3ar+1

90     continue
80     continue
70     continue
30     continue
60     continue
50     continue
40     continue
20     continue

      totararar1=totararar1+totalddd
      totararar2=totararar2+totalddq+totaldqd+totalqdd
      totararar3=totararar3+totalqdq+totalqqd+totaldqq
      totararar4=totararar4+totalqqq
      totararar5=totararar5+totald4
      sumararar= sumararar+totalddd+totalddq+totaldqd+
     +totalqdd+totalqdq+totalqqd+totaldqq+totalqqq+
     +totald4

10     continue


      return
      end

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

      subroutine pot3bararo(sumararo,totararo1,totararo2,
     +totararo3,totararo4,totararo5)



      common /param3bararo/oddd,oddq,odqd,oqdd,odqq,oqdq,oqqd,
     +                oqqq,od41,od42,od43

      common /arpos/ nar,naruc, xar(1000),yar(1000),zar(1000)
      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arpostem/xartem,yartem,zartem
      common /oxpos/ nox,xox(5184),yox(5184),zox(5184)
      common /oxpostem/xoxtem,yoxtem,zoxtem
      common /cut2b3b/ hcut
      common /highenergy/ en3ar,en3o,en3si
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag
      common/bounds/xlow,xhigh,ylow,yhigh,zlow,zhigh
      common/check/natlj,nrtlj,nabfw,nrbfw,naelj,nrelj,na3ar,nr3ar,
     +na3o,nr3o,na3si,nr3si

      na3o=0
      nr3o=0
      sumararo=0.0
      totararo1=0.0
      totararo2=0.0
      totararo3=0.0
```

```
      totararo4=0.0
      totararo5=0.0

      do 10 il=1,nardum

      xar0=xar(il)
      yar0=yar(il)
      zar0=zar(il)

      totalddd=0.00
      totalddq=0.00
      totaldqd=0.00
      totalqdd=0.00
      totaldqq=0.00
      totalqdq=0.00
      totalqqd=0.00
      totalqqq=0.00
      totald4=0.000

      do 20 im=il+1,nar

      do 40 i=-nreplica,nreplica
      do 50 j=-nreplica,nreplica
      do 60 k=-nreplica,nreplica

c     Convert argon coordinates to reduced units

      xar(im)=xar(im)*(rbohr/(3*auc))
      yar(im)=yar(im)*(rbohr/(3*buc))
      zar(im)=zar(im)*(rbohr/(3*cuc))

c     Store coordinates of replicas of original simulation box.

      xartem=(3*auc/rbohr)*(xar(im)+i)
      yartem=(3*buc/rbohr)*(yar(im)+j)
      zartem=(3*cuc/rbohr)*(zar(im)+k)

      xar(im)=(3*auc/rbohr)*xar(im)
      yar(im)=(3*buc/rbohr)*yar(im)
      zar(im)=(3*cuc/rbohr)*zar(im)
c
      xar1=xartem
      yar1=yartem
      zar1=zartem
      dif1x=xar1-xar0
      dif1y=yar1-yar0
      dif1z=zar1-zar0
      rar0ar1=sqrt(dif1x*dif1x+dif1y*dif1y+dif1z*dif1z)
c
      if(rar0ar1.gt.hcut) then
      nr3o=nr3o+1
      go to 60
      endif
c
      do 30 in=1,nox

      do 70 l=-nreplica,nreplica
      do 80 m=-nreplica,nreplica
```

```
      do 90 n=-nreplica,nreplica

c     Convert argon coordinates to reduced units

      xox(in)=xox(in)*(rbohr/(3*auc))
      yox(in)=yox(in)*(rbohr/(3*buc))
      zox(in)=zox(in)*(rbohr/(3*cuc))

c     Store coordinates of replicas of original simulation box.

      xoxtem=(3*auc/rbohr)*(xox(in)+l)
      yoxtem=(3*buc/rbohr)*(yox(in)+m)
      zoxtem=(3*cuc/rbohr)*(zox(in)+n)

      xox(in)=(3*auc/rbohr)*xox(in)
      yox(in)=(3*buc/rbohr)*yox(in)
      zox(in)=(3*cuc/rbohr)*zox(in)
c
      xo2=xoxtem
      yo2=yoxtem
      zo2=zoxtem
      dif2x=xo2-xar0
      dif2y=yo2-yar0
      dif2z=zo2-zar0
c
      dif3x=xo2-xar1
      dif3y=yo2-yar1
      dif3z=zo2-zar1
      rar0o2=sqrt(dif2x*dif2x+dif2y*dif2y+dif2z*dif2z)
      rar1o2=sqrt(dif3x*dif3x+dif3y*dif3y+dif3z*dif3z)
c
      if(rar0o2.gt.hcut) then
      nr3o=nr3o+1
      go to 90
      endif
c
      cos1=(dif1x*dif2x+dif1y*dif2y+dif1z*dif2z)/(rar0ar1*rar0o2)
      cos2=(-dif1x*dif3x-dif1y*dif3y-dif1z*dif3z)/(rar0ar1*rar1o2)
      cos3=(dif2x*dif3x+dif2y*dif3y+dif2z*dif3z)/(rar0o2*rar1o2)
c
      cosangledif1=cos1*cos2+sqrt(abs((1-cos1**2)*(1-cos2**2)))
      cosangledif2=cos2*cos3+sqrt(abs((1-cos2**2)*(1-cos3**2)))
      cosangledif3=cos3*cos1+sqrt(abs((1-cos3**2)*(1-cos1**2)))
c
      cos3fi1=-3*cos1+4*cos1**3
      cos3fi2=-3*cos2+4*cos2**3
      cos3fi3=-3*cos3+4*cos3**3
c
      cos2fi1=2*cos1**2-1
      cos2fi2=2*cos2**2-1
      cos2fi3=2*cos3**2-1
c
      cos2angledif1=2*(cosangledif1)**2-1
      cos2angledif2=2*(cosangledif2)**2-1
      cos2angledif3=2*(cosangledif3)**2-1
c
      cos4fi1=8*cos1**3*sqrt(abs(1-cos1**2))-4*cos1*sqrt(abs(1-cos1**2))
      cos4fi2=8*cos2**3*sqrt(abs(1-cos2**2))-4*cos2*sqrt(abs(1-cos2**2))
```

```
      cos4fi3=8*cos3**3*sqrt(abs(1-cos3**2))-4*cos3*sqrt(abs(1-cos3**2))

c     in all following functions, the order ar0,ar1,o2 is considered
c
c     function three body dipole-dipole-dipole
c
      fddd=3.0*oddd*(1.0+3.0*cos1*cos2*cos3)*
     +(rar0ar1*rar0o2*rar1o2)**(-3.0)
c
c     function three body dipole-dipole-quadrupole
c
      fddq=oddq*3.0/(16.0*(rar0ar1**3)*(rar0o2*rar1o2)**4)*
     +((9.0*cos3-25.0*cos3fi3)+6.0*cosangledif1*(3.0+5.0*cos2fi3))
c
c     function three body dipole-quadrupole-dipole
c
      fdqd=odqd*3.0/(16.0*(rar0o2**3)*(rar1o2*rar0ar1)**4)*
     +((9.0*cos2-25.0*cos3fi2)+6.0*cosangledif3*(3.0+5.0*cos2fi2))
c
c     function three body quadrupole-dipole-dipole
c
      fqdd=oqdd*3.0/(16.0*(rar1o2**3)*(rar0o2*rar0ar1)**4)*
     +((9.0*cos1-25.0*cos3fi1)+6.0*cosangledif2*(3.0+5.0*cos2fi1))
c
c     function three body dipole-dipole-dipole fourth order
c
      fd4=od41*(1.0/(rar0ar1*rar1o2)**6)*(1.0+cos2*cos2)+
     +od42*1.0/((rar1o2*rar0o2)**6)*(1.0+cos3*cos3)+
     +od43*1.0/((rar0o2*rar0ar1)**6)*(1.0+cos1*cos1)
c
c     function three body dipole-quadrupole-quadrupole
c
      fdqq=odqq*15.0/(64.0*(rar1o2**5)*(rar0ar1*rar0o2)**4)*
     +(3.0*(cos1+5.0*cos3fi1)+20.0*cosangledif2*(1.0-3.0*cos2fi1)+
     +70.0*cos2angledif2*cos1)
c
c     function three body quadrupole-dipole-quadrupole
c
      fqdq=oqdq*15.0/(64.0*(rar0o2**5)*(rar1o2*rar0ar1)**4)*
     +(3.0*(cos2+5.0*cos3fi2)+20.0*cosangledif3*(1.0-3.0*cos2fi2)+
     +70.0*cos2angledif3*cos2)
c
c     function three body quadrupole-quadrupole-dipole
c
      fqqd=oqqd*15.0/(64.0*(rar0ar1**5)*(rar1o2*rar0o2)**4)*
     +(3.0*(cos3+5.0*cos3fi3)+20.0*cosangledif1*(1.0-3.0*cos2fi3)+
     +70.0*cos2angledif1*cos3)
c
c     function three body quadrupole-quadrupole-quadrupole
c
      fqqq=oqqq*15.0/(128.0*(rar0ar1*rar1o2*rar0o2)**5)*
     +(-27.0+220.0*cos1*cos2*cos3+490.0*cos2fi1*cos2fi2*cos2fi3+
     +175.0*(cos2angledif1+cos2angledif2+cos2angledif3))

c     variable test3b checks for high-energy 3b interactions

      test3b=fddd+fddq+fdqd+fqdd+fd4+fdqq+fqdq+fqqd+fqqq
```

```
c     make the sum at each step for each sub-term
c
      totalddd=totalddd+fddd
c
      totalddq=totalddq+fddq
      totaldqd=totaldqd+fdqd
      totalqdd=totalqdd+fqdd
c
      totald4=totald4+fd4
c
      totaldqq=totaldqq+fdqq
      totalqdq=totalqdq+fqdq
      totalqqd=totalqqd+fqqd
c
      totalqqq=totalqqq+fqqq
c
      na3o=na3o+1

90    continue
80    continue
70    continue
30    continue
60    continue
50    continue
40    continue
20    continue

      totararo1=totararo1+totalddd
      totararo2=totararo2+totalddq+totaldqd+totalqdd
      totararo3=totararo3+totalqdq+totalqqd+totaldqq
      totararo4=totararo4+totalqqq
      totararo5=totararo5+totald4
      sumararo= sumararo+totalddd+totalddq+totaldqd+
     +totalqdd+totalqdq+totalqqd+totaldqq+totalqqq+
     +totald4

10    continue

      return
      end

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

      subroutine pot3bararsi(sumararsi,totararsi1,totararsi2,
     +totararsi3,totararsi4,totararsi5)



      common /param3bararsi/ sddd,sddq,sdqd,sqdd,sdqq,sqdq,sqqd,
     +                 sqqq,sd41,sd42,sd43
      common /cut2b3b/ heut
      common /highenergy/ en3ar,en3o,en3si
      common/bounds/xlow,xhigh,ylow,yhigh,zlow,zhigh
      common /sipos/ nsi,xsi(2592),ysi(2592),zsi(2592)
      common /sipostem/ xsitem,ysitem,zsitem
      common /arpos/ nar,naruc, xar(1000),yar(1000),zar(1000)
      common/ardum/ nardum,xardum(50),yardum(50),zardum(50)
      common /arpostem/xartem,yartem,zartem
```

```
      common/check/natlj,nrtlj,nabfw,nrbfw,naelj,nrelj,na3ar,nr3ar,
     +na3o,nr3o,na3si,nr3si
      common /values/ auc,buc,cuc,rbohr,nreplica,calcflag

      na3si=0
      nr3si=0
      sumararsi=0.0
      totararsi1=0.0
      totararsi2=0.0
      totararsi3=0.0
      totararsi4=0.0
      totararsi5=0.0

      do 10 il=1,nardum

      xar0=xar(il)
      yar0=yar(il)
      zar0=zar(il)

      totalddd=0.00
      totalddq=0.00
      totaldqd=0.00
      totalqdd=0.00
      totaldqq=0.00
      totalqdq=0.00
      totalqqd=0.00
      totalqqq=0.00
      totald4=0.000

      do 20 im=(il+1),nar

      do 40 i=-nreplica,nreplica
      do 50 j=-nreplica,nreplica
      do 60 k=-nreplica,nreplica

c     Convert argon coordinates to reduced units

      xar(im)=xar(im)*(rbohr/(3*auc))
      yar(im)=yar(im)*(rbohr/(3*buc))
      zar(im)=zar(im)*(rbohr/(3*cuc))

c     Store coordinates of replicas of original simulation box.

      xartem=(3*auc/rbohr)*(xar(im)+i)
      yartem=(3*buc/rbohr)*(yar(im)+j)
      zartem=(3*cuc/rbohr)*(zar(im)+k)

      xar(im)=(3*auc/rbohr)*xar(im)
      yar(im)=(3*buc/rbohr)*yar(im)
      zar(im)=(3*cuc/rbohr)*zar(im)

      xar1=xartem
      yar1=yartem
      zar1=zartem
      dif1x=xar1-xar0
      dif1y=yar1-yar0
      dif1z=zar1-zar0
      rar0ar1=sqrt(dif1x*dif1x+dif1y*dif1y+dif1z*dif1z)
```

```
c
      if(rar0ar1.gt.hcut) then
      nr3si=nr3si+1
      go to 60
      endif
c
      do 30 in=1,nsi

      do 70 l=-nreplica,nreplica
      do 80 m=-nreplica,nreplica
      do 90 n=-nreplica,nreplica

c     Convert silicon coordinates to reduced units

      xsi(in)=xsi(in)*(rbohr/(3*auc))
      ysi(in)=ysi(in)*(rbohr/(3*buc))
      zsi(in)=zsi(in)*(rbohr/(3*cuc))

c     Store coordinates of replicas of original simulation box.

      xsitem=(3*auc/rbohr)*(xsi(in)+l)
      ysitem=(3*buc/rbohr)*(ysi(in)+m)
      zsitem=(3*cuc/rbohr)*(zsi(in)+n)

      xsi(in)=(3*auc/rbohr)*xsi(in)
      ysi(in)=(3*buc/rbohr)*ysi(in)
      zsi(in)=(3*cuc/rbohr)*zsi(in)

      xsi2=xsitem
      ysi2=ysitem
      zsi2=zsitem
      dif2x=xsi2-xar0
      dif2y=ysi2-yar0
      dif2z=zsi2-zar0
c
      dif3x=xsi2-xar1
      dif3y=ysi2-yar1
      dif3z=zsi2-zar1
      rar0si2=sqrt(dif2x*dif2x+dif2y*dif2y+dif2z*dif2z)
      rar1si2=sqrt(dif3x*dif3x+dif3y*dif3y+dif3z*dif3z)
c
      if(rar0si2.gt.hcut) then
      nr3si=nr3si+1
      go to 90
      endif
c
      cos1=(dif1x*dif2x+dif1y*dif2y+dif1z*dif2z)/(rar0ar1*rar0si2)
      cos2=(-dif1x*dif3x-dif1y*dif3y-dif1z*dif3z)/(rar0ar1*rar1si2)
      cos3=(dif2x*dif3x+dif2y*dif3y+dif2z*dif3z)/(rar0si2*rar1si2)
c
      cosangledif1=cos1*cos2+sqrt(abs((1-cos1**2)*(1-cos2**2)))
      cosangledif2=cos2*cos3+sqrt(abs((1-cos2**2)*(1-cos3**2)))
      cosangledif3=cos3*cos1+sqrt(abs((1-cos3**2)*(1-cos1**2)))
c
      cos3fi1=-3*cos1+4*cos1**3
      cos3fi2=-3*cos2+4*cos2**3
      cos3fi3=-3*cos3+4*cos3**3
c
```

```
      cos2fi1=2*cos1**2-1
      cos2fi2=2*cos2**2-1
      cos2fi3=2*cos3**2-1
c
      cos2angledif1=2*(cosangledif1)**2-1
      cos2angledif2=2*(cosangledif2)**2-1
      cos2angledif3=2*(cosangledif3)**2-1
c
      cos4fi1=8*cos1**3*sqrt(abs(1-cos1**2))-4*cos1*sqrt(abs(1-cos1**2))
      cos4fi2=8*cos2**3*sqrt(abs(1-cos2**2))-4*cos2*sqrt(abs(1-cos2**2))
      cos4fi3=8*cos3**3*sqrt(abs(1-cos3**2))-4*cos3*sqrt(abs(1-cos3**2))

c     in all following functions, the order ar0,ar1,o2 is considered
c
c     function three body dipole-dipole-dipole
c
      fddd=3.0*sddd*(1.0+3.0*cos1*cos2*cos3)*
     +(rar0ar1*rar0si2*rar1si2)**(-3.0)
c
c     function three body dipole-dipole-quadrupole
c
      fddq=sddq*3.0/(16.0*(rar0ar1**3)*(rar0si2*rar1si2)**4)*
     +((9.0*cos3-25.0*cos3fi3)+6.0*cosangledif1*(3.0+5.0*cos2fi3))
c
c     function three body dipole-quadrupole-dipole
c
      fdqd=sdqd*3.0/(16.0*(rar0si2**3)*(rar1si2*rar0ar1)**4)*
     +((9.0*cos2-25.0*cos3fi2)+6.0*cosangledif3*(3.0+5.0*cos2fi2))
c
c     function three body quadrupole-dipole-dipole
c
      fqdd=sqdd*3.0/(16.0*(rar1si2**3)*(rar0si2*rar0ar1)**4)*
     +((9.0*cos1-25.0*cos3fi1)+6.0*cosangledif2*(3.0+5.0*cos2fi1))
c
c     function three body dipole-dipole-dipole fourth order
c
      fd4=sd41*(1.0/(rar0ar1*rar1si2)**6)*(1.0+cos2*cos2)+
     +sd42*1.0/((rar1si2*rar0si2)**6)*(1.0+cos3*cos3)+
     +sd43*1.0/((rar0si2*rar0ar1)**6)*(1.0+cos1*cos1)
c
c     function three body dipole-quadrupole-quadrupole
c
      fdqq=sdqq*15.0/(64.0*(rar1si2**5)*(rar0ar1*rar0si2)**4)*
     +(3.0*(cos1+5.0*cos3fi1)+20.0*cosangledif2*(1.0-3.0*cos2fi1)+
     +70.0*cos2angledif2*cos1)
c
c     function three body quadrupole-dipole-quadrupole
c
      fqdq=sqdq*15.0/(64.0*(rar0si2**5)*(rar1si2*rar0ar1)**4)*
     +(3.0*(cos2+5.0*cos3fi2)+20.0*cosangledif3*(1.0-3.0*cos2fi2)+
     +70.0*cos2angledif3*cos2)
c
c     function three body quadrupole-quadrupole-dipole
c
      fqqd=sqqd*15.0/(64.0*(rar0ar1**5)*(rar1si2*rar0si2)**4)*
     +(3.0*(cos3+5.0*cos3fi3)+20.0*cosangledif1*(1.0-3.0*cos2fi3)+
     +70.0*cos2angledif1*cos3)
c
```

```
c     function three body quadrupole-quadrupole-quadrupole
c
      fqqq=sqqq*15.0/(128.0*(rar0ar1*rar1si2*rar0si2)**5)*
     +(-27.0+220.0*cos1*cos2*cos3+490.0*cos2fi1*cos2fi2*cos2fi3+
     +175.0*(cos2angledif1+cos2angledif2+cos2angledif3))

c
c     variable test3b checks for high-energy 3b interactions

      test3b=fddd+fddq+fdqd+fqdd+fd4+fdqq+fqdq+fqqd+fqqq

c     make the sum at each step for each sub-term
c
      totalddd=totalddd+fddd
c
      totalddq=totalddq+fddq
      totaldqd=totaldqd+fdqd
      totalqdd=totalqdd+fqdd
c
      totald4=totald4+fd4
c
      totaldqq=totaldqq+fdqq
      totalqdq=totalqdq+fqdq
      totalqqd=totalqqd+fqqd
c
      totalqqq=totalqqq+fqqq
      na3si=na3si+1

90    continue
80    continue
70    continue
30    continue
60    continue
50    continue
40    continue
20    continue

      totararsi1=totararsi1+totalddd
      totararsi2=totararsi2+totalddq+totaldqd+totalqdd
      totararsi3=totararsi3+totalqdq+totalqqd+totaldqq
      totararsi4=totararsi4+totalqqq
      totararsi5=totararsi5+totald4
      sumararsi= sumararsi+totalddd+totalddq+totaldqd+
     +totalqdd+totalqdq+totalqqd+totaldqq+totalqqq+
     +totald4

10    continue

      return
      end

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

      subroutine convertok(something)


      common /bfwparams/ a0arar,a1arar,a2arar,a3arar,a4arar,a5arar,
```

```
     +              aarar,deltarar,epsilbfw,radbfw

c
c     converts from atomic units to kelvin
c
      something=something*epsilbfw

      return
      end
```