NPL
National Physical Laboratory

REPORT

NPL Report MAT 44

Project A27: Molecular modelling Final Report

John Blackburn, Paul Weaver, Hugh Davies, Michael Agar, Anna Kimmel

April 2010

National Physical Laboratory, Hampton Road, Teddington, Middlesex, United Kingdom, TW11 0LW
Switchboard 020 8977 3222 NPL Helpline 020 8943 6880 Fax 020 8614 0446 E-mail enquiry@npl.co.uk www.npl.co.uk

NPL Report MAT 44

Project A27: Molecular modelling Final Report

John Blackburn, Paul Weaver, Hugh Davies, Michael Agar, Anna Kimmel
Materials Group

April 2010

ABSTRACT

This report describes molecular modelling simulations undertaken as part of the A27 project. Molecular dynamics and Monte Carlo simulations were carried out on a wide variety of material and temperature regimes, from argon at 100 K to the copper/lead binary alloy at 1200K. Also considered are water and CO_2 simulations and simulations of the piezoelectric barium titanate. We have not contented ourselves with "snapshots" of simulated atoms in this report. Rather, we believe that molecular simulations must "earn their keep" by providing agreement to experimental findings in accordance with NPL's focus on metrology. Consequently, we have compared all simulation results to experimental data. We have also shown comparison to approximate analytical equations which, in some cases, can rival the accuracy of the molecular codes. In particular, we have focused on reproducing *phase diagrams* using the molecular models as these are amongst the most striking and characteristic real-world material properties. Phase diagrams considered in this report include temperature-density phase diagrams (ie solid/liquid/gas phase transitions), temperature-molar fraction diagrams (for binary alloys etc) and symmetry phase transitions in piezoelectics (with consequent hysteretic polarisation-electric field plots predicted). We have investigated the ways in which molecular modelling results can help improve predictions made by MTDATA, NPL's flagship commercial materials modelling program. We have shown that the realistic predictions made by the molecular models can be input to MTDATA, producing better quality phase diagrams than the ideal gas approximations MTDATA would otherwise use. We have written "driver" programs for the modelling codes allowing people who are not experts in molecular simulation to generate phase diagrams through repeated calls to these simulation codes. We have also introduced parallelisation schemes to make use of the NPL Grid. This project acts as a showcase for MDL, a NPL-written molecular dynamics code which is an important piece of IP we could exploit in future. To assess the accuracy of MDL, we have compared it to three other solvers. We have also upgraded MDL, during the course of this project, to support calculation of chemical potentials. These are essential to assess mixing behaviour in gas mixtures (e.g. binary alloys), which are of direct relevance to MTDATA. The full set of codes is listed in Table 1.2, which gives links for downloading these programs.

NPL Report MAT 44

National Physical Laboratory,
Hampton Road, Teddington, Middlesex, United Kingdom TW11 0LW

We gratefully acknowledge the financial support of the UK Department for Innovation, Universities and Skills (National Measurement System Directorate)

Approved on behalf of the Managing Director, NPL by Markys Cain,
Knowledge leader for the Materials Team

Contents

Chapter 1

Introduction

This project has the simple title of "molecular modelling". This term essentially refers to performing atomistic calculations in order to deduce various material properties and behaviour. Forces are calculated between atoms (which have a realistic charge and mass) and atom positions are updated over many time steps. Average atomic positions and velocities can be used to deduce the state of the system in terms of thermodynamic quantities such as temperature, pressure and so on. So far, the description is vague and could relate to a near infinity of possible calculations. The aim of this chapter is to outline more specifically what we actually did and give a rationale for why we focussed on the specific topics presented in this report.

1.1 Molecular dynamics and molecular mechanics

In this work, we will focus primarily on *molecular dynamics* which involves moving the atoms realistically according to Newton's Laws of Motion. MD operates at a non-zero temperature, indicating that statistics is involved, although explicit thermodynamic relations generally only appear when we calculate quantities like temperature. Another important concept is that atoms are confined in a *simulation box*, which represents the systems interaction with an external reservoir with which it exchanged energy and volume in such a way as to achieve thermodynamic equilibrium.[1] It is important to distinguish MD from *Molecular Mechanics* (MM). There is a fundamental difference in the philosophies of MM and MD which we must discuss straight away (since defining what MD is *not* will help to define what it actually is). In MM, there is no concept of *statistics*, no temperature and no enclosing box: the atoms just move in a deterministic manner, according, usually, to Newton's Laws. Friction may also be added so that atoms tend to slow down and eventually reach their minimum potential energy positions. In MM studies, we often see a complex molecule wandering across space and perhaps interacting with an other molecule (this is often done in biological studies, calculating the properties of drugs and so on). Researchers then argue that this is precisely what would happen in reality: the trajectory shown by the computer, and the final resting place of the molecule, is taken very seriously. There is no concept of temperature in these deterministic MM simulations (nor pressure: as there is no simulation box). In MD, on the other hand, we have a very different picture: not one but many hundreds of molecules moving around in a box. The trajectory of each molecule is not taken seriously at all. For one thing, when atoms and molecules start interacting in complex ways, it is not possible to solve Newton's laws beyond a very few timesteps (else the authors would have won the lottery by now!). But the more important point is, we are not *interested* in the atom trajectories per se, but rather the statistics the atoms "cook up" between them as a result of mixing. Put more formally, we want the system to explore all regions of phase space with equal probability (a "point" in multi-dimensional phase space being the set of all atomic coordinates at an instant in time). We then

[1] It should be noted that this box is a fundamentally *thermodynamic* concept. It represents volume, one of the quantities with respect to which entropy is maximised, the others being energy and number of particles of each type

take statistical averages to establish bulk quantities for the system which depend on temperature. In this sense, we use Newton's laws of motion only as a convenient algorithm to make the system "walk" from one point in phase space to another, in such a way as to sample each phase space point equally. In this sense it does not matter whether the atoms are really moving in the way they would "actually move" in experimental reality. The molecular dynamicist, in contrast to the molecular mechanic, has a healthy scepticism about the trajectories of individual atoms reported by the computer. In a sense, MM has more in common with astronomy (plotting the precise motion of planets etc) than it does with MD.

In a similar vein, it is often stated that MD calculations are limited because we cannot possibly predict the motion of each atom in a glass of water (for example). This is true, of course, and computer power limitations do impose restrictions on the accuracy of MD calculations. But in another sense, this is missing the point. Even if we *could* predict, with total accuracy the x, y, z motion of 10^{25} atoms, this information would tell us absolutely nothing about experimental reality. That is because all experimental results are *statistical* results: they all depend on temperature, which is a kind of dustbin for those 10^{25} redundant degrees of freedom. When you dip a thermometer in a glass of water, the thermometer is doing a statistical average over the atomic positions and velocities. Atomic trajectories are useless by themselves, only by performing statistical calculations can we obtain quantities which can be compared to experiment. These statistical calculations must be derived from statistical thermodynamics showing that MD is not a brute force substitute for knowing this theory. On the contrary, MD results are made useful only by application of such theory. Of course, a molecular dynamicist would be delighted to have a computer powerful enough to solve the motion of 10^{25} atoms, but he would not use it in this way. Having won the lottery, the molecular dynamicist would then use the ultracomputer to simulate 100-1000 atoms (in general 100-1000 *molecules*) but over millions of *time steps*.

1.2 Molecular dynamics and the Monte Carlo technique

This idea of exploring phase space links nicely to an alternative technique which fits in the scope of molecular modelling: the *Monte Carlo* technique (MC). In MC, we do not evolve the atoms according to Newton's Laws but rather "walk" from one point in phase space to another according to a set of probabilities. The probabilities are modelled on the Boltzmann distribution $\exp -U/kT$ which ensure that phase space points are sampled appropriately (for the *canonical ensemble*, see below for more discussion). There is no need to store velocities in MC since atoms do not really move but are simply placed in new positions according to the probabilistic algorithm. MC is really taking the statistical nature of MD, discussed above, to its ultimate conclusion.

The choice of whether to use MD or MC is essentially a subjective one, dependent on what previous experience a researcher has and what tools are available. There is no clear evidence that one technique is superior to another in all circumstances. MC is generally favoured over MD for the calculation of phase transformations, while MD is the only technique available for non-equilibrium thermodynamics. In this report we are interested primarily in phase transformations at equilibrium. Despite this we have chosen to use molecular dynamics for the most part: the reason is familiarity and convenience. At NPL, we have an in-house MD code called MDL which was convenient to use. MDL is a very powerful program and a valuable tool for NPL's future theoretical work. It was written by Dr Vladimir Sokhan while employed at NPL so there are no restrictions in using or modifying this program. (modifying the code was essential to program in chemical potentials for example). MDL is also a significant part of NPL's intellectual property (IP) and one outcome of this project is to assess the accuracy of MDL, which we may be able to market in future. A major deliverable of this project was to prepare our own software tools, and become expert at using such tools: MDL is therefore a logical choice. We also worked with an open source MC code called TOWHEE, [1], and a comparison between Towhee and MDL's results is given in Chapter 4. One concern we had with Towhee was the number of numerical parameters that need to be entered in the input files (this includes probabilities of atoms being interchanged in various ways). Without considerable experience, it's hard to guess suitable number for such non-physical parameters. Of course, numerical factors (convergence tolerances and so on) must be input in all types of simulations, but in (basic) MD

there is only one such parameter: the system timestep which is has an obvious physical meaning. In short, we tend to favour MD because of its comparative closeness to physical reality. Finally, it should be noted that solid state atomistic calculations, including piezoelectrics, tend to be performed using MD rather than MC codes.

It should be borne in mind, however, that certain simulations can *only* be done in MC. These include systems in which atom positions, dipole orientations etc, are considered discretised. The archetype for such problems is the Ising Model, [2], in which dipole values are assigned values of ± 1. Performing Newtonian MD would make no sense in such a scheme.

The distinction between MD and MC becomes more blurred when we consider advanced MD and MC features. The simplest form of MD operates in the *NVE ensemble*. In this ensemble, the Number of particles, the Volume and the Energy are held fixed (hence: NVE). A set of atoms, evolved according to Newton's laws of motion will naturally conserve total energy (kinetic + potential energy). However, we often want to consider other ensembles such as the constant number of particles, volume and temperature (NVT) and number of particles, pressure and temperature (NPT) ensembles. To do this requires iterating atoms, not according to Newton's Laws but according to other dynamic schemes, Table 2.1. These generally involve several extra variables (degrees of freedom) which represent a thermostat or barostat to which the system is connected. In this report, we generally use the NVT ensemble.

On the other hand, MC codes lacking any "default" dynamics, have explicit functions built in to them to perform statistical sampling. The common Boltzman factor $\exp(-U/kT)$ is used for the canonical (NVT) ensemble. Other factors could be equally applied to give the grand canonical[2] and the isothermal-isobaric (NPT) ensembles, [3]. In each case, an extra degree of freedom is added in (e.g. volume or number of particles). Extra complexity may also be added to MC codes to allow them to solve non-equilibrium problems. The reader is directed to [3] for further discussion.

All these developments blur the line between MD and MC. In fact, much code can be reused between the two types of programs, including the calculation of force which is identical in MD and MC codes. Calculation of forces between atoms is an N^2 process and dominates the time it takes to run either type of code (it also amounts to a big chunk of the number of lines of code). The choice between MD and MC partly comes down to whether one considers it worthwhile to maintain, store and update *velocities* as well as atomic positions.

Our main reference in this project is "Computer simulation of liquids" by Allen and Tildesley. This excellent book gives detailed instructions on how to program a *liquid* MD or MC code. There is no discussion on solid state modelling: as we will show in Chapter 2 liquid and solid MD modelling are quite different in nature. The Tildesley book contains chapters about calculating forces (which are common to MD and MC) and this includes constraint and rigid body dynamics. There are separate chapters for the MD and MC update schemes, with discussions on how to implement the different ensembles. Finally there is a chapter on statistical physics, showing algorithms to convert atomic trajectory data into useful thermodynamic quantities such as temperature and pressure. In this report, we will go considerably beyond this discussion to show how *phase transitions* may be calculated from MD results, Chapter 3.

Another useful guide is the DL_POLY manual [4] which provides a detailed account of the algorithms used in the code (these algorithms, or close variants, are used in all MD codes). In particular the various ensembles are described in a step by step way. It is very useful to have these ensembles in one place rather than scattered over two dozen research papers.

1.3 The rationale of this project

Leaving aside the differences between MD and MC, there are several different possible aims for a molecular modelling project. A great number of molecular modelling studies in literature aim only to calculate the atomic positions as function of time (or perhaps the lowest energy state is presented). Atomic positions are then shown as a sequence of snapshots from which conclusions can be drawn about the behaviour of the atoms in a real system. An example of this was mentioned above: calculating two molecules moving

[2]Where the chemical potential, volume and temperature are fixed: μVT

towards each other to see if and how they will interact. Another example is the first author's PhD thesis, [5], in which atomic motion in a complex mineral was calculated and plots of domain formation as a function of time were shown. These "cartoons" were the end product of the thesis and no additional "measurable" calculation was presented. Having established the atomic evolution in some starting or default configuration, the authors will typically vary the conditions so as to make the atoms move in a different way and hence comment on how materials could be engineered to achieve some goal. The trouble with "atom snapshot" papers like these is that they convey only qualitative information. It is not possible to take a corresponding photograph of atoms at the Angstrom scale (MD simulation are typically done with 10^2-10^6 atoms in a box of order 100 Åon a side). This does not stop some researchers putting their MD/MC results next to experimental photographs, often with very different length scales!

Discussions about how the atoms evolve in these simulation are subjective by nature. Recommendations for improvement based on such qualitative judgements may or may not be useful in the real world. In this sense, these "atom snapshot" simulations are somewhat unsatisfactory. This is a shame as, often, enormous effort has been used to model the complex atomic motions involved (in the case of the author's thesis, 10^5 atoms were painstakingly connected by 10^6 springs, but the result was just a series of "cartoons" of atomic time-evolution which could only be discussed by hand waving arguments).

In this study, we have instead performed *quantitative* simulations in which bulk quantities are calculated which can be compared with experiment. These include: pressure, volume, temperature, chemical potential, in liquid simulations; and polarisation, electric field, stress and strain in solid structures. In all cases, simulation results were compared to experimental data. In some cases, we have fitted MD to experiment by adjusting the parameters dictating inter-atomic forces. This "inverse modelling" approach is an important technique for fitting these force parameters, which otherwise have unknown values.[3]

The requirement for quantitative data is inspired by a key goal of this project: *to link MD code with* MTDATA. MTDATA is NPL's flagship thermodynamics code and is available commercially. MTDATA works by maintaining a large database of free energy functions for various materials. When several materials are combined, MTDATA can predict consequent material properties such as pressure-temperature variations, concentrations and mass fractions of different species. One of the primary goals of MTDATA is to calculate *phase transitions* in materials, so, in this project, we have largely focussed on modelling phase transitions in our MD/MC code. Phase transitions certainly fit into the area of "experimentally-verifiable phenomena", mentioned above. Indeed these are among the most common of experiments and familiar even to non-scientists.[4] Phase transitions occur when the system suddenly "jumps" from one free energy minimum to another. They are among the most striking and easily measurable phenomena in nature and serve as fundamental characteristics of materials. While it may be tempting to start simulating very exotic systems immediately, our belief is that there is no point expecting MD to predict complex systems if it cannot predict that water boils at 100 C (at 1 atmosphere). In fact, this particular prediction turns out to be very difficult for an MD code. This temperature and pressure range is far from the critical point so the phase transition is very dramatic causing various numerical problems discussed in Chapter 5. Water has always been a challenging molecule to model (www.lsbu.ac.uk/water) and phase transitions, in general, are among the most difficult phenomena to model accurately. One of the goals of this project is to describe which phenomena can be modelled easily and accurately in MD and which are more intractable.

MTDATA can essentially predict two types of phase transition shown in Figure 1.1, both of which we have repeated with MD simulations. The first type of phase transition, Figure 1.1(a), involves a single material whose volume varies (fixed number of atoms). As volume decreases, pressure increases but there is a range of volume for which the pressure remains constant as volume continues to fall. To put it another way: if we slowly increase the pressure, the volume will suddenly collapse signifying a phase transition from gas to much denser liquid. Of course, another phase transition will occur at still lower volumes and higher pressures: liquid to solid. This is much more subtle and difficult to see in MD. Another consideration is that "fluid" molecular dynamics codes do not work very well for modelling solids as will be shown in Chapter 2. There is no molecular code which can handle both solids and liquids well, so

[3] In principle these values could be calculated quantum mechanically, but parametrising the energy surface in terms of simple "pair potentials" is difficult, even in cases where the energy surface can be accurately calculated

[4] The liquid-gas phase transition in water is observed by millions of people every day when they boil a kettle of water

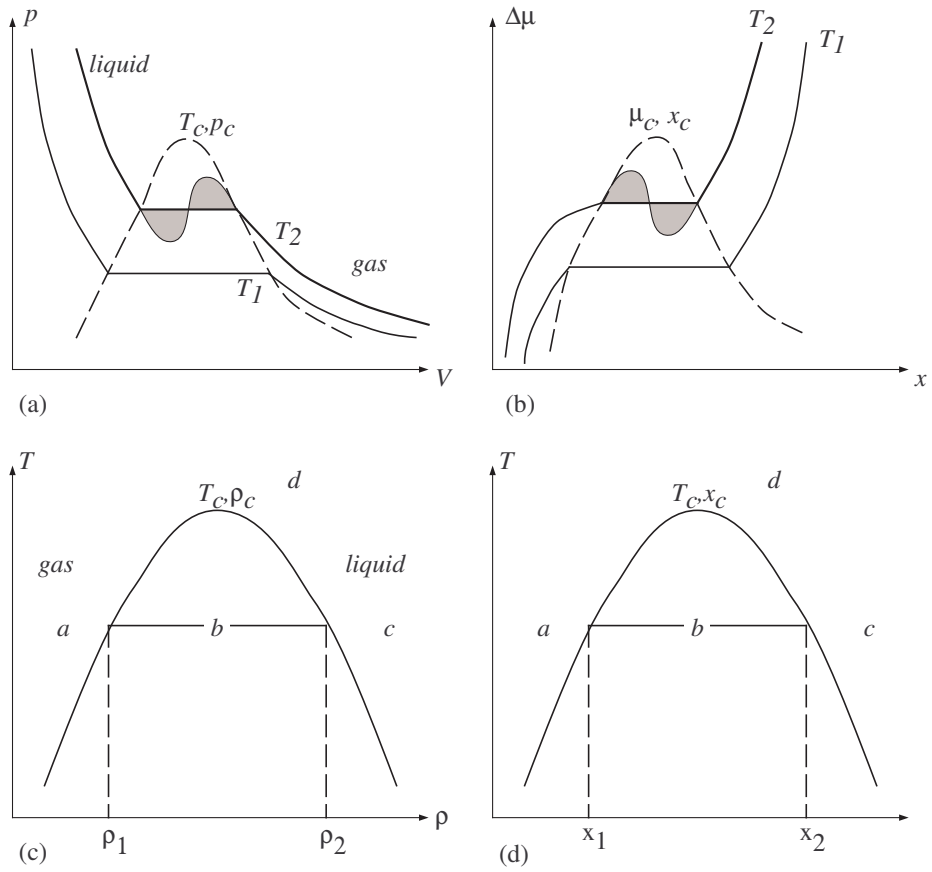


Figure 1.1: The two phase transitions considered in this project. (a) p-V isotherms showing single phase and two phase solution (solid line with flat part) and critical point, $T_2 > T_1$; (b) $\Delta\mu - x$ phase diagram where $x = [0, 1]$ is the mass fraction of two species; (c) $T - \rho$ phase diagram derived from "a"; (d) $T - x$ phase diagram derived from "b"

the melting solid-liquid phase transition is hard to model. In this report we concentrate primarily on *liquid-gas phase transitions* in our fluid simulations and *symmetry changing phase transitions* in solid systems (see below for the latter).

The type of phase transition, in Figure 1.1(a), due to density changes, can be represented using phase diagrams such as shown in Figure 1.1(c). There are several types of phase diagram: pressure-temperature, pressure-volume, temperature-volume. Here we concentrate on temperature-volume, or rather on temperature-density ($T - \rho$). If the system is put at point a, with low density and moderate temperature, it will stay as a single low-density phase which we call gas. If the system is put at high density c, it will stay as a single phase: liquid. However, if we put the system at b, it will spontaneously separate into two phases: liquid and gas. The density of the liquid and gas phases are different (they have the same pressure however) and are calculated from the intercept of a horizontal line thorugh c with the *coexistence curve*, as in Figure 1.1(c), giving values ρ_1 and ρ_2. At high temperatures, such as point d, the system remains as a single phase and there is no concept of a liquid or gas. The top of the coexistence curve is the critical point showing the critical density and critical temperature. This phase diagram can be calculated from the "raw" p-V curves in Figure 1.1(a). Real experimental phase diagrams for argon are shown in Chapter 6.

The other type of phase diagram, which can be calculated in MTDATA, involves more than one species: let us say two types of atoms called A and B. We keep the total number density of particles constant but vary the type of particle according to the fraction x which goes from 0-1, Figure 1.1(b). We then generate a T-x phase diagram much the same as the $T - \rho$ diagram, Figure 1.1(d). Again, points a, b and d result in a single phase but point c results in separation of two phases with different x values each (note: the *overall* ratio x is fixed by the position of point b). These phases coexist at the same temperature.

These two phase transitions may be regarded as the *fundamental phase transitions* of liquid structures. In reality, both types of phase transition will combine and this (combined with increasing the number of species) gives rise to enormously complex phase diagrams. It is MTDATA's job to predict such complex phase transitions. In this report we focus on getting MD codes to predict the two fundamental phase diagrams shown in Figure 1.1.

We have also done some MD simulation of solids in this project: in particular on *piezoelectrics*. These are materials which show coupling between strain and polarisation: when the material is strained it produces a polarisation and stress. Alternatively, applying an electric field can cause strain and polarisation in the material. The exact way this happens is characterised by polarisation-electric field loops (P-E loops) and strain-field loops ($\epsilon - E$ loops). The latter are commonly refered as "butterfly loops".

These P-E and butterfly loops are really a type of phase transition: dramatic jumps in polarisation/strain occur as a result of the system jumping from one free energy minimum to another. Normally, however, when talking about phase transitions in piezoelectrics, one refers to varying the temperature and observing transitions between different symmetries: rhombohedral, orthorhombic, tetragonal and cubic in the case of barium titanate (BTO), the piezoelectric to which we will normally refer in this report. These temperature-based transitions, along with the P-E and butterfly loops are investigated in Chapter 8 and Chapter 9.

Note that, while transforming its symmetry, BTO (for example) remains a solid. Its density hardly varies and it only undergoes a solid-liquid phase transition at very high temperatures. We will avoid such temperatures regimes and consider the "symmetry" phase transitions only in BTO. Thus, we have a list of several phase transitions which we investigate separately: density variation, species fraction variation (liquids), P-E and butterfly loops and symmetry transitions (piezoelectrics).

Finally it should be noted, that none of the MD codes "know" that a phase transition is "meant" to take place at all. In MD, we connect atoms up according to forces which vary with distance. The forces *do not* very with pressure or temperature. Thus, in showing a phase transition, the MD code is already making a non-trivial prediction. If the transition happens at the right temperature, pressure and so on, then this is all the more impressive. The MD code just simulates a box of atoms, it has no knowledge of "liquid" or "gas". Such assignments are left to the human operator and are only meaningful by reference to phase diagrams like Figure 1.1.

1.4 Calculating phase transitions using MD

The most obvious calculation done by an MD code is the evolution of atomic positions and velocities. However, in order to process this mass of data into thermodynamic quantities such as temperature and pressure requires a further calculation, derivable from statistical mechanics. These derivations involve subtle and intricate knowledge of thermodynamics and are described in detail in [3]. Basically we will rely on three algorithms: calculation of temperature, pressure and chemical potential (of each species). Temperature can be calculated by the realisation, from statistical mechanics, that kinetic energy is proportional to temperature. Since the former can be easily assessed from the atoms velocities the system temperature is calculated and averaged over time. So we have

$$\langle \sum_{i=1}^{N} |\mathbf{p}_i|^2/m_i \rangle = 2\langle K \rangle = 3Nk_BT \tag{1.1}$$

where $\mathbf{p}_i$ are the atomic momenta m_i their masses, K the kinetic energy, N the total number of atoms, k_B the Boltzmann constant and T the temperature in Kelvin.

One approach to calculating the pressure would be to run the simulation in a box with the atoms bouncing off the sides of the box. By assessing the momentum impact, one can deduce the pressure applied by the system on surrounding material. However, MD codes, which historically follow from Allen and Tildesley, prefer instead to use *periodic boundaries* in which atoms leaving the box are inserted into the other side. In this case there is no solid box for the atoms to bounce off and we use instead the *virial theorem* to estimate the pressure. It should be noted that the correctness of the virial theorem is still under debate and some researchers have claimed it is incorrect, [6]. The majority, however, consider the theorem to be correct (indeed few would give it a second thought[5]) and it is coded into every molecular dynamics program available. The Virial theorem is:

$$pV = Nk_BT - \frac{1}{3}\langle \sum_i \sum_{j>i} \mathbf{f}_{ij} \cdot \mathbf{r}_{ij} \rangle \tag{1.2}$$

where $\mathbf{f}_{ij}$ and $\mathbf{r}_{ij}$ are the forces and displacements between pairs of atoms. Also, p is the pressure and V is the volume.

Chemical potentials are calculated using the "ghost insertion method" whereby extra "ghost" atoms are inserted into the system. The energy change due to the ghost atom is calculated and this is used to assess the chemical potential:

$$\mu^{ex} = -k_BT\langle \exp(-V_{test}/k_BT) \rangle \tag{1.3}$$

where μ^{ex} is the excess chemical potential in addition to the ideal gas chemical potential for the species. V_{test} is the change in potential energy as a result of inserting the atom. Eq. (1.3) can be used for each species in turn, [7], giving the chemical potential for that species in the presence of the other species (this depends on the nature of the other species as well, it is not a fundamental property of one species). The ideal gas chemical potential is

$$\mu^{id} = -k_BT \log(n_QV/N) \tag{1.4}$$

with the *quantum concentration*

$$n_Q = (mk_BT/2\pi\hbar^2)^{3/2} \tag{1.5}$$

Planks constant $\hbar$ enters into the equation only as a conventional scaling factor: we do not perform any quantum calculations in this project.

These thermodynamic calculations are done within the MD code. However, to calculate actual phase transitions involves an additional program. Firstly, it should be noted that MD codes run at only one setting of pressure, volume, temperature, number of particles and so on. However, to discover phase diagrams or isotherm plots such as Figure 1.1(a), it is necessary to run the simulation many times and

[5] In our experience, the average molecular dynamicist takes a very pragmatic view and is not interested in fundamental questions of statistical mechanics

this requires a "driver" program, which repeatedly calls the MD code, gathers results from all the runs and performs additional thermodynamic calculations to establish the phase transitions themselves. To see why the latter are needed, look again at Figure 1.1(a). The actual output from the MD code will be the non-monotonic curve, eg, at T_2. This is because the system will remain as a single phase and will not spontaneously split into two phases: the overall maximisation of entropy through phase separation is beyond the scope of the MD/MC code and must be accounted for separately.[6].

In order to obtain the bold flat line in Figure 1.1(a) it is necessary to find the overall free energy minimum for the conditions of volume, temperature and pressure specified. MD codes do not calculate entropy-like quantities like free energy directly but rather derivatives of these, such as pressure and chemical potential. The "driver" program must therefore integrate over the pressure-volume curve (for example) to obtain the free energy: a process which is fully described in Chapter 3. From the flat curve, we can deduce the high and low volume and pressure of phase transition (in the case of Figure 1.1(a)). Running many such isotherms gives the phase diagram. Thus, there are three stages to calculating phase diagrams in MD:

1. Calculate forces on all atoms and update over many time steps
2. Accumulate statistical data and calculate thermodynamic quantities such as pressure, temperature etc.
3. Gather a large number of such results, scanning over volume (or number of particles) and temperature. Determine solutions for minimum free energy (maximum entropy). Hence deduce phase diagrams.

Task 3 is performed by the driver program. Although its task may seem mundane compared to the "core" MD/MC code, such a program can nonetheless become very complicated. Gathering the data from millions of MD runs clearly requires careful organisation. The driver program also needs to set atoms at appropriate initial positions at the beginning of the run. It therefore needs to have built-in "knowledge" of the various molecular or crystal geometries that will be needed. Note that the MD/MC codes used in this project have no built in knowledge of appropriate inter-atomic forces or molecule geometries (they don't even know the mass of each element). All this must be explicitly specified in control files.

1.5 Programs used in this project

The main code we will use is MDL, a molecular dynamics code developed in-house at NPL. MDL has clearly been influenced by DL_POLY and has very similar input files and commands. DL_POLY is a very famous *liquid* MD code and is essentially an implementation of the Allen and Tildesley book [3]: a box of atoms is run according to Newton's laws of motion with periodic boundary conditions. The atoms connect across the boundary according to the minimum image convention which (as shown in Chapter 2) is appropriate to liquids, but not to solid lattices. There is provision for constrained and rigid body dynamics. Many different ensembles are supported in DL_POLY, [4], including NVE, NVT and NPT. The latter ensemble types are implemented in many ways giving a total of 26 ensembles. The DL_POLY manual contains no reference to solids, unit cells, crystal lattices, space groups or any of the concepts associated with solid state modelling. It is clearly intended for liquid/gas simulations. However, in principle, DL_POLY (and hence MDL) *can* be used for solid calculations as shown in Chapter 2. Our experience with DL_POLY for solids, however, has not been very promising. For solids it is preferable to use solid state MD codes such as GULP (see below).

The similarity between DL_POLY and MDL can be seen from Table 1.1 which lists the names of the input and output files. The commands appearing in the input files `FIELD` and `CONTROL` are also very similar for both programs as are the algorithms used including the set of ensembles (although MDL supports "only"

[6] Actually this point is disputed: in Allen and Tildesley [3] it is stated that, in a large MD simulation, phase separation will be observed, but we saw no sign of it in any of our simulations, despite using 4 different MD/MC codes

File type	MDL	DL_POLY
Initial atom positions	config.inp	CONFIG
Inter-atomic forces	FIELD	FIELD
Control parameters	CONTROL	CONTROL
Main output	output.mdl	OUTPUT
Final atom positions	runconf.mdl	REVCON
Restart file	restart.new	REVIVE

Table 1.1: Input and output files for MDL and DL_POLY

8 ensembles). MDL supports almost all of the features of DL_POLY with one exception: the shell model. In MDL, each atom is given a partial charge and electrostatic interactions exists between atoms. In the shell model, atoms have both a positively charged core (representing the nucleus) and a shell (representing the electron cloud). A spring exists between the two holding them together. Core and shell are treated as separate particles and interact in the normal way with the rest of the simulation. MDL's author, Dr Sokhan, did not write a manual for the code. Accordingly, we have written our own manual, Appendix A. Full documentation is supplied with the code which is available within NPL.

Another MD solver used in this report is GULP which, unlike DL_POLY/MDL, is designed specifically for solids. In GULP, atoms may be entered using space groups or with fractional coordinates. GULP can also calculate stress, strain and polarisation directly. These are fairly minor deficiencies, however, and can be overcome with pre- and post-processing codes. Far more fundamental is the way DL_POLY (and MDL) and GULP connect atoms. As we will show in Chapter 2, DL_POLY (and MDL) use the minimum image convention whereby (roughly speaking) atoms only interact within the simulation box. In solid modelling, the simulation box generally represents one or more *unit cells* of a crystal, so connections between the simulation box and surrounding boxes are generally needed up to much further distances. This fundamental limitation makes it extremely difficult to model solids in DL_POLY/MDL.

Both DL_POLY and GULP have an *academic licence* whereby academics can access the code free of charge. Unfortunately, in NPL we are not regarded as academics even when working on non-industrial projects. So these codes are not free to us. This was one of the reasons that Dr Sokhan wrote MDL.

We have written two "driver" codes which repeatedly call the MD programs as described Chapter 3. These are SIMRUNNER and NSIM, which is shown in Figure 1.2. SIMRUNNER varies volume to calculate phase diagrams via pressure while NSIM varies number of particles to calculate phase diagrams via chemical potentials. SIMRUNNER can calculate $T-\rho$ phase diagrams while NSIM can calculate both $T-\rho$ and $T-x$ phase diagrams. Because of the similarity between the MD codes, both drivers can be made to call either MDL or DL_POLY(NSIM can also call TOWHEE)

We have also used two open source codes: TOWHEE (an MC code refered earlier) and FERAM which is an *effective Hamiltonian* code aimed at modelling piezoelectrics. A descrption of this code is given in Appendix B. Effective Hamiltonian codes are a common way to model piezoelectrics in literature. Rather than modelling the whole unit cell of BTO (for example), the unit cell is replaced by a single dipole whose orientation is represented by just 3 coordinates. The effective interaction between such dipoles requires a complex pre-calculation and many input parameters are needed, some only tenuously related to the "real" physics of the system. We have successfully modelled BTO P-E and butterfly loops using this code, Chapter 8.

Finally, we have written an MD code of our own, during the course of this project: LDM. LDM is heavily influenced by MDL but is very short (only 2000 lines long). It was written for testing the operation of MDL to ensure that (a) MDL works and (b) we have understood how MDL works sufficiently to simulate what we *think* we are simulating. LDM is unlikely to be much use to others, but we are happy to supply it if needed.

A full list of codes is given in Table 1.2.

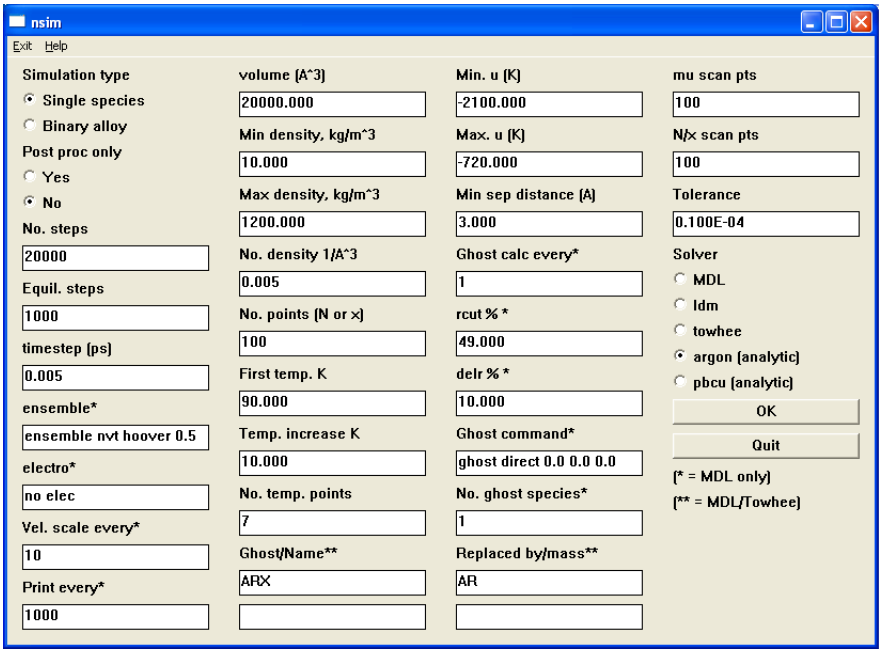


Figure 1.2: The driver program, NSIM. The user can enter required densities and so on in SI units and is shielded from the details of the MD/MC solver

program	origin	Availability
MDL	NPL written	`T:\public\Functional\johnb\mdl`
LDM	NPL written	On request
DL_POLY	Daresbury laboratory, UK	Free for academics only
GULP	Curtin University of Technology, Australia	Free for academics only (Sold by Accelrys)
TOWHEE	University of Minnesota, USA	Open Source
FERAM	Tohoku University, Japan	Open source
SIMRUNNER	NPL written	`T:\public\Functional\johnb\simrunner`
NSIM	NPL written	`T:\public\Functional\johnb\nsim`

Table 1.2: Programs used in this project

1.6 The structure of this document

We begin, in Chapter 2, with a description of how MD works. Some of this repeats concepts described in [3] but there are some additional clarifications which we hope will be useful. In particular, we emphasise the distinction between liquid/gas and solid state molecular modelling. There is also a description of the "excluded bonds within molecules" concept which is widely used but rarely described in detail. We also compare and contrast some of the solvers we will use in this report. In Chapter 3 we show how to use the "raw" data output by MD/MC solvers to calculate phase diagrams including $T - \rho$ and $T - x$ phase diagrams. To this end, we make use of analytical approximations for systems corresponding to an argon system and copper/lead mixture respectively. These analytical results are compared to simulation results in chapters 4 and 7.

Chapter 4 describes the simulation results for argon, comparing 4 simulation programs and commenting on which is most accurate. We also illustrate the inverse modelling facilities of SIMRUNNER by fitting the simulation results to experiment by adjusting the Lennard-Jones parameters automatically. Various difficulties forming phase diagrams are highlighted and overcome.

In Chapter 5 we move on to multi-atom molecules and describe water simulations. Good agreement is shown with experimental water phase diagrams and also with published radial distribution functions. Some discussion is given regarding adjusting numerical parameters to give optimal results.

In Chapter 6 we show how accurate models of Ar and CO_2 are important for real world industrial problems. The PVT behaviour of CO_2 as predicted by MDL, is input to MTDATA to give an improved prediction of phase diagrams. This chapter shows how molecular dynamics can improve the accuracy of MTDATA results. Comparisons are shown between MDL and experimental PVT and enthalpy curves for CO_2. Extreme pressures and temperatures and also considered in this chapter. Such regimes are of profound industrial importance but no experimental data exists. Thus, the MDL results are of direct industrial importance.

In Chapter 7 we consider the calculation of a lead/copper binary alloy using MDL and TOWHEE both driven by NSIM. Results are compared to experimental data. The technique shown here could be readily extended to any number of species in complex mixtures.

The final two chapters deal with MD modelling of piezoelectrics. In Chapter 8, Barium Titanate (BTO) is modelled using the effective Hamiltonian code, FERAM. This code "compacts" the unit cell into a single dipole. Phase diagrams and hysteretic P-E and butterfly loops are calculated and compared to experiment. Comparison is also given to Landau-Devonshire theory with discussion on the relative accuracy of the two techniques.

Finally, Chapter 9 shows another attempt to model BTO, this time using more conventional MD. The full unit cell of BTO is modelled directly using GULP and comparison given to the results of the previous chapter. This chapter, which is a preliminary study, shows calculations only at zero temperature (energy minimisation).

Chapter 2

How MD works

The introduction has sketched out some of the basic concepts of MD. Here we want to develop these ideas in more detail. For more comprehensive instructions, the reader is refered to [3].

2.1 Basic concepts

An MD code consists of a time loop. Within the loop two tasks must be accomplished:

1. Calculate the forces acting on atoms
2. Update the atoms according to the dynamics scheme

We will consider task 2 first which is easier to describe. Having calculated the forces on all the atoms, the atomic positions and velocities are updated. The simplest way to do this is according to Newton's laws of motion in which case we will have an NVE ensemble. As mentioned in the introduction, NVT and NPT ensembles are commonly used also. In fact there are many different ways to code up the NVT and NPT ensembles. Table 2.1 shows a comparison of ensembles available in DL_POLY and MDL.

The NST algorithms are for constant Number of particles, Stress and Temperature. In a sense, however, this is something of a misnomer since the user is not free to set the required stress tensor. The user may only set the trace[1] of this tensor (ie the pressure). The distinction between the NPT and NST is that the latter allows the simulation box to change its shape and become, in general, a parallelapiped. The NPT ensemble only allows the box to be scaled up by a single multiplication factor in all directions, from its initial shape. Normally, in this case, the box is set initially as a cube and remains a cube but with variable volume.

We will not go into much detail on these ensembles since they are fully described in the (commendably thorough) DL_POLY manual. One can see, looking at Table 2.1 just how many ways there are to do MD! The different algorithms (Hoover, Langevin etc) are almost entirely different from one another mathematically. To the user, however, it is easy to switch between algorithms by simply specifying the needed ensemble in the input file. Note that the NVT algorithms generally require an extra parameter: the relaxation time of the thermostat. We have found that the system behaviour is not very sensitive to the value chosen but its worth trying different values. The NPT/NST ensembles require one further parameter relating to the relaxation time of the barostat.

In NVT, the user specifies a target temperature, while in NPT/NST, both temperature and pressure must be specified. In all cases, the actual instantaneous temperature and pressure are also calculated so the user can check to see if the algorithm used was successful in maintaining the pressure/temperature required. In NVE, a temperature is still specified and the atomic velocities are set according to a Maxwellian distribution based on this value. Thus, in NVE, the initial temperature is specified but will tend to change to another value as the system achieves equilibrium.

[1] Sum of matrix diagonal elements

Ensemble	Available in DL_POLY	Available in MDL
NVE	yes	yes
NVT Evans	yes	yes
NVT Gauss	no	yes
NVT Langevin	yes	no
NVT Berendsen	yes	yes
NVT Hoover	yes	yes
NPT Langevin	yes	no
NPT Berendsen	yes	yes
NPT Hoover	yes	yes
NPT MTK	yes	no
NPT Langevin	yes	no
NST Berendsen	yes	yes
NST Hoover	yes	yes
NST MTK	yes	no

Table 2.1: Ensembles available in DL_POLY and MDL. From [4]

A further choice to make is how to update the atoms. There are two commonly used schemes available: the velocity Verlet scheme and the leapfrog scheme [3]. DL_POLY implements both schemes which, when combined with the ensembles of Table 2.1 means a total of 28 atomic update schemes have been written in DL_POLY! MDL only uses the leapfrog scheme. In reality, the user is unlikely to experience any difference between the schemes except perhaps in extreme cases.

Considering now task 1, there are many different types of forces that can exist between atoms and considerable coding effort is needed to implement all these forces types. We may divide these forces into several kinds:

1. Van der Waal (VDW) forces existing between different particular pairs of atoms depending on the species of each atom. These are often refered as short range or non-bonded forces. These forces are applied between the atoms in different molecules but not, generally, within a particular molecule.

2. Electrostatic forces. These long range forces apply between all atoms which are charged, irrespective of species. Since the force is long range $1/r$ it is essential to apply special evaluation techniques such as the Ewald sum (see below)

3. Forces that exist within molecules. These include flexible bonds such as harmonic springs etc. It is also possible to set constraints whereby two nominated atoms are held a certain distance apart. The ultimate case of this is when the whole molecule is held rigid in which case rigid body mechanics can be used.

Note that the VDW forces exist between all atoms of the specified species irrespective of which atom it is. Electrostatic forces are even more general purpose, existing between all atom pairs with charge. On the other hand, bond/constraint forces exist between *particle pairs of numbered atoms*. Figure 2.1 shows the different forces that can apply between water molecules. Note that, as a general convention, *setting a bond between two atoms deactivates the VDW and electrostatic forces which would otherwise exist between those atoms*. So, in the figure, there is no VDW or electrostatic force between O and H in the same molecule. Such forces are allowed between atoms in different molecules. This convention is used in MDL and DL_POLY but is not really explicitly mentioned either in documentation or in textbooks. It is hinted at in both the DL_POLY manual and in [3] but never stated explicitly.[2]

Figure 2.2 shows what would happen if the intra-molecular bonds were *not* specified. In this case, VDW and electrostatic forces would apply between all pairs of atoms.

[2]This sort of confusion shows how it is sometimes necessary (however laborious) to read the program's source code: not everything is specified in the documentation and one can easily end up simulating a different problem than one thought

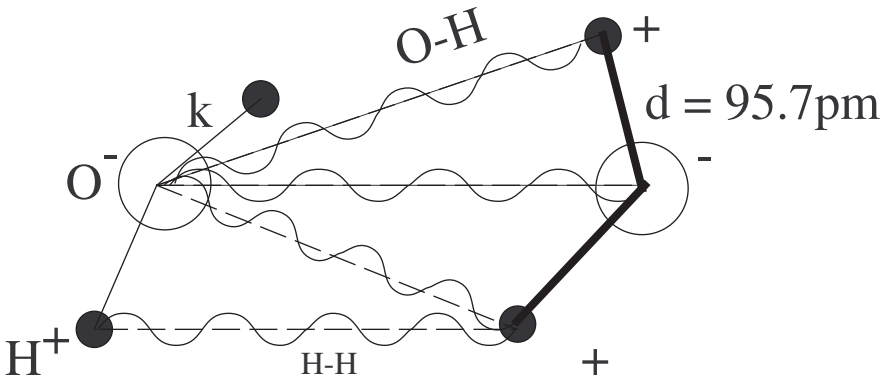


Figure 2.1: Interactions between two water molecules. Spring bonds are shown as solid lines (spring constant k), van der Waals shown as dashed lines, electrostatics shown as wavey lines, constraints shown as thick lines. (not all interactions are shown)

ltpvdw	specifier	name	parameters	pair potential (E_0)
1	12-6	12-6 potential	a,b	$(a/r^6 - b)/r^6$
2	lj	Lennard-Jones	ϵ, σ	$4\epsilon(\sigma/r)^6[(\sigma/r)^6 - 1]$
3	nm	n-m potential	a,b,c,d	$a/(b-c) \cdot (c(d/r)^b - b(d/r)^c)$
4	buck	Buckingham	a,b,c	$a\exp(-r/b) - c/r^6$
5	bhm	Born-Huggins-Meyer	a,b,c,d,e	$a\exp(b(c-r)) - d/r^6 - e/r^8$
6	hbnd	H-bond	a,b	$(a/r^2 - b)/r^{10}$
7	mors	Morse potential	a,b,c	$a[(1 - \exp[-c(r-b)])^2 - 1]$

Table 2.2: van der Waals potentials. The parameter column shows the list of parameters in sequence which should be given in `FIELD` file. So a 12-6 potential must be entered `12-6 a b`

MD codes go to considerable trouble to exclude "false" VDW and electrostatic forces between atoms already connected by intra-molecular bonds. They maintain "exclude lists" of atoms so affected. To find out more details, search the DL_POLY manual for "exclude" or "exclusion".

A list of the various VDW forces and intra-molecular bonds available in MDL is shown in Table 2.2 and Table 2.3. This selection is fairly typical: the most common VDW interaction is the Lennard-Jones. Buckingham potentials are also common especially in solids. The most common intra-molecular bond is probably the simple harmonic bond. However, in our simulations of multi-atom molecules we have generally eschewed intra-molecular bonds entirely instead setting the whole molecule as a rigid body.

keybnd	specifier	name	parameters	pair potential (E_0)
1	harm	Harmonic bond	k,l	$\frac{1}{2}k(r-l)^2$
2	mors	Morse potential	a,b,c	$a[(1 - \exp[-c(r-b)])^2 - 1]$
3	12-6	12-6 potential	a,b	$(a/r^6 - b)/r^6$
4	rhrm	restrained harmonic	a,b,c	$\frac{1}{2}a\min(\lvert r-b\rvert, c)^2 + ac\max(\lvert r-b\rvert - c, 0)$
5	quar	quartic potential	a,b,c,d	$\frac{1}{2}a(r-b)^2 + \frac{1}{3}c(r-b)^3 + \frac{1}{4}d(r-b)^4$

Table 2.3: Intra-molecular bonds. The parameter column shows the list of parameters in sequence which should be given in `FIELD` file. So a harmonic potential must be entered `harm k l`

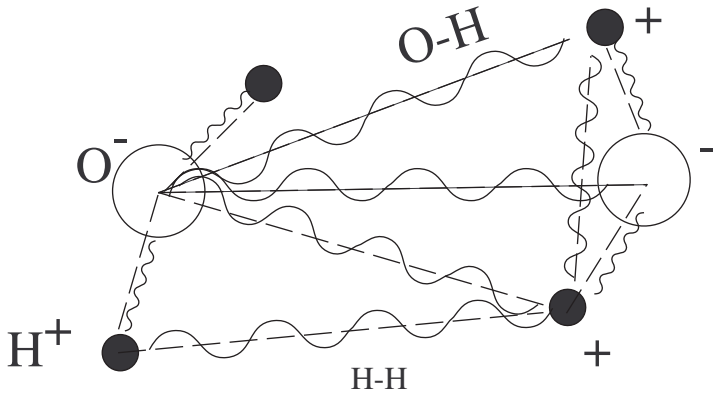


Figure 2.2: Interactions between two water molecules. van der Waals shown as dashed lines, electrostatics shown as wavey lines.

2.2 Shell model

So far, we have assigned partial charges to each atom. The shell model is more realistic in that both a negatively charged cloud and a positively charged core are specified for each atom. The setup is shown in Figure 2.3. The core and shell charges are chosen so that the total charge of each atom is appropriate. In the case of water, for example, the total charge on hydrogen atoms should be positive and that of the oxygen atoms negative. A simple spring connects the core and the shell (although, in general, anharmonic and/or anisotropic springs may be used). Note that intra-molecular bonds and VDW forces apply only between *shells*. Also, there is *no* electrostatic interaction between an atom's core and its shell. The only forces applied to a core are: the spring connecting it to its shell and the electrostatic interaction to other atoms (both core and shell).

The shell model may be implemented in one of two ways: with dynamic or static shells. In dynamic shell model, the shells and cores are both assigned mass (shell mass generally much lower than that of the core), both are iterated according to the ensemble chosen. This technique is just like the partial charges technique except that twice as many particles need to be considered. In that static shell model, shells have zero mass and are relaxed to their minimum potential energy positions between core position updates. Cores are updated according to the usual dynamic rules. This approach is time consuming since many shell relaxation steps are needed between each MD timestep.

2.3 The minimum image convention

We have already mentioned that MD in liquids is generally carried out with periodic boundary conditions. It turns out this boundary condition has a profound impact on the way forces are calculated between atoms. This situation is illustrated in Figure 2.4. Here we have a simulation box of side L_x, L_y, L_z with 3 diatomic molecules. A bond exists between the atoms in each molecule. Because of the periodic boundaries, the simulation box we are modelling is conceptually surrounded by an infinite number of identical boxes as shown in the figure. In the minimum image convention, VDW forces (which are perhaps the most important in liquid simulations) are applied *between nearest atoms in the infinite array*. The figure illustrates all 12 VDW interactions that would exist in this case. It can be seen that some interactions are between the "central" box being simulated (shown grey) while some exist between atoms in the simulated box and atoms in neighbouring boxes.

For example, consider atom 5. A VDW interaction must be established with all the other atoms,

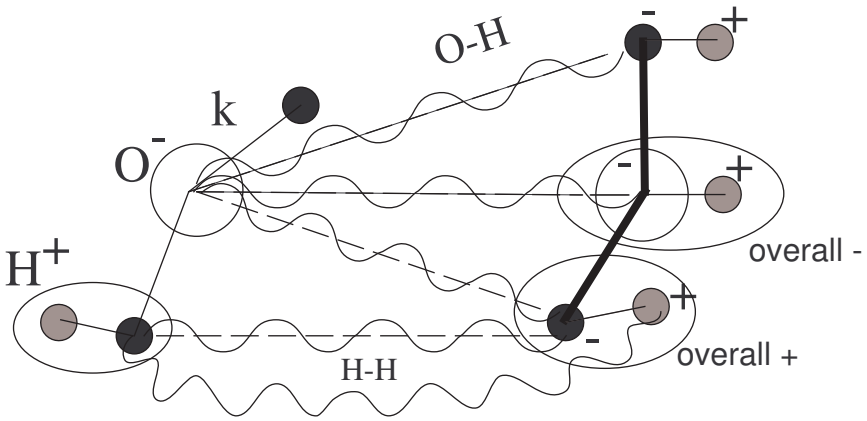


Figure 2.3: Shell model. Atom cores are shown grey. Solid lines are intra-molecular bonds; dashed lines VDW; wavey lines electrostatic; bold lines constraints. In 3 of the atoms, cores as well as shells are shown.

excluding only atom 6, to which it is attached by a intra-molecular bond. The nearest copy of atom 4, is atom 4 itself in the central box. However, the nearest copy of atom 1, is $1'$ which exists in the next box along the y-direction. In total atom 1 connects to atoms $(1', 2', 3', 4)$. Of course, we do not need to store the positions of the "primed" atoms: they are simply the unprimed atoms with a box vector added (in this case it is simply a shift of L_y along the y-direction).

Working out which is the closest atom out of all possibilities might seem a daunting task but actually it is simply accomplished using a standard function `nint` which is available (perhaps under a different name) in all programming languages. This function simply returns the nearest integer, eg `nint(-1.2)=-1`, `nint(1.7)=2` etc. The simplest algorithm for evaluating VDW forces can be written in the pseudocode shown in Figure 2.5. In the figure, there are n atoms and the atom positions are stored in arrays `rx,ry,rz`. Total forces are accumulated into corresponding arrays `fx,fy,fz`. The subroutine `fvdw` calculates the energy (`en`) and force factor `ffac` between atoms depending on the atom names and the distance between them. The energy functions come straight from Table 2.2 and the force factor is

$$f_{fac} = \frac{1}{r}\frac{\partial U}{\partial r} \tag{2.1}$$

The algorithm Figure 2.5 shows that forces are calculated from

$$\mathbf{f}_i = f_{fac}\mathbf{r}_{ij} \quad \mathbf{f}_j = -f_{fac}\mathbf{r}_{ij} \tag{2.2}$$

Notice that potential energy and the virial are accumulated at the same time the forces are applied to atoms. As a slight extra refinement, forces are only applied between atoms separated by distance less than r_c (shown as `rcut` in the algorithm).

Of course, a real algorithm would be more complicated. For one thing interactions between bonded atoms need to be excluded and care should be taken to avoid unnecessary square root operations. Nonetheless, Figure 2.5 represents the core of an MD program and it is remarkably simple.

An important point to note is that the algorithm Figure 2.5 takes N^2 steps to complete. Updating atom positions is linear in N and so the force calculation dominates in large N systems. It is in this area that most refinements have been introduced to improve the calculation speed, we will mention some of these briefly and then refer to [3] for more details.

Although the algorithm excludes atoms at distance more than r_c it would be better if it did not waste time calculating these distances in the first place. The *Verlet neighbour list* method maintains a list of all

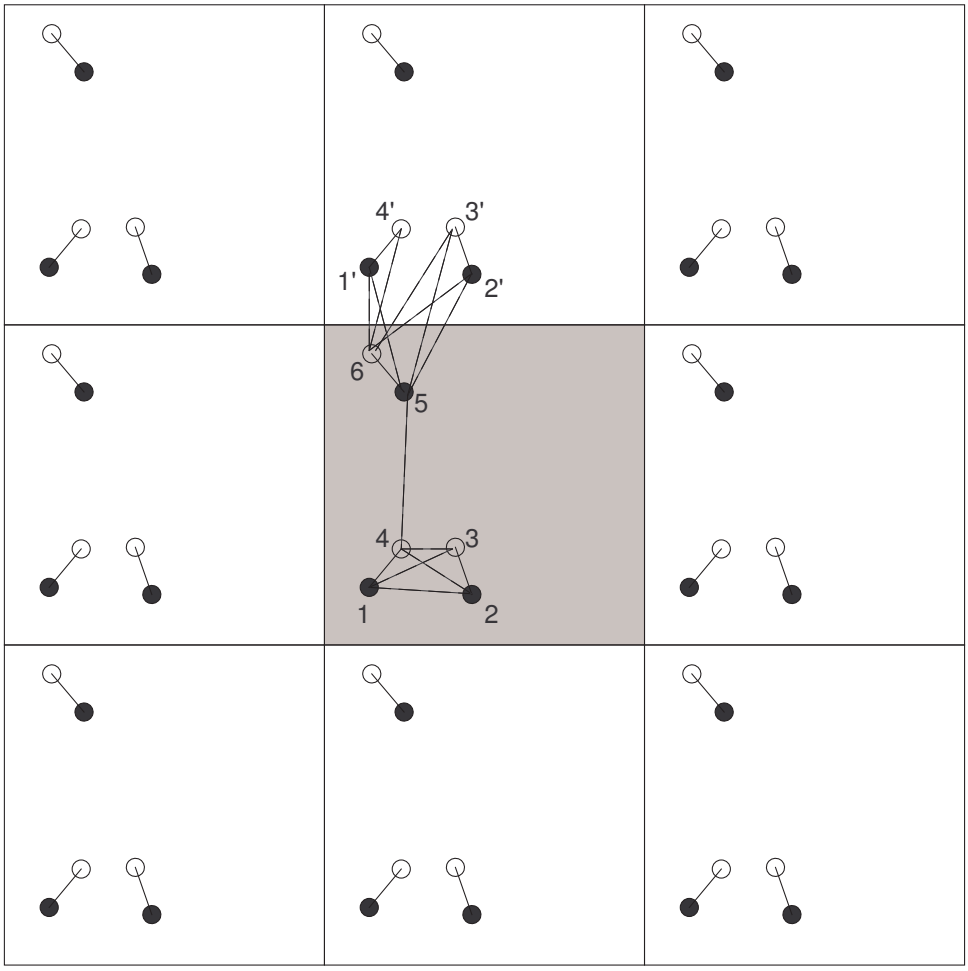


Figure 2.4: The minimum image convention. Intra-molecular bonds are shown solid, VDW interactions are dashed.

```
energy=0; virial=0

fx(1:n)=0; fy(1:n)=0; fz(1:n)=0

do i=1,n-1
   do j=i+1,n

      anamei=aname(i)
      anamej=aname(j)

      rxij=rx(j)-rx(i)
      ryij=ry(j)-ry(i)
      rzij=rz(j)-rz(i)

      rxij=rxij-Lx*nint(rxij/Lx)
      ryij=ryij-Ly*nint(ryij/Ly)
      rzij=rzij-Lz*nint(rzij/Lz)

      dist=sqrt(rxij**2+ryij**2+rzij**2)

      if (dist < rcut) then

         call fvdw(dist,anamei,anamej,en,ffac)

         energy=energy+en
         virial=virial-ffac*dist**2

         fx(i)=fx(i)+ffac*rxij
         fy(i)=fy(i)+ffac*ryij
         fz(i)=fz(i)+ffac*rzij

         fx(i)=fx(i)-ffac*rxij
         fy(i)=fy(i)-ffac*ryij
         fz(i)=fz(i)-ffac*rzij
      endif

   enddo
enddo
```

Figure 2.5: Algorithm to determine VDW forces using the minimum image convention (in Fortran-like pseudocode)

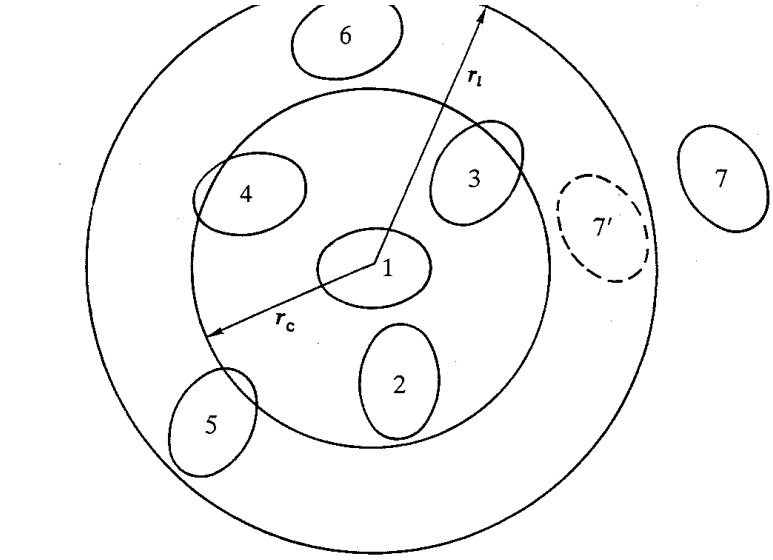


Figure 2.6: Verlet neighbour list method. From [3]

the closest atoms to each given atom (taking account the minimum image convention of course). Only these atoms are considered when it comes to VDW calculations. The list is periodically updated. In this algorithm, the cutoff radius is surrounded by a "skin" of thickness $r_l - r_c$ as shown in Figure 2.6. When an atom penetrates this extra thickness it is time to update the neighbour list.

Another method is the *link cell* algorithm in which the simulation box is divided into sub-boxes. The atoms are sorted into these boxes using a recursive algorithm similar to a linked list Figure 2.7. When considering which atoms may be in range of VDW calculations, only atoms in the same box and neighbouring boxes need be considered. As atoms move around, they need to be resorted into their appropriate boxes again. The sorting algorithm is very fast and can be done at each time step. The link cell method is most effective when there are a large number of atoms and the box size is much greater than the cutoff length r_c. In this case, it is pointless to calculate interactions between atoms at opposite ends of the box.

MDL chooses either the Verlet algorithm or the link cell algorithm automatically depending on the size of the box and number of atoms. The shell thickness `delr` $= r_l - r_c$ must be entered. The technique employed by DL_POLY depends on which version one considers. Roughly speaking, DL_POLY2 uses the Verlet list and DL_POLY3 uses the link cell algorithm. The primary difference between the two is ease of parallelisation. The link cell algorithm is well suited to this since it involves *domain decomposition*. Thus, DL_POLY3 is considered preferable for very large problems running on massively parallel computers with millions of atoms. Note that MDL does not currently support parallel computing and runs on one computer only. However, the driver code SIMRUNNER will cause different instances of MDL to run on different processors if available (on a single computer).

2.4 Calculation of electrostatic interactions

Due to periodic boundary conditions, a conceptual infinite array of identical simulation boxes surrounds the "central box" we actually simulate, Figure 2.4. So far, we have not taken this construction very seriously: it simply allows us to account for VDW interactions within the main simulation box by shifting atoms along box lengths to conceptual image boxes. Since electrostatic interactions are much longer-ranging than VDW interactions, to calculate these forces and energies, we need to account for *all* inter-

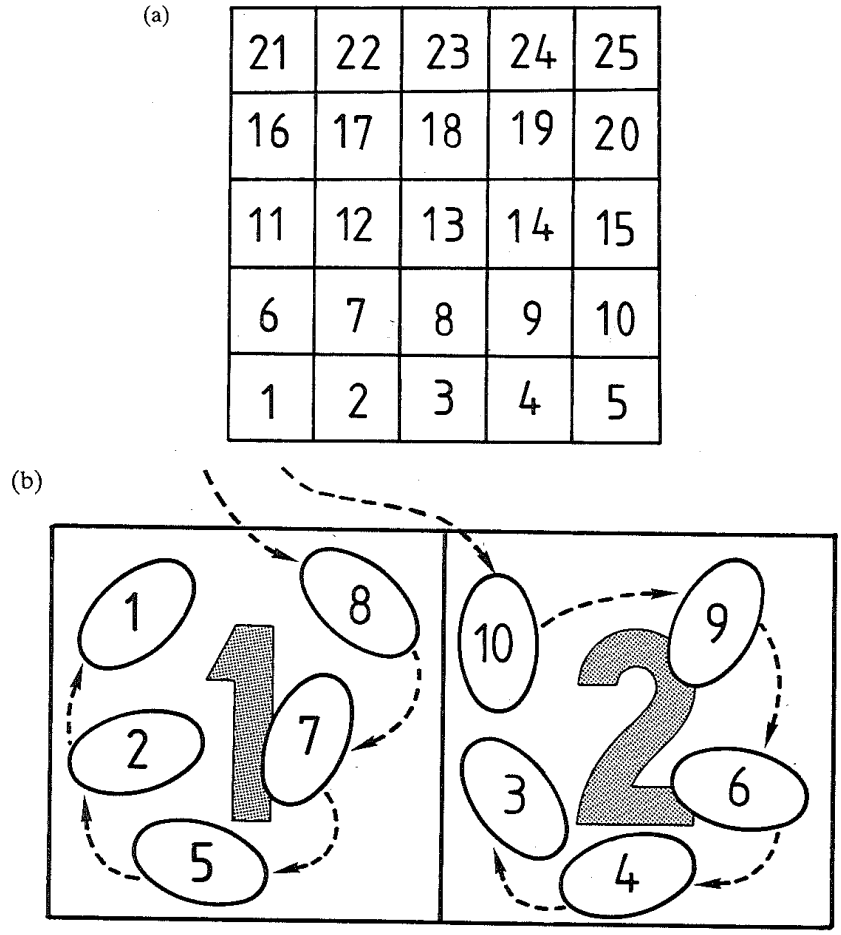


Figure 2.7: Link cell method (a) The simulation box is divided into sub-boxes; (b) Close up of cells 1 and 2 showing atoms and linked list structure. Taken from [3]

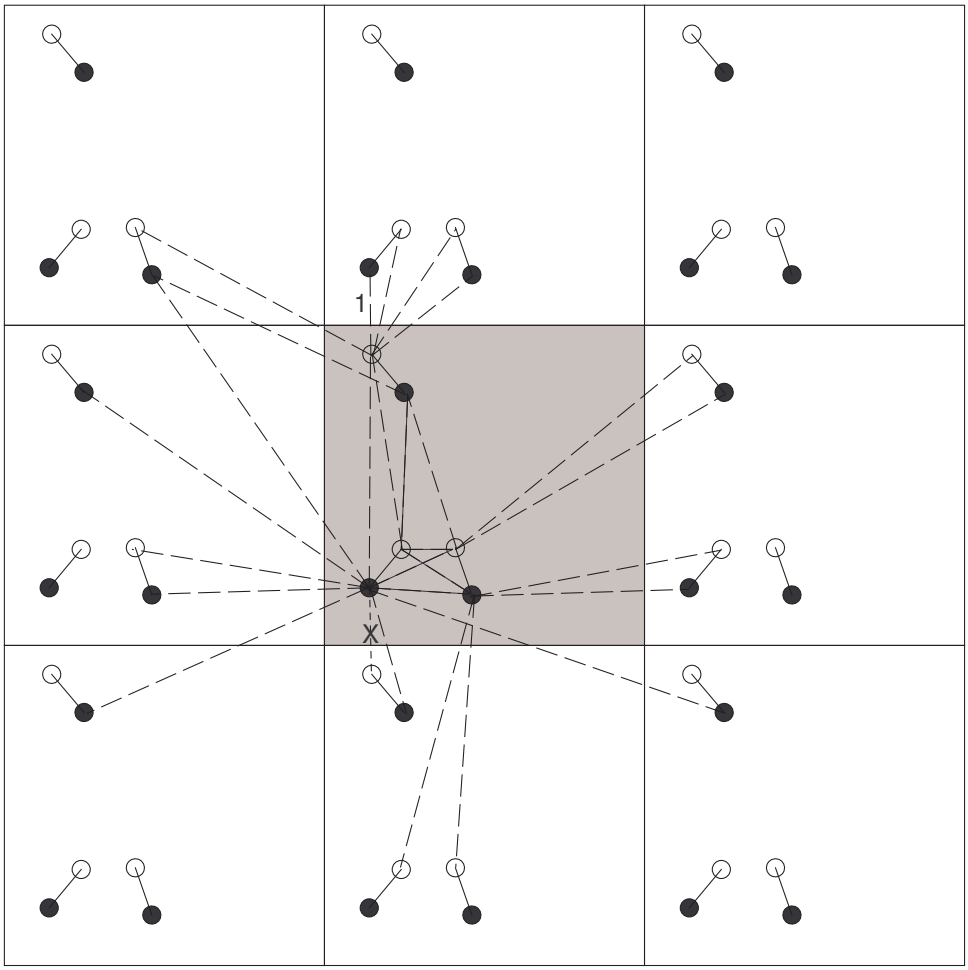


Figure 2.8: Calculation of electrostatic interactions (only some are shown). The interaction shown "X" should not be included as this would be double counting of the interaction "1"

actions between the simulation box and surrounding boxes, potentially going out to very large distances (not just the nearest set of surrounding boxes). That is, we regard the array of image boxes as actually "existing" and carrying potential energy, rather like a charged crystal lattice. [3] The situation is illustrated in Figure 2.8. *The minimum image convention no longer applies here:* atoms are not simply connected to the nearest image of each other atom but *to all images, as well as the atoms in the central simulation box.* Note, however, that each interaction must start and/or end on an atom in the simulation box. We do not connect pairs of atoms image boxes to each other. Also it is important not to duplicate interactions, eg, the interaction marked "X" should not be included since this is already accounted for by interaction "1". The electrostatic energy we calculate should be the energy *per simulation box.* That is, if we were to take the dashed lines emanating from the central square in Figure 2.8 and copy them into all the image boxes, then each atom would be connected exactly once to every other atom. In the case of Figure 2.8, this would give us 9 times as much energy as the actual construction in the figure, so the construction shown gives the correct electrostatic energy per simulation box.

The electrostatic energy can be calculated from

$$U = \frac{1}{8\pi\epsilon_0}{\sum_{\mathbf{R}}}' \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{q_i q_j}{|\mathbf{r}_{ij}+\mathbf{R}|} \tag{2.3}$$

where $\mathbf{R} = l\mathbf{a} + m\mathbf{b} + n\mathbf{c}$ is a lattice vector and $\mathbf{a}$ etc are the box side vectors ($\mathbf{a} = L_x\mathbf{x}$ etc). The prime above indicates that $i = j$ should not be included in the $\mathbf{R} = 0$ term of the summation. The above algorithm will double count each interaction so an extra factor of $1/2$ has been included in the overall factor.

However, a direct summation like (2.3) will *not* normally give a correct result since the summation is only conditionally convergent, [8]. The standard way to calculate the electrostatic energy is using a reciprocal space approach called the Ewald summation:

$$\begin{aligned} U &= \frac{1}{2V_0\epsilon_0}\sum_{\mathbf{k}\neq 0}^{\infty}\frac{e^{-k^2/4\alpha^2}}{k^2}\left|\sum_j^N q_j e^{-i\mathbf{k}\cdot\mathbf{r}_j}\right|^2 \\ &+ \frac{1}{4\pi\epsilon_0}\sum_{n<j}^{N^*}\frac{q_j q_n}{r_{nj}} erfc(\alpha r_{nj}) \\ &- \frac{1}{4\pi\epsilon_0}\sum_{molecules}\sum_{l\leq m}^{M^*} q_l q_m\left[\delta_{lm}\frac{\alpha}{\sqrt{\pi}} + \frac{erf(\alpha r_{lm})}{r_{lm}^{1-\delta_{lm}}}\right] \end{aligned} \tag{2.4}$$

Here $\mathbf{k}$ is a reciprocal lattice vector $\mathbf{k} = p\mathbf{a}^* + q\mathbf{b}^* + r\mathbf{c}^*$ with $\mathbf{a}^*, \mathbf{b}^*, \mathbf{c}^*$ the usual reciprocal space lattice vectors and α the reciprocal distance general taken as $5/L$ where L is the shortest box side length. This parameter α must be large enough so that the real space summation (second summation) gives a small contribution. The major contribution should come from the first term (reciprocal lattice summation). The last summation is a correction factor where we subtract off intra-molecular electrostatics which should be switched off by the presence of bonds. Here sum over N^* means calculate interactions between all atoms *but exclude interactions where bonds are present.* The sum with M^* means consider only such "excluded" electrostatic interactions which are then subtracted from the total. Note that the last term also includes a self interaction term which occurs in the reciprocal space summation and must be subtracted.

2.5 Atomic updates in the minimum image convention

Figure 2.9 shows what happens when some of the molecules exit the simulation box. In this case molecules from neighbouring boxes enter the simulation box. There are two ways to store the atoms that we actually

[3] This is a rather profound conceptual change. Our simulation box is not isolated anymore but considered to be surrounded by an infinite amount of material similar to the simulation box itself.

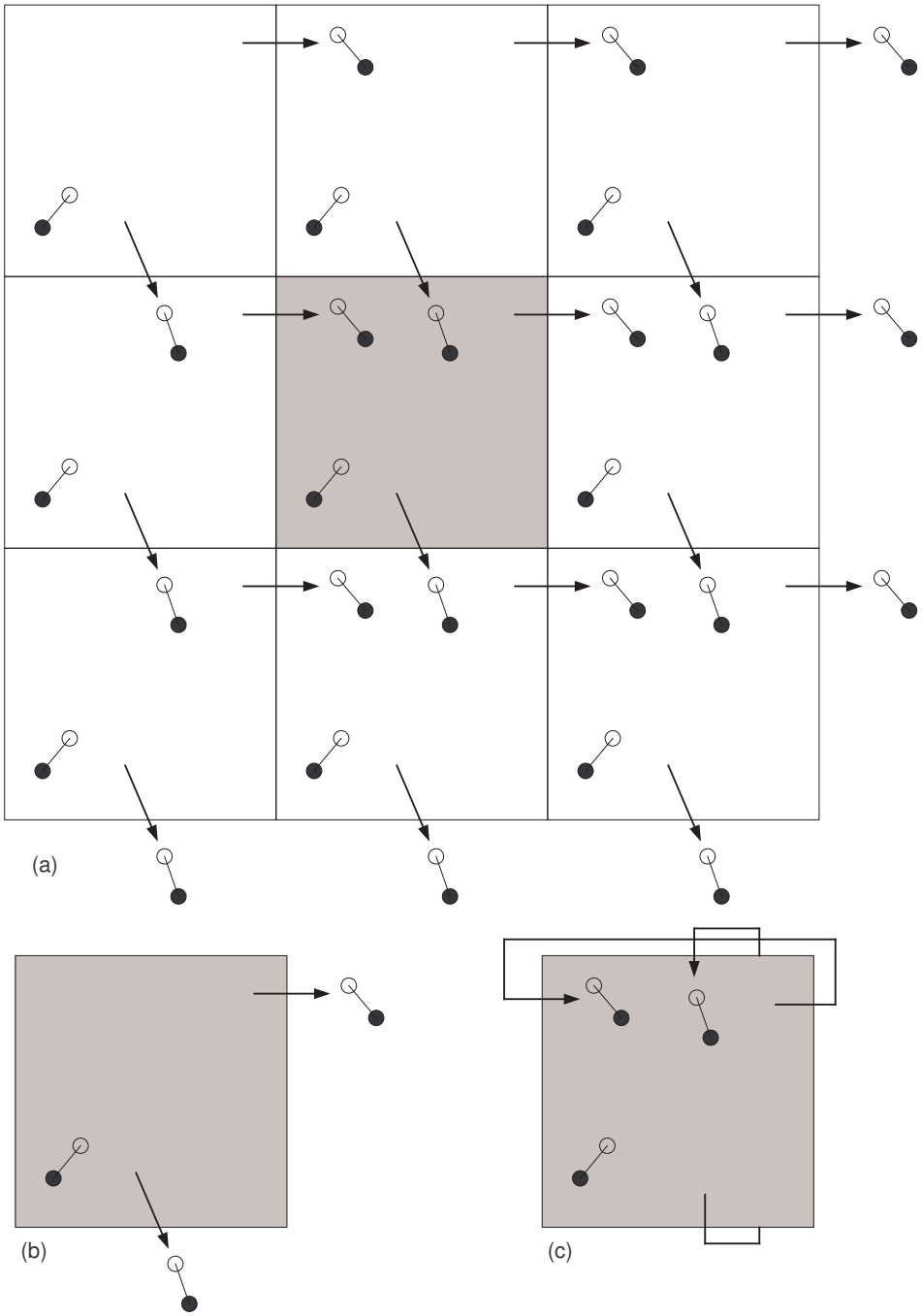


Figure 2.9: (a) Illustration of what happens when atoms move outside the simulation box (shown grey). When atoms cross the boundary of their "own" box, an arrow is shown; (b) What we would actually store in the simulation; (c) An alternative approach in which atoms are wrapped back into the box when they leave it

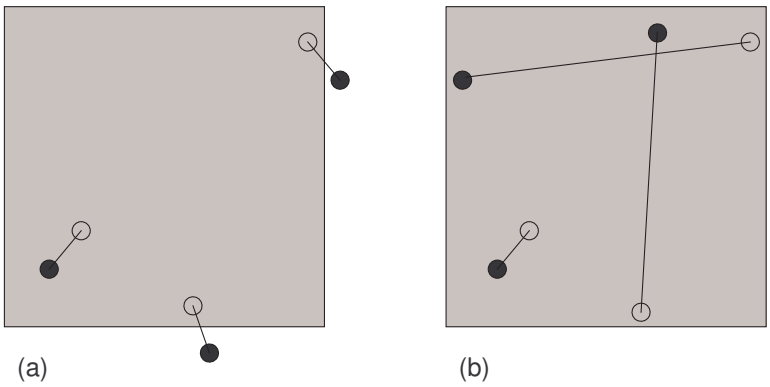


Figure 2.10: (a) What happens when some of a molecule leave the box (no wrapping); (b) same but here atoms are wrapped round the box. The molecules look much to big but this is fixed by Figure 2.5. Intra-molecular forces will still be calculated correctly.

simulate. In Figure 2.9(b) we simply allow the atoms to leave the central box. This does not affect the forces applied to atoms, due to the force calculation algorithm of Figure 2.5. Another approach, Figure 2.9(c), is to make the atoms "wrap round" the simulation box. This approach is more convenient for visualisation since the atoms are confined to a certain area. In other respects, however, the two schemes are identical. They will give the same results (temperature and pressure etc).

One difference between the two storage schemes is shown in Figure 2.10. When one of the atoms in a molecule moves across the boundary, then the "wrap around" scheme produces a strange looking result. This is not a problem provided the `nint` function is used for calculation of intra-molecular bonds/constraints. To be on the safe side, it's wise to process *all* inter-atom displacements using the `nint` function before calculating forces of all kinds (VDW, bonds etc).

A possible problem with the scheme Figure 2.10(b) is that the final position of atoms is often used as the starting point for future simulations. This is fine, but we must *keep the box shape the same*. If the atom positions shown in Figure 2.10(b) are fed into another simulation and the box volume is changed, this might give a nonsensical result. We ran into this problem when we tried to vary volumes so as to get isotherm plots like those of Figure 1.1. The problem only exists for multi-atom molecules, but for safety it is better to prepare a new set of atom positions from scratch for each volume rather than using a set of final atom positions from another simulation.

2.6 Modelling of solids

So far we have spoken primarily of liquid MD modelling. In liquid modelling we have a box of molecules which interact via intra-molecular forces, short range VDW forces (based on the minimum image convention) and long ranging electrostatic forces. The situation in solid crystals is subtly different as shown in Figure 2.11. Here, we are modelling a CsCl type lattice, which is body centred cubic. The simulation box is now a *unit cell* of the crystal (in general not the primitive unit cell). In more complex simulations, a whole number of unit cells may be simulated in each direction.

Figure 2.11(a) shows the atoms that would actually be simulated in this crystal. Here we have chosen to connect nearest neighbour atoms by bonds. These bonds are *quite different to the bonds in liquid simulations*. To specify these bonds we need to nominate atoms at both ends of the bond and also to specify and explicit *unit cell offset*. Thus the bond marked "a" is specified as:

Connect atom 1 to atom 2 via an offset $(-L_x, 0, 0)$.

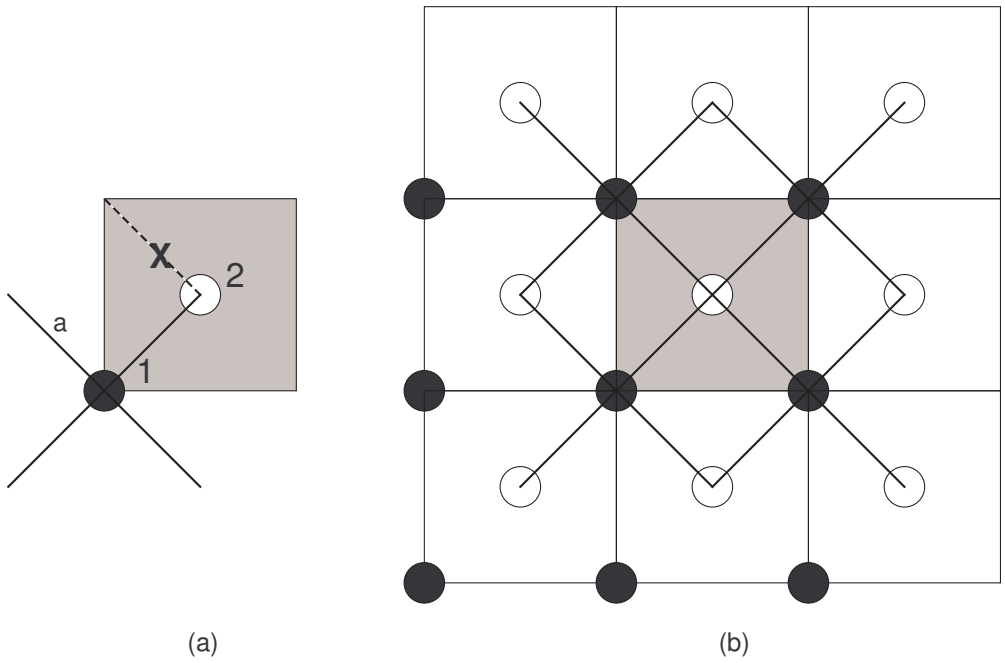


Figure 2.11: Crystal modelling using GULP or similar packages. (a) The simulation unit cell. Here we have set explicit bonds between atoms. The bond marked "X" should not be set as this would be double counting. (b) the consequent lattice with, in this example, bonds connecting all nearest neighbours

This is quite a different paradigm from the minimum image convention of Figure 2.4. No facility to connect atoms by explicit offsets is allowed in DL_POLY or MDL, both of which are completely based on the minimum image principle.

Van der Waals interactions are also common in crystal simulations. Here we would specify

Connect all atoms of type A to all atoms of type B up to 10 Å

for example. Starting with the central simulation unit cell, the system then scans over all surrounding unit cells and tries to connect each pair of atoms found. The program works outward in concentric shells of unit cells until no further interactions can be found within the (here) 10Å cuttoff. Again this is quite different from the minimum image convention VDW of MDL and DL_POLY in which only *nearest atoms in surrounding cells are connected.*

Finally, the electrostatic calculation in crystals is modelled in exactly the same way as in liquids: using the Ewald summation. The Ewald summation naturally relates to lattices so there is no need to generalise the concepts of (2.4).

The paradigm set out here, is that used by GULP: *it is clearly the appropriate paradigm for modelling crystals.*

In principle, it is possible to "trick" DL_POLY or MDL into modelling a crystal by setting up more than one unit cell in the simulation box. This is shown in Figure 2.12. Basically a supercell of size at least $2 \times 2 \times 2$ needs to be set up. MDL/DL_POLY will then set up the minimum image convention in such a way as to create bonds to other supercell images. For example, we would create a bond (either explicitly or through VDW interactions) between atoms 1 and 4. Owing to the minimum image convention, atom 1 would actually be connected to atom $4'$ as shown (etc). This method is clearly tedious and, in fact, was not successful in our attempts. Therefore, we do not recommend the use of these liquid solvers for solid problems.

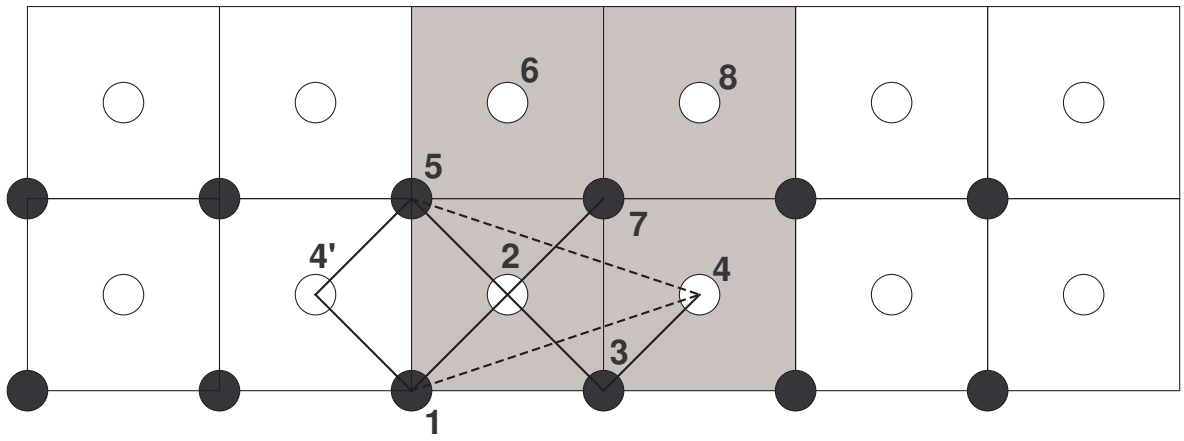


Figure 2.12: How to trick DL_POLY or MDL into modelling a crystal. A $2 \times 2 \times 2$ (minimum) supercell is used. In simulation, bond 1-4 will be replaced by bond $1 - 4'$ etc, according to the minimum image convention

2.7 A note on units

The choice of units in MD/MC programs is a source of endless confusion. No simulation programs use SI units since the numbers would be very small (although this would not cause any technical difficulty). Most are based, internally at least, on the fundamental units of Angstroms, picoseconds, atomic mass unit and electronic charge. This means that the natural unit of energy works out as

$$E_0 = 1.66053886 \times 10^{-23} \text{ J} \tag{2.5}$$

Unfortunately, this unit is not generally used in literature. Instead, energies are normally quoted in Kelvin (divide by k_B), eV, "kJ/mol" or "kcal/mol". The latter two are given in quotes throughout this document since they are not energy densities but *absolute energies*. Specifically, 1 "kJ/mol" is $1000/N_A$ J of energy ($N_A = 6.03 \times 10^{23}$). Similarly, 1 "kcal/mol" is $4200/N_A$ J of energy. If you run MDL, for example, with 100 atoms, you might get a total energy of 1 "kJ/mol". If you run with 200 atoms, you will get 2 "kJ/mol", which shows "kJ/mol" is *not* a specific energy. To get the actual number of kJ/mol (without quotes) you need to divide the "kJ/mol" figure by the number of molecules you are simulating. Or, which amounts to the same thing, multiply by the conversion factor above to get into Joules, then scale to 1 mol by using Avagadro's number.

Force constants are generally entered in terms of energy. So, a spring constant might be in terms of eV/Å^2 for example. In MDL, it is necessary for the user to specify which energy unit he is using and all force constants must follow consistently from this.

Further details on MDL units are given in Section A.9.

Chapter 3

The thermodynamics of phase diagrams

In Chapter 1, we indicated that finding phase diagrams from MD requires three stages of calculation:

1. Calculate forces on all atoms and update over time
2. Accumulate statistical data and calculate thermodynamic quantities such as pressure, temperature etc.
3. Gather a large number of such results, scanning over volume (or number of particles) and temperature. Determine solutions for minimum free energy (maximum entropy). Hence deduce phase diagrams.

In this chapter we wish to examine stage 3, which is the stage not done automatically in MD programs available to us. This further task is done in our "driver" programs and we must therefore develop suitable algorithms and code these up. In order to get an idea what the phase diagrams will look like, we consider approximate analytical forms for the pressure, volume and chemical potential curves. We use realistic values for the parameters of these curves so as to give a good understanding of the range of temperatures, pressures and so on expected. Although we could use arbitrary value to illustrate the basic points, it is worth calculating specific results since, in general, the phase transition area occupies a small area of pressure/volume (etc) space and it is important to have at least a rough idea which areas to look. We will therefore consider two problems: a single species (which we will take to be argon gas: a very simple case) and a binary alloy system: a lead/copper mixture. In each case we will work out the thermodynamic equilibrium state and phase transition graphs and show how these can be derived from MD results.

3.1 Van der Waals Gas

Consider, then, the *van der Waals* equations of state, for a single-species system:

$$(p + N^2 a/V^2)(V - Nb) = Nk_B T \tag{3.1}$$

where p, V, T, N are the pressure, volume, temperature, number of particles. The form of the curve is dictated by the parameters a and b which roughly represent the attraction between atoms and the finite volume occupied by each atom respectively. It can be shown that this equation results in the following critical point (cf Figure 1.1):

$$p_c = a/27b^2; \quad V_c = 3Nb, \quad T_c = 8a/27bk_B \tag{3.2}$$

As a first guess we can choose a and b values which give a suitable critical point – we will comment in Section 3.5 whether this is the best choice. Since there are only 2 parameters we cannot fit all three critical parameters so a compromise will be needed. At first, we fit the critical temperature and pressure. For argon, these are 150 K and 5 MPa respectively. Solving gives,

$$a = 3.64 \times 10^{-49}\mathrm{Jm}^3 \quad b = 5.2 \times 10^{-29}\mathrm{m}^3 \tag{3.3}$$

We will consider $N = 125$ atoms as this is the sort of value we will use in MD simulations. Consequently, the volumes we will consider will be of order cubic nanometers, $10^{-27}\mathrm{m}^3$. Of course, the van der Waals formula is not restricted to such tiny length scales but this will give an idea what to expect in MD. The van der Waals free energy is

$$F = -Nk_BT\left[\log\left(n_Q\frac{V-Nb}{N}\right)+1\right] - \frac{N^2a}{V} \tag{3.4}$$

with the *quantum concentration* given as

$$n_Q = (mk_BT/2\pi\hbar^2)^{3/2} \tag{3.5}$$

where m is the atomic mass, which is 40 a.m.u. in argon.[1]. The factor of $\hbar$ is really just a convention since we are not doing quantum mechanical calculations. Replacing $\hbar$ with another value would not alter the phase transition results.[2] Note that this free energy is just the ideal gas free energy with an extra factor

$$\Delta F = \Delta U = -\frac{N^2a}{V} \tag{3.6}$$

which represents the attraction between atoms due to the parameter a and is calculated using a mean field theory.

Figure 3.1 shows the pressure/volume curve from (3.1) for a range of temperatures. The first thing to note is that the pressure becomes negative in certain cases. This initially caused us some consternation. However, it must be remembered that the assumption that lead to (3.1) is that the atoms stay as a single phase. In reality, the system will split into two phases: a gas and liquid phase which coexist. As we shall see, this always results in the system having a positive pressure. The process of splitting into two phases is not accounted for in (3.1) and must be added separately which is precisely the theory we now develop. In exactly the same way, for just the same reason, the MD results will also give negative pressures and we must apply the same theory to these calculations also.

Also the pressure diverges if $V < Nb$ in accordance to (3.1) which accords with the assumption within (3.1) that the atoms are hard spheres each with volume b.[3] As shown in (3.1) we can perform the *Maxwell construction* to give coexistence lines. A flat line is drawn between the "up and down" portion of the graph (shown here at 130 K). The area above and below the flat line must be equal, as illustrated schematically in Figure 1.1.

While useful conceptually, the Maxwell construction is not easy to implement automatically using a computer program. To prepare Figure 3.1, the Maxwell construction was done indirectly using Gibbs free energy plots as shown in Figure 3.2. The Gibbs free energy is given for the van der Waals gas as

$$G = F + pV = \frac{Nk_BTV}{V-Nb} - \frac{2N^2a}{V} - Nk_BT\{\log[n_Q(V-Nb)/N]+1\} \tag{3.7}$$

where we have used (3.4) and (3.1). For each T and p, there are 1 or 3 solutions for V, from (3.1). Substituting these volumes in (3.7) gives at most 3 different Gibbs free energies for each T, p. Taking

[1] The atomic mass unit is 1.66×10^{-27} kg

[2] We have checked that TOWHEE uses this convention also, in chemical potential calculations

[3] The pressure does regain finite values between $V = Nb$ and $V = 0$ but such results are quite meaningless physically

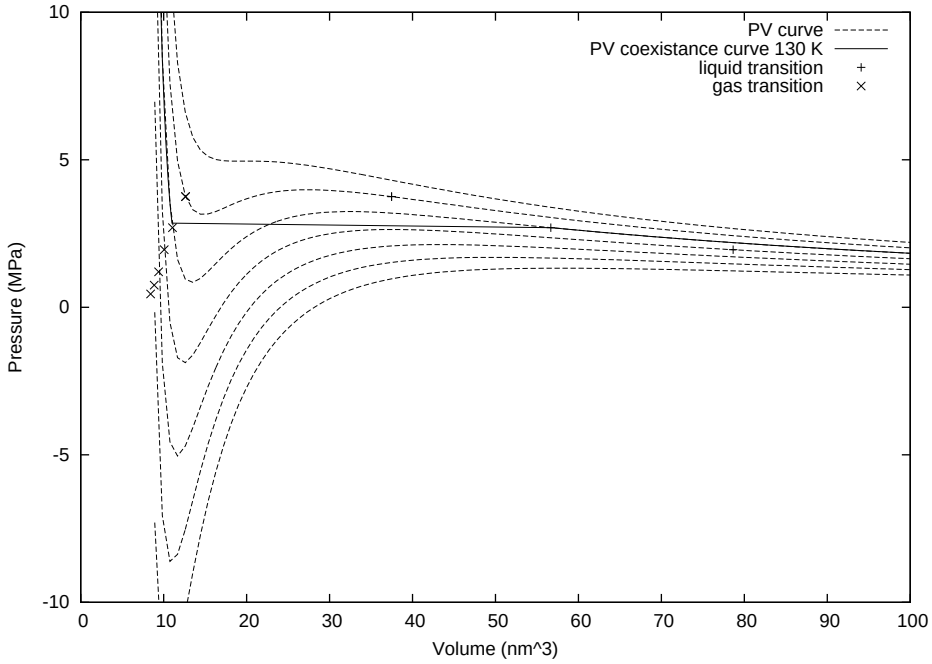


Figure 3.1: Isotherms and underlying phase coexistence curve (symbols) for van der Walls gas

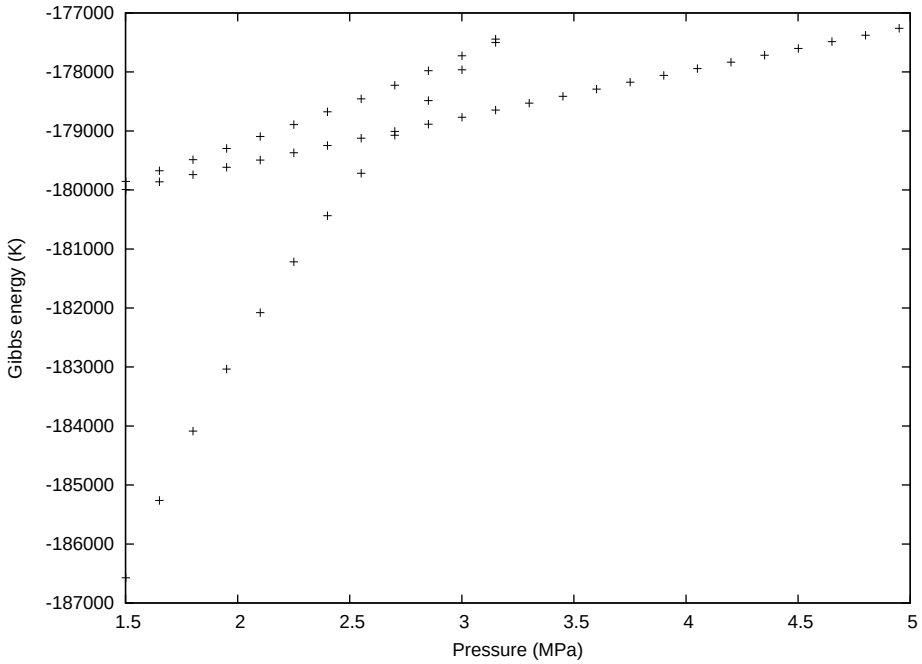


Figure 3.2: Gibbs free energy at 130 K. In general there are 3 values of G for each pressure and temperature.

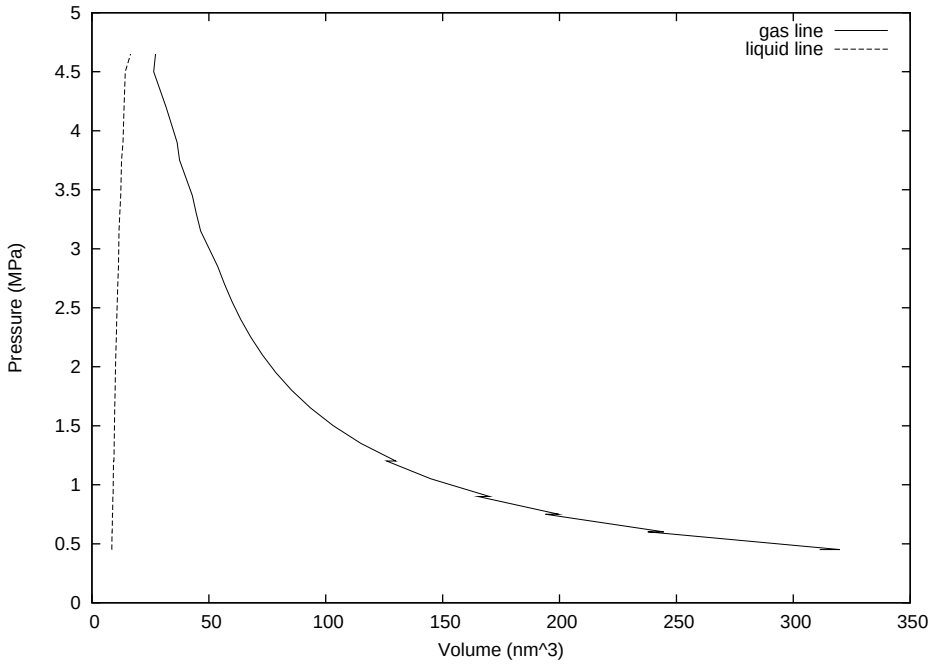


Figure 3.3: Argon p-V phase diagram

the lowest Gibbs energy point gives the "flat" phase coexistence $p(V)$ curve[4] in Figure 3.1. From Figure 3.2, the top Gibbs energy branch can be discarded at each volume since it is always an unstable point. When the other two points converge we obtain the phase transition pressure and the gas and liquid phase transition volumes (shown as symbols in Figure 3.1). For example, with $T = 130$ K, Figure 3.2, the crossover occurs about 2.7 MPa so that is the transition pressure, see Figure 3.1. This technique is used in the MD "driver" programs.

Figure 3.3 shows the pressure-volume phase transition diagram obtained from the symbols in Figure 3.1. Figure 3.4 shows a similar phase diagram in T-V space. Note, these graphs do not show the critical point itself as it becomes hard for the computer to find the transition points when they are close together. Note that the curves become rather jagged due to round off errors. These errors are expected to be more severe in the actual MD calculations.

Figure 3.1 shows how the van der Waals curves can show extreme swings from positive to negative values of pressure. This is particularly acute at low temperatures far from the critical point. *It is therefore difficult to find phase transitions at $T \ll T_c$* which is a problem we encountered in initial simulation runs at very low temperatures. Hence, most of the results shown in Chapter 4 and Chapter 5 are near to the critical point.

3.2 More detailed calculation, single phase case

In the last section we found the phase transitions by minimising Gibbs free energy in a rather ad-hoc way. In this section we return to the fundamental principle of entropy maximisation to prove the same

[4]We will often refer to the "flat" $p(V)$ or $\mu(N)$ curve in this report. This means the monotonic $p(V)$ or $\mu(N)$ curve that would be obtain be appling global entropy maximisation allowing the system under study to reach equilibrium with a reservoir. These curves generally have a flat part in which volume (number of particles) varies but pressure (chemical potential) stays constant. This is in contrast to the non-monotonic "up and down" curves obtained directly from MD simulations. To go from the non-monotonic to the flat curve requires an additional calculation beyond the MD result: this calculation is described in this chapter

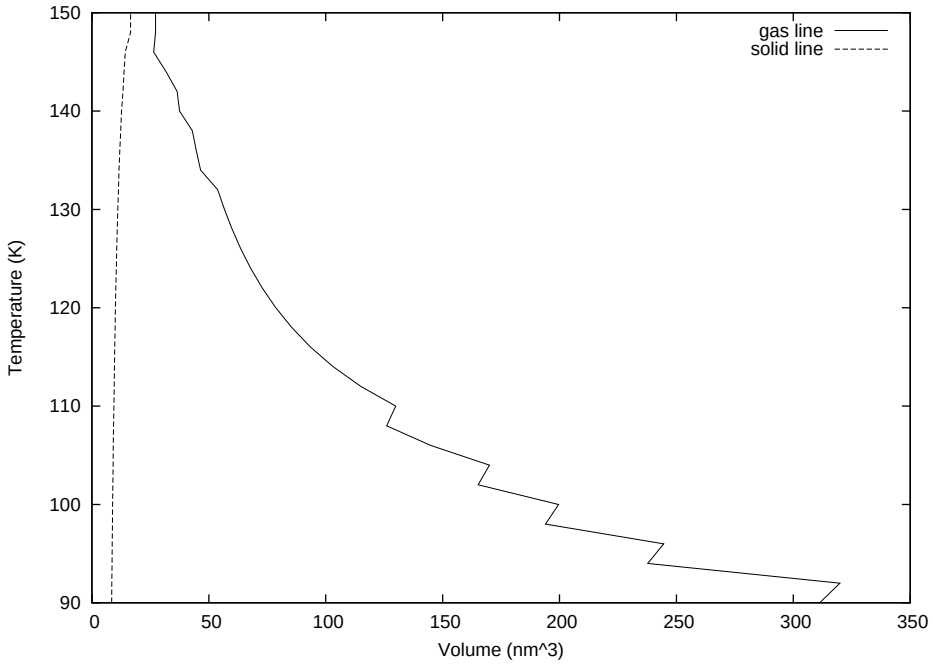


Figure 3.4: Argon T-V phase diagram

result and show some more characteristic "double dip" free energy curves. This will allow us to calculate precisely which functions need to be minimised in the more complicated 2-box and 2-phase systems we will encounter later in later parts of this chapter.

The fundamental principle is that the total entropy of a closed system will be maximised with respect to the energy, volume and number of particles (of each type when several species are considered). Let us consider two subsystems 1 and 2 and assume they can exchange energy and volume such that the total energy and volume is conserved (there is no particle exchange in this calculation). We can express by saying that

$$\Phi = S_1(U_1, V_1, N_1) + S_2(U_2, V_2, N_2) + \lambda_U(U_1 + U_2 - U_t) + \lambda_V(V_1 + V_2 - V_t) \tag{3.8}$$

should be maximised with respect to $U_1, V_1, U_2, V_2, \lambda_U, \lambda_V$. Here U_t and V_t are the total energy and volume and the λ coefficients are Lagrange multipliers. The Lagrange multipliers allow us to maximise without explicit constraint on the energies and volumes.

Let us say that subsystem 1 is the Van der Waals gas of interest and subsystem 2 is a generic heat bath. We do not know the entropy function for the heat bath but we can measure the temperature and pressure at equilibrium: T_0, p_0 (the zero subscript is to emphasise that these are constant values not related to the subsystem energies, volumes). From the relation

$$\mathrm{d}U = T\mathrm{d}S - p\mathrm{d}V + \mu\mathrm{d}N \tag{3.9}$$

we see that $\lambda_U = -1/T_0$ and $\lambda_V = -p_0/T_0$. Since we "know" what these are, the multipliers are just constants in the maximisation. Dropping the constant terms U_t, V_t, which have no effect on maximisation, we find

$$\Phi = S_1(U_1, V_1, N_1) - \frac{1}{T_0}U_1 - \frac{p_0}{T_0}V_1 + S_2(U_2, V_2, N_2) - \frac{1}{T_0}U_2 - \frac{p_0}{T_0}V_2 \tag{3.10}$$

The function is a sum of two functions in different variables. Thus, each function needs to be maximised separately. For system 1, we maximise

$$\Phi_1 = S_1(U_1, V_1, N_1) - \frac{1}{T_0}U_1 - \frac{p_0}{T_0}V_1 \tag{3.11}$$

Since we are only interested in subsystem 1, we drop the subscript from now on:

$$\Phi = S(U,V,T) - \frac{1}{T_0}U - \frac{p_0}{T_0}V \tag{3.12}$$

Multiplying by $-T_0$ shows that this corresponds to minimising

$$G = -T_0 S(U,V,N) + U + p_0 V \tag{3.13}$$

which is the Gibbs energy. Note that the Gibbs energy must be minimised with respect to U, V. This minimisation can be done graphically or by using a numerical minimiser on $G(U,V)$. Or, we can perform the minimisation algebraically

$$\left(\frac{\partial G}{\partial U} = 0,\ \frac{\partial G}{\partial V} = 0\right) \Rightarrow \left(\frac{\partial S}{\partial U} = \frac{1}{T_0},\ \frac{\partial S}{\partial V} = \frac{p_0}{T_0}\right) \tag{3.14}$$

this will give two equations in two unknowns from which we can determine the minimum (note: these equations simply amount to setting the temperature and pressure equal to T_0, p_0). But there will, in general, be multiple solutions corresponding to several minima in (3.13). To see which minimum is deepest, we need to plug solutions back into (3.13) and check which is lowest. This is essentially what was happening in Section 1, Figure 3.2.

A third approach, which gives good insight, is to minimise with respect to U

$$\frac{\partial G}{\partial U} = 0 \Rightarrow \frac{\partial S}{\partial U} = \frac{1}{T_0} \tag{3.15}$$

this gives a trajectory $U(V)$. We can then plug this result into (3.13) to get a function $G'(V)$ which we can graphically minimise over just V. In doing this we are exploring the function over the trajectory $U(V)$ which is bound to contain the minimum(s). This approach may not work in every case however: it will work when $\partial G/\partial U$ yields a single valued function $U(V)$.

For the vdW gas we have the fundamental law

$$S = kN\left\{\log\left[\left(\frac{M}{2\pi\hbar^2}\right)^{3/2}\left(\frac{2U}{3N} + \frac{2Na}{3V}\right)^{3/2}\frac{V - Nb}{N}\right] + \frac{5}{2}\right\} \tag{3.16}$$

From (3.15) we obtain

$$U = \frac{3}{2}NkT_0 - \frac{N^2 a}{V} \tag{3.17}$$

which is just the energy of the vdW gas. Substituting in (3.13) with (3.16) gives

$$G'(V) = -NkT_0\left[\log\left(n_Q\frac{V - Nb}{N}\right) + 1\right] - \frac{N^2 a}{V} + p_0 V \tag{3.18}$$

with $n_Q = (MkT_0/2\pi\hbar^2)^{3/2}$.

Shortcut: A simpler way to get (3.18) is to write the free energy in terms of temperature and number of particles. Then add $p_0 V$ to get the Gibbs free energy:

$$G(V) = F(V) + p_0 V \tag{3.19}$$

which gives (3.18) immediately.

Figure 3.5 shows some plots of $G'(V)$ for 130 K. At low pressure the deepest minimum is at high volumes (gas). At high pressure the deepest minimum is at low volume (liquid). The phase transition occurs around $p = 2.7$ MPa when both minima are the same depth. This is just the vapour pressure predicted in Figure 3.1. This is a classic "double dip" graph one would expect in a first order phase transition.

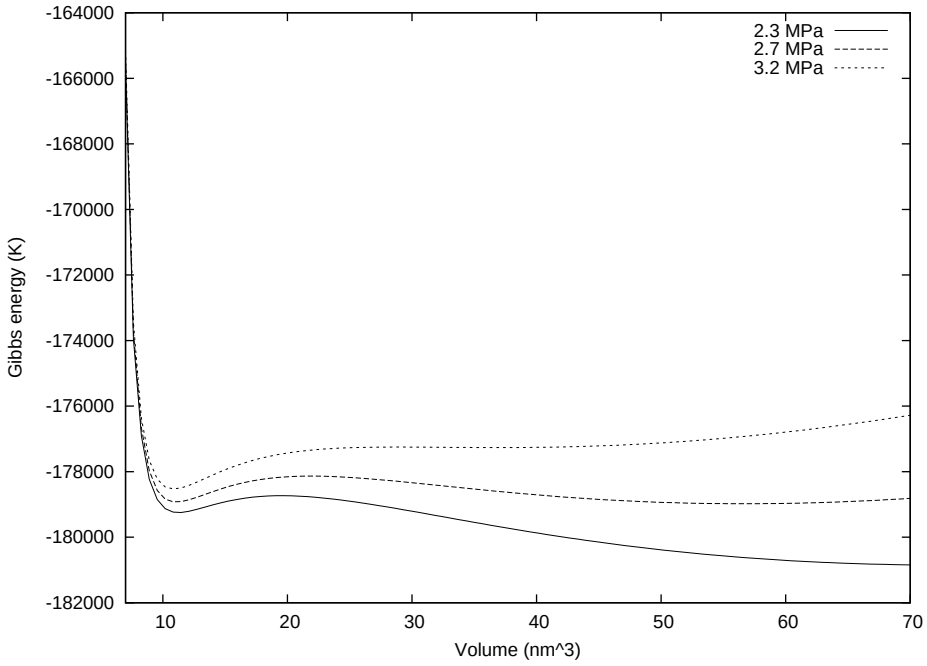


Figure 3.5: Gibbs energy vs. volume for 130 K and a range of pressures

3.3 Two phase case

Previously we have fixed the pressure, temperature and number of atoms. This always leads to a single phase situation: if the pressure is fixed just above the vapour pressure, we get a single liquid phase; when it is just below we get a single gas phase. If we instead fix the total volume V, number of atoms N and temperature, we will obtain a two phase system over some range of volumes as can be seen from Figure 3.1. We now attempt to solve such a system for given T, V, N. We will discover how many atoms are in each coexisting phase and what volume each phase occupies.

The function to be maximised is now

$$\Phi = S_1(U_1, V_1, N_1) + S_2(U_2, V_2, N_2) - \frac{1}{T_0}(U_1 + U_2) + \lambda_V(V_1 + V_2 - V) + \lambda_N(N_1 + N_2 - N) \tag{3.20}$$

Note that the energy multiplier has been identified as T_0 from the outset. Maximising this with respect to energy, volume and number of particles gives

$$\frac{\partial S_1}{\partial U_1} = \frac{1}{T_0} \tag{3.21}$$

$$\frac{\partial S_2}{\partial U_2} = \frac{1}{T_0} \tag{3.22}$$

$$\frac{\partial S_1}{\partial V_1} = \frac{\partial S_2}{\partial V_2} \tag{3.23}$$

$$\frac{\partial S_1}{\partial N_1} = \frac{\partial S_2}{\partial N_2} \tag{3.24}$$

$$V_1 + V_2 = V \tag{3.25}$$

$$N_1 + N_2 = N \tag{3.26}$$

which is 6 equations in 6 unknowns. (3.21) gives

$$k\left(\frac{2U_1}{3N_1}+\frac{2N_1a}{3V_1}\right)^{-1}=\frac{1}{T_0} \tag{3.27}$$

so that

$$U_1=\frac{3}{2}N_1kT_0-\frac{N_1^2a}{V_1} \tag{3.28}$$

and, from (3.22)

$$U_2=\frac{3}{2}N_2kT_0-\frac{N_2^2a}{V_2} \tag{3.29}$$

From (3.16), we obtain

$$\frac{\partial S}{\partial V}=\frac{kN^2a}{V^2}\left(\frac{2U}{3N}+\frac{2Na}{3V}\right)^{-1}+\frac{kN}{V-Nb} \tag{3.30}$$

From (3.28), this simplifies to

$$\frac{\partial S}{\partial V}=-\frac{N^2a}{T_0V^2}+\frac{kN}{V-Nb} \tag{3.31}$$

Thus, (3.23) becomes

$$-\frac{N_1^2a}{T_0V_1^2}+\frac{kN_1}{V_1-N_1b}=-\frac{N_2^2a}{T_0V_2^2}+\frac{kN_2}{V_2-N_2b} \tag{3.32}$$

which just amounts to the pressure being the same in the two phases. From (3.16) we obtain

$$\frac{\partial S}{\partial N} = k\log\left[\left(\frac{M}{2\pi\hbar^2}\right)^{3/2}\left(\frac{2U}{3N}+\frac{2Na}{3V}\right)^{3/2}\frac{V-Nb}{N}\right] \tag{3.33}$$

$$+ kN\left[\frac{-U/N^2+a/V}{2U/3N+2Na/3V}-\frac{V}{N(V-Nb)}\right]+\frac{5}{2}k \tag{3.34}$$

which from (3.28) becomes

$$\frac{\partial S}{\partial N}=k\log\left[n_Q\frac{V-Nb}{N}\right]+k+\frac{2aN}{VT_0}-\frac{kV}{V-Nb} \tag{3.35}$$

Therefore, (3.24) becomes

$$\begin{aligned}&k\log\left[n_Q\frac{V_1-N_1b}{N_1}\right]+\frac{2aN_1}{V_1T_0}-\frac{kV_1}{V_1-N_1b}\\ &=k\log\left[n_Q\frac{V_2-N_2b}{N_2}\right]+\frac{2aN_2}{V_2T_0}-\frac{kV_2}{V_2-N_2b}\end{aligned} \tag{3.36}$$

which just amounts to chemical potentials being equal.

Eqs (3.25), (3.26), (3.32), (3.36) may now be solved for V_1, V_2, N_1, N_2. Multiple solutions will be obtained and we need to plug into (3.20) to see which is the global extremum. Note that Φ can be simplified, for this purpose, to

$$\Phi=S_1(U_1,V_1,N_1)+S_2(U_2,V_2,N_2)-\frac{1}{T_0}(U_1+U_2) \tag{3.37}$$

since we are insisting on the constraints (3.25), (3.26).

Shortcut: As in the previous section, the energy terms dropped out and we could have eliminated them from the start. Multiplying (3.20) by $-T_0$ gives

$$F=F_1(V_1,N_1)+F_2(V_2,N_2)+\lambda_V(V_1+V_2-V)+\lambda_N(N_1+N_2-N) \tag{3.38}$$

where the Helmholtz free energy $F(V, N; T_0) = -T_0 S + U$ has been expressed in terms of temperature, volume and number of particles. *Thus we are minimising total Helmholtz free energy subject to constraints (3.25), (3.26).* In the last section we minimised Gibbs energy without constraints. The precise thermodynamic potential to extremise (Gibbs, Helmholtz or plain entropy) depends on what conjugate variable we intend to measure (temperature, pressure, chemical potential) and what constraints we have. In this case, we measure the equilibrium temperature but impose a constraint to fix the pressure and chemical potential (we eliminate both of these: they do not appear in our final equations). Thus the Helmholtz, not Gibbs energy is appropriate in the section. Note that in the van der Waals gas case, unlike the ideal gas, energy depends on both volume and number of particles according to (3.28).

3.4 Single phase case with particle exchange

In Section 3.1 we considered a box with a fixed number of particles exchanging volume with a heat bath. Now we consider a fixed volume box exchanging *particles* with the heat bath. For the fixed volume, we will drive the system with a chemical potential which will cause the number of particles to vary. At the phase transition, the number of particles will suddenly jump. This is equivalent to Section 3.1 where we drove volume with pressure and noted a sudden increase in volume. In density terms, *the phase transitions should occur at the same place.*

Using the same argument we discover that the function

$$\phi = S(U, N) - \frac{U}{T_0} + \frac{\mu_0}{T_0} N \tag{3.39}$$

ought to be maximised. We assume that we can simply eliminate the energy in terms of temperature as before. Doing this and multiplying by $-T_0$ shows we should minimise

$$\Psi = F(N) - \mu_0 N \tag{3.40}$$

for the given temperature. We will call Ψ the N-potential. Differentiating gives

$$\left(\frac{\partial F}{\partial N}\right)_{V,T_0} = \mu_0 \tag{3.41}$$

which is simply the statement that the chemical potential should be equal to its equilibrium value. The vdW free energy is

$$F = -N k_B T \left[\log\left(n_Q \frac{V - Nb}{N}\right) + 1\right] - \frac{N^2 a}{V} \tag{3.42}$$

so that

$$\mu = -k_B T \log\left(n_Q \frac{V - Nb}{N}\right) + \frac{k_B T V}{V - Nb} - k_B T - \frac{2Na}{V} \tag{3.43}$$

so (3.41) is a non-linear equation with a maximum of 3 solutions for N. We must plug each solution back into (3.40) to see which N-potential is lowest.

Figure 3.6 shows the chemical potential vs N plot for 90, 100 K. The underlying curve is shown dashed. For each chemical potential, there are at most 3 solutions for N. In the solid curves the lowest Ψ solution has been chosen indicating a phase transition. Note that the chemical potential varies from $\mu = -\infty$ at $N = 0$ to $\mu = +\infty$ when $N \to V/b$. (ie system is squashed beyond the minimum volume per particle b). The transition chemical potentials all appear to be negative.

Figure 3.7 shows a series of such plots with the lowest Ψ points taken, over a temperature range. The transition number of particles is shown with symbols indicating a phase diagram. This is shown again in Figure 3.8. Note that the phase diagram is rather ragged owing to poor μ sampling. The chemical potential plot is harder to sample than the pressure plots of Figure 3.1, especially at the low-N range where the chemical potential remains flat down to low N values before rapidly plunging to $-\infty$. At

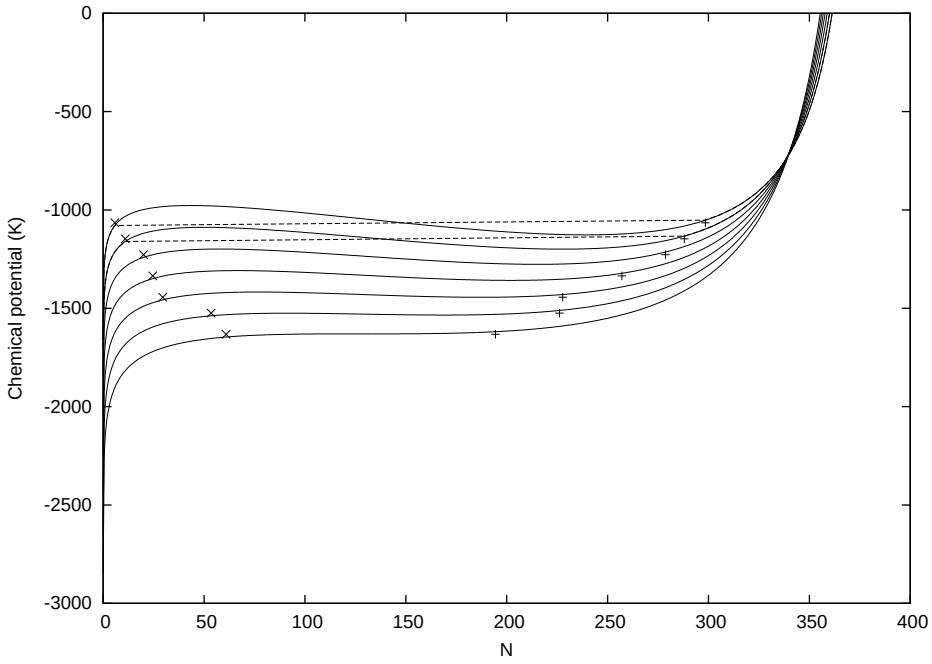


Figure 3.6: Chem. potential vs number of particles. The dashed curve shows the minimum N-potential solution for 90 and 100 K. $V = 20\text{nm}^3$

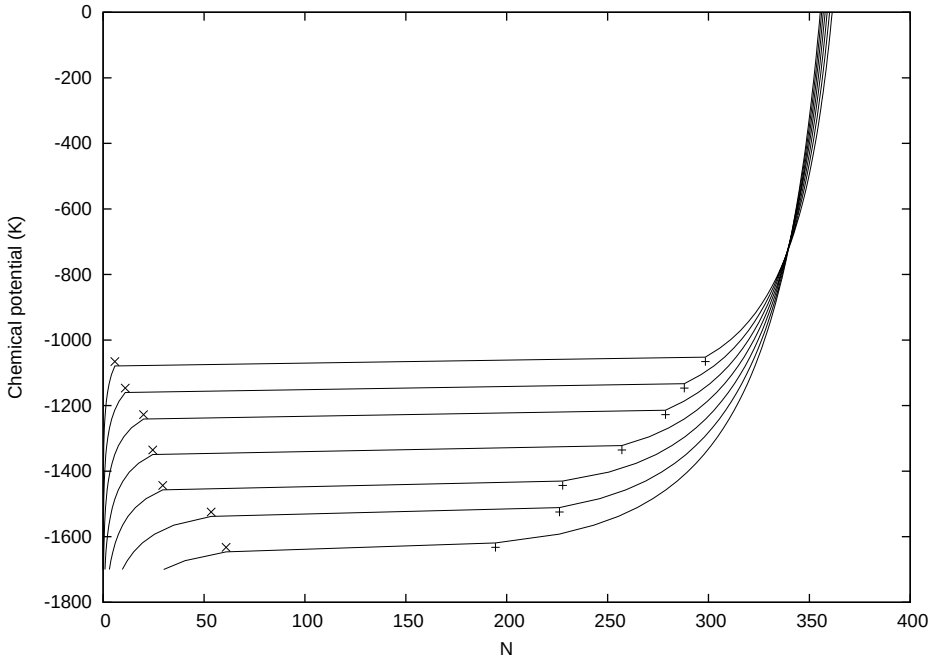


Figure 3.7: Chem. potential vs number of particles plots for temperaturs 90 K (top curve) to 150 K (bottom curve). Transition N indicated by symbols.

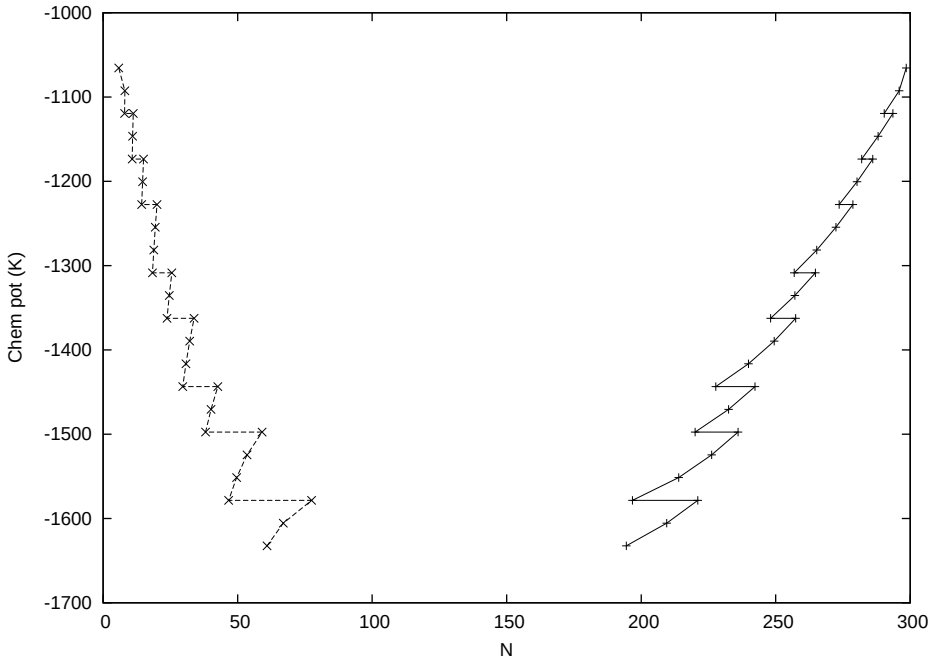


Figure 3.8: $\mu - N$ phase diagram.

low temperatures, this final divergence of μ does not occur until $N < 1$ which is fine for our analytical function (3.43) but not possible in MD code, where N must be a whole number.

Finally, Figure 3.9 shows the $T - \rho$ phase diagram where we have converted to density $n = N/V$. This is compared with Figure 3.4 which has also been converted to density on the x-axis. The two phase diagrams are the same (apart from numerical errors) so that the phase diagram can be generated either by varying volume or number of particles. However, it seems the chemical potential is more difficult to work with numerically and produces greater error. The figure also shows experimental data. Although we have fitted the a and b parameters, the fit is not very good and the theory systematically underestimates the liquid densities.

3.5 Fitting to experimental data

As mentioned in Chapter 1, inverse modelling, the process whereby theoretical parameters are adjusted to give a good fit to experimental data, is an important part of this project. Here, we consider adjusting the a and b parameters to fit experimental data. In Chapter 4, we will see whether the MD gives a better fit to experiment than simple functions like the van der Waal gas. If it does not, then there is little point using MD, given its much greater complexity and calculation time.

Figure 3.10 shows a basic comparison between theory and experiment isotherms with parameters (3.3). The experimental data was initially presented on a density scale and we have kept this form. The experimental data shows "flat" lines with pressure taking on only positive values. Figure 3.11 shows the theoretical curves converted into flat line graphs and these are compared to experiment. The experimental density rises to as much as 1400 kg/m^3 while the theory goes only to 1000 kg/m^3 explaining the anomalous results of Figure 3.9. Of course, the phase transition temperature agrees since this is how we decided a and b. However, it might be preferable to fit the theory more broadly to experiment so that the liquid densities are also well represented. In Figure 3.12 these parameters have been fitted to experiment according to the Levenberg-Marquardt procedure [9] for least squares fitting. For each available experimental pressure, the theoretical value is established and the parameters varied so as to

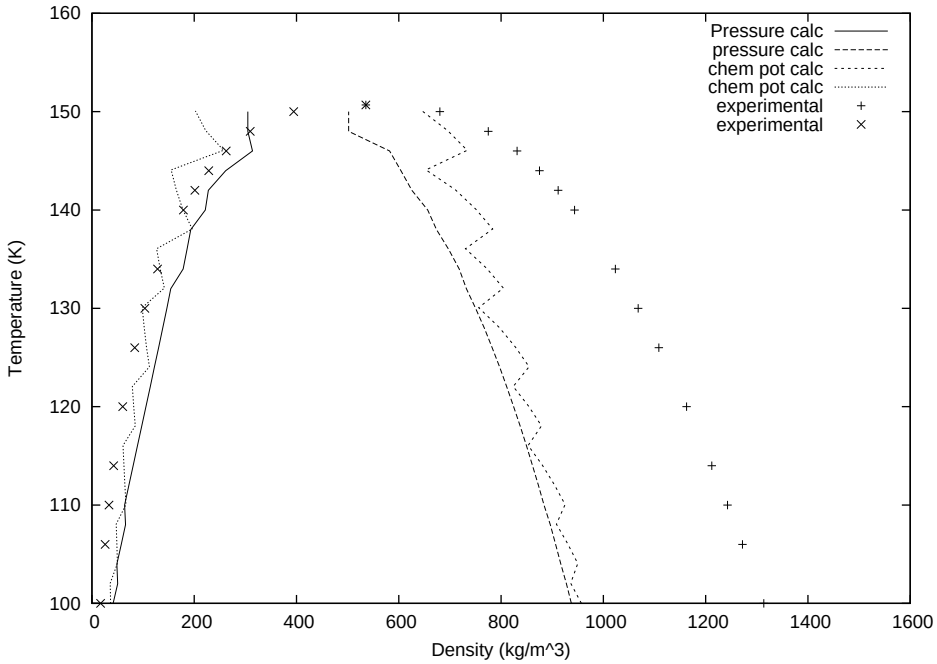


Figure 3.9: $T - \rho$ phase diagram calculated using both pressure (from Figure 3.4) and chemical potential formulations. Experimental data also shown

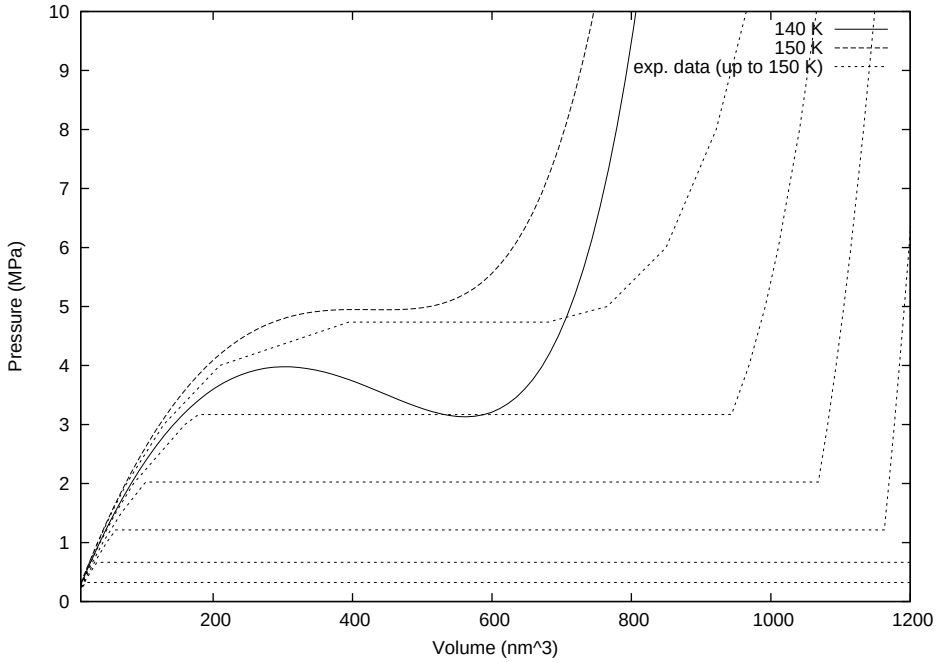


Figure 3.10: Van der Waals isotherms for argon compared to experimental data.

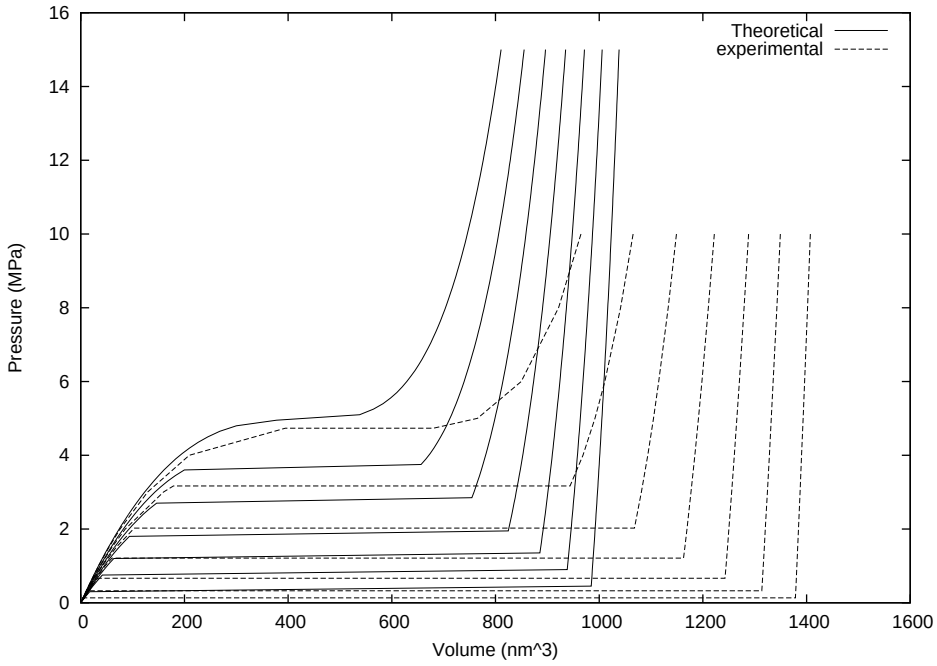


Figure 3.11: Flat van der Waals curves compared to experiment

minimise the sum-square difference between them. Note that there are no experimental points along the "flat" portion of the curve so the fact that the theory gives a non-monotonic curve, while the experimental curves are monotonic, does not cause difficulty. Although it represents a minimum least squares solution the fit is not very good in Figure 3.12: the 150K theoretical curve lies along the 140K experimental curve for example. This indicates that the van der Waal function cannot reproduce experimental results with great accuracy, something we would expect given its simplicity. The best fit values were:

$$a = 2.71 \times 10^{-49}\mathrm{Jm}^3, \quad b = 3.77 \times 10^{-29}\mathrm{m}^3 \tag{3.44}$$

which gives $p_c = 7.06$ MPa and $T_c = 154$ K. The resulting temperature-density phase diagram is shown in Figure 3.13

3.6 Binary system, single phase

We now consider the case where there are multiple species in a box. Let us work from the point of view of constant volume and changing number of particles. On the face of it, introducing another species increases the number of variables in the system. We can now vary the concentration (ratio of number of atoms of each type) and the total number of particles (and hence the overall number density). However, the tacit assumption made in all literature on this subject is that the *total number of particles is fixed* (and volume is fixed so overall density is fixed). Thus, we have a rather different paradigm than before: we have a *binary alloy model* in which the total number of sites is fixed and all that happens is particles of type A can substitute for particles of type B on each lattice site. Each site must always contain an atom of one or other type.

Let us say the number of A type particles is N_A and the number of B type particles is N_B. Thus

$$N = N_A + N_B = \text{const} \tag{3.45}$$

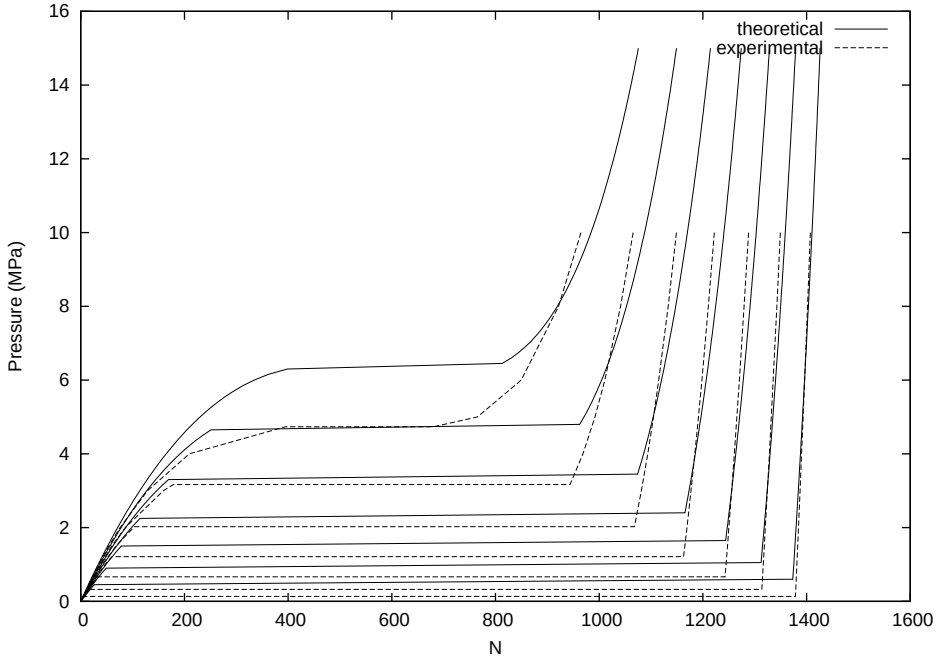


Figure 3.12: Van der Waals and experimental curves for best fit values of a and b

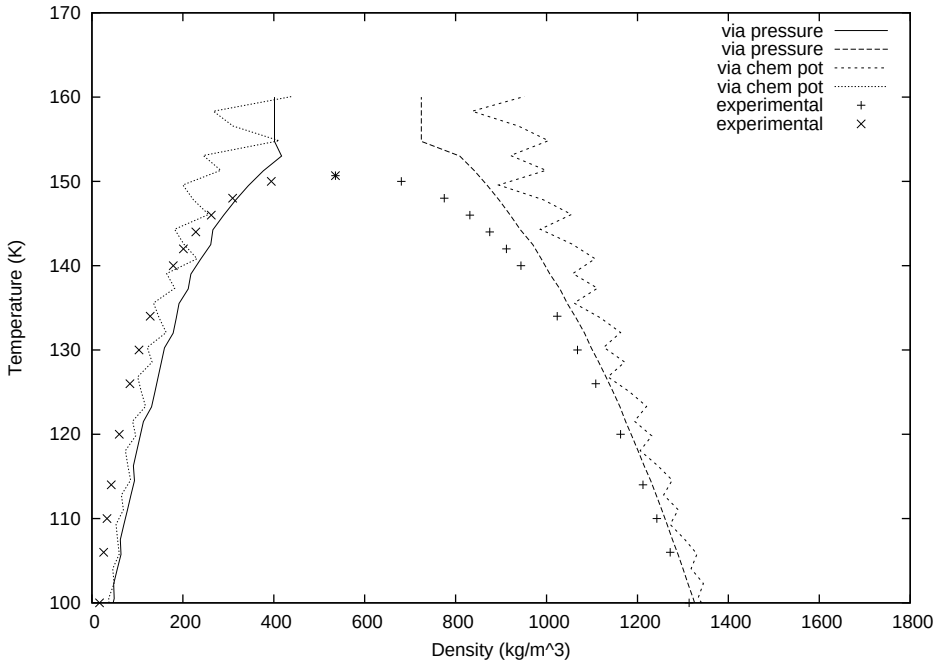


Figure 3.13: $T - \rho$ phase diagram for best fit values of a and b, calculated using pressure and chemical potential and compared to experiment

We can introduce the concentration $x = N_B/N$ so that

$$N_A = (1-x)N; \quad N_B = xN \tag{3.46}$$

In "binary alloy" theory graphs are generally plotted in terms of x which takes the place of the density in phase diagrams such as Figure 3.1.

The fundamental principle of thermodynamics is that entropy should be maximised with respect to energy and number of particles of each species (that is N_A and N_B). Concentrations x do not appear in the theory directly: we cannot, say, minimise free energy with respect to x. We must first work with N_A and N_B to finally derive equations in terms of x. This will also tell us how the outputs of molecular modelling calculations can be used to predict phase diagrams.

Consider 2 systems in equilibrium which can exchange energy and number of particles of each species. We add the *further constraint* that the total number of particles in the system of interest is constant. Thus, we maximise

$$\Phi = S(U, N_A, N_B) + S_0(U_0, N_{A0}, N_{B0}) + \lambda_U(U + U_0 - U_t) \tag{3.47}$$

$$+ \lambda_A(N_A + N_{A0} - N_{At}) + \lambda_B(N_A + N_{B0} - N_{Bt}) + \lambda(N_A + N_B - N) \tag{3.48}$$

Maximising with respect to each of $(U, U_0, N_A, N_B, N_{A0}, N_{B0})$ gives

$$\frac{\partial S}{\partial U} + \lambda_U = 0 \tag{3.49}$$

$$\frac{\partial S_0}{\partial U_0} + \lambda_U = 0 \tag{3.50}$$

$$\frac{\partial S}{\partial N_A} + \lambda_A + \lambda = 0 \tag{3.51}$$

$$\frac{\partial S_0}{\partial N_{A0}} + \lambda_A = 0 \tag{3.52}$$

$$\frac{\partial S}{\partial N_B} + \lambda_B + \lambda = 0 \tag{3.53}$$

$$\frac{\partial S_0}{\partial N_{B0}} + \lambda_B = 0 \tag{3.54}$$

The first two equations tell us that

$$\partial S/\partial U = \partial S_0/\partial U_0 = 1/T_{eq} = -\lambda_U \tag{3.55}$$

In (3.52) and (3.54) we identify

$$\frac{\partial S_0}{\partial N_{A0}} = -\mu_{A0}/T_0 = -\mu_{A0}/T_{eq} \tag{3.56}$$

$$\frac{\partial S_0}{\partial N_{B0}} = -\mu_{B0}/T_0 = -\mu_{B0}/T_{eq} \tag{3.57}$$

Thus we have $\lambda_A = \mu_{A0}/T_{eq}$ and $\lambda_B = \mu_{B0}/T_{eq}$. Substituting this into (3.51), (3.53) gives

$$\frac{\partial S}{\partial N_A} + \frac{\mu_{A0}}{T_{eq}} + \lambda = 0 \tag{3.58}$$

$$\frac{\partial S}{\partial N_B} + \frac{\mu_{B0}}{T_{eq}} + \lambda = 0 \tag{3.59}$$

Combining these equations gives

$$\frac{\partial S}{\partial N_B} - \frac{\partial S}{\partial N_A} = \mu_{A0}/T_{eq} - \mu_{B0}/T_{eq} \tag{3.60}$$

which implies

$$\mu_B - \mu_A = \mu_{B0} - \mu_{A0} \tag{3.61}$$

Thus, the *difference* in chemical potential is the same for the system as for the reservoir. The individual chemical potentials are *not* equal at equilibrium, in contrast to the result of Section 3.4. This comes about because of the extra constraint (3.45).

As usual, we can write a minimisation scheme for just the system of interest. Separating (3.48), we have

$$\psi = S - U/T_{eq} + \frac{\mu_{A0} N_A}{T_{eq}} + \frac{\mu_{B0} N_B}{T_{eq}} \tag{3.62}$$

which we must minimise subject to the constraint $N_A + N_B = N$.

Multiplying this by $-T_{eq}$ gives the following function to be minimised:

$$\psi = F(N_A, N_B) - \mu_{A0} N_A - \mu_{B0} N_B \tag{3.63}$$

with the constraint $N_A + N_B = N$. To check this is right, consider minimising without using the Lagrange multiplier. We have

$$\frac{\mathrm{d}\psi}{\mathrm{d}N_A} = \frac{\partial \psi}{\partial N_A} + \frac{\partial \psi}{\partial N_B}\frac{\mathrm{d}N_B}{\mathrm{d}N_A} = 0 \tag{3.64}$$

Thus

$$\frac{\partial F}{\partial N_A} - \mu_{A0} = \frac{\partial F}{\partial N_B} - \mu_{B0} = 0 \tag{3.65}$$

Since $\partial F/\partial N = \mu$ this gives

$$\mu_B - \mu_A = \mu_{B0} - \mu_{A0} \tag{3.66}$$

Which is the same answer.

3.7 Alloys/mixtures with more than 2 species

Consider a trinary alloy with species A, B and C. Using Lagrange multiplier, we minimise

$$\psi = F(N_A, N_B, N_C) - \mu_{A0} N_A - \mu_{B0} N_B - \mu_{C0} N_C + \lambda(N_A + N_B + N_C - N) \tag{3.67}$$

This gives equilibrium equations

$$\frac{\partial \psi'}{\partial N_A} = \mu_A - \mu_{A0} + \lambda = 0 \tag{3.68}$$

$$\frac{\partial \psi'}{\partial N_B} = \mu_B - \mu_{B0} + \lambda = 0 \tag{3.69}$$

$$\frac{\partial \psi'}{\partial N_C} = \mu_C - \mu_{C0} + \lambda = 0 \tag{3.70}$$

Eliminating λ gives

$$\mu_A - \mu_{A0} = \mu_B - \mu_{B0} = \mu_C - \mu_{C0} \tag{3.71}$$

or

$$\mu_B - \mu_A = \mu_{B0} - \mu_{A0} \tag{3.72}$$

$$\mu_C - \mu_B = \mu_{C0} - \mu_{B0} \tag{3.73}$$

$$\mu_A - \mu_C = \mu_{A0} - \mu_{C0} \tag{3.74}$$

These concepts can clearly be extended to any number of species.

3.8 Minimisation function in terms of x

Sustituting (3.63) with (3.46) we get

$$\psi' = \psi/N = f(x) - (1-x)\mu_{A0} - x\mu_{B0} \tag{3.75}$$

We can minimise this function with respect to x to discover the equilibrium concentration. To prove this consider minimising (3.75) with respect to x. This gives

$$\frac{\mathrm{d}\psi}{\mathrm{d}x} = \frac{\partial F}{\partial N_A}\frac{\partial N_A}{\partial x} + \frac{\partial F}{\partial N_B}\frac{\partial N_B}{\partial x} - \mu_{A0}\frac{\partial N_A}{\partial x} - \mu_{B0}\frac{\partial N_B}{\partial x} = 0 \tag{3.76}$$

and we obtain

$$-N\mu_A + N\mu_B + N\mu_{A0} - N\mu_{B0} = 0 \tag{3.77}$$

so that $\mu_B - \mu_A = \mu_{B0} - \mu_{A0}$ as before. Also,

$$\frac{\mathrm{d}F}{\mathrm{d}x} = \frac{\partial F}{\partial N_A}\frac{\partial N_A}{\partial x} + \frac{\partial F}{\partial N_B}\frac{\partial N_B}{\partial x} = (\mu_B - \mu_A)N \tag{3.78}$$

so that

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{1}{N}\frac{\mathrm{d}F}{\mathrm{d}x} = \mu_B - \mu_A \tag{3.79}$$

In molecular dynamics, we can calculate μ_A and μ_B. Taking the difference between the two and integrating gives $f(x)$. Hence ψ' can be evaluated to see which is the deeper minimum (for a given choice of $\mu_{B0} - \mu_{A0}$). This is precisely the technique adopted in NSIM.

3.8.1 Specific example

The entropy, and hence, free energy, of a binary alloy is easy to determine from a binomial equation. We now calculate the free energy and minimise with respect to N_A and N_B and then with respect to x to show the result is the same.

We have

$$S = k\log\frac{N!}{N_A!N_B!} = k[N\log N - N_A\log N_A - N_B\log N_B] \tag{3.80}$$

using the Stirling approximation. Here, we assume entropy is a function only of number of particles, in general it would depend on the energy and hence, through elimination of energy, on temperature.

We can use a mean field expansion to estimate the energy of the system as [10], chapter 11.

$$U = \frac{1}{2}Np\left[\left(\frac{N_A}{N}\right)^2 u_{AA} + 2\frac{N_B N_A}{N^2}u_{AB} + \left(\frac{N_B}{N}\right)^2 u_{BB}\right] \tag{3.81}$$

where u_{AA} etc are the interaction energies between the species and p is the coordination numnber. Thus (using T for the equilibrium temperature),

$$\begin{aligned} F = U - TS &= \frac{1}{2}Np\left[\left(\frac{N_A}{N}\right)^2 u_{AA} + 2\frac{N_B N_A}{N^2}u_{AB} + \left(\frac{N_B}{N}\right)^2 u_{BB}\right] \\ &- kT[N\log N - N_A\log N_A - N_B\log N_B] \end{aligned} \tag{3.82}$$

We therefore have

$$\mu_A = \frac{\partial F}{\partial N_A} = \frac{Np}{2}\left(\frac{2N_A u_{AA}}{N^2} + \frac{2N_B u_{AB}}{N^2}\right) + kT(\log(N_A) + 1) \tag{3.83}$$

$$\mu_B = \frac{\partial F}{\partial N_B} = \frac{Np}{2}\left(\frac{2N_A u_{AB}}{N^2} + \frac{2N_B u_{BB}}{N^2}\right) + kT(\log(N_B) + 1) \tag{3.84}$$

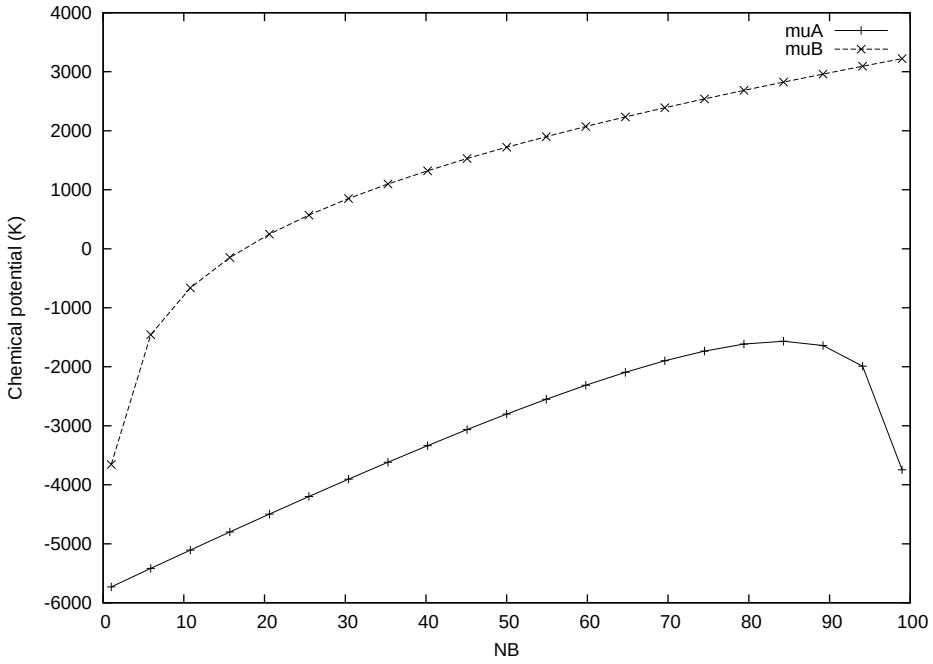


Figure 3.14: Chemical potentials of species A and B vs N_B with $N_A = N - N_B$

For equilibrium, we must solve

$$\mu_B - \mu_A = \mu_{B0} - \mu_{A0} \equiv \Delta\mu \tag{3.85}$$

subject to the constraint $N_A + N_B = N$. There will be several solutions. To find the deepest minimum, we substitute in (3.63) which can be written in the form

$$\psi = F - \mu_{A0}(N - N_B) - \mu_{B0}N_B = F - N_B\Delta\mu + \text{const} \tag{3.86}$$

In terms of x, we obtain

$$f(x) = \frac{p}{2}[(1-x)^2 u_{AA} + 2x(1-x)u_{AB} + x^2 u_{BB}] + kT[x\log x + (1-x)\log(1-x)] \tag{3.87}$$

We must solve the equation

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \Delta\mu \tag{3.88}$$

To find the deepest minimum we should test

$$\psi' = f(x) - (1-x)\mu_{A0} - x\mu_{B0} = f(x) - x(\mu_{B0} - \mu_{A0}) = f(x) - x\Delta\mu \tag{3.89}$$

(ignoring a constant).

In this section, we will consider a copper/lead mixture (A = Cu, B = Pb). This particular example will be simulated using MD in Chapter 7. We set $p = 6$, $N = 100$, $u_{AA} = -0.18$ eV, $u_{BB} = -0.05$ eV, $u_{AB} = -0.07$ eV and $T = 1200$ K (see Chapter 7 for more discussion on these parameters). Note that lowest energy is obtained when AA and BB clusters form, particularly due to the very strong A-A (ie, Cu-Cu) attraction. Figure 3.14 shows the chemical potentials for the 2 species as N_B is varied. When $N_B \to 0$, $\mu \to -\infty$ owing to the log function. Similarly, when N_B goes to 100, N_A vanishes and μ_A diverges.

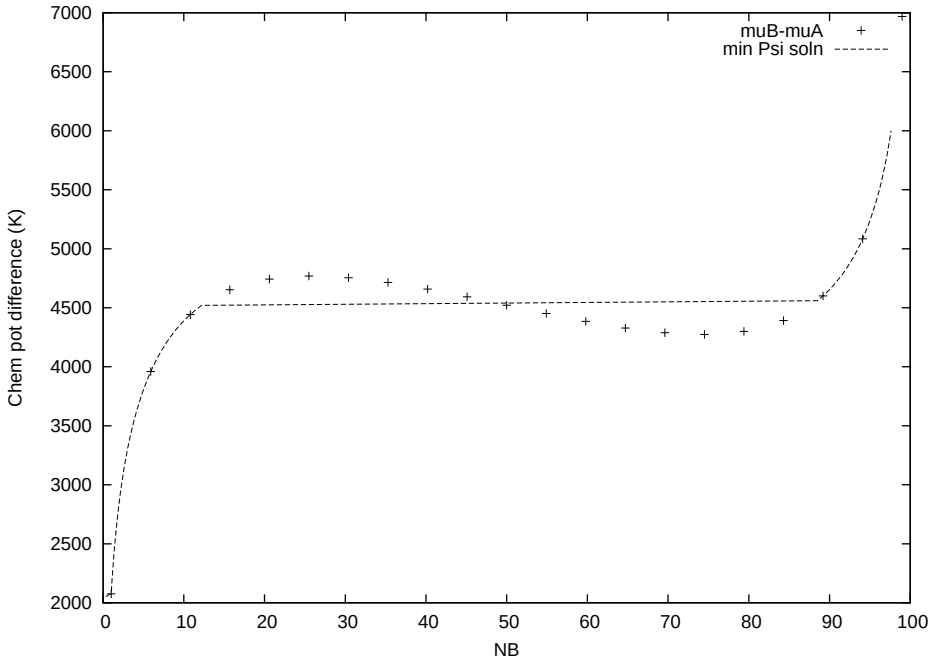


Figure 3.15: Chemical potential difference $\mu_B - \mu_A$ (symbols). Straight line shows minimum ψ solution for given $\Delta\mu$

The difference between the 2 chemical potential curves is given in Figure 3.15 (symbols). This figure is reminiscent of Figure 3.6. For a given $\Delta\mu$ there are several solutions for the number of particles N_B. The solid line shows the minimum ψ solution for each $\Delta\mu$. At $\Delta\mu = 2$, the minimum ψ solution jumps from $N_B \sim 5$ to $N_B \sim 95$ indicating a phase transition.

Figure 3.16 shows the free energy $f(x)$. The form of this curve is discussed in detail in [10], chapter 11. Here we note that the curve points up at $x = 0, 1$ and shows non-monotonic behaviour in between. The analytical curve (3.87) is shown along with a numerical curve obtained by integrating $\mathrm{d}f/\mathrm{d}x$: the two curves differ by a constant which is unimportant. This numerical integration would be necessary for MD calculations.

Figure 3.17 shows a plot of $\mathrm{d}f/\mathrm{d}x$. The minimum ψ solution of (3.85) is shown as a solid line. This line shows a jump at $\Delta\mu = 2$ just as before. Figs 3.15 and 3.17 were calculated quite independently but clearly give identical results.

Another important quantity for binary alloy measurements is the enthalpy of mixing. This is the difference in enthalpy for a mixture compared to having two separate systems of 100% type A and B atoms (the same number of each type of atom must be used in the comparison). In our simple analytical theory, we cannot calculate enthalpy since we have taken no account of pressure or volume. However, in Pb/Cu at 1200 K, enthalpy is known experimentally to be similar to energy which can be calculated from (3.81). Figure 3.18 shows the energy as a function of N_B for our system. The curve is higher than the corresponding straight line between points $N_B = 0$ and $N_B = N$ indicating there is a positive energy of mixing. The difference is shown in Figure 3.19, the maximum energy of mixing is 800 K per atom which equates to about 6700 J/mol which agrees with experimental findings shown in Figure 3.20 and Figure 3.21. The data in these figures was calculated with MTDATA (version 4.81) using thermodynamic data from the SGTE Solution Database (version 4.31). In the simple analytical theory of (3.82), the maximum

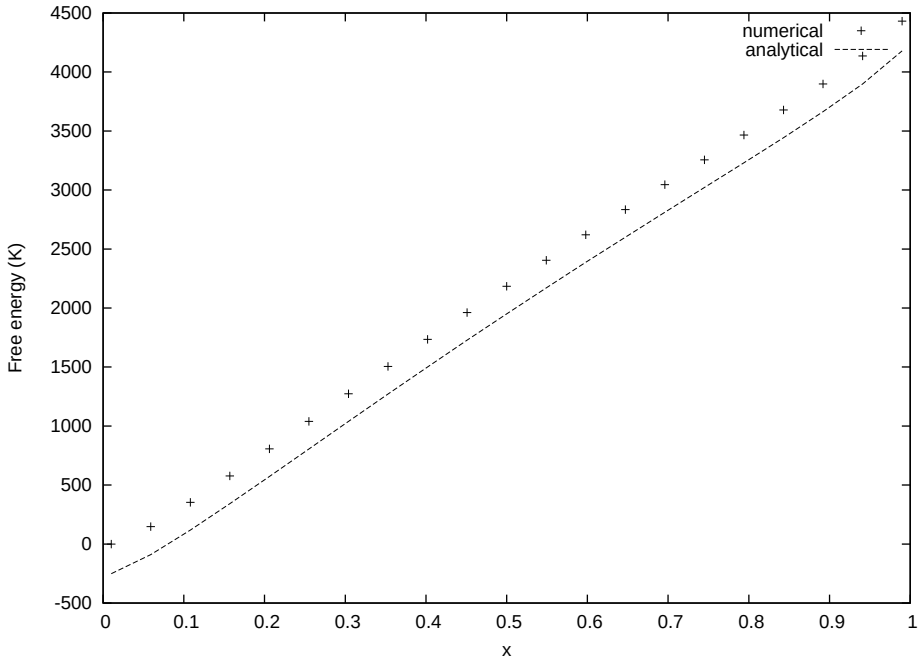


Figure 3.16: Free energy $f(x)$ and function calculated by numerically integrating $\mathrm{d}f/\mathrm{d}x$.

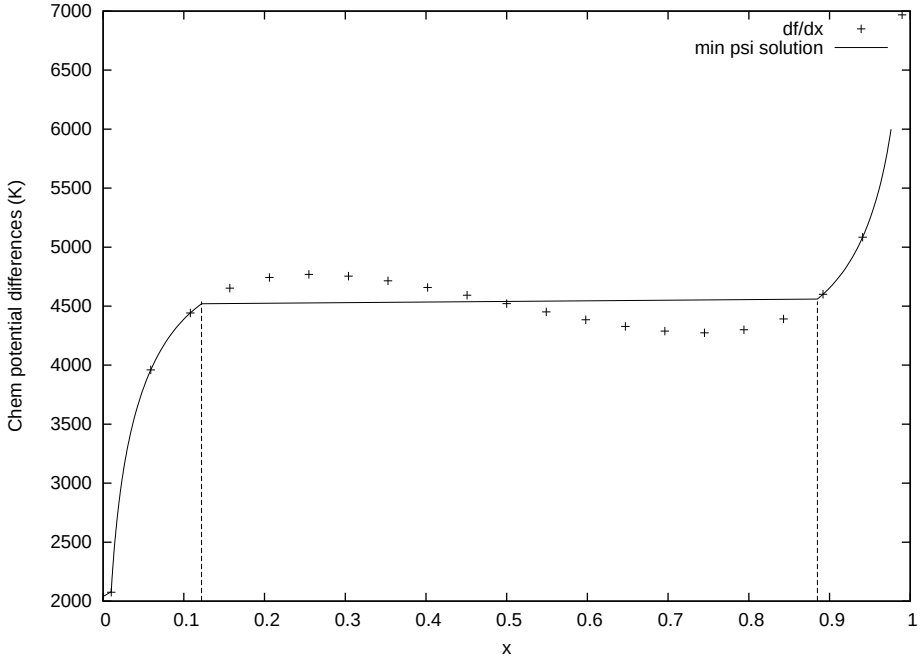


Figure 3.17: Chemical potential difference $\mathrm{d}f/\mathrm{d}x = \mu_B - \mu_A$ (symbols). Straight line shows minimum ψ solution for given $\Delta\mu$

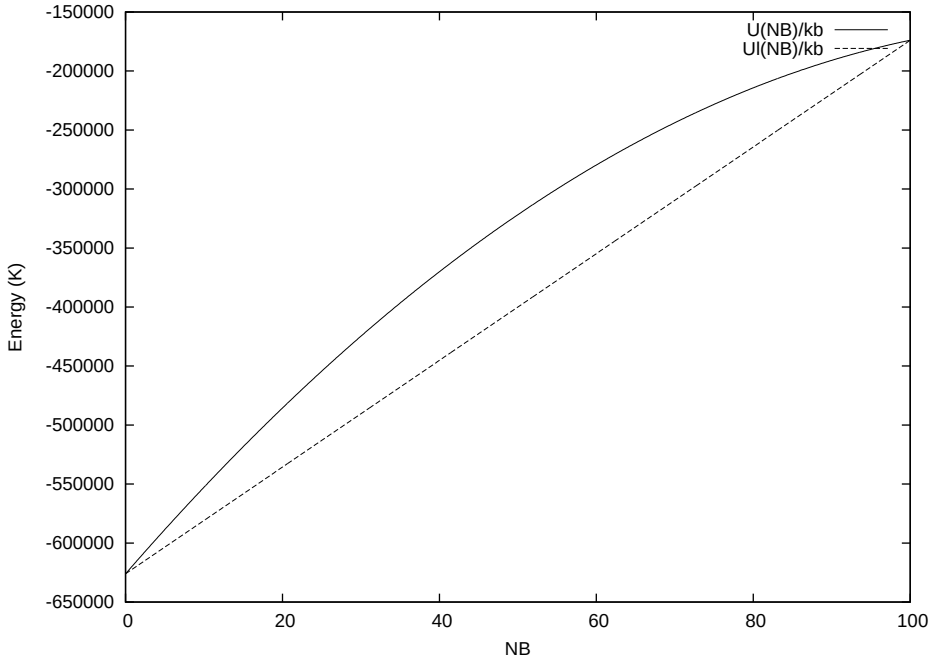


Figure 3.18: Total energy of system compared to energy of separate systems (straight line)

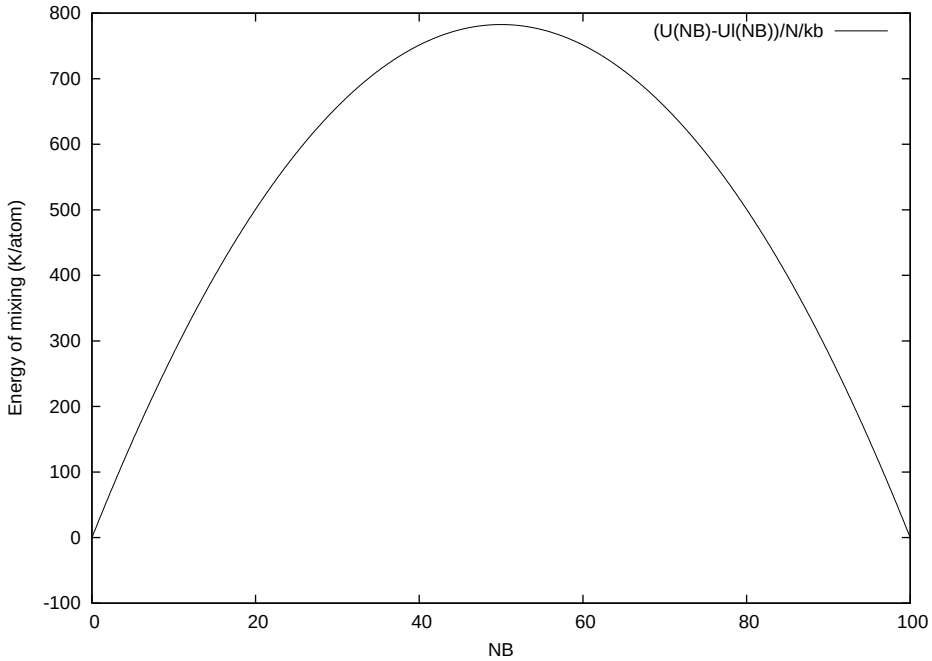


Figure 3.19: Excess energy of mixing

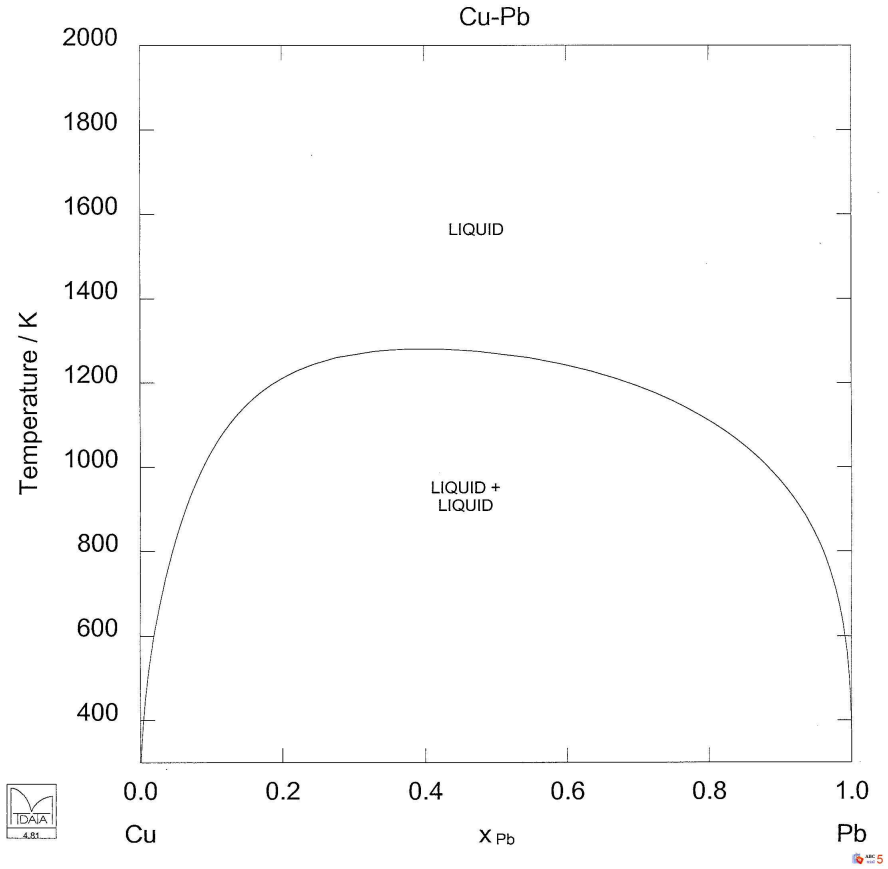


Figure 3.20: Phase diagram for Cu/Pb system as calculated by MTDATA

energy of mixing and the mixing critical temperature are given by

$$\Delta U_{max} = \frac{pN}{4}\left(u_{AB} - \frac{1}{2}(u_{AA} + u_{BB}\right) \tag{3.90}$$

$$T_c = \frac{p}{2k_B}\left(u_{AB} - \frac{1}{2}(u_{AA} + u_{BB})\right) \tag{3.91}$$

These equations can be used to fit suitable values for the interaction energies which is what we will do in Chapter 7.

3.9 Conclusions

In this chapter, we have shown how the pressure-volume or chemical potential-number of particles plots output by MD codes are only the beginning of the calculation needed to produce phase diagrams. To produce phase diagrams requires an additional calculation involving maximisation of the total entropy of the system under study and an external heat reservoir. Several cases can be distinguished: we can have a single box of material under study (so that the pressure is fixed) or two boxes of interacting material (so that the total volume is fixed). We can consider either exchange of volume or number of particles. Either calculation should give the same result, and we have shown that this is the case for a simple van der Waals gas model. In binary alloys, the total number of particles *and* volume is held fixed but the proportion of atoms of each type varies. That is, particles of types A and B are exchanged with the heat

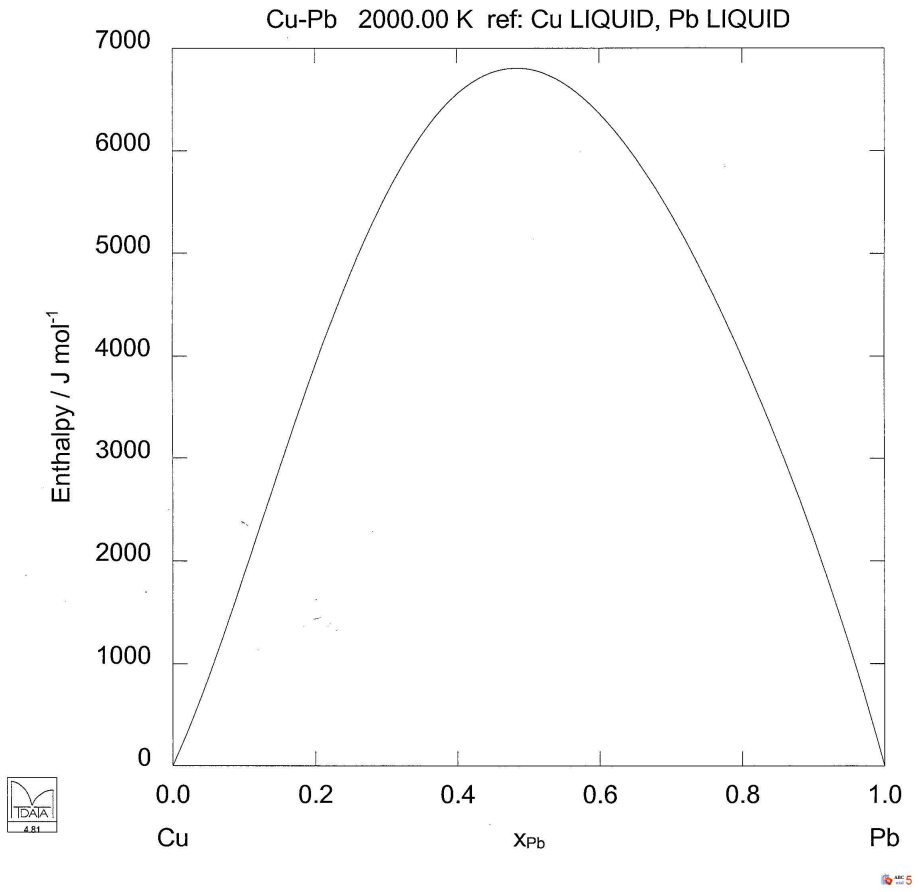


Figure 3.21: Enthalpy of mixing of Cu/Pb system as calculated by MTDATA

reservoir in such a way as to conserve the number of particles of each type in the system we are studying. In each of these cases, the entropy maximisation is brought about mathematically by minimisation of a thermodynamic function. In the case of single-species with volume exchange that function is the well-known Gibbs free energy. In the more complex cases, the nature of the function is not so obvious. In all cases, we have derived the function to minimise from first principles by applying the principle of total entropy maximisation.

In this chapter, we have put flesh on the bones of the fundamental thermodynamic theory by refering to specific analytical formulas for argon and Cu/Pb systems. These formulas are, of course, approximate, but they give very useful insights into the nature of the expected results. We have substituted realistic parameters into the analytical function giving us a good first guess at the order of magnitude pressures, chemical potentials and so on, expected in the more realistic MD calculations of Chapter 4 and Chapter 7.

Chapter 4

Argon

In this chapter, we consider MD calculations on argon gas. Argon is clearly a very simple system and will allow us to test out the capabilities of the various MD/MC codes and also test the operation of the driver code SIMRUNNER. Argon results can also be rescaled onto a fundamental master curve which, in fact, describes all single-atom simulations. By rescaling and comparing results to published findings, we will be able to comment on which of our simulation codes is the most accurate.

4.1 Argon simulation

Argon is a one-atom molecule with no charge. We will characterise the interactions between argon atoms by the Lennard-Jones energy, [3]:

$$U(r) = 4\epsilon \left[(\sigma/r)^{12} - (\sigma/r)^{6} \right] \tag{4.1}$$

Thus, the entire set of forces between atoms is characterised by two numbers: ϵ and σ. In literature, [3], these are given as

$$\epsilon = 119 k_B, \quad \sigma = 3.41\text{Å} \tag{4.2}$$

In this chapter we will proceed much as in Chapter 3: we will first plot MD results against experiment with these figures and then fit the ϵ, σ parameters to the data to give best-fit values. To give a comparison of codes, we will run the argon simulation using MDL, LDM, DL_POLY and TOWHEE.

Figure 4.1 shows an example isotherm generated by MDL which was driven by SIMRUNNER. The MD result is rather noisy but shows a clear monotonic curve. Using the algorithm in Chapter 3, a flat coexistence curve was also generated as shown.

Figure 4.2 shows the comparison between experimental and MDL and TOWHEE $p - V$ curves (experimental data taken from Figure 6.1 and Figure 6.2). There is considerable discrepancy between the simulation and experimental curves with MDL generally underestimating the critical temperature and TOWHEE overestimating it.

Taking the phase transition volumes from Figure 4.2 gives the $T - \rho$ phase transition curves in Figure 4.3. Here, a comparison is given with all 4 simulation codes. It is interesting to see how similar the DL_POLY result is to the TOWHEE result despite MDL being algorithmically much more similar to DL_POLY. Note that the same ensemble (and other numerical details) were used in all codes as far as possible. However, there are certainly algorithmic differences, for instance MDL uses the leapfrog update scheme compared to DL_POLY's velocity Verlet scheme. However, in principle, these algorithmic differences should be greater moving from MD to MC.

Figure 4.4 shows the pressure-temperature phase diagram as calculated by the various simulation codes. This confirms the result of Figure 4.2 that DL_POLY and TOWHEE overestimate the critical point, while MDL and LDM underestimate them.

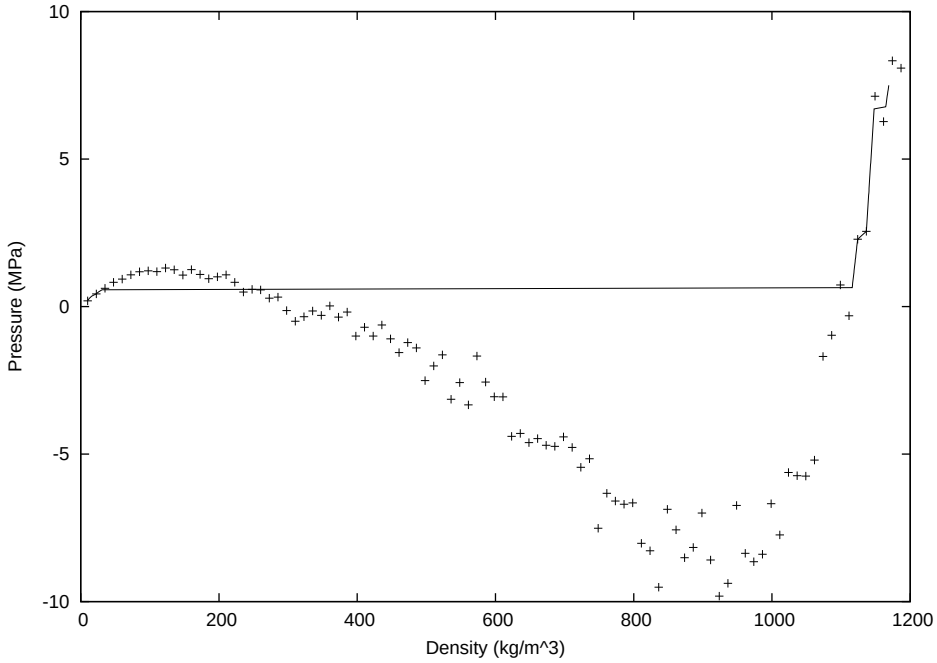


Figure 4.1: Argon 100 K isotherm. Raw data shown as symbols and flat coexistence curve as solid line

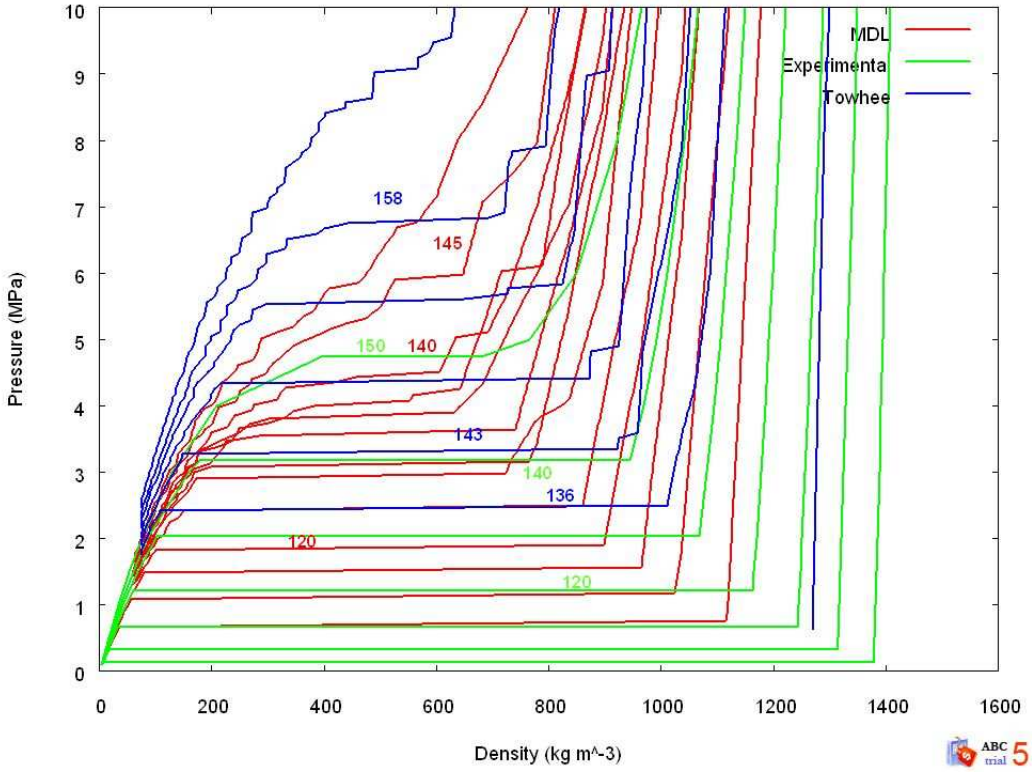


Figure 4.2: Comparison between MDL,TOWHEE and experimental data

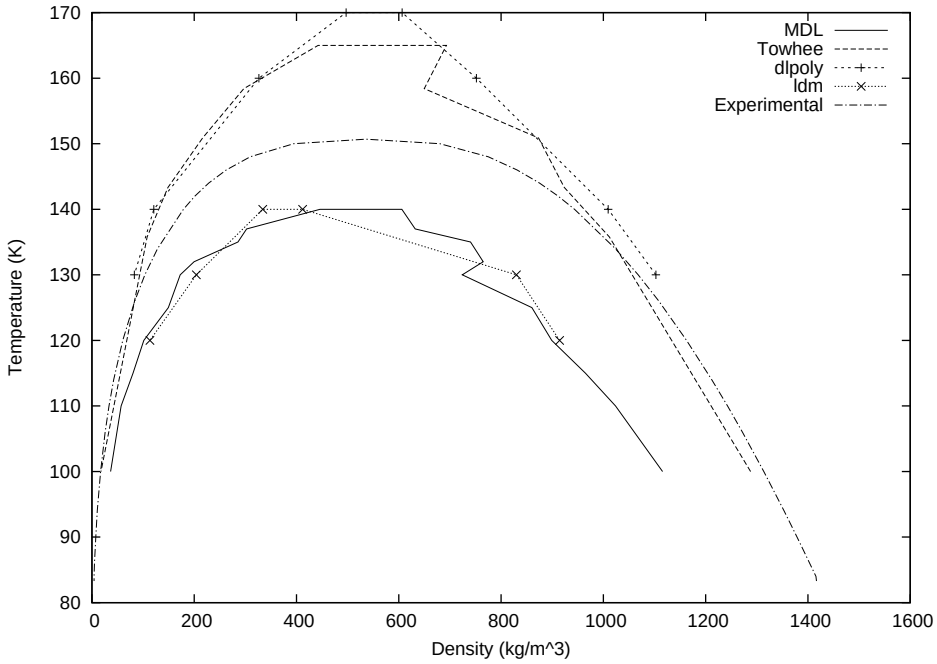


Figure 4.3: $T - \rho$ phase diagram as calculated by various simulation codes, compared to experimental results

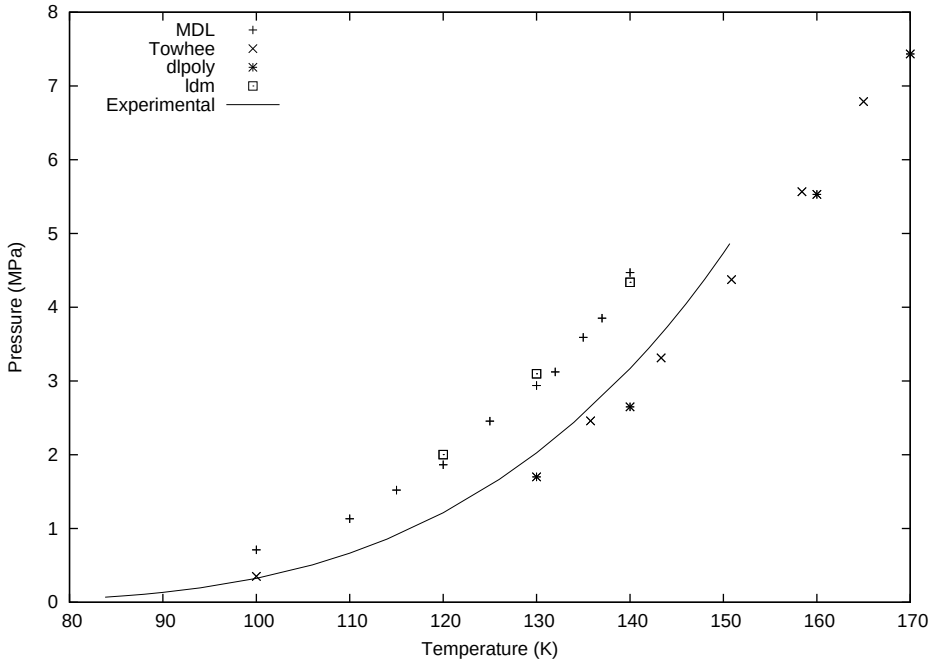


Figure 4.4: $p - T$ phase diagram as calculated by various codes compared to experiment

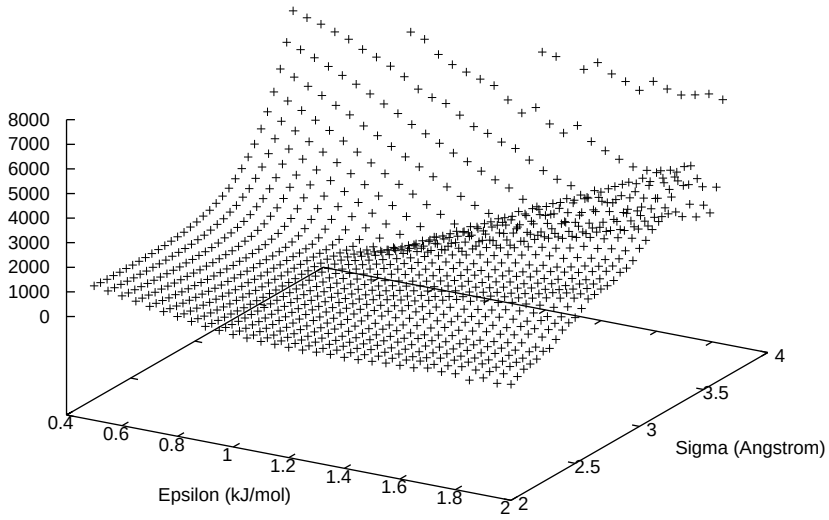


Figure 4.5: Sum of square differences between MDL and experimental results for various ϵ, σ values

4.2 Inverse modelling

Concentrating on the MDL code, we now try to adjust the Lennard-Jones parameters so as to match experimental data. To accomplish this, SIMRUNNER was set to vary ϵ and σ and the square difference between MD and experimental data was calculated in each case. The result in shown in Figure 4.5. Sum square differences were added over a number of temperatures as well as volumes for each point. In total MDL was called one million times to obtain Figure 4.5. SIMRUNNER ran for 43 hours on a quad core worktation computer. MDL jobs were run simultaneously on each of the 4 cores to speed up the calculation.

The best fit result is

$$\epsilon = 132.2k_B, \quad \sigma = 3.30\text{Å} \tag{4.3}$$

which gives the isotherms shown in Figure 4.6. With these fitted values, the agreement is excellent and much better than the fitted van der Waals function of Section 3.5. This indicates that the MD code indeed represents experimental reality better than the simple van der Waals function.

The $T - \rho$ phase diagram is shown in Figure 4.7 showing good agreement with experimental data.

4.3 Numerical difficulties

In forming the "flat" $p - V$ graphs and consequent phase diagrams we encountered some numerical difficulties which on occasion had to be corrected manually. Figure 4.8 shows a situation where the program discovers the flat curve correctly but fails to identify the position of the phase transition. The reason is that pressure is incremented in equal intervals and because of the long "tail" of the $p - V$ curve, the volume distance between points at high volume (eg 600 nm^3) can be greater then at the phase transition itself (at 50 nm^3). The driver SIMRUNNER detects the phase transitions as the point where the volume jumps the most so, in this case, gets the wrong volume. A better strategy would be to detect the maximum *density jump* which, as shown in Figure 4.9, is clearly the greatest at the phase transition.

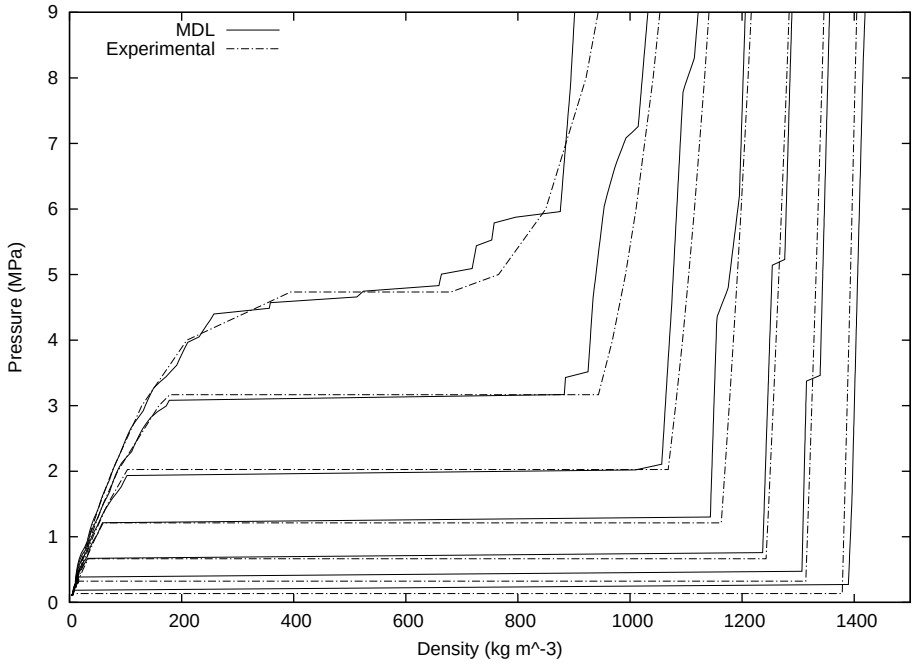


Figure 4.6: Comparison between MDL and experiment for best fit values of ϵ, σ

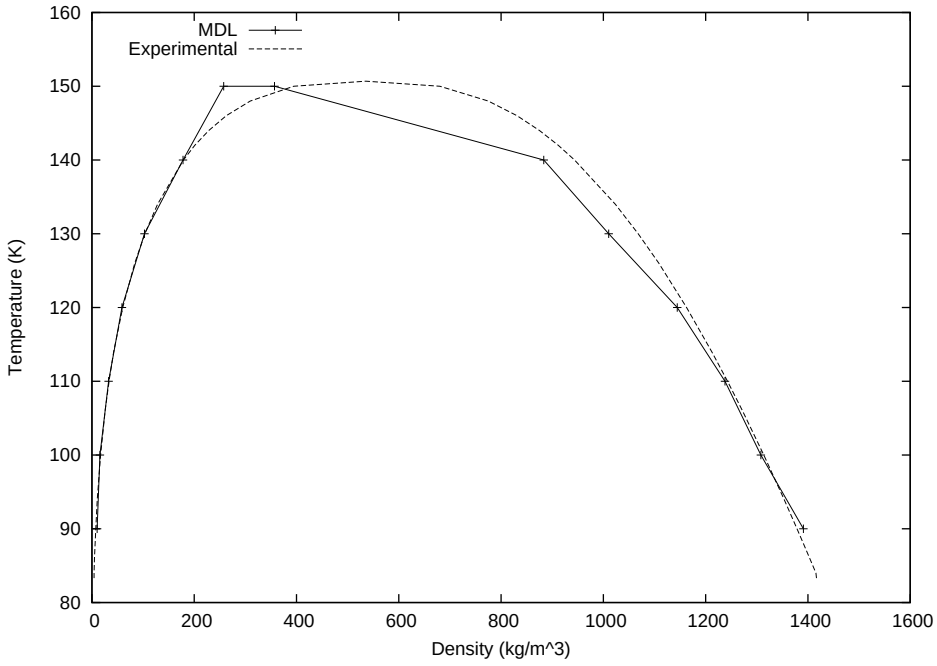


Figure 4.7: $T - \rho$ phase diagram using best fit values, compared to experiment

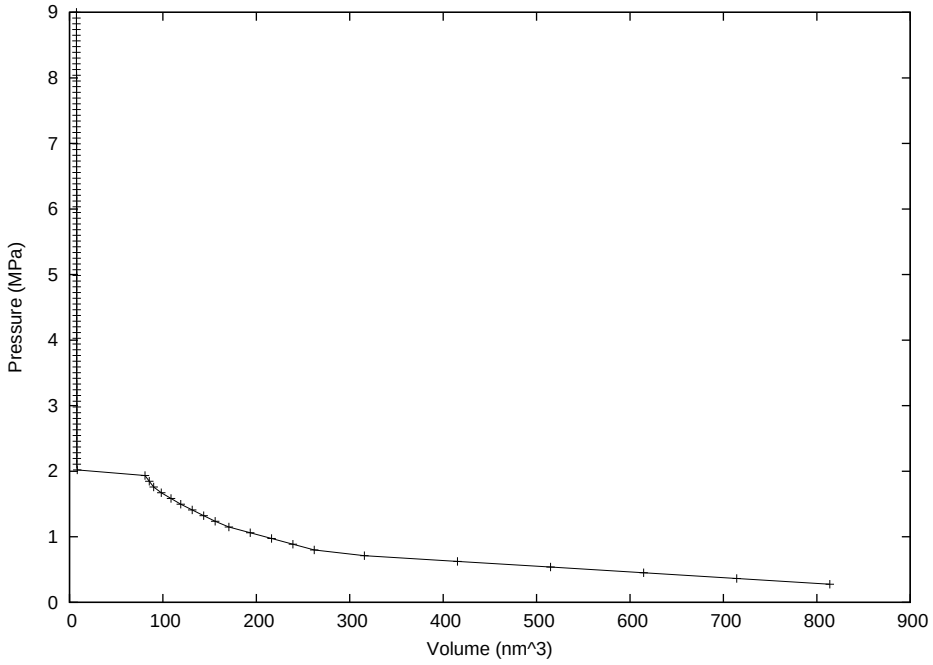


Figure 4.8: Pathological case where SIMRUNNER will find the wrong phase transition point

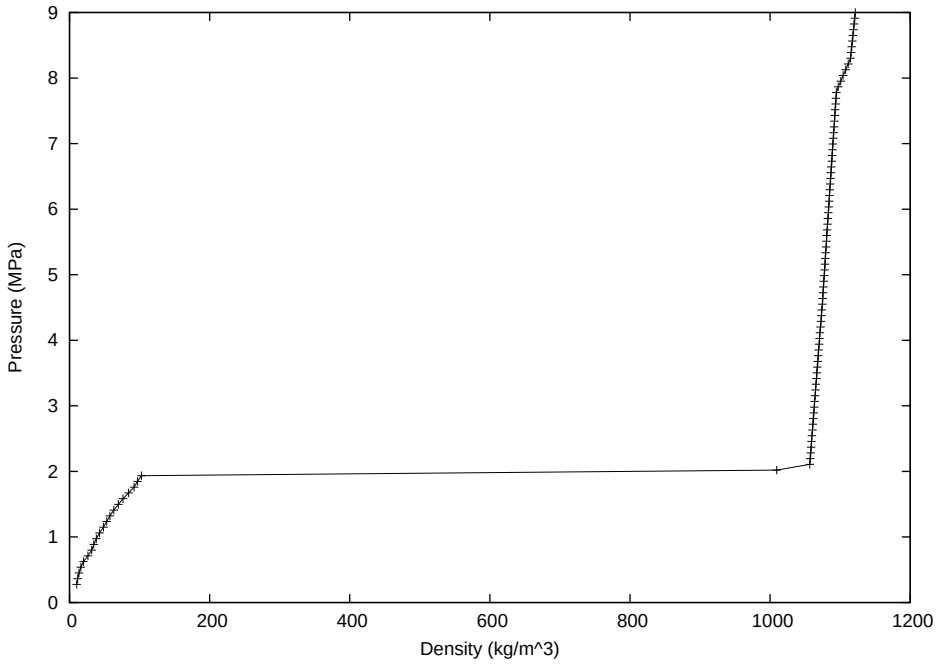


Figure 4.9: Same as previous graph on a density scale. In this case, the phase transition is clear

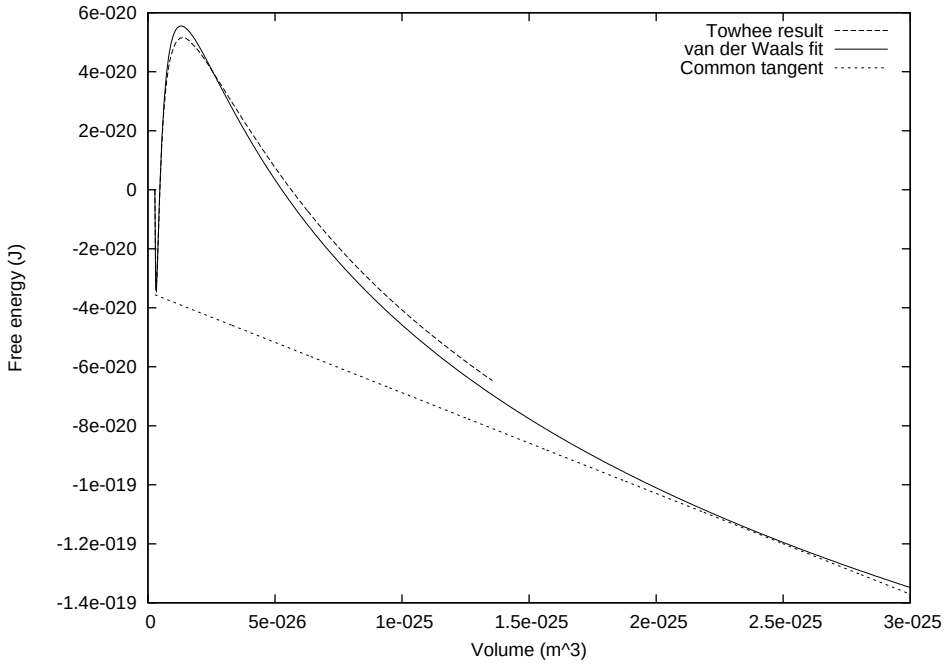


Figure 4.10: Free energy from MDL extrapolated using the van der Waals free energy. A common tangent is shown

Another issue which can arise is that the volume range is not great enough to see both ends of the phase transition. This is especially acute when we are simulating a range of temperatures each of which will have different phase transition points (SIMRUNNER uses a single, user specified, density range). In this case, rather than running the simulation again, it is possible to extrapolate the curves by comparison, for example, to the van der Waals analytical result. This technique is shown in Figure 4.10 which has been fitted to the van der Waals free energy (3.42). We can find a common tangent to the extrapolated curve giving a transition pressure of 0.35 MPa. The Gibbs free energy function with this pressure is shown in Figure 4.11 showing two equally deep minima of equal depth. These are seen more clearly in the logarithmic graph of Figure 4.12. It is clear that the van der Waals Gibbs energy approximates the MD Gibbs energy rather well.

4.4 Accuracy of simulation codes

The argon simulation consists of single-atom molecules connected by a single Lennard-Jones pair potential. All such problems can be mapped onto a series of master curves via a change of variables. If we write distances in terms of σ, energies in ϵ, masses in terms of the atomic mass being simulated etc, all equations governing the simulation can be written with σ, ϵ and atomic mass set to 1. Results are then given in terms of reduced Lennard Jones Units:

$$\begin{aligned} T^* &= \frac{k_B T/\epsilon}{} \\ \rho^* &= \rho\sigma^3 \\ p^* &= p\frac{\sigma^3}{\epsilon} \end{aligned} \tag{4.4}$$

The single particle Lennard-Jones fluid has been well studied in literature. The most accurate estimate

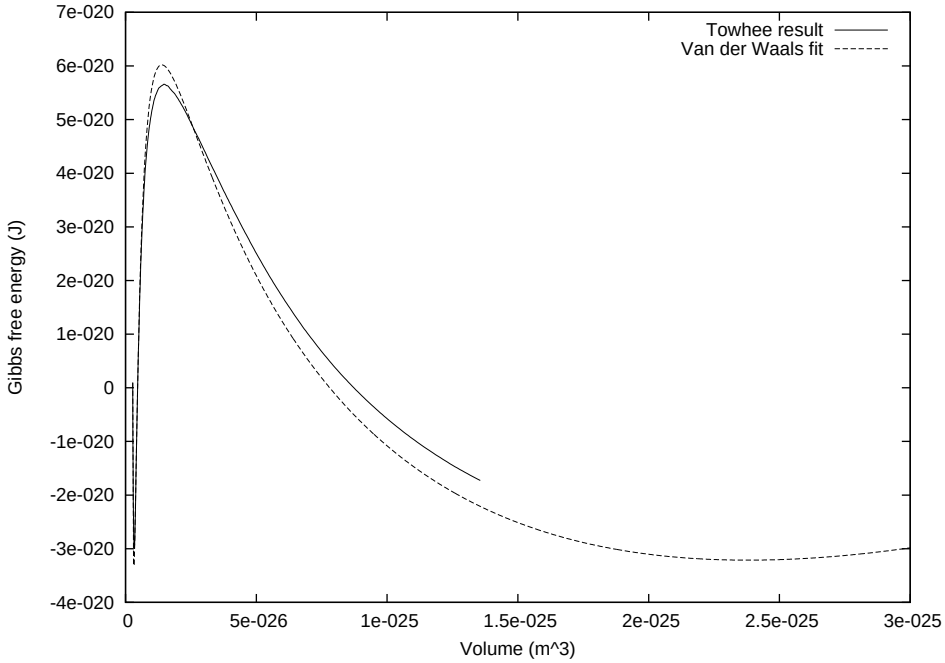


Figure 4.11: Corresponding Gibbs energy at 0.35 MPa

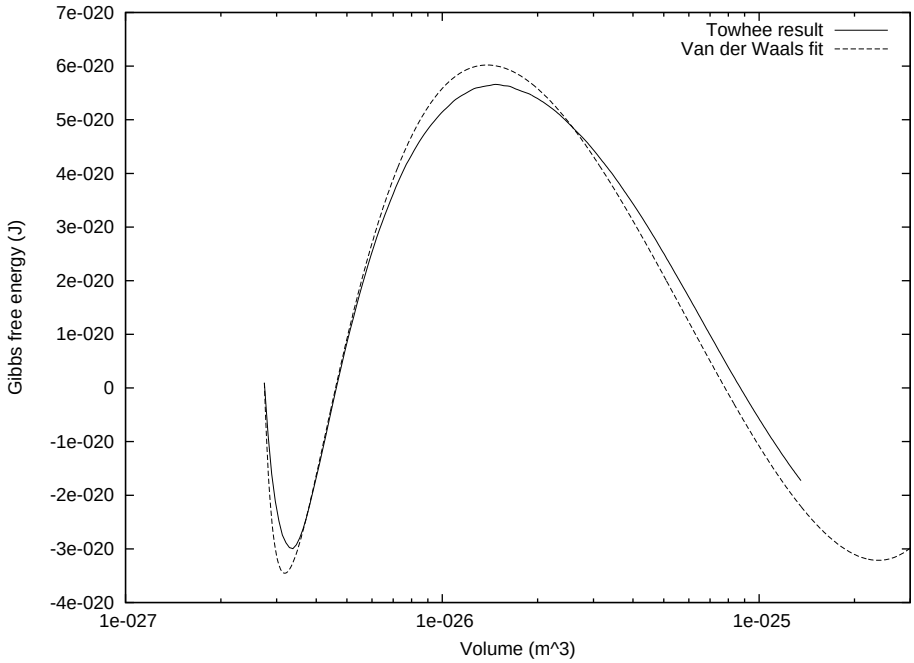


Figure 4.12: Previous graph on a logarithmic scale

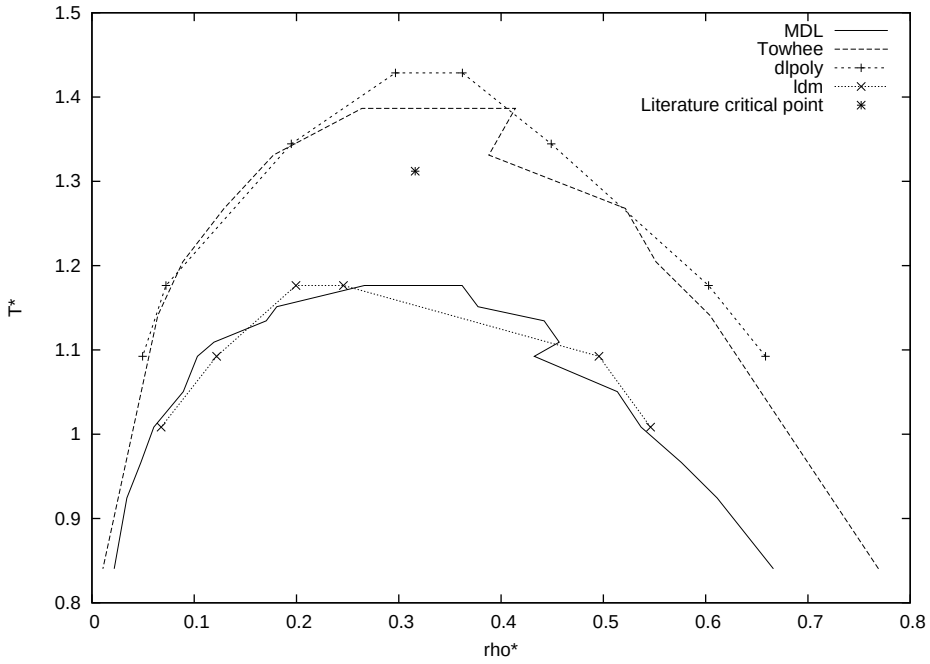


Figure 4.13: Phase diagram in reduced units with comparison to literature critical points, [11]

of the critical point, in reduced units is, [11]:

$$T_c^* = 1.3120, \quad \rho_c = 0.316, \quad p_c = 0.1279 \tag{4.5}$$

Figure 4.13 shows the $T-\rho$ phase diagram from the 4 simulation codes in reduced units, compared to this critical point. It seems that DL_POLY and TOWHEE give a more accurate critical point than MDL/LDM. In Section 4.2, we adjusted ϵ and σ to give a best fit to experimental data using MDL giving best fit values (4.3). We could equally well have done this for DL_POLY or TOWHEE giving another set of best fit values. Such a fit would give $\epsilon < 119k_B$ rather than greater as in the MDL case. Thus, when performing inverse modelling, it is important to specify which solver is being used. To get to the "true" value of pair potential parameters would ideally involve an average over many simulation codes. The question of whether the best fit values (4.3) are actually better than the literature values (4.2) remains a difficult one. In general, we will not have the luxury of scaling laws like (4.4).

An interesting corollary of (4.4), is

$$T_c = \frac{\epsilon T_c^*}{k_B} \tag{4.6}$$

$$\rho_c = \frac{\rho_c^*}{\sigma^3} \tag{4.7}$$

$$p_c = p_c^* \frac{\epsilon}{\sigma^3} \tag{4.8}$$

where we can use the literature values (4.5). Thus, if the value of ϵ is doubled in the simulation code, the predicted critical temperature will double, which critical density varies according to the average volume occupied by atoms given as σ^3. Of course, these conclusions only apply to the ideal Lennard-Jones fluid.

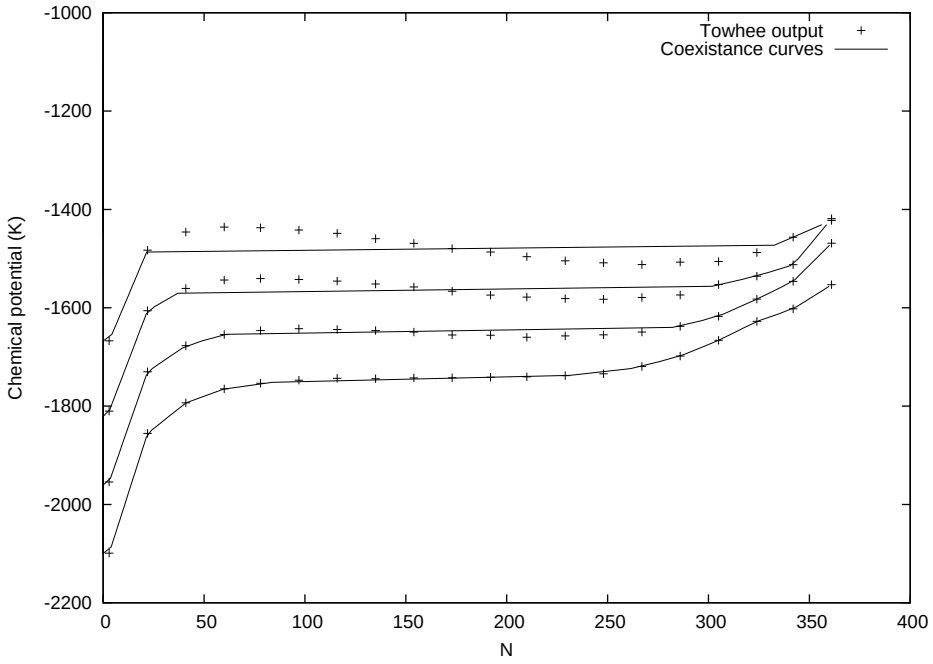


Figure 4.14: Argon chemical potential calculated by TOWHEE

4.5 Variation of number of particles

In the previous section, we have taken a fixed number of argon atoms, varied the volume and measured the pressure. The alternative way to calculate the phase diagram is to fix the volume, vary the number of particles and measure the chemical potential. It is clear from Figure 3.9 that this technique is not as efficient as the method of varying the volume but we do it here to test how accurately the various solvers can calculate chemical potential. Calculation of chemical potential is essential for the binary alloy calculations (and multiple species mixtures in general) we will carry out in Chapter 7.

MDL does not have chemical potential calculations built it (nor does DL_POLY) so we have implemented a technique whereby "ghost" atoms are present. These atoms are evolved by MDL just like normal atoms but, by default, do not interact with other atoms or with each other. However, these ghost atoms are periodically replaced with "real" atoms (argon in this case) and the potential energy difference as a result of these atoms being present is used to calculate the chemical potential as shown in (1.3). (The MDL subroutine which calculates energies is called twice with the above mentioned rearrangement in between. This avoids the need for significant reprogramming of MDL, see Appendix A for details). TOWHEE has a built-in facility to calculate chemical potentials: it also uses a variant of the ghost insertion method. Both MDL and TOWHEE where run via the NSIM driver to produce the results in this section.

Figure 4.14 shows the TOWHEE result when number of atoms is varied. The chemical potentials show non-monotonic curves which are indicative of a phase transition and may be compared to the analytical result Figure 3.6. It is remarkable how well TOWHEE does considering that some of the simulations have a very low number of atoms (the lowest-N simulation has only $N = 3$ atoms). A phase diagram was prepared from this graph, Figure 4.15, and gives reasonable agreement with experimental data. Again, TOWHEE tends to overestimate the critical temperature.

Figure 4.16 shows how *not* to calculate chemical potential in MDL. Here, in an attempt to improve statistics, we inserted 8 ghost argon atoms rather than 1. The total potential energy difference was calculated and divided by 8 to give an average. This seems like a good way to improve statistics but it

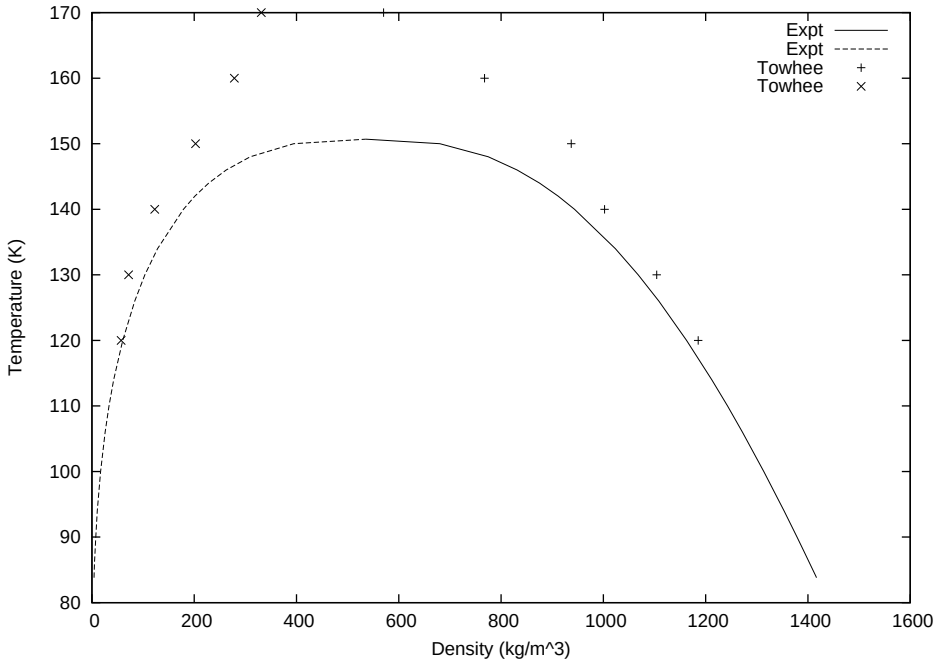


Figure 4.15: Argon phase diagram calculated using TOWHEE chemical potentials, compared to experimental data.

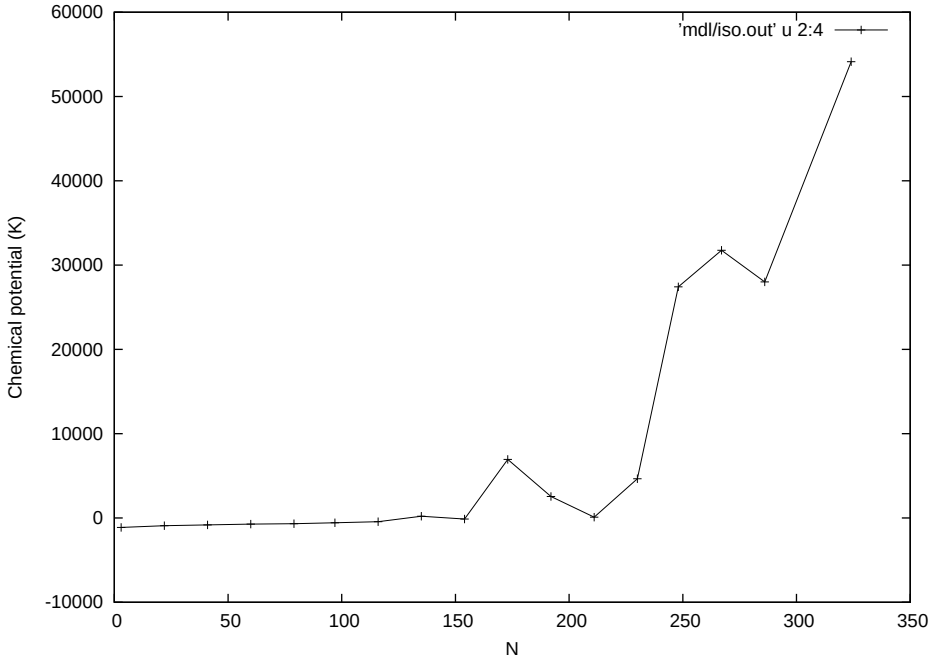


Figure 4.16: The wrong way to calculate chemical potential, using 8 ghost atoms in MDL

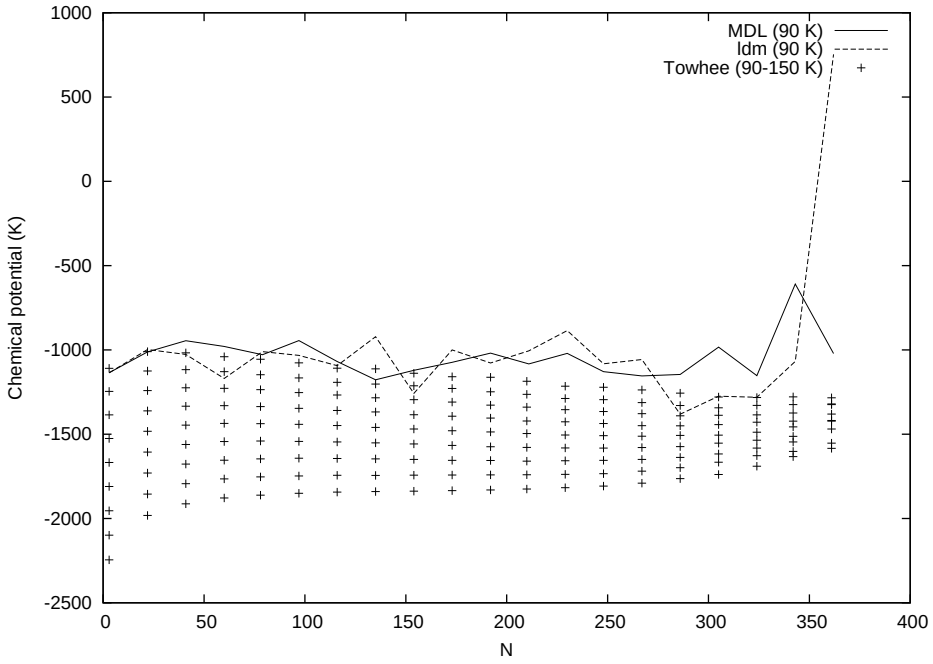


Figure 4.17: MDL/LDM chemical potentials using 1 ghost atom compared to TOWHEE result

fails because

$$\langle \exp(-V_{test}/k_B T) \rangle \neq \exp(-\langle V_{test} \rangle / k_B T) \tag{4.9}$$

Specifically, with 8 ghost atoms present it is likely that one will overlap a real atom causing $\langle V_{test} \rangle$ to diverge resulting in the very large numbers in (1.3).

Figure 4.17 shows the MDL and LDM results with just 1 ghost atom compared to TOWHEE. The results are more reasonable now but still the MDL/LDM results are very noisy compared to TOWHEE and are unlikely to produce sensible phase diagrams. Increasing the number of time steps does not seem to improve this. It seems that TOWHEE is using a more intelligent algorithm for chemical potential calculation, [1].

4.6 Conclusions

In this chapter, we have simulated argon gas using several molecular dynamics and Monte Carlo codes. Using the textbook values of pair potentials gives a good set of phase transitions near to the experimental critical point and the critical point itself is reasonably well reproduced compared to experiment. We have modelled argon gas as a simple Lennard-Jones fluid and MD/MC predictions of all such fluids can be mapped onto a single set of dimentionless master curves. The position of the critical point in these "Lennard-Jones units" has been thoroughly researched and is known to good accuracy (though it cannot be derived analytically: the best estimates come from length numerical simulations). When comparing this known result to the various simulation packages, it seems that TOWHEE and DL_POLY give the best prediction with MDL underestimating the critical temperature and the density at the liquid side of the transitions. The fact that TOWHEE and DL_POLY agree despite being based on completely different numerical techniques seems to indicate that MDL may be at fault. However, adjusting the paramters away from their textbook values using inverse modelling in SIMRUNNER allows MDL to reproduce experimental results extremely well and significantly better than the analyical van der Waals formula. This shows that MD results can, as would be expected, give better agreement to experiment than analytical formulas with

the same number of adjustable parameters. The automatic inverse-modelling feature of SIMRUNNER will be useful in future, in terms of fitting unknown pair potential paramters.

We have also shown that the free energy surface can be explored by either varying the volume or the number of particles and measuring (in MD) the pressure and chemical potential that results. The two approaches should give the same result. This seems to work well in TOWHEE but is problematic in MDL despite the additional programming we have added to MDL to calculate chemical potentials using the ghost insertion method (neither MDL no DL_POLY support chemical potential calculation "out of the box": they should!). It seems that TOWHEE is using a more sophisticated technique to model chemical potentials and we need to investigate this further. Of course, there is no real need for chemical potential modelling in a single species calculation: phase diagrams can be obtained more easily from pressure-volume scans. Where chemical potentials *are* needed is in multiple species systems such as the binary alloys of Chapter 7. That is why we have tested the accuracy of chemical potential calculation in this chapter.

Finally, some discussion has been given on how to get the best phase diagrams from the MD/MC results. There are some difficulties due to finite scanning intervals and insufficient range of volume or number of particles, and we have given advise on how to overcome these problems.

Chapter 5

Water

The first multi-atom molecular we will consider is water. Here, we use the SPC/E model for water, [12]. In MD parlance, a "model" is simply a set of pair potential values for a particular system. In this case, the H-H and H-O interactions may be neglected (since the strength of interactions is generally proportional on atomic number). The O-O potential is modelled using a Lennard-Jones potential with $\epsilon = 0.65015$ "kJ/mol" and $\sigma = 3.1656$ Å. The partial charge on the hydrogen atoms is $0.4238|e|$ and that on the oxygen molecules is $-0.8476|e|$. The water molecule is modelled as a rigid body and this is implemented using the quaternion algorithm [3]. Water molecules are set up with an O-H bond length of 95.7 pm and a H-O-H bond angle of 104.5°. A picture of the simulation is shown in Figure 5.1. The simulation was run in MDL at 298K.

Water was one of the first simulations we did in this project and the first thing we checked was the radial distribution function (RDF). This is plotted in Figure 5.2 against published results from another MD code, [12]. Our result agrees very well with the published data. However, this is really just proof that the simulation was set up correctly as RDF in MD is largely depending on initial atom positions: it is not a prediction as such (www.lsbu.ac.uk/water, see Chapter 10). We used Figure 5.2 as a test for adjusting various simulation parameters like number of times steps, time step length, thermostat relaxation time and so on. Details are given in [13], basically the simulation is fairly insensitive to these factors although, if time step is increased too much the simulation will eventually diverge. We also tried using constraint rather than rigid body dynamics which is another way to set a (partially) rigid molecule in MDL. We also considered having an angular spring in the H-O-H angle with fixed O-H lengths. All of these simulations give similar results for the RDF which proves the correct implementation of these features of MDL.

The simulation results seemed very promising. However, using exactly the same simulation did not produce sensible results for pressure so we were not able to produce an isotherm at 298K. The pressure seemed to fluctuate wildly such that the fluctuation was greater than the pressure average and we were not able to see a phase transition. At 298K, well away from the critical point (647 K), water is a very dense, low pressure liquid and simulating such liquids far from the transition point turns out to be very difficult in MD. However, we then simulated water close to the critical point (575-675 K) and obtained good results as shown in Figure 5.3. Transforming these results into equlibrium isotherms gives Figure 5.4 showing a critical point of 600-625 K and a critical density of 300 kg/m^3 in good agreement with experimental results (critical point at 22.06 MPa, 321.4 kgm^{-3}, 647 K, [14]). Of course, it is not surprising that the results agree with experiment since the pair potential values in the SPC/E model are tuned to give the correct critical point. However, this shows that we can successfully calculate phase transitions in multi-atom molecules, despite the initial difficulties mentioned above. Modelling the molecule as a rigid body seems to give accurate results. In the next chapter, we will investigate another 3-atom molecule, namely carbon dioxide. We will model CO_2 at very high temperatures and pressures to see if MD can produce sensible results outside of the "comfort zone" where parameters were fitted.

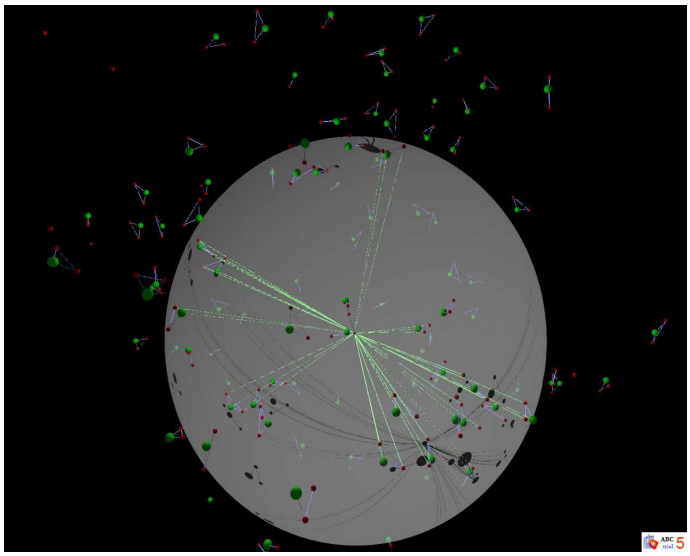


Figure 5.1: Water simulation showing the Verlet neighbour list surrounding one atom

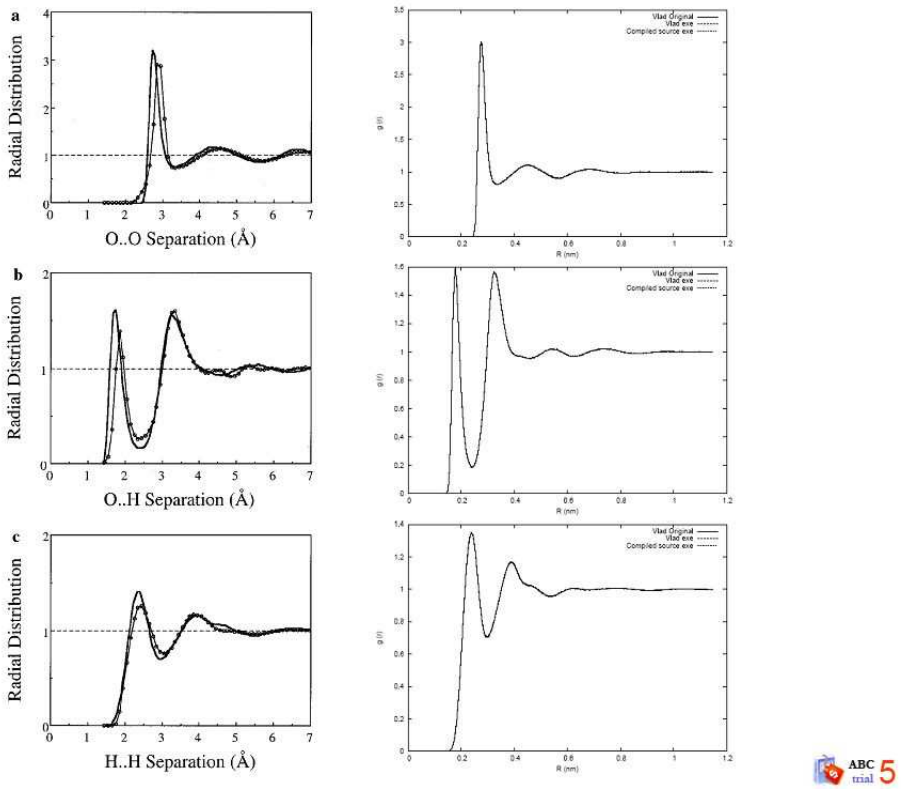


Figure 5.2: Comparison of RDF as predicted by MDL (right) and published values, [12] (left)

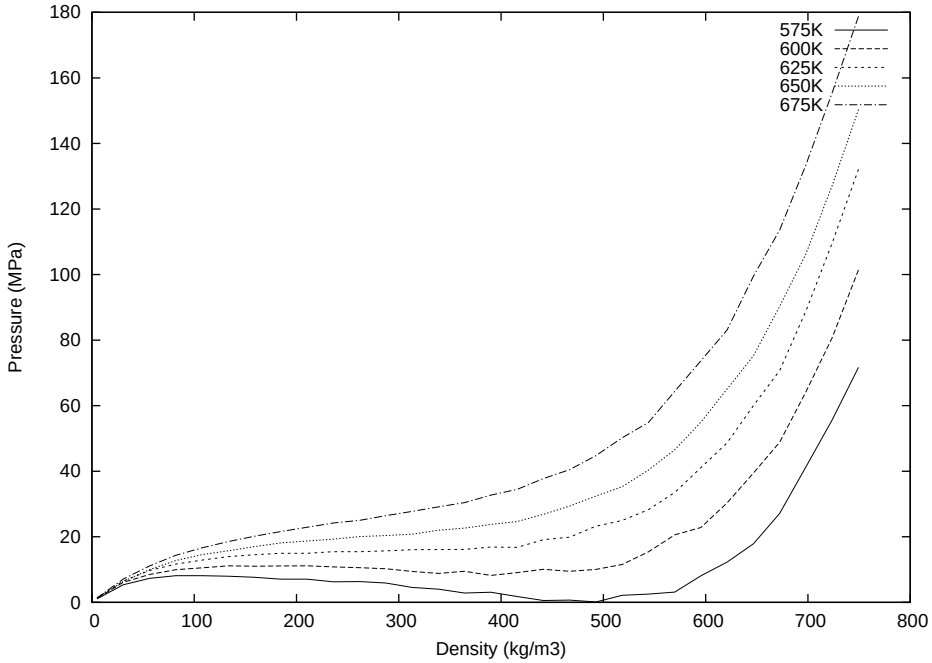


Figure 5.3: Isotherms for water simulated by MDL

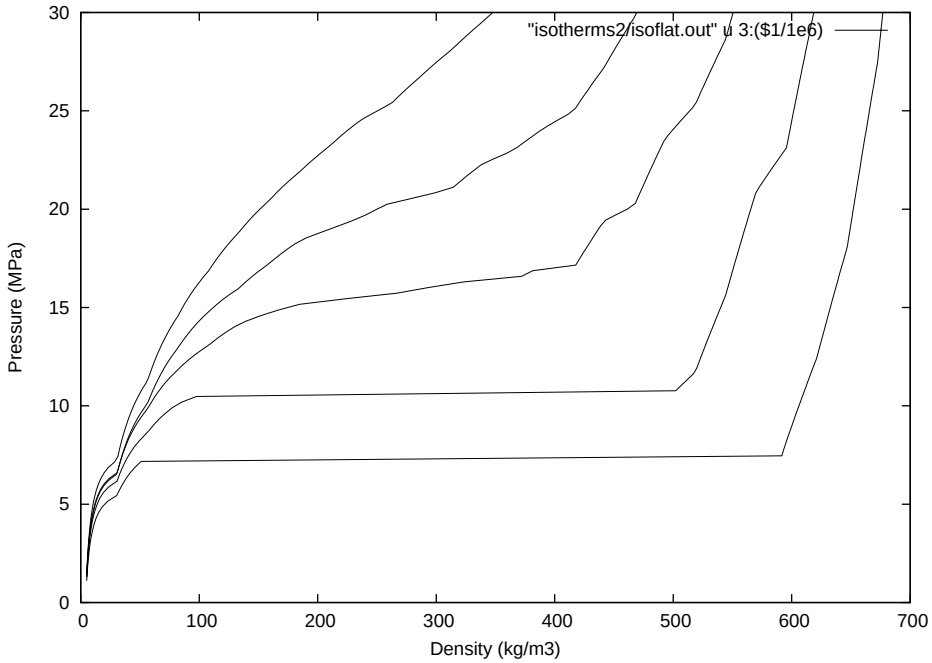


Figure 5.4: Equilibrium isotherms for water from MDL. Bottom to top, $T = 575$, 600, 625, 650, 675 K

Chapter 6

Molecular dynamics of Ar and CO_2 - thermodynamic data extraction

6.1 Introduction

This chapter describes the extraction of thermodynamic data relevant to the modelling of real gas behaviour from molecular dynamics simulations. Examples are give for both Ar and CO_2 showing comparisons between thermodynamic data derived from simulations. (P-V-T and enthalpy) and well-established experimental values. The procedures for assessing real gas model parameters for use by NPL's MTDATA code are also described. A practical application related to the decomposition of carbonate minerals show the benefits of using real gas models. In contrast to previous chapters, this chapter deals with system well above the critical temperature and pressure. As we shall see, these regimes are of considerable industrial importance.

6.2 Thermodynamic modelling of real gases

A real gas differs from an ideal gas in that the Pressure-Volume-Temperature (PVT) properties are not well described, for a particular application, by the ideal gas equation:

$$PV = RT$$

Where P is the pressure, V is the molar volume, R is the molar gas constant (8.314472 $J.K^{-1}.mol^{-1}$) and T is the thermodynamic temperature.

Many equations of state, that reduce to the ideal gas relation $PV = RT$ as $P \rightarrow 0$ have been devised. References [15] and [16] describe examples, with many adjustable parameters, that aim to represent all the PVT and thermodynamic properties of the gas/fluid above and below the critical temperature. However for many practical applications of computational thermodynamics it is only the initial departure from ideality that needs to the correctly modelled. The behaviour of a real gas in terms on the PVT properties is shown in figures 6.1-6.4 for argon based on the evaluation of Tegeler et al. [15]. In this chapter we shall focus on the corrections from ideality that will allow real gases to be used in the prediction of complex phase equilibria with potentially many solid and/or liquid phases. Other chapters in this report investigate the prediction of liquid-vapour equilibria using molecular dynamics.

An additional factor found in many computation thermodynamic and phase equilibrium problems is that a rich chemical speciation may be present even for a relatively small number of elements. An extreme example of this would be the behaviour of sulphur vapour. The chemical state of the vapour is well established as containing (at least) species S, S_2, S_3, S_4, S_5, S_6, S_7 and S_8. The consideration of multiple chemical species, in themselves obeying the ideal gas equation, and a simple ideal mixing

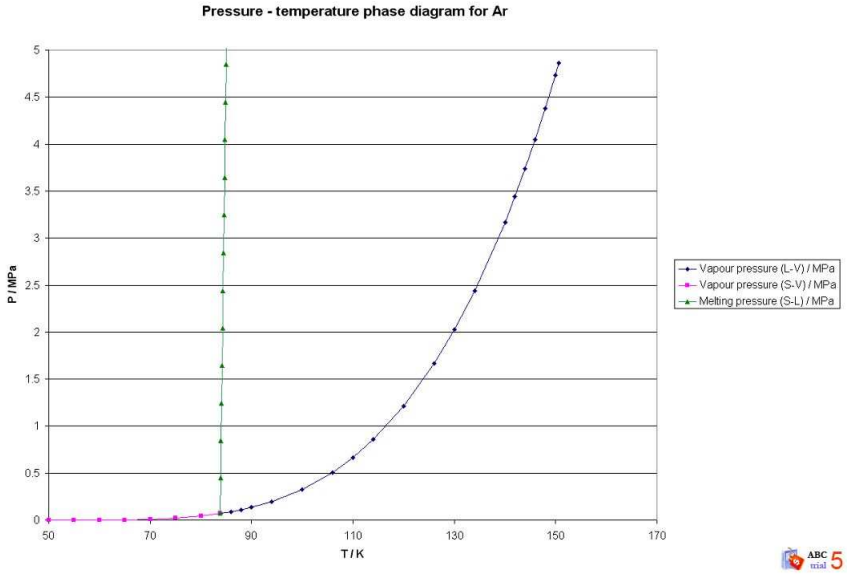


Figure 6.1: Experimental P-T phase diagram for Ar

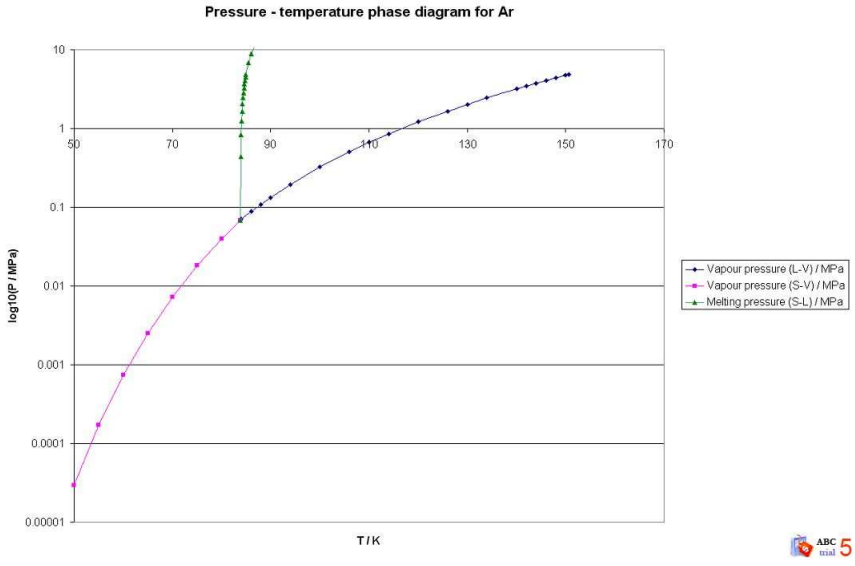


Figure 6.2: Experimental $\log_{10}(P) - T$ phase diagram for Ar

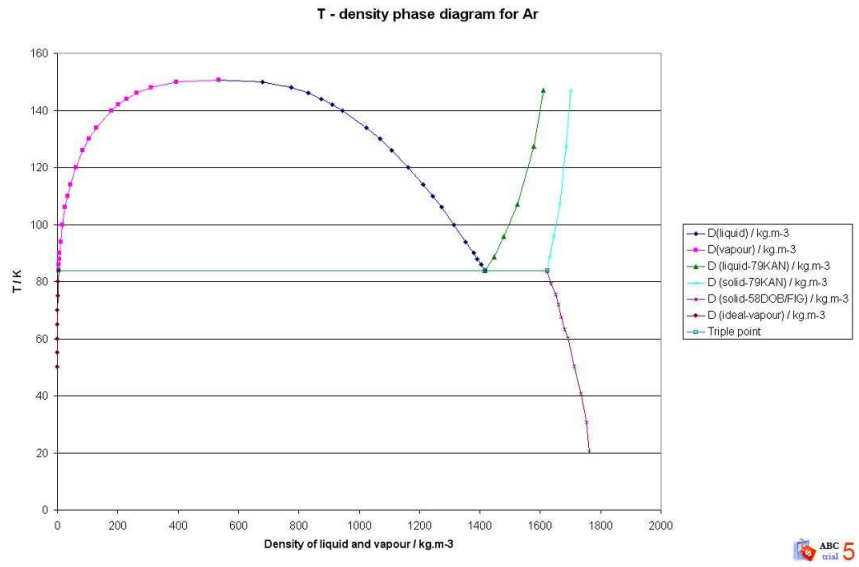


Figure 6.3: Experimental T - density phase diagram for Ar

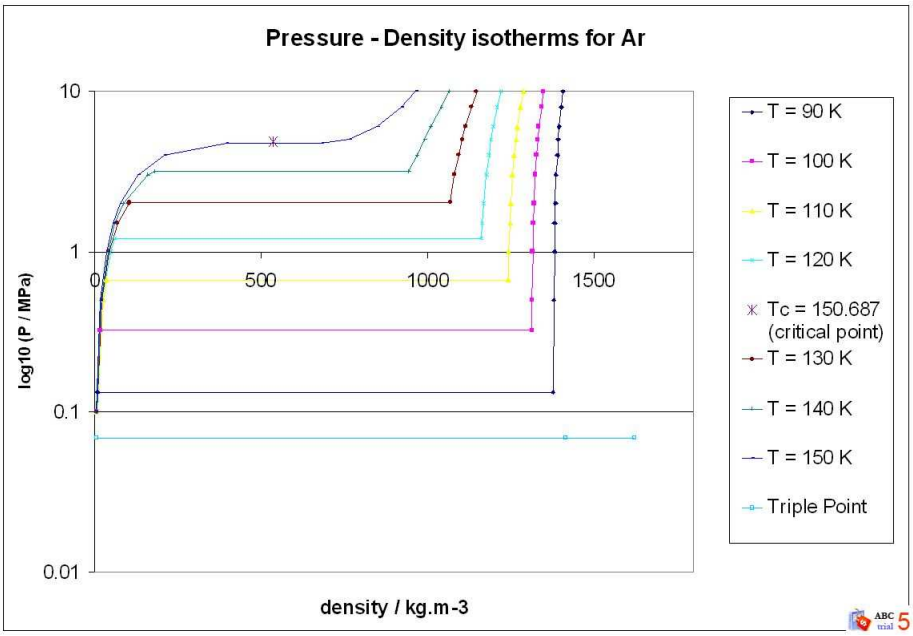


Figure 6.4: Experimental P - density isotherms for Ar

Pressure / MPa	T(ideal) / K	T(exp) / K	ΔT (ideal-exp) / K
0.01	319.023	318.958	0.065
0.1	373.216	372.756	0.460
1	456.362	453.036	3.036
10	617.019	584.149	32.87

Table 6.1: Liquid and gaseous water coexistence, calculated using ideal gas and experimental result

Pressure / MPa	T(dimer) / K	T(exp) / K	(ΔT (dimer-exp) / K
0.01	318.978	318.958	0.020
0.1	372.790	372.756	0.034
1	453.130	453.036	0.094
10	598.608	584.149	14.459

Table 6.2: Liquid and gaseous water coexistence, calculated using non-ideal dimer gas and experimental result

entropy creates a model that is effectively non-ideal (ie a real gas model). This is discussed in more detail latter in this chapter with reference to H_2O vapour pressures and Ar P-V-T behaviour.

With the requirements to model the departure from ideality of a potential complex, richly speciated system, it is clear that an equation of state with many parameters is not appropriate.

A selection of the application areas where real gas models may be appropriate are:

1. High pressure aqueous systems (eg PWR)
2. High precision humidity calculations (enhancement factors)
3. Fossil fuel combustion plant ($p > 1.5$ MPa)
4. High pressure lighting systems
5. Energy storage in pressurised gas and/or gas+liquid systems

The prediction of vapour pressure provides a simple illustration of the systematic errors that may result from inappropriate ideal gas assumptions. For a given pressure, below the critical temperature, liquid and gaseous water are in equilibrium at particular temperatures. These temperatures, calculated using an ideal gas model, are shown in Table 6.1 together with the well-established experimental values, [17].

Specific examples for Ar and CO_2 will be found later in this chapter.

6.2.1 Deviations from the ideal gas limit

Table 6.2 shows the results from repeating the calculations for water vapour pressure, shown in the table above, but now including a simple "hypothetical" water molecule dimer to effectively introduce non-ideality into the gas model.

These results show that very good correction for the non-ideality of the gas phase is provided up to a temperature of 450 K. However above this, although still an improvement on the ideal model, the results get progressively worse.

A similar example for argon is shown in Figure 6.5. Here the experimental gas density on the liquid-gas saturation line is compared with that calculated from an ideal gas model and also a model using an Ar_2 gaseous species whose thermodynamic properties were calculated using ab initio quantum mechanics.

Another simple model for the correction of the PVT properties of the ideal gas to approximate a real gas is via a virial expansion such as:

$$PV/RT = 1 + B'/V + C'/V^2 + D'/V^3 + ...$$

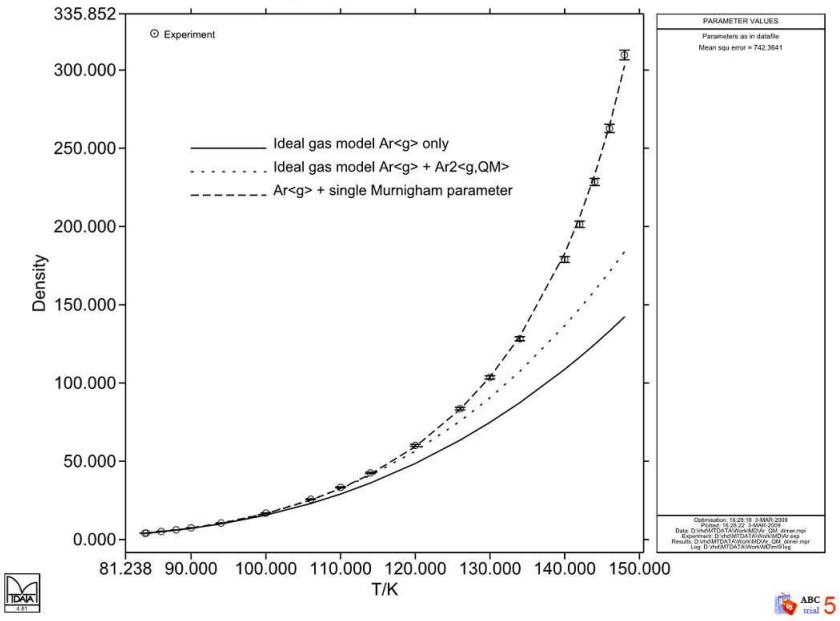


Figure 6.5: Comparison of experimental saturated vapour density for Ar with simple model predictions

An alternative form in terms of pressure is written:

$$PV/RT = 1 + BP/RT + CP^2/R^2T^2 + DP^3/R^3T^3 + ...$$

In general $B \neq B'$, however as $P \to 0$, $B \to B'$

Figure 6.6 shows a comparison of the second virial coefficient (B) of CO_2 as a function of temperature. This is discussed in detail later in this chapter.

NPL's MTDATA [18] code (from version 5.01) contains an implementation of the Murnaghan model (93DIN) modified for real gases. The Gibbs energy becomes:

$$G = G^o + RT\ln(P) + G_{pres}(T, P)$$

The original Murnaghan model was developed for liquids and solids, however the addition of the $RT\ln(P)$ contribution to the Gibbs energy makes this equally application for real gases.

G_{pres} is a complex function of T and P:

At the zero pressure limit as $P \to 0$, $G_{pres} \to 0$

Hence $G = G^o + RT\ln(P)$

$$V = (\partial G/\partial P) = RT/P$$

the ideal gas equation.

At finite pressures, where $G_{pres}(T, P)$ becomes non-zero, the limiting case deviation from ideal behaviour may be described by:

$$G = G^o + RTln(P) + PV'(T)$$

Where V' is a function of T only, hence:

$$V = (\partial G/\partial P) = RT/P + V'$$

or more usually:

$$PV = RT + V'P$$

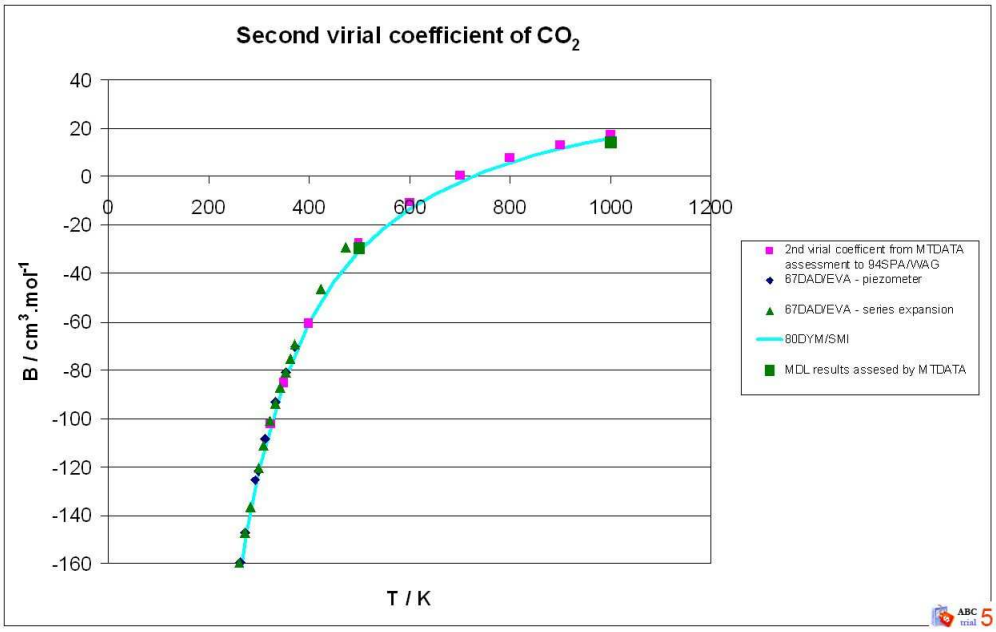


Figure 6.6: Second virial coefficient (B) of CO_2, comparison of values

Thus V' as $P \to 0$ becomes identical to the second virial coefficient (B). Furthermore from the expression for G_{pres} above it can be seen that V' has a temperature dependence at the limit of $P \to 0$ of:

$$V' = Aexp(a_0 + a_1 T^2/2 + a_2 T^3/3 + a_3 T^{-1})$$

This functional form is similar to the common function used for second virial coefficients [19]:

$$B = a - b\exp(c/T)$$

where $a = 0$, $A = -b$ and $a_3 = c$. At present the behaviour of MTDATA's real gas model limits the maximum value of the second viral coefficient to zero (effectively ideal behaviour) at high temperature.

See Figure 6.5 for a comparison of the experimental argon gas density on the liquid-gas saturation line compared with that calculated from an ideal gas model, a model using an Ar_2 gaseous species and also a model based on a single second virial coefficient (Murnaghan parameter A). It can be seen that the use of a dimer makes a useful but incomplete correction for the non-ideality of the gas. The use of the Murnaghan parameter, A, allows for a very effective correction.

6.2.2 Simulations using molecular dynamics

Molecular dynamics offers the possibility of performing virtual experiments from which useful thermodynamic properties may be extracted. In this chapter results are presented using the MDL code (version 0.91 (build 16.03.08)) with application to both argon and carbon dioxide. These two molecules were chosen because of the availability of high quality experimental thermodynamic data for comparisons ([15] and [16]), their technological importance and also because they represent two simple but significantly difference classes of molecule in terms of the form of the interaction potentials used.

In both cases the MDL code was used with the NVT ensemble and Nose-Hoover thermostat. Properties for 125 molecules were averaged over 5000 steps for Ar and 50000 steps for CO_2. The bulk phase SPC/E model with rigid body algorithm was used for both series of simulations.

6.2.3 Intermolecular potentials

In the MDL simulations reported in this chapter the following intermolecular potential and parameter values were used.

Ar

The inter-atomic potential used the parameters:

ϵ ("kJ/mol")	σ (Å)
0.99607	3.4050

CO_2

The inter-molecular potential used the following inter-atomic parameters, [20]:

i-j	ϵ ("kJ/mol")	σ (Å)
O-O	0.66937	3.0330
C-O	0.39566	2.8950
C-C	0.23388	2.7570

In addition to the above vdW potential electrostatic interactions were considered based on the following fractional charges on atoms:

Atom	Charge (e)
O	-0.3256
C	$+0.6512$
O	-0.3256

It is interesting to note that for simple Lennard-Jones type potentials the form:

$$U(r) = M/r^m - N/r^n$$

(where M and N are constants and n and m are usually fixed at 12 and 6 respectively) it has been shown [21] that analytical solutions for the second virial coefficient, B, are possible where $m = 2n$ (eg $m = 12$ and $n = 6$).

6.3 Results from molecular dynamics simulations

6.3.1 Argon - PVT results

This section presents the results of molecular dynamics simulations and comparisons with experimental data.

Figures 6.7 and 6.8 present pressure - density isotherms at 500 K for Ar. At 500 K argon is well above the critical temperature of 150.687 K so no indication of phase separation would be expected. The range of pressure and density is large and the rapid breakdown of the ideal gas assumptions is evident for pressures above 300 bar and densities above 400 kg.m^{-3}. The results of the molecular dynamics simulations (solid triangles) show a good degree of prediction well above the limits where an ideal gas model fails. To emphasis the validity of the molecular dynamics predictions Figure 6.8 shows the low pressure ($<$ 300 bar) and density ($<$ 400 kg.m^{-3}) region. Here the simulation results show very good agreement with experiment. Note this low-pressure region is where the argon atoms are attracted to each other and hence the pressure falls below the ideal gas value.

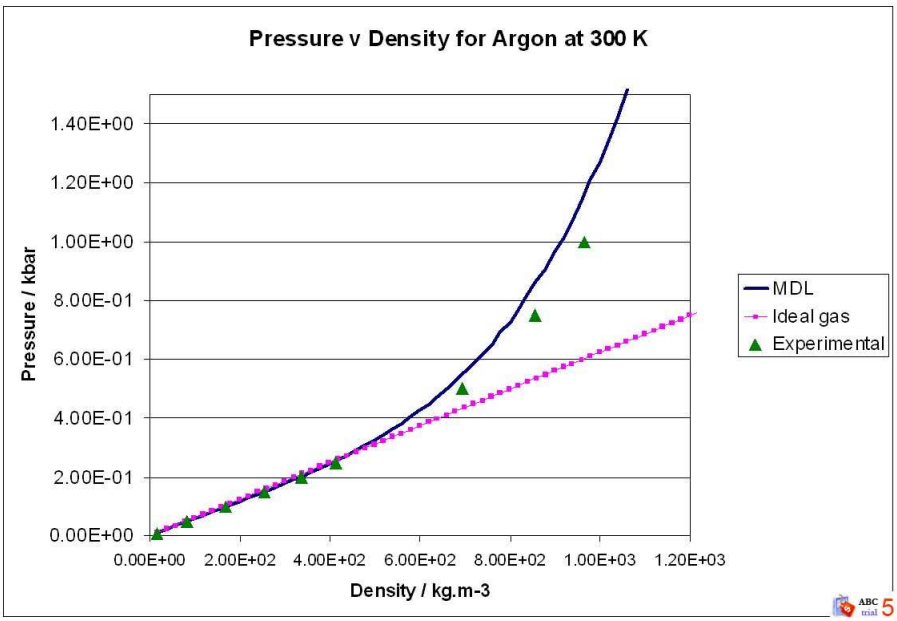


Figure 6.7: Comparison of molecular dynamics (MDL) simulation of PVT data for Ar and experiment at 300 K

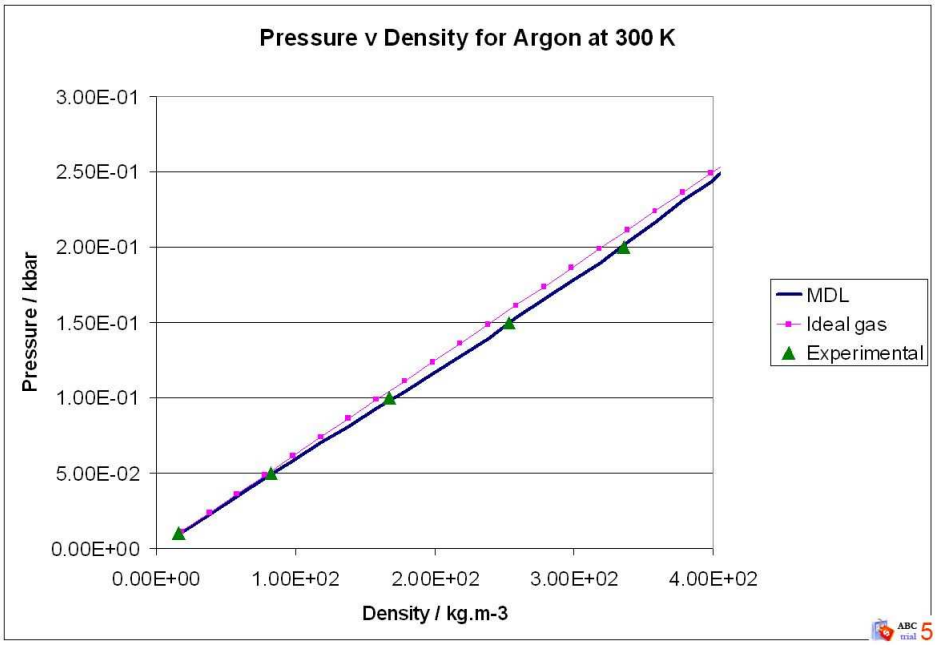


Figure 6.8: Comparison (low-pressures) of molecular dynamics (MDL) simulation of PVT data for Ar and experiment at 300 K

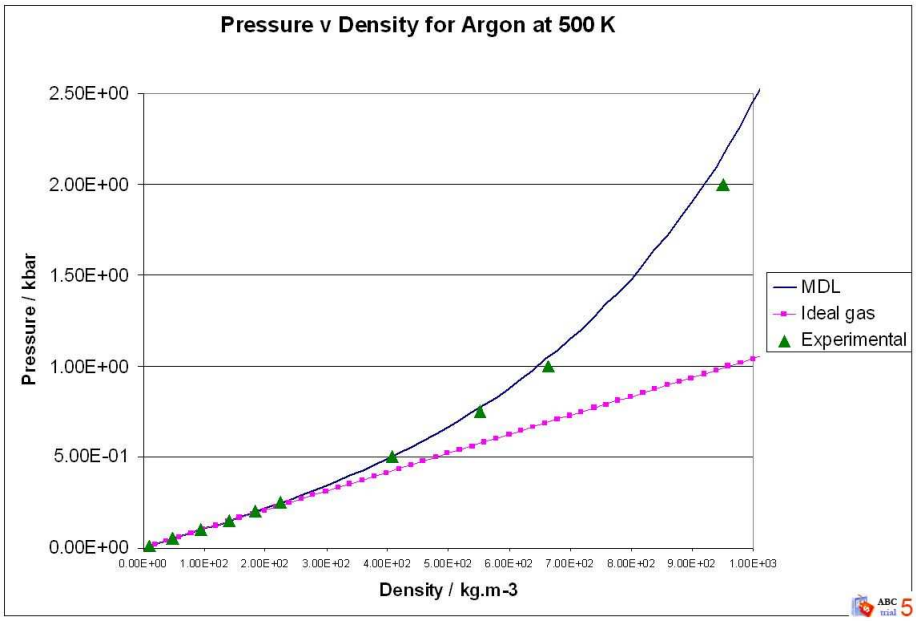


Figure 6.9: Comparison of molecular dynamics (MDL) simulation of PVT data for Ar and experiment at 500 K

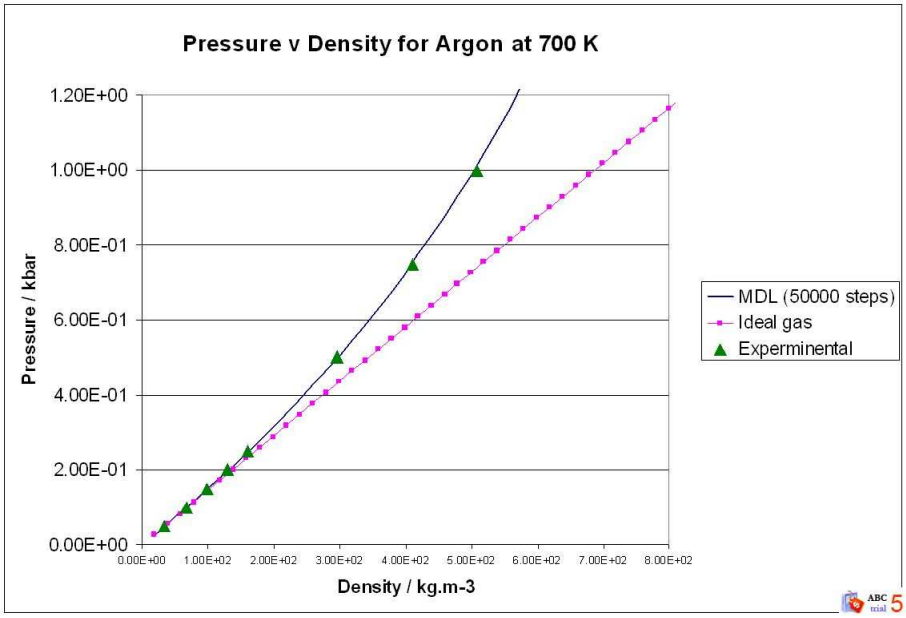


Figure 6.10: Comparison of molecular dynamics (MDL) simulation of PVT data for Ar and experiment at 700 K

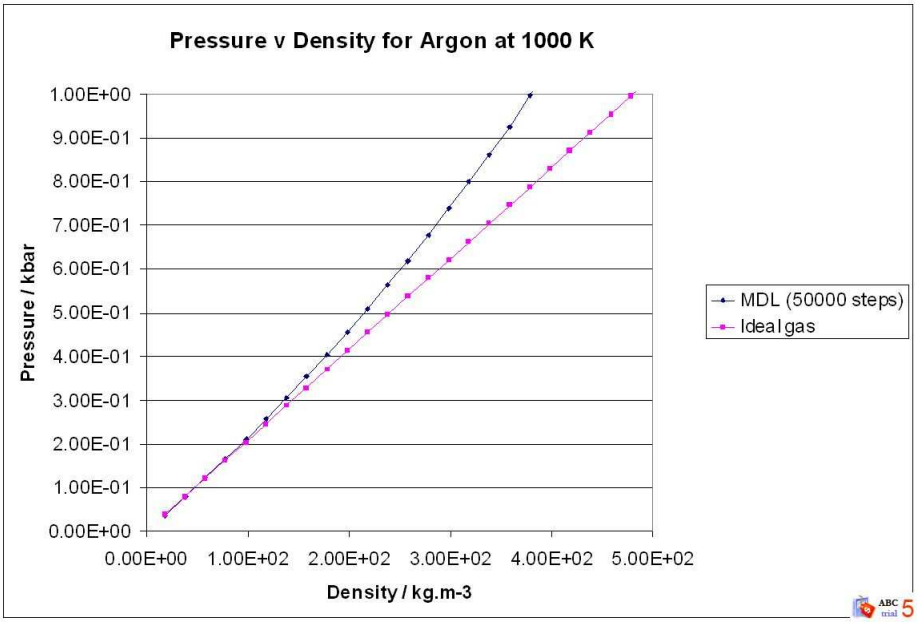


Figure 6.11: Molecular dynamics (MDL) simulation of PVT data for Ar at 1000 K

Figures 6.9-6.11 show simulation results for argon at 500, 700 and 1000 K respectively. In all figures the ideal gas line is shown. Experimental data from [15] were not available at 1000 K. In figures 6.9 and 6.10 where comparison with experimental data was possible the simulation results are good.

In all cases, up to densities of about 500 kg.m^{-3}, the agreement between simulation and experiment is good. This indicates that we are likely to have a reliable basis for the assessment of parameters for MTDATA's real gas model. This assessment process is described later in this chapter.

6.3.2 CO_2 - PVT results

The presentation of simulation results for carbon dioxide follows a similar format to the above. Figures 6.12 and 6.13 show the results of the isotherm at 500 K in wide and lower pressure-density regimes. In both cases the agreement of the simulations with experiment is good. At this relatively low temperature ($T_c = 304.128$ K) the simulation results start to depart from experiment at about a density of 300 kg.m^{-3}.

A confirmation of the validity of the molecular dynamics simulations for carbon dioxide is shown in Figure 6.6. This shows the second virial coefficient as a function of temperature. The molecular dynamics simulations are shown as large solid squares at 500 and 1000 K. Other values shown on figure Figure 6.6 are the precise measurements of Dadson et al [22] as triangles and diamonds. The small squares result from an MTDATA assessment of the definitive evaluation of Span and Wagner [16]. Finally the solid line is from Dymond and Smiths compilation of virial coefficients [19].

Figures 6.14 and 6.15 show the results of the isotherm at 1000 K in wide and lower pressure-density regimes. The experimental data are shown as solid circles and the molecular dynamics simulation results as solid triangles. Note older MD results ("CO2 test 2") are included to show the very poor results when using an incorrect intermolecular potential. Also the "CO2 test 2" results are noteworthy as the simulations were performed using only 5000 averaging steps. This small number of averaging steps results in a very apparent irregularity to the pressure-density isotherm. Finally the solid line (no symbols) shows the results of MTDATA calculations using a fixed value of the Murnaghan A parameter. The agreement between the experimental data and the "MDL - CO2 test 3" molecular dynamics simulations is very good up to a density of about 300 kg.m^{-3}.

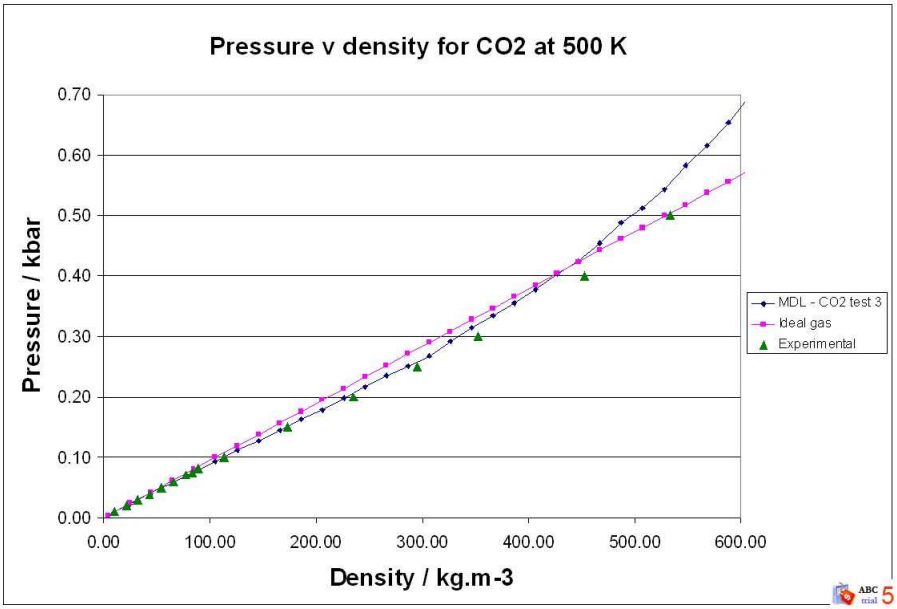


Figure 6.12: Comparison of molecular dynamics (MDL) simulation of PVT data for CO_2 and experiment at 500 K

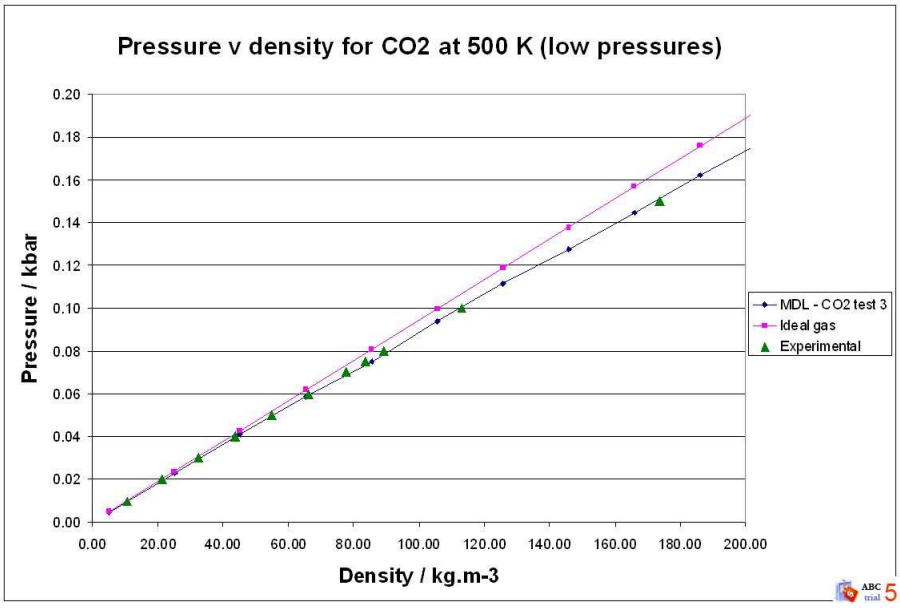


Figure 6.13: Comparison (low-pressures) of molecular dynamics (MDL) simulation of PVT data for CO_2 and experiment at 500 K

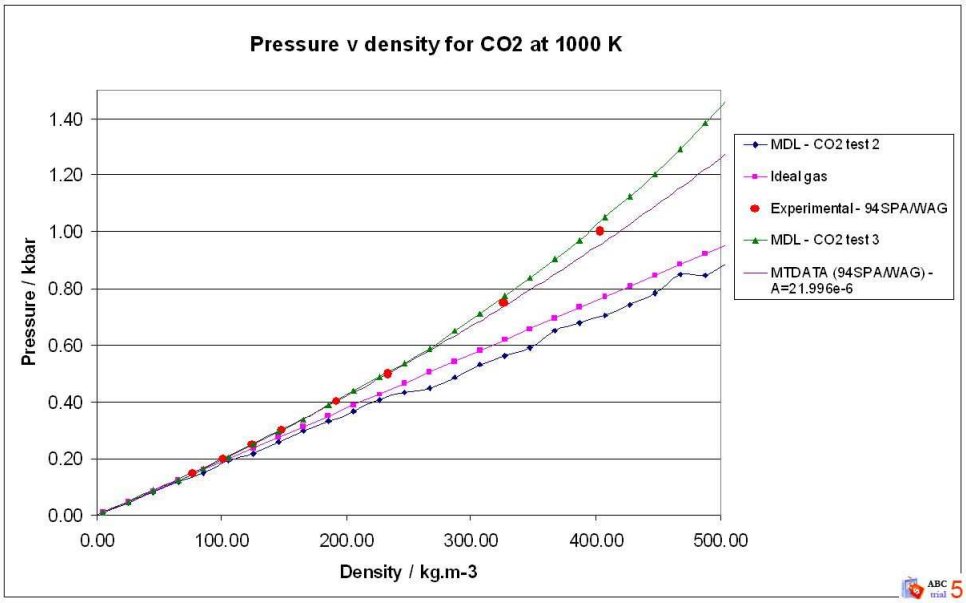


Figure 6.14: Comparison of molecular dynamics (MDL) simulation of PVT data for CO_2 and experiment at 1000 K

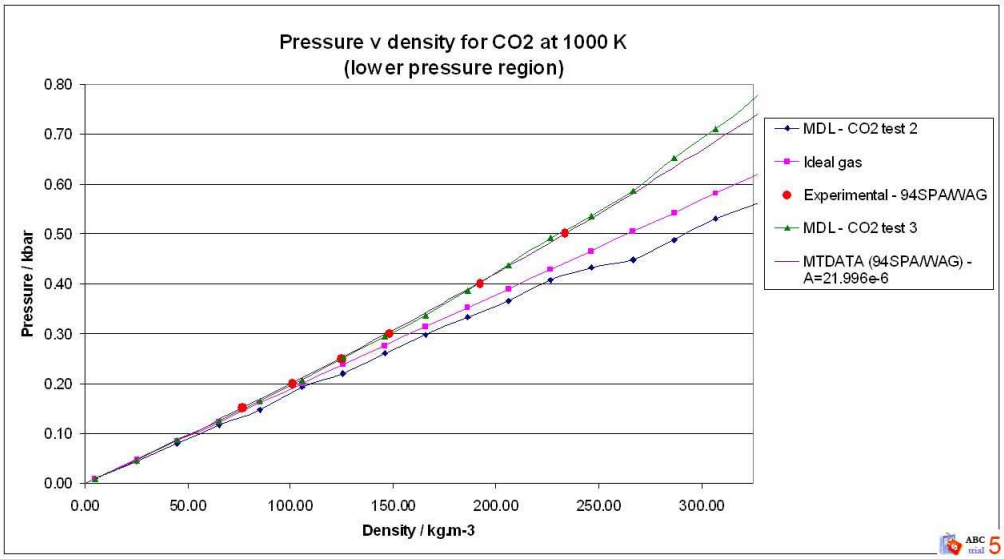


Figure 6.15: Comparison (low pressures) of molecular dynamics (MDL) simulation of PVT data for CO_2 and experiment at 1000 K

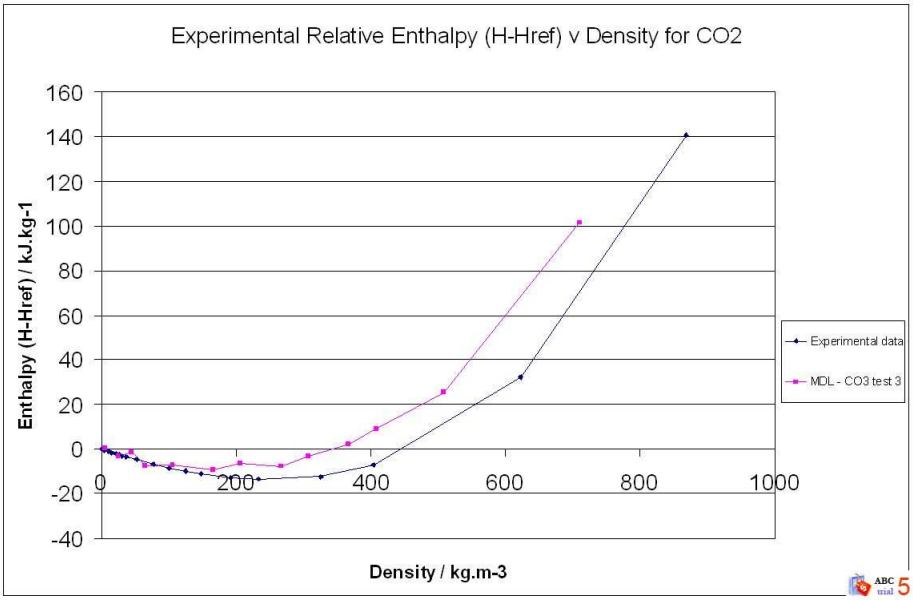


Figure 6.16: MDL simulated Enthalpy -density isotherms for CO_2 at 1000 K compared with experimental values

6.3.3 CO_2 - enthalpy results

The results of molecular dynamics simulations also contain information about the enthalpy or internal energy of the real gas. Figure 6.16 shows a comparison of the experimental data at 1000 K, evaluated by Span and Wagner [16] and our molecular dynamics results. Overall the results are promising and show clearly the movement from attractive to repulsive regimes.

6.4 Practical analysis and application of results using MTDATA

6.4.1 Analysis

This section describes the practical analysis (assessment) of PVT data using MTDATA [18]. The model for the real gas used by MTDATA was a real gas modification of the Murnaghan model [23].

$$G_{pres} = \frac{A\exp(a_0T + a_1T^2/2 + a_2T^3/3 + a_3T^{-1})}{(K_0 + K_1T + K_2T^2)^{n-1}}\left[(1 + nP(K_0 + K_1T + K_2T^2))^{1-1/n} - 1\right]$$

In the above expression for the correction to the Gibbs energy for the real gas the A, a_n, n and K_n parameters could potentially be optimised.

MTDATA's Assessment module contains a comprehensive system for the non-linear optimisation of the parameters based on an objective function created from the differences between experimental and calculated property values. For example, in Figure 6.5, the long-dashed line shows the results of a calculation based on a single parameter (A) optimised using the saturation vapour density of argon. This shows the flexibility of the above function as the normal expectation would be that to cover a range of temperature both A and a_3 parameters would need to be used. This was done for argon using the experimental saturated vapour density and presented in the MTDATA Assessment module output graph shown in Figure 6.17. Note, Figure 6.17 is presented as a pressure-vapour density plot, whereas Figure 6.5 is pressure-temperature plot.

Figures 6.18-6.22 present the MTDATA analysis of experimental PVT isotherms from Span and Wagner [16] using various G_{pres} parameterisation schemes. Figure 6.18 shows the isotherm at 500 K

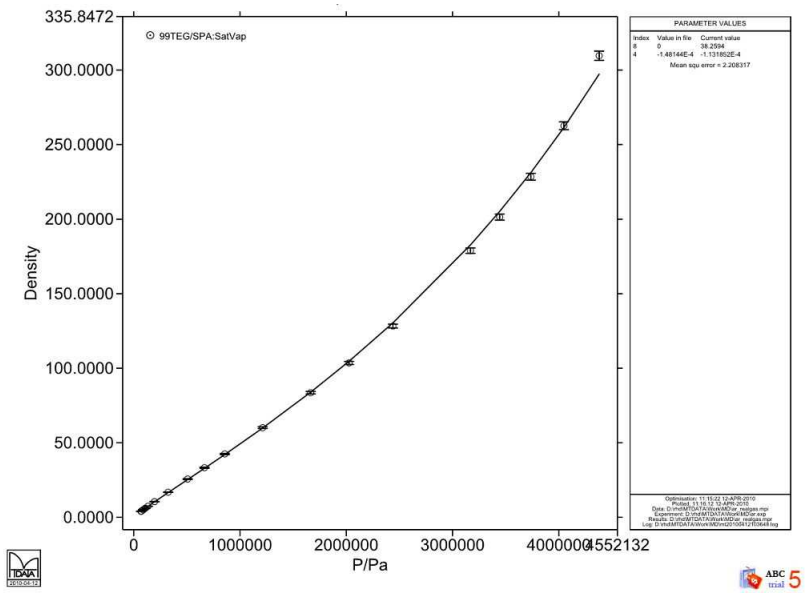


Figure 6.17: MDL simulated gas density - on liquid-vapour saturation line for Ar compared with two parameter (A and a_3) model using MTDATA

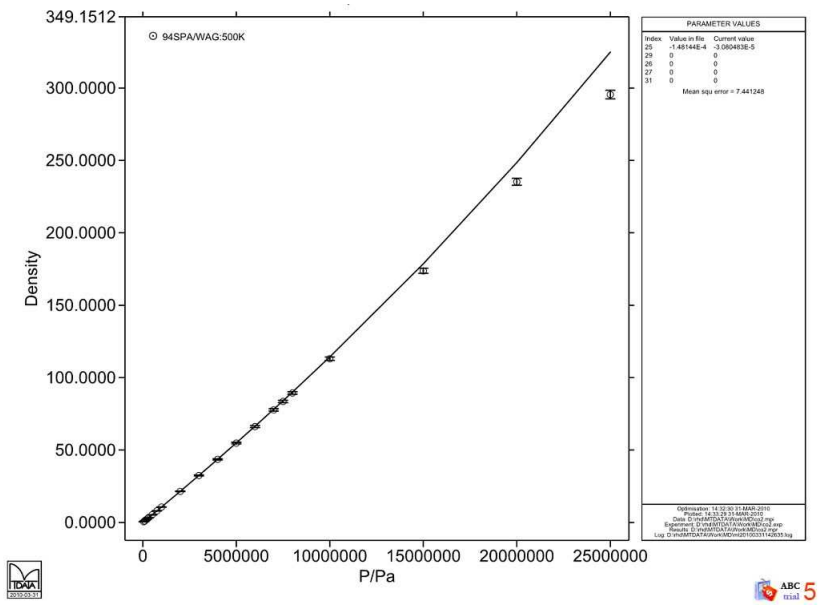


Figure 6.18: Experimental density - pressure isotherm for CO_2 at 500 K modelled, using MTDATA with fixed Murnagham A term only

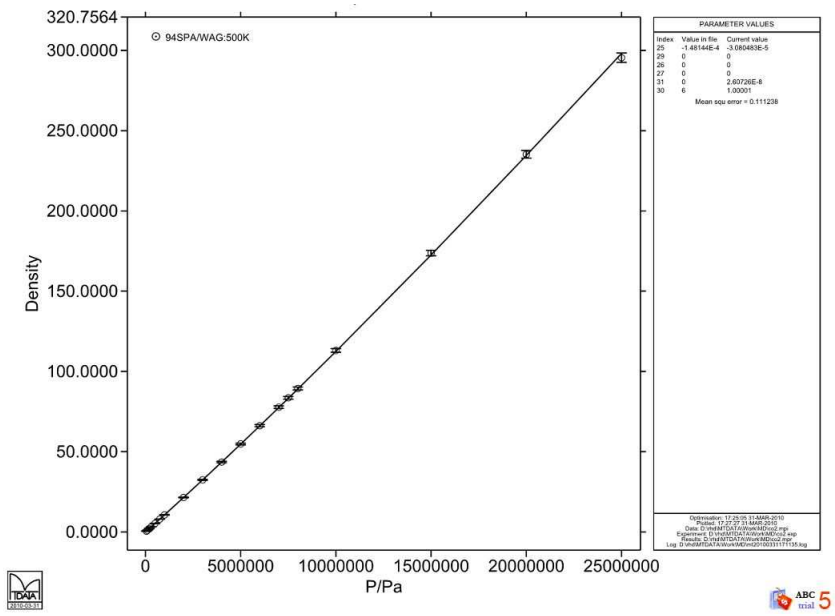


Figure 6.19: Experimental density - pressure isotherm for CO_2 at 500 K modelled, using MTDATA with assessed K_0 only (A and n fixed)

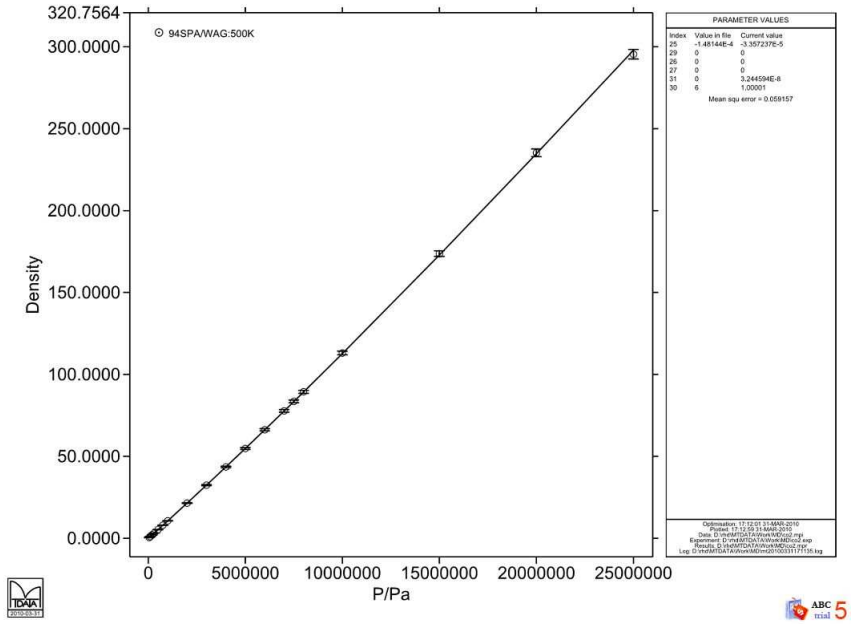


Figure 6.20: Experimental density - pressure isotherm for CO_2 at 500 K modelled, using MTDATA with assessed K_0, A and n parameters

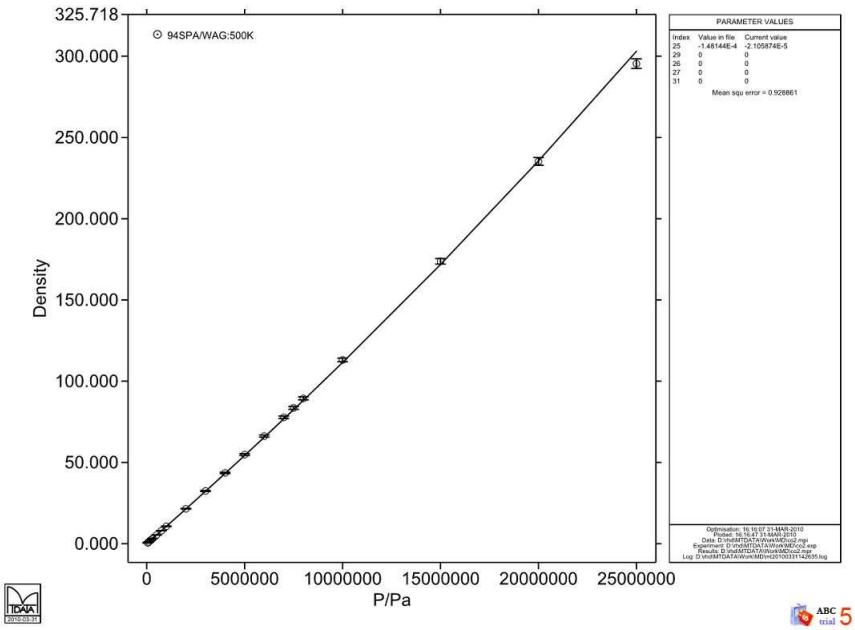


Figure 6.21: Experimental density - pressure isotherm for CO_2 at 500 K modelled, using MTDATA with assessed A parameter only

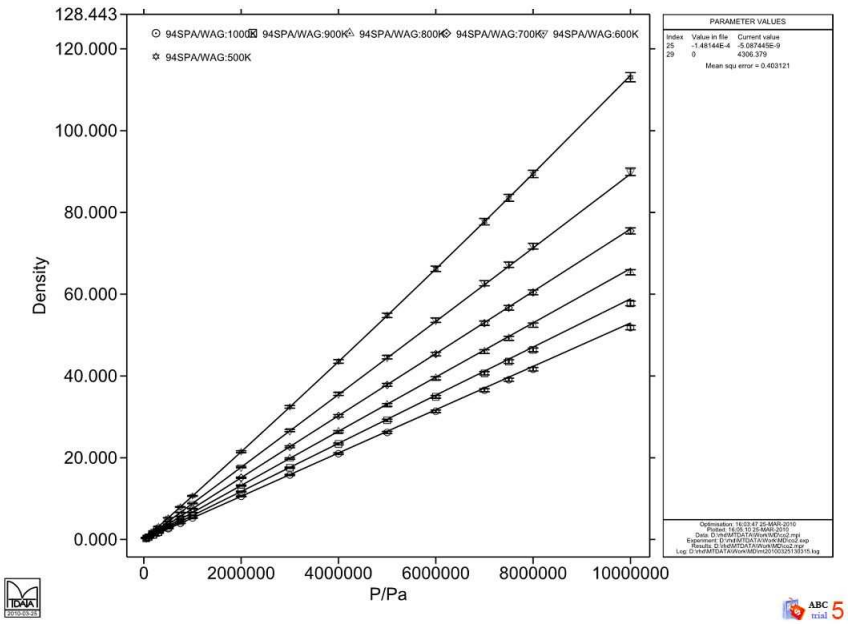


Figure 6.22: Experimental density - pressure isotherms for CO_2 at 500 -1000 K modelled, using MTDATA with assessed A and a_3 parameters (Low pressures $P < 10$ MPa)

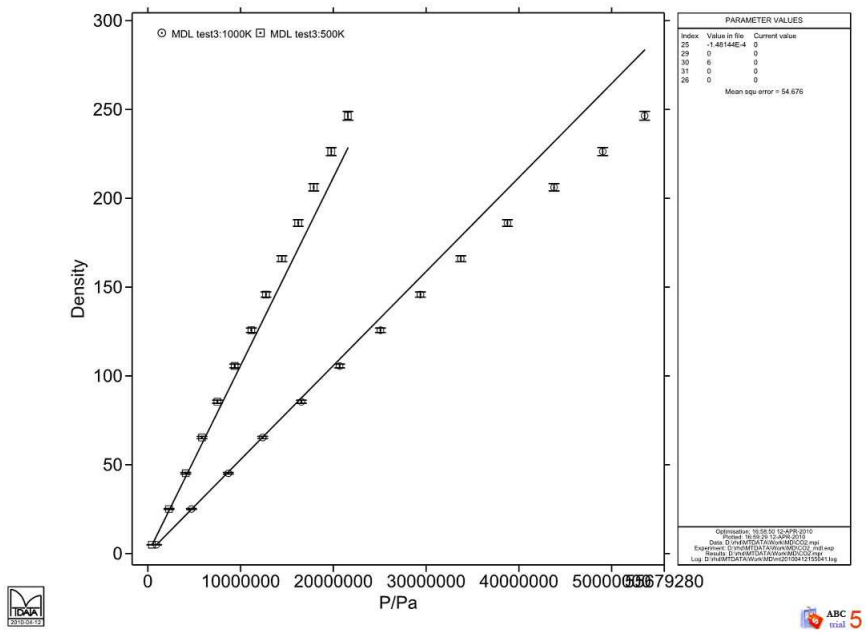


Figure 6.23: MDL simulated density - pressure isotherms for CO_2 at 500 and 1000 K compared with MTDATA ideal gas model

with only an A parameter in the real gas model based on an optimisation to lower pressure data. The optimisation of A to low-pressure data should theoretically produce A parameters equivalent to the second virial coefficients - this is confirmed by the results presented in Figure 6.6. Figure 6.19 presents the results of an assessment where the A parameter value is fixed to that found using lower pressure data but now with the K_0 parameter being optimised. Note the n parameter is typically set to a constant in the Murnaghan model (eg 5.0 or 6.0). For the real gas adaptation of the model a value of $n = 1.0001$ was used. The difference between figures 6.18 and 6.19 essentially shows the effectiveness of using optimised K_0 parameter. This isothermal 3-parameter parameterisation scheme results in effectively a one-parameter model as n is fixed and A determined from limiting low-pressure deviations from the idea gas behaviour. Allowing all 3 parameters to be optimised shows a slightly better representation of the experimental data and is shown in Figure 6.20. Figure 6.21 shows results that explore further the use of a one-parameter (A) real gas model. Here the results are from an assessment where A has been optimised to data over the full pressure range (not just the more theoretically justifiable low pressure range). Compared to Figure 6.18 there appear to be some benefits, however the theoretical limiting behaviour of the model at low pressure is lost. Finally Figure 6.22 presents low-pressure results using a 2-parameter temperature depended model (A and a_3 parameters were optimised). The results show a very good representation of the carbon dioxide PVT properties over the ranges $1000 > T/K > 300$, $120 >$ density/kg.m$^{-3} > 0$ and $10 >$ P/MPa > 0.

The above discussion has related to the assessment of real gas model parameters for use in MTDATA from an analysis of evaluated experimental data. The following now focuses on the assessment of MTDATA real gas model parameters from molecular dynamics results.

Figure 6.23 shows, at 500 and 1000 K, the molecular dynamics PVT results compared with ideal gas model calculations from MTDATA. This figure shows data over the full pressure and density range (P / MPa < 500 and density/kg.m$^{-3} < 250$). The simplest parameterisation for this data set is to optimise A and a_3 to lower pressures (P / MPa < 250) data - the results of this are shown in Figure 6.24. The results complete multi-parameter (A, a_0, a_3, n and K_0) optimisation is shown in Figure 6.25.

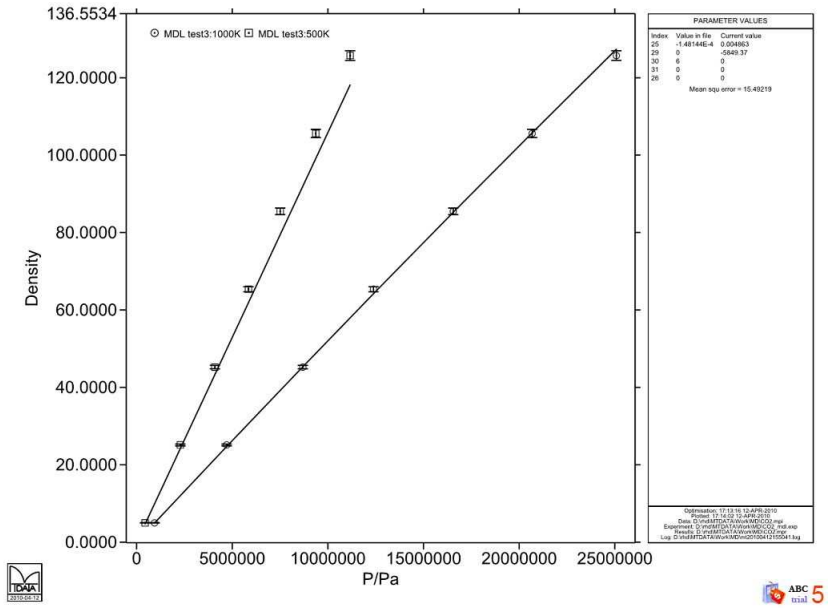


Figure 6.24: MDL simulated density - low pressure isotherms for CO_2 at 500 and 1000 K compared with two parameter (A and a_3) model using MTDATA

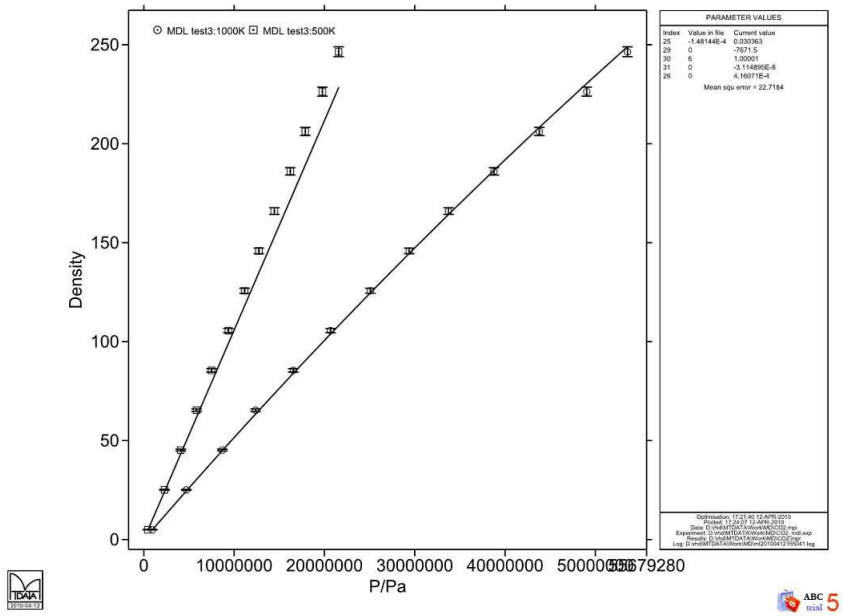


Figure 6.25: MDL simulated density -pressure isotherms for CO_2 at 500 and 1000 K compared with multi-parameter (A, a_0, a_3, n and K_0) model using MTDATA

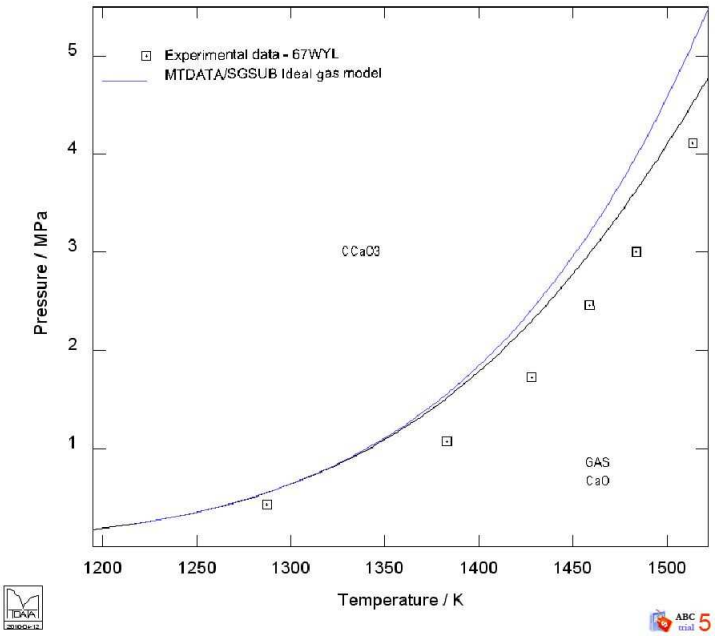


Figure 6.26: Calculated decomposition pressures, using ideal and real gas models, of $CaCO_3$ compared with experiment

6.4.2 Application

One application of improved gas modelling is related to the decomposition (or formation) of carbonate minerals. The following example relates to the CaO-CO_2 system and in particular the decomposition of $CaCO_3$.

Figure 6.26 shows the calculated $CaCO_3$ decomposition pressures as a function of temperature with experimental data from Wyllie [24]. The solid lines show the pressures calculated using MTDATA with either an ideal gas model or real gas model using the parameterisation presented in Figure 6.25. MTDATA calculated lines use version 11.0 of the SGTE Substance Database for the solid phases CaO and $CaCO_3$. There is a clear benefit in adopting the more realistic model of the gas phase even at the rather modest pressure of 5 MPa.

The significant deviation (error of ca. 30 K) between the experimental values and ideal gas based predictions over the whole temperature range could be possibly be caused by errors in thermodynamic data for CaO and/or $CaCO_3$. Having a robust and reliable model for the non-ideal gas phase makes it much easier to identify the other sources of error. In fact adjusting the enthalpy of formation of $CaCO_3$ by -2.0 kJ.mol^{-1} yields the results shown in Figure 6.27. The new value for $\Delta_f H^0$ (298.15 K) is -1208.60 kJ.mol^{-1}, this may be compared with the original SGTE value of -1206.60 kJ.mol^{-1} and the value presented by Garvin et al [25] of -1207.59 kJ.mol^{-1}.

6.5 Conclusions

The main conclusions that may be made from the work presented in this chapter are:

- Molecular dynamics can provide a useful source of simulated PVT data for simple molecules
- These data are quantitatively in agreement with the best experimental data up to densities of about 300 to 500 kg.m^{-3}
- The simulation of enthalpy data up to a density of about 250 kg.m^{-3} shows promise

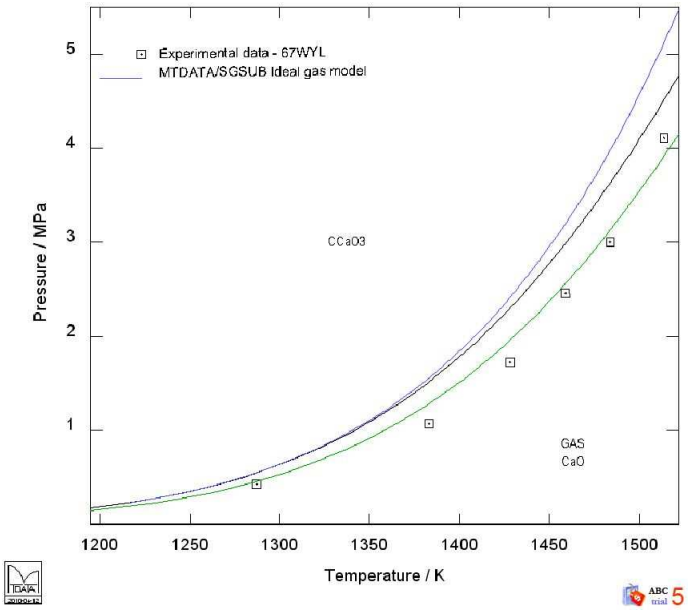


Figure 6.27: Calculated decomposition pressures, using ideal and real gas models, of $CaCO_3$ compared with experiment, data for CaCO3 corrected: $\Delta_f H$ offset by -2.0 kJ.mol-1

- It is possible to parameterise simple non-ideal gas models for use in MTDATA using data directly from molecular dynamics simulations
- Derived second virial coefficients are in very good agreement with evaluated and experimental data
- Calculations using MTDATA with non-ideal gas models produce useful results in themselves and assist in determined sources of error in non-gaseous data
- A reliable strategy for analysing PVT data has been established that retains continuity with limiting behaviour (second virial coefficient) as the ideal gas limit is approached.

Additional conclusions specifically related to the use of molecular dynamics are:

- Use the largest number of averaging steps possible
- Always compare results with ideal gas predictions for a reality check of the appropriateness of the inter-molecular potential being used
- The calculation of the second virial coefficient from output PVT results provides a useful check
- No special computational resources are required for simple molecules and up to 50000 averaging steps

Chapter 7

Binary alloys

In Chapter 4, we stressed the importance of being able to calculate chemical potentials. Many solvers, including DL_POLY, do not support this calculation. We were obliged to add extra code for the chemical potential calculation into MDL. TOWHEE, however, has chemical potentials built in. In this chapter, we put chemical potentials to good use in the calculation of binary alloys. It should be noted, of course, that MDL and DL_POLY would allow the calculation of binary alloys, or, indeed, any arbitrary set of atoms. However, it is only through chemical potentials that we can distinguish in a thermodynamic sense between the behaviour of different components in order to look for $T - x$ phase diagrams and other phenomena which relate to the *difference* between the behaviour of species. In this chapter, we will consider the Pb/Cu binary alloy which is the theoretical example discussed in Chapter 3. The techniques we use are, of course, equally valid for other binary alloys and could be extended to multiple species problems via the discussion of Section 3.7. We will use TOWHEE and MDL driven by the NSIM driver program.

The interactions between Cu and Pb were difficult to obtain. Of course, both species exhibit metallic bonding, which, strictly, cannot be modelled using pair potentials. Rather, metallic interactions are usually modelled using the Embedded Atom Method (EAM). In this method, rather than adding up pair potentials between atoms, each atom contributes a set amount of charge onto other atoms. This interacts with the atom's own charge giving an attractive potential. Repulsive potentials are calculated as pair potentials in the usual way giving the total system potential energy as

$$E(r_i) = \sum_i F\left[\sum_{j\neq i}\rho(r_{ij})\right] + \frac{1}{2}\sum_{i,j}\Phi(r_{ij}) \tag{7.1}$$

where F is the embedding, ρ are the atomic charge densities (specific to charges being considered) and Φ are the repulsive pair potentials. This is, of course, different from the pair potential paradigm used in normal MD/MC. However, if the embedding function F is close to linear, the energy can be written in an approximate pair potential format.

One of the first papers to discuss the EAM is [26]. Embedding functions and the repulsion functions $\Phi(r)$ were given but charge distributions were not specified preventing the reader from reconstructing the total energy (7.1). However, TOWHEE supports the EAM system and even has a tutorial example for Pb/Cu mixture based on the EAM with a citation given to [26]. Unfortunately, when we ran this example, it did not work correctly and showed "divide by zero" errors. We were able to salvage the enthalpy variation with the atomic fraction x from the output files but the enthalpy change from $x = 0$ to $x = 0.5$ appeared to be 100 times greater than expected from experiment. We analysed the TOWHEE Cu/Pb input file and discovered that the effective pair potentials were as given in Figure 7.1 (the Towhee embedding function does agree with [26]). Comparing this to the effective pair potential published by Hoyt [27] shown in Figure 7.2, shows that the TOWHEE potential well depth for Cu is 10 times greater than in the Hoyt paper. The well depth in Figure 7.2 are much more realistic in terms of the experimental mixing critical temperature (1280 K) and the mixing excess enthalpy (6800 J/mol), see Figure 3.20 and

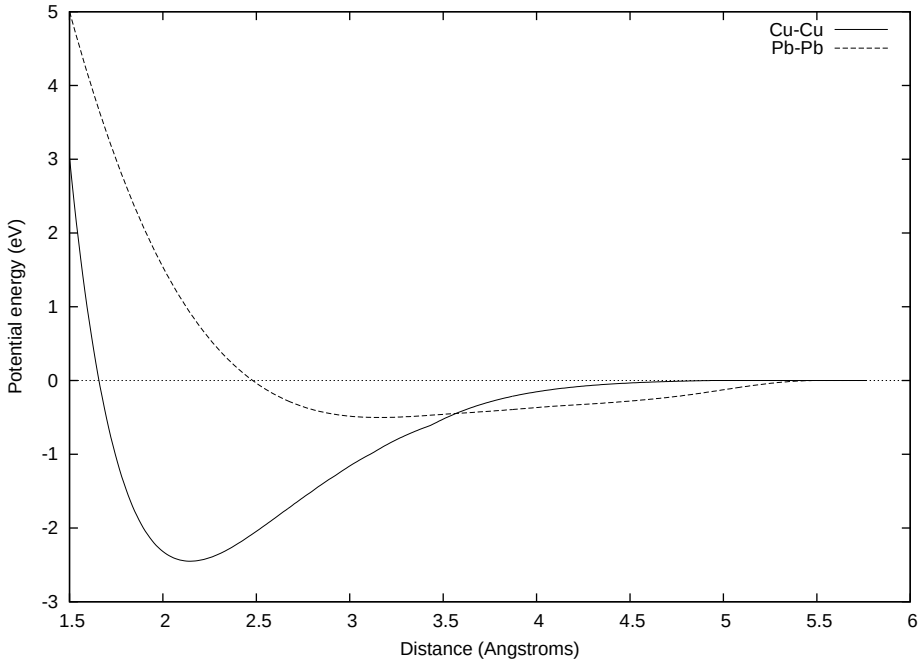


Figure 7.1: Effective pair potentials of Cu-Cu and Pb-Pb as derived from [26] and the TOWHEE example manual

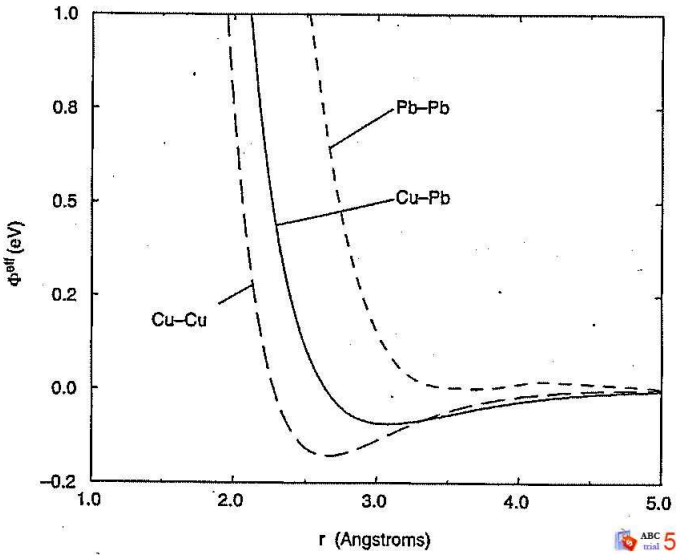


Figure 7.2: Effective pair potentials from the Hoyt paper, [27]

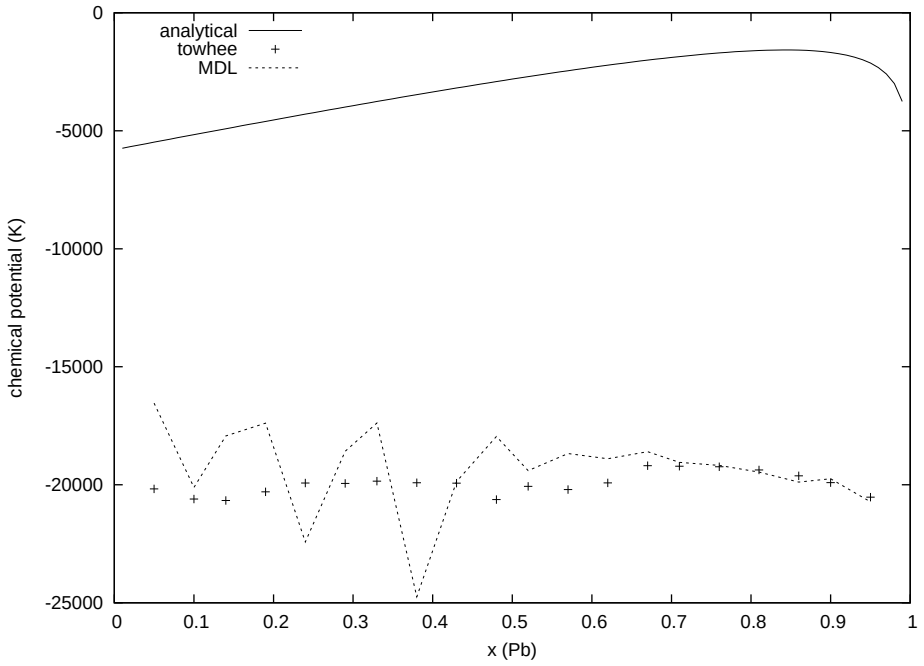


Figure 7.3: μ_{cu} calculated analytically and using MDL and TOWHEE

Figure 3.21. The divide by zero errors and multiple warnings in the TOWHEE output file are more serious still, and, in the end we were forced to conclude some error exists in TOWHEE's EAM implementation. We have therefore used a simple Lennard Jones pair potential for Cu-Cu, Pb-Pb and Cu-Pb in both the TOWHEE and MDL simulations. The potentials were taken from the Hoyt paper, [27], and were adjusted so that, according to the analytical theory of Section 3.8.1, they give a $T - x$ critical temperature of around 1560 K and an excess enthalpy of 0.07 eV/atom (6800 J/mol) just as were used in the analytical discussion of Section 3.8.1. This T_c is rather too high but we cannot (analytically) fit both enthalpy and the critical point (see (3.90), (3.91)) so "went" with the enthalpy. We also considered the $T - \rho$ (melting) critical point of copper and lead though these values are not well known. The final choice was:

Interaction	ϵ (eV)	σ (Å)
Cu-Cu	0.18	2.5
Cu-Pb	0.07	3.0
Pb-Pb	0.05	3.5

Of course, both atoms have zero charge. Note that the Cu-Pb interaction is actually stronger than the Pb-Pb interaction. Nevertheless, Cu and Pb will tend to separate at low temperature owing to the huge Cu-Cu attraction.

Figure 7.3 shows $\mu_{Cu}(x)$ calculated using MDL and TOWHEE and compared to the analytical theory of Section 3.8.1. The MDL and TOWHEE results are in reasonable agreement and differ completely from the analytical result which shows how approximate the latter is. Both simulation codes showed a fair amount of random variation which did not seem to improve by running the simulation longer. The MDL result is more noisy as might be expected from the extremely inaccurate MDL chemical potential results shown in Chapter 4.

Figure 7.4 shows $\mu_{Pb}(x)$. Again there is good agreement between the two simulation codes which differ by a constant from the analytical result. It should be noted that the $\mu_{Pb}(x)$ function is much smoother and "better behaved" than the $\mu_{Cu}(x)$ result which may be due to the much weaker Pb-Pb

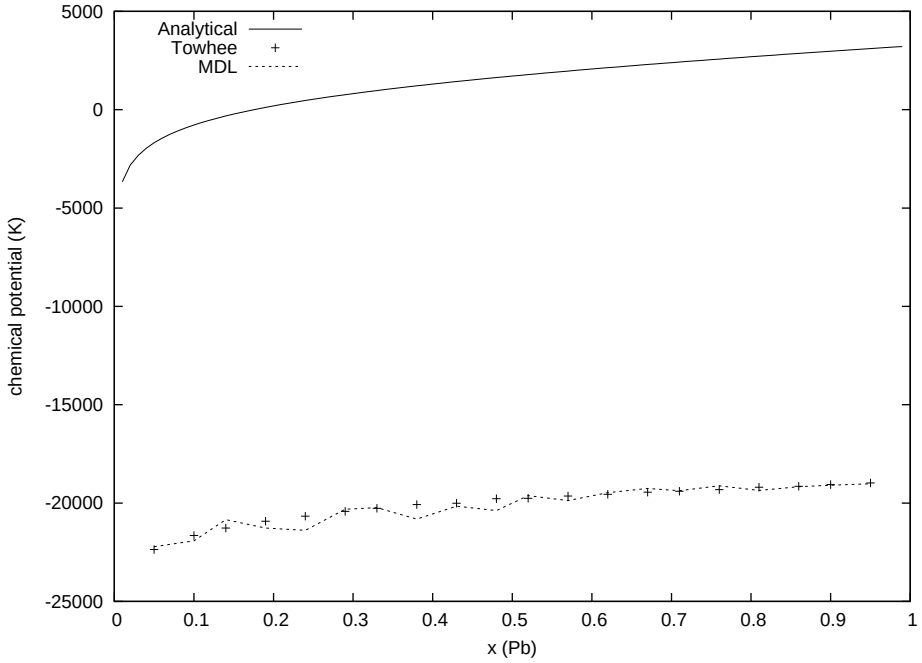


Figure 7.4: μ_{pb} calculated analytically and using MDL and TOWHEE

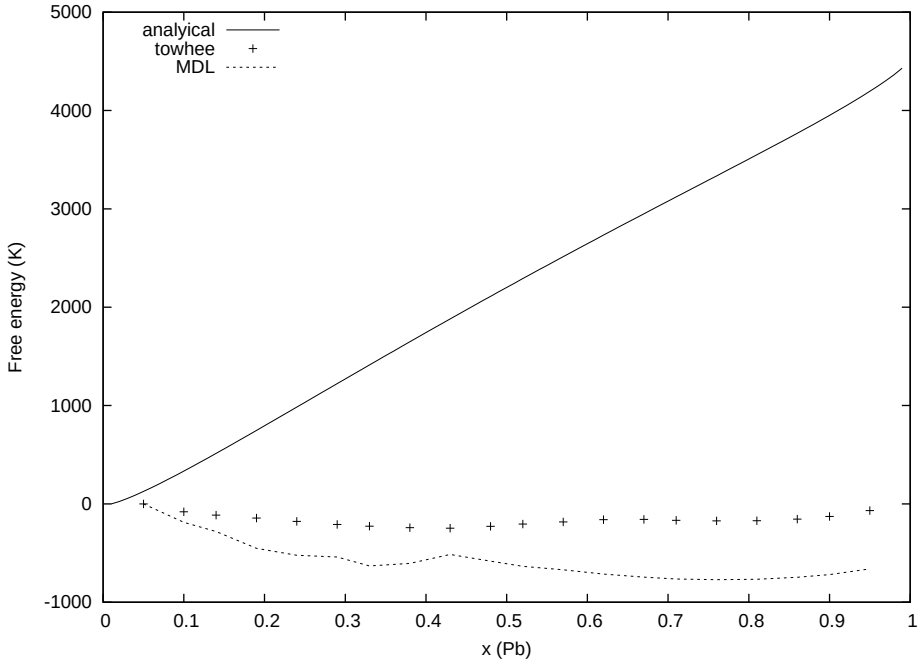


Figure 7.5: Free energy for Cu/Pb alloy at 1200 K

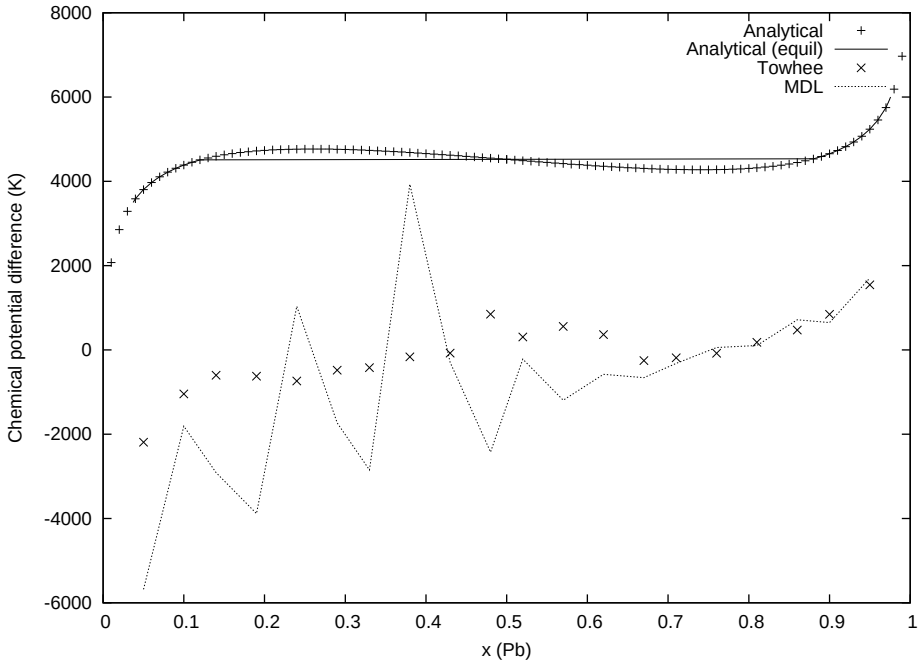


Figure 7.6: $\mu_{pb} - \mu_{cu}$ calculated analytically and using MDL and TOWHEE

interaction. Figure 7.6 shows the chemical potential difference which is unfortunately rather a noisy graph mostly due to the $\mu_{Cu}(x)$ function. However, the free energy, Figure 7.5, still shows a convincing miscibility gap and, as such, NSIM was able to find a phase transition. This is shown again in Figure 7.7 with other temperature runs building up to a $\Delta\mu - x$ phase diagram. The quality of this diagram might be improved by increasing the number of atoms. The corresponding $T - x$ phase diagram is in Figure 7.8 showing a critical temperature of 1500 K at $x_c = 0.4$. These results are in reasonable agreement with experiment but clearly there is room to improve the statistics. MDL does reasonably well and seems to perform better than in the single-species case, perhaps because the number of atoms remains constant.

The enthalpy gap turned out to be difficult to calculate accurately. The absolute enthalpy is shown in Figure 7.9 where the straight line shows the enthalpy for unmixed Cu and Pb in corresponding amounts. In Figure 7.10 we have subtracted the straight line to give the mixing enthalpy as in Section 3.8.1. The maximum enthalpy of mixing is 3000 "kJ/mol" which corresponds to 0.26 eV/atom, considerably greater than the experimental enthalpy of mixing 0.07 eV/atom. Figure 7.10 is clearly not reliable and it seems that longer runs or more atoms are needed in order to get accurate figures for enthalpy. The reader should refer back to Figure 3.20 and Figure 3.21 for experimental comparison.

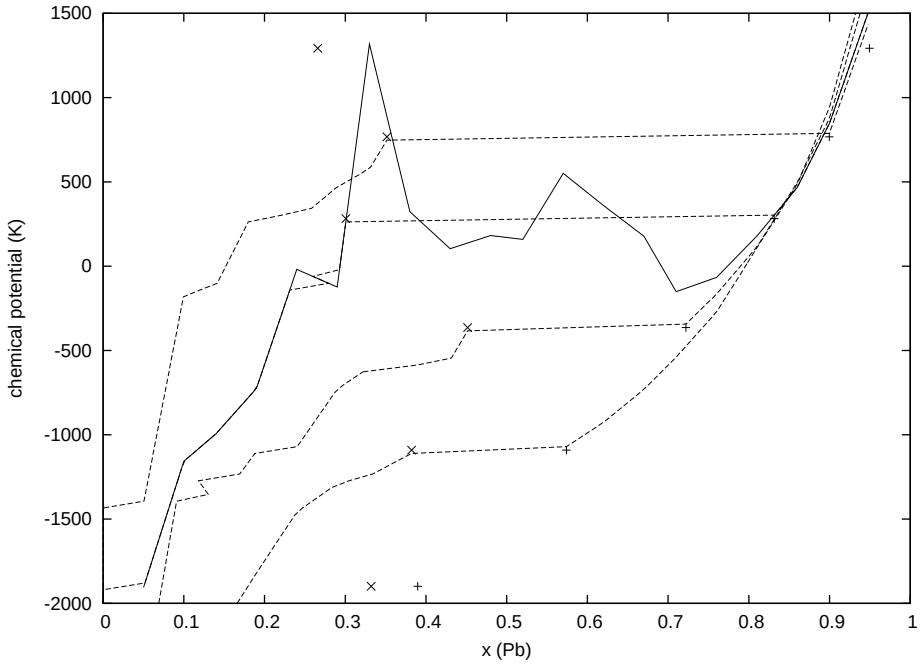


Figure 7.7: Flat $\Delta\mu - x$ curves at various temperatures calculated using TOWHEE. The raw MC curve for 1200 K is shown solid. Symbols show computed phase transition points

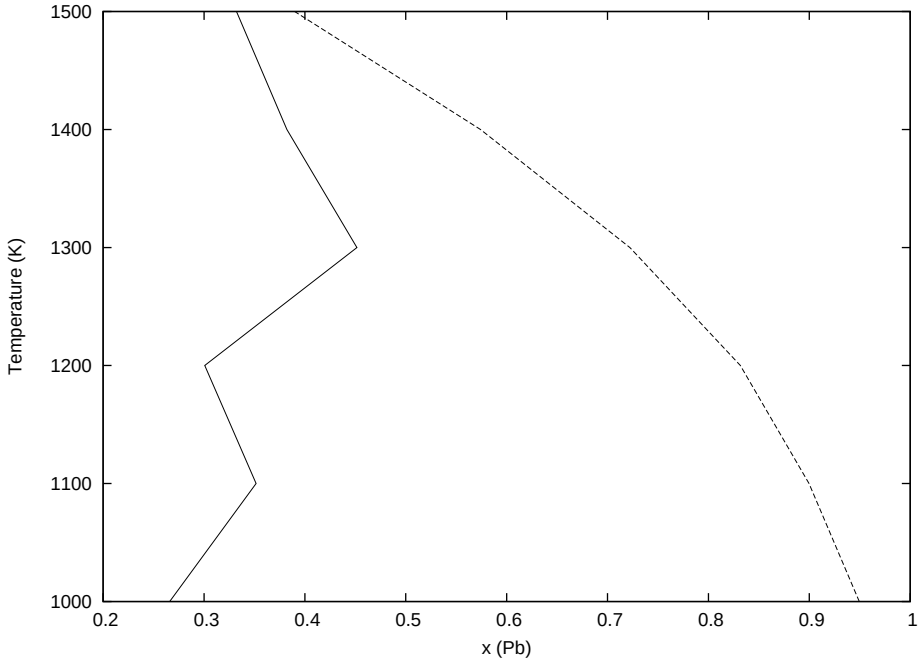


Figure 7.8: $T - x$ phase diagram calculated by TOWHEE for Cu-Pb

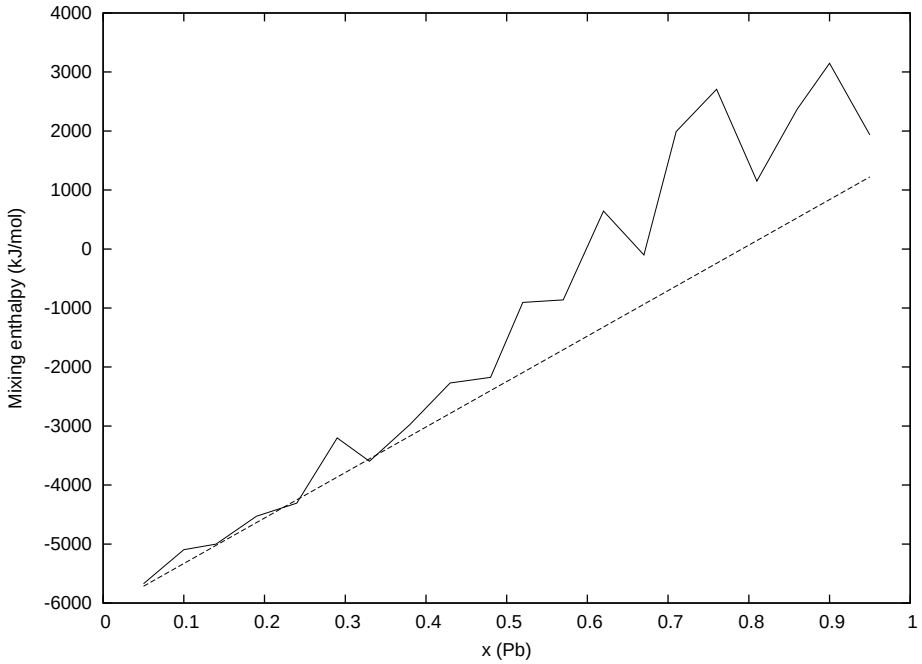


Figure 7.9: Cu/Pb mixture enthalpy as calculated by TOWHEEfor 100 atoms. Straight line shows enthalpy for equal quantities of unmixed Cu and Pb

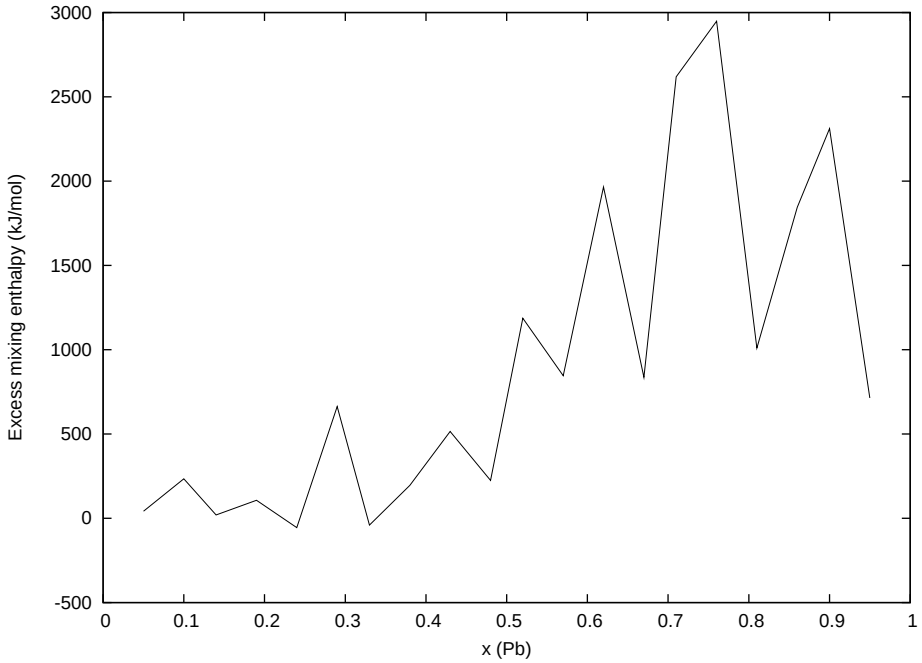


Figure 7.10: Excess Cu/Pb enthalpy of mixing calculated by TOWHEEfor 100 atoms

Chapter 8

Landau-Devonshire and effective Hamiltonian theory for ferroelectrics

8.1 Introduction

This chapter describes approaches to the modelling of ferroelectric materials (barium titanate in particular) and draws comparison with experimental results. Firstly, a description of thermodynamic phenomenological methods is given including the Landau-Devonshire theory. This is followed by a review of first principles computational modelling techniques. Modelling results are presented based on effective Hamiltonian theory using the program FERAM. Simulation results for temperature and strain coupling are then compared with experiment. The chapter can be compared to Chapter 9, in which direct MD simulation is used to model barium titanate.

8.2 Thermodynamic Theory

8.2.1 Landau-Devonshire Theory

In 1949 Devonshire published a paper describing a thermodynamic theory for the ferroelectric behaviour of Barium Titanate [28]. It is often described as a phenomenological theory i.e. it describes the relationships between various macroscopic quantities without any reference to the underlying microscopic mechanisms. Devonshire's theory is based on simple thermodynamic and symmetry arguments but is nonetheless able to explain a wide range of ferroelectric phenomena using a simple set of parameters. The theory is closely related to theories of phase transitions developed by Landau is therefore sometimes referred to as Landau-Devonshire (LD) theory.

In this theory it is assumed that the free energy of the crystal in the absence of applied stress (i.e. zero elastic energy) can be written as a Taylor expansion in the polarisation (the order parameter):

$$\begin{aligned} G =& \alpha_1(P_x^2 + P_y^2 + P_z^2) + \\ & \alpha_{11}(P_x^4 + P_y^4 + P_z^4) + \alpha_{12}(P_x^2P_y^2 + P_y^2P_z^2 + P_z^2P_x^2) + \\ & \alpha_{111}(P_x^6 + P_y^6 + P_z^6)...etc. \end{aligned} \tag{8.1}$$

where the free energy zero is equated to the free energy at $P_x = P_y = P_z = 0$. This expansion is based on a cubic prototype phase i.e. all directions in the crystal are treated equally, although we shall see that the free energy minima can occur off-centre, thus breaking the symmetry under certain conditions. This symmetry of the prototype phase dictates that only terms of even order can be included. The expansion is usually truncated arbitrarily when enough terms have been included to account for the observed behaviour. For example, basic ferroelectric behaviour with a second order phase transition from

a paraelectric phase to a ferroelectric one can be described satisfactorily by including only terms up to the fourth power in the polarisation. This truncation is based on this phenomenological approach rather than an approximation to a convergent Taylor expansion. In Devonshire's paper the expansion was limited to the terms shown in 8.1. This is sufficient to give satisfactory account of the basic ferroelectric properties and phase changes in Barium Titanate, although expansions to higher order have been published [29].

An important part of the theory is that the coefficient of the quadratic term is temperature dependent, with the following dependency:

$$\alpha_1 = \frac{T - T_0}{2\epsilon_0 C} \tag{8.2}$$

where C is the Curie constant (around 3.6×10^{5}°C for Barium Titanate), T_0 is the Curie-Weiss temperature (around 383K for Barium Titanate) and ϵ_0 is the permittivity of free space. The ferroelectric behaviour and phase transitions can be replicated assuming the remaining coefficients are independent of temperature, but more sophisticated theories incorporate some temperature dependence in these terms [30, 31]. For a first order phase transition (which is generally thought to be the case for Barium Titanate) α_{11} is negative and α_{12} and α_{111} are positive.

To find the equilibrium state of the crystal we need to calculate the minima of the free energy function (equation 8.1). To visualise the contributions to the free energy and the positions of the minima Figure 8.1 shows the free energy terms in a conceptual 2D system as a function of P_x and P_y. It can be seen that some of the terms (e.g. 8.1(c)) tend to promote minima along the x, y axes whilst others (e.g. 8.1(b)) tend to promote minima along the diagonals. The polar directions within the crystal therefore depend on the temperature and the coefficients of the free energy expansion. It is this dependency which allows the theory to model not only the paraelectric to ferroelectric phase transition but also ferroelectric to ferroelectric transitions associated with changes in the crystal symmetry.

This can all be done mathematically by minimising the free energy function with respect to the polarisation by setting the derivatives of the free energy (with respect to each of the polarisation components) equal to zero. Note that the derivative of the free energy with respect to P_z is equal to the electric field, E_z (and similar relations for all three components):

$$E_z = \frac{\partial G}{\partial P_z} = 2\alpha_1 P_z + 4\alpha_{11} P_z^3 + 2\alpha_{12} P_z (P_x^2 + P_y^2) + 6\alpha_{111} P_z^5 \tag{8.3}$$

Setting the electric field components to zero therefore provides the following equations for the polarisation at the free energy minima:

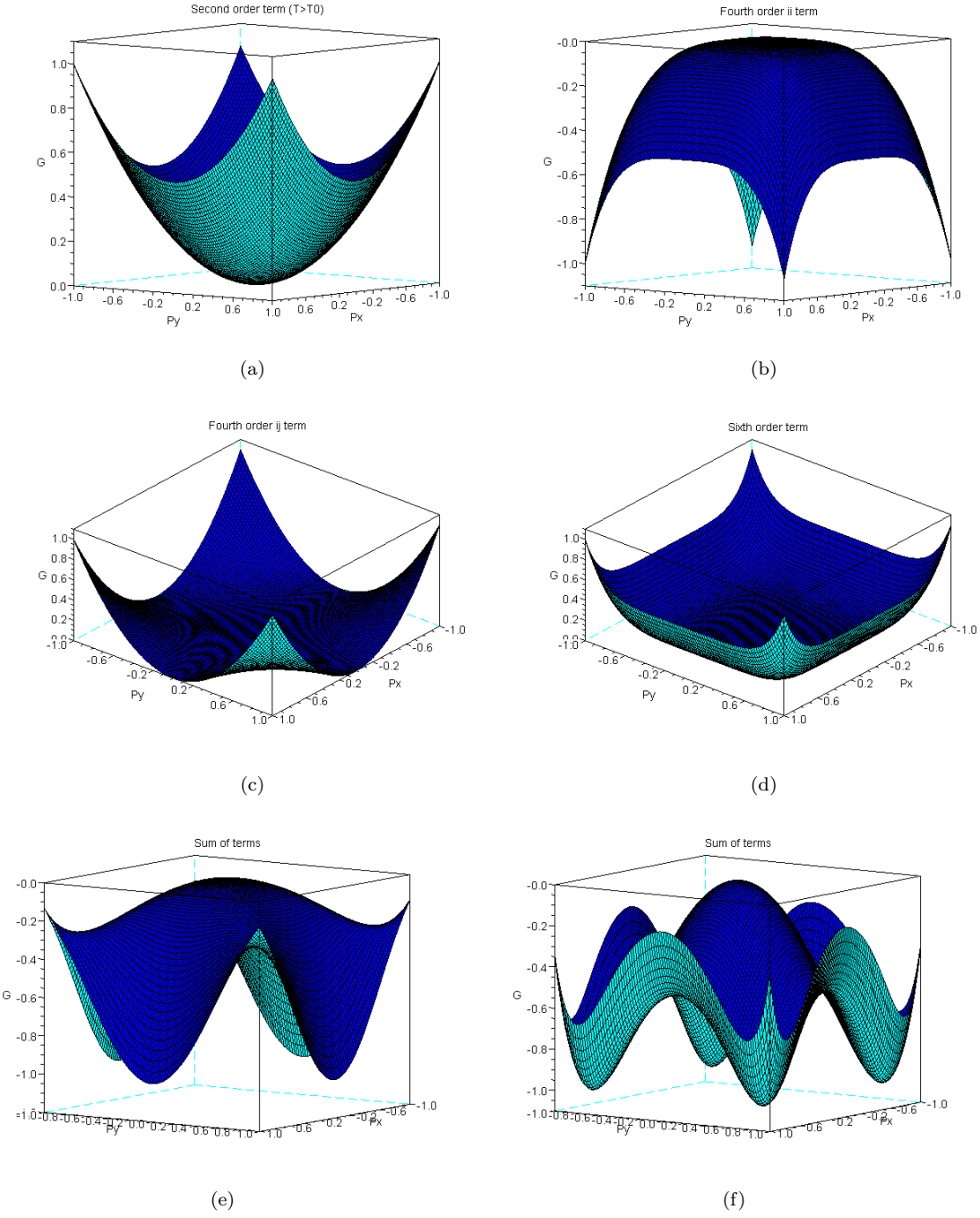
$$\begin{aligned} P_x = 0 \;\text{ or }\; & 2\alpha_1 + 4\alpha_{11} P_x^2 + 2\alpha_{12}(P_y^2 + P_z^2) + 6\alpha_{111} P_x^4 = 0 \\ P_y = 0 \;\text{ or }\; & 2\alpha_1 + 4\alpha_{11} P_y^2 + 2\alpha_{12}(P_z^2 + P_x^2) + 6\alpha_{111} P_y^4 = 0 \\ P_z = 0 \;\text{ or }\; & 2\alpha_1 + 4\alpha_{11} P_z^2 + 2\alpha_{12}(P_x^2 + P_y^2) + 6\alpha_{111} P_z^4 = 0 \end{aligned} \tag{8.4}$$

This provides four sets of solutions which may correspond to free energy minima (with equivalent solutions also obtained by permuting x, y, z):

$$\begin{aligned} P_x = P_y = P_z = 0 & \\ P_x = P_y = 0 \;\;;\;\; & 2\alpha_1 + 4\alpha_{11} P_z^2 + 6\alpha_{111} P_z^4 = 0 \\ P_x = 0 \;\;;\;\; & P_y = P_z \;;\; 2\alpha_1 + (4\alpha_{11} + 2\alpha_{12}) P_z^2 + 6\alpha_{111} P_z^4 = 0 \\ P_x = P_y = P_z \;\;;\;\; & 2\alpha_1 + 4(\alpha_{11} + \alpha_{12}) P_z^2 + 6\alpha_{111} P_z^4 = 0 \end{aligned} \tag{8.5}$$

Equations 8.5 are readily solved for the components of P corresponding to possible minima. These can then be substituted into equation 8.1 to obtain corresponding expressions for the free energy at these minima:

$$\begin{aligned} G =& 0 \\ G =& \alpha_1 P_z^2 + \alpha_{11} P_z^4 + \alpha_{111} P_z^6 \\ G =& 2\alpha_1 P_z^2 + (2\alpha_{11} + \alpha_{12}) P_z^4 + 2\alpha_{111} P_z^6 \\ G =& 3\alpha_1 P_z^2 + 3(\alpha_{11} P_z^4 + \alpha_{12}) P_z^4 + 3\alpha_{111} P_z^6 \end{aligned} \tag{8.6}$$



(a) (b)

(c) (d)

(e) (f)

Figure 8.1: Free energy plots in 2D showing the contributions from a) the second order term ($T > T_c$), b) the fourth order ii term, c) the fourth order ij term, d) the sixth order term, e), f) the sum of the terms for two different parameter sets. All units are arbitrary.

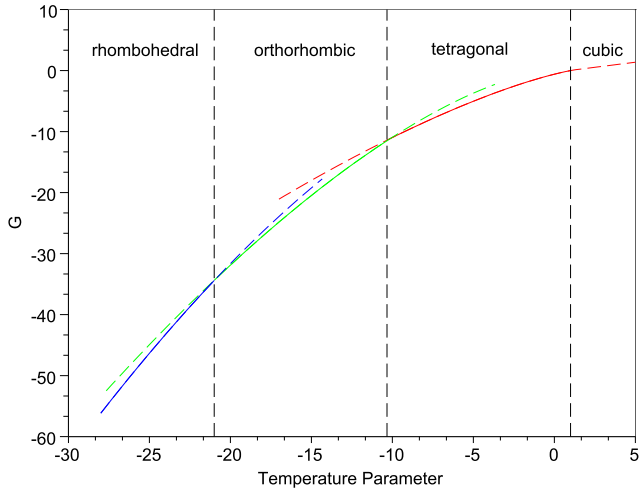


Figure 8.2: Free energy solutions from Equation 8.6 (arbitrary units)

The resulting relations for the free energy are plotted in Figure 8.2. As the temperature is lowered the lines corresponding to the solutions in Equation 8.6 cross. This means that the particular solution for the energy minimum changes at these crossing points resulting in discontinuities in the slope of the free energy minimum. These discontinuities are also associated with a change in the alignment of the polar axis and therefore of the crystal symmetry according to the labels on Figure 8.2. In these plots the values of the expansion parameters used in the Devonshire paper [28] were used. These were obtained by comparison with experimental values of permittivity, polarisation and transition temperatures.

8.2.2 PE Loops and Coercive Field

The Landau Devonshire theory can also be used to model PE loops and to obtain estimates of the coercive field. For simplicity we consider only the z component of the field (from Equation 8.3) in the tetragonal phase i.e. $P_x = P_y = 0$.

$$E = \frac{\partial G}{\partial P} = 2\alpha_1 P + 4\alpha_{11} P^3 + 6\alpha_{111} P^5 \tag{8.7}$$

where $E = E_z$ and $P = P_z$. This function is plotted in Figure 8.4 The value of E passes through a minimum and then a maximum as the polarisation increases. In an experiment where the electric field is the controlled parameter it can only increase beyond these extrema by a discontinuous jump in the polarisation. These extrema in E therefore correspond to the coercive field E_c. The polarisation values corresponding to E_c can be calculated by differentiating Equation 8.7 with respect to P and setting the result to zero:

$$\frac{\partial E}{\partial P} = 2\alpha_1 + 12\alpha_{11} P^2 + 30\alpha_{111} P^4 = 0 \tag{8.8}$$

Equation 8.8 can be solved for P_c^2 where P_c is the polarisation at Ec. Note that this is slightly different to the experimental definition of the coercive field which is the field at $P = 0$. This is because the value of $E = 0$ for $P = 0$ from Equation 8.7 corresponds to an unstable state which is not the coercive field. The tie line for the discontinuity in Figure 8.4 passes through $P = 0$ at the same value of E as the extremum in E. The solution for P_c^2 is (taking only positive roots):

$$P_c = \sqrt{\frac{-12\alpha_{11} + \sqrt{(12\alpha_{11})^2 - 240\alpha_1\alpha_{111}}}{60\alpha_{111}}} \tag{8.9}$$

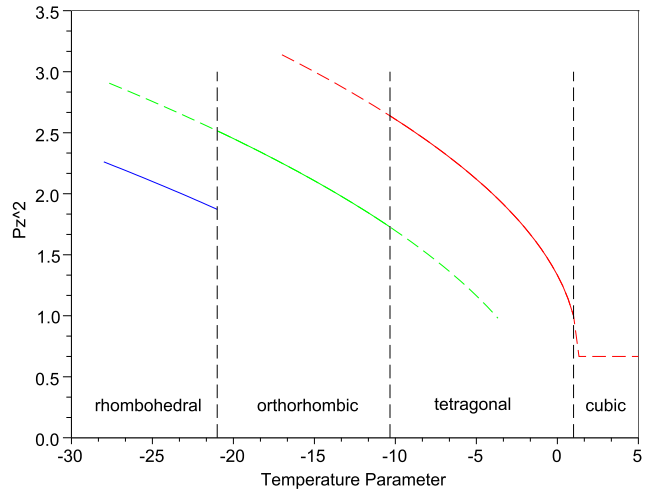


Figure 8.3: Polarisation (P_z) solutions of Equation8.5 (arbitrary units)

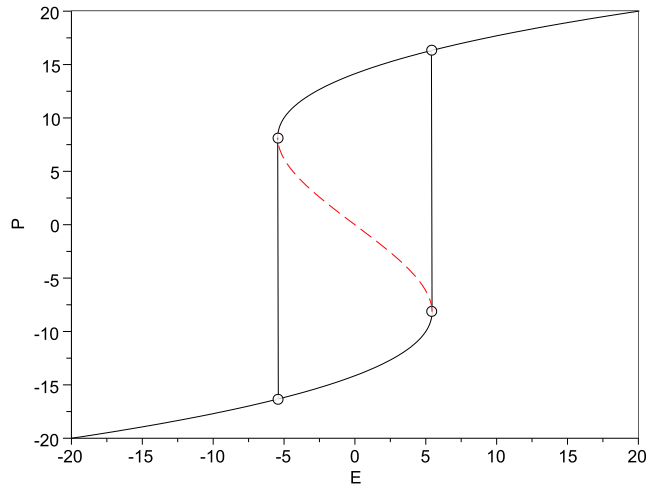


Figure 8.4: PE loop from Equation 8.7 (arbitrary units)

and the coercive field is given by:

$$E_c = 2\alpha_1 P_c + 4\alpha_{11} P_c^3 + 6\alpha_{111} P_c^5 \tag{8.10}$$

Various different values of the expansion coefficients are available in the literature. Values from Bell & Cross [31] and from Ahluwalia & Srolovitz [32] were used to calculate the coercive field shown in Figure 8.5(a). These coefficients are tabulated in Table 8.1

Table 8.1: Landau coefficients

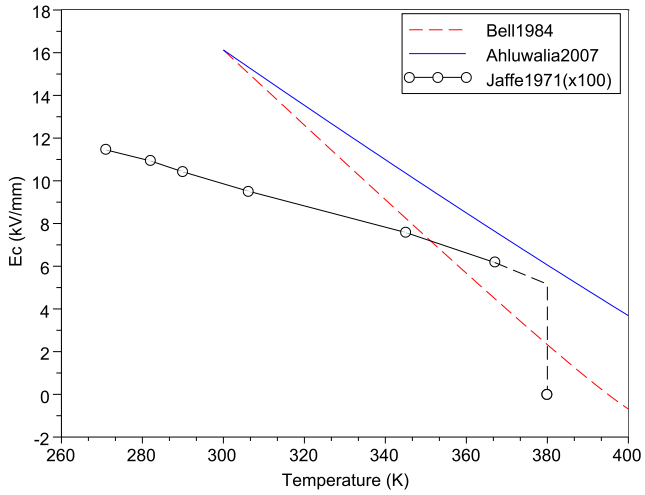
Coeff	Unit	Bell [31]	Ahluwalia2007 [32]
α_1	$C^{-2}m^2N$	$3.34 \times 10^5(T - 381)$	$3.34 \times 10^5(T - 381)$
α_{11}	$C^{-4}m^6N$	$4.69 \times 10^6(T - 393) - 2.02 \times 10^8$	-6.381×10^8
α_{111}	$C^{-6}m^{10}N$	$-5.52 \times 10^7(T - 393) + 2.76 \times 10^9$	7.89×10^9

Note that with these units the free energy is expressed in J m^{-3}. It is apparent from Figure 8.5(a) that there are significant discrepancies in the coercive field between implementations of the theories. Although only one set of experimental results are shown here it is also likely that there are significant variations in the experimental data. It is well known that the coercive field is highly dependent on the frequency of the applied field, and the quality of the crystal. Despite these variations it is quite clear that there is a major discrepancy (2 orders of magnitude) between values of E_c from theory and experiment (the remanent polarisation shown in Figure 8.5(b) fares rather better). This discrepancy is referred to as the Landauer [33] paradox. The reason for this is is as follows. Even a very small applied electric field creates a difference in energy between the state with the polarisation aligned to the electric field and that opposed to it i.e. the anti-aligned state is located at a local minimum and is therefore metastable. At some value of the electric field the energy barrier separating the two states disappears. The anti-aligned state becomes unstable and the system switches discontinuously to the aligned state. This is the value of the coercive field in the Landau-Devonshire theory. In practice the switching process occurs by nucleation and growth of domains. This is thermally activated. Microscopic thermal fluctuations can cause small regions of the crystal to spontaneously switch polarisation. This means that switching can occur at much lower values of electric field than that predicted by Landa-Devonshire theory. The theory represents a limiting case in the absence of domain processes.

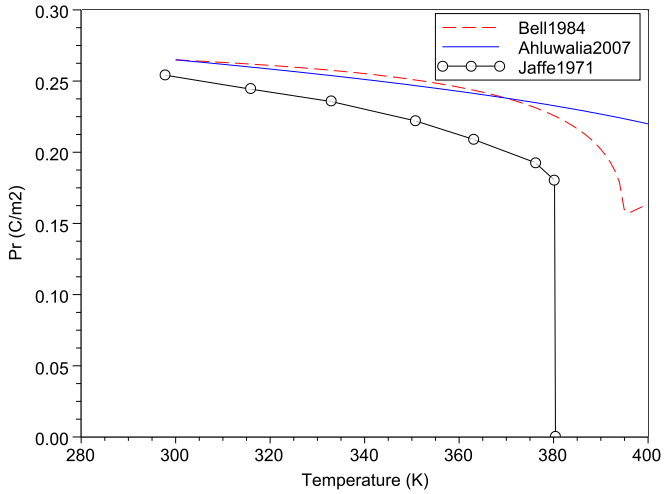
8.3 First Principles Modelling

Ferroelectric materials are of extremely widespread technological interest for their switchable polarisation (memory), high permittivity (capacitors), piezoelectric and pyroelectric properties (sensors and actuators). Although these materials have been commercially available for over half a century, new forms of ferroelectrics are creating huge new commercial opportunities in worldwide markets such as computer memory and electronics, magnetic sensing and microfluidics[34]. They are also of considerable scientific interest because their complex behaviour provides stringent tests of modern physics theory.

The reversible spontaneous polarisation that typifies a ferroelectric material is the result of a fine balance between long range dipole forces promoting polar ordering and short range forces promoting symmetrical non-polar structures. The energy differences required to tip the balance one way or the other are extremely small compared to the total energy of the crystal structure. Calculation to sufficient accuracy to provide a meaningful description of ferroelectricity places very high demands on the accuracy of the calculations. Ferroelectric materials are often deliberately designed to be compositionally close to a transition in material properties such as the morphotropic phase boundary (MPB) in the Lead Zirconate Titanate (PZT) system. This results in an unstable crystal structure which enhances the ferroelectric properties. The effectiveness of materials such as relaxor ferroelectrics is inherently based on disorder in the structure [35]. A characteristic of ferroelectric materials is the ordering of the polarisation into domains. This ordering is brought about by long range electrostatic forces, so common methods



(a)



(b)

Figure 8.5: Coercive field (a) and remanent polarisation (b) from Landau-Devonshire theory compared to experiment. Note that experimental values for E_c are multiplied by 100 in plot (a).

to simplify the problem by only considering nearest neighbour interactions or neglecting long range forces omit some of the essential physics of these materials. Many ferroelectric compositions include transition metals and heavy elements such as Lead or Barium. This leads to considerable hybridisation and co-valency in the bonding. The distinction between core and valence electrons is blurred and a large set of electrons need to be dealt with explicitly, and consideration given to the polarisability of the core shells [36]. Having solved these problems there remains one of interpretation of the results. The calculated electron density cannot be simply interpreted as a polarisation [37]. Electrical and mechanical boundary conditions can play a significant role especially in thin films and nano-structured materials. The ferroelectric material must be considered in relation to its environment, size, substrate and electrodes [38].

Strong theoretical interest developed in the 1940's with the widespread technological applications made possible by the discovery of the ferroelectric properties of Barium Titanate, and later by the development of PZT [39]. Technological interest in ferroelectric materials continues to develop rapidly with recent developments in thin films [40] and nano-structured materials [41, 42] providing new challenges for the theoretical description of ferrolectric materials. The introduction of magnetic properties into these materials either at a unit cell level or by creating composites, and the resulting multiferroic properties are attracting significant interest for a range of possible new applications [43, 44]. In principle all the structure and properties of a ferroelectric material can be calculated by solving the many body Schrodinger equation for all the electrons and ions in the material. The size of the computational task for all but the simplest molecules is much too large for present or envisaged computational power. Much of the effort and ingenuity of the research in this area is in finding suitable approximations that preserve the essential physics and provide reasonable quantitative prediction of material and device properties. This section provides an overview of the theoretical approaches adopted in modern first principles (ab initio) work and briefly surveys recent developments. In this short section we cannot describe in detail the full status of the subject and the interested reader is referred to fuller recent reviews [45, 38].

8.3.1 Electronic structure and total energy

The standard method for solid state calculations is to assume a potential that is periodic in the lattice symmetry so that only a relatively small number of particles need to be considered. Calculations are made of the electronic structure and total energy for a given range of ionic configurations, assuming fixed ion positions (Born-Oppenheimer approximation). The ground state structure is obtained by energy minimisation with respect to changes in the ion positions. It is usual to discriminate between core electrons which are not considered separately, but combined with the ionic potential to form an effective potential in which the remaining (valence) electrons move. This distinction is complicated in the case of typical ferroelectric perovskite oxides by the high polarisability of some of the heavy ions such as Pb and Ba. This means that the electronic structure calculations must include 'semi-core' electrons, and the potential must be constructed carefully to produce accurate calculations [36]. Commonly used are ultra-soft pseudopotentials developed by Vanderbilt [46]. The softness refers to the ability of the potential to replicate the true potential in close proximity to the ion core. This is important where electrons are localised in the vicinity of the ions. Most modern calculations employ density functional theory (DFT) using the local density approximation (LDA) [36] or generalized gradient approximation (GGA) [47] to solve the many-electron Schrodinger equation for the ground state (zero temperature, zero external field). This approach can be problematic for ferroelectrics where the ferroelectric ordering and domain formation is determined by long range dipole interactions, and thermal fluctuations, and we are usually interested in the effects of an externally applied electric field. This field may be rather high and is certainly high enough to cause structural changes in the material. Many first principles calculations are therefore limited to zero external electrical field or small perturbations about zero. Recent work includes strain and electric field as a perturbation (density-functional perturbation theory DFPT [48]) even extending the theory to finite electrical fields [49].

8.3.2 The Effective Hamiltonian Method

The electronic structure calculation provides data on the ground state (zero temperature) properties of small cells (typically up to 100 atoms) subject to periodic boundary conditions. To extend the theory to finite and inhomogeneous crystal structures at above zero temperatures, the method of the effective Hamiltonian has been developed [50, 51]. The electronic structure calculation yields the total cohesive energy of the crystal as a function of ionic co-ordinates i.e. a total energy surface in co-ordinate space. This total energy surface is used to calculate potential energies and forces associated with distortions of the lattice [36]. Of particular interest are the soft mode distortions i.e. the static polar distortions of the lattice observed below the Curie temperature which are responsible for the ferroelectric (and all "ferroic"-like) behaviour. This potential is then used to calculate lattice dynamic properties using molecular dynamic or Monte Carlo techniques simulating thousands of unit cells [45]. These methods are extremely powerful and provide calculation of finite temperature effects, phase change and piezoelectric response. This type of approach has been used to obtain significant improvements in the calculations for more complex systems such as PZT [52] or for study of temperature dependence, phase transitions and surface effects [53]. This inclusion of long range dipolar forces and dynamic co-operative interactions within the material has made possible the calculation of domain structures. Theoretical work in this area has predicted the existence of novel toroidal domain structures which have been confirmed experimentally [54]. In the area of nano-ferroelectrics the interaction of this type of theoretical work with the challenging experimental investigations is crucial to develop our understanding of this new class of ferroelectric material.

8.3.3 Molecular Dynamics Method

A complementary approach to complex systems is to use molecular dynamics based on first principles calculations of the interaction potentials [55, 56]. Because this method includes all degrees of freedom it provides good results in materials such as $PbTiO_3$ where the rotational and ferroelectric distortions need to be considered explicitly. In molecular dynamics, each atom is connected to a well-defined set of other atoms via bonds, generally simulated as springs, but which may be anharmonic. The atoms vibrate and may spread out (thus exhibiting strain) but retain the same connectivity. In piezoelectric and ferroelectric simulations the key observables are stress, strain and polarisation. The effect of an applied electric field is also important. The simulations show spontaneous domain formation due to strain compatibility between adjacent unit cells. Thus, we will observe hysteresis effects: when the system is all in one domain it takes a lot of applied field to reverse domain directions (the system remembers its previous state). Polarisation-electric field and strain-electric field diagrams are obtained and can be compared with experimental findings. Domain structures and ferroelastic properties in mineral systems such as cordierite [57, 58, 59] have been studied successfully by molecular modelling, which could be extended to charged systems, such as piezoelectrics.

8.3.4 Polarisation

In most technological applications of ferroelectrics we are interested in the interaction of the material with an electric field, and the resultant movement of charges i.e. changes in the macroscopic polarisation of the material. Unfortunately the calculation of macroscopic polarisation from microscopic structure is not straightforward. This is not simply a matter of computational difficulty. One might think that it's simply a matter of integrating the charge distribution (known from the electronic wavefunction) over the unit cell:

$$\mathbf{P} \;=\; V^{-1} \int_V \mathbf{r}\rho(\mathbf{r}) d^3\mathbf{r} \tag{8.11}$$

However, this is approach is incorrect in principle [37] As Vanderbilt and King-Smith point out the polarisation "can be made to take on any value by a sufficiently pathological choice of unit cell" [60]. In quantum mechanics most observable quantities are related to the expectation value of an operator, but this is not the case for the polarisation. To resolve this difficulty, the phase of the wavefunction must be considered using a 'Berry Phase' formulation [61, 60, 62]. The wavefunctions and energies are calculated

as above, then polarisation changes are calculated from the phase of the resulting wavefunction. This theory provides a complete quantum-mechanical description of polarisation and is used in most modern calculations of ferroelectric properties.

8.3.5 Future Trends

- Defects and domains - dynamic behaviour [45]
- Thin films, nano-ferroelectrics, electrode effects [63, 64, 53]
- Complex materials - PZT, ceramics, layered perovskites, [65].
- Domain structures and self assembly [54, 63, 66, 65]
- Multiferroics
- Finite temperature and field effects, phase change, thermal expansion [53].
- New materials - lead free piezoelectrics, relaxor ferroelectrics

8.4 Feram Simulation of Barium Titanate

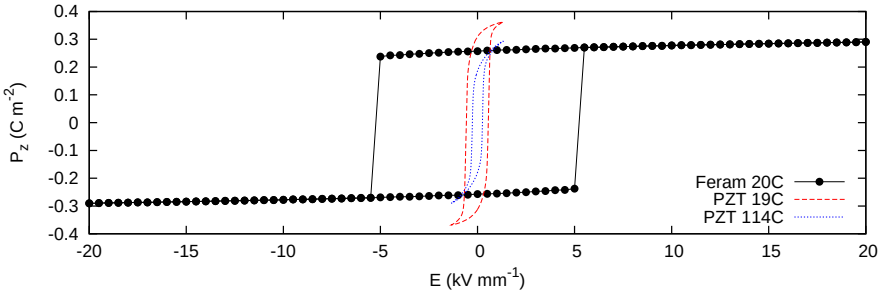
FERAM applies the effective Hamiltonian method of describing ferroelectric properties to finite size, finite temperature simulation using molecular dynamics methods. It is a first-principles calculation in that it (mostly) requiries no empirical data other than a listing of the atoms in the system, and some fundamental constants. In practice, it does make use of the known symmetry and cell structure of the paraelectric phase, and (I think) an empirical value for the cell constant. Interatomic potentials are obtained from previously published density functional theory (DFT) calculations for Barium Titanate. The parameters from these DFT calculations lead to an underestimate of the Curie temperature, so, in line with these previous works, a negative pressure, p=-5GPa, was applied in FERAM. A description of the background and physical basis for the code has been published [53]. We have also investigated the source code and a report is given in Appendix B. As described, there are some significant differences between the actual operation of the code and the description of [53].

Simulations were performed using the FERAM code which is described in more detail by the authors [53]. The source code was downloaded from http://loto.sourceforge.net/feram, compiled and run on a Pentium 4 dual core 2.8GHz pc under Ubuntu Linux 8.10. For most of the results reported here, simulations were performed using a simulation size of 16x16x16 cells using the bulk setting. All other parameters were unchanged from the example hysteresis loop provided.

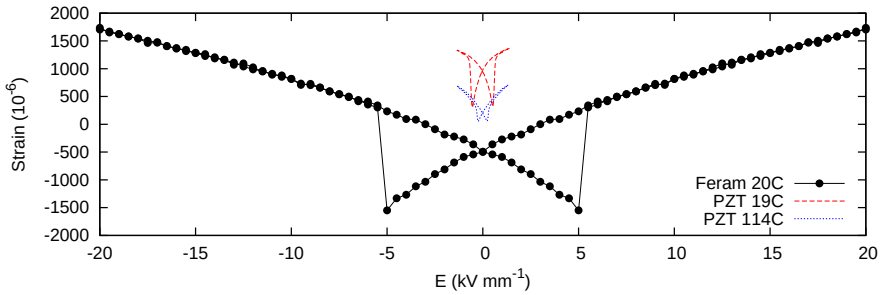
8.4.1 Unipolar and bipolar loops

Figure 8.6 shows single loops calculated using FERAM at a fixed temperature of 20°C. These are compared with measured loops for PZT at the same temperature, but also with PZT loops measured at the same value of T/T_c. This is because FERAM calculates a much lower value for T_c than experiment, so scaling to T_c should provide a more meaningful comparison (see below for further discussion of temperature effects).

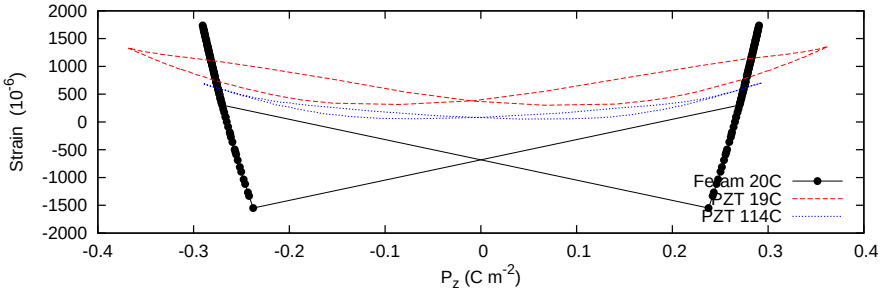
A major difference is the extremely large value of the coercive field obtained from FERAM. This is discussed further below. The magnitude of the polarisation was similar, reflecting the similar values of remanent polarisation, as expected. The overall strain values from the PZT experiments were smaller, but this is because of the extremely high electric field applied in the simulation to achieve a full switching loop. A simulation was therefore also conducted at much smaller values of maximum electric field (2kV/mm) more typical of experiments. The field was much smaller than the coercive field, so no switching, and an approximately linear piezoelectric response was observed. The FERAM d31 coefficient was -54pC/N. This compares to a reported value of -33pC/N [67]. The figure from the PZT loops was -250pC/N,



(a)

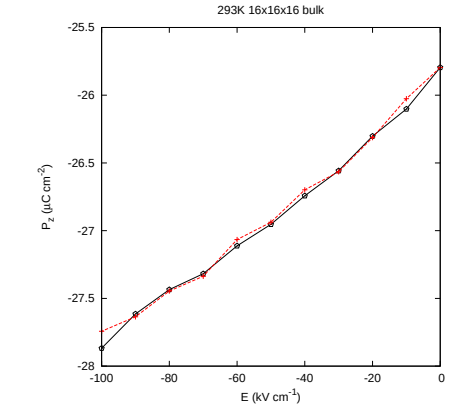


(b)

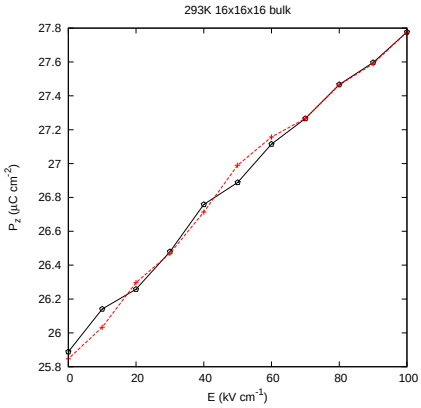


(c)

Figure 8.6: Strain, electric field and polarisation loops compared to experimental PZT results. Results were compared at the same actual temperature, and at the same value of T/T_c. Note that FERAM strain was the strain in the direction of the field (3) while the PZT was transverse (1).



(a)



(b)

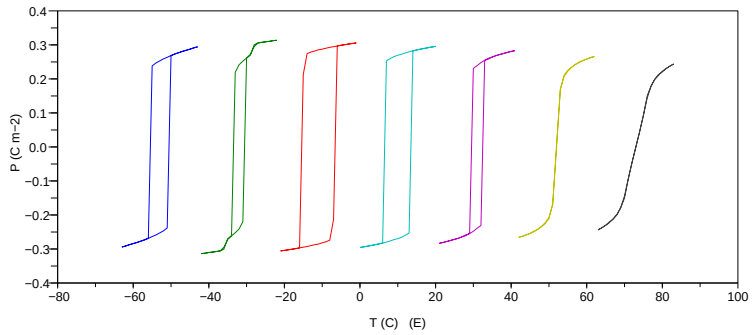
Figure 8.7: Unipolar loops 293K

which compares with a datasheet value of -330pC/N. This is interesting because it implies no significant diminution of piezoelectric activity in nano-scale structures. To verify this we would need to run some simulations on larger systems.

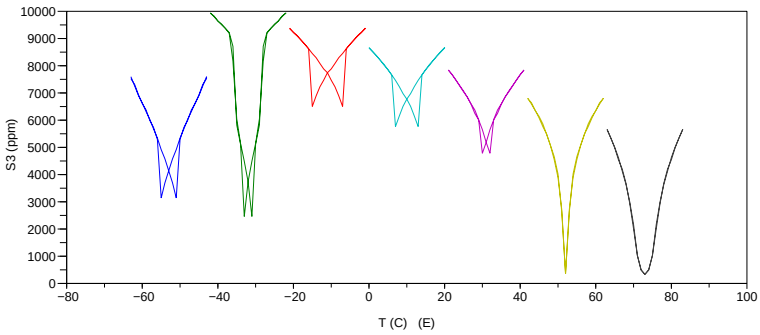
Another observation was that there was no hysteresis between increasing and decreasing electric fields, except that due to switching at the coercive field. To confirm the absence of hysteresis two non-switching (i.e. unipolar) simulations were performed - one with the electric field decreasing from -10kV/mm to zero then reversing the sequence, and one increasing the field from zero to -10kV/mm and then reversing the sequence. Results shown in Figure 8.7 confirm that there is no hysteresis outside the coercive region. This could be related to the high values of the coercive field, and the smallness of the sample discussed above, in that there are no domain motion processes operating, and therefore there is no hysteresis outside of the coercive region.

8.4.2 Temperature Effects

Figure 8.8 shows PE and SE loops generated by FERAM over the temperature range -53°C to +73°C (220K-346K). Above 40°C, the behaviour closely resembles electrostriction. Between 40°C and -20°C normal ferroelectric behaviour is observed, whilst below -20°C the ferroelectric behaviour appears to be suppressed above a certain electric field strength. The paraelectric to ferroelectric phase change temperature is significantly lower than experimental values reported in the literature for the cubic to



(a)



(b)

Figure 8.8: Composite plots showing a) polarisation-field and b) strain-field loops at different temperatures. The x axis values for each loop (electric field or polarisation) were offset to centre each loop on the temperature at which it was measured. Electric field units are arbitrary, but the same for all plots.

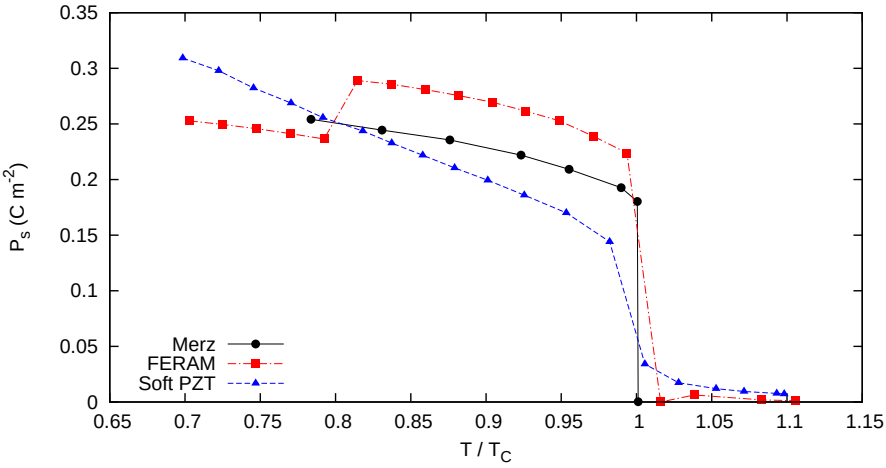


Figure 8.9: Remanent polarisation from FERAM compared to experimental data for single crystal spontaneous polarisation [70] and remanent polarisation in soft PZT. The following values of T_c were used: FERAM 40°C, Merz 107°C, PZT 145°C.

tetragonal phase transition in BTO (approximately 110°C [68]). However, the results from our simulations agree with published results for FERAM [53]. This discrepancy is seen in most ab-initio calculations - both effective Hamiltonian methods (which report similar results to ours) and shell model methods [69]. This is thought to be caused by the DFT/LDA methods used as a basis for these calculations underestimating the ferroelectric instabilities of $BaTiO_3$ and the fact that ferroelectricity is very sensitive to volume. Such models may also fail to account for domain processes generally which have a dramatic effect on macroscopic response.

To account for the discrepancy in transition temperature, results below are plotted as a function of T/T_c.

The remanent polarisation was measured from the PE loops by taking the value of the polarisation at zero applied field. Figure 8.9 shows the temperature variation of the remanent polarisation compared to experiment [70]. At higher temperatures it was zero, as expected for the paraelectric phase with transition to a finite value at approximately 45°C. Below the transition temperature it follows a similar trend to the experimental data, with values slightly higher than experiment. The remanent polarisation is also similar to that for soft PZT (Fuji C91) which is also shown. The second phase transition is clearly seen at approximately -20°C, corresponding to the tetragonal to orthorhombic transition. This can be compared with the experimental value approximately 0°C [68] in Figure 8.10. Again, the predicted values are much lower than experiment, but higher than might be expected from scaling to T_c. The strain values calculated (Figure 8.10) are slightly lower than the a/c values from experiment, but they are of the correct magnitude and follow a similar trend to experiment. It should be noted that the FERAM results show no thermal expansion. This is most clearly seen in the high temperature paraelectric phase. This is a significant limitation of the effective Hamiltonian method. The reason for this [71] is that even though the potentials used in FERAM are anharmonic, anharmonic interactions among phonon modes are not taken into account. It is these phonon-phonon interactions that are responsible for thermal expansion. The shell model of Tinte [71] claims to properly describe thermal expansion, but still suffers from the same underestimation of transition temperatures. Figure 8.11 compares the coercive field from FERAM with experiment and with Landau-Devonshire theory described in Section 8.2. The clustering of the FERAM values is due to the coarseness of interval between electric field steps. The experimental figures for bulk BTO are two orders of magnitude smaller than the simulation. Overestimation of the coercive field is common in theoretical calculations [37] which do not account for domain processes i.e. the crystal is a single domain and the whole crystal switches instantaneously at E_c. The FERAM results agree to

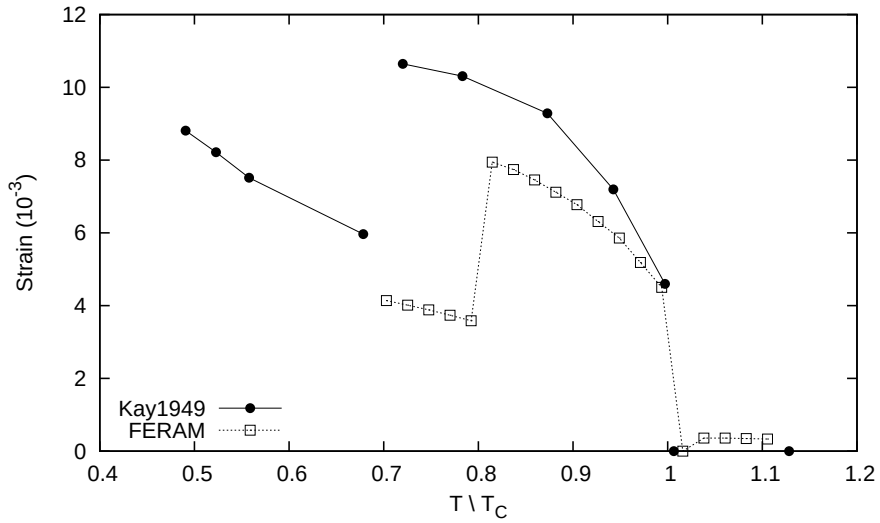


Figure 8.10: Comparison of computed strain from FERAM with experimental data for single crystal values for $\frac{a}{c} - 1$ [68]. The following values of T_c were used: FERAM 40°C, Kay 114°C.

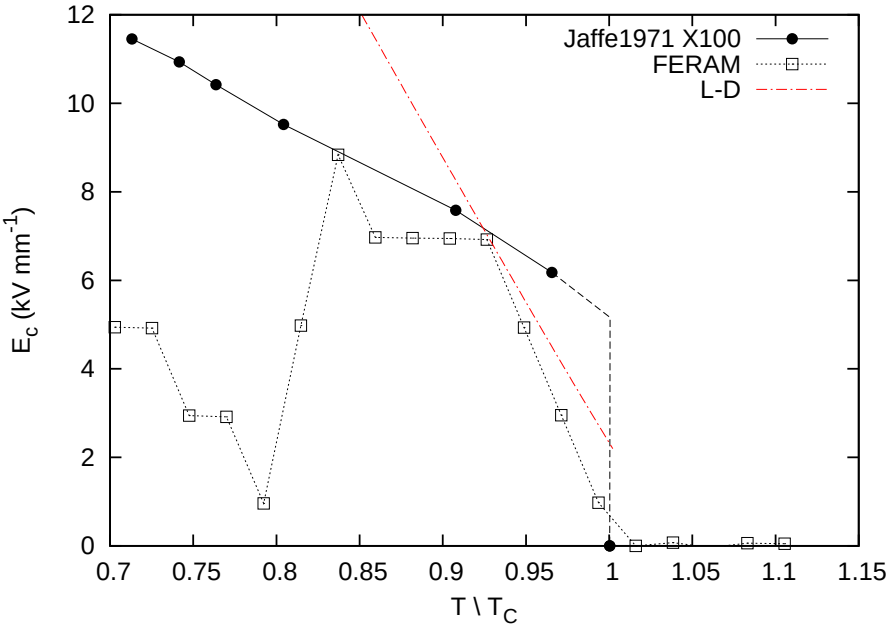
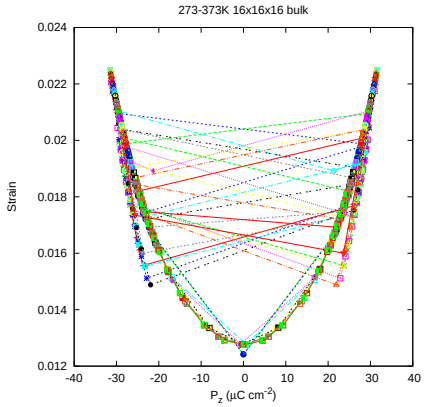
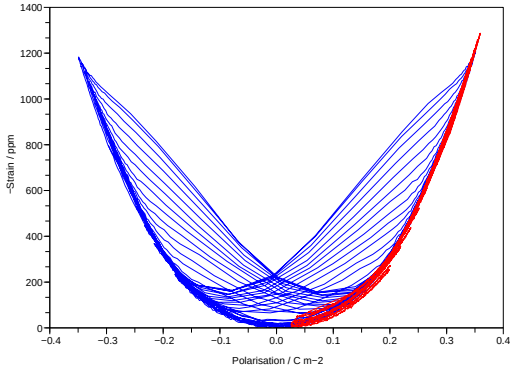


Figure 8.11: Coercive field from FERAM compared with single crystal experimental data [72] and Landau-Devonshire theory (using coefficients from Bell [31]. NOTE: the Jaffe values have been multiplied by 100. The following values of T_c were used: FERAM 40°C, Jaffe 107°C.



(a)



(b)

Figure 8.12: Strain plotted as a function of polarisation. a) FERAM b) soft PZT. The loops from all the temperature steps are shown superimposed.

the correct order of magnitude with the Landau-Devonshire theory, although the trend is different. This indicates that the free energy potentials derived via Landau Devonshire theory from experiment are not far off those use in FERAM. In practice, domains nucleate and grow to accomplish the switching over a longer period of time, at a much lower field. A consequence of this is that the coercive field decreases with the frequency - a factor which is not present in the simulation or in Landau-Devonshire theory. Another factor is the length scaling of the coercive field. The simulation volume was just 16 unit cells on a side i.e. approximately 6.4nm. At these dimensions one would expect from the Kay-Dunn law [73] that the coercive field in the nano-structure would be significantly larger than in bulk. This point was discussed by the authors of the program [53], but they were not able to reproduce the Kay-Dunn scaling in FERAM.

Figure 8.12(a) shows the strain plotted as a function of polarisation from the FERAM simulation. Away from the coercive region, most of the loops appear to align quite closely to a parabolic function. The exceptions in the FERAM results are the loops from the orthorhombic phase below -20°C. Very similar behaviour is observed with soft PZT (Figure 8.12(b)), demonstrating that even in the ferroelectric phase there is an underlying electrostrictive behaviour. It is therefore instructive to examine the electrostriction coefficients. The transverse electrostriction coefficient, Q_{12} is plotted as a function of temperature in Figure 8.13. Excellent agreement with experimental results (obtained from d_{31} and permittivity measurements is seen above the transition temperature, with a slight divergence at lower

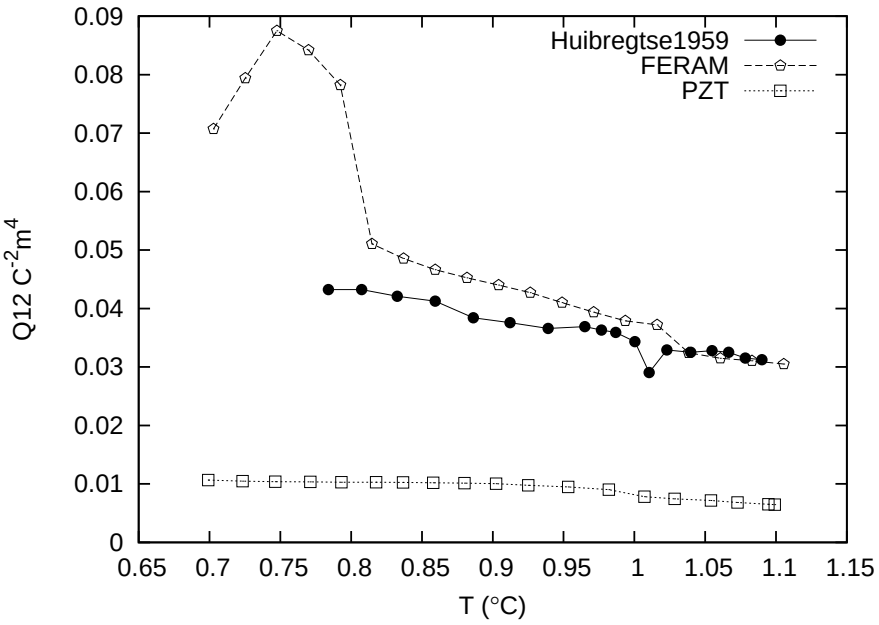


Figure 8.13: Transverse electrostriction coefficients, Q_{12} for FERAM, single crystal experimental data [67] and soft PZT. The following values of T_c were used: FERAM 40°C, Jaffe 114°C, PZT 145°C.

temperatures. The sudden rise in Q_{12} at $T/T_c = 0.8$ is associated with the tetragonal to orthorhombic phase transition, which, as noted earlier, occurs at a lower temperature ($T/T_c = 0.7$) than the range covered by the experimental data. BTO electrostrictive coefficients are significantly larger than those measured for soft PZT, and also previously reported values of $0.016C^{-2}m^4$ for undoped PZT ceramic [74].

8.4.3 Computational Parameters

The effects of FERAM input parameters `P_alpha`, `P_kappa2` and `P_gamma=0` were examined by setting each of them separately to zero with the remaining two at their default values. `P_alpha` returned `NaN` values in the output, and cycled indefinitely without completing a single point. With `P_kappa2` and `P_gamma` set to zero, the calculations completed successfully. Results are shown in Figure 8.14, along with the effects of setting `Zstar`, `B` and `j` to zero. `P_kappa2=0` appeared to produce spontaneous polarisation (half the plot is just minus the first half), but no switching - it's possible that the coercive field became very high or infinite. All the others produced a linear dielectric response.

8.4.4 Bulk, Thin Film and Thermalize

Figure 8.15 shows field loop simulations at 175K with the epitaxial option selected. Calculations were performed for two different values of n_thermalize - the default value 40,000, and a reduced value of 4,000. Reducing n_thermalize by a factor of 10 had little effect on the results except in the coercive region, where it led to an increase in coercive field. Maximum and remanent strain and polarisation values were unchanged. The minimum strain value was also unchanged even though this is reached in the coercive region. The lowering of n_thermalize approximately halved the simulation time from 10 minutes to 5 minutes per point.

The most noticeable effect of the epitaxial selection was to increase the coercive field by an order of magnitude, compared to the values for bulk simulations described above.

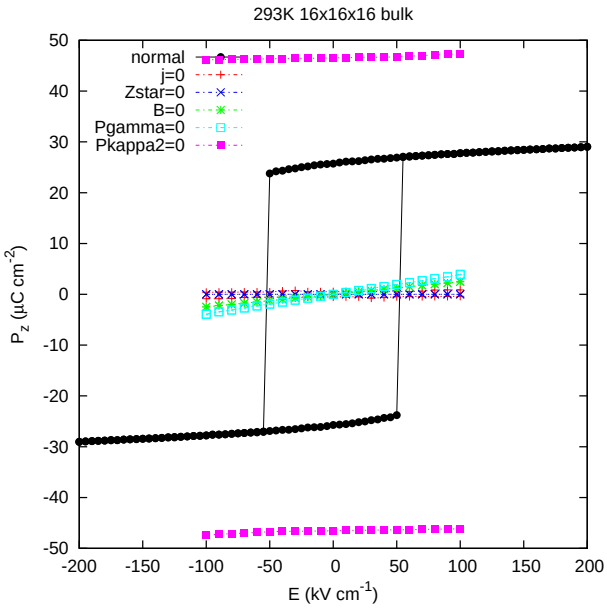


Figure 8.14: FERAM PE loops with parameters set to zero.

8.4.5 Some Ideas for Future Work using FERAM

Whilst we may wish to develop ferroelectric modelling in our own direction, particularly with regard to being able to simulate thermal expansion and domain dynamics, there is some useful work we could conduct using the existing FERAM package.

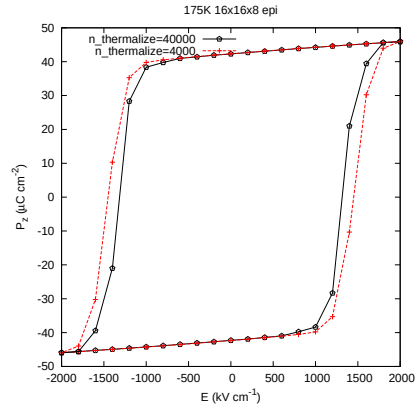
- Simulate PZT using the virtual crystal approximation (VCA) [52]. This replaces the Pb Ti_x 0_2 and Pb Zr_{1-x} O_2 by a uniform but composition dependent "virtual" ABO_3 system using an average pseudopotential.
- Find out how to extract domain structure information from FERAM output.
- Run simulations on larger structures.

8.5 Conclusions

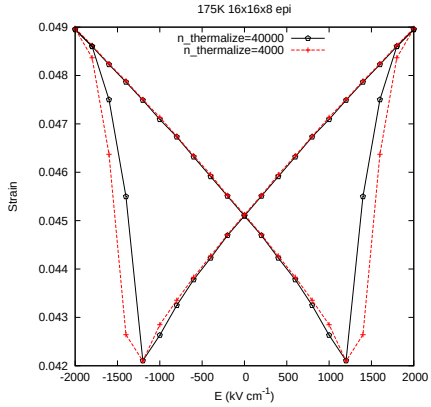
Landau-Devonshire theory is a powerful thermodynamic approach to modelling ferroelctric phase changes and other ferroelectric properties such as P-E loops. Temperature dependent changes in the free energy surface can reproduce complicated sequences of phase changes (such as correctly predicting the sequence of changes in crystal symmetry associated with the ferroelectric phase changes in Barium Titanate) based on a small number of parameters that can be determined from experiment or from atomistic models. This close link with atomistic models such as the Effective Hamitonian model makes an understanding of Landau-Devonshire theory important also for first principles methods.

Modern electronic structure calculations are capable of reproducing well the ground structure of ferroelectrics, but the large number of individual cells required to model finite temperature phenomena mean that some kind of molecular dynamics calculation must be employed.

There are two main methods reported in the literature of atomistic modelling of ferroelectics. The Effective Hamiltonian method employs a systematic reduction of the number of degrees of freedom of the full Hamiltonian of the system. This allows the separation of electronic degrees of freedom and



(a)



(b)

Figure 8.15: Epitaxial 16 x 16 x 8 cells at 175K with n_thermalize set to 40,000 and 4,000

those associated with the ferroelectric distortion. Large thermal simulations can be performed with this method. Shell model methods allow more degrees of freedom of atomic movements, but as this involves more variables, simulation size is smaller than can be achieved with effective Hamiltonian.

Results are presented of the temperature dependency of ferroelectric and piezoelectric propoerties calculated using the Effective Hamiltonian method using the FERAM code. The phase change sequence for Barium Titanate is correctly predicted, although with errors in the transition temperatures as reported in the literature. In common with Landau-Devonshire theory, the simulation overestimates the coercive field by about 2 orders of magnitude.

The Effective Hamiltonian simulation does not show hysteresis for unipolar loops. This is because the crystal is treated as a monodomain, although there are reports in the literature of multi-domain structures. This is also why the coercive field is not well described both by the Effective Hamiltonian and Landau Devonshire theories.

Domain processes play a key role in determining ferroelectric properties, and they are not easily described by atomistic models. This is in part because domain processes operate over length scales and time scales inaccessible to atomistic simualtions, even for nano-scale structures.

Effective Hamiltonian simulations of strain loops show approximately quadratic electrostrictive behaviour. Electrostriction coefficients for the tetragonal and cubic phases of Barium Titanate are close to those reported in the literature, and significantly higher than for PZT.

Chapter 9

Atomistic modelling of ferroelectric properties of $BaTiO_3$ nanoparticles

9.1 Introduction

Nano-structured ferroelectric materials (10-100 nm structures) are at the frontier of current materials research with a broad and cross-sectoral range of potential new applications such as fast terabit-per-square-inch memory, cheap room-temperature magnetic-field detectors, microfluidic systems, electrocaloric coolers, and miniature x-ray and neutron sources, electronically tuneable dielectrics, and frequency agile mobile communications. Nanostructured piezoelectric ceramics could form the basis of the next generation of high performance industrial inkjet printers (billboards to plastic electronics). Dynamic processes govern device performance (e.g. memory access time) and stability of ferroelectric ordering at the nanoscale. Current worldwide research is focused primarily on two general questions: i) are there new phenomena associated with the nano-scale and ii) how best to exploit this in new devices. The overall aim of our work at NPL and the Thomas Young Centre is to experimentally[1] and theoretically investigate the interplay between the scale, geometry, and surface termination of nano-scale ferroelectrics and the structure of the domains that subsequently form in these constrained structures, and link this to the materials' macroscopic behaviour. In this work, NPL are actively developing new capability to perform leading edge metrology of domain engineered ferroelectric materials and their dynamic responses at the nano-scale following the experimental plan above and the theoretical modelling plan next described.

In this chapter, we have developed a set of shell model interatomic potentials, which reproduce the experimentally observed properties of tetragonal phase of the bulk $BaTiO_3$ (BTO). This set of the potentials has been used to model nanoparticles of BTO, investigate the structure of 180° domains there and to model the effects of the external electrical field. We use the GULP program to model BTO and predict a polarisation-electric field hysteresis loop. The results of this chapter may be compared with the effective Hamiltonian results and Landau-Devonshire theory described in Chapter 8.

[1] Ferroelectric materials have been widely used in bulk form as sensors and actuators, but are currently offering possibilities in thin film form for a variety of applications including functional MEMS devices and memory applications (FeRAM). As potential applications tend towards smaller sizes, there is a need for precise characterisation at these nanometre scales. Measurements of the piezoelectric properties can be achieved by using piezoresponse force microscopy (PFM). This technique can qualitatively yield domain orientation, as well as providing a quantitative analysis of the piezoelectric (vertical and lateral) response with a resolution of 10 nm or smaller, but is limited in its ability to divulge grain orientation information. One technique that may be used to obtain such information is electron backscatter diffraction (EBSD). Both grain orientation and ferroelectric domain structure can be quantitatively analysed. With a resolution as small as 20 nm, EBSD offers a viable technique for studying ferroelectric films, which may have grain sizes smaller than 100 nm. Used in conjunction with PFM from the same region, a quantified map of the piezoelectric response of ferroelectric thin films may be acquired.

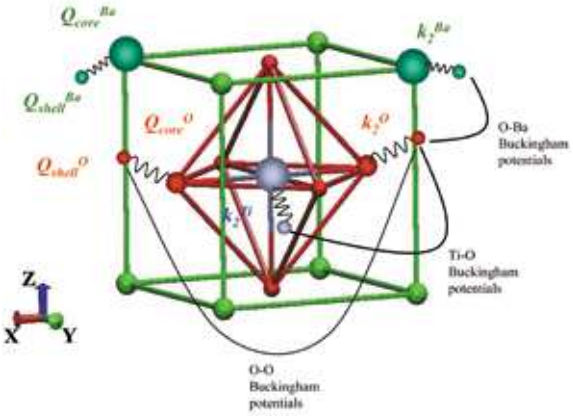


Figure 9.1: Schematic representation of the shell model.

9.2 Interatomic potentials for $BaTiO_3$

9.2.1 Shell model approach

The shell model approach, [75], is routinely used in atomistic simulations of materials properties and has been successfully applied in studies of physical properties of ideal and defect-containing wide band gap materials, [76, 77, 78]. In particular, ferroelectrics and properties of point defects in the bulk and at surfaces of ferroelectrics have been considered in, for example, refs. [79, 80, 81, 82].

Within the shell model, each atom is represented using two centres, denoted as a *core* and a *shell* and assigned charges Q_{core} and Q_{shell}, respectively, as illustrated schematically in Fig 9.1.

Displacement of the shell with respect to the core mimics the atom polarisation, induced by an external perturbation, and results in the positive contribution to the total energy of the system. Cores and shells of different atoms interact with each other electrostatically. In addition, short-range repulsion and van der Waals interaction between atoms is simulated using interatomic potentials of a predefined analytic form.

9.2.2 Initial shell model for BTO

Tinte et. al. have developed a set of shell model potentials for the bulk BTO on the basis of density functional calculations within the local density approximation (LDA), [71]. This set includes the Buckingham-type potentials for interatomic interactions and quasi-harmonic anisotropic intra-atomic potentials for the core-shell interactions of oxygen ions and harmonic isotropic core-shell interactions for Ba and Ti atoms.

The electrostatic interaction between centres i and j belonging to different atoms and separated by the distance r_{ij} is given by

$$V_{ij}^{Coulomb} = \frac{Q_i Q_j}{r_{ij}}.$$

The Buckingham potential

$$V_{ij}^{Buckingham} = Ae^{-\frac{r_{ij}}{\rho}} - \frac{C}{r_{ij}^6},$$

where parameters A_{ij}, ρ_{ij}, and C_{ij} are positive, describes the short-range interaction between atoms i and j and is usually defined between the shells of these atoms. Finally, intra-atomic polarisation is modelled using harmonic and quazi-harmonic potentials:

$$V^{spring} = \frac{1}{2}k_2|\mathbf{r}_{core} - \mathbf{r}_{shell}|^2 + \frac{1}{12}k_4|\mathbf{r}_{core} - \mathbf{r}_{shell}|^4,$$

where $\mathbf{r}_{core}$ and $\mathbf{r}_{shell}$ define positions of the core and the shell centres and k_2 and k_4 are positive. The values of parameters A, ρ, C, k_2, and k_4 are, in general, fitted for each type of the interatomic interaction, e.g., Ba–O, Ti–O, and O–O.

The parametrisation proposed by Tinte et. al. successfully reproduces the cubic and tetragonal crystal structures, the temperature-dependent phase transition and the fundamental properties of the single crystal.

9.2.3 Modified shell model for BTO

In spite of its successes, Tinte's parametrisation has significant disadvantages for its practical use: it uses anisotropic polarisation potential for oxygen atoms. In particular, the value of the parameter k_4 depends on the direction, in which the shell displaces with respect to the core. Practical implementation of this condition in standard computer codes is problematic. More importantly, this parametrisation requires that the direction of the ferroelectric displacements is predefined, which restricts the variety of domain boundaries that can be considered.

To avoid this restriction we chose to abandon the quasi-harmonic term k_4 together with the condition of the orientation-dependent polarisability. Instead, to describe the tetragonal phase of BTO, we introduce a corrective Morse potential between Ti and O ions:

$$V_{ij} = D[(1 - e^{-\alpha(r_{ij}-r_0)})^2 - 1],$$

where parameters D, α, and r_0 are positive. The Morse potential, [83], has a convenient functional form for the corrective potential because all three parameters D, r_0, and α have a transparent physical meaning of the potential well depth, equilibrium distance between ions, and vibrational frequencies at the equilibrium configuration, respectively.

By choosing the value of r_0 to be slightly smaller than the Ti–O interatomic distance, we modify the potential energy surface so that the ferroelectric displacement is equally likely to occur in all directions equivalent to the (001) crystalline axes. Then, parameters D, α, and r_0 have been adjusted in order to reproduce the experimental macroscopic crystal properties of the tetragonal BTO, such as lattice parameters and elastic constants, [84, 85]. The optimization procedure was performed by minimizing a square sum of errors between model and target properties using the Newton-Raphson method as it is implemented in the GULP (General Utility Lattice Program) code, [86].

9.2.4 Validation

The resulting set of the optimal parameters of the Morse and Buckingham potentials, core and shell charges (Q_{core}, Q_{shell}), and spring constants (k_2) are presented in Table 9.1. The modified potential set reproduces correctly, when energy is minimised, the structure of the tetragonal phase of BTO. In particular, the tetragonality ratio c/a is 1.0102, which is in a good agreement with the experimental value of 1.01, [84]. The calculated lattice parameters a=3.92 Å and c=3.96 Å are slightly underestimated with respect to the experimental values of a=3.99 Å and c=4.036 Å, [84]. This set of potentials also provides a reasonable agreement of the elastic constants with their experimental values (See Table 9.2), [85]. The calculated value of the macroscopic spontaneous polarization P in the bulk BTO crystal is 1.57 10^{-2} $e/\text{Å}^2$= 24.6 C/m^2.

9.3 BTO nanoparticles

We constructed stoichiometric BTO nano-size systems and applied the developed potential set to evaluate the structural and ferroelectrical properties of these nano-particles. The GULP code has been used in all simulations described below.

Buckingham potentials	A **(eV)**	ρ $(\AA)$	C **(eV**$\AA^6$**)**
Ba–O	864.536	0.38729	0.0
Ti–O	4526.635	0.25239	0.0
O–O	4102.743	0.29581	300.0
Morse potential	D **(eV)**	α $(\AA^{-1})$	r_0 $(\AA)$
Ti–O	4.75	45.231631	1.7129
Spring Constants	k_2 **(eV**$\AA^{-2}$**)**		
Ti	321		
Ba	251		
O	101.27		
Charges	Q_{core} (e)	Q_{shell} (e)	
Ti	4.76	-1.58	
Ba	5.62	-3.76	
O	0.91	-2.59	

Table 9.1: Optimized potential parameters set for tetragonal phase of BTO.

9.3.1 Constructing BTO nano-particles

Properties of the nanoparticles can be strongly affected by their shape and the structure of the terminating surfaces. In the case of ferroelectrics, the electrostatic field, produced at the surface, may compensate the lattice ferrorelectric distortion at these length scales. Detailed studies of the surface structure and its effects on the properties of nanoscale BTO will be investigated at a later stage of the project discussed in Section 9.1.

At present, we chose to construct BTO nano-particle (NPs) using bulk-like $[\mathrm{Ti}(\frac{1}{8}\mathrm{Ba})_8(\frac{1}{2}\mathrm{O})_6]$ building blocks. A lattice formed by periodical translations of these building blocks coincides with that of BTO. In the case of the cubic phase, these building blocks are neutral and have zero dipole and quadrupole moments, which guarantees absolute and fast convergence of the electrostatic potential with the size of the nano-particle. Nano-particles are constructed in the form of cuboids containing $n \times m \times k$ building blocks.

9.3.2 Boundary effects

We constructed several $N \times N \times N$ BTO nanostructures (N=3, ..., 6) and minimised the total energy with respect to the coordinates of all cores and shells in each case. We found that BTO NPs exhibit a noticeable relaxation in vicinity of the border. However, this relaxation is mostly confined to a near-surface region, which is approximately one unit cell thick. The effect of the surface on the internal structure of the NPs

	Calculated	Experiment
C_{11}/C_{22}	421	222
C_{33}	203	151
C_{12}	150	107
C_{13}	122	111

Table 9.2: Comparison of elastic constants of $BaTiO_3$ obtained from the optimized shell model and experiment ref .[85]. Values are given in GPa.

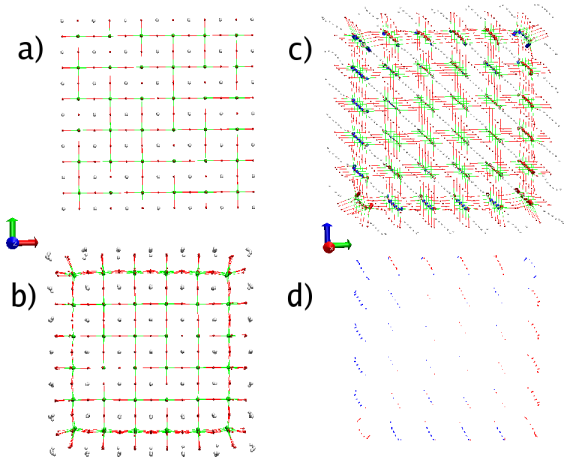


Figure 9.2: Relaxation of BTO nanoparticle leads to appearance of compensating dipoles at the border. Spatial distribution of the local dipoles is shown by a set of red and blue sticks corresponding to parallel and anti-parallel direction of a local dipole moment with respect to Z axis. a) Initial configuration, b) final relaxed configuration, c) finite configuration with local dipoles, d) spatial distribution of local dipoles in relaxed configuration.

is negligible.

This is illustrated on the example to the 6×6×6 NP in Figs 9.2(a, c), where we compare the initial and final configurations for this nanoparticle. The lattice distortions at the border induce compensating local dipoles. To visualise the distribution of these dipoles, we use a *local dipole* representation in which the dipole moment is calculated separately for each $Ti(\frac{1}{8}Ba)_8(\frac{1}{2}O)_6$ building block. The magnitude and the direction of the dipoles are represented by a bar placed at the Ti position and oriented accordingly. For clarity, we use red and blue colours to indicate orientation of the dipoles with respect to Z axis: dipoles directed parallel to Z axis are coloured red, while those oriented in the anti-parallel direction are coloured blue.

One can see that the distribution of local dipoles at the border has alternating character. This distribution corresponds to the minimum of the total energy of the system and takes into account the balance between the elastic energy of the near-surface region and electrostatic energy of the interacting local dipoles.

9.3.3 Modelling 180° domain walls

In order to represent domains in BTO nanoparticles, the initial distribution of the local dipole moments is pre-set using an analytic function f:

$$f = \arctan(n\sigma),$$

where n is a number of the unit cell along a given direction, and parameter σ defines the thickness of the domain wall. Modulating the local atomic structure with the function f allows us to preform two

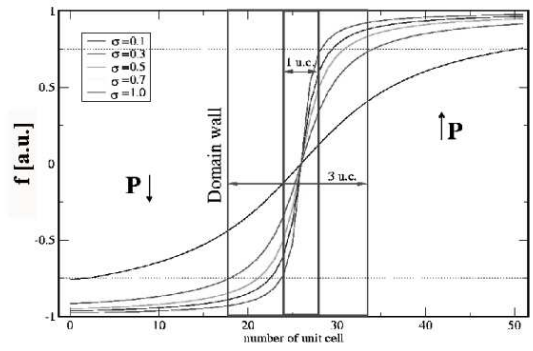


Figure 9.3: Analytical function used for description of the polarization P along BTO cluster. Variation of parameters σ allows to describe different thickness of domain wall.

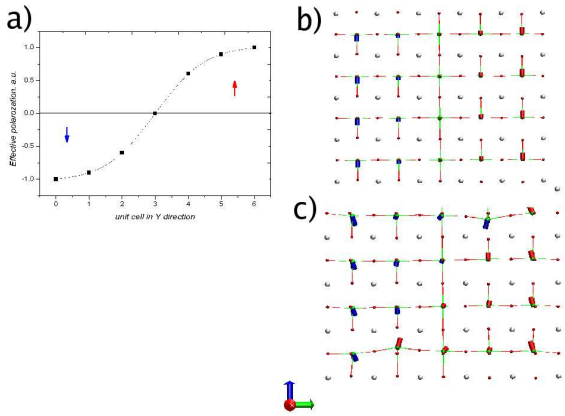


Figure 9.4: Distribution of local dipoles in 3×5×4 BTO nanoparticle. a) analytical function which describes a distribution of polarization and the width of domain wall in the cluster. Initial (b) and final (c) atomic configurations and spatial distribution of local dipoles in the system. One can see that the shape of domain wall is changed as system relaxed. This is due to boundary and finite size effects.

domains with the opposite orientations of the macroscopic polarization vector P (see Fig. 9.3).

This approach has been applied to a stoichiometric 3×5×4 BTO NP of BTO. In this case the initial coordinates of the system were set up so that one 180° domain wall is positioned at the centre of the "long" dimension of the nanoparticle and is oriented parallel to a crystalline plane equivalent to (001). The value of the parameter σ was set to σ=0.1 (see Fig. 9.4a).

The initial atomic structure of the NP, together with the initial distribution of the local dipoles, is shown in Fig. 9.4b. One can see that the domain wall is located in the middle of the NP and is oriented parallel to [001] plane where local dipoles have zero values. Relaxation of the system leads to a distortion of the lattice in vicinity of the border. The latter induces a perturbation of dipole distribution at the border and in the bulk. As a result of the relaxation, the domain wall reorients (see Fig.9.4c).

9.3.4 Effects of the external electrical field

External electric field has been modelled using two sets of point charges arranged on a square grid of m×m points. Each charge has a value of $+q$ on one plate and $-q$ on the other and the area of each plate S is chosen so as the field produced by the plates is constant. The charge density on each plate, given

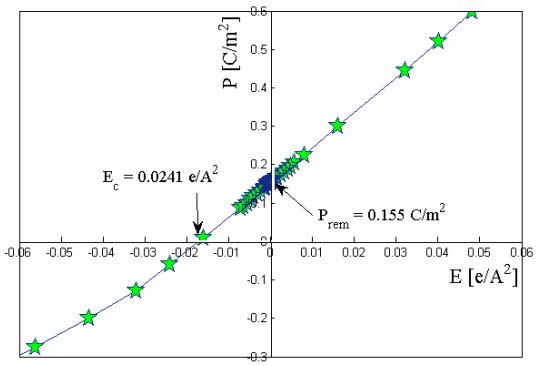


Figure 9.5: Dependence of macroscopic polarization (P) in 3×3×3 NP of BTO with respect to electric field (E). At zero field the particle has polarization of P_{rem}=0.155 C/m^2, which is about half the modelled spontaneous polarization of bulk BTO. This effect is due to relaxation of the surface, which suppresses the polarization of the particle. For small NP the $P(E)$ dependence has linear character typical for most of dielectrics. This is also due to the finite size effect and limitation of the shell-model in description of the charge redistribution.

by γ=m^2q/S defines the magnitude of the electric field (see Figure 9.5). The intensity of the electrical field can be varied by changing the value of q.

The effect of the external field has been considered on the example of a cubic 3×3×3 BTO nanoparticle. The dependence of macroscopic polarisation P, distribution of local dipoles on the intensity of the external field, E, have been considered for the range of the E values from -0.06 to 0.06 e/A^2.

Initial relaxation of the NP without application of the external electric field leads to relaxation of its surface and consequent appearance of compensating dipoles at the border similar to that shown in Figure 9.2. The polarization of the particle is P_{rem}=0.155 C/m^2, which is about twice smaller than the value of spontaneous polarization modelled in *bulk* BTO. This is due to finite size effect and relaxation of the surface that suppresses polarization of the small NP.

Application of the external electric field induces displacements of ions of NP from their equilibrium positions so as to counterbalance the external field (See Figure 9.6). This consequently modifies the distribution of the local dipole moments, affecting the macroscopic polarisation of the system. Figure 9.5

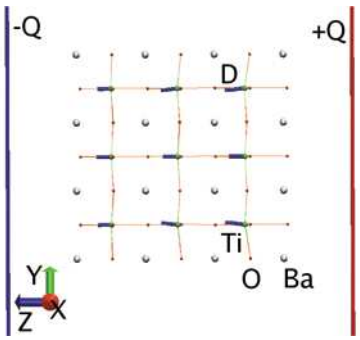


Figure 9.6: Relaxation of cubic 3×3×3 BTO nanoparticle in the external electrical field. The electrical field, set by charged plates with the charge $\pm Q$ on both sides of nanoparticle, leads to a distortion of the system's geometry with respect to tetragonal one. This induces a modification of distribution of local dipoles and consequent change of macroscopic polarization P. This is similar to "poling" the material

demonstrates a change of P of modelled NP with respect to the external electrical field E. One can see that polarization of NP is reversed at E_c=0.0241 e/A^2.

However, we were not able to reproduce hysteresis behaviour due to the finite size effect and limitations of the shell-model. Indeed, the effect of the surface relaxation on relatively small nanoparticles of BTO is large enough to suppress the presence of two (in one-dimensional case) equivalent energy minima configurations with opposite direction of polarization vector. This leads to linear dependence of $P(E)$, which is typical for most dielectric materials. Therefore, as a next step we will conduct a sequence of classical simulations to study the effect of the surface on polarization behaviour of BTO nanoparticles, and quantum mechanical studies to reproduce this type of charge redistribution at the presence of the electrical field. One problem is that GULP only allows external electric fields to be applied across a finite sized nanoparticle with open boundaries. This is because actual "capacitor plates" need to be inroduced in the simulation which breaks the periodic boundary conditions. To simulate bulk properties, it would be better to use periodic boundaries and apply the external field directly by adding a potential energy function. It seems the current version of GULP cannot do this and we will investigate whether the code can be altered to allow this approach.

9.4 Conclusions

In this chapter, we have shown some preliminary results for molecular modelling of barium titanate using the GULP simulation code. In contrast to Chapter 8, where the unit cell is condensed into a single effective dipole, here we model the whole unit cell in a realistic way with all atoms in place. We first attempted to reproduce the calculations of Tinte [71], but these did not produce a spontaneous tetragonal unit cell but rather a cubic unit cell at lowest energy. It appears that Tinte has achieved the tetragonal behaviour by using anisotropic core-shell springs which are not supported by GULP. We therefore added another Morse potential between titanium and oxygen atoms so that the lowest potential energy is achieved when the atoms adopt off-centre positions resulting in spontaneous strain and polarisation. We have set up a simulation with many unit cells and atoms oriented so as to form two domains with a domain wall in between. By relaxing the system to its lowest energy we have shown how the domain wall would appear on the nanoscale in terms of atomic displacements. It should be noted that, in this calculation, we set up the domains explicitly, the system does not yet show spontaneous domain formation from a random state.

We have encountered problems observing the expected ferroelectric switching behaviour. The spontaneous polarisation is correct but when we applied electric field to the system it did not "switch" to one of the other tetragonal states as expected. Therefore, the $P(E)$ curve is a straight line rather than a hysteretic curve. There are several reasons for this: it may be that, although we have set the pair potentials (including the extra Morse interaction) to give an energy minimum for a tetragonal state, still other minima may exist (eg corresponding to cubic symmetry atomic positions) and the system is not able to find the tetragonal state spontaneously when the atoms are initially set in a non-tetragonal arrangement. This may explain why the system does not switch from one tetragonal state to another symmetrically-identical state. Another condition for domain formation is that the energy must be lower when the "spins" of neighbouring unit cells align rather than anti-align. We have not yet demonstrated that this is the case, and this result is not trivial.

We are hindered in this work by the fact that GULP does not support the application of arbitrary external electric fields. Electric fields must be applied by putting actual "capacitor plates" across the sample which then prevents the use of periodic boundaries. We do eventually want to simulate finite size nanoparticles but it would be nice to calculate bulk effects like $P(E)$ loops in the first instance using periodic boundaries. We will study GULP further to see if applied field and periodic boundaries can be specified. During the course of this project, we have also taken steps to write our own energy minimisation code and this will allow more flexibility in terms of boundary conditions.

Although we are experiencing some difficulties with shell model calculation of piezoelectrics, the shell model may turn to be more useful in the long term than the effective Hamiltonian approach of Chapter

8. The shell model has the advantage of having input parameters (pair potential parameters) which have direct physical meaning. The same cannot be said for the effective Hamiltonian technique in which the input paramters must be derived from the raw pair potentials by a complex pre-calculation. We do not currently have the expertise to implement such a calculation and there seems to be sparse literature on the subject.

Chapter 10

Conclusions

In this report, we have presented a variety of MD/MC simulation codes. A wide range of materials and conditions were considered from argon gas at 90K to lead/copper mixtures at 1200 K. Gas, liquid and solid phases were considered encompassing water, carbon dioxide, metallic alloys and piezoelectrics. It is remarkable that such a range of phenomena can be investigated by the fundamentally simple processes of MD/MC which can be summarised as: calculate forces/energies between atoms, update their positions. Of course, there is a considerable gulf between merely calculating atomic trajectories and actually getting thermodynamic data, particularly in the case of that most characteristically thermodynamic result, the phase diagram. Many researchers never cross this gap, presenting only cartoons of atomic positions as the final result of their studies. We consider such studies to be rather unsatisfactory in that the result can never be proved right or wrong. Furthermore, such results cannot really influence the design and optimization of industrial processes. In this work, we have compared each simulation result to experimental data. Comparisons have also been made with analytical theories where available. In some cases, such as the Landau-Devonshire theory, Chapter 8, the analytical model gives results which are as accurate as the the full MD model. Specifically: for barium titanate, the difference between theory and experiment is much greater than the difference between the analytical model and the MD results. This is probably due to both theories omitting several key mechanisms such as domain wall motion. As a result of these comparisons between models, theory and experiment, we believe we have quantified the typical accuracy of MD/MC simulations in a variety of systems.

The reason for our very "hard headed" attitude to molecular modelling is the requirement, in this project, to match MD results to MTDATA. It would be hard to imagine a more practical, pragmatic group of people than the users of MTDATA: these are industrialists who use the phase diagram predictions of MTDATA to design refinery processes, investigate mining for ores, improve smelting efficiency etc. These people need to know exactly how pressure will really scale with temperature in a boiler operating at 1000 atmospheres, when phase transitions will really occur in industrial processes and so on. Great accuracy is required in all such theoretical predictions with errors potentially resulting in refinery explosions, loss of life and material damage. Refinery engineers would clearly be unimpressed by cartoons of simulated atomic motion. Even quantities like radial distribution function (RDF), while technically measurable using neutron diffraction, are of little interest to this audience. It is not RDF but PSI[1] that makes the boiler explode and that is what we have calculated in this project!

MTDATA works by maintaining a large database of free energies for many different species. But, however large the database, it is impossible to maintain free energies for every possible *combination* of species. This is especially true when we consider that MTDATA is regularly used to perform calculations on mixtures with up to 20 species! To overcome this, MTDATA uses simple algorithms to combine individual free energies to give the overall free energy of the mixture. The simplest way to do this is to simply add free energies together. While certainly correct for ideal gases, this is not accurate for more complex systems. Other algorithms are available, but it is hard to know how accurate such analytical mixing rules

[1] Pounds per square inch. Admittedly, we have used metric units in this report but the conversion is, of course, simple!

are. This is an area where MD can really boost the accuracy of MTDATA predictions. Direct simulations can be performed on the mixture in question giving the thermodynamic potentials and phase diagrams directly. These simulations take the full coupling interactions between the species into account: there is no need to use mixing rules. Unfortunately, the results of Chapter 7 show that we still have a long way to go to accurately model chemical potentials in mixtures using MD/MC. At the beginning of this project, the only solver we had capable of predicting chemical potentials was the Monte Carlo code TOWHEE. Surprisingly, chemical potential calculation is not commonly implemented in MD codes, perhaps because it is so difficult, as we discovered, to obtain accurate results. Chemical potentials are not supported in DL_POLY, for example, although they are an obvious quantity to calculate in liquid simulations.[2] We did program chemical potential calculations into MDL using the ghost insertion method, (1.3). The results are still quite inaccurate compared to TOWHEE showing that a straightforward implementation of the ghost insertion method may not be sufficient (TOWHEE uses a more sophisticated technique). Even with TOWHEE, however, we did not achieve accurate results for the copper/lead mixture we simulated. Results are good enough to make us think the simulation contains meaningful physics (and the agreement with MDL gives us confidence both simulations are performing the correct calculation) but considerable refinement will be needed to obtain the accuracy needed in industry. This will possibly involve simulating with more atoms or over longer time periods. (In fact, the lead chemical potential seemed to be accurate, it was the copper chemical potential that "let the side down", presumably due to the very strong copper-copper interaction. Phase diagrams of alloys are driven by the difference in chemical potentials so we cannot use the better quality lead simulation results on their own)

Although we are not (yet) very successful with mixtures, we *were* able to provide a benefit to MTDATA using single-species gaseous simulations. MTDATA normally uses ideal gas approximations (or other simple algorithms) to model gases. In Chapter 6, we showed that the gas models of CO_2 calculated by MD codes are more accurate than ideal gas models. This was proved by comparison to experiment. However, experimental results run out at the very high temperatures and pressures often used in industry and we were able to show that MDL works well at simulating gases in these regimes. A phase diagram graph was presented in this chapter, using the real gas predictions of MDL and compared to MTDATA's usual methods. The MDL-based phase diagram is shown to be closer to experimental reality indicating that MD simulations can be used to improve MTDATA's results.

In principle, MD can be used to model all three phases of matter. In practice, no program exists which can successfully model solids, liquids and gases (at least not one that can predict the very different quantities and behaviour we are normally interested in when studying each phase of matter). The division is really between solids, on the one hand, and liquids/gases (i.e. fluids) on the other. In solids, we wish to model crystal lattices with very specific symmetry and geometry, with well defined connectivity between atoms. In fluids, we model many molecules, interacting with each other in an amorphous, random way. This has resulted in very different interatomic force calculations being undertaken in fluid MD programs (like DL_POLY) and solid MD programs (like GULP). In fluids, atoms interact with each other over short ranges via the minimum image convention which sits at the heart of the code and is generally enshrined in advanced techniques such as the Verlet neighbour list. In solids, the minimum image convention is not used and we are more interested in forming crystal lattices. Verlet lists are not appropriate and would cause difficulties as the lattice expands and contracts. Basically, solid and fluid MD simulations have a common ancestor but have long since diverged into different species. In a way, this situation is unfortunate, it brings fragmentation to the subject area. But it is inevitable when we consider the very different phenomena needed from liquid and solid simulations (for example, liquid/gas phase transitions for the former; symmetry-based phase transitions, hysteresis, functional material effects – such as piezoelectricity – in the latter).

In principle, liquid programs like DL_POLY, could be "tricked" into solving crystal lattices as shown in Chapter 2 but we strongly recommend against this as there is too much chance of making mistakes setting up the simulation. MD simulations are much harder to set up correctly than other physics simulations. For example, finite element codes set out to solve a set of equations. We can tell straight

[2] Chemical potentials basically represent the link between molecular modelling and chemistry: which could be defined as the science of interacting mixtures of species

away whether such a program has succeeded by substituting the result back into the original equation. No such "equation" exists in MD, the output just consists of a list of atomic trajectories and one list looks much the same as another. If the input files are incorrectly set, these mistakes can be very hard to spot. If the "trick" shown in Figure 2.12 is used incorrectly, one could end up failing to connect the lattice correctly resulting, not in barium titanate, but a gas! To put it in financial terms: if you already own a liquid simulation program and want to reuse it to do solid modelling, forget it! The "total cost of ownership" of purchasing (or writing) a solid simulation code (and training to use it) is less, even in the short term, than struggling on with the liquid code.

Although solid symmetry-based phase transitions (and related hysteretic effects) are inaccessible to liquid solvers, in principle it should be possible to predict the density mediated solid/liquid/gas phase transition. The "solid" in question would still be random in appearance, however, it is unreaslistic to expect a barium titanate lattice, for example, to form spontaneously from randomly-positioned atoms. In fact, we are not aware of any MD studies in which all three phases of matter have been shown to coexist: the solid phase is definitely the "elephant in the room" in conventional, "liquid" MD literature. Even in the comprehensive study of the single atom Lennard-Jones system presented in [11], no mention is made of a solid phase and the only phase transition that seems to be considered is the liquid/gas transition.

Even within the world of fluids, we have found difficulty simulating the liquid phase itself. To be sure, we have shown phase transitions between liquid and gas, but we could only obtain meaningful results close to the critical point where the fluid is still rather "gas-like" even on the liquid side of the transition. For one thing, the increase in pressure is less extreme as we cross the phase transition line near the critical point. At low temperatures, far from the critical point, the pressure increases extremely rapidly as volume is compressed and the $p(V)$ curve is extremely hard to measure in simulations. But even at individual PVT points, we found it hard to achieve pressure convergence in the deep liquid region of the phase diagram. For example, in room temperature water, we could not get sensible results for pressure: the fluctuations from time step to time step were greater than the average pressure even after long simulation runs. This may be due to the very high density, low pressure and incompressibility of liquid water causing problems with equilibration. It should be noted that the same simulation which produced such meaningless pressure results, produced very *good* results for radial distribution functions (RDF), Chapter 5 and these RDF calculations were initially used as proof of MDL's accuracy. It seems that RDF is not a good measure of MD simulation accuracy. This finding is echoed in Martin Chaplin's website (www.lsbu.ac.uk/water), which contains more than 1600 references to MD/MC simulations of water. To quote:

> Also, tests for 'fit' often seem to be completed with publication in mind rather than rigor; thus many papers use the radial distribution fit with diffraction data as their 'gold standard' in spite of the major fitted peaks (where the agreement looks so impressive 'by eye') being derived from the tetrahedral nature of water that is built into every model and overpowering any disagreement in the fine detail.

In other words, agreement with published RDF just means the simulation has been set up correctly, not that it is running to produce a correct, non-trivial result.

Thus, even using a liquid MD solver, there are large areas of the gas/liquid phase diagram which are very hard to solve for accurately, something one would not expect from the "rose tinted" impression given by the majority of MD papers! In general, it is important to keep an eye on convergence information for all MD simulations. The pressure output might not have converged properly, for example. Also you might *think* you are running a simulation at 100 K, but the algorithm that maintains temperature in MD might fail in certain circumstances resulting in an actual temperature of 50 K. Constant vigilance is essential making MD a very labour intensive subject of study.

In this report, we have treated experimental data as the "gold standard" and assessed the validity of computer simulation models based on accuracy compared to experiment. Thus, simulation models can be used to extrapolate or interpolate experimental data, filling in the blanks of experimental studies. Sometimes the process works in reverse: experimental data can be used to deduce MD pair potentials via inverse modelling. In this sense, MDL, for example, is just an expensive replacement for $p(V)$ function

like (3.1). Given this approach, it is worth comparing the various analytical theories discussed in this report with the MD/MC results. An MD or MC program involves writing 10^5 lines of code and needs a large computer to run. Even if the code is already available, the user must set many obscure parameters in the input file, which involves considerable expert knowledge and experience. As mentioned above, it is all too easy to set up an MD code incorrectly and end up simulating a completely different system from that intended. Such errors can be very hard to spot. By contrast to all this difficulty, analytical formulations like (3.1) are very easy to implement, even using a pocket calculator. Is MD worth the extra effort? In Chapter 4, we showed that MD was better able to represent experimental data than the van der Waals function (3.1) even when parameters of both theories (ϵ, σ and a, b respectively) where fitted to the experimental results. (we also noted that, simply getting the algorithm to agree with experiment on the actual critical temperature and density does not mean the specific $p(V)$ curves will agree). The analytical approximation for mixtures, given in (3.82), is probably even more inaccurate as shown by the difference between the analytical results and MD/MC findings in Chapter 7 (not surprising given the equation ignores effects of volume and pressure entirely). Nonetheless there was still considerable disagreement with experimental findings in all types of theory making the distinction between the MD/MC and analytic results insignificant by comparison. This theme also emerged in the barium titanate experimental comparisons shown in Chapter 8. Here, the approximate Landau-Devonshire result does nearly as good a job as the MD result of FERAM, although both theories are a long way from experimental reality (at least in terms of transition temperatures). *The Landau-Devonshire theory thus provides a useful approximation for practical piezoelectric calculations.* It seems that, in cases where MD gets close to experiment, it will be more accurate than competing analytical expressions. In the large number of cases where MD does a poor job of approximating experiment, analytical calculations may be prefered and are certainly much more convenient.

Even when using MD, it is wise to use analytical calculations, however approximate, first. This will give an idea of the order of magnitude pressures, chemical potentials etc to expect and what volume, number of particles (etc) regime should be considered. MD is expensive in every way (puchase/writing of software, staff time, training, computer hardware) and anything that can be done to minimise MD runs is worth while.

In this report, we have simulated piezoelectrics using two different molecular modelling approaches. In Chapter 8, we use the effective Hamiltonian method coded in FERAM. The operation of FERAM is described in the paper [53]. However, when we looked through the code, Appendix B, we found that what it actually does differs significantly from this paper. The way FERAM works is to combine all the atoms in each unit cell into an effective "dipole". This reduces the computational difficulty of the simulation but at the expense of introducing an extra layer of theoretical abstraction between the actual crystal and the code itself. By contrast, in Chapter 9, we have simulated the barium titanate crystal directly, accounting for the position of all the atoms and introducing simple pair potentials between them. The FERAM code is rather a strange mixture of atomistic and phenomenological modelling. Elastic properties must be *input* to FERAM, while these quantities are *predicted* by GULP in our direct BTO simulations, Table 9.2. The other quantities input to FERAM include short and long range interactions and Fourier transformed interaction strengths (the total number of parameters is rather large). Unfortunately, none of these quantities has a direct physical meaning: they must be derived from pair potentials through some complex pre-processing calculation to which we do not have access currently. In the direct BTO calculations, the input parameters are straightforward pair potentials between atom types, with direct physical meaning. In order to use FERAM effectively, it is necessary to understand the effective Hamiltonian theory well enough to be able to formulate the rather abstract input parameters. We have not mastered this theory yet, and there seem to be no textbooks on this subject. Finally, FERAM, having eliminated individual atoms, does not produce atomic coordinates as output so it is impossible to get a feel for where atoms are going when the piezoelectric forms domains etc.

However, it must be admitted that FERAM is currently much more successful at producing realistic piezoelectric data than the direct MD methods of Chapter 9. FERAM successfully produces the rhombohedral-orthorhombic-tetragonal-cubic phase transitions found experimentally (albeit in a temperature-scaled plot) and also shows fairly hysteretic P-E and strain-E loops. So far, we have not reproduced these

results using GULP: plots of P-E do not show hysteretic switching behaviour, although the remanent polarisation is accurate. This may be because we are currently simulating a small "nanoparticle" with open boundaries rather than a true crystal lattice. This is because GULP only seems to allow the imposition of an external electric field by setting actual "capacitor plates" around the sample. This breaks the translational symmetry and prevents us using periodic boundaries. We will try to find a way round this limitation or write our own code if necessary (we have started writing a "GULP-replacement" already. Since GULP is not open source, this may be the best strategy in the long term). Papers such as [71] show full hysteretic behaviour using direct MD so it seems likely we will eventually succeed. Based on the discussion above, a direct MD code would be much more useful for our purposes than the effective Hamiltonian FERAM code so we will continue to work in this direction.

In principle, it is possible to simulate any material using MD. In practice, the accuracy of MD simulations depends on finding correct pair potential parameters. Unfortunately, no single database exists which lists, say, Lennard-Jones parameters for every pair of elements. Such a table would not be unduly long as there are 100 elements so 10^4 table entries. Of course, we understand that the strength of, say, a O-O bond depends on what molecule it is in, but a table of average values would give a useful start. As it is, simulating a new system involves a time-consuming hunt through a sea of MD literature with inevitable confusion over notation, units and so on. Our attempts to find Cu-Pb interactions, Chapter 7, turned into a particularly lengthy saga. NPL might be the very place to host such a database of pair potentials and this would help to bring a degree of coherency to a currently fragmented subject. For example, the Martin Chaplin website contains 1600 citations to water MD simulation papers! Can each of these papers really give a fresh insight into a molecule consisting of 3 atoms? With each substance in nature is attracting thousands of papers, the MD subject area is becoming impossibly incoherent and unwieldy.

Currently, running MD requires significant manual labour. First there is a, sometimes lengthy, hunt for input parameters, as mentioned above. Then the simulation needs to be run with many obscure computational parameters (particularly in the case of TOWHEE) to give good convergence. Many thousands or millions of MD simulations must be run to obtain phase diagrams or other experimentally-verifiable quantities. We now propose some suggestions to counteract these problems. In this report, we have alleviated at least the last problem by writing driver programs such as SIMRUNNER which take some of the burden off the user and allow people without MD training to calculate measurable thermodynamic quantities. We have written and own both MDL and SIMRUNNER and now should promote these programs within NPL and possibly beyond. This, combined with an NPL-maintained database of pair potential parameters would allow non-experts to simulate substances using molecular dynamics. Such a program could even be incorporated into the MTDATA package. In the mean time we hope that this report gives a useful insight into the *realistic* capabilities of MD/MC code.

Appendix A

MDL manual and programming guide

A.1 Introduction

This chapter describes the operation of the molecular dynamics program MDL, written by Vladimir Sokhan, SSfM, NPL. The code is capable of simulating a wide range of molecular problem including gases, liquids and solids. Many pair potentials can be input including intra-molecular bonds, inter-molecular van der Waal forces, electrostatic forces and many more. The program can optionally implement constraint and/or rigid body dynamics. MDL has been written as part of Dr Sokhan's Strategic Research project in NPL and is intended for general circulation to staff members. The source code is stored under CVS (Control Versioning System) on computer `new_kirk`, module `MDL`. The latest stable version is available from:

`T:\PUBLIC\Functional\johnb\MDL.`

This chapter describes in detail how the code works and is intended for people wanted to update, generalise or maintain the code. Description is given of variables, subprograms etc. The accompanying document: MDL*: a molecular dynamics code: User Manual*, which is supplied with the code, is aimed at people who just want to use the program without needing to know its programming details. In fact, this chapter contains all the information in the *User manual* but with extra programming details.

Since the code is written internally, there is no restriction on number of users and no problem parallelising the code (the version considered here is not parallel, however.). Also, we have access to the source code, allowing us to maintain, bug fix and generalise the code when needed. Given the heterogenous nature of molecular dynamics, generalisation will probably be needed to allow different boundary conditions, forces etc. There is no such thing a "standard MD", the subject is too large. At some stage, bespoke programming is always needed for specific projects. One cannot purchase or download a "black box" MD solver which does everything.

For Quality purposes, it is good practice for several people to be involved in writing/checking program code to be certain that it works correctly. At the minimum, we should run MDL on several different machines/compilers to check it gives the same answers.[1]. Also, one compiler may spot errors missed by others.

MDL is complicated, primarily because it does a lot of things, but it must be said that the code structure could be improved for readibility. The program description part of this chapter will allow the reader to understand what the code does much more easily. The description is fairly detailed but by no means line-by-line. Large portions of the code have not been described at all here. In that case,

[1] MDL normally sets atomic velocities to be random. To input fixed initial velocities for checking, use `keycfg`=1 and the `restart noscale` command, see Section A.6

we indicate what input setting to use (or not use) to avoid such portions being executed. Thus, this chapter is a subset of a longer description to be prepared later. It allows the user to understand program operation for a fairly restricted set of input files: these restrictions are clearly stated.

A.2 Program description

MDL (Molecular Dynamics Laboratory) is a molecular dynamics simulation program written by Vlad Sokhan. It uses Newton's laws of motion (and other dynamics schemes when appropriate) to evolve the motion of a large number of atoms. From the motion (trajectory and velocities) of the atoms, the program deduces the energy, temperature and virial of the system. It also calculates the stress tensor of the system using a generalisation of the virial theorem to account for anisotropy such as that found in solids. The program has no graphical capability: it reads information about the initial coordinates of atoms, forces between them and other specifications from several text input files. It then iterates atomic coordinates for a fixed number of timsteps and writes the final position of the atoms. It also accumulates and writes out instantaneous/averaged energies etc, as mentioned above. All data is written in text files when the program finishes, the user does not interact with the code during running. However, the program can be made to continue from the result of an old run (NB: the restart file, containing atomic coordinates etc, is written in a binary, not text format). The program is written in Fortran 90 and consists of about 10,000 lines of code arranged in more than 100 subprograms.

The program allows a large range of atomic and molecular interactions (pair potential forces) to be specified. The main work of the code is to calculate the forces on all atoms of the simulation and then update their positions according to the dynamic scheme (eg, Newton's laws). In the following, a brief description of these tasks is given, which gives an idea of the generality of the program. In later sections we describe the subroutines introduced here in much greater detail.

A.2.1 Available forces

Atoms per se can interact with other atoms via electrostatic forces depending on the partial charges assigned to atoms, or through *van der Waals* forces which are specific to the type of atom pair involved and must be specified for each atom pairing. These types of force are specific to pairs of atoms (depending on their charge and type respectively) and are usually irrespective of whether atoms happen to be in molecules or not. *The exception is that the 4 nearest neighbours in a molecule do not interact by van der Waals and the interaction is excluded using "exclude lists" set by sub* `booklists`. (conversation with Vladimir Sokhan). Clamped atoms are also excluded from van der Waal or electrostatic forces. The program maintains a *Verlet neighbour list* for each atom and allows electrostatic and vdW interactions only between atoms on the list, Section A.2. Only atoms at a distance less than the cutoff r_c, Section A.7, within the minimum image convention will be on this list and so allowed to interact.

The other major type of forces that can be specified are forces *inside* molecules. These are specific to particular atoms in the molecule and do not occur externally. In a water simulation, for example, it would be possible to put, say, harmonic springs of specific strengths between the oxygen and the hydrogens and between the two hydrogens. These forces would *not* occur between different molecules. Analogous to this, one can specify intra-molecular angular bonds in which the forces depend on the angle between 3 atoms. There are also molecular dihedral forces, but I have not yet explored these. Atoms in molecules can be marked as *tethered* in which case they are linked by a force (chosen by the user from several types) to their initial positions. Finally, molecular atoms can be flagged as *clamped*, in which case they do not move during the simulation.

Somewhat analogous to molecule bonds is *molecular constraints* where atoms are at fixed distances to each other, equivalent to setting the "bond" between them infinitely stiff. This constraint considerably increases the complexity of the simulation and the algorithm used is called `SHAKE` taken from [3], Section 3.4.

As well as adding specific constraints inside molecules it is possible to set a molecule to be *completely rigid*. The molecule's motion, including rotation, is then calculated using a quaternion algorithm from [3],

Section 3.3. The inclusion of a rigid bodies command almost completely alters the code that is executed with alternative versions of most subroutines prepared for the rigid case. It could be said that MDL is really two codes depending on whether rigid bodies are used or not. It is possible to combine rigid bodies with bonds, constraints and other forces (e.g., one molecule is rigid, other are constrained, others have soft harmonic bonds etc). The program does a large number of sanity checks when setting up so should flag an error if any of these are incompatible.

In addition to van der Waals forces, it is possible to specify 3 body potential forces specfic to atom triplets irrespective of whether the atom are in molecule(s) or not. Apparently the operation of these forces is not fully implemented (conversation with Vladimir Sokhan) to we will not go into details here. Finally one can specify external forces such as gravity, applied electric field etc.

All of the types of forces above can be of several different kinds. Eg, the van der Waals forces could be Lenard-Jones, hydrogen bonds, 12-6 potential etc, the user can specify which in each case, see Section A.10, Section A.11.

A.2.2 Program operation

After initial setup, the core of the program is a loop where forces between atoms are calculated and then atomic positions, velocities and accelerations are updated. As can be appreciated from the above description, calculating the forces occupies a big part of the code. Updating the atomic positions is done using the leapfrog algorithm [3], Section 3.2. There are many ways of doing this depending on what *ensemble* the user has specified. The simplest one is the NVE ensemble in which a constant number of atoms, volume and total energy are maintained (hence, NVE). This ensemble is easy to do since Newton's laws of motion will always maintain constant total energy (kinetic + potential) for a closed system of particles (in the case where an appiled field is used, "total energy" includes the potential energy due to the applied forces, else it will not be constant). Other ensembles include NVT (constant number of atoms, volume and temperature) and NPT (constant number of atoms, pressure and temperature). These latter ensembles are achieved by interating the atoms, not according to Newton's laws but according to an alternative dynamic scheme which maintains the appropriate statistics. Several implementations have been provided for NVT and NPT and a complete set of alternative routines have been written in the case of rigid body dynamics. In total there are 11 different ensemble routines, with names such as `nve, nvt_gs, nvt_br` etc, for updating atom coordinates once the forces are known: the user picks one of these. Note that it is within these routines that the program decides whether constraint dynamics is needed and consequently calls subroutine `shake` to do this when necesary (if molecular constraints have been specified)

A.2.3 Periodic boundaries

The *volume* mentioned is more specifically the shape of a box which surrounds the atoms. The user must specify this box which can in general be a parallelapiped but simpler shapes can be specified. The program uses *periodic boundary conditions* throughout along with the *minimum image convention* so an atom near to the edge of the simulation box will likely interact with a copy of the other atoms shifted one period along. Atoms going outside the box will be moved to the other side (periodic boundary) and the distance between atoms is always calculated as the minimum distance when a whole number of box periods is added to the "true" position of the atom. So if the O in a water molecule goes out of the box it "wraps round" to the other side. The distance between O and the H's is not then the simple distance (which would be almost the box length) but the minimum distance when a period has been added/subtracted to the H positions (of order the O-H natural length). The minimum image convention applies to both van der Waals and intra-molecular forces.

A.2.4 The Verlet neighbour list

In order to minimise expensive distance calculations between atoms (for van der Waals and electrostatic force calculations), the program maintains a *Verlet neighbour list* of atom connectivities, see [3] Section 5.3, this is updated as appropriate, subroutine `verlist`. In fact, the program has another way to do this, a *link cell* method in sub `parlink`. The program decides which one to use depending on the shape of the initial box. It seems that `verlist` is used more often; if link cells are used, the program states this in an output file. *We have discovered bugs with the Link cell algorithm and recommend not using it.*

Also, for van der Waals forces, the program does not directly evaluate the forces but uses preset interpolation tables of the force magnitude versus inter-atomic distance. This is only used for van der Waals, not electrostatic, intra-molecular or external forces. Thus, the program is highly optimised for efficient evaluation of van der Waals forces which have historically been the main concern in liquid simulations. However, this does make the code correspondingly more complicated and difficult to read/maintain.

Finally, the program supports a concept called "neutral groups". If neutral groups are switched on, a whole different set of forces subprograms must be called, nearly doubling the size of the code! Since I don't know what neutral groups are, I will not mention this again and assume this is always switched off.

A.3 Program structure, possible improvements

The program is written in Fortran 90 but comparatively little use is made of F90's new data structures beyond F77. The program uses only a modest number of direct matrix operations, normally `do` loops are preferred. It does not use pointers or derived types. Modules are used, but only as a direct substitute for common blocks from F77. The program does make extensive use of C-like branching commands such as `select case, cycle, exit`. Occasional use is made of nested subprograms (ie, subroutines or functions inside other subprograms), eg, in subroutine `sys_setup`.

The main program is called `mdl` and this directly or indirectly calls all of the subprograms. There are 3 modules called `constants, sysdefaults` and `globals`[2]. These serve to provided global variables to subprograms that need them: they are `use`d in most subprograms. `constants` provides constant values such as $1/2$ and π. There are also physical constants such as the charge on an electron but note that these physical constants are given in an internal system of units specific to the program Section A.9. The user must be aware of these units to specify constants for the pair potentials the program uses. These will not be in SI.

Module `sysdefaults` contains mostly unit numbers for file handling and a few constants for array dimensioning. However, very few arrays are dimensioned at compile time, rather the arrays are marked as `allocatable`. The program reads the input files, assesses how much space will be needed, then allocates the arrays using the Fortran 90 `allocate` command. For this reason the input file `FIELD` is read twice, first to assess array sizes and then to actually fill the arrays.

Module `globals` contains all the variables which are shared between subprograms other than those passed through the calling sequence. It contains about 200 variables which are grouped, in the module, by whether they are integer or real and how many dimensions they have: 0, 1 or 2. (There are also a very few character and complex variables) The variables are not grouped by physical meaning, making it difficult to see what their uses are. *It would be better if the variables were regrouped in a logical sequence.*

Almost all the arrays in the program are passed between subprograms via module `globals`. On the other hand most scalar variables are passed through the argument list. (There are a few exceptions to both rules). This rather strange choice leads to many anomolies. For example, `nmoltp`, the number of molecule types, is used by most subprograms but is not in the `globals` module. It is therefore passed repeadly by argument to subprograms some nested many layers deep. On the other hand `ntypmol` is just used to assess the number of molecule types. It is used only briefly by 2 setup subroutines and then not refered again, but *this* variable is global!

[2] and another, `stringutils` which merely groups subprograms together in a namespace

There are many variables in `globals` which should not really be there: they are only used by a few subprograms and should be passed by argument between these. On the other hand, many of the variables which *are* passed by argument should really be global variables.

The following calls, are typical

```
call lex_fld(ng_defined,flagfrc,lbop,prmbop,keyfld,ngrp,nmoltp,n3b_types,alpha, &
             dlrpot,drewd,engunit,rcut,rcut_bop,rcmax3b)

call lex_cfg(loglnk,ng_defined,cbanner,flagpbc,ensemble,flagfrc,keyrst,flagcfg, &
             nmoltp,delr,rcut,bvolume,box)
```

In the first call, `nmoltp` is set by `lex_fld`. Sub `lex_cfg`, like most other subprograms, needs `nmoltp` so this time it is being passed in. In fact most of the variables in the `lex_cfg` argument list are passed in: its main task is not, as might be supposed to set these variables, but rather to set a large number of global variables (in module `globals`). Clearly, `ng_defined, flagfrc, rcut` could also be global, they are repeatedly passed to many subprograms. Also the input and output variables tend to be scattered randomly in the argument list.

As another example, the call

```
call forces(engcoul,vircoul,engvdw,virvdw,stress,box,bvolume,rcut,alpha,    &
            kvecx,kvecy,kvecz,flagpbc,flagfrc,istep,nsteql,nstbgr,lewald,   &
            lspme,loglnk,lgofr,numrdf,spl_order,dlrpot,drewd,epsq,dimp)
```

gives little idea of what `forces` is meant to do. The variables in the argument list are mainly intent in to the subroutine: they are variables which should really be global and are just being passed as "boilerplate" (Only variables `engcoul,vircoul,engvdw,virvdw,stress` are actually set by `forces`). In fact the main purpose of `forces` is to accummulate the variables `fx,fy,fz` which are in *globals*.

Often, there is unnecessary splitting of tasks into two or more subprograms spawning large numbers of global variables in `globals`. For instance, subroutine `sys_setup` is called which assesses size of arrays and stores these using variables in `globals` (these variables are among the comparatively few scalars in `globals`). Immediately after `sys_setup` exits, `configure_arrays` is called which actually `allocate`s the arrays in question using these sizing variables from `globals`. Since the subroutines are called in sequence it would be possible to merge these two routines together so that these sizing variables (perhaps 20!) can be removed from `globals`. (or have a separate module only `use`d by these two subroutines.)

The program readability could be improved by a massive overhaul of local/global variables and introducing several additional modules so that closely-related subprograms can have their own private storage area, shared only between them. The program is now too large to have a single global pool of variables. But even this single pool is much larger than it needs to be if the program operation were rationalised. Some specific suggestions are give in Section A.8.

It should be noted that the excessive size of `globals` is somewhat mitigated by the `ONLY` statement, eg, in `forces`

```
USE globals,   ONLY:  natms,rsqdf,drx,rx,dry,ry,drz,rz,ilist,jlist, &
                      last_neighbr,lexatm,list,nexatm
```

This states that only the specified list of variables from `globals` is used in this subroutine not all 200 of them.

A.4 Input files

The main 3 input files for the program are `FIELD`, `CONTROL` and `config.inp`. `FIELD` gives the full specification of the forces between atoms and allows atoms to be clustered into molecules. The initial coordinates of the atoms are given in `config.inp` and *these must correspond to the atoms mentioned*

in ***FIELD*** more details are given below. File `CONTROL` specifies many options for the program, most important being the ensemble to be used and the temperature. Note the temperature must be specified even for a NVE simulation, the computer then gives an initial random kinetic energy corresponding to this temperature to the atoms hence fixing the total energy which will be output. It is not possible, therefore, to specify the amount of energy directly. For NPT simulations, the pressure must also be specified. There are a number of parameters which must be given, others are optional.

Additionally, it is possible to specify van der Waals forces by preparing a look up table, rather than selecting from one of the programmed force types. In that case a file `table`, or `force.tbl` (*which?*) is also input, I am not yet clear of its format.

If the program restarts from a previous state, the file `restart` is read in (MDL *creates a file* ***restart.new****; the user must change the name of this to* ***restart*** *if a restart is needed*). File `restart.new` is always written out by the program at the end of each simulation. Note that this file contains accumulated statistics rather than atomic positions. Initial positions are always taken from `config.inp`. To continue where you left off, copy `runconf.mdl` to `config.inp`. Restarting is complicated and discussed in detail in Section A.7.

A.5 Output files

The major output file is called `output.mdl`. This contains a summary of the specifications in `CONTROL` and `FIELD` including the nature of molecules and inter/intra molecular forces. Then, for each timestep (or every timestep where data is written) the program outputs the current energy, temperature, volume, virial and pressure of the system. The energy, temperature and virial are split into several parts. E.g., energy is written as kinetic and potential parts with the potential energy further split into bond, van der Waal components etc. At the end of the simulation, the final average of all these is output along with the average stress tensor. Also, a sample of the final atomic positions is printed.

A typical printout of energies etc looks like this

```
   -------------------------------------------------------
     step     eng_tot    temp_tot     eng_cfg     vir_cfg
 time(ps)      eng_pv    temp_rot     eng_vdw     vir_vdw
 cpu  (s)      volume    pressure    eng_coul    vir_coul
 -------------------------------------------------------
        1  1.5226E+02  1.0000E+02 -2.3889E+00  1.4319E+01
    0.005  2.5059E+02  0.0000E+00 -2.3889E+00  1.4319E+01
     0.03  1.4061E+05  1.1460E-02  0.0000E+00  0.0000E+00

  rolling  1.5226E+02  1.0000E+02 -2.3889E+00  1.4319E+01
 averages  2.5059E+02  0.0000E+00 -2.3889E+00  1.4319E+01
           1.4061E+05  1.1460E-02  0.0000E+00  0.0000E+00
```

Here

eng_tot Total energy (kinetic + potential) of system plus `consv` which depends on the ensemble Section A.8.2.

temp_tot Temperature of system (K)

eng_cfg Potential energy of system

vir_cfg Total virial of system

eng_pv Enthalpy of system, $E + pV$

temp_rot Rotational kinetic energy of system. Only non-zero for rigid body motion.

eng_vdw,vir_vdw potential energy and virial due to van der Waal forces

volume Volume of system (cubic Angstroms)

pressure Pressure of system (kbar)

eng_coul,vir_coul potential energy and virial due to electrostatic forces

Energy is in the units specified by the user, kJ/mol, kcal/mol or electron volts. Note that despite some of these names, *all energies are total energies*: they are not specific energies. "kJ/mol" is simply $1000/N_A$ J of energy where N_A is Avogadro's number (see Section A.9)

In cases with intermolecular bonds, angular or dihedral forces, information about these will be output also. When tethers are used, information is given about these. If the system changes its shape, the shape angles are also output.

The full set of final positions and velocities is output in `runconf.mdl` which corresponds in format to the input file `config.inp`.

File `restart.new` gives the final set of accumulated statistical quantities to be used for restarting the code. Note that it does not contain the final atomic positions, these are in `runconf.mdl`.

Optional outputs include `rdf.mdl`, written by `print_rdf` containing the systems RDF (Radial Distribution Function); and `zprofs.txt`, written by `print_zden` containing the systems "z density".

A.6 Files `FIELD` and `config.inp`

We now consider the nature of the input file `FIELD` in more detail. To illustrate, we use the following example `FIELD` file:

```
! Major commands: units, neut, molecules, vdW, tbp, extern, close
! NB electrostatics inc Ewald are set in CONTROL
! within molecules: atoms, nummol, bonds, constr, angles, dihedr, rigid
! teth, finish
Units kJ
molecules   2
carbon monoxide
    nummols  15
    atoms    2
!   atom       mass      charge repeat clamped  neutral
       C     12.000      0.5000      1       0        1
       O     15.999     -0.5000      1       1        1
    bonds 1
       harm 1 2 1.0 2.0
finish
water
    nummols 10
    atoms    3
       H           1          0.15       1    0    0
       H           1          0.15       1    0    0
       O          16          0.25       1    0    0
    bonds 3
       harm 1  2 1.99 0.32
       mors 1  3 1. 2. 3. 4.
       quar 2  3 1. 2. 3. 4.
finish
vdW  6
```

```
      C       C  12-6      3.0000      3.8100
      C       O  buck       0.5354      3.6050
      C       H  bhm        1.2314      3.8100
      O       O  lj         1.2314      3.8100
      O       H  lj         0.2314      3.8100
      H       H  hbnd       2.0000      3.8100
close
```

We now describe the meaning of these commands and also show how these commands cause variables (mostly in `globals`) to be set. The file consists of a number of commands and may not contain blank lines. The first line is a comment which the program ignores. The major commands are `units, neut, molecules, vdW, tbp, extern, close` and the exclamation mark (`!`) can be used for comments. `close` simply terminates the file and `units` sets the energy units to be used: this then influences how the user should enter pair potential parameters etc: choices include `ev,kj,kc,int` causing `engunit` to be set to different values (see Section A.9). `neut` sets `ng_defined=true` activating neutral groups. This is a major switch in the code.

The command `molecules` allows the user to set the molecules in the system. Immediately after the command the number of molecule types is entered which is stored as `nmoltp`. This is followed by specification of the molecules (2 in this case: CO and water). Each molecule description starts with the molecule's name on its own line and then several commands pertaining to the molecule. The allowed commands for molecule specification are: `atoms, nummol, bonds, constr, angles, dihedr, rigid, teth, finish`. The `finish` command naturally finishes the specification of the molecule, `nummols` gives the number of molecules of this type (here there are 15 CO molecules specified). The other molecule commands must be followed by tables. An `atoms` command is compulsory, giving the number of atoms in the molecule. This is then followed by a table of atoms giving the name of each, its mass (in atomic units), charge compared to that of an electron, number of repetitions of this atom and then 2 flags to specify whether the atom is clamped or in a neutral group (the neutral group information is ignored if neutral groups have not been activated in `CONTROL`). These flags are 1 for "yes" and 0 for "no". The table must be of the right length as specified in the `atom` command. The repetition number is optional, for instance instead of typing two identical H commands in the water molecule specification, we could simply enter one with a repetition of 2.

The `bonds` command, if present, specifies interatomic bonds in this molecule. The command is followed by the number of bonds and then a table of appropriate length specifying the bond details. In the CO molecule, there is only 1 bond, between the C and the O. The line specifies the local number of the atom pair (here, 1 and 2), the type of bond force, here `harm`onic spring and several parameters. For the spring, just two are needed: the spring stiffness and length, in that order, see Section A.11. These are *not* in SI units, see Section A.9. There are 5 different bonds possible including `harm, more, 12-6, rhrm, quar` which cause `keybnd` to be set to `1,2,3,4,5` respectively, more on this later. The number and meaning of the real-valued parameters in a bond line depends on its type, Section A.11.

The remaining molecular commands, `constr, angles, dihedr, rigid, teth` act in a similar way but I have not yet explored these (but some description of constraints implementation is given in Section A.8.2). In terms of program operation the constraint and rigid body commands dramatically alter the program operation, while `angles,dihedr` may be taken as similar to `bonds`.

Finally, in this file, there is a van der Waals command, `vdW`. The table then lists the forces that occur between atoms of certain types. The type of pair potential should be given followed by a list of parameters specifying its nature. The number and meaning of parameters depends on the potential type, see Section A.10. The types of potential include `12-6, lj, nm, buck, bhm, hbnd, mors, tab` causing `keypot=1, 2, 3, 4, 5, 6, 7, 0` respectively. NB: the `tab` option, `keypot=0` corresponds to inputting a table from a file. If the list given is not exhaustive, the program will give a warning and set omitted potentials to zero. Details of these potentials are given in Section A.10.

In order to make sense of how this data is actually stored in variables, I have added `write` commands to the MDL source to output the relevant variable contents corresponding to the above `FIELD` file. In

terms of molecular information, we get:

```
nmoltp,ntypmol= 2 2
molnam=carbon monoxide                              water
nmmols= 15 10
nsites= 2 3
sitnam=C   O   H   H   O
sitmas= 12.000 15.999 1. 1. 16.
sitchr= 0.5 -0.5 0.15 0.15 0.25
sitfrz= 0 1 0 0 0
nugrp= 1 1 0 0 0
nvdw_types (unique)= 3
unqatm=C   O   H   ngs\JFB
ltpsit (unique)= 1 2 3 3 2
```

So, for example `nmoltp` is set to 2, the number of types of molecules. (`ntypmol` is just an estimate to size the arrays that follow.) `molnam` is a 1-D array giving the names of the molecules; `nmmols` gives the actual number of molecules of each type and `nsites` the number of atoms (sites) in each molecule. `sitnam` gives the names of the atoms in the molecules, note that this is a 1-D array and goes *across molecules*. Thus, all of the atoms mentioned in the field file are listed here, in the order that they appear. The arrays `sitmas, sitchr, sitfrz, nugrp` have the same form: these arrays can be regarded as an archive of the `FIELD` file with no processing yet done on the data. In fact `sitnam,sitchr,sitfrz,nugrp` although global are only used for a brief period. They are soon processed into another group of arrays which give information atom by atom as presented below.

Now, `nvdw_types` is the number of unique atoms that have been listed. So although we specified 5 sites, there are only 3 unique atoms, namely C, O and H listed in `unqatm`. `ltpsit` gives a mapping between the order the atoms were read in from `FIELD` and their unique number. Thus, the first atom read in was C which is marked with unique number 1, the next was O (number 2) and so on, giving the sequence 1, 2, 3, 3, 2 in this case. `ltpsit` is used to prepare "atomic" information as shown below.

Now, as mentioned, the atoms listed in `FIELD` must correspond to those in `config.inp`. The corresponing `config.inp` file for this simulation is:

```
CO, water mix
    60   0   2
 18.0000000000   18.0000000000   18.0000000000    0.0000000000    0.0000000000    0.0000000000
C    0.0000000000    0.0000000000    0.0000000000
O    2.2500000000    2.2500000000    2.2500000000
C    4.5000000000    0.0000000000    0.0000000000
O    6.7500000000    2.2500000000    2.2500000000
C    9.0000000000    0.0000000000    0.0000000000
O   11.2500000000    2.2500000000    2.2500000000
C   13.5000000000    0.0000000000    0.0000000000
O   15.7500000000    2.2500000000    2.2500000000
C    0.0000000000    4.5000000000    0.0000000000
O    2.2500000000    6.7500000000    2.2500000000
C    4.5000000000    4.5000000000    0.0000000000
O    6.7500000000    6.7500000000    2.2500000000
C    9.0000000000    4.5000000000    0.0000000000
O   11.2500000000    6.7500000000    2.2500000000
C   13.5000000000    4.5000000000    0.0000000000
O   15.7500000000    6.7500000000    2.2500000000
C    0.0000000000    9.0000000000    0.0000000000
O    2.2500000000   11.2500000000    2.2500000000
C    4.5000000000    9.0000000000    0.0000000000
O    6.7500000000   11.2500000000    2.2500000000
C    9.0000000000    9.0000000000    0.0000000000
O   11.2500000000   11.2500000000    2.2500000000
C   13.5000000000    9.0000000000    0.0000000000
O   15.7500000000   11.2500000000    2.2500000000
C    0.0000000000   13.5000000000    0.0000000000
O    2.2500000000   15.7500000000    2.2500000000
C    4.5000000000   13.5000000000    0.0000000000
O    6.7500000000   15.7500000000    2.2500000000
C    9.0000000000   13.5000000000    0.0000000000
```

```
O   11.2500000000   15.7500000000    2.2500000000
H   13.5000000000   13.5000000000    0.0000000000
H   15.7500000000   15.7500000000    2.2500000000
O    0.0000000000    0.0000000000    4.5000000000
H    2.2500000000    2.2500000000    6.7500000000
H    4.5000000000    0.0000000000    4.5000000000
O    6.7500000000    2.2500000000    6.7500000000
H    9.0000000000    0.0000000000    4.5000000000
H   11.2500000000    2.2500000000    6.7500000000
O   13.5000000000    0.0000000000    4.5000000000
H   15.7500000000    2.2500000000    6.7500000000
H    0.0000000000    4.5000000000    4.5000000000
O    2.2500000000    6.7500000000    6.7500000000
H    4.5000000000    4.5000000000    4.5000000000
H    6.7500000000    6.7500000000    6.7500000000
O    9.0000000000    4.5000000000    4.5000000000
H   11.2500000000    6.7500000000    6.7500000000
H   13.5000000000    4.5000000000    4.5000000000
O   15.7500000000    6.7500000000    6.7500000000
H    0.0000000000    9.0000000000    4.5000000000
H    2.2500000000   11.2500000000    6.7500000000
O    4.5000000000    9.0000000000    4.5000000000
H    6.7500000000   11.2500000000    6.7500000000
H    9.0000000000    9.0000000000    4.5000000000
O   11.2500000000   11.2500000000    6.7500000000
H   13.5000000000    9.0000000000    4.5000000000
H   15.7500000000   11.2500000000    6.7500000000
O    0.0000000000   13.5000000000    4.5000000000
H    2.2500000000   15.7500000000    6.7500000000
H    4.5000000000   13.5000000000    4.5000000000
O    6.7500000000   15.7500000000    6.7500000000

Notes:
No comments, line order and vertical spacing important

atoms in molecule must be listed in order of FIELD.
All molecules of one type together then all of next type etc.

First line is cbanner
Then natmcfg, flagcfg, flagpbc:
natmcfg which must equal natms got from first read of FIELD
flagcfg: 0, vel set to zero: otherwise, read in velocities too
flagpbc: 0 to 7. 0 is non-periodic, others types of periodicity

Then, on the next line: box(9) in order: 1, 5, 9, 2, 3, 6
Program sys_setup > cfgscan then sets 4=2, 7=3, 8=6
NB: box line is not read if flagpbc=0
In that case box() determined from position of atoms (sys_setup)
Box(9) altered from read-in value if keypbc=7
```

Note that all the CO molecules are listed first and atoms must be in the order C then O as in `FIELD`. Then, all the water molecules are listed in order two H and one O as in `FIELD`. That is, atoms in a molecule must be listed in order of `FIELD`. All molecules of one type are listed together then all of next type etc. Thus, the `FIELD` and `config.inp` files are closely related. Each line in `config.inp` gives the x, y, z coordinates of the atom and its name as shown. The first line of the file is a name for the simulation stored in `cbanner`. The 3 numbers on the next line are read in as `natms, flagcfg, flagpbc`. `natms` is the total number of atoms in the simulation. If `flagcfg=1` then velocities must also be specified. `flagpbc` gives the type of periodicity of the simulation box, 0 is non-periodic and, in that case, the box is simply set as the smallest (cuboid?) perimeter surrounding atoms. The next line gives the box parameters (Å) which are read into 1-D array `box` in order `box(1),box(5),box(9),box(2),box(3),box(6)`. The program then sets the remaining `box` components in the scheme `4=2, 7=3, 8=6`. Note that this line is ignored if `flagpbc=0`. The precise meaning of the `box` components depends on `flagpbc`: in this case we are simply specifying a square box of size $18 \times 18 \times 18$ Å. Box information is written in `output.mdl`

Note that comments are not allowed within `config.inp` but you can put them at the end of the file. Line breaks are not allowed.

It's worth saying at this point which subroutines do what in terms of reading in these files as it is not very obvious. The main program first calls `sys_setup` which has 3 internal subroutines: `fldscan, cfgscan, lex_cntrl` which are called in that order. `fldscan` parses through `FIELD` to ascertain array

sizes needed but *does not actually read the content of this file.* `cfgscan` does indeed read in `config.inp` and even `allocate`s memory to store the atomic positions. `lex_cntrl` opens and reads from `CONTROL`. Sub `sys_setup` then does some sanity checks and exits.

The main program then calls `configure_arrays` which `allocate`s a large number of arrays needed to store atomic bonds etc. Then, main calls `lex_fld` which opens `FIELD` again and this time reads the detailed contents. `lex_fld` sets `sitnam` etc as listed above. Then, main calls `lex_cfg` to process the information in `config.inp` and `FIELD`. NB: this sub does not actually reopen `config.inp`. `lex_cfg` generates the following 1-D arrays in our example:

```
ltype= 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2
       3 3 2 3 3 2 3 3 2 3 3 2 3 3 2 3 3 2 3 3 2 3 3 2 3 3 2 3 3 2
amass= 12.000 15.999 12.000 15.999 12.000 15.999 12.000 15.999 12.000 15.999
       12.000 15.999 12.000 15.999 12.000 15.999 12.000 15.999 12.000 15.999
       12.000 15.999 12.000 15.999 12.000 15.999 12.000 15.999 12.000 15.999
       1. 1. 16. 1. 1. 16. 1. 1. 16. 1. 1. 16. 1. 1. 16. 1. 1. 16. 1. 1. 16. 1. 1. 16.
       1. 1. 16. 1. 1. 16.
chrge= 0.5 -0.5 0.5 -0.5 0.5 -0.5 0.5 -0.5 0.5 -0.5 0.5 -0.5 0.5 -0.5 0.5 -0.5
       0.5 -0.5 0.5 -0.5 0.5 -0.5 0.5 -0.5 0.5 -0.5 0.5 -0.5 0.5 -0.5
       0.15 0.15 0.25 0.15 0.15 0.25 0.15 0.15 0.25 0.15 0.15 0.25
       0.15 0.15 0.25 0.15 0.15 0.25 0.15 0.15 0.25 0.15 0.15 0.25
       0.15 0.15 0.25 0.15 0.15 0.25
asymbl=C   O   C   O   C   O   C   O   C   O   C   O   C   O   C   O   C   O   C   O
       C   O   C   O   C   O   C   O   C   O
       H   H   O   H   H   O   H   H   O   H   H   O   H   H   O
       H   H   O   H   H   O   H   H   O   H   H   O   H   H   O
clamped= F T F T F T F T F T F T F T F T F T F T F T F T F T F T F T
         F F F F F F F F F F F F F F F F F F F F F F F F F F F F
```

All these arrays of of length number of atoms `natms`. `ltype` gives the unique number, from `unqatm,ltpsit`, of each atom in the order they were found in `config.inp`. `amass,chrge,asymbl,clamped` give corresponding properties atom by atom (`clamped` is a logical array). It is these variables which are used in the "business end" of the code, when forces are actually calculated. `sitnam` etc *are not used beyond this setup phase and it would be better if they could be removed from* `globals`.

Note that it is entirely possible to give the same type of atoms different masses and charges as has been done here for illustration. The program merely consults the above lists and does not flag inconsistencies. In fact, `ltype,asymbl` values are mainly used to decide what van der Waal force to apply, they are not consulted for atomic masses or charges.

Note also that "molecules" may be defined purely as a repeating pattern of mass, charge, etc to "paint" the atoms in `config.inp`. If commands like `bonds` were omitted the molecules would exist only in this sense. Thus it is necessary to set "molecules" even when they do not physically exist (i.e. even when chemical bonds are absent).

The bond specifications in `molecule` causes the following variables to be set

```
nbonds,nbonds= 4 45
mbonds= 1 3
keybnd= 1 1 2 5
lstbnd:
1 2
1 2
1 3
2 3
prmbnd:
100. 2. 0. 0.
199. 0.32 0. 0.
```

```
100. 2. 3. 4.
100. 2. 300. 400.
listbnd:
1 1 2
1 3 4
1 5 6
1 7 8
1 9 10
1 11 12
1 13 14
1 15 16
1 17 18
1 19 20
1 21 22
1 23 24
1 25 26
1 27 28
1 29 30
2 31 32
3 31 33
4 32 33
2 34 35
3 34 36
4 35 36
2 37 38
3 37 39
4 38 39
2 40 41
3 40 42
4 41 42
2 43 44
3 43 45
4 44 45
2 46 47
3 46 48
4 47 48
2 49 50
3 49 51
4 50 51
2 52 53
3 52 54
4 53 54
2 55 56
3 55 57
4 56 57
2 58 59
3 58 60
4 59 60
```

`nbonds` is the total number of bonds setup in `FIELD` irrespective of which molecule. Array `mbonds` gives the number of bonds per molecule. `keybnd` gives the type of each bond numbered according to the scheme above. Note that this 1-D array goes over molecules: the first entry relates to CO and the last 3 to water in this case.

Now `lstbnd` gives the bond connections again going over molecules and in the order the bonds were given in the input file. `prmbnd` is a 2-D array giving the bond parameters as found in the input file and going over molecules. Note that no interpretation is done on the meaning of these parameters at this

stage, `prmbnd` is merely a dump of the input file. Notice that some of the parameters have been multiplied by 100 (depending on what type of bond): this factor, `engunit` is dependent on the choice of units (see Section A.9).

So far, all these variables have been set by `lex_fld`. Sub `molecules`, called by `booklists` then prepares `listbnd`, of size `nbonds` $\times$ 3, from `lstbnd`. It similarly prepares: `listang` from `lstang`; `listcon` from `lstcon`; `listdih` from `lstdih`; `listtet` from `lsttet`. These relate to angles, constraints, dihedrals and tethered atoms. *`molecules` then appears to do something relating to code parallelisation.*

Columns 2 and 3 of `listbnd` contain the global atom numbers of atoms which are bonded (in our case, there are 60 atoms in total). The first column contains the order number of the bond as read from the input file (it is not the bond type, that must be found from `keybnd(column 1)`). Thus atoms 31 and 32 are connected by a bond of the second type defined in the file. From `keybnd` this is a type 1 bond, harmonic. From `prmbnd`, it has parameters 199 and 0.32.

The subroutine which evaluates bond forces is called `forces_bnd` and this uses only: `prmbnd, keybnd, listbnd, nbonds` and `fx,rx` etc from globals. Thus, it might be possible to take the intermediate stage, `lstbnd` out of the `globals` module (and also remove `lstang,lstcon,lstdih,lsttet`).

The van der Waal information is stored in the following way:

```
ntpvdw= 6
lstvdw= 1 2 4 3 5 6 0
ltpvdw= 1 4 5 2 2 6 0
prmvdw:
300. 381. 0. 0. 0.
53.54 3.605 0. 0. 0.
123.14 3.81 0. 0. 0.
123.14 3.81 0. 0. 0.
23.14 3.81 0. 0. 0.
200. 381. 0. 0. 0.
0. 0. 0. 0. 0.
```

`ntpvdw` gives the number of van der Waal potentials specified. `ltpvdw` gives the type of each bond in the order read and `prmvdw` gives parameters in the order read.

`lstvdw` gives the order the potential appeared in `FIELD` for a given pairing of unique atom index numbers in `ltype`. For example, considering a C-H pair, the unique numbers are `ai=1` and `aj=3`. We then use the formula

```
if (ai > aj)  then
  ab = half*ai*(ai - one) + aj + half
else
  ab = half*aj*(aj - one) + ai + half
endif
```

to get a single number `ab` representing the pairing. In this case, `ab=4.5` which is rounded down to 4. `lstvdw(4)=3` so the interaction parameters are is stored on the *third* line of `prmvdw` and, from `ltpvdw` is of type 5, a "bhm" bond. Comparison with `FIELD` shows that this is the correct potential between C-H.

The sub that calculates van der Waals forces is `forces_vdw`. This sub does not calculate the forces directly but instead uses lookup tables stored in `vvv,ggg` in globals. These arrays are in turn calculated by sub `gentables` called by `lex_fld`. `ggg(i,j)` contains the force magnitude at atom separation given by index `i` for pair potential `j`. Note that `j` is the *order* of the pair potential found in the input file, not its index number, cf `prmvdw` etc (`vvv` gives the corresponding potential energy contribution). `gentables` uses `ltpvdw,prmvdw` to calculate the interpolation tables. Thus, sub `forces_vdw` itself only needs `ltype, lstvdw, ggg, vvv` in order to calculated the van der Waals forces (in fact, it does use `ltpvdw` but just to see if a potential is non-existent).

A.7 The CONTROL file

A typical control file looks like this:

```
  Piezoelectric crystal: test case
temperature            300
steps               5000
timestep               0.01 ps

 ensemble  nvt hoover 0.6
# ensemble  nvt evans
# ensemble nve

cutoff         7.55
delr           2.0

collect statistics
# restart noscale
print every   100 steps

job time       1000000 seconds
close time        200 seconds
#traj     1  20  3

no elec
rdf sampling every  200 steps

#cap forces      1.0E+3
#scale every         1 step
finish
```

Lines beginning with `#,!` or a blank space are ignored. Extra vertical spaces are allowed. There are a large number of commands which set variables as follows:

timestep Simulation timestep in ps. Sets `tstep` to value, `lstep` set true.

temp sets `temp`, `ltemp=true`. Temperature (K)

pres sets `press`, `lpres=true`. Desired pressure (NPT only). This is in kbar, see Section A.9.

cut : sets `rcut`, `lcut=true`, cutoff distance for forces, r_c, Å.

delr : `delr, ldelr=true`, Verlet shell skin thickness, Å.

eps : `epsq`, rel. permittivity used in Coulombic interactions

shake : sets `tolnce`. Tolerance for shake algorithm (constraints)

quaternion : set `quattol`. Quaternion tolerance (rigid body)

job time : sets `ujobtime`, maximum job time, seconds.

close time : sets `clostim`. Time needed to finish job, seconds. Program will run for `ujobtime-clostim` seconds.

nospl : `nospl` Ewald spml interp order

steps : `nstrun`. Number of time steps

equil : `nsteql`. Number of steps until equilibrium. Note, forces are capped and velocities scaled only before equilibrium. RDFs and z densities are only accumulated after equilibrium has been reached.

restart : `keyrst=1`. Read from `restart` file, so continue statistics from where we left off. (see below)

restart scale : `keyrst=2`. Use velocities specified in `config.inp` but scaled. (does not read `restart` file)

restart noscale : `keyrst=3`. Use velocities specified in `config.inp` (does not read `restart` file)

If the `restart` command is absent, `keyrst=0` and velocities will be drawn from a Maxwellian distribution.

scale set `nstbts`, `ltscal=true`. Velocity scaling before equilibrium every `nstbts` steps.

rdf set `nstbgr`, `lgofr=true`. Calculate radial distribution function

zden `lzprofiles=true`

print rdf : `lgr=lgofr`. Can only print rdf if rdf collected

zero `loptim = .TRUE., ltemp = .TRUE., ltscal = .FALSE.`. Optimisation to minimum potential energy

print : `nstbo` (min 1) printing interval

traj : `nstraj, istraj, keytrj, ltraj=true` Output atomic positions starting at timstep `nstraj`, interval `itraj`, type `keytraj` (default 0). `keytraj=0,1,2` output positions; positions and velocities; positions, velocities and forces. `keytraj=3` output snapshot in Povray format.

ewald, spme : `lewald, lspme`. complicated options (smooth particle mesh ewald). `flagfrc=2,4` resp

coul, shif : `flagfrc = 6, 8` lforc set to TRUE (must be only one of these) These are options for electrostatics.

cap : `ifcap=true, fcap`. NB, neither SI nor MDL units. See code!

no vdw : lvdw = .FALSE. van der Waal switched off

no elec : `flagfrc=0`, switch off coulombic interactions including with external field

finish end of file (program checks all the logicals above to see if mandatory fields set) Other checks also

Note that restart does *not* cause the atoms to continue where they left off. Atomic coordinates/velocities are not stored in `restart`, only the accumulated statistics. The bare `restart` command causes `restart` to be read so statistics can continue to be accumulated where we left off. (the other two restarts do not cause this file to be read). To continue the atoms from their final positions, copy `runconf.mdl` to `config.inp`.

In the absence of a `restart` command, velocities are calculated by the program using random numbers (Maxwellian distribution) depending on the temperature *even if they have been specified in `config.inp`*. To make the program actually use the velocities in `config.inp` use one of the restart commands. If you do not want to continue statistics from a previous run, use `restart noscale`. `restart scale` uses the given velocites but scales them.

The following specifications set `ensemble` giving the ensemble type

ensemble nve : `ensemble=0`

flagfrc	Meaning
0,1	No electrostatics
2,3	Smooth particle mesh Ewald
4,5	Ewald Sum
6,7	Coulombic
8,9	truncated and shifted coulombic

Table A.1: Meaning of global variable `flagfrc`. Odd numbers means van der Waals is switched on.

ensemble nvt gau/ber/hoo `ensemble=1,2,3` NVT Gauss needs no further parameters. Berendsen and Nose-Hoover need a relaxation time parameter given as `taut`, ps.

ensemble npt ber/hoo `ensemble=4,5`. Needs two relaxation time parameters entered as `taut, taup`, ps.

ensemble nst ber/hoo `ensemble=6,7` (anisotropic version of NPT)

ensemble pmf potential mean force (NVE) `ensemble=8`

Note that it is possible to pad the input lines with other words. Eg, the RDF specification reads `rdf sampling every 200 steps`. The program reads "rdf" as the keyword, sets `nstbgr=200,lgofr=true` and ignores "sampling, every" and "steps"

The logical variables mentioned above are all initialised `false`. If the command is given they are set `true` as indicated. The program therefore checks whether compulsory values have been set, these include: `temp, timestep, rcut, delr`. If `zero` command is given this takes the place of a `temp` command.

`flagfrc` is an important option controlling the handling of electrostatic forces. It is set by `ewald, spme, coul, shif, no elec` commands, one of which must be present. If `lvdw=true`, ie, van der Waals forces are on, then `flagfrc` is increased by one. That is, if van der waals is active, `flagfrc` is odd. If it is even, then the van der Waals in `FIELD` will be ignored. NB: `lvdw` is local, it is not used outside of `sys_setup`. The full list is shown in Table 1.

The following except from `output.mdl` shows what variables have been set

```
    Simulation parameters
-----------------------------------------------

    temperature / K                    300.00      ! temp
    number of timesteps                  5000      ! nstrun
    simulation timestep                0.0100      ! tstep
    Nose-Hoover thermostat:                        ! ensemble=3
      thermostat relaxation time       0.6000      ! taut
    real space cut off                  7.550      ! rcut
    Verlet shell width                  2.000      ! delr
    equilibr. included in averages                 ! lzeql=FALSE
    data printing interval                100      ! nstbpo
    user allocated job time / s       1000000.0      ! ujobtime
    job closure time / s                200.0      ! clostim

    electrostatic potential terms off              ! flagfrc=0 (1 added later)
    RDFs will be calculated                        ! lgofr=TRUE
      binning interval                    200      ! nstbgr
```

A.8 Program operation

We now describe how the program actually operates. Some of the description was also given in Section A.6. Note: when there are parts of the code I don't understand or think are in error, I have written this in *italics*.

A.8.1 Initialisation

The main program first calls `sys_setup` which has 3 internal subroutines: `fldscan, cfgscan, lex_cntrl` which are called in that order. `fldscan` parses through `FIELD` to ascertain array sizes needed but *does not actually read the content of this file*. `cfgscan` does indeed read in `config.inp` and `allocate`s memory to store the atomic positions. `lex_cntrl` opens and reads from `CONTROL` setting the variables listed above. Sub `sys_setup` then does some sanity checks and exits.

The main program then calls `configure_arrays` which `allocate`s a large number of arrays needed to store atomic bonds etc. Then, main calls `lex_fld` which opens `FIELD` again and this time reads the detailed contents as described in Section A.6. `lex_fld` sets `sitnam` etc as listed in that Section (i.e. arrays consisting of a dump of the `FIELD` forces information including `prm...` variables). This sub also calls `gentables` to generate the interpolated pair potentials `ggg,vvv`. Then, main calls `lex_cfg` to process the information in `config.inp` and `FIELD`. NB: this sub does not actually reopen `config.inp`. As well as creating `ltype` etc, this routine makes final adjustments to `flagpbc,box` and calculates `bvolume` and `loglnk`. If the latter is true, it writes "link cell algorithm in use" in the output file.

Then, main calls `booklists` which creates several bookkeeping lists. If neutral groups are active `ng_defined` then these neutral lists are prepared. sub `quatbook` sets up quaternion booklists for rigid bodies. This also sets `ngrp` as the sum of the product of `nmmols` times `numgrp` for each molecule type. If there are no rigid bodies, `ngrp=0`.

Ignoring the neutral groups issue, the sub then calls `exclude_list` and then `exclude_link` or `exclude_atom` depending on whether `loglnk=true,false` (normally false). These subroutines are very complicated and I do not know how they work. Their purpose is to generate, eg, `lexsit` (2-D), which is an exclude list of (neighbour?) atoms associated with each atom. These appear to be pairings which will not be included in the Verlet neighbour list. The list is of length `nexsit` (1-D) for each atom. Similarly, `lexatm`, `nexatm` are generated. `booklists` then calls `molecules` which as we've seen, essentially creates `listbnd` from `lstbnd` etc.

From main, `settemp` is called. If no restart has been requested, a Maxwellian distribution of velocities if prepared *overwriting any velocity specification given in* `config.inp`. These veclocities are then scaled by `scale_vel`. Velocities are also scaled in the case `keyrst=2` corresponding to the `restart scaled` command. There is also rigid body scaling when appropriate which I will not describe. `settemp` *tinkers with the input variables* `degfre, degrot`. *This is not good programming, these variables were first set in* `booklists` *and could be set to final values there.*

Finally, `read_restart` is called which reads the restart file if `keyrst=1` otherwise it sets all accumulators to zero. *Note: the restart file is only read when a bare* `restart` *command has been issued.* This also calls `lrc` to calculate long range energy corrections stored in `englrc,virlrc`.

A.8.2 Main loop

We now enter the main loop of the code

```
do while (istep<nstrun .and. ujobtime-timelp>clostim)
   istep = istep + 1      !- increase step counter
```

The initial value of `istep` is set by `read_restart` and will be zero unless a restart was requested.

The atomic forces `fx,fy,fz` are zeroed along with various energies, virials and the stress tensor. `stres_lrc` *is set zero and never adjusted again, this should be removed.* `vertest` is called which checks to see if a new Verlet neighbour list must be prepared. If it is needed, then the list is created by one of 4

subs depending on whether neutral groups and/or link cell algorithm. Normally, `verlist` will be called. The Verlet list is stored in `list` which is a 2-D array giving a list of neighbours connected to each atom. The number of neighbours is stored in `lentry` which is 1-D. `verlist` makes use of `lexatm,nexatm` as prepared by `booklists`. *It sets `noxatm` on globals. This should be made local as it is only used here.*

An example Verlet list is shown here. Atom 1 is connected to 3-11, 15-18, 24-28, 30-31 and so on. Note that there is no duplication. Atom 1 is connected to atom 2 but atom 2 is *not* listed as connected back to 1. This is important in force calculation.

```
 1 : 3 4 5 6 7 8 9 10 11 15 16 17 18 24 25 26 27 28 30 31
 2 : 3 5 7 9 11 13 15 17 19 25 27 32
 3 : 5 6 7 8 9 10 11 12 13 18 19 20 25 26 27 28 29 30 32 33
 4 : 5 7 9 11 13 15 19 21 27 29 33 34
 5 : 7 8 11 12 13 14 15 20 21 22 26 27 28 29 30 31 32 35
 6 : 7 9 11 13 15 21 23 29 31 32 33 35 36
```

Then, `forces` is called which calculates the electrostatic and van der Waal forces in the system (in the case of neutral groups this is replaced by `forces_ng`). `forces` accumulates onto `fx,fy,fz` and also prepares energies `engcoul,engvdw` and equivalent virials. Behaviour depends primarily on `flagfrc` which Ewald and spmf being the most complex options: I will not go into these here. `force` has a loop over atoms and, if `flagfrc` is odd it calls `forces_vdw` for each atom. It pulls out the neighbour list for the atom into `ilist` and prepares a set of displacements between the atom and its neighbour on `drx,dry,drz` (these are adjusted using the minimum image convention). `ilist,drx` etc are passed to `forces_vdw` via `globals` *which is not good programming, better to pass by argument.* `forces` passes the index of the atom and the length of the neighbour list by argument to `forces_vdw`.

`forces_vdw` then calculates the force on the atom in question and also on the neighbour atoms (Newton's 3rd law). Since the Verlet list does not repeat this does not overcount. `forces_vdw` uses `drx, dry, drz, rsqdf, ilist, ltype, lstvdw, ltpvdw, vvv, ggg` from global, see Section A.6. From the neighbour list, only neighbours less than `rcut` distance away are considered.

In the case, for example, `flagfrc=6,7`, sub `coulomb` is called to do the electrostatic forces. This also makes use of `ilist,drx` etc and forces are only applied on atoms separated less than `rcut`. The force magnitude depends on charges from `chrge`. `clmshf` is called similarly in case `flagfrc=8,9`. `forces` accumulates energy and virial contributions passed back from `forces_vdw` etc. But these routines themselves accumulate `stress` which `forces` simply passes through to them.

Back in the main program, `forces_bnd` is called if `nbonds` is non-zero. As described in Section A.6, this uses `prmbnd, keybnd, listbnd, nbonds` to calculated atomic forces. *This routine sets atomic distances on globals `xdab,ydab,zdab` but these are not needed by other routines.* It seems they are global purely to conserve memory. In that case they should be in a different module earmarked for that purpose.

`forces_bnd` also calculates an energy, virial contribution, `engbnd,virbnd` and also accumulates onto `stress`. Similar subroutines are called to prepare other forces: `forces_3b, forces_dih, forces_ang, forces_dih, tethfrc, forces_ext`. *In `forces_ext`, the energy is not set due to gravity or applied electric field. The virial `virfld=0` always.* Note that the external electric field is applied even when `no elec` has been selected in `CONTROL`.

The program then calls `clampatoms` to zero forces on clamped atoms. Long range corrections are added:

```
!- long range corrections (energy and virial)
engvdw = engvdw + eng_lrc
virvdw = virvdw + vir_lrc

!- add long range correction to diagonal terms of stress tensor
stress(1) = stress(1) - onethird*(vir_lrc + vlrcm(1))
stress(5) = stress(5) - onethird*(vir_lrc + vlrcm(1))
stress(9) = stress(9) - onethird*(vir_lrc + vlrcm(1))
```

Note that `vlrcm=0` always. This should be removed

In the case, `ifcap=true` and before equilibrium, `cap_forces` is called to make sure no force is more than `fcap`. If `loptim=true`, `zerokelvin` adjusts forces so the system tends towards a position of minimum potential energy. *Question: Does this account for all forces including external?*. The total virial is created as

```
virtot = vircoul + virvdw + virbnd + virtbp + virfld + virang + virtet
```

`virfld`, always zero, is not used. `virtot` is not written out but passed to some of the NPT ensemble routines.

Then one of 11 ensemble subroutines are called to update the atoms positions. Which one is needed depends on `ngrp=0`, ie if rigid bodies are present, and `ensemble`. Here, we only comment on `nve`. The routine updates `rx,ry,rz,vx,vy,vz` based on the forces creating intermediate velocities stored as globals `uxx,uyy,uzz`: *it is not clear why these are global*. The kinetic energy `engkin` is updated and a velocity contribution added to `stress`. Periodic boundary conditions are applied to map the atomic positions back in the box.

If constraints are in operation, `nve` calls `shake`. Before doing so it prepares `dxx,dyy,dzz`, from `listcon` which is passed by `globals` to `shake` (*better to pass by argument*). Several variables apparently acting as locals in `nve` are for some reason in `globals`: `drx,dry,drz,uxx,uyy,uzz`. Similarly, it is not clear for whose benefit `shake` sets the following variables: `xxt,yyt,zzt,dxt,dyt,dzt`. *These should all be made local.*

Having called `shake`, `nve` then corrects the intermediate velocities `uxx` etc and forces `fx` etc.

Main then applies temperature scaling if `ltscal=true` but only before equilibrium. It calls `scale_vel` to do this. In NPT ensembles, it similarly scales positions.

Then, `averages` is called which calculates total energy, virial etc and many other quantities written to the output file. It stores these in an array `stpval` with the following meaning:

```
stpval( 1)  = stpcns/engunit              !- total energy
stpval( 2)  = stptmp                      !- instantaneous temperature
stpval( 3)  = stpcfg/engunit              !- configurational energy
stpval( 4)  = stpvir/engunit              !- configurational virial
stpval( 5)  = engbnd/engunit              !- bond energy
stpval( 6)  = virbnd/engunit              !- bond virial
stpval( 7)  = rad2grad*acos(dimp(6))      !- alpha  (cos(b,c)
stpval( 8)  = engtet/engunit              !- energy of tethered atoms

stpval( 9)  = stpeth/engunit              !- P*V energy (enthalpy)
stpval(10)  = stprot                      !- rotational temperature
stpval(11)  = engvdw/engunit              !- vdW energy
stpval(12)  = virvdw/engunit              !- vdW virial
stpval(13)  = (engang + engtbp)/engunit   !- energy due to intramolecular angles
stpval(14)  = (virtbp + virang)/engunit   !- virial due to intramolecular angles
stpval(15)  = rad2grad*acos(dimp(5))      !- beta  (cos(a,c)
stpval(16)  = virtet/engunit              !- virial due to tethered atoms

stpval(17)  = stpvol                      !- volume
stpval(18)  = stpprs                      !- instantaneous pressure
stpval(19)  = engcoul/engunit             !- Coulomb energy
stpval(20)  = vircoul/engunit             !- Coulomb virial
stpval(21)  = engdih/engunit              !- energy of dihedrals
stpval(22)  = vircon/engunit              !- virial of constraints
stpval(23)  = rad2grad*acos(dimp(4))      !- gamma  (cos(a,b))
```

The following code is used to generate some of the above `stpval` values

```
stpcfg = engvdw + engcoul + engbnd + engang + engdih + engfld + engtbp
stpvir = virvdw + vircoul + virbnd + virang + vircon + vircom + virtbp + virtet
stpeng = stpcfg + engkin + engrot
stprot = two*engrot/(boltz*max(one,degrot))
stptmp = two*(engkin + engrot)/(boltz*degfre)

stpprs = (two*engkin - stpvir)/(3.0_rdp*stpvol)

stpeth = stpeng + stpprs*stpvol
stpcns = stpeng + consv
stpprs = stpprs*prsunt    !- convert in kbar
```

We see that `stpcfg` is the total potential energy of the system. `stpeng` is the total energy, `stpcns` is this plus `consv`. `consv` is normally zero. It is set to zero at the start of main and changed only by NVT and NPT ensemble routines `nvt_nh,nvt_nhr,npt_nha, npt_nhi, npt_nhr`. `stpval` is then written to output by main: it is also used to prepared rolling averages in `sumval`. These are written out only at the end of the program by sub `summary`.

Trajectory data is then written out depending on `ltraj` and a restart file is dumped every `ndump` timesteps. If the equilibration period has ended, this is written out and temperature scaling is switched off, `ltscal=false`.

The main loop then terminates and all that is left to do is call `summary`, close files and exit.

A.9 Units

MDL does not use SI units. The fundamental units are

- Length – 10^{-10} m (Angstrom, Å)
- Time – 10^{-12} s (picosecond)
- Mass – $1.66053886 \times 10^{-27}$ kg (atomic mass, u)
- Temperature – Kelvin (K)
- Charge – $1.60217653 \times 10^{-19}$ C, (elementary charge, e)

Thus, we might specify 2 Å length or 2 u mass. Here A and u can be regarded as unit symbols, replacements for m (metres) and kg (kilograms).

From these it is possible to derive all internal units. A systematic way to do this is introduce variables A, ps, u, e as conversion factors with values given above. Let us say we have a time $t = 2$ s. We have

$$t = \frac{2 \text{ (s)}}{ps \text{ (s/ps)}} \text{ ps} = 2 \times 10^{12} \text{ ps} \tag{A.1}$$

Say we have an energy $E = 1$ J, $E = 1$ kg m^{2}s^{-2}. In that case

$$E = \frac{1}{uA^2ps^{-2}} \text{uA}^2\text{ps}^{-2} = \frac{1}{1.66 \times 10^{-27}(10^{-10})^2(10^{-12})^{-2}} = \frac{1}{1.66 \times 10^{-23}} = 6.02 \times 10^{22} \text{ uA}^2\text{ps}^{-2} \tag{A.2}$$

In MDL, the 1 J energy must be entered as 6.02×10^{22}. We can introduce the unit of energy as

$$E_0 = 1.66053886 \times 10^{-23} \text{ J} \tag{A.3}$$

Thus, 1 E_0 = 1 uÅ^2 ps^{-2}. We thus have an additional variable E_0 with this value. Similarly, we define the unit of pressure as

$$p_0 = 1.66053886 \times 10^7 \text{Pa} \tag{A.4}$$

A more complex example is the permittivity of free space ϵ_0. In SI we have

$$\epsilon_0 = 8.854 \times 10^{-12} \text{ F/m} \tag{A.5}$$

Now a Farad (F) is 1 C^2/J so this becomes

$$\epsilon_0 = 8.854 \times 10^{-12} \text{ } C^2 J^{-1} m^{-1} \tag{A.6}$$

We therefore obtain

$$\epsilon_0 = 8.854 \times 10^{-12} \frac{1}{e^2 E_0^{-1} A^{-1}} \text{ } e^2 E_0^{-1} A^{-1} = 5.743 \times 10^{-7} \text{ } e^2 E_0^{-1} A^{-1} \tag{A.7}$$

The constant $f_0 = 1/4\pi\epsilon_0$ is thus entered into MDL as

$$f_0 = 138935.456 \text{ } E_0 A e^{-2} \tag{A.8}$$

see `constants.f90`. The only other fundamental constant is the Boltzmann constant which is given as

$$k_B = 0.8314472 \text{ } E_0 K^{-1} \tag{A.9}$$

MDL uses this system of variables internally. Since the system is consistent, all SI equations are also valid in MDL units.

However, there is further complexity in that the user can enter different energy units in the input files. MDL will then scale force field parameters (from `FIELD`) dependent on energy to give the internal units. Note that all calculation is done on the internal units, MDL scales immediately after inputting the values (although the values are echoed to the output file before the conversion).

The allowed energy input units are kJ/mol, kcal/mol and eV. The first two terms are misleading as energy density is not output, MDL always outputs total energy. By "kJ/mol" Dr Sokhan simply means $1000/N_A$ J of energy where N_A is Avogadro's number for a gram mole 6.03×10^{23}. This is simply $100E_0$ so that variable `engunit` is set to 100 when kJ/mol is chosen. field parameters, eg, Lennard Jones parameters, are then scaled up by `engunit=100` to give the internal units. (see, `lex_fld` for the two other options). When energies are output, they are similarly divided by `engunit`.

To make life still more complicated, pressures are input and output in katm (though Dr Sokhan refers to this as kbar) rather than the internal units p_0. The constant `prsunt=0.1639` converts between the two and this is done immediately after the pressures are input.

A.10 van der Waals potentials

There are currently seven van der Waals forces supported by MDL with 2-5 parameters needed for each. These are specified in sub `gentables`. The meaning of the parameters is given in Table 2. Note that some of these parameters will be multiplied by `engunit`. Thus, in the example file of Section A.6, we specify a O-O Lennard-Jones interaction with $\epsilon = 1.2314 \times 100$ E_0 (since kJ were selected for the energy unit) and $\sigma = 3.81$ Å.

A.11 Intra-molecular bonds

There are currently 5 bonds available with 2-4 parameters each, as shown in Table 3. Forces and energies due to these bonds are calculated directly in `forces_bnd`. Again, the program will multiply some of these parameters by `engunit` to get them into E_0 units.

A.12 Known bugs

Sub `forces_ext` does not add energy due to external electric or magnetic fields.

ltpvdw	specifier	name	parameters	pair potential (E_0)
1	12-6	12-6 potential	a,b	$(a/r^6 - b)/r^6$
2	lj	Lennard-Jones	ϵ, σ	$4\epsilon(\sigma/r)^6[(\sigma/r)^6 - 1]$
3	nm	n-m potential	a,b,c,d	$a/(b-c) \cdot (c(d/r)^b - b(d/r)^c)$
4	buck	Buckingham	a,b,c	$a\exp(-r/b) - c/r^6$
5	bhm	Born-Huggins-Meyer	a,b,c,d,e	$a\exp(b(c-r)) - d/r^6 - e/r^8$
6	hbnd	H-bond	a,b	$(a/r^2 - b)/r^{10}$
7	mors	Morse potential	a,b,c	$a[(1 - \exp[-c(r-b)])^2 - 1]$

Table A.2: van der Waals potentials. The parameter column shows the list of parameters in sequence which should be given in `FIELD` file. So a 12-6 potential must be entered `12-6 a b`

keybnd	specifier	name	parameters	pair potential (E_0)
1	harm	Harmonic bond	k,l	$\frac{1}{2}k(r-l)^2$
2	mors	Morse potential	a,b,c	$a[(1 - \exp[-c(r-b)])^2 - 1]$
3	12-6	12-6 potential	a,b	$(a/r^6 - b)/r^6$
4	rhrm	restrained harmonic	a,b,c	$\frac{1}{2}a\min(\lvert r-b\rvert, c)^2 + ac\max(\lvert r-b\rvert - c, 0)$
5	quar	quartic potential	a,b,c,d	$\frac{1}{2}a(r-b)^2 + \frac{1}{3}c(r-b)^3 + \frac{1}{4}d(r-b)^4$

Table A.3: Intra-molecular bonds. The parameter column shows the list of parameters in sequence which should be given in `FIELD` file. So a harmonic potential must be entered `harm k l`

A.13 New features

There are several new commands available in the CONTROL file. These commands must be added *after* the `finish` command; they are implemented in `mdl.f90`

debug writes variable values, described in Section A.6, to file `debug.txt`. Also writes the verlet list to file `verlist.txt`. The full Verlet list is written for each timestep and followed by the timestep (integer) on a line by itself.

ghost Allows calculation of chemical potentials for each "ghost atom". Must be followed by a line with the number of ghost atoms and the stride between chemical potential calculations, then a table of atoms and replacement atom names. Each entry in the table is the ghost name, its replacement atom name and the number of ghost atoms in total. The final set of chemical potentials is written to `output.mdl`. *Note that chemical potentials can only be calculated for single-atom molecules.* Only van der Waal energies are accounted for when calculating chemical potential.

To give an example of the chemical potential calculation, suppose the `FIELD` file is

```
molecules 3

MoleculeA
   nummol 20
   atoms 1
      A 39.0 0.0
finish

MoleculeB
   nummol 20
   atoms 1
      A 39.0 0.0
```

```
finish

MoleculeC
   nummol 20
   atom 1
      A 39.0 0.0

MoleculeAX
   nummol 20
   atoms 1
      AX 39.0 0.0  1 1  <=== clamped
finish

MoleculeBX
   nummol 20
   atoms 1
      BX 39.0 0.0  1 1  <=== clamped
finish

MoleculeCX
   nummol 20
   atoms 1
      CX 39.0 0.0  1 1  <=== clamped
finish

vdw 21
    A    A    lj   1.0  3.0
    A    B    lj   2.0  3.0
    A    C    lj   3.0  3.0
    B    B    lj   2.0  3.0
    etc

    A    AX   lj   0.0  0.0
    A    BX   lj   0.0  0.0
    AX   AX   lj   0.0  0.0
    etc

close
```

then the ghost atom specification could be:

```
ghost
3  10          ! no ghosts, stride between updates
  AX    A    20
  BX    B    20
  CX    C    20
```

Normally, the ghost atoms, AX, BX, CX do not interact (as can be seen from the FIELD file). However, every 10 timesteps (in this case) after equilibrium, the ghost vdW interactions AX-A, AX-B etc are replaced with real interactions A-A, A-B etc (this is done from the table above, in `CONTROL`). Note that self interactions AX-AX etc are *not* turned on. The potential energy including these new interactions is calculated and the potential energy difference is used to calculate the chemical potentials

of AX, BX, CX (ie of A, B, and C). The effect is as if an extra set of atoms of type A, B, C was inserted into the system every so often and potential energy differences calculated. These ghost atoms do not truly interact with the other atoms and do not affect the trajectories of "real" atoms. *Note: only van der Waal energy differences are considered. Intra-molecule forces are not accounted for. Only the chemical potentials of single-atom molecules can be calculated*

Note that the ghost atoms are just ordinary atoms as far as MDL is concerned, *it is up to the user to set their interactions to zero.* The user may also chose to clamp the atoms, as shown in the above `FIELD` file. This is not strictly necessary as the ghost atoms do not interact and will not move unless given non-zero initial velocities. The ghost atoms must be accounted for in the config.inp file, according to the usual rules Section A.6.

This method works by replacing columns in the `vvv` and `ggg` arrays. As mentioned, Section A.6, these arrays are look-up tables of van der Waal potential energy and forces between different atom types. Every `nghoststr` steps after equilibrium, MDL does the usual forces calculation and the result is stored in `fx,fy,fz`. The potential energy is also calculated and stored, as usual in `engcoul` and `engvdw`. These force arrays are then copied for safe keeping in `fxcopy,fycopy,fzcopy`. Then, columns in `vvv` and `ggg` are changed so that eg, `AX-A` becomes like `A-A` etc, according to the specification in `CONTROL`. The force subroutine is called again. The only thing we are interested in is the vdw and Coulombic potential energy. The difference between these energies and their previous values is used to calculate the chemical potential. `vvv` and `ggg` are then reset and we go on to the next ghost atom (note that only one ghost species is changed at a time), allowing us to contribute to the next chemical potential and so on. When all chemical potentials have been set, the forces `fx,fy,fz` are put back to their original values and MDL goes on to update atom positions.

Appendix B

Description of feram

B.1 Introduction

Feram is a molecular dynamics code specifically designed for analysing perovskite ferroelectric systems. Rather than iterating the motion of each atom in each unit cell, the code uses a transformation whereby the lattice is reduced to a cubic lattice with a single atom (refered to as a "dipole") in each unit cell. The atom is specified by generalised coordinates: in fact the program maintain the atom's displacement from a mean position. These quasi atoms are then iterated according to interatomic forces using Newton's laws of motion (in fact Nose constant temperature iteration but still classical). It is not clear how reducing the unit cell to a single atom can maintain the physics, it seems likely that some information is lost. The authors would presumably argue that such internal degrees of freedom are unimportant for the P-E and P-strain loops the program sets out to calculate. The lattice actually considered by the program is a cubic lattice specified completely by the number of cells `Lx`, `Ly`, `Lz` and the lattice constant `a0`. Because the reduced lattice is cubic, it seems probably that the program can only handle systems with cubic lattices (eg perovskites). All examples given with the code are for $BaTiO_3$.

The quasi atoms iterated by the program have an effective mass and are coupled by effective potentials. These must be calculated from the real lattice in a separate calculation, Feram does not do this. To calculate the effective mass, for example, involves solving for the normal modes (phonons) of the lattice. A program such as GULP should be able to do this. The authors have calculated an effective mass of 39 amu for $BaTiO_3$.

The program does not calculate forces between atoms in the conventional way but rather using a Fourier transform. This assumes (maximum) third nearest neightbours connectivity. The Fourier transform is done once in the simulation to form a matrix and then forces can be quickly derived from this matrix at each timestep. This technique only works for the harmonic part of the force/energy (see below). The unharmonic parts must be processed at each time step in the traditional way. The authors claim a cost reduction from $O(N^2)$ to $O(N \log N)$ for force calcultion, but note that this speedup is only for the harmonic forces (there are many other forces as detailed below). Having obtained the forces, the simulation proceeds conventionally: momenta are updated from the forces and displacements from the momenta at each timestep.

B.2 Forces and energies

The dipole-dipole potential energy of the system is given by the 4th order polynomial:

```
P_4(u) = kappa2*u^2 + alpha*u^4+ gamma*(u_y*u_z+u_z*u_x+u_x*u_y)
where u^2 = u_x^2 + u_y^2 + u_z^2
```

for each atom. The total potential energy is the sum of these. Notice that the energy depends only on the individual displacements not terms like (u2-u1)**2 indicating use of generalised coordinates. These variables are entered as

```
P_kappa2, P_alpha, P_gamma
```

in the input file.

Note: The `kappa2*u**2` term is harmonic and processed by `dVddi_dipole_dipole` (with pre-processing in `dipole_dipole_short_range`, which actually uses `P_kappa2`) while the remaining unharmonic terms is processed by `dVddi_unharmonic` (these subroutines accumulate forces and potential energy). `dVddi_E_field` adds energy and forces due to the external electric field (it calls `dipole_E_field_energy` in `energy_module`).

The user can also enter short-range inter-site interaction coefficients `j` and elastic coefficients `B11, B12, B44, B1xx, B1yy, B4yz`. These elastic coefficients are pre-processed by elastic and then used in the MD by `optimize_homo_strain` and `optimize_inho_strain`. The forces and energy obtained are then added to give total force and energy.

The pressure entered by the user `p%GPa` is converted to `p%pNa03` which is only used in optimize-homo-strain.

B.3 Program description

The program is written in Fortran 90. There are a few compiler directives (beginning #). These cause conditional compilation based on the compiler being used (see feram.f and fft.f in particular, search for "#"). There are no global variables, instead use is made of large user defined types to store simulation specifications. Variables of these types are passed through as actual arguments to most subprograms.

The file `define.h` is included in most subprograms and contains compiler constants.

Library functions used: `derfc, iargc, getarg` (see `decay-functions.f, feram.f`).

The program also comes with two directories `libblaslapack` and `libbuilt_in_fft` containing source code which is compiled along with the main program. Program `fft.f` in the main source calls `fft3d` in `libbuilt_in_fft/`. I have not seen any of the functions in `libblaslapack/` being called.

The tokens `__FILE__, __LINE__` appear in the source code as arguments to msg. These tokens are interpreted by the C preprocessor and correspond to the source code file and current line being executed. They are not legal Fortran names. FERAM produces lots of debugging information of this sort while it runs, giving very large file sizes. An option to decrease this verbosity would be useful!

B.4 Using feram. Operation of code

To run feram, use

```
feram filename
```

where filename is the main input file. Other input files include `restart.coord` for optional restart, `local.field` (optional) and `ionic.configuration` (not implemented). Output files have the form `filename.coord, filename.fft, filename.avg, filename.param` etc.

Modules include `average_module`, `energy_module, param_module` and `coord_module`. These modules contain defined types (structures) and subroutines which operate on these types (in an object oriented style, but note that the majority of subprogram are external not in modules). These modules do NOT contain global variables: all inter-program communication thus occurs via the argument list. To facilitate this, user types `param_type` and `coord_type` have been defined in their respective modules. `type(param_type) p` is a variable, defined in the main program, which contains all the user input from the input file. `type(coord_type) c` contains the entire state of the system at any instant including atomic coordinates, momenta and forces. Variables `p` and `c` are passed through to most subprograms.

B.5 System setup

The main program, feram, is in `feram.f`. It first calls `read_param` (in `module param_module`) which reads and parses the input file. Then `make_param sets` some elements of `type(param_type) p` in terms of each other. `write_param` writes various details of the input file out. `alloc_coords` (in `coord_module`) allocates memory for the various array members of `type(coord_type) c`. (Note in F90/95 derived type members may not be allocatable arrays, this is allowed in F03. Compiler directives in this file use either allocatable arrays or pointers depening on the compiler).

Then, subroutine potentials sets various details of the potentials and forces used in the simulation. This subroutine merely calls `dipole_dipole` and `elatic`. `dipole_dipole` calls `dipole_dipole_long_range` and `dipole_dipole_short_range` and adds the results from each to form 4th rank array stored as `c%dipoI`. If the user requested (`p%plot_dispersion`), it calls `print_eigenvalues` for short and long range forces and the sum of both.

```
potentials
dipole-dipole
dipole_dipole_long_range (electrostatics, may not be fully implemented)
dipole_dipole_short_range (contribution from c\%j parameters)
print_eigenvalues (writes to file *.dat)
elastic
```

`dipole_dipole_long_range` sets TMPreal based on lattice vectors and functions `B_decay` and `C_decay` (file `decay_functions.f`), and based on `p%kappa` (note this is NOT `kappa2`). It then fourier transforms to `TMPcomplex` using

```
call fft_r2c_3x3upper(p,TMPreal, TMPcomplex)
```

long_range (return value of subroutine) is then set to real part of `TMPcomplex`. The subroutine then performs a reciprocal space calculation and adds this to `long_range`. The key physics is in

```
kG(3) = two_pi_over_a*(k(3)+nz)
kG(2) = two_pi_over_a*(k(2)+ny)
kG(1) = two_pi_over_a*(k(1)+nx)
kGkG = kG(1)**2 + kG(2)**2 + kG(3)**2
long_range(ix,iy,iz,alpha,beta) = &
          & long_range(ix,iy,iz,alpha,beta) &
          & + four_pi_over_unit_cell_volume * kG(alpha) * kG(beta) &
          &   * exp(-kGkG/four_kappa2) / kGkG
```

where k is a wavevector.

`dipole_dipole_short_range` sets short range coupling depending on whether we are dealing with first, second or third nearest neighbours. Var `TMPreal` is set depending on the `p%j` parameters and `P_kappa2`. It then calls

```
call fft_r2c_3x3upper(p, TMPreal, TMPcomplex)
```

and returns the real part of `TMPcomplex`

Subroutine `elastic` sets the `c%homogenous` and `c%homogenous_coupling2` elements based on the `p%B11` etc values. It also sets `c%inhomo_K` and `c%inhomo_coup2_K` values based on `p%Lx` etc and `p%a0`.

Main then calls `make_mass_matrix` which sets the `c%m_inv` components based on `p%film_thickness` and `p%gap`.

B.6 Molecular dynamics loop

Main then calls molecular_dynamics which is the main part of the code.

```
molecular_dynamics
initialize_E_field
initialize_dipoR
dipoFFT
dVddi_dipole_dipole (harmonic part of forces)
dVddi_unharmonic
dVddi_E_field
optimize_homo_strain
dVddi_homo_coupling
optimize_inho_strain

write_Energy
Nose_Poincare_Hamiltonian
reset_Average

(MD timestep loop)
nose_poincare
dipoFFT
dVddi_dipole_dipole
dVddi_unharmonic
dVddi_E_field
optimize_homo_strain   (calculates strain)
dVddi_homo_coupling
optimize_inho_strain

write_system (in module coords_module)
write_fft    (in module coords_module)
write_Energy
add_to_Average
(end timestep loop)

divide_and_write_Average
```

`initialize_E_field` reads the local field from file `local.field` if it exists and adds this to the external field (p%external_E_field) to form c%E_field.

Sub `initialize_dipoR` checks if `restart.coord` exists. If it does, it loads the atomic displacements `c%dipoR` from this file. Otherwise it sets the displacements using a random normal distribution from `p%init_dipo_avg, p%init_dipo_dev`. Note the no of lattice sites is `Lx*Ly*Lz` (from p). Initial momenta `c%dipoP` are set to zero. Several corrections are made for defects and mirrors and for 'film' and 'epit', see `p%defect_position` and `p%bulk_or_film/`. The initial kinetic energy `e%dipo_kinetic` is set. We then

```
call dipoFFT(p,c)
call   dVddi_dipole_dipole(p,c,e)
call      dVddi_unharmonic(p,c,e)
call         dVddi_E_field(p,c,e)
call  optimize_homo_strain(p,c,e)
call  optimize_inho_strain(p,c,e)
```

`dipoFFT` calculates `c%didiR, c%dipoK and c%didiK` from `c%dipoR`

`dVddi_dipole_dipole` uses `c%dipoI` and `c%dipoK` to form a temporary vector which is FT'd to form `c%dVddi`. This is the force due to the harmonic springs. Corrections are again made based on `p%bulk_or_film`. The harmonic energy, `e%dipole_dipole` is set.

`dVddi_unharmonic` adds to `c%dVddi` based on `c%dipoR, p%P_alpha and p%P_gamma`. It prepares `e%unharmonic`

`dVddi_E_field` adds to `c%dVddi` based on `p%Z_star and c%E_field`

Back in `molecular_dynamics`, we enter the main loop

```
do i_step = 1, p\%n_thermalize+p\%n_average
```

This iterates over time steps and prints out the atomic coordinate and ffts every so many steps (`call write_system and call write_fft`). Averages are accumulated and written out at the end using `call divide_and_write_Average`. The main function is `nose_poincare` which updates atomic positions, momenta and forces.

`nose_poincare` updates atomic momenta using

```
c\%dipoP(:,:,:,:) = c\%dipoP(:,:,:,:) - (p\%dt_2*c\%s_Nose) * c\%dVddi(:,:,:,:)
```

It updates displacements using

```
 c\%dipoR(:,:,:,:) = c\%dipoR(:,:,:,:) &
    & + p\%dt_2 * (1/c\%s_Nose+1/s_Nose_old) * c\%dipoP(:,:,:,:) * c\%m_inv(:,:,:,:)
```

Then

```
call dipoFFT(p,c)
call  dVddi_dipole_dipole(p,c,e)
call     dVddi_unharmonic(p,c,e)
call        dVddi_E_field(p,c,e)
call optimize_homo_strain(p,c,e)   ! It calls dVddi_homo_coupling().
call optimize_inho_strain(p,c,e)
```

Just as in `initialise_dipoR` but based on the new displacements `c%dipoR`. This sets the "forces" in `c%dVddi`. The subroutine also calls `Nose_Poincare_Hamiltonian (in energy_module)` to calculate the energy of the system.

The form of the `.avg` output file is:

```
 write(UNIT_AVG,"(f6.1,3f9.5,26e13.5)")&
      p\%kelvin, p\%external_E_field(1:3), avg\%strain(1:6), avg\%u(1:3), avg\%uu(1:6),&
      avg%energy%dipo_kinetic    ,&
      avg%energy%dipole_dipole   ,&
      avg%energy%dipole_E_field  ,&
      avg%energy%unharmonic      ,&
      avg%energy%homo_strain     ,&
      avg%energy%homo_coupling   ,&
      avg%energy%inho_strain     ,&
      avg%energy%inho_coupling   ,&
      avg%energy%total           ,&
      avg%energy%H_Nose_Poincare ,&
      avg%e2
```

B.7 Physics in Feram

We now compare the program functionality to a paper published by the author, [53]. In the following, equation numbers refer to this paper.

dipole_dipole_long_range fills its real matrix based on functions B_decay and C_decay which depend on kappa. Note that this is *note* kappa2 as mentioned in the paper eq (8). It is related to Ewald summation and set as

```
p\%kappa = sqrt(M_PI) / p\%a0 / (dble(p\%Lx*p\%Ly*p\%Lz))**(1.0d0/6)
```

The decay functions are

```
real*8 function B_decay(r,rr,kappa)
  implicit none
  real*8 r,rr,kappa,derfc
  real*8, parameter :: M_2_SQRTPI = 1.128379167095512573 90d0
  B_decay  =  derfc(kappa*r)/(r*rr) + M_2_SQRTPI * kappa * exp(-kappa**2*rr) / rr
end function B_decay

real*8 function Crr(r,rr,kappa)
  implicit none
  real*8 r,rr,kappa,derfc
  real*8, parameter :: M_2_SQRTPI = 1.128379167095512573 90d0
  Crr=3*derfc(kappa*r)/(r*rr) + M_2_SQRTPI * kappa * exp(-kappa**2*rr) * (2*kappa**2+3/rr)
end function Crr

real*8 function C_decay(r,rr,kappa)
  implicit none
  real*8 r,rr,kappa,Crr
  external Crr
  C_decay=Crr(r,rr,kappa)/rr
end function C_decay
```

They are not mentioned in the paper. derfc looks like the error function. We have not found this function in the code.

the following reciprocal space terms are then added on to long_range

```
long_range(ix,iy,iz,alpha,beta) = &
   & long_range(ix,iy,iz,alpha,beta) &
   & + four_pi_over_unit_cell_volume * kG(alpha) * kG(beta) &
   &   * exp(-kGkG/four_kappa2) / kGkG
```

The real and reciprocal space summations in dipole_dipole_long_range appear to be Fourier transforms of (10) and take account of long ranging electrostatic interactions via Ewald summation to give V^{dpl}. Note that $\Phi_{alpha,beta}$ of (9) is not calculated directly but rather a Fourier transformed version thereof. Back in dipole_dipole, this is added to the short range contribution to get Phi^{quad} of (13)

dipole_dipole_short_range adds contributions depending kappa2 (8) and the J parameters c%j(7). This contributes to Φ^{quad} in (13). On return, dipole_dipole adds the results from long and short range terms to form Φ^{quad} of (13), stored as dipoI. This Φ^{quad} contains the u^2 terms from (8), (9) and (12).

Subroutine elastic forms $\Phi^{elas,inho}$ called inhomo_K from B11, B12, B44 (17-19) and $\tilde{B}$ stored as inhomo_coup2_K from B1xx, B1yy, B4yz (23). All this is done directly in reciprocal space with no need of Fourier transforms. It also forms a matrix relating to (16) (stored as homogenous) and the C-matrix of (21) (stored as homogenous_coupling_2)

dvddi_dipole_dipole makes use of dipoI (which is just Φ^{quad} in (13)) to calculate the harmonic forces on atoms (contribution to dVddi) by Fourier transformation. This also uses dipoK which is the Fourier transform of dipoR (the actual displacements). This routine calculates the forces in reciprocal space then FTs back to get them in real space as dVddi.

dvddi_unharmonic adds on the unharmonic forces in (8) depending on alpha and gamma. This makes use of didiR which is a rearrangment of dipoR. The sum is done in real space. There is no Fourier transform involved.

dvddi_E_Field adds on the force due to electric field as charge times E-field.

optimize_homo_strain uses homogenous, homogenous_coupling_2 (which depend on the B parameters) and didiK (FT of displacements) to calculate the strain, "eta" in the paper. It also calls dvddi_homo_coupling which adjusts the forces dvddi due to this strain.

optimize_inho_strain uses inhomo_K and inhomo_coup2_K, calculated by elastic to adjust the forces on atoms depending on atomic positions and their fourier transforms. It does not affect strain calculation.

B.8 Undocumented features

Several parameters can be set in the input file but are not mentioned in readme.html.

mass_amu. The effective mass of the dipole in a.m.u. Note that p%mass_dipo=mass_amu/9648.53377163899 and this is actually used in the code.

k_stripe. Three numbers k_stripe(3). Form domain stripes. This is a wavevector. plot_dispersion (logical). If .true. write eigenvalues to filename.dipole-dipole-long.dat, filename-dipole-dipole-short.dat, filename-dipole-dipole-long+short.dat, filename.inhomo-K.dat

B.9 References cited in source code

```
---------- DIPOLE-DIPOLE-SHORT-RANGE.F
! ref_1: W. Zhong, David Vanderbilt and K. M. Rabe: PRB Vol.52 p.6301 (1995)
                !(e) in TABLE 1 of ref_1 is incorrect.

---------- ELASTIC.F
! ref_1: R. D. King-Smith and David Vanderbilt: PRB Vol.49 p.5828 (1994)
  ! See equation (A1) and (A2) of ref_1

---------- MOLECULAR-DYNAMICS.F
! ref_1: Bond

---------- NOSE-POINCARE.F
! ref_1: Stephen D. Bond, Benedict J. Leimkuhler and Brian B. Laird:
!         J. Comput. Phys. Vol.151 p.114 (1999)

---------- OPTIMIZE-HOMO-STRAIN.F
! ref_1: R. D. King-Smith and David Vanderbilt: PRB Vol.49 p.5828 (1994)
     ! See equation (A1) and (A2) of ref_1.
```

B.10 Fourier Tranforms (fft.f)

The subs fft_r2c_3x3upper, fft_r2c, fft_c2r (real to complex and complex to real) are called by the code and listed here. They in turn call dfftw_plan_dft_r2c_3d (this file) dfftw_plan_dft_c2r_3d (this file) (dfftw_execute and dfftw_destroy_plan are also called in case HAVE_LIBFFTW3)

These subroutines are implemented here in various ways depending on pre-compiler variables being defined (depending on computer system). The relevant one seems to be the last case BUILT_IN_FFT which calls fft3d in libbuilt_in_fft/fft3d.f

B.11 Conditional compilation

Here, we list all instances of conditional compilation in the code using the C preprocessor.

```
---------- COORD_MODULE.F
#if defined HAVE_CONFIG_H
#  if defined(SR11000)

---------- FERAM.F
#if defined HAVE_CONFIG_H
#if defined(SR11000)
#elif defined(__PGI)

---------- FFT.F
#if defined HAVE_CONFIG_H
#if defined(HAVE_LIBFFTW3)
#if defined(SR11000)
#if defined(HAVE_LIBACML) ||  defined(HAVE_L
#if defined(BUILT_IN_FFT)
```

Bibliography

[1] Martin, M. G. and Siepmann, J. I. *J Phys Chem B* **103**, 4508 (1999).

[2] Landau, L. D. and Lifshitz, E. M. *Statistical physics, part 1.* Butterworth-Heinemann, third edition, (1980).

[3] Allen, M. P. and Tildesley, D. J. *Computer Simulation of Liquids.* Oxford Science Publications, (2004).

[4] Todorov, I. T. and Smith, W. *The dlpoly3 user manual, version 3.10.0.* Daresbury Laboratory, (2009). Manual available at: www.cse.scitech.ac.uk/ccg/software/DL_POLY/index.shtml.

[5] Blackburn, J. F. *Time evolution of microstructure in complex systems: a computer simulation study.* Cambridge University, Cambridge, (1999).

[6] Zhou, M. *Proc R Soc Lond A* **459**, 2347–92 (2003).

[7] Widom, B. *Journal of Chemical Physics* **39**(11), 2808 (1963).

[8] Evjen, H. M. *Phys Rev* **39**, 675 (1932).

[9] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, Cambridge, England, (1986).

[10] Kittel, C. and Kroemer, H. *Thermal Physics.* Freeman and Co, second edition, (1980).

[11] Potoff, J. J. and Panagiotopoulos, A. Z. *Journal of Chemical Physics* **109**(24), 10914 (1998).

[12] Levitt, M., Hirshberg, M., Sharon, R., Laidig, K. E., and Daggett, V. *J Phys Chem B* **101**, 5051–5061 (1997).

[13] Agar, M. *Determining phase change equilibria.* National Physical Laboratory, (2008).

[14] Kaye, G. W. C. and Laby, T. H. *Tables of Physical and Chemical Constants.* National Physical Laboratory, London, web (sixteenth) edition, (1995). Available online: www.kayelaby.npl.co.uk.

[15] Tegler, C., Span, R., and Wagner, W. *J Phys Chem Ref Data* **28**(3), 779–850 (1999).

[16] Span, R. and Wagner, W. J. *J Chem Phys Ref Data* **25**(6), 1509–96 (1996).

[17] Wagner, W. *ASME J Eng Gas Turbines and Power* **122**, 150–182 (2000).

[18] Davies, R. H., Dinsdale, A. T., Gibsby, J. A., Robinson, J. A. J., and Martin, S. M. *CALPHAD* **26**(2), 229–271 (2002).

[19] Dymond, J. H. and Smith, E. B. *The Virial Coefficients of pure gases and mixtures.* Clarendon Press, Oxford, second edition, (1980).

[20] Lopez-Rendon, R. and Alejandre, J. *J Mex Chem Soc* **52**(1), 88–92 (2008).

[21] Garrett, A. J. M. *J Phys A* **13**, 379–380 (1980).

[22] Dadson, R. S., Evans, E. J., and King, J. H. *Proc Phys Soc* **92**, 1115–1121 (1967).

[23] Dinsdale, A. T. *J Chem Phys* **90**, 189–200 (1993).

[24] Wyllie, P. J. *J Am Ceram Soc* **50**(1), 43 (1967).

[25] Gavin, D., Parker, V. B., and White, H. J. *CODATA thermodynamic tables. Selections for some compounds of calcium and related mixtures: a prototype set of tables*. Hemisphere, New York, (1987).

[26] Foiles, S. M., Baskes, M. I., and Daw, M. S. *Physical Review B* **33**(12), 7893 (1986).

[27] Hoyt, J. J., Garvin, J. W., Webb, E. B., and Asta, M. *Modelling Simul Mater Sci Eng* **11**, 287–299 (2003).

[28] Devonshire, A. F. *Philos. Mag* **40**, 1040–1055 (1949).

[29] Li, Y., Cross, L., and Chen, L. *Journal of Applied Physics* **98**, 064101 (2005).

[30] Buessem, W. and LE CROSS, A. *Journal of the American Ceramic Society* **49**(1), 33–36 (2006).

[31] Bell, A. and Cross, L. *Ferroelectrics* **59**(1), 197–203 (1984).

[32] Ahluwalia, R. and Srolovitz, D. *Physical Review B* **76**(17), 174121 (2007).

[33] Landauer, R., Young, D. R., and Drougard, M. E. *Journal of Applied Physics* **27**(7), 752–758 (1956).

[34] Scott, J. F. *Science* **315**(5814), 954–959 FEB 16 (2007).

[35] Cross, L. *Ferroelectrics* **76**(3-4), 241–267 (1987).

[36] Kingsmith, R. and Vanderbilt, D. *Physical Review B* **49**(9), 5828–5844 Mar (1994).

[37] Lines, M. and Glass, A. *Principles and Applications of Ferroelectrics and Related Materials*. Clarendon, Oxford, (1977).

[38] Ghosez, P. and Junquera, J. *Arxiv preprint cond-mat/0605299* (2006).

[39] Gonzalo, J. and Jimnez, B., editors. *Ferroelectricity: The Fundamentals Collection*. Wiley, (2005).

[40] Dawber, M., Rabe, K., and Scott, J. *Reviews Of Modern Physics* **77**(4), 1083–1130 OCT (2005).

[41] Scott, J. F. *Journal of Physics: Condensed Matter* **18**(17), R361–R386 May (2006).

[42] Gregg, J. M. *Nature Nanotechnology* **3**(7), 380–381 JUL (2008).

[43] Bibes, M. and Barthelemy, A. *Nat Mater* **7**(6), 425–426 June (2008).

[44] Eerenstein, W., Mathur, N., and Scott, J. *Nature* **442**(7104), 759–765 (2006).

[45] Rabe, K. A., Dawber, M., Lichtensteiger, C., Ahn, C. H., and Triscone, J.-M. *Physics of Ferroelectrics: A Modern Perspective* **105**, 1–30 (2007).

[46] Vanderbilt, D. *Phys. Rev. B* **41**(11), 7892–7895 Apr (1990).

[47] Saghi-Szabo, G., Cohen, R. E., and Krakauer, H. *Phys. Rev. B* **59**(20), 12771–12776 May (1999).

[48] Wu, X., Vanderbilt, D., and Hamann, D. R. *Phys. Rev. B* **72**(3), 035105 Jul (2005).

[49] Fu, H. and Bellaiche, L. *Phys. Rev. Lett.* **91**(5), 057601 Jul (2003).

[50] Zhong, W., Vanderbilt, D., and Rabe, K. M. *Phys. Rev. Lett.* **73**(13), 1861–1864 Sep (1994).

[51] Zhong, W., Vanderbilt, D., and Rabe, K. M. *Phys. Rev. B* **52**(9), 6301–6312 Sep (1995).

[52] Bellaiche, L., Garcia, A., and Vanderbilt, D. *Phys. Rev. Lett.* **84**(23), 5427–5430 Jun (2000).

[53] Nishimatsu, T., Waghmare, U. V., Kawazoe, Y., and Vanderbilt, D. *Physical Review B* **78**(10) SEP (2008).

[54] Gruverman, A., Wu, D., Fan, H.-J., Vrejoiu, I., Alexe, M., Harrison, R. J., and Scott, J. F. *Journal Of Physics-Condensed Matter* **20**(34) AUG 27 (2008).

[55] Sepliarsky, M., Asthagiri, A., Phillpot, S., Stachiotti, M., and Migoni, R. *Current Opinion in Solid State & Materials Science* **9**(3), 107–113 (2005).

[56] Sepliarsky, M., Stachiotti, M., and Migoni, R. *Physical Review B* **72**(1), 14110 (2005).

[57] Blackburn, J. F. and Salje, E. K. H. *Journal Of Physics Condensed Matter* **11**, 4747–4766 (1999).

[58] Blackburn, J. F. and Salje, E. K. H. *Physics And Chemistry Of Minerals* **26**, 275–296 (1999).

[59] Blackburn, J. F. and Salje, E. K. H. *Journal Of Applied Physics* **85**, 2414–2422 (1999).

[60] Vanderbilt, D. and King-Smith, R. D. *Phys. Rev. B* **48**(7), 4442–4455 Aug (1993).

[61] Berry, M. V. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences (1934-1990)* **392**(1802), 45–57 March (1984).

[62] Resta, R. *Rev. Mod. Phys.* **66**(3), 899–915 Jul (1994).

[63] Scott, J. F., Gruverman, A., Wu, D., Vrejoiu, I., and Alexe, M. *Journal of Physics-Condensed Matter* **20**(42) OCT 22 (2008).

[64] Dieguez, O. and Vanderbilt, D. *Arxiv preprint arXiv:0801.4071* (2008).

[65] Beckman, S., Wang, X., Rabe, K., and Vanderbilt, D. *Arxiv preprint arXiv:0812.0960* (2008).

[66] Prosandeev, S., Ponomareva, I., Naumov, I., Kornev, I., and Bellaiche, L. *Journal Of Physics Condensed Matter* **20**(19), 193201 (2008).

[67] Huibregtse, E. J., Bessey, W. H., and Drougard, M. E. *Journal of Applied Physics* **30**(6), 899–905 (1959).

[68] Kay, H. and Vousden, P. *Philosophical Magazine (Series 7)* **40**(309), 1019–1040 (1949).

[69] Tinte, S., Iniguez, J., Rabe, K., and Vanderbilt, D. *Phys. Rev. B* **67**(6), 064106 Feb (2003).

[70] Merz, W. J. *Phys. Rev.* **91**(3), 513–517 Aug (1953).

[71] Tinte, S., Stachiotti, M. G., Sepliarsky, M., Migoni, R. L., and Rodriguez, C. O. *Journal of Physics: Condensed Matter* **11**(48), 9679–9690 (1999).

[72] Jaffe, B., Cook, W., and Jaffe, H. *Piezoelectric ceramics*. Academic Press, (1971).

[73] Kay, H. and Dunn, J. *Philosophical Magazine* **7**(84), 2027–2034 (1962).

[74] Haun, M., Zhuang, Z., Furman, E., Jang, S., and Cross, L. *Journal of the American Ceramic Society* **72**(7), 1140–1144 (1989).

[75] Dick, B. J. and Overhauser, A. W. *Phys. Rev.* **112**, 90 (1958).

[76] Sanders, M. J., Leslie, M., and Catlow, C. R. A. *J Chem Soc Commun* , 1271 (1984).

[77] Cratlow, C. R. A., Faux, I. D., and Norgett, M. J. *J Phys C: Solid State Physics* **9**, 419 (1976).

[78] Sepliarsky, M., Stachiotti, M. G., and Migoni, R. L. *Ferroelectrics* **186**, 69 (1996).

[79] Lewis, G. V. and Catlow, C. R. A. *J Phys C: Solid State Physics* **18**, 1149 (1985).

[80] Shimada, T., Takahara, K., Wakahara, K., Umeno, Y., and Kitamura, T. *J Phys: Condens Matter* **20**, 325225 (2008).

[81] Heifets, E., Dorfman, S., Fuks, D., and Kotomin, E. *Thin Solid Films* **269**, 76 (1997).

[82] Heifets, E., Kotomin, E., and Maier, J. *Surface Science* **462**, 19 (2000).

[83] Morse, P. M. *Phys Rev* **34**, 57–64 (1929).

[84] Rhodes, R. G. *Acta Cryst* **4**, 105 (1951).

[85] Zgonik, M., Bernaskoni, P., Duelli, M., Schlesser, R., Guner, P., Garrett, M. H., Rytz, D., Zhu, Y., and Wu, X. *Phys Rev B* **50**, 5941 (1994).

[86] Gale, J. D. *J Chem Soc Faraday Trans* **93**, 629 (1997).