Molecular Dynamics Simulations of Fluorite Structure Crystals

Thesis submitted by

Andrew Michael Brass

for the degree of
Doctor of Philosophy

University of Edinburgh

Febuary 1987

To Kathy and my parents.

'Tis nature still, but nature methodised.

Alexander Pope

A copy of the universe is not what is required of art;
one of the damned things is ample.

Rebecca West

Acknowledgements

I would like to thank my supervisor Professor G S Pawley for initiating the work in this thesis and providing advice and encouragement during my time in Edinburgh. Thanks are also due to Keith Refson for much invaluable advice and discussion on the techniques of MD and computing, to Larry Boyer for firing my enthusiasm about superionic conductors and to Martin Dove, Brian Pendleton and Ian Cox for their many helpful suggestions.

I would also like to acknowledge the award of a Science and Engineering Research Council Information Technology grant and the support of the Edinburgh Regional Computing Centre, both in their support of the DAPs and the computational work involved in this work.

I am happy thank all the people who have made the last three years in Edinburgh so much fun, in particular Simon Hands, Catherine Chalmers, Susanna Williamson, Douglas McKnight and Duncan Roweth. In particular I would like to thank my parents for their constant encouragement and support, and Kathy Else for for helping me to keep a sense of perspective no matter how badly the program was working. Finally I would like to thank all the ex members of the Accies, from whom I learnt perseverance.

Declaration

The work in section 2.3 was done in collaberation with K Refson and the work done in section 4.6 was done in collaberation with Dr. L. Boyer. All the other work is my own, except where otherwise stated.

Abstact

The thesis describes molecular dynamics simulation studies of fluorite structure ionic crystals, the simulations being performed on systems containing 3630 ions.

Molecular dynamics integration algorithms are investigated in detail, particularly with regard to the accuracy and stability of particle trajectories generated by the simulation and the influence of chaos on these trajectories. A new function is described which enables the rate at which a simulation reaches equilibrium to be accurately determined as well as providing information on whether the equilibrium state is infact ergodic and in the micro-canonical ensemble. A detailed study is also made of the calculation of scattering functions in MD simulations. In particular an explanation is given as to why the scattering functions as calculated in an MD simulation can be different from those calculated experimentally.

The particle-particle/particle-mesh algorithm for calculation of long range forces is described. The implementation of this algorithm for the calculation of ionic forces on highly parallel computers is also discussed.

The MD simulation simulations are of three compounds; $SrCl_2$, PbF_2 and CaF_2. The defect structure of these compounds is examined in detail, both above and below the superionic transition temperature. The simulations on large systems appear to support the point defect model of Gillan rather than the defect cluster model of Hutchings *et al.* Using lattice dynamics a mechanism is discussed which may be responsible for driving the superionic transition in fluorites.

TABLE OF CONTENTS

CHAPTER 1

AN INTRODUCTION TO SUPERIONIC CONDUCTORS

1.1. Introduction

Superionic solids are ionic compounds whose electrical conductivity is comparable with those of ionic melts or electrolyte solutions (these materials are sometimes referred to as "fast-ion conductors" or "solid electrolytes"). Superionic solids have electrical conductivities of 10^{-1}-10^{-4} $ohm^{-1}cm^{-1}$ in comparison with room temperature electrical conductivities for normal ionic solids of 10^{-12}-10^{-16} $ohm^{-1}cm^{-1}$.

A perfect ionic crystal would be an insulator, in order for the ionic solids to conduct electricity they must contain defects or be disordered in some way. Ionic crystals can be divided into three categories as regards their electrical and defect properties. Type I compounds contain a low concentration of point defects and so have only limited ion transport; examples of such solids include NaCl, BrI etc. Type II compounds contain a high concentration of point defects which can be used in the ion transport mechanisms; examples of such solids include CaF_2, $SrCl_2$, stabilised zirconia etc. Type III ionic compounds are compounds for which one of the ionic sublattices has become highly disordered, resembling liquid-like behaviour; examples of 'molten sublattice' compounds include β-alumina and Li_3N.

Most superionic materials attain high electrical conductivities above some temperature which may or may not be well defined. Some compounds show a gradual increase in conductivity with increasing temperatures (eg β-alumina or CaF_2, whereas in others the conductivity suddenly increases at a well defined temperature (β-AgI, $RbAg_4I_5$). Sometimes the abrupt change in conductivity is related to a structural phase transition (such as the change from the low conductivity of β-AgI to the superionic conductivity of α-AgI), whereas sometimes the cause of the sudden transition is not so clear (as in $RbAg_4I_5$). Figure 1.1 shows a plot of conductivity versus temperature for a variety of ionic and superionic compounds.

A wide variety of ionic materials show superionic behaviour, and both cations

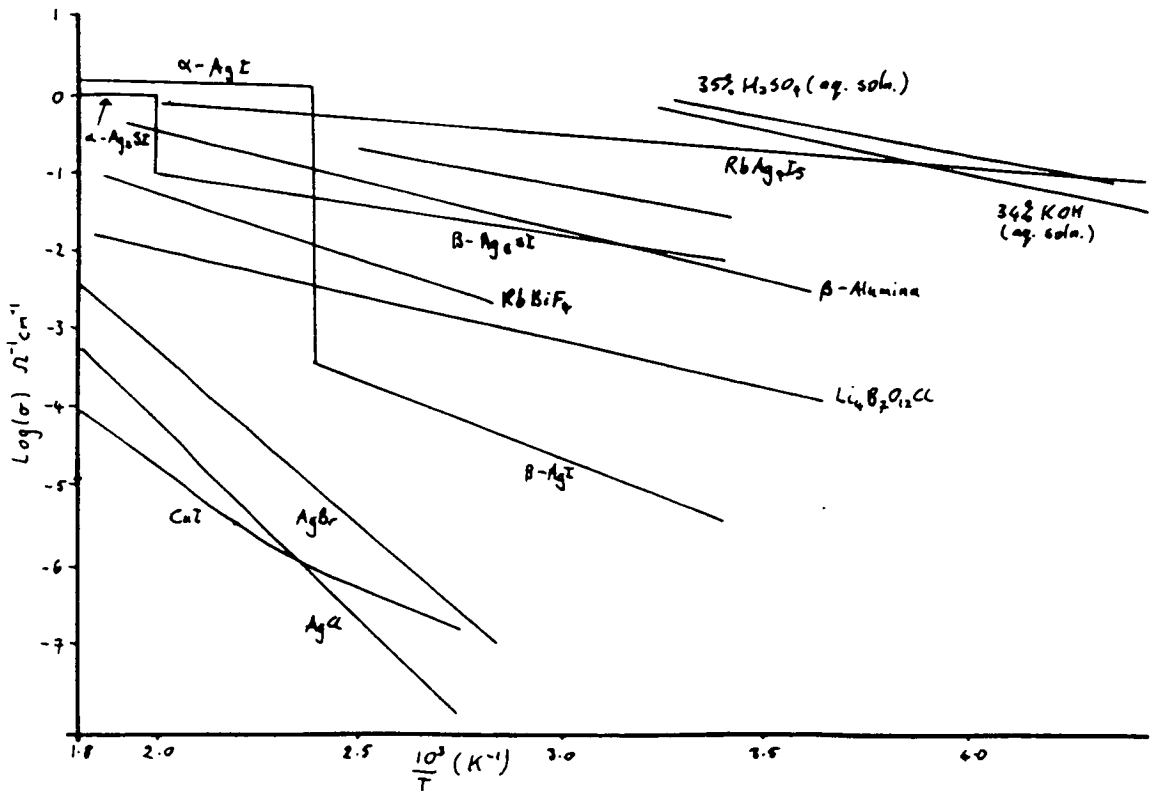


Figure 1.1 Electrical conductivity versus temperature for some ionic and superionic solids.

and anions, eg Li^+, Na^+, K^+, Ag^+, Cu^+, F^-, O^{2-} and Cl^- can be the mobile ions.

1.2. Superionic Materials

1.2.1. Silver Ion Conductors

The first superionic conductor to be discovered was the silver conducting superionic solid Ag_2S, which was described by Faraday in 1833. Tubandt (1932) then discovered that the compounds Ag_2Se and Ag_2Te showed similar behaviour. The most important silver ion conductors so far discovered are the silver-halides. The first silver-halide to be investigated was AgI (Tubandt and Lorenz 1913). The conductivity of AgI increases by more than three orders of magnitude at 137°C, the temperature at which it switches from a cubic (α-AgI) to a wurtzite (β-AgI) structure. Most good silver ion conductors are obtained by partially substituting different ions either for the Ag^+ ion (KAg_4I_5, $RbAg_4I_5$, α-Ag_2HgI_4), for the I^- ion ($Ag_7I_4AsO_4$, β-Ag_3SI), or for both (KAg_4I_4CN, $RbAg_4I_4CN$). The silver ion conductors based on AgI are important because some of them show superionic behaviour at room temperatures. The silver ion conductors to have received the most attention are the ones based on the formula MAg_4I_5, M=K,Rb,NH_4 as $RbAg_4I_5$ has the highest room temperature conductivity of any compound so far discovered (0.27 $ohm^{-1}cm^{-1}$).

All of the AgI-type compounds show very similar dynamical behaviour. The disorder occurs on the cation sublattice, the cations showing liquid-like behaviour. In the crystal the potential barriers between anion sites are very low, of the order of the thermal energy. All the AgI type compounds show channel-like pathways connecting the sites through the face-shared iodine polyhedra (Geller 1973). Because of the existence of these channels with low potential barriers the cations can move readily between the lattice sites. The cation motion is via a jump-diffusion process with correlations between the cations being very important in the hopping process. The cation conduction in AgI is therefore a result of the existence of channels connecting the anion tetrahedra, the mobile cations following almost linear paths (Funke 1976).

1.2.2. Alkali Metal Conductors

The most important alkali metal ion conductors are those based on the β-alumina structure as a lot of these compounds have significant conductivities at room temperature (the name β-alumina is in fact a misnomer as the structure is not, as first thought, a crystallographic modification alumina).

The name β-alumina refers to a large class of compounds, all of them related to $M_2O.xAl_2O_3$; x=5-11, M=Na,K,Rb,NH_4. It is also possible to produce β-alumina compounds with Al replaced by Fe or Ga. It is the unique crystal structure of β-alumina that gives it such high ionic conductivities (Beevers and Ross 1937, Whittingham and Huggins 1972). The β-aluminas have a layered, hexagonal structure, the alkali metal ions being situated in planes perpendicular to the c-axis. The structure of β-alumina resembles that of $MgAl_2O_4$ (Spinel) and is therefore sometimes referred to as a "spinel block" structure. The ion transport is exclusively along the alkali ion planes and so is highly anisotropic along these channels; β-alumina is a type III superionic conductor. The transport of the cations appears to depend on correlations between the mobile cations (Murch and Thorn 1977). The size of the mobile species also appears to be important, Na^+ and Ag^+ being more mobile than Li^+ or Rb^+ in the cation channels.

Many compounds, other than just β-alumina, show alkali metal conduction. For example alkakli metal silicates, aluminosilicates, tantalates, solid solutions with phosphates, nitrides etc. all show alkakli ion conduction, although the number of Li^+ conductors exceeds the number of Na^+ conductors, and the number of Na^+ conductors exceeds the number of K^+ conductors etc. (Huggins 1977). All of these compounds are normally classed as having a channelled structure (type III superionics). For example Li_3N contains hexagonal planes of Li_2N connected by lithium ions which form a N-Li-N bridge (Wolf *et al* 1984). This structure therefore provides large empty intersecting tunnels which can be used for lithium transport. There are also type II alkali metal conductors, especially compounds with the anti-fluorite structure such as Li_2SO_4 and Li_5ZnO_4.

1.2.3. Oxygen Ion Conductors

Most of the useful oxide electrolytes developed so far are type II fluorite structure superionic conductors, for example ThO_2, zirconia and CeO_2. The fluorite structure can be thought of as a cubic lattice of anions with a cation in the centre of alternate anion cube centres. The empty cube centre sites can then provide pathways between anion lattice sites for the mobile O^{2-} ions. Most of the oxide electrolytes only become superionic at high temperatures ($>1000^{\circ}C$), More will be said about the fluorite structure superionics in section 1.4.

1.2.4. Halide Ion Conductors

The other class of anion superionic conductors are those in which the mobile ions are halides. As with the oxygen ion superionics, most of the halide ion superionics have the fluorite structure. Examples of such compounds include CaF_2, β-PbF_2, $SrCl_2$, SrF_2, BaF_2 and $SrBr_2$. For most of the fluorite structure superionic compounds the conductivity increases smoothly with temperature, unlike compounds such as AgI which show an abrupt increase in conductivity at some temperature. The fluorite structure, halide transporting superionic compounds are discussed in greater detail in section 1.4.

1.3. Applications of Superionic Conductors

The uses of superionic compounds in industrial processes and as research probes are wide and varied, from pollution control to the manufacture of fuel cells (see Chandra, Chapter 6). The first reported use of these materials was in a paper by Kiukkola and Wagner (1957) who demonstrated the use of oxygen ion electrolytes in the determination of the thermodynamic properties of high-temperature materials. The next major development in the use of superionic solids was the discovery of the room temperature superionic conductors β-alumina (Yao and Kummer 1967) and MAg_4I_5 (M = Rb, K, NH_4) (Bradley and Greene 1967) and the application of these compounds to the development of high-energy density batteries. Some of the applications of superionic compounds are described below.

1.3.1. Thermodynamic Studies

The formation energy of a chemical compound is known as the Gibbs free energy, ΔG. Consider for example the reaction:

$$M + X \rightarrow MX$$

If a cell is constructed such that M and X are brought together via ion transport in a superionic conductor, then the value of the emf developed in the determines ΔG. For example consider following cell used to determine ΔG for the compound MX:

$$Pt \mid M \mid Y \mid X \mid Pt$$

where Y is a superionic compound that conducts either M ions, X ions or both types of ions. Platinum wires are used simply because platinum is inert and will therefore not be involved in the reaction.

Knowing the emf, E, developed by the reaction, then ΔG can be determined via:

$$\Delta G = E/|Z_X|F \qquad (1.1)$$

where $|Z_X|$ is the absolute valency of X and F is Faraday's constant.

As an example consider calculating ΔG for the simple compound AgI. The cell required to calculate ΔG for this compound is simply:

$$Ag \mid AgI \mid I_2$$

The Ag^+ ions migrate through AgI and combine to form AgI at the AgI-I_2 interface. The emf developed is $\sim$ .69V giving a value of 15.8kcal/mole for the formation of AgI. Similar techniques can be applied to determine the energy of formation of a wide variety of compounds.

1.3.2. Sensors and Partial pressure gauges

Consider the device shown in figure 1.2. Using such devices it is possible to calculate the unknown pressure, P_1, very accurately given the pressure P_2. Consider an oxide electrolyte, such as calcia stabilised zirconia, with oxygen at pressure P_1 at one side and oxygen at pressure P_2 at the other at a temperature at which the zirconia is in the superionic phase (>1217K). If the pressures on each side are equal, the oxygen in the zirconia lattice and the oxygen on each side of the zirconia will come into equilibrium and no emf will be generated. If, however, P_1 and P_2 are not equal then oxygen will flow through the zirconia and an emf will be generated, the emf being given by:

$$E = (RT/|Z_{O2}|F)\ln(P_1/P_2);\ P_1 > P_2 \qquad (1.2)$$

where Z_{O2} is the absolute valency of oxygen (Sproule 1974). Cells such as these have found many uses, including the measurement of the amount of oxygen dissolved in molten metals (Fitterer 1972).

1.3.3. The Construction of Fuel Cells

Fuel cells are devices which can provide continuous electrical voltages and currents through the constant consumption of one of the electrolytes. In many ways fuel cells resemble the partial pressure sensors described in the previous subsection. For example consider the cell:

$$Pt \mid air(O_2) \mid \text{oxide electrolyte} \mid (H_2,H_2O) \mid Pt$$

As there is a difference in the oxygen pressure on either side of the superionic solid oxygen will be transported through the solid, an emf being generated across the cell as a result of this transport (see equation (1.2)). The difference in oxygen pressure is maintained by burning the oxygen in hydrogen gas as soon as it is carried across the solid electrolyte (see Etsell and Flengas 1970 and Takahashi 1973).

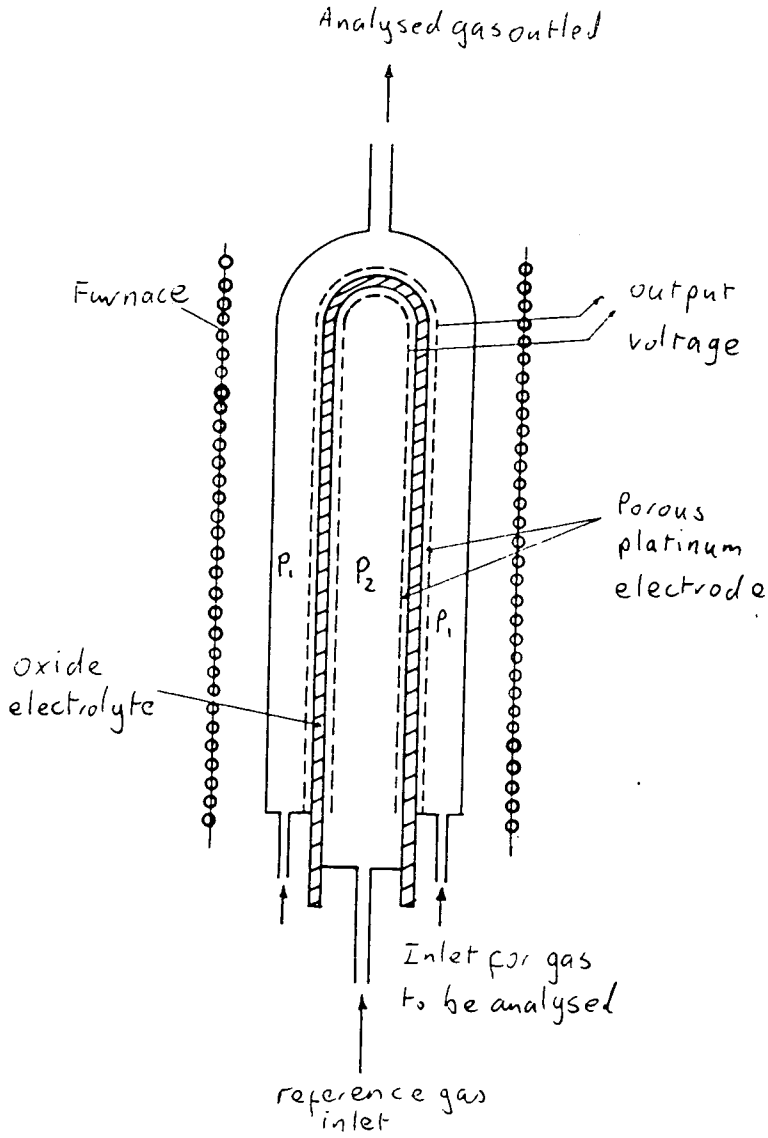


Figure 1.2 Schematic diagram of an oxygen gauge (from Chandra, chapter 6)

1.3.4. Solid State Batteries

Traditional electrochemical galvanic cells such as Leclanche cells or Voltaic cells suffer from several disadvantages; their range of operation is restricted to the range 0–100°C, they are not particularly rugged, corrosion of electrodes by the electrolyte gives them a limited shelf life and they can not be miniaturised. The use of solid electrolytes in battery construction could, in principle, overcome all of these problems.

The principle behind solid state batteries is very straightforward; an emf will be generated if an electrolyte is sandwiched between two electrode materials with different chemical potentials. If a constant supply of ions can be maintained through the electrolyte then the battery will produce a current when it is connected across a load resistor. For example the cell:

$$M|MX|X$$

can be used as an electrochemical cell, the emf generated by the cell being given by:

$$E = -\Delta G/|Z_X|F \text{ (see equation (1.1))}$$

where ΔG is the Gibbs free energy of MX.

In order to produce a successful battery then several points must be born in mind. Firstly the Gibbs free energy for the reaction must be large and negative. Secondly the ionic conductivity of the superionic compound must be high in order to minimise the internal resistance. Thirdly the electronic conductivity of the solid electrolyte must be low to avoid the problems of internal short circuits, and finally the electrodes and electrolyte must not react with each other - the interfaces must be chemically inert (for a review of solid state batteries see Gross 1976).

1.3.5. Decomposition of Gases

Another important application of superionic solids is in the decomposition of gases, in particular the use of oxygen ion conductors in promoting the reactions:

$$2CO_2 + 4e + 2V_O \rightarrow 2CO + O_2$$

$$2H_2O + 4e + 2V_O \rightarrow 2H_2 + O_2$$

$$2NO + 4e + 2V_O \rightarrow N_2 + O_2$$

where V_O is an lattice oxygen vacancy. The apparatus used is shown in figure 1.3.

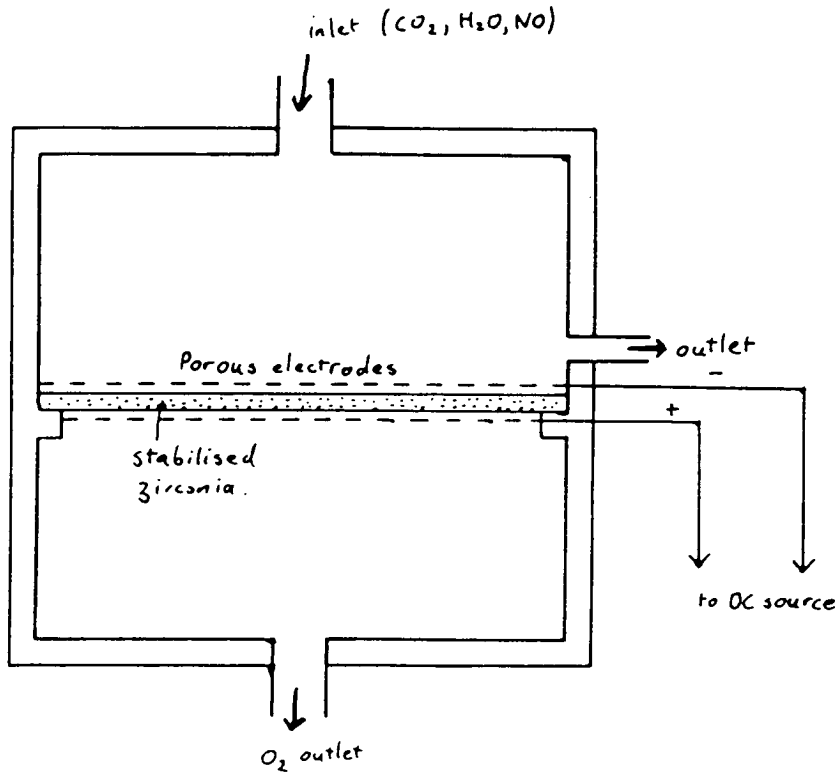


Figure 1.3 Apparatus for the decomposition of gases (from Chandra, chapter 6)

The first reaction is potentially useful in spacecraft life-support units (Weissart and Smart 1970), the second reaction is a cheap method of producing hydrogen for use as a fuel and the third reaction is useful for removing the

industrial enviromental pollutant nitrous oxide (Pancharatnam *et al* 1975).

1.3.6. Other Applications

There are numerous other ways in which superionic conductors are used in industry, from electrochemical capacitors (Raleigh 1976) to solid state displays (Hurditch 1976, Chang 1975) and high temperature thermometers (Piazzini and Bianchi 1973) to thermoelectric generators (Weber 1974).

Given the numerous industrial and research applications of the superionic solids it is important that we should have a good theoretical understanding of the conduction processes in these compounds.

1.4. The Fluorite Structure Superionic Conductors

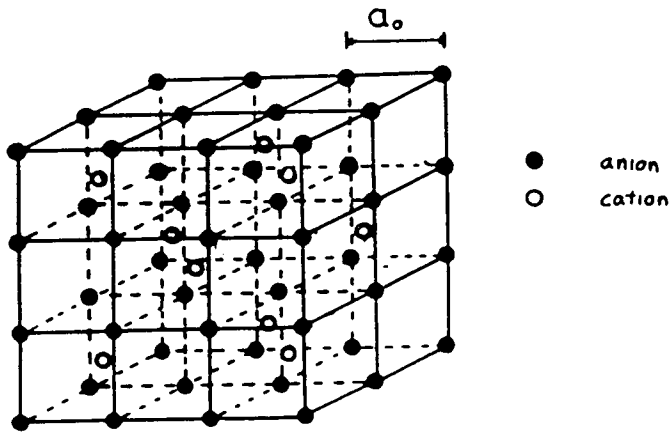
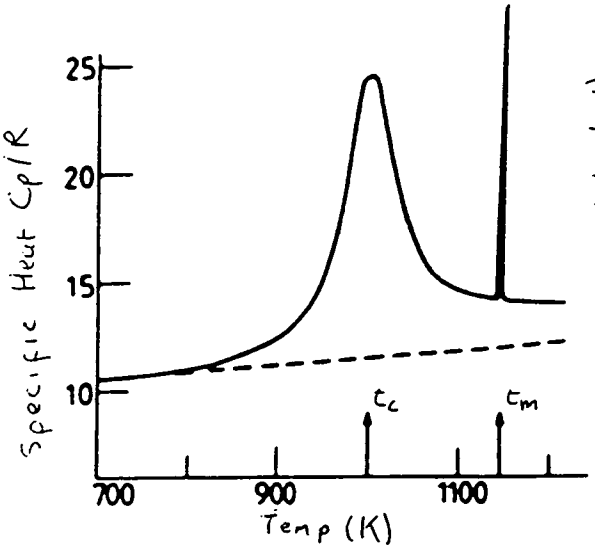


Figure 1.4 The fluorite crystal stucture, Fm3m, a_0 is the anion-anion nearest neighbour separation. The cation-cation nearest-neighbour separation will be referred to as a. These conventions will be used throughout the thesis.

The fluorite structure crystals provide one of the simplest systems in which to study the phenomena of superionic conduction. The fluorite structure, Fm3m, can be viewed as a simple cubic array of anions with every other cube centre site being occupied by a cation (see figure 1.4). Most fluorites exhibit a Schottky-like anomaly in their specific heat (figure 1.5) at a temperature, t_c, a

few hundred degrees below their melting temperature, t_m (Dworkin and Bredig 1968). This peak in the specific heat is associated with the onset of dynamic disorder on the anion sublattice, the disorder on the anion sublattice increasing the ionic conductivity to values typical of the ionic melt (figure 1.6).



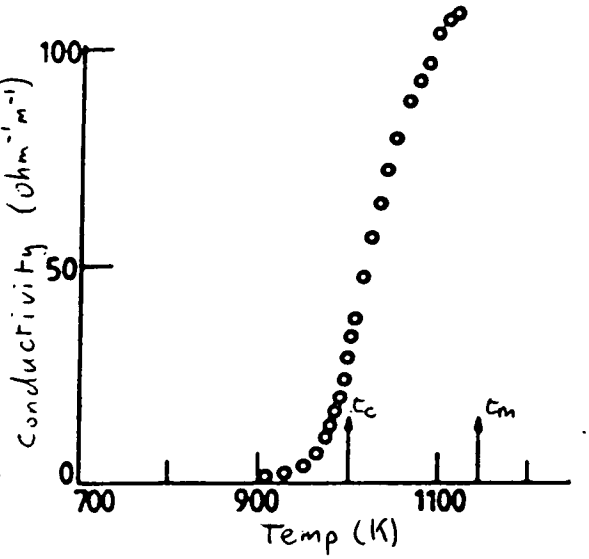


Figure 1.5 Temperature variation of the specific heat (Schroter and Nolting 1980)
Figure 1.6 Temperature variation of the electrical conductivity of $SrCl_2$ (Carr *et al* 1978).

Some workers have interpreted this peak in the specific heat as being associated with the latent heat of melting of one of the sublattices, suggesting that the mobile anions will show liquid-like diffusive behaviour (O'Keefe 1976). The anomaly in the specific heat and high conductivities are sometimes referred to as being a Faraday transition (O'Keefe and Hyde 1976), however the transition does not appear to involve singularities in any of the thermodynamic functions and so is not a true phase transition (Hayes 1978). Table 1.1 gives values of t_c and t_m for a variety of fluorite structure superionics.

Table 1.1 Values of t_c and t_m for PbF_2, CaF_2 and $SrCl_2$.

Compound	t_c(K)	t_m(K)
PbF_2	711	1128
$SrCl_2$	1001	1146
CaF_2	1430	1696

Conductivity measurements using tracer diffusion techniques (Beniere *et al* 1979, Hood and Morrison 1967), and lattice statics calculations (Catlow *et al* 1977) have all shown that the disorder in the superionic phase is confined to the anion sublattice.

Two techniques have proved particularly useful in analysing the anion disorder in fluorite structure crystals; molecular dynamics computer simulations and neutron scattering. Molecular dynamics provides the only way of studying in detail the microscopic behaviour of realistic models of superionic conductors, and neutron scattering allows investigation of the crystal dynamics at length and time scales relevant to the superionic diffusion processes.

At temperatures below t_c it is now well established that the principle thermally induced defect in the fluorite lattice is an anion Frenkel defect pair consisting of an interstitial residing in the centre of one of the empty cube centre sites and an associated vacancy on the anion lattice, with the vacancy and interstitial being at least next-nearest neighbours (Catlow *et al* 1977, Gillan and Richardson 1979, Lidiard 1974). Early theories of the properties of the high temperature phase of the fluorites assumed that such cube centre interstitials would play an important role in the diffusion mechanism (Shapiro 1976, Dickens *et al* 1979). However recent molecular dynamics computer simulations (Dixon and Gillan 1978), and detailed examination of the neutron scattering and X-ray diffraction data (Shapiro and Reidinger 1979, Dickens *et al* 1979, Koto *et al* 1980) have shown that the cube centre site is not appreciably occupied for any length of time. Lattice statics calculations (Gillan and Richardson 1979) suggest that this is due destabilisation of the cube centre site by nearby vacancies.

The question of the actual number of defects present in the superionic state has received much attention in recent years. Part of the controversy arose because of the differences in the definition of a Frenkel defect. From the neutron scattering data it is possible to determine the number of anions which have left their lattice site. Some of the anions will simply be undergoing large anharmonic vibrations about a lattice site whereas others will have left their lattice sites to form part of a Frenkel pair. If the neutron scattering data is analysed carefully (Dickens *et al* 1979) it can be shown that although up to 40% of the anions may have left their lattice sites, only 3% have left their

regular lattice sites to move to interstitial positions. The molecular dynamics calculations of Gillan and Dixon (1980) have shown that if a dynamic criteria is used to define the defects, the number of true defects per lattice site is of the order of a few percent. The picture that we obtain from the neutron scattering results and the MD simulations is therefore of a crystal containing a small percentage of highly mobile point defects.

Incoherent quasielastic neutron scattering techniques enable the self pair distribution function, $G_s^D(\mathbf{r},t)$, to be calculated. By examining the form of $G_s^D(\mathbf{r},t)$ it is possible to determine whether the mobile anions move via a diffusive process, i.e. obeying Fick's law, or via random jumps (Chudley and Elliott 1961). It was found (Dickens *et al* 1983, Schnabel *et al* 1983) that the best fit to the neutron scattering data was obtained using a hopping model in which the anions hop between anion sites in the 100 and 110 directions, the proportion of 100 to 110 hops being 73±5 to 25±5. Computer simulations (Gillan and Dixon 1980, Dixon and Gillan 1980, Jacucci and Rahman 1978) also suggest that the anions hop directly between anion sites. The molecular dynamics and neutron scattering results provide no support for the molten sublattice interpretation the of fluorite structure crystals.

Coherent neutron scattering allows investigation of the correlated anion motion. Hutchings *et al* (1984) have performed extensive measurements of inelastic, coherent neutron scattering from PbF_2, CaF_2 and $SrCl_2$. These measurements show a strong quasielastic contribution to the scattering which appears for temperatures above t_c, the main feature of the scattering being a peak in the intensity along the 100 direction near the 200 point (figure 1.7). The intensity of this peak was found to depend strongly on temperature, increasing rapidly as the temperature is increased through t_c. The intensity was also found to be very dependent on the wavevector **q**. The energy width increased rapidly with the temperature and corresponded to a decay time of approximately 1ps. As this feature is connected with the superionic conduction of the fluorites, understanding the mechanism which give rise to this peak should significantly increase our understanding of the ionic conduction processes in fluorite structure superionic compounds.

The observation of this coherent diffuse scattering implies that the system is disordered in some manner. It has been suggested that this disorder arises

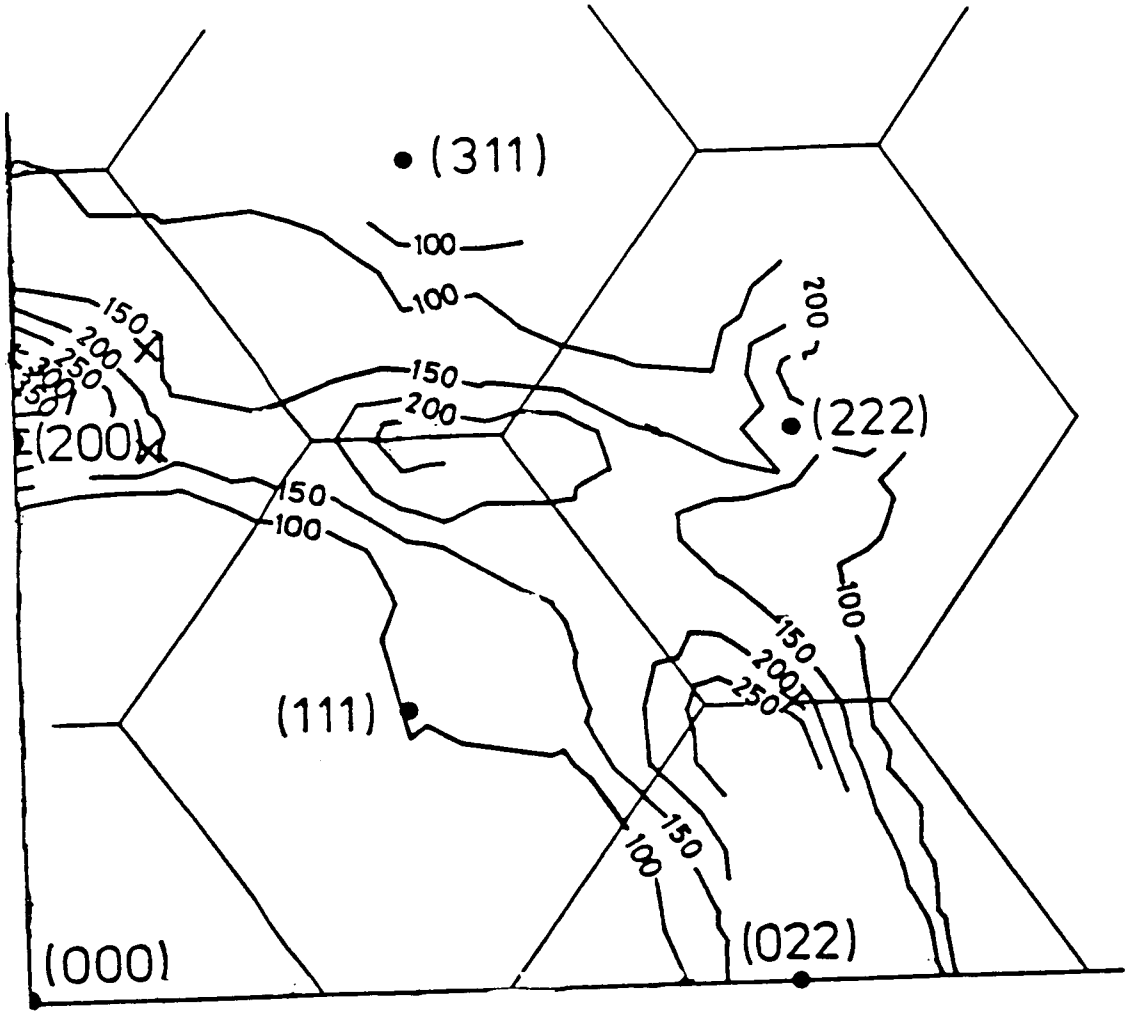


Figure 1.7 The measured distribution of $S^{D}(\mathbf{Q})$, the coherent diffuse quasielastic scattering in the 011 plane of reciprocal space for CaF_2 at 1473K (Hutchings *et al* 1984).

from independent, short-lived clusters of anion Frenkel pairs with lattice relaxation around the interstitials (Hutchings *et al* 1984, Clausen *et al* 1981). It is assumed in these defect cluster models that the cations always occupy their regular sites and that there is no relaxation around a vacancy. Each cluster comprises Frenkel pairs of interstitial anions and vacancies in which neighbouring anions are relaxed into the empty anion cubes. Several different model clusters have been proposed in which either one or two Frenkel defect pairs and different relaxation fields are involved. The different cluster types can be classified by three numbers, V:I:R; where V is the number of vacancies, I is the number of true interstitials and R is the number of relaxed anions. The simplest cluster considered by Hutchings *et al* was the 3:1:2 cluster illustrated in figure 1.8.

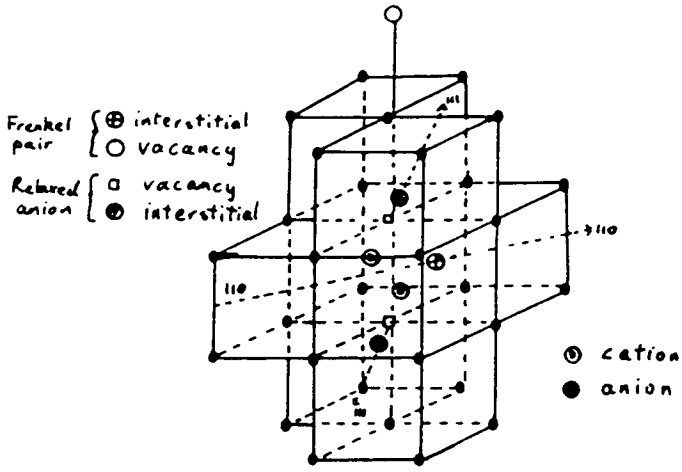


Figure 1.8 The 3:1:2 cluster showing the positions of the Frenkel interstitial ion, the Frenkel anion vacancy site, and the relaxed anion sites.

In the 3:1:2 cluster the vacancy is located two lattice units from the interstitial (Catlow and Hayes 1982) and the two nearest neighbours to the interstitial are relaxed along the 111 directions towards the empty cube centre sites. Other clusters considered where the 9:1:8 cluster, essentially a 3:1:2 cluster with additional relaxation around the interstitial, a 4:2:2 cluster which can be considered as a 3:1:2 cluster with an additional true interstitial relaxed from the mid-anion site in the opposite direction to the 3:1:2 interstitial, and two cube centre clusters in which the interstitial is located at the cube centre. Table 1.2 shows the displacements of the interstitials and relaxed anions in each type of cluster. Hutchings *et al* found that the best fit to the experimental data for all three fluorites was given by the 9:1:8 clusters. Both the cube

centre clusters proved to give very unsatisfactory fits to the observed neutron scattering data.

Table 1.2 Displacements of interstitial and relaxed anions towards the empty cube sites for the various clusters described above, a) calculated using a hard-sphere model at the given temperature, b) static energy calculation (Catlow and Hayes 1982), c) experimental values from diffraction data (taken from Hutchings *et al* 1984).

i) displacement of the mid-anion interstitial in the 110 direction in units of a_0.

Cluster		PbF_2 at 892K	$SrCl_2$ at 1078K	CaF_2 at 1473K
3:1:2	a	0.32	0.28	0.30
4:1:3	a	0.20	0.20	0.20
4:2:2	a	0.32	0.28	0.30
4:2:2	b	-	0.34	0.34
9:1:8	a	0.28	0.16	0.22
9:1:8-CC110	a	0.14	0.22	0.18
3:1:2	c	0.14	0.08	-

ii) relaxation from the regular anion site in a 111 direction in units of a_0

Cluster		PbF_2 at 892K	$SrCl_2$ at 1078K	CaF_2 at 1473K
3:1:2	a	0.22	0.30	0.24
4:1:3	a	0.20	0.32	0.28
4:2:2	a	0.22	0.30	0.24
4:2:2	b	-	0.34	0.32
9:1:8 NN	a	0.30	0.36	0.30
NNN		0.06	0.18	0.12
9:1:8-CC110	a	0.06	0.18	0.12
3:1:2	c	0.24	0.26	0.16

The neutron scattering results have been interpreted as suggesting that in the superionic fluorites most of the ion transport is via the vacancies whilst the interstitials are tied up in clusters (Moscinski and Jacobs 1985, Hutchings *et al* 1984, Catlow and Hayes 1982, Dickens *et al* 1983). The justification for this is that the incoherent neutron scattering results show the presence of highly mobile anions hopping between lattice sites and these anions are associated with the vacancy motion, the coherent scattering results show the presence of clusters of defects which can be identified with the interstitials.

The molecular dynamics calculations of Gillan (1986b) have suggested a different interpretation which can be placed on the neutron scattering results.

The interpretation of Hutchings *et al* treats the coherent and incoherent scattering as arising from independent causes, the former being due to vacancy motion and the latter being due to the formation of defect clusters. In a dense solid, however, the diffusion of ions must require a large degree of correlated motion, a hopping ion must produce a strong disturbance in the surrounding lattice. Gillan therefore argues that the coherent and incoherent scattering must be connected in some way, the incoherent scattering arising from the hopping behaviour of the anions and the coherent scattering from the lattice distortion arising from the motion of the hopping ions. It has been shown (Gillan 1986a) that the quasielastic coherent peak near the 200 point is reproduced in molecular dynamics simulations. As the scattering from the computer simulations is very similar to the scattering observed experimentally, the processes that cause the 200 scattering in real fluorite structures should also be present in the computer simulation. If an explanation can be found for the neutron scattering spectrum in the simulation, then the same explanation should also hold true in the real crystals.

Gillan (1986b) showed that the quasielastic peak could be described in terms of lattice relaxations around highly mobile, non interacting point defects, both the vacancy and interstitial defects being thought of as residing on anion lattice sites. In the picture of superionic conduction proposed by Gillan there are no interstitial sites in the anion lattice except for the equilibrium lattice sites themselves.

There is a wide measure agreement between the models of superionic conduction in fluorite structure crystals obtained from computer simulation and neutron scattering. In both cases the mobile anions hop between lattice sites, predominantly in the 100 and 110 directions, and in both cases the dominant conduction mechanism is seen to be the hopping motion of a small number of highly mobile point defects, the concentration of point defects being about 3% per lattice site. However the two models differ greatly in their interpretation of these point defects. The molecular dynamics results suggest that both vacancy and interstitial defects are equally mobile, both types of defect being regarded as residing on lattice sites. In this model the defects do not interact with each other, the mobile ions do, however, carry a relaxation field around with as they hop between lattice sites. The neutron scattering results have been interpreted as suggesting that the bulk of the ion transport

arises from the hopping motion of vacancy defects between anion lattice sites with the interstitials being bound up in short lived, fluctuating defect clusters, the interstitial residing close to the mid-anion position.

It has been suggested that part of the discrepancy between the neutron scattering and MD results arises because the computer simulations have been on systems that are too small to contain a defect cluster (Moscinski and Jacob 1985). In order to see whether the defect clusters can be seen in a molecular dynamics simulation it was decided to attempt molecular dynamics simulations on systems at least an order of magnitude larger than the systems on which previous calculations had been attempted (systems of 3420 ions as opposed to 324 ions). Using systems of this size it should also be possible to examine the low temperature behaviour of the fluorites and gain some understanding of the processes involved close to the superionic transition temperature when the crystals begin to show the fast ion behaviour.

CHAPTER 2

MOLECULAR DYNAMICS

Many analytical techniques, from lattice dynamics to statistical physics, have been developed to investigate the complex many-body systems encountered in solid-state physics. These methods have successfully described many physical systems ranging from ionic crystals to semi-conductors and metals. There are, however, several systems for which analytical theories are unable to provide a complete description of the observed physical processes. In particular it is very difficult to solve analytically systems involving disorder or for which the harmonic approximation does not hold true. Examples of such systems include glasses, plastic crystals, liquids, liquid crystals and superionic conductors. In order to gain a deeper understanding of such systems we must take advantage of the computational power offered by todays high speed digital computers. Two computational techniques in particular have proved themselves to be particularly useful in describing complex physical systems; the techniques of Monte Carlo and molecular dynamics simulation (Bowler and Pawley, 1984)

Many physical quantities such as temperature, pressure, total magnetisation and susceptibility of magnetic systems etc. can be expressed analytically in terms of configurational averages. The Monte Carlo method (Metropolis *et al*, 1953) gives us a way of calculating such quantities on the computer. Many thousands of configurations of the system under investigation are generated, the probability of getting a particular configuration being give by $\exp(-\beta H)$, where H is the Hamiltonian of the configuration and $\beta=1/(kT)$. By generating configurations in this way, so that the low energy configurations occur more often than the high energy ones at a given temperature, it is possible to get a representative collection of configurations in configuration space without having to generate all possible configurations. It can be shown analytically that configurational averages calculated over the Monte Carlo generated configurations are equivalent to the true configurational averages. The Monte Carlo algorithm has been applied with great success to systems as diverse as the Ising model (Pawley *et al* 1984) and quantum chromo-dynamics (Bowler *et al* 1984).

Although Monte Carlo is very efficient at calculating time independent properties in the canonical ensemble, it cannot provide information on time-dependent quantities. It is the implicit time independence of Monte Carlo that makes it unsuitable for simulating superionic crystals as most of the quantities that we would like to be able to calculate in the simulation, such as hopping rates, the dynamical structure factor, diffusion coefficients etc. are time dependent. The molecular dynamics (MD) algorithm can give us access to such time dependent quantities. Given an initial configuration of the system, Newton's equations of motion can be solved numerically for all the particles in the simulation. By analysing the path of the configuration in phase space it is possible to calculate the time dependent properties of the system.

The first molecular dynamics simulations were carried out by Alder and Wainwright (1954) who simulated a system of elastic hard spheres. The only interactions included in the system were the collisions of the particles, the rest of the time the particles moved at constant velocities. The first MD simulations of a system of particles interacting via a realistic potential were carried out by Rahman (1964) on liquid argon.

Since the first simulations by Rahman the MD algorithm has been applied with great success to a wide range of complex systems including plastic crystals (Dove and Pawley 1983, Pawley 1981, Pawley and Dove 1983, Refson 1985) ionic melts (Sangster and Dixon 1976), liquids (Verlet 1967), micro-crystalites (Boyer 1986), physisorbed layers (Peters and Klein 1984), quantum systems (De Raedt *et al* 1984) and ionic glasses (De Leeuw and Thorpe 1986).

2.1. The Molecular Dynamics Algorithm

It is normally assumed in an MD simulation that the inter-particle potentials are pairwise, additive and central, i.e. that the total potential energy of the crystal, Φ, is given by:

$$\Phi = \sum_{i<j} \phi_{ij}(r_{ij}) \qquad (2.1)$$

where ϕ_{ij} is the potential between ions i and j which are separated by a distance $r_{ij}=|r_i-r_j|$

Given the total potential energy of the system, the force on particle i is given by:

$$\mathbf{F}_i(\{\mathbf{r}\}) = -\sum_{i<j} \nabla\phi_{ij}(\mathbf{r}_{ij}) \tag{2.2}$$

where the notation $\mathbf{F}_i(\{\mathbf{r}\})$ refers to the fact that the force on particle i can depend on the position of all the other particles in the system.

Given the forces on all the particles, the particle trajectories can be obtained by numerically integrating Newton's equations of motion:

$$(d^2\mathbf{r}_i/dt^2) = (1/m_i)\mathbf{F}_i(\{\mathbf{r}\}) \tag{2.3}$$

The numerical integration of the equations of motion will give us the total configuration of the system at successive timesteps. Time dependent properties of the system can be obtained explicitly from consecutive configurations, equilibrium properties can be obtained by averaging over long runs.

2.1.1. Initial Conditions

In order to minimise the time needed for the system to reach thermal equilibrium care should be taken in choosing this starting point. For a simulation of a solid it is sensible to start the simulation from an ordered phase, with the atoms being placed on the equilibrium lattice sites. The velocities can then be assigned randomly to the particles with values chosen from a Gaussian distribution with the constraint that the system has no net linear momentum.

$$\sum_i m_i\mathbf{v}_i = 0 \tag{2.4}$$

There are many possibilities for the boundary conditions to be placed on the MD cell from simulations of micro-crystalites to periodic boundary conditions imposed on a Wigner-Seitz cell (Sangster and Dixon 1976) and skew cyclic

boundary conditions (Refson 1985). An MD simulation is restricted in the number of ions that can be included in simulation, for simulations of more than a few thousand particles the computer time needed to update a configuration becomes impractically large, in practise many MD simulations are of only a few hundred particles. Using periodic boundary conditions, in which particles leaving the MD box through a particular face enters the simulation via an equivalent point on the opposite face (equivalent to filling space with period copies of the MD cell), can help to minimise these finite size effects.

The choice of boundary conditions can have a significant effect on the physics of the system, for a small number of ions in a simulation with no periodic boundary conditions then nearly all the ions will be surface ions, for the same simulation with periodic boundary conditions then none of the ions will be surface ions. Periodic boundary conditions can have more subtle effects on the physics of the simulation. Some of the consequences of periodic boundary conditions on the calculation of scattering functions will be described later on in this chapter.

2.1.2. Integrating the Equations of Motion

There are many techniques available for numerically integrating Newton's equations of motion, although in general the discrete approximations are linear multistep equations relating function values at discrete timelevels.

The simplest of these methods is the Verlet scheme (Verlet 1967). Taylor expanding the positional coordinates of the system we obtain:

$$\mathbf{r}_i(t+\delta t) = \mathbf{r}_i(t)+\mathbf{v}_i(t)\delta t+\mathbf{a}_i(t)\delta t^2/2 \qquad (2.5)$$

The velocity at time t can be written in terms of the positions via:

$$\mathbf{v}_i(t) = (\mathbf{r}_i(t+\delta t)-\mathbf{r}_i(t-\delta t)/(2\delta t) \qquad (2.6)$$

Substituting equation (2.6) into equation (2.5) we obtain:

$$\mathbf{r}_i(t+\delta t) = 2\mathbf{r}_i(t)-\mathbf{r}_i(t-\delta t)+\mathbf{a}_i\delta t^2 \qquad (2.7)$$

Using equation (2.7) we can calculate the positional coordinates at time $t+\delta t$ using our knowledge of the system at time t.

Another possible integration scheme is the leapfrog algorithm. This algorithm consists of three stages.

1. Calculate all the accelerations on the particles at time t, a(t).
2. Using the accelerations at time t calculate the velocity at time $t+\delta t/2$ using:

$$v(t+\delta t/2) = v(t-\delta t/2)+a(t)\ \delta(t) \qquad (2.8)$$

3. Calculate the positions at time $t+\delta t$:

$$x(t+\delta t) = x(t)+v(t+\delta t/2)\delta t \qquad (2.9)$$

It is often mistakenly believed that the Verlet and leapfrog algorithms are equivalent (Sangster and Dixon 1976), however this is not the case. Although both algorithms would be equivalent using infinite precision arithmetic, the leapfrog algorithm is a much more accurate than the Verlet algorithm when the effects of round-off error due to finite precision arithmetic are considered (BJ Pendleton and S Duane, private communication). This becomes obvious comparing equations (2.7) and (2.9). The Verlet algorithm computes the positions at $t+\delta t$ by adding a term of order δt^2 to a term of order 1, the leapfrog algorithm computes the positions at $t+\delta t$ by adding a term of δt to a term of order one. As δt is typically of the order 0.01 this means that the leapfrog algorithm will have much better round-off behaviour. ~~Also the leading error term in leapfrog is of order δt^3 compared with a leading error term of δt^2 in Verlet (as the leapfrog discretisation is more symmetric than the Verlet, the order δt^2 terms cancel in leapfrog but not in Verlet).~~ The leapfrog

integration algorithm is inherently more accurate than the Verlet algorithm.

Another integration algorithm is the one developed by Beeman (Schofield 1973). This algorithm has been found to give better energy conservation than the Verlet algorithm when dealing with the Lennard-Jones potential (Beeman 1976). Extending the Taylor series for $\mathbf{r}_i(t+\delta t)$ to third order then we get:

$$\mathbf{r}_i(t+\delta t) = \mathbf{r}_i(t)+\mathbf{v}_i(t)\delta t+\mathbf{a}_i(t)\delta t^2/2+\dot{\mathbf{a}}_i(t)\delta t^3/6 \qquad (2.10)$$

We can approximate the value of $\dot{\mathbf{a}}_i(t)$ by a finite difference between known values:

$$\dot{\mathbf{a}}_i(t) = (\mathbf{a}_i(t)-\mathbf{a}_i(t-\delta t)/\delta t \qquad (2.11)$$

Hence

$$\mathbf{r}_i(t+\delta t) = \mathbf{r}_i(t)+\mathbf{v}_i(t)\delta t+(1/6)\ [4\mathbf{a}_i(t)-\mathbf{a}_i(t-\delta t)]\delta t^2 \qquad (2.12)$$

As before the velocity can be found by finite differencing the positions.

$$\mathbf{v}_i(t+\delta t/2) = (\mathbf{r}_i(t+\delta t)-\mathbf{r}_i(t))/\delta t \qquad (2.13)$$

$$= \mathbf{v}_i(t)+[4\mathbf{a}_i(t)-\mathbf{a}_i(t-\delta t)]\delta t/6 \qquad (2.14)$$

We now add δt to all the times:

$$\mathbf{v}_i(t+3\delta t/2) = \mathbf{v}_i(t+\delta t)+[4\mathbf{a}_i(t+\delta t)-\mathbf{a}_i(t)]\delta t/6 \qquad (2.15)$$

Making the approximation:

$$\mathbf{a}_i(t+\delta t)\delta t = \mathbf{v}_i(t+3\delta t/2) -\mathbf{v}_i(t+\delta t/2) \qquad (2.16)$$

we can use equation (2.16) to express $\mathbf{v}_i(t+3\delta t/2)$ and equation (2.14) for

$\mathbf{v}_i(t+\delta t/2)$. Substituting these forms in equation (2.15) we obtain:

$$\mathbf{v}_i(t+\delta t) = \mathbf{v}_i(t)+[2\mathbf{a}_i(t+\delta t)+5\mathbf{a}_i(t)-\mathbf{a}_i(t-\delta t)]\delta t/6 \qquad (2.17)$$

Equations (2.12) and (2.17) together represent the Beeman algorithm. The Beeman algorithm has been found to be more energy stable than the Verlet algorithm. The increased stability of the Beeman algorithm is probably due to its more accurate calculation of the velocities and hence a more accurate calculated kinetic energy. This is demonstrated by Beeman's (1976) results for the simulation of argon-like atoms. His results showed that although the simulations run with the two different algorithms had the same overall drift in average total energy, the local energy fluctuations in the simulation using the Verlet algorithm were much larger than in the simulation run using the Beeman algorithm.

By using the explicit Runge-Kutta method (Henrici, 1962, Chapter 2) it is possible to calculate $\mathbf{r}_i(t+\delta t)$ to a higher order in δt (Hockney and Eastwood, 1981, Chapter4). The disadvantage of the Runge-Kutta type integration algorithms is that although they give good energy stability and do not require much knowledge of the past history of the system, they are expensive both in calculation time and storage. It can be shown (Hockney and Eastwood 1981) that integration algorithms which increase the accuracy of the integration to more than $O(\delta t)^2$ by referencing more timesteps than either the leapfrog or Beeman algorithms are inherently unstable. The problems arise when the number of characteristic solutions to the difference equation exceed the number of characteristic solutions of the differential equation being solved. In this case the extra 'parasitic' solutions of the difference equations swamp the true solutions of the physical system for all but the smallest values of the timestep. This effect is well demonstrated by the results quoted by Beeman (1976). The higher order difference schemes where more accurate than the low order schemes at a timestep of 0.01ps, but at a timestep of 0.03ps the low order schemes were significantly more stable to energy drift. By using a low order difference scheme such as Beeman or leapfrog the simulation can be run at higher values of δt, and hence more efficiently, than using the high order schemes.

2.1.3. Thermal Equilibrium

Initially the simulation is in a highly non-equilibrium state. The particles are on the lattice sites with them random velocities and so the balance between the potential and kinetic energies in the simulation will not be the same as the energy balance in an equilibrated system at the same temperature. In a normal MD simulation we are attempting to model the real system as closely as possible, therefore as real systems are usually in thermodynamic equilibrium we would like to generate configurations of the simulated systems which are also in equilibrium. A method is needed to determine just how quickly the simulation equilibrates.

There are three methods that are commonly used to determine when a system reaches thermal equilibrium:

1. Assume that after a few picoseconds of simulation time the system will have reached equilibrium. This is the commonest technique, mainly because of its simplicity.

2. Monitor the kinetic energy contained in the various degrees of freedom in the system and define the system to be in equilibrium when it reaches equipartition.

3. Monitor the fluctuations in kinetic energy of the system and define the simulation to be in equilibrium when the size of these fluctuations reaches a constant value

The least satisfactory of these methods is obviously the first, we simply assume that the system is anharmonic enough for it to equilibrate quickly. We have no way of knowing whether this is the case or not. It could be that data is being collected over unphysical non-equilibrated configurations or it could be that computer time is being wasted by equilibrating the system for too long. The second method is more satisfactory in that it does give us some measure on when the system equilibrates. However this method assumes that equipartition and equilibration happen at the same time. Equipartition and equilibration do not necessarily happen simultaneously, it is possible for a system to equipartition before it is truly equilibrated (see figure 2.1). Of the three methods for determining the onset of equilibration the third method is by far the most powerful. By examining the size of the fluctuations we can not

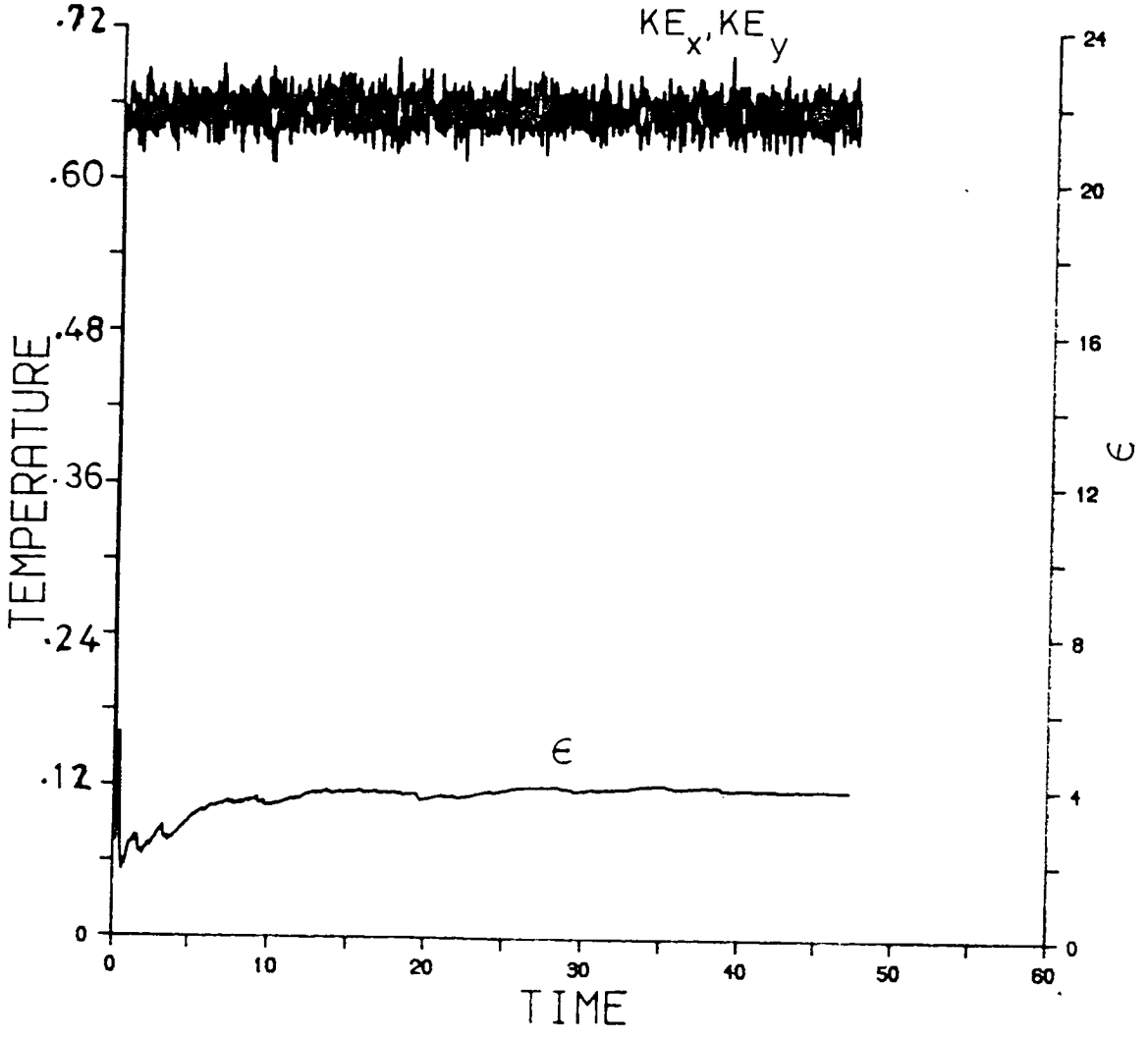


Figure 2.1 The data for these graphs came from a simulation of a two-dimensional system of 64x64 particles interacting via a Lennard-Jones potential with $\rho^* = .92$, $T^* = .64$ and $\delta t^* = .012$. Before the data was collected the system was run for 10 timesteps during which the velocities of the particles where rescaled so as to keep the temperature of the entire system equal to .64. The time shown on the x-axis of the graph is measured from the point at which the velocity rescaling was switched off. The top two plots show the kinetic energy of the system of the system in the x and y directions (measured in temperature units).By examining these two plots we can see that after the temperature rescaling the system has reached equipartition. Equipartition is reached after only ten timesteps. The bottom plot shows the function $\epsilon(t)$ calculated using equation 2.26. The value of $\epsilon(t)$ for an ergodic micro-canonical system is 4, $\epsilon(t)$ reaches this value at t=10. It takes significantly longer for the function $\epsilon(t)$ to reach its equilibrium value than it does for the simulation to reach equipartition. Once the value of $\epsilon(t)$ reaches 4 then we can be confident that our simulation has equilibrated properly.

only determine when the system equilibrates but also determine exactly which ensemble the system is in (e.g. is it canonical or micro-canonical) and whether the system is ergodic or non-ergodic.

2.1.3.1. Calculation of the Kinetic Energy Distribution

Using the techniques of statistical physics it is possible to determine the distribution of the total kinetic energy of the simulation in various ensembles (Fukugita *et al* 1985).

The partition function of a canonical system is given by:

$$\begin{aligned} Z &= \int dpdq \exp[-\beta H] \\ &= \int dE \exp[-\beta E] \int dpdq\ \delta(E-H) \\ &= \int dE\ \Omega(E) \exp[-\beta E] \end{aligned} \tag{2.18}$$

where p and q are the momentum and position coordinates of the system, β is proportional to the inverse temperature, H is the Hamiltonian of the system, E is the total energy of a configuration and Ω is the micro-canonical partition function. Using these functions the ensemble average of a variable A is given by:

$$\begin{aligned} \langle A \rangle_{mc} &= \Omega^{-1} \int dpdq\ A\ \delta(E-H) \\ \langle A \rangle_{c} &= Z^{-1} \int dpdq\ A \exp[-\beta H] \end{aligned} \tag{2.19}$$

where $\langle A \rangle_{mc}$ is the micro-canonical ensemble average and $\langle A \rangle_{c}$ is the canonical ensemble average.

Using equations (2.19) the kinetic energy (K) distribution can be defined by:

$$\begin{aligned} \rho_{mc}(K) &= \Omega(E)^{-1} \int \prod_i dp_i dq_i \delta(K-\Sigma p_i^2/2)\delta(E-H) \\ \rho_{c}(K) &= Z^{-1} \int \prod_i dp_i dq_i \delta(K-\Sigma p_i^2/2)\exp[-\beta H] \end{aligned} \tag{2.20}$$

Both these equations can be calculated analytically. For example:

$$\rho_{mc}(K) = \Gamma(N)/(\Gamma(N/2))^2(1/E)(K/E)^{N/2-1}(1-K/E)^{N/2-1} \quad (2.21)$$

where N is the number of degrees of freedom of the system. If N is large then equation (2.21) can be simplified.

$$\rho_{mc}(K) = (1/E)\sqrt{(2N/\pi)}\exp[-2(N-2)(K/E-.5)^2] \quad (2.22)$$

Note that this equation is only true for harmonic potentials. If the potential is not harmonic then (2.22) depends on the specific heat of the simulated compound (see equation 2.29)

Similarly

$$\rho_c(K) = (1/E)\sqrt{(N/\pi)}\exp[-(N-2)(K/E-.5)^2] \quad (2.23)$$

By examining equation (2.22) and (2.23) it can be seen that in both the micro-canonical and canonical ensembles the total kinetic energy has a Gaussian distribution. Note, however, that the width of the canonical distribution is $\sqrt{2}$ the width of the micro-canonical ensemble.

Fukugita *et al* also extended a similar analysis to the case of a time average, not a configurational average, of a harmonic Hamiltonian with incommensurate frequencies. In this case it is not obvious that the system will be ergodic, indeed in many cases the system can be non-ergodic (the KAM theory; Kolmogorov 1954, Arnold 1963 and Moser 1962). Again the distribution of the kinetic energy can be shown to be Gaussian:

$$\rho_t(K) = (2/E)\sqrt{(N/\pi)}\exp[-4N(K/E-.5)^2] \quad (2.24)$$

but with a width $\sqrt{2}$ times smaller than the width of the micro-canonical distribution.

Equations (2.22), (2.23) and (2.24) can be summarised as:

$$\begin{aligned} \langle K^2\rangle - \langle K\rangle^2 &= (1/2N)E^2 \text{ (canonical)} \\ \langle K^2\rangle - \langle K\rangle^2 &= (1/4N)E^2 \text{ (micro-canonical)} \\ \langle K^2\rangle - \langle K\rangle^2 &= (1/8N)E^2 \text{ (time average)} \end{aligned} \quad (2.25)$$

Note that E is the total energy of the system with the zero of the energy being defined as the energy of the system at zero temperature.

We now define a function $\varepsilon(t)$ by:

$$\varepsilon(t) = E(t)^2/[N(<K^2>-<K>^2)] \qquad (2.26)$$

where $\varepsilon(t)$ is the total energy of the system at time t divided by the number of degrees of freedom in the system and the variance squared of the total kinetic energy of the system over all the previous timesteps. If the value of $\varepsilon(t)$ is monitored then the system is defined to be in equilibrium when $\varepsilon(t)$ reaches a constant value. If $\varepsilon(t)$ settles down to a value of 4 then the simulation is ergodic and in the micro-canonical ensemble. If $\varepsilon(t)$ has a value greater than 4 then the system is non-ergodic, and if the value of $\varepsilon(t)$ is less than 4 then we are not in a micro-canonical ensemble and we should check again that our simulation is conserving energy. Figure 2.1 shows the results of various test for equilibration applied to a system of 64x64 particles interacting via a Lennard-Jones potential in two dimensions. Note that the results quoted above for the micro-canonical ensemble is only valid for harmonic potentials. If the potential is not harmonic then the value $\varepsilon(t)$ is a function of the specific heat of the compound (equation 2.29).

2.1.4. MD and Ensembles

The MD algorithms described thus far are all for simulations of systems in the micro-canonical ensemble; simulations for which the number of ions, the volume of the MD box and the total energy of the system are constant, but for which the pressure and temperature can fluctuate. Molecular dynamics does allow us the freedom to choose other ensembles (unlike Monte Carlo simulations which are always of the canonical ensemble). Of these alternative ensembles the most useful are the constant pressure, constant stress and constant temperature ensembles.

There are several in which a constant pressure can be maintained in a simulation. The most successful approach appears to be the one adopted by Parrinello and Rahman (1981), in which the unit vectors describing the MD volume are introduced as dynamical variables into the Hamiltonian of the system. A similar technique had been introduced by Anderson (1980) which allowed the system to maintain pressure by isotropically changing volume, but

which did not allow the MD cell to change shape anisotropically. The Parinello-Rahman algorithm removes this constraint and hence allows the system to maintain a zero-stress. This technique has proved to be particularly useful in simulating systems undergoing structural phase transitions (Parinello and Rahman 1980).

Two methods are commonly used to generate constant temperature MD simulations. The first is to use an isokinetic time integration algorithm (Brown and Clark 1984). These algorithms maintain a constant temperature by adding extra factors in the velocity rescaling step of the Verlet algorithm so as to maintain constant kinetic energy. Another way of maintaining a constant temperature in a simulation is to rewrite the Hamiltonian of the system so that that the temperature is a conserved quantity of the associated Lagrangian (Nose and Klein 1984).

2.1.5. Measurement of Time Dependent and Equilibrium Properties

The temperature of an MD simulation is defined via the total kinetic energy (K). Assuming that the equipartition theorem holds then for a system of N particles each with n degrees of freedom the temperature is given by:

$$T = 2\langle K\rangle/(nNk_B) \tag{2.27}$$

where

$$K = \sum_i m_i \mathbf{v}_i^2/2 \tag{2.28}$$

and $\langle..\rangle$ represents a configurational average. The assumption made in MD simulations is that the configurational average of a quantity can be approximated by a time average over a run. The validity of this approximation will be discussed in section 2.3.

By examining the fluctuations in the kinetic energy of a system in thermal equilibrium it is possible to calculate the specific heat at constant volume, C_v (Lebowitz *et al* 1967).

$$<K^2>-<K>^2 = (3/2)Nk_B^2T^2(1-(3/2)k_B/C_v) \quad (2.29)$$

The pressure in the system is calculated via the virial (Hansen 1976)

$$p = \rho k_B<T> - (1/3V)< \sum F_{ij}.r_{ij} > \quad (2.30)$$

The total potential energy Φ is calculated by summing up the potential energy of all the bonds

$$\Phi = \sum_{i<j} \phi(r_{ij}) \quad (2.31)$$

The total energy of the system is simply the sum of the kinetic and potential energies

The mean square displacements for ions of type β are calculated via the equation

$$<r_\beta^2(t)> = (1/N_\beta) \sum_i <|r_i(t)-r_i(0)|^2> \quad (2.32)$$

The time averaging <..> is to be understood as an average over time origins.

In order to calculate the radial distribution function, $g_{\alpha\beta}(r)$, we set up a vector BIN(N) on the computer where n is the number of points at which the distribution function is to be calculated. We divide the range of r (which is limited to half the smallest box dimension) into n equal parts each of length $\varepsilon=(L/2)/n$. We now calculate the distance between all pairs of particles α and β, $r_{\alpha\beta}$, incrementing the value of BIN(INT($r_{\alpha\beta}/\varepsilon$)) by 1 each time. If there are N particles of types α and β in a box of volume V then the value of $g_{\alpha\beta}(r)$ is given by:

$$g_{\alpha\beta}(r) = 2V/N^2[1/(4\pi r^2)]BIN(INT(r/\varepsilon))/\varepsilon \qquad (2.33)$$

With this definition of $g_{\alpha\beta}(r)$ then $g(r)\rightarrow 1$ as $r\rightarrow\infty$.

In this subsection a few of the methods used to calculate some of the physical quantities that can be obtained from an MD simulation have been described. However we have so far neglected to mention how to calculate some of the most useful functions that can be measured in an MD simulation, the scattering functions. The calculation of these functions will be discussed in section 2.3.

2.2. The Accuracy of the MD algorithm

The MD algorithm attempts to solve a system of several thousand coupled differential equations. It is in no way obvious that the algorithm will solve these equations to any degree of accuracy. There are several tests that can be applied to examine the accuracy of the algorithm. The first test we can apply is to calculate the conserved quantities of the appropriate Hamiltonian and look to see how well they are conserved in the simulated system. Secondly we can test the stability of the MD trajectories to round-off errors, if we examine the MD equations it is possible to see whether round-off errors introduced at one timestep decay or grow as the simulation proceeds.

In this section dimensionless units will be used to describe the properties of the simulation (Heyes 1983). If the Lennard-Jones potential is written in the form:

$$V(r) = 4\varepsilon[(\sigma/r)^{12}-(\sigma/r)^{6}] \qquad (2.34)$$

then time (t^*) is measured in units of $\sigma(m/\varepsilon)^{\frac{1}{2}}$, density ($\rho^*$) is measured in units of $N\sigma^d/V$ where d is the dimensionality of the system and V is the volume (3-d) or area (2-d) of the MD cell, temperature (T^*) is measured in units of k/ε and energy is measured in units of $1/\varepsilon$.

2.2.1. Conservation of Energy and Momentum

The MD algorithm can keep the conserved quantities of the appropriate Hamiltonian constant to a remarkable degree. One of the most surprising features of an MD simulation is the way that round-off errors seem to cancel rather than accumulate. Providing that the timestep being used is small enough (see section 2.2.2), the drift in the total energy per timestep for a micro-canonical simulation can be smaller than the arithmetic accuracy of the computer arithmetic. For example in a simulation of a 2-dimensional system of atoms interacting via a Lennard-Jones potential at $\rho^*=0.92$, $T^*=0.43$, and $\delta t=0.012$ over a run of 7000 timesteps the total energy drift per timestep was approximately one part in 10^9 even though the local energy fluctuations were of the order of 2 or 3 parts in 10^6. A similar degree of stability was also observed in the centre of mass of the system, the drift per timestep being less than 1 part in 10^8 of the inter-atomic spacing. The same simulation can be run at different values of the size of the timestep in order to investigate the energy drift per timestep as a function of the timestep. The results of these runs are shown in table 2.1.

TABLE 2.1 The variation of the drift in the total energy per timestep as a function of the size of the timestep.

Timestep (δt^*)	Drift in the Total Energy per Timestep
0.006	1.5×10^{-10}
0.012	1.1×10^{-9}
0.024	4.9×10^{-10}
0.030	1.9×10^{-8}
0.036	2.2×10^{-7}
0.042	3.2×10^{-6}

From table 2.1 it can be seen that the energy drift per timestep remains constant for a range of timestep values from $\delta t=0.006$ to 0.024. However for values of the timestep greater than 0.0237 the size of of the energy drift grows exponentially with the timestep until the simulation becomes completely energy unstable. There appears to be a critical value of the timestep, δt_c (=0.03 in this case). For any value of the timestep less than δt_c the simulation will be very stable with respect to energy drift, the only advantage of running

a simulation with a timestep much less than δt_c being that the local energy fluctuations will be smaller. For values of the timestep greater than δt_c then the simulation is unstable.

2.2.2. Error Propagation and Timestep Size

Because of the finite nature of computer arithmetic, any MD simulation will suffer from the effects of round-off. Adapting a technique described by Hockney and Eastwood (1981) it is possible to investigate how these round-off errors will propagate from timestep to timestep. Consider an MD simulation of N particles using the Beeman integration algorithms.

Define $\mathbf{x}^n$ to be a 3N vector describing the positions of the particles in the simulation n timesteps away from the start of the simulation. Define $\mathbf{X}^n$ to be the positions of the particles at the same time that would be obtained if the equations of motion were integrated to infinite precision. Assume that the integration algorithm we are using is exact except at the second timestep when a small rounding error, ϵ^2, is introduced.

$$\epsilon^2 = \mathbf{x}^2 - \mathbf{X}^2 \tag{2.35}$$

and that from the third timestep onwards no further rounding errors are introduced. It is now possible to examine the propagation of the initial error vector. Equation (2.7) can be rewritten as :

$$(\mathbf{X}^3+\epsilon^3)-2(\mathbf{X}^2+\epsilon^2)+\mathbf{X}^1 = \mathbf{F}(\mathbf{X}^2+\epsilon^2)(\delta t)^2/m \tag{2.36}$$

where the calculated positions are substituted for the true positions plus the error vector. Equation (2.36) is true for both the Verlet and Beeman algorithms. Calculating this equation for a few more timesteps and applying Taylor's theorem we can obtain a difference equation for the error terms.

$$\epsilon^{n+1}-2\epsilon^n+\epsilon^{n-1} = \epsilon^n(\partial \mathbf{F}(\mathbf{x})/\partial \mathbf{x}|_{\mathbf{x}=\mathbf{X}^n}\delta t^2/m) \tag{2.37}$$

This equation is as difficult to solve as equation (2.7) unless $\partial F(x)/\partial(x)$=const. As we are only interested in the stability we replace $\partial F(x)/\partial x)|_{x=X_n}$ by its maximum negative value to obtain the error propagation equation

$$\varepsilon^{n+1}-2\varepsilon^{n}+\varepsilon^{n-1} = -(1/m)|\partial F(x)/\partial x|_{MAX}(\delta t)^2\varepsilon^{n} = -\Omega^2(\delta t)^2\varepsilon^{n} \qquad (2.38)$$

where the -ve sign appears because of the assumption that equation (2.7) will have an oscillatory solution. Note that in making the approximation about the force derivative, we have effectively decoupled the differential equations (2.37). The error in the position of particle i at timestep n, ε_i^n is independent of the position of any of the other particles. We have turned the original 3N coupled, non-linear differential equations into 3N independent linear differential equations which are easily solved. Equation (2.38) is linear in ε and so will have a solution of the form $\varepsilon^n=\lambda^n=\exp(i\omega n\delta t)$. Substituting this trial solution into one of the equations in equation (2.38) we obtain a quadratic in ω:

$$\lambda^2-2\lambda+1 = -(\Omega\delta t)^2\lambda \qquad (2.39)$$

The general solution to equation (2.38) can therefore be written as

$$\varepsilon^n = a\lambda_+^n + b\lambda_-^n \qquad (2.40)$$

where λ_+ and λ_- are the two solutions of equation (2.39). If the error introduced at the second timestep is to decay then we require that

$$|\lambda_+| < 1, |\lambda_-| < 1 \qquad (2.41)$$

Examining the roots of equation (2.39) it can be seen that this condition is satisfied if

$$\delta t < \delta t_c = 2/\Omega = 2\sqrt{(m/|\partial F(x)/\partial x|_{MAX})} \qquad (2.42)$$

As an example of the typical size of δt_c consider a set of particles on a square lattice interacting via the Lennard-Jones potential

$$V(r) = - A/r^6+B/r^{12} \tag{2.43}$$

In order to calculate δt_c we must calculate the value of $|\partial F(x)/\partial x|_{MAX}$. After some calculation we obtain:

$$|\partial F(x)/\partial x|_{MAX} < \{d(6A/\Lambda^8-12B/\Lambda^{14})+$$
$$d^2(13a^2-10a\Lambda+8\Lambda^2)(48A/\Lambda^{10}-168B/\Lambda^{16}\} \tag{2.44}$$

where a is the lattice spacing, d is the dimensionality of the system and Λ is the minimum separation of any two particles in the simulation. If v_0 is defined as the velocity for which the probability of a particle having velocity greater than v_0=1/N, i.e. the largest velocity that we can reasonably expect at least one of the N particles to attain, then we can define Λ to be the distance apart two particles need to be for the pair potential to equal mv_0^2.

The Maxwell velocity distribution is given by:

$$dN_v/dv = 2N/(\sqrt{2\pi})(m/kT)^{3/2}v^2\exp(-mv^2/2kT) \tag{2.45}$$

where N_v is the number of particles in the system with velocity v. We can use this equation to determine v_0.

$$1 = \int_{v_0}^{\infty} \beta v^2\exp(-\alpha v^2)dv \tag{2.46}$$

where

$$\alpha = m/(2kT)$$
$$\beta = 2N/(\sqrt{2\pi})(m/kT)^{1.5}$$

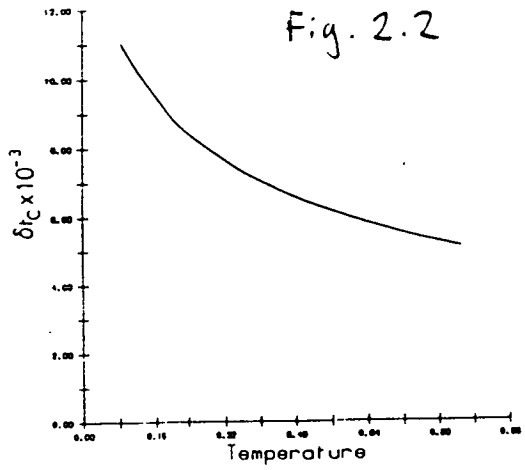
On performing the integral in equation (2.45) we get:

$$1 = a/(2\alpha)v_0\exp(-\alpha v_0^2)+A/(2\alpha)\sqrt{(\pi/\alpha)}\,\mathrm{erfc}(\sqrt{(\alpha)}v_0) \qquad (2.47)$$

where erfc is the complimentary error function. Using equation (2.47) it is possible to derive the value of v_0 for a system of any given size or temperature. Given v_0, Λ is found from the equation

$$(m/2)v_0^2 = -A/\Lambda^6+B/\Lambda^{12} \qquad (2.48)$$

This value of Λ can then be substituted in equation (2.44) to obtain a value for $|\partial F(x)/\partial x|_{MAX}$ which we can then use to find δt_c (equation (2.42).



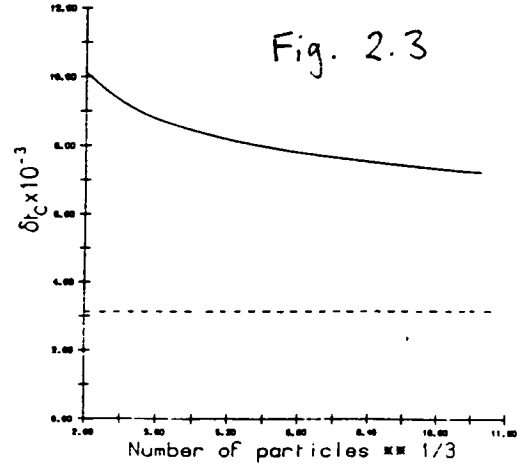


Figure 2.2 The variation in the size of the critical timestep size (δt_c) with the temperature of the simulation.

Figure 2.3 The variation in the size of he critical timestep size (δt_c) with the cube-root of the number of particles in the simulation of a cubic system of particles. The dashed line shows the value of δt_c we get for a system with Avagadros number of particles

Using the above equations we can investigate the variation of δt_c with both system size and temperature. Figure 2.2 shows the variation of δt_c with

temperature for a system of 64x64 particles at $\rho^* = 0.92$ and figure 2.3 shows the variation of δt_c with the number of particles in the system for a simulation at $\rho^* = 0.92$ and $T^* = 0.43$. By examining figures 2.2 and 2.3 we can see that the size of the critical timestep is close to 0.01 compared to the value 0.03 obtained from a computer simulation of the equivalent system (section 2.2.1). It would expected the calculated value of δt_c is lower than the experimental value as all the way through the above calculations worst case approximations have been used. From the two graphs we can see that the value of τ_c is only weakly dependent on the size and temperature of the simulated system other than at low temperatures or small system sizes. In fact if we have Avagadro's number of particles in the system then the value of δt_c is only half the value obtained for 64x64 atoms. The graphs do show, however, that in going from a simulation of the solid to a simulation of the liquid (the system described above was found to be liquid at $T^* = 1.0$) the value of the timestep should be reduced by a factor of about a half and similarly that in going from simulations of small systems to larger systems the timestep might have to be decreased by as much as 25%.

2.2.3. Stability of MD Trajectories

We can use a velocity cross-correlation function to test how fast the MD trajectories are liable to diverge from the true trajectories. (Corbin 1985). If we start two simulations at the same point in phase space and then integrate them forward in time using two different timesteps, then the value of the velocity cross-correlation function between these two configuration gives us a measure of the separation between the two different configurations. The time that the two configurations stay correlated gives us an upper bound on the time that the MD generated trajectories stay close to the true trajectories.

The velocity cross-correlation function between two simulations is given by:

$$C(t) = \langle \mathbf{v}_i^1 . \mathbf{v}_i^2 \rangle / v_i^1 v_i^2 \qquad (2.49)$$

where $\mathbf{v}_i^1$ is the velocity of particle i at time t in one of the simulations and the angular brackets denote an average over all atoms in the system. Figure 2.4 shows the velocity cross-correlation function calculated for two different

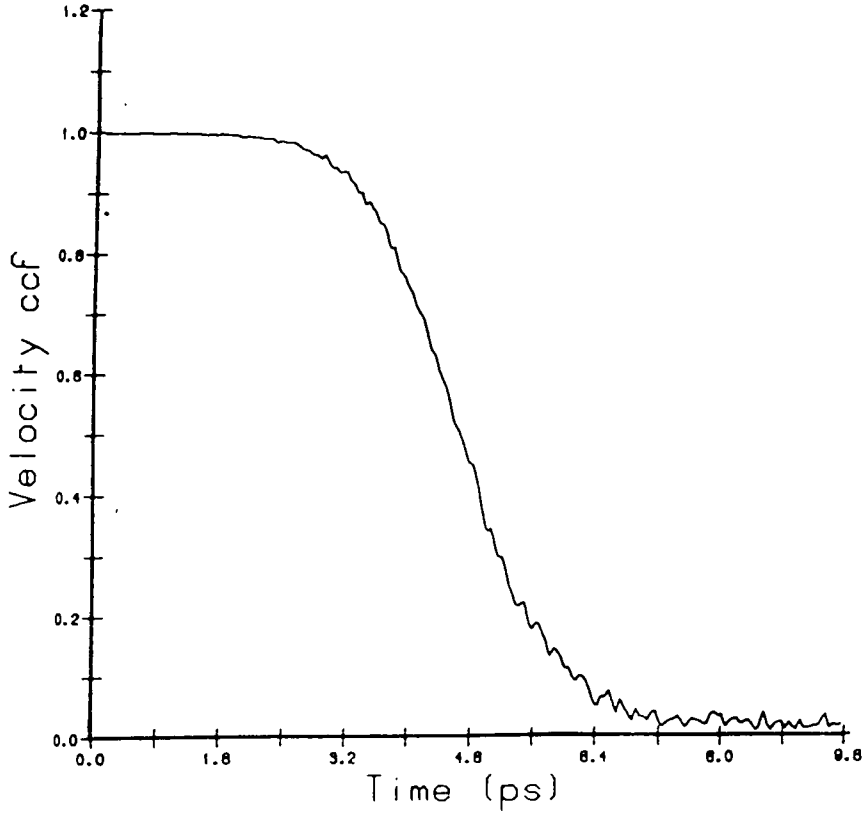


Figure 2.4 Velocity cross-correlation function (ccf) vs. time for a simulation of particles interacting via a Lennard-Jones potential in two-dimensions at $\rho^* = .92$ and $T^* = .43$.

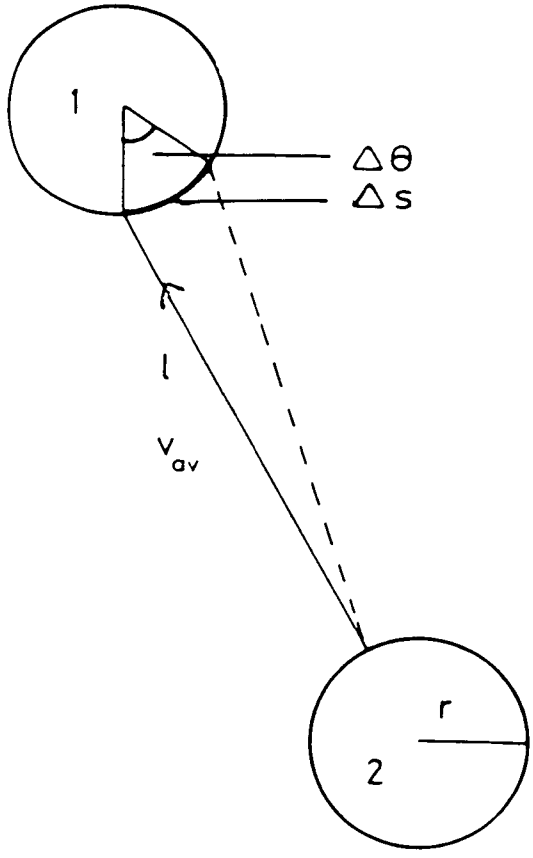


Figure 2.5 The collision of two hard spheres.

simulations of a 2-dimensional system interacting via a Lennard-Jones potential at ρ^*=0.92 and T^* = 0.42. From the form of the correlation function it appears that the two systems remain well correlated up until about 3ps at which point the correlation decays rapidly to zero. Corbin wondered whether this effect might in some way be due to inadequacies in the MD algorithm. However it can be demonstrated that this rapid decay in the correlation function is due not to inadequacies of the MD algorithm but rather to instabilities which are implicit in the original model. Any system of interacting spheres is inherently unstable with respect to any perturbation.

To demonstrate this instability consider oxygen at NTP. Neglecting the gravitional interaction of an electron at the edge of the known universe with the oxygen atoms introduces an error in the angle of motion of any oxygen atom that has undergone 56 collisions of one radian (Berry). A similar analysis to that used by Berry can be applied to an MD simulation of hard spheres.

Consider the position shown in figure 2.5. Assume that the position of the point of impact between atoms 1 and 2 is uncertain by Δs (perhaps due to round-off errors). The uncertainty in the angle of reflection after one collision will be:

$$\Delta\theta_1 = \Delta s/r \tag{2.50}$$

where r is the interaction radius of an atom. Molecule 1 will now travel on average a distance ℓ before its next collision where ℓ is the mean free path of the system. The uncertainty in the point of collision for this new collision will be:

$$\Delta s_2 = \ell\Delta\theta_1 = \ell\Delta s/r \tag{2.51}$$

The uncertainty in the angle of reflection will be:

$$\Delta\theta_2 = (\ell/r)(\Delta s/r) \tag{2.52}$$

After n collisions the uncertainty in the angle of reflection will be:

$$\Delta\theta_n = (1/r)^{n-1}\Delta s/r \tag{2.53}$$

Consider the effect of this uncertainty in the direction of motion on the velocity cross correlation function.

In effect equation (2.49) measures the average value of $\cos(\theta_i)$ where θ_i is the difference in angle between the directions of motion of the same atoms in simulations 1 and 2. Assume that at time t each atom has undergone n collisions then the uncertainty in the angle between the direction of motion in the two simulations will be ε where $\varepsilon=\Delta\theta_n$. If we assume that the uncertainty in the angle is evenly distributed over the range $0-\varepsilon$ and that the number of particles in the system $N \to \infty$ then:

$$C(t) = 1/N \sum \cos(\theta_i) \tag{2.54}$$

$$= \int_{-\varepsilon}^{\varepsilon} \cos(\theta)d\theta/(2\varepsilon) \tag{2.55}$$

$$= \sin(\varepsilon)/\varepsilon \tag{2.56}$$

We can calculate the approximate form of C(t) for a systems of particles interacting via a Lennard-Jones potential at $\rho^*=0.92$, $T^*=0.43$ by assuming that the particles in the simulation can be approximated by hard spheres of radius r, where r is the separation of two particles at which pair potential of the two particles is equal to their average kinetic energy. Approximate the mean free path ℓ by the average inter-particle spacing a. Δs is set to be round-off error (= 1×10^{-6}). Using these parameters it is possible to calculate the velocity cross-correlation function as a function of the average number of collisions undergone by a particle.

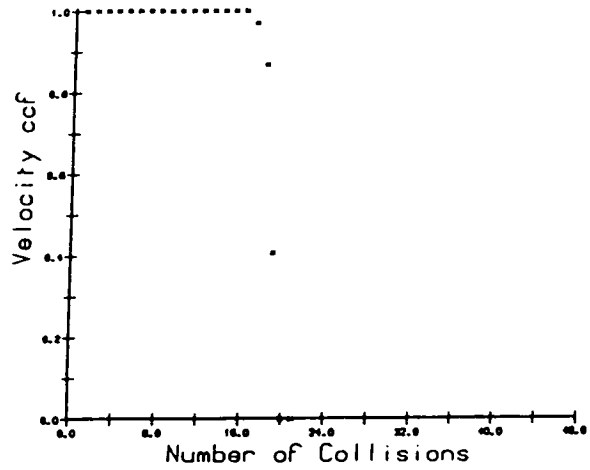


Figure 2.6 The value of the velocity ccf vs. the number of collisions for a system at the same state point as in figure 2.4.

Figure 2.6 shows the calculated form of the correlation function. It can be seen from the comparison of figures 2.4 and 2.6 that the form of the velocity cross-correlation function as calculated via the above simple analysis and via the computer simulation are very similar. Decreasing the round-off error in the above calculation from $1x10^{-6}$ to $1x10^{-12}$ increases the number of collisions needed for the two trajectories to diverge and the correlation to decay to zero by a factor of two. In fact it is possible to show that the divergence of the trajectories goes exponentially as the round-off error (Powles 1986), increasing the accuracy of the arithmetic will only have a small effect on the time taken for C(t) to reach zero.

As the decay in the velocity cross-correlation is an inherent feature of systems with spherically symmetric potentials, the fact that the velocity cross-correlation function as calculated via an MD simulation also decays rapidly does not point to any breakdown of the MD method. In fact for a 2-d system of particles at $\rho^*=0.92$, $T^*=0.12$, which is well into the solid phase and for which many of the approximations made in the preceding analysis are false, the velocity cross-correlation function stayed at unity for approximately 20ps, suggesting that the trajectories followed by the simulation through phase space follow the true trajectory for a similar length of time. The rapid decay in the correlation function observed by Corbin would appear to be

feature of the system itself rather than an inherent instability in the MD simulation algorithm.

2.3. Scattering Functions and Molecular Dynamics

A large fraction of the information that has been obtained about crystalline materials has been obtained via experiments in which neutrons or X-rays are scattered off a sample of the solid. By examining the various scattering cross-sections obtained in such experiments it is possible to obtain information on the crystal structure, the collective dynamics of its constituent particles and single particle properties. For example if neutrons are scattered off the crystal it is possible to measure three separate scattering cross-sections; the coherent elastic, coherent inelastic and incoherent scattering cross-sections. The coherent elastic scattering provides information about the structure of the crystal, the inelastic coherent scattering provides information on the dynamics of the whole crystal and the incoherent scattering provides information on single particle properties.

All these functions can be conveniently expressed in terms of the dynamical structure factors, $S(Q,\omega)$. The differential cross section for coherent inelastic neutron scattering is given by (Marshall and Lovesy 1971):

$$(\partial^2\sigma/\partial E\,\partial\Omega)_{coh} = (k_f/\hbar k_i) \sum_{\alpha,\beta} \sqrt{N}_\alpha \sqrt{N}_\beta b_\alpha b_\beta S^{coh}_{\alpha\beta}(\mathbf{q},\omega) \qquad (2.57)$$

where k_i and k_f are the initial and final neutron wavevectors, $\hbar\mathbf{q}$ and $\hbar\omega$ represent the momentum and energy transfers, and α and β label the ionic species with N_α and N_β being the number of atoms of these species with mean scattering lengths of b_α and b_β. A similar expression also gives the incoherent inelastic scattering cross section with $S^{coh}_{\alpha\beta}$ being replaced by $S^{inc}_{\alpha\beta}$. The functions $S_{\alpha\beta}(\mathbf{q},\omega)$ describe the dynamics of the system. The other coherent scattering functions can be expressed in terms of $S^{coh}_{\alpha,\beta}(\mathbf{q},\omega)$; the coherent elastic scattering function is simply given by $S(\mathbf{q},0)$, and the coherent X-ray scattering function can be expressed in terms of:

$$S(\mathbf{q}) = \int S(\mathbf{q},\omega)d\omega. \tag{2.58}$$

2.3.1. Calculation of S(q,ω) in an MD Simulation

There are several techniques that can be used to calculate the coherent and incoherent S(q,ω) in an MD simulation. $S^{coh}_{\alpha\beta}(\mathbf{q},\omega)$ is the time Fourier transform of the intermediate scattering function $F^{coh}_{\alpha\beta}(\mathbf{q},t)$.

$$S^{coh}_{\alpha\beta}(\mathbf{q},\omega) = (1/2\pi) \int_{-\infty}^{\infty} dt\, e^{i\omega t} F^{coh}_{\alpha\beta}(\mathbf{q},t) \tag{2.59}$$

where $F^{coh}_{\alpha\beta}(\mathbf{q},t)$ are the time dependent density fluctuation correlation function at wavevector **q**.

$$F^{coh}_{\alpha\beta}(\mathbf{q},t) = \langle \rho_\alpha(\mathbf{q},t), \rho_\beta(\mathbf{q},0)^* \rangle \tag{2.60}$$

The functions $\rho_\alpha(\mathbf{q},t)$ are the positional Fourier transforms of the density of particles of type α.

$$\rho_\alpha = (1/\sqrt{N_\alpha}) \sum_i \exp(i\mathbf{q}.\mathbf{r}_{\alpha i}(t)) \tag{2.61}$$

where $\mathbf{r}_{\alpha i}(t)$ is the position of ion i of species α at time t.

$S^{inc}{}_{\alpha\beta}(\mathbf{q},\omega)$ can be calculated in a similar way, substituting $F^{inc}_{\alpha\beta}$ for $F^{coh}_{\alpha\beta}$ in equation (2.59) where $F^{inc}_{\alpha\beta}$ are the positional Fourier transforms of the displacements of all the particles from their lattice sites:

$$F^{inc}_{\alpha}(\mathbf{q},t) = \langle \rho'_\alpha(\mathbf{q},t), \rho'_\alpha(\mathbf{q},0)^* \rangle \tag{2.62}$$

where ρ'_α is given by:

$$\rho'_\alpha = (1/\sqrt{N_\alpha}) \sum_i \exp(i\mathbf{q}.(r_{\alpha i}(t) - r_{0i}(0)) \quad (2.63)$$

where $\mathbf{R}_i$ are the coordinates of the nearest lattice site to particle i.

Using equations (2.59) to (2.63) the coherent or incoherent $S_{\alpha\beta}(\mathbf{q},\omega)$ can be calculated from the positions of the particles in the simulatulation at every timestep. Using the particle positions $\rho_\alpha(\mathbf{q},t)$ can be directly calculated using equation (2.61) (or equation (2.63) for the incoherent $S(\mathbf{q},\omega)$). The intermediate scattering functions for each time difference t can then be calculated by averaging the transforms over all time origins. The dynamical scattering functions are then calculated by numerically time Fourier transforming the appropriate coherent or incoherent intermediate scattering functions. Were we simply to cut off the data at t_{max} termination ripples would be seen in the Fourier transformed data. In order to avoid these ripples we can multiply the data to be transformed by a window function which is unity for small times but decays smoothly to zero at t_{max}. Note that we are assuming that we can replace the ensemble average in equations (2.60) and (2.62) by time averages. It will be shown later that this approximation is valid provided that the data is collected for a time T which is greater than twice the time period of the lowest frequency phonon, and that the simulation itself is ergodic, i.e. that the simulation samples the entire H=E surface in phase space. Making these assumptions then it is possible to reduce the amount of calculation needed to calculate $S(\mathbf{q},\omega)$ (Hansen and Klein 1974).

$S(\mathbf{q},\omega)$ is the time transform of the intermediate scattering function $F(\mathbf{q},t)$ i.e.

$$S(\mathbf{q},\omega) = 1/N \int_{-\infty}^{\infty} e^{i\omega t} \langle \rho(\mathbf{q},t)\rho^*(\mathbf{q},0)\rangle dt \quad (2.64)$$

Replacing the ensemble average by the time average over all time origins we obtain:

$$S(\mathbf{q},\omega) = 1/N \int_{-\infty}^{\infty} dt\, e^{i\omega t} \int dt' \, \rho(\mathbf{q},t'+t)\rho^*(\mathbf{q},t') \quad (2.65)$$

Note that equation (2.65) is the Fourier transform of a convolution. Using the convolution theorem, equation (2.65) can be expressed as:

$$S(\mathbf{q},\omega) = 1/N \int_{-\infty}^{\infty} dt\ e^{i\omega t}\ \rho(\mathbf{q},t) \oplus \rho^{*}(\mathbf{q},t) \tag{2.66}$$

$$= 1/N\ \rho(\mathbf{q},\omega)\rho^{*}(\mathbf{q},\omega) \tag{2.67}$$

$$= 1/N\ |\rho(\mathbf{q},\omega)|^2 \tag{2.68}$$

where $\oplus$ represents convolution and $\rho(\mathbf{q},\omega)$ is the time Fourier transform of $\rho(\mathbf{q},t)$.

Using the definition of $\rho(\mathbf{q},t)$ given in equation (2.63) we calculate the multi-phonon $S(\mathbf{q},\omega)$. However writing the position of ion i at time t, $\mathbf{r}_i(t)$, as:

$$\mathbf{r}_i(t) = \mathbf{R}_i + \mathbf{u}_i(t) \tag{2.69}$$

where $\mathbf{R}_i$ is $<\mathbf{r}_i(t)>$ and $\mathbf{u}_i(t)$ is the the displacement of i from $\mathbf{R}_i$ at time t, then the exponential in equation (2.61) can be expanded as:

$$\rho(\mathbf{Q},t) = \sum_{l} \exp(i\mathbf{q}.\mathbf{R}_l) \sum_{n} (i^n/n!)[\mathbf{Q}.\mathbf{u}_l]^n \tag{2.70}$$

Writing

$$\rho(\mathbf{Q},t) = \rho^0(\mathbf{Q},t) + \rho^1(\mathbf{Q},t) + \ldots + \rho^n(\mathbf{Q},t) + \ldots. \tag{2.71}$$

where:

$$\rho^n(\mathbf{Q},t) = \sum_{l} \exp(i\mathbf{Q}.\mathbf{R}_l)(i^n/n!)[\mathbf{Q}.\mathbf{u}_l]^n \tag{2.72}$$

we define $F^{nm}(\mathbf{Q},t)$ to be equal to:

$$F^{nm}(\mathbf{Q},t) = <\rho^n(\mathbf{Q},t)\rho^m(\mathbf{Q},t)^{*}> \tag{2.73}$$

Using these definitions then the one-phonon intermediate scattering function is given by $F^{11}(\mathbf{Q},t)$. Note that the multi-phonon intermediate scattering function can be written as:

$$F(Q,t) = <(\rho^0(Q,t)+\rho^1(Q,t)+....)(\rho^0(Q,t)^*+\rho^1(Q,t)^*+....)> \quad (2.74)$$

i.e.

$$F(Q,t) = F^{00}(Q,t)+F^{10}(Q,t)+F^{01}(Q,t)+... \quad (2.75)$$

2.3.2. Properties of the Scattering Functions

Assuming that the simulation is ergodic and symmetric under time reversal, then scattering functions calculated using the equations given in the previous subsection will have the following properties:

a) $\rho^*(\mathbf{q},t) = \rho(-\mathbf{q},t)$

This property follows from the definition of $\rho(\mathbf{q},t)$ given in equation (2.61).

b) $F(\mathbf{q},t)$ is a real valued function.

$$\begin{aligned} F^*(\mathbf{q},t) &= < \rho(\mathbf{q},t)\rho^*(\mathbf{q},0) >^* \\ &= < \rho^*(\mathbf{q},t)\rho(\mathbf{q},0) > \\ &= < \rho^*(\mathbf{q},0)\rho(\mathbf{q},-t) > \\ &= <\rho(\mathbf{q},-t)\rho^*(\mathbf{q},0) >. \\ &= F(\mathbf{q},-t) = F(\mathbf{q},t) \end{aligned} \quad (2.76)$$

The last equality follows from the time reversal symmetry. The functions $F^{nn}(\mathbf{q},t)$ must also be real by a similar argument.

c) $S(\mathbf{q},\omega)$ is a real function.

$S(\mathbf{q},\omega)$ is the Fourier transform of a real, even function and therefore must itself be a real function.

2.3.3. Scattering Functions in the Harmonic Approximation

We now have a prescription for calculating the intermediate scattering functions, and hence the scattering cross-sections, from an MD simulation. In order to examine the consequences of the various assumptions we have made in calculating the intermediate scattering functions an analytic form must be found for $F(\mathbf{Q},t)$ in terms of the normal modes of the system.

Making the harmonic approximation the displacement of an ion from its lattice site, $\mathbf{u}_l(t)$,can be expanded in terms of the normal modes of the lattice.

$$\mathbf{u}_l(t)= \sum_{q} a_q E_q \mathrm{Re}\{\exp(-i[\mathbf{q}.\mathbf{R}_l-\omega_q t-\phi_q])\} \tag{2.77}$$

where E_q is the eigenvector of mode $\mathbf{q}$ which has frequency ω_q, amplitude a_q and phase-factor ϕ_q. Substituting equation (2.77) into equation (2.72) with n set equal to 1 we can express the one phonon density function $\rho^1(\mathbf{Q},t)$ as:

$$\rho^1(\mathbf{Q},t) = (1/\sqrt{N})\sum_{l=1}^{N} \exp(i\mathbf{R}_l.\mathbf{Q})\mathbf{Q}.\left[\sum_{q} a_q E_q \mathrm{Re}\{\exp(-i[\mathbf{q}.\mathbf{R}_l-\omega_q t-\phi_q])\}\right] \tag{2.78}$$

Using the identity:

$$\mathrm{Re}\{\exp(ix)\} = \cos(x) =(1/2)[\exp(ix)+\exp(-ix)]$$

equation (2.78) becomes:

$$\rho^1(\mathbf{Q},t) = (1/2\sqrt{N}) \sum_{q} a_q(E_q.\mathbf{Q}) \times$$
$$\sum_{l=1}^{N} \left(\exp(i[\mathbf{Q}-\mathbf{q}].\mathbf{R}_l)\exp(i[\omega_q t+\phi_q])+\exp(i[\mathbf{Q}+\mathbf{q}].\mathbf{R}_l)\exp(-i[\omega_{-q} t+\phi_{-q}])\right) \tag{2.79}$$

If we assume that $a_q=a_{-q}$ then $\mathbf{q}$ in the second term of equation (2.79) can be replaced by $-\mathbf{q}$. The sum over l turns the exponential terms which include $(\mathbf{Q}-\mathbf{q}).\mathbf{R}_l$ into δ-functions.

$$\rho^1(\mathbf{Q},t) = (\sqrt{N}/2) \sum_{q,G} a_q(\mathbf{E}_q\cdot\mathbf{Q})\delta(\mathbf{Q}-\mathbf{q}-\mathbf{G})\{\exp(i[\omega_q t+\phi_q])+\exp(-i[\omega_q t+\phi_{-q}])\} \quad (2.80)$$

where **G** are reciprocal lattice vectors and we assume that $\omega_q=\omega_{-q}$. Equation (2.80) can finally be expressed as:

$$\rho^1(\mathbf{Q},t) = (\sqrt{N}/2) \sum_{q,G} a_q(\mathbf{E}_q\cdot\mathbf{Q})\delta(\mathbf{Q}-\mathbf{q}-\mathbf{G})\exp(i[\phi_q-\phi_{-q}]/2)\cos[\omega_q t+(\phi_q+\phi_{-q})/2] \quad (2.81)$$

This expression for $\rho^1(\mathbf{Q},t)$ can now be used to calculate the one-phonon intermediate scattering function $F^{11}(\mathbf{Q},t)$. Using equation (2.81) we can compare F^{11} as calculated via a true ensemble average with F^{11} as calculated via a time average. We define $F^{11}(\mathbf{Q},t)_t$ to be the value of the scattering function calculated using a time average and $F^{11}(\mathbf{Q},t)_e$ to the value of the scattering function obtained using an ensemble average.

In order to calculate both these functions terms such as

$$\rho^1(\mathbf{Q},t+\tau)\rho^1(-\mathbf{Q},\tau) \quad (2.82)$$

should be considered. Substituting equation (2.81) into equation (2.82) we obtain:

$$\rho(\mathbf{Q},t+\tau)\rho(\mathbf{Q},\tau)^* = (N/4) \sum_{q,G}\sum_{q',G'}(a_q a_{q'})(\mathbf{Q}\cdot\mathbf{E}_q)(-\mathbf{Q}\cdot\mathbf{E}_{q'})\delta(\mathbf{Q}-\mathbf{q}-\mathbf{G})\delta(-\mathbf{Q}-\mathbf{q}'-\mathbf{G}') \times$$
$$\exp(i[\phi_q-\phi_{-q}])\exp(i[\phi_{q'}-\phi_{-q'}])\cos[\omega_q(t+\tau)+(\phi_q+\phi_{-q})/2]\cos[\omega_{q'}(t)+(\phi_{q'}+\phi_{-q'})/2] \quad (2.83)$$

This expression will only be non-zero when the arguments of the two δ-functions are simultaneously zero, i.e. when:

$$\mathbf{Q} = \mathbf{q}-\mathbf{G}$$

$$\mathbf{Q} = -\mathbf{q}'-\mathbf{G}'$$

In order that the product of the two δ-functions does not give us zero $\mathbf{q}'$ must be equal to $-\mathbf{q}+\mathbf{G}$, where **G** is some reciprocal lattice vector. As the

effect of adding a reciprocal lattice vector to a **q**-vector is to move the vector into the equivalent position in another Brillouin zone, then **q**+**G** will be equivalent to **q** in the above equations. Substituting $\mathbf{q}'=-\mathbf{q}$ into equation (2.83) we get:

$$\rho(\mathbf{Q},t+\tau)\rho(\mathbf{Q},\tau)^{*} = (N/4)\sum_{q}\sum_{G} a_q^2(\mathbf{Q}\cdot\mathbf{E}_q)^2\delta(\mathbf{Q}-\mathbf{q}-\mathbf{G}) \times$$
$$\cos[\omega_q(t+\tau)+(\phi_q+\phi_{-q})/2]\cos[\omega_q(t)+(\phi_{-q}+\phi_q)/2] \qquad (2.84)$$

Note that the exponential phase factors all cancel out. Expanding the cosine terms we finally obtain:

$$\rho(\mathbf{Q},t+\tau)\rho(\mathbf{Q},\tau)^{*} = (N/4)\sum_{G}\sum_{q} (a_q^2(\mathbf{Q}\cdot\mathbf{E}_q)^2\delta(\mathbf{Q}-\mathbf{q}-\mathbf{G}) \times$$
$$\{\cos[\omega_q(t+2\tau)+(\phi_q+\phi_{-q})]+\cos[\omega_q(t)]\} \qquad (2.85)$$

Using equation (2.85) $F^{11}(\mathbf{Q},t)_e$ can be compared to $F^{11}(\mathbf{Q},t)_t$.

$F^{11}(\mathbf{Q},t)_e$ is defined by:

$$F^{11}(\mathbf{Q},t)_e = \langle \rho^1(\mathbf{Q},t)\rho^1(-\mathbf{Q},0)\rangle \qquad (2.86)$$

This function can be calculated by taking the ensemble average of equation (2.85) with $\tau=0$. On performing ensemble average we must calculate an integral over all ϕ_q, when we do this integral the first cosine term in equation (2.85) vanishes. Therefore:

$$F^{11}(\mathbf{Q},t)_e = \langle (N/4)\sum_{q}\sum_{G}(a_q^2(\mathbf{Q}\cdot\mathbf{E}_q)^2\delta(\mathbf{Q}-\mathbf{q}-\mathbf{G})\cos[\omega_q(t)]\rangle \qquad (2.87)$$

$F^{11}(\mathbf{Q},t)_e$ is indeed a real function.

$F^{11}(\mathbf{Q},t)_t$ is defined by:

$$F^{11}(\mathbf{Q},t)_t = (1/T)\int_{-T/2}^{T/2} \rho(\mathbf{Q},t+\tau)\rho(\mathbf{Q},\tau)^{*}d\tau \qquad (2.88)$$

Substituting equation (2.85) into equation (2.88) we obtain:

$$F^{11}(\mathbf{Q},t)_t = (1/T)\int_{-T/2}^{T/2}(N/4)\sum_{G}\sum_{q}(a_q^2(\mathbf{Q}.\mathbf{E}_q)^2\delta(\mathbf{Q}-\mathbf{q}-\mathbf{G})\times \{\cos[\omega_q(t+2\tau)+(\phi_q+\phi_{-q})]+\cos[\omega_q(t)]\}\,d\tau \qquad (2.89)$$

On performing the integration over τ this becomes:

$$F^{11}(\mathbf{Q},t)_t = (N/4)\sum_{G}\sum_{q}(a_q^2(\mathbf{Q}.\mathbf{E}_q)^2\delta(\mathbf{Q}-\mathbf{q}-\mathbf{G})\times \{[\sin(\omega T)/\omega T]\sin(-\omega_q T+\Phi)+\cos(\omega_q t)\} \qquad (2.90)$$

Note that $F^{11}(\mathbf{Q},t)_t$ is not equivalent to $F^{11}(\mathbf{Q},t)_e$. In doing the time integral an extra term is generated, we have added an extra term to the $\cos(\omega_q t)$ term. This extra term will be negligible if $\omega T << 2$, i.e it will be negligible provided that data is collected for a time greater than twice the time period of the lowest frequency phonon. Note that this means that the greater the number of particles in the simulation the longer the time over which the data on the density fluctuations must be collected. If we consider an acoustic phonon branch close to the zone centre then the frequency of modes on this branch will depend linearly on the **q**-vector of the mode. The lowest allowed **q**-vector in the system is directly proportional to the cube root of the number of particles in the system. Therefore if we have a large system the acoustic mode with the **q**-vector closest to the zone-centre will have a low frequency therefore we must collect the density data for longer. We would expect that the length of time over which we must collect density fluctuation data in order that $F^{11}(\mathbf{Q},t)_t$ is equivalent to $F^{11}(\mathbf{Q},t)_e$ would increase as the cube root of the number of particles in the system.

Finally it should be noted that with the approximations made at the start of this section both $F^{11}(\mathbf{Q},t)_e$ and $F^{11}(\mathbf{Q},t)_t$ are real functions.

2.3.4. Calculation of F(Q,t)

Using the techniques described in the previous sections we can now calculate $F(\mathbf{Q},t)$ for any given MD simulation. In most cases this calculation is very

straightforward and many simulations have been carried out in which the one-phonon intermediate scattering functions have been calculated and then used to calculate the appropriate scattering cross-sections.

However, in examining the results given for F(Q,t) in some papers it is obvious that problems can occur in calculating the intermediate function (Dove *et al* 1986); in some circumstances the intermediate scattering function, which we showed to be real in the previous section, is found to have an oscillating imaginary component. This component can be shown to be more than just a noise term and does not go away with improved statistics. Even in a system as apparently straightforward as 4096 particles interacting via a Lennard-Jones potential in 2-d, we can demonstrate the existence of such an imaginary component of F(Q,t). Figure 2.7 shows the value of $F^{11}(Q,t)$ calculated for the system at $\rho^*=0.92$ and $\delta t^*=0.012$ for various values of Q and T^*. It can be seen from these figures that the imaginary component of the intermediate scattering function is present at all q-vectors and at all temperatures, even at temperatures close to the melting temperature (this system was a liquid at $T^*=0.8$). By examining the frequency spectrum of the real and imaginary parts of the scattering function it is possible to show that they both oscillate at exactly the same frequency. Note also the appearance of what seem to be beats in the amplitudes of the oscillations of the scattering functions with a beat period of approximately 6-10ps. At q-vectors close to the zone boundary the intermediate scattering functions decays rapidly in the first three or four ps (as would be expected from the results of section 2.2.3 in which it was shown that MD configurations of particles interacting via the Lennard-Jones potential in 2-d tend to stay correlated for about 3ps) before entering a regime in which the amplitude seem to be dominated by a beat frequency. For q-vectors close to the zone centre then the amplitude of F(Q,t) shows no sign of decaying; at t=20ps the amplitude of the scattering function is comparable to amplitude of the scattering function at t=0. This behaviour of the intermediate scattering function closely matches the behaviour observed in SF_6 (Dove *et al* 1986).

The anomalous behaviour of the intermediate scattering function has been observed in two completely different simulations; one a simulation of point particles in two dimensions, the other a simulation of molecules in three dimensions. There is no reason to believe that this behaviour will not be

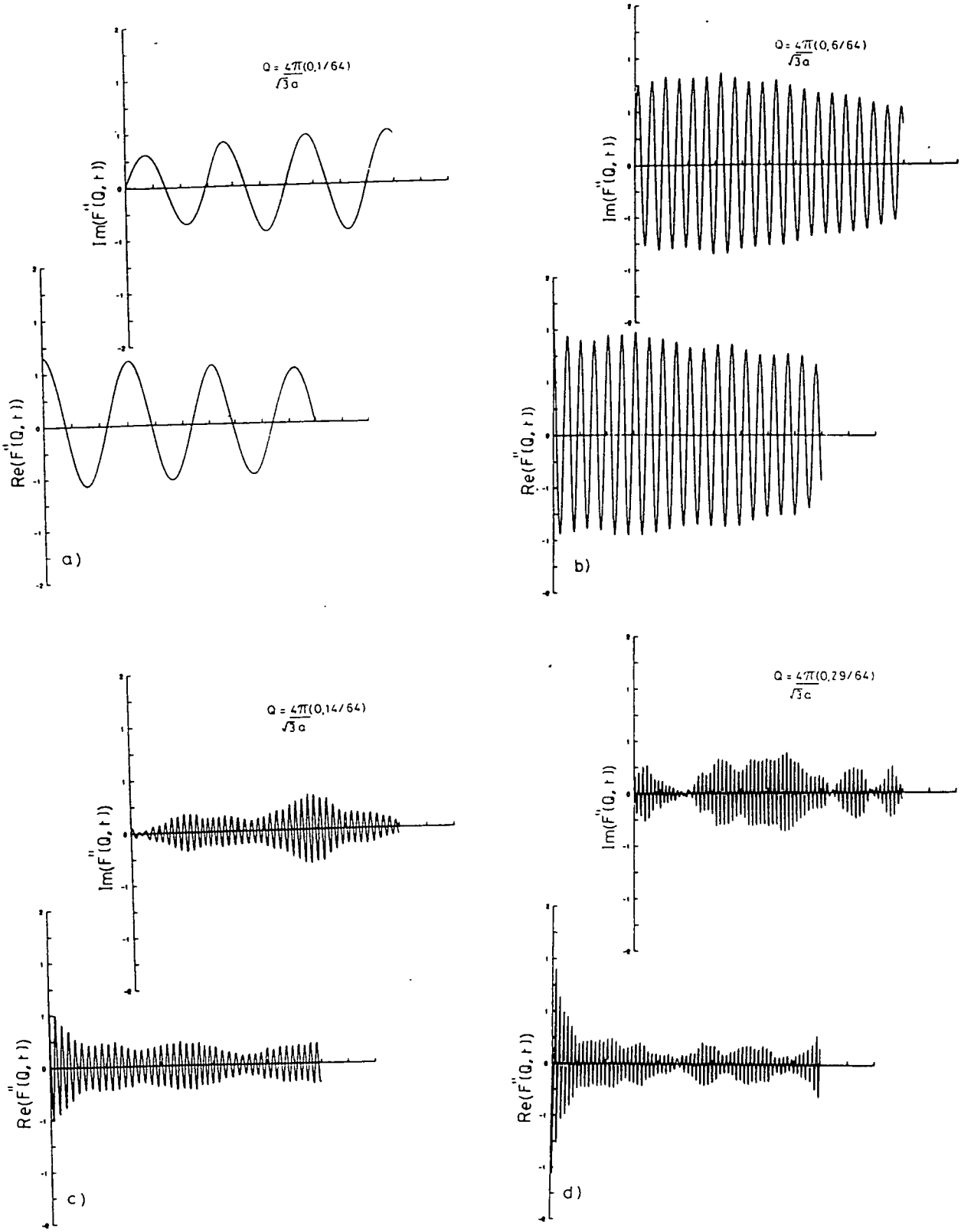
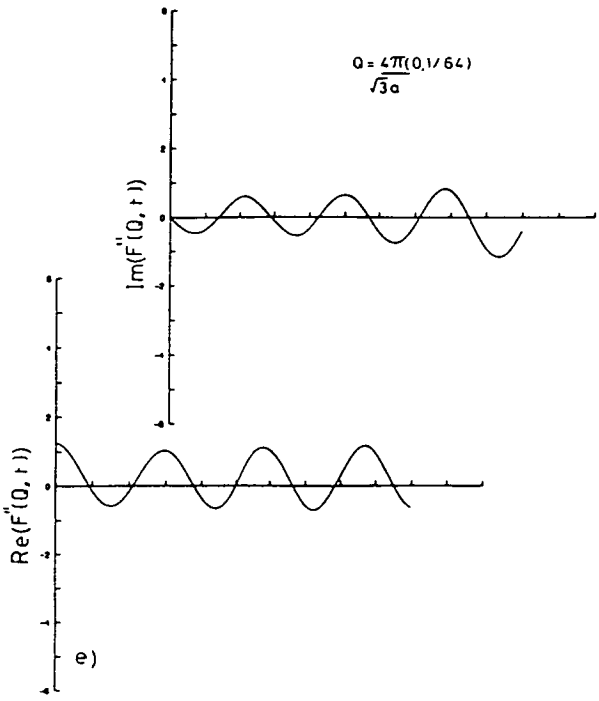
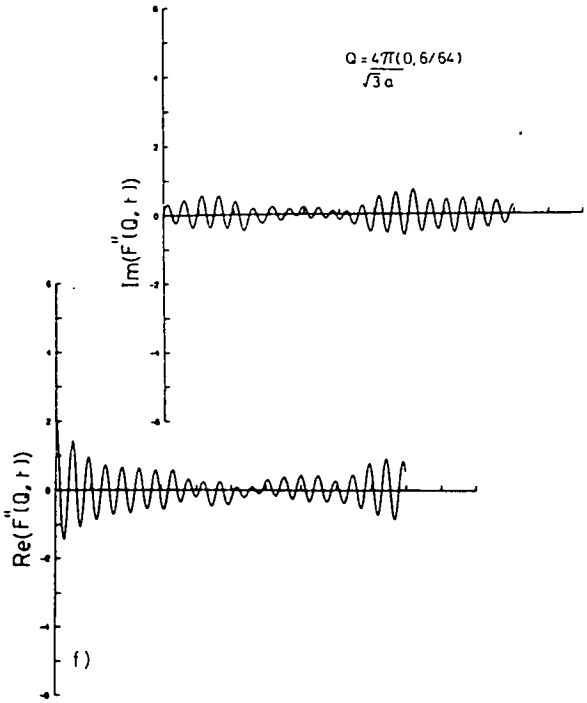
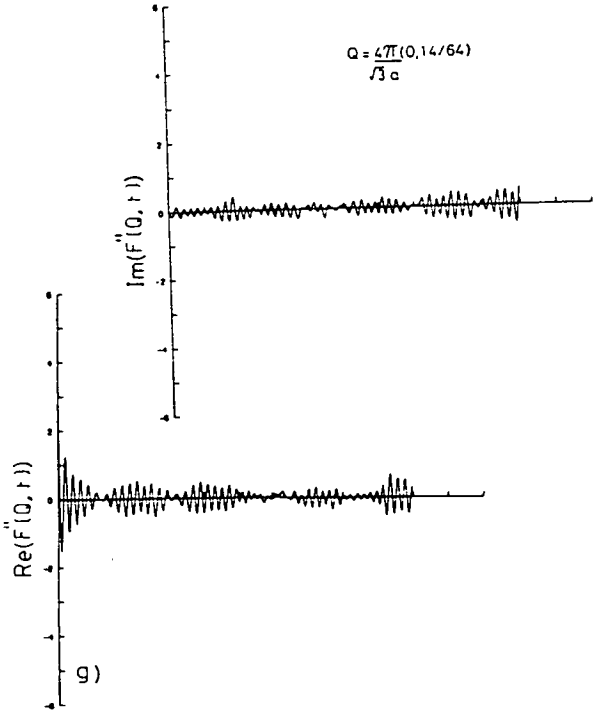
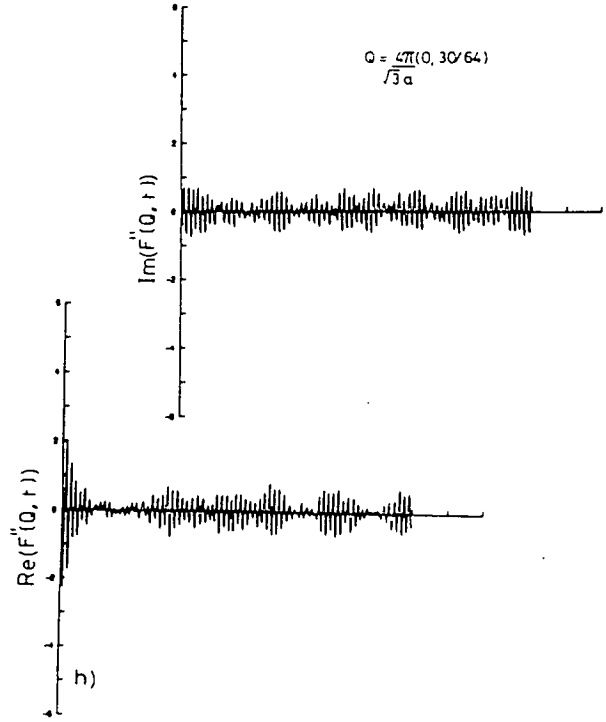


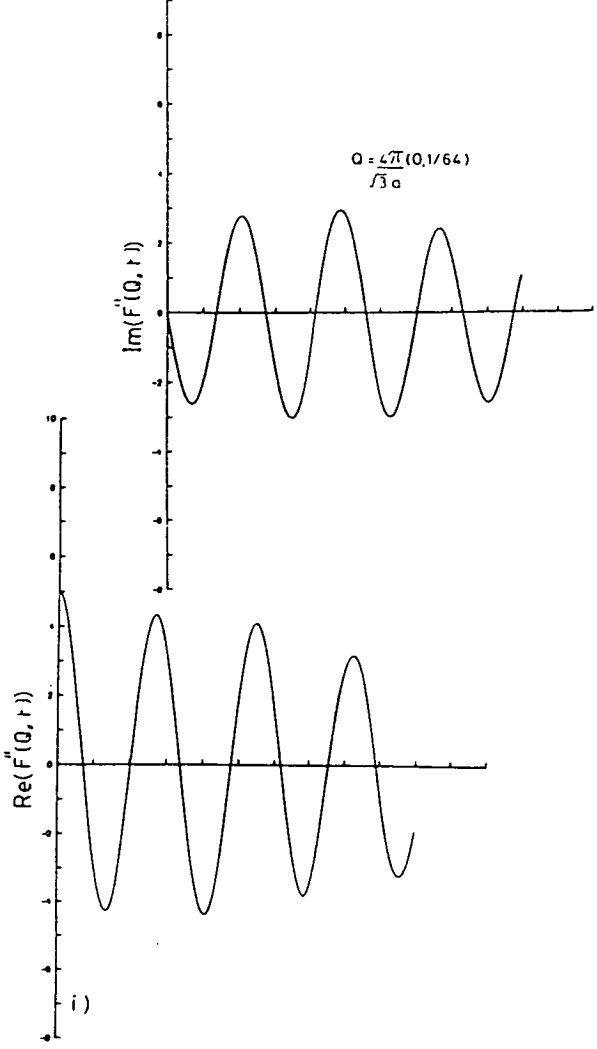
Figure 2.7 A selection of graphs of $F^{11}(Q,t)$ for a two-dimensional system of particles interacting via a Lennard-Jones potential at $\rho^* = .42$. The simulation was equilibrated for 8000 timesteps before the data for the intermediate scattering function was collected. The data for the scattering function was then collected over the next 4000 timesteps. The x-axis of each graph represents the time variable, each division corresponding to 2 ps. The top graph of each pair shows the imaginary component of $F^{11}(Q,t)$ and the bottom graph shows the real component. Figures 2.7 a-d are for a simulation at $T^* = .085$, graphs 2.7 e-h are for a simulation at $T^* = .298$ and graphs i-l are for a simulation at $T^* = .596$. The Q-vectors are all in the (0,1) direction and described in terms of reduced wave-vectors. All the graphs are for $F^{11}(Q,t)$ with t ranging from 0 to 20ps.

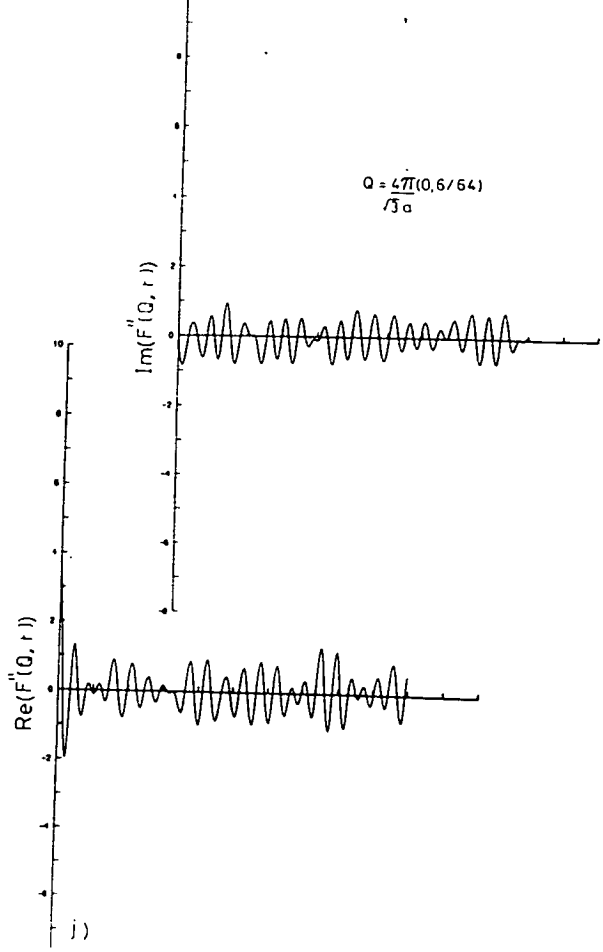

Q = 4π(0,1/64)
√3a
Im(F''(Q, t))
Re(F''(Q, t))
e)

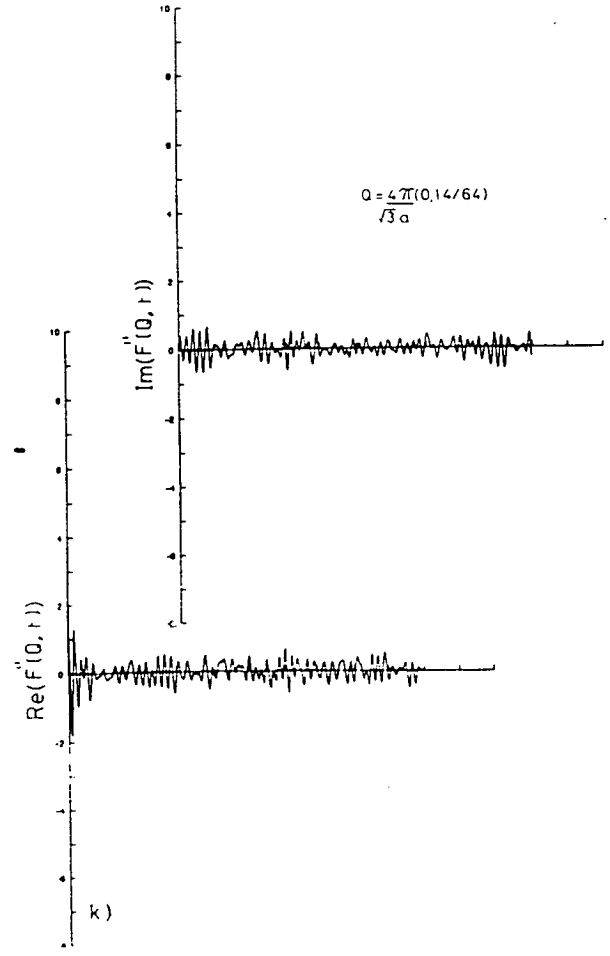

Q = 4π(0,6/64)
√3a
Im(F''(Q, t))
Re(F''(Q, t))
f)

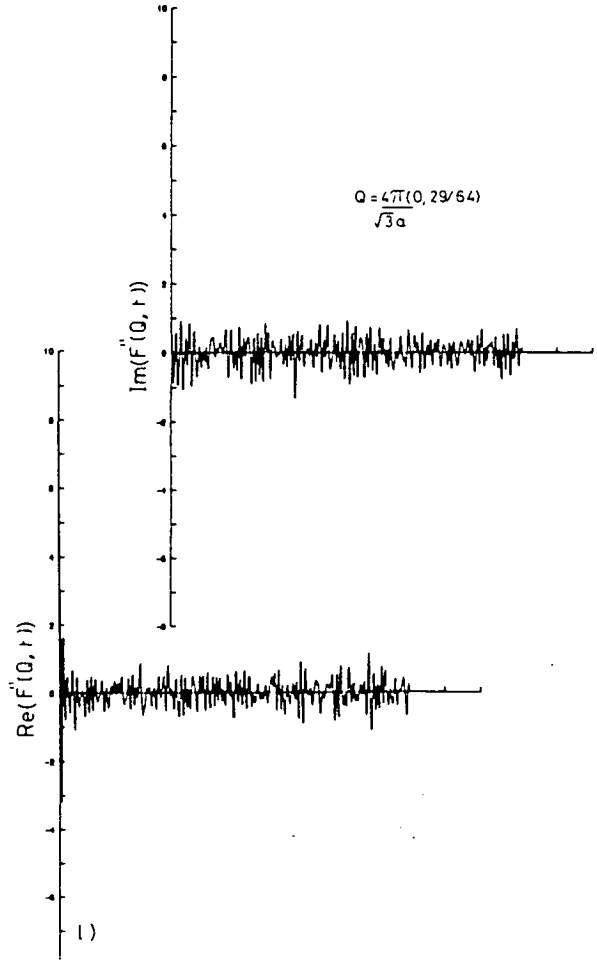

Q = 4π(0,14/64)
√3a
Im(F''(Q, t))
Re(F''(Q, t))
g)


Q = 4π(0,30/64)
√3a
Im(F''(Q, t))
Re(F''(Q, t))
h)


Q = 4π(0,1/64)
√3 a
Im(F''(Q, t))
Re(F''(Q, t))
i)


Q = 4π(0,6/64)
√3 a
Im(F''(Q, t))
Re(F''(Q, t))
j)


Q = 4π(0,14/64)
√3 a
Im(F''(Q, t))
Re(F''(Q, t))
k)


Q = 4π(0, 29/64)
√3 a
Im(F''(Q, t))
Re(F''(Q, t))
l)

observed in many other simulations and it is therefore important that we understand the origin of this behaviour to see whether it is a sign that something is going wrong with the MD algorithm or whether it is an effect we can ignore.

2.3.4.1. The Origin of the Imaginary Component of $F^{11}(Q,t)$

The first question that must be asked in looking for the origin the the imaginary component of $F(Q,t)$ is whether the simulation is well behaved, is it an ergodic simulation of a system in the micro-canonical ensemble? This can be tested using the results of section 2.1.3.1. Calcuating the equilibrium value of $\varepsilon(t)$ for the two-dimensional system we obtain a value for $\varepsilon(t)$ of 4.1 ± 0.2. Were the simulation non-ergodic, as could well be the case in two dimensions for a system of harmonic oscillators, $\varepsilon(t)$ would have an equilibrium value of 8 and not 4. The fact that the equilibrium value of $\varepsilon(t)$ equals 4 means that the simulation is indeed ergodic and micro-canonical.

If a mode in a system only interacts weakly other modes in the system then this mode will not equilibrate properly, it will carry on oscillating with the amplitudes it had right at the very start of the simulation. We will no longer be entitled to assume that $a_q=a_{-q}$. Consider the implications of this inequality in equation (2.79). If we cannot assume that $a_q=a_{-q}$ then the one-phonon density, $\rho^1(q,t)$, can be expressed in the harmonic approximation as:

$$\rho^1(Q,t)= (1/2\sqrt{N}) \sum_{q} a_q(E_q\cdot Q) \times$$
$$\sum_{i} \left(\exp(i[Q-q].R_i)\exp(i[\omega_q t+\phi_q])+\exp(i[Q+q].R_i)\exp(-i[\omega_{-q} t+\phi_{-q}])\right)$$

Therefore the expression for $F^{11}(Q,t)$ in the harmonic approximation becomes:

$$F^{11}(Q,t) = <\rho^1(q,t)\rho^1(-q,0)> =$$
$$< (N/4) \sum_{q} \sum_{G} (E_q\cdot Q)^2 \delta(Q-q-G) \times$$
$$[a_q\exp[i(\omega_q t+\phi_q)]+a_{-q}\exp[-i(\omega_{-q} t+\phi_{-q})]] \times$$
$$[\ a_{-q}\exp[i(\omega_{-q} 0+\phi_{-q})]+a_q\exp[-i(\omega_q 0+\phi_q)]\ ] > \qquad (2.91)$$

which can be rewritten as:

$$F^{11}(\mathbf{Q},t) = < (N/4) \sum_{i} \sum_{\omega} (E_q.\mathbf{Q})^2 \times$$
$$[\ a_q^2 \exp[i\omega_q t] + a_{-q}^2 \exp[-i\omega_q t]\ ] +$$
$$a_q a_{-q} [\exp[i(\omega_q t + \phi_q + \phi_{-q})] + \exp[-i(\omega_q t + \phi_q + \phi_{-q})]\ ] > \qquad (2.92)$$

where it is assumed that $\omega_q = \omega_{-q}$. If we also assume that the configurational averaging step will have the effect of setting the terms with the random phase-factors to 0 (which need not be true in a non-ergodic system) then we finally obtain:

$$F^{11}(\mathbf{Q},t) = < (N/4) \sum_{i} \sum_{\omega} (E_q.\mathbf{Q})^2 \times$$
$$[\ \cos(\omega_q t)[a_q^2 + a_{-q}^2] + i\sin(\omega_q t)[a_q^2 - a_{-q}^2]\ ] > \qquad (2.93)$$

Examining this equation we see that $F^{11}(\mathbf{Q},t)$ can have an imaginary oscillating part when a_q is not equal to a_{-q}.

This theory can be directly tested in the simulation. If we set up the simulation so that it contains standing wave at $\mathbf{Q}=\mathbf{Q}'$ we can then examine $F^{11}(\mathbf{Q}',t)$. As we have a standing wave with wave-vector $\mathbf{Q}'$, $a_{Q'}$ will be equal to $a_{-Q'}$. If the above analysis is valid then $F^{11}(\mathbf{Q}',t)$ should be a real function. Figure 2.8 shows $F^{11}(\mathbf{Q},t)$ calculated for a two different simulations, one in which a standing wave was set up at $\mathbf{Q}=[4\pi/(\sqrt{3}a)](0,1/64)$ and the other in which a standing wave was set up with a $\mathbf{Q}$-vector of $[4\pi/(\sqrt{3}a)](0,2/64)$. Examining the values of $F^{11}(\mathbf{Q},t)$ at $\mathbf{Q}$-vectors which correspond to the standing waves then we can see that the magnitude of the imaginary component of the scattering function is indeed totally negligible when a_Q is equal to a_{-Q}.

So far we have only looked at the one-phonon terms. The situation becomes a lot more complicated if we look at the multi-phonon terms. The intermediate scattering function for the multi-phonon case (see equations (2.69) and (2.75)) contains terms such as $F^{nm}(\mathbf{Q},t)$. If n+m is odd then it is fairly straightforward to show that the configurational average acting on the random phase-factors ensures that $F^{nm}(\mathbf{Q},t)$ is zero. However if the configurational average is

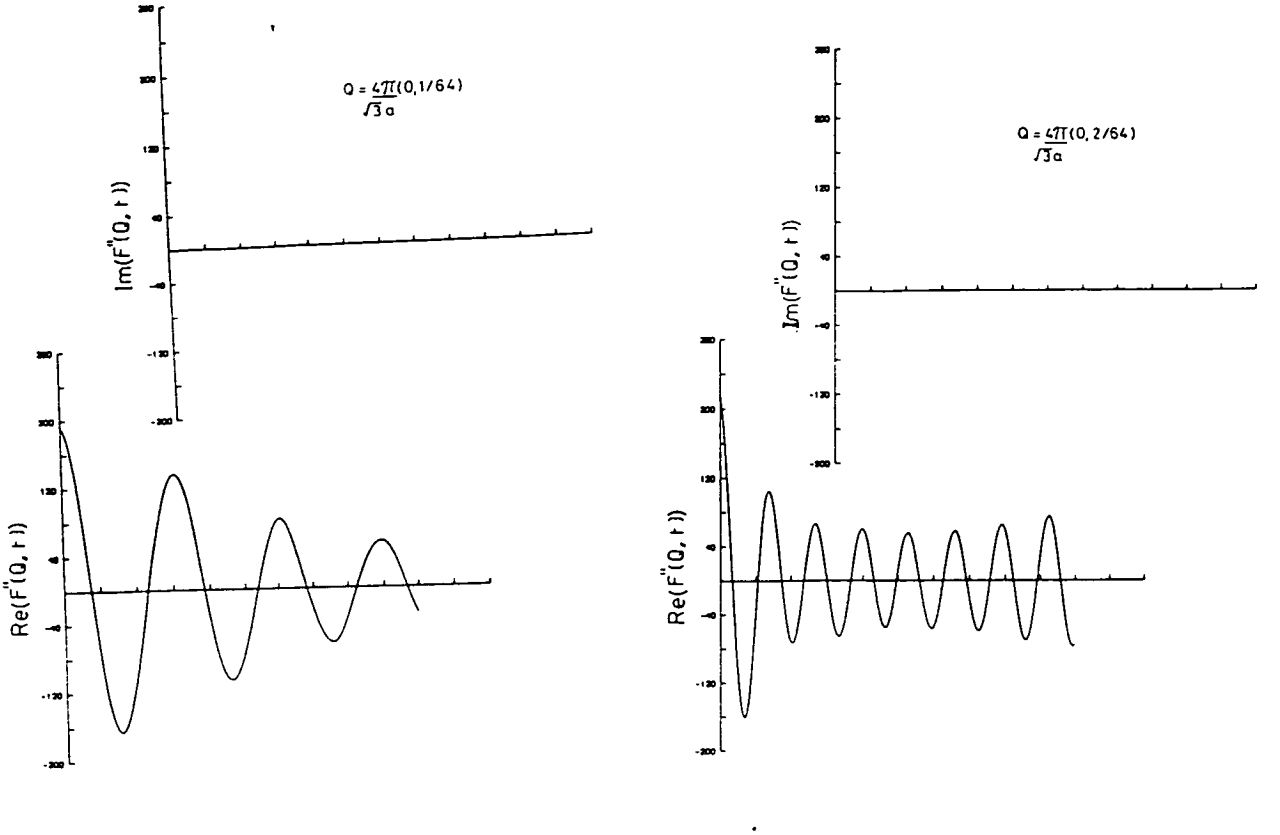


Figure 2.8 The real and imaginary components of $F^{11}(Q,t)$ corresponding to Q-vectors at which we have set standing waves. In both cases the imaginary component is negligable when compared with the real component. These graphs should be compared with $F^{11}(Q,t)$ calculated at similar Q-vectors but for which we did not have standing waves (Figure 2.7).

replaced by a time average, as must happen in an MD simulation, then the situation becomes a lot less straightforward. Even if we assume that a_q is equal to a_{-q} it is possible to get an imaginary component of $F^{nm}(Q,t)$ which oscillates at a frequency close to ω_Q and which only decays very slowly. We would not expect the multi-phonon intermediate scattering function to be a real function when calculated via an MD simulation, however it is difficult to calculate the magnitude of these imaginary oscillations theoretically and so we cannot predict if these contributions will appear as anything other than noise.

2.3.5. The Effect of the Boundary Conditions on $F^{11}(Q,t)$

Figure 2.9 shows the real part of $F^{11}(Q,t)$ calculated for the configuration at $\rho^*=0.92, T^*=0.085$ for several different **q**-vectors in the Brillouin zone. It can be seen for these figures that the amplitude of $F^{11}(Q,t)$ decays for a few picoseconds before increasing again and appearing to 'beat' with a period of several picoseconds.

In the real system we would expect $F^{11}(Q,t)$ to decay to zero in a few picoseconds, the time taken to decay to zero being related to the phonon lifetime (which can be measured experimentally from the width of peak corresponding to the appropriate phonon in a neutron scattering experiment). However in the MD simulation the intermediate scattering function does not decay to zero, even after 20ps. For some reason the density fluctuations in the simulated system remain correlated for much longer than they do in the physical system.

It has been known for some time that simulations using periodic boundary conditions are subject to spurious correlations (Vesely 1978, Berne 1971, Hansen 1976) related to the 'recurrence time', τ_c of the system:

$$\tau_c = L/v_s \tag{2.94}$$

where L is the length of one of the sides of the MD cell and v_s is the speed of sound in the simulation. If a disturbance happens at a point **x** at a time t then it will propagate away from **x** with approximately the speed of sound to return to **x** at time $t+\tau_c$. This 'echo' of the original signal means that there will be a

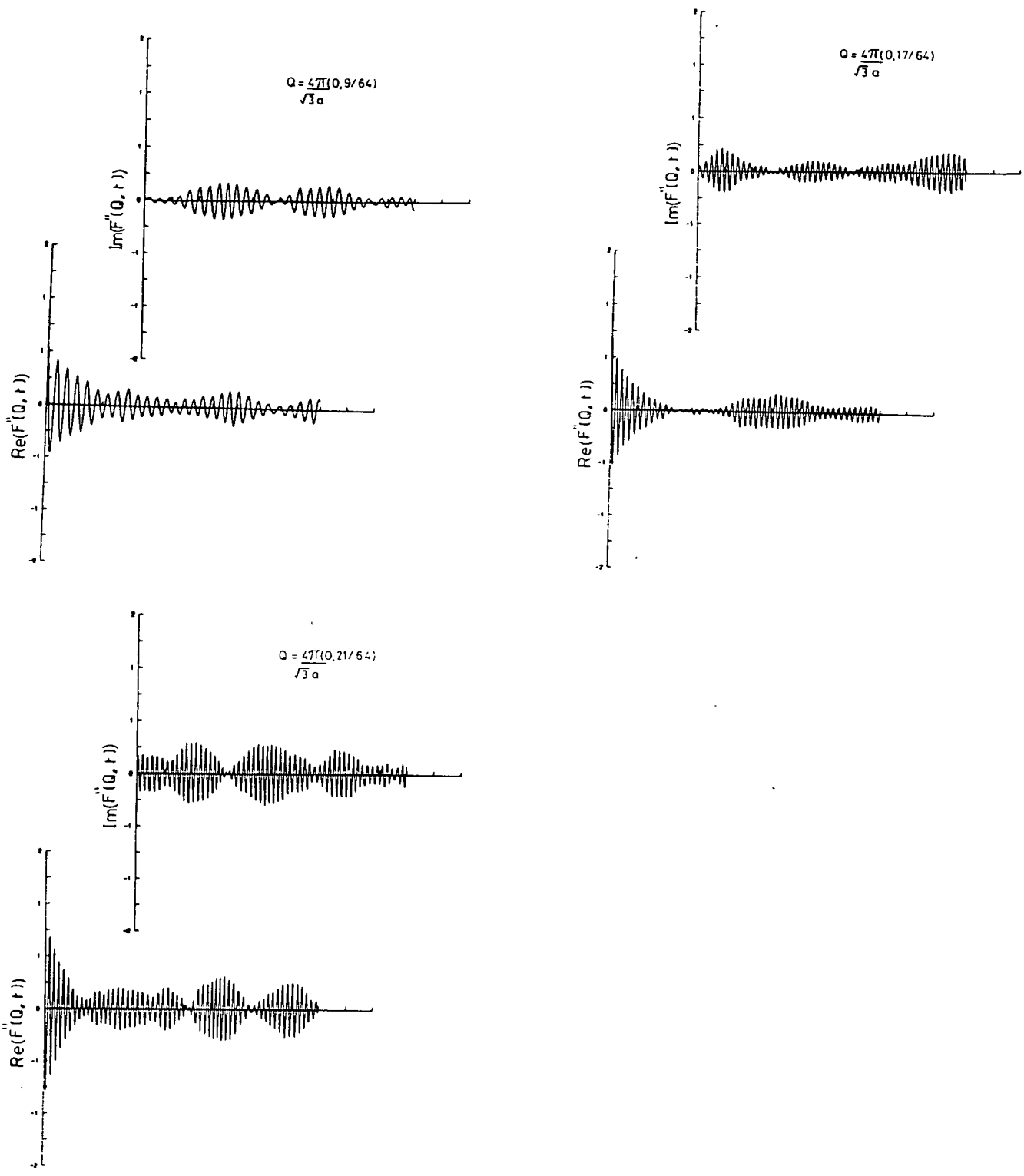


Figure 2.9 A selection of $F^{11}(Q,t)$ chosen to demonstrate the effects caused by the use of periodic boundary conditions on a finite system. The scattering functions were collected from a simulation which had a recurrence time of approximately 7ps.

correlation between events at **x** at time t and events at **x** at time $t+n\tau_c$ as the sound waves cross and recross the MD cell.

This spurious correlation can be seen in the intermediate scattering function calculated from the MD simulation. If we calculate the speed of sound by measuring the gradient of the dispersion curve at a q-vector corresponding to the lowest allowed q-vector then we can calculate τ_c for the MD simulation. For the simulation for which the graphs in figure 2.9 are taken τ_c is approximately 7ps. Examining figure 2.9 we can see that the observed 'beat' period corresponds fairly well to the recurrence time. The 'beats' observed in the intermediate scattering function are a finite size effect caused by using periodic boundary conditions on a finite system.

Although these finite size effects do not affect the frequency of the oscillations in the intermediate scattering function they will affect the dynamical scattering functions calculated by Fourier transforming $F^{11}(Q,t)$. In particular the widths of the peaks, and hence the calculated phonon lifetimes, will be too large in the simulation even though the centre of the peak will be at the correct frequency. In order that phonon phonon lifetimes are not overestimated $F^{11}(Q,t)$ data should not be used for $t>\tau_c$, indeed the values of any time correlation functions cannot be trusted for values of t greater than τ-(c).

2.3.6. Conclusions

In this section the calculation of scattering functions from an MD simulation has been discussed. In most cases these techniques work well and it is a straightforward procedure to calculate scattering cross-sections which are almost identical to those determined from neutron or X-ray scattering experiments. There are however two problems that can arise in these calculations.

The first concerns the calculation of scattering functions for slowly decaying, i.e. under-damped modes. For these modes we must be aware of the problems that can arise when the time taken for a signal to propagate all the way round the MD cell is similar to the time taken for the mode itself to decay. In these circumstances the system can suffer from a feedback effect which tends to make the lifetime of the under-damped modes too large. We should neglect

values of time correlation functions which are greater than the τ_c, such correlations are simply a function of the periodic boundary conditions. We should be particularly aware of these problems when dealing with zone-centre modes which are usually have the least damping and the longest periods of any of the modes in the Brillouin zone.

The second problem again mainly concerns the harmonic zone-centre modes. If these modes are not properly equilibrated then it is possible for their intermediate scattering functions to have a significant imaginary component. For modes closer to the zone boundary the maximum amplitude of this imaginary component is a lot smaller than the largest amplitude of the real component, the size of the imaginary component being similar to the size of the real component generated by the finite size effects described in the previous paragraph. For the zone-boundary modes we can probably neglect the effect of the imaginary component and simply use the real part of $F^{11}(Q,t)$ for $t<t_{rec}$ to calculate the dynamical scattering functions. Recent results from quantum field theory computer simulations (Duane and Kogut 1985) have suggested a way in which harmonic modes can be precisely equilibrated in MD simulations. If, during the equilibration period of the simulation, about 2% of the MD time integration steps are replaced by Langevin time integration steps (equivalent to throwing away all the particle velocities and replacing them with values chosen from an appropriate Gaussian distribution) then it can be shown analytically that the harmonic modes will equilibrate properly. It would be interesting to see whether such a 'hybrid' integration scheme could remove the problem of imaginary scattering functions in MD simulations.

CHAPTER 3

THE MD SIMULATION PROGRAM

This chapter describes the application of the molecular dynamics algorithm to the simulation of a fluorite structure crystal. The specific example given will be a simulation of strontium chloride ($SrCl_2$), though everything discussed in this chapter will also be applicable to the simulation of other simple fluorite structure crystals such as calcium fluoride (CaF_2) and lead fluoride (PbF_2).

3.1. Setting up the Simulation

The first question that has to be considered before designing any of the computer code is how large should the simulation be. Both the choice of algorithm used to calculate the Coulombic forces and the computational cost of the simulation depend critically on the number of ions being simulated. If the simulation is of only a few hundred particles then it might be possible to calculate the Coulombic forces directly as a sum over all pairs of particles. For larger systems (<= 600 particles) techniques such as Ewald Summation can be used to calculate the Coulombic forces (Sangster and Dixon 1976, Ewald 1921). For systems of more than 600 ions the Particle-Particle/Particle-Mesh (PPPM) algorithm (Hockney and Eastwood 1981) becomes computationally more efficient than the Ewald method. Most simulations of Superionic conductors reported in the literature have been on systems of between 200 and 600 ions (for example, Gillan and Dixon 1979, Walker *et al* 1982, Moscinski and Jacobs 1985). However there are several disadvantages to simulations of this size.

Firstly, for simulations of small systems the number of allowed q-vectors is very limited. Because the allowed q-vectors are so far apart it is difficult to get good resolution of features in reciprocal space. A lot of interest has been generated recently by the discovery of a peak in the coherent quasi-elastic neutron scattering spectrum of CaF_2 around the (2,0,0) position in reciprocal space (Dickens *et al* 1978, 1979). This peak has been reproduced in an MD simulation (Gillan 1980) but because the simulation was on a small system (324 ions), the allowed q-vectors were too far apart to give good resolution of

the peak. Were one to simulate a larger system one would be able to see this peak in more detail and so get a better understanding of the diffusion mechanisms in superionic conductors (see the next chapter).

Secondly there has been some debate in the literature about the existence or non-existence of interacting defects or clusters of defects in superionic conductors. For example Hutchings *et al* (1984) found that the defect cluster which best fitted their scattering data contained 9 vacancies, one interstitial and 8 relaxed ions (see fig 1.8). It would be difficult to simulate clusters such as these in a system of only a few hundred ions. Although one of the clusters could be fitted into the computational box, the effect of the periodic boundary conditions would be to place these defects close together. This would result in a larger defect density than the expected equilibrium density (Moscinski and Jacobs 1985). It would be unlikely that such clusters would form in a small simulation, even if such defect clusters occurred in the real system. By simulating a larger system it is possible to increase the distance between such clusters and hence make it possible for them to form.

Problems can also arise when trying to calculate long range forces on a small periodic lattice. Because the Coulombic potential is so long ranged several thousand of these image charges must be included every time the forces between two charges in the computational box are calculated. Consider a periodic LxL box. If two ions are positioned along some symmetry direction of the box, e.g. along a box diagonal, then the only effect of the image charges is to reduce the size of the force between the ions, not the force direction. However when the ions are not symmetrically placed with respect to the box the effects can be dramatic. Consider placing the two particles in the box such that the line joining their centres makes an angle of 30° with the x-axis of the box. If we calculate the angle between the x-axis and the force vector on one of the atoms for various particle separations expressed as a fraction of D (= (√3/4)L) we obtain the results shown in figure 3.1.

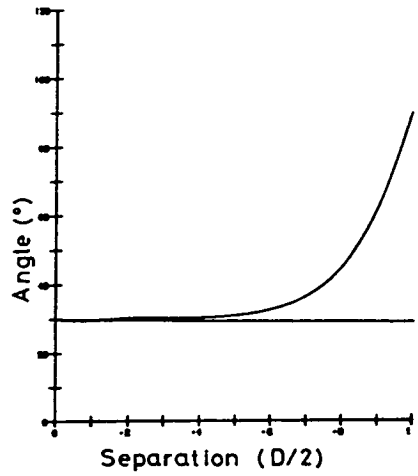


Figure 3.1 The angle that the force between two particles at 30° to the x-axis makes with the x-axis as a function of the particle separation.

It can be seen from the graph that for particle separations of less than 0.7D the force vectors are almost colinear with the line joining the particle centres, however for particle separations greater than 0.7D the two directions diverge rapidly. At first sight it would seem that any simulation of a Coulombic system in a periodic box will suffer from finite-size effects, however for large enough systems this need not be the case.

In real crystals one can use the Thomas-Fermi technique to estimate the effect of screening. In this theory the effective electric potential has the form $(Q/r)\exp(-k_0 r)$, i.e. the Coulombic potential multiplied by an exponential damping term, where k_0 is proportional to the electron density and the inverse of the lattice constant. Typical screening lengths (= $1/k_0$) are of the order of one or two lattice spacings. It should be possible to neglect the Coulombic interactions for particles whose separation is greater than some effective screening length. If the simulation is large enough such that for particles separated by the screening length the forces are still along the lines joining the particles then the finite size effects caused by the periodic boundary conditions should be negligible. Assuming that the screening length is approximately twice the lattice constant, a, then the condition that the simulation must satisfy so that it will not suffer from finite-size effects is that $L>6.6a$, i.e the MD cell must contain at least 7x7x7 unit cells.

Another advantage of simulating large systems is the improved statistics obtained from any data analysis. On a system of 3630 ions the effect of noise in any measurements will be reduced by a factor of $\sqrt{10}$ when compared to a system of 324 ions. Similarly it is much easier to spot rare events in a large system.

For these reasons it would seem sensible to try to simulate the largest system that is economically possible. As the system will contain several thousand ions the PPPM algorithm must be used to calculate the long range ionic forces. The PPPM algorithm places a mesh on the computational box, assigns the charges to the mesh points and then solves Poisson's equation on the mesh to obtain the Coulombic potential at every mesh point. The largest mesh size that can be handled conveniently on the DAP is a mesh of 32^3 points. It is this constraint that gives us the greatest number of ions that can be simulated. The PPPM algorithm has one tuneable parameter, a, the ratio of the cut-off radius of the short range force (r_e) to the mesh spacing h. For the PPPM algorithm to be efficient a should be approximately 3. In order to cut down the number of short range interactions r_e needs to be approximately equal to d, the spacing between two strontiums. Let n be the number of strontiums placed along one edge of the computational box. Using $h = nd/32$, we compute that n = 10 or 11. This means that we can simulate a maximum of 11x11x11 primitive unit cells of $SrCl_2$. As $SrCl_2$ has a face centred cubic structure then there must be an even number of primitive unit cells in the z direction to satisfy the periodic boundary conditions (see fig. 3.2). The system chosen for the simulation is therefore one of 11x11x10 primitive unit cells, or 3630 ions.

The major part of any MD simulation program is the calculation of the interparticle forces. Compared with these calculations the rest of the MD algorithm, updating coordinates, calculating accelerations etc, is straightforward. In the program the calculation of the inter-particle forces is split into two parts. We first evaluate the Van der Waals forces and the electron shell repulsion forces and then the contribution from the Coulombic forces. The calculation of the short range forces is discussed in section 3.3. The calculation of the long range forces is discussed in section 3.4.

In setting up a simulation program for $SrCl_2$ on the DAP we are faced with a problem that does not cause any difficulty for a simulation on a serial or

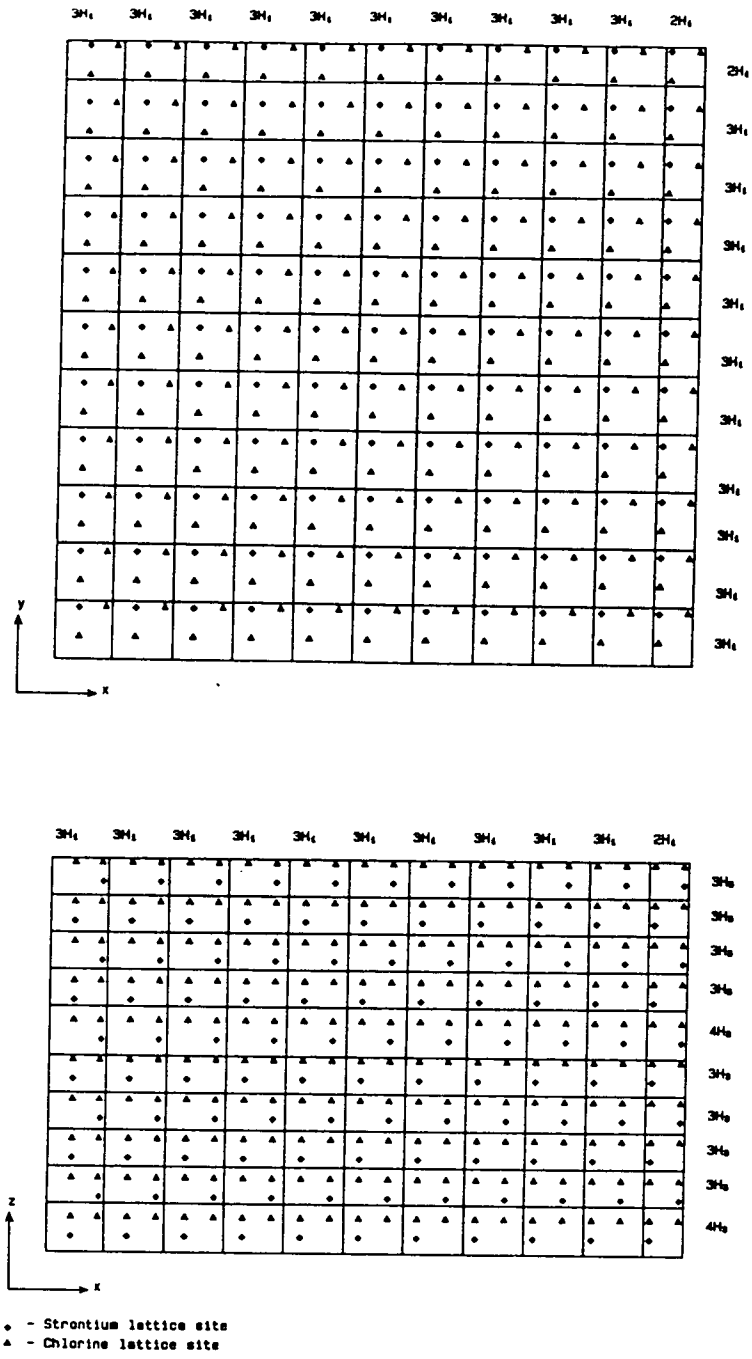


Figure 3.2 The division of the computational box into 11x11x10 Particle-Particle cells each containing two chlorine lattice sites and one strontium lattice site.

vector processor but which does cause difficulties on a parallel processor.

3.2. The Mobile Ion Problem

In $SrCl_2$ at least some of the anions can hop between different lattice sites. This property makes simulating superionics on a parallel processor more complicated than simulating crystals without diffusing ions. On a parallel processor the links between PEs are fixed. A PE on the DAP will always be connected to the same four nearest neighbour PEs. In an MD simulation of a solid on the DAP particles are mapped directly onto PEs, the coordinates for one particle always being found on the same PE. This simple mapping of particles to PEs requires that particles keep the same nearest neighbours throughout the simulation. However when a chlorine ion hops from one anion lattice site to another it completely changes all its nearest neighbours. Another way must be found for mapping the chlorines onto the DAP.

The problem cam be solved by mapping well defined volumes in the MD cell, as opposed to particles, onto the PEs. All the coordinates of a chlorines in a volume of the crystal controlled by a PE will be stored on that PE. This mapping then solves the problem of changing neighbour lists, different parts of the computational box will always maintain their same relative spatial positions. The MD cell is therefore divided into 11x11x10 different cells. For simplicity each of these cells is made orthorhombic. We would also like each cell to contain, on average, one strontium and two chlorines. For technical reasons each of the box sides must be a multiple of the PPPM mesh spacings. Figure 3.2 shows how this division of the computational box into cells has been achieved. Note that the mesh spacing in the z direction is $10/(11\sqrt{2})$ the mesh spacing in the x and y directions. Each of these cells is mapped onto 3 PEs, one for all the x coordinates, one for the y coordinates and one for the z coordinates (See Fig. 3.3), this mapping of particle coordinate data onto PEs will be referred to as the particle-particle data-representation (see section 3.5.1) By mapping all the x, y and z coordinates onto the one plane it is possible to use 3630 out of the 4096 available PEs.

Because the strontiums do not diffuse it is possible to store all the data for a given strontium on the PE which controls the cell containing the strontium

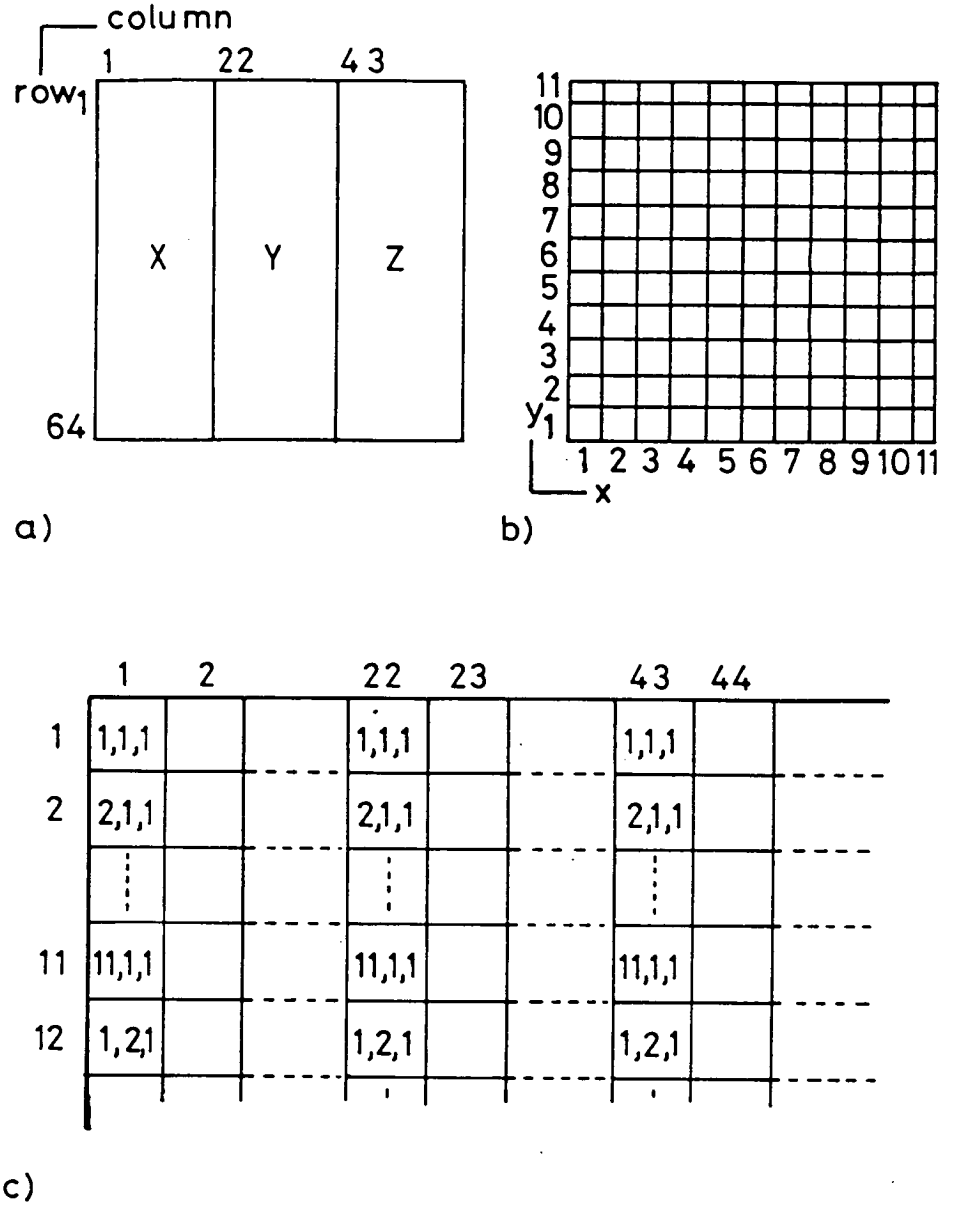


Figure 3.3 The mapping of the particle data from the PP data-representation onto the 64x64 array of PEs on the DAP. a) How the x, y and z coordinates from an 11x11x10 array of PP cells are mapped onto a DAP data-plane. The x-coordinates are stored in columns 1-21 of the array, the y-coordinates are stored in columns 22-42 and the z-coordinates are stored in columns 43-63. b) The first z-plane of the PP cells. c) Where the data from this PP plane finishes up on 64x64 grid of PEs, the labels shown referring to the PP cell coordinates. Note that x, y and z data from a given PP cell is stored in equivalent locations 21 columns apart.

lattice site, this mapping will stay constant throughout the simulation. Although each cell will contain on average 2 chlorines it can contain anything from 0 to 4 chlorines (the electrostatic repulsion between the chlorines should prevent any more than 4 ions being in the same cell simultaneously). We therefore allow room on each PE to store the coordinates of up to 4 chlorine ions. The four chlorine data planes are numbered from 1 to 4. The chlorine data is stored on these data planes in such a way that if there is only one chlorine in a cell then its coordinates will be stored on plane 1, if there are two chlorines their coordinates will be stored on data planes one and two etc. By storing the data in this way we insure that data plane four will be unoccupied if at all possible.

At the end of every coordinate updating step in the integration algorithm we check to see if any of the chlorines in the fourth data plane have x coordinates that are out of range of their present cell. If any are out of range we then examine the fourth data plane of the neighbouring cell in the appropriate x direction. If this plane is empty at the new site we then calculate the position of the chlorine with respect to the origin of its new cell and move all of its coordinates onto the fourth chlorine data plane at the PEs controlling the new cell. We move the chlorine data on the data planes so that the chlorine data is always stored on the lowest possible data planes. We then repeat the process with chlorines from the third data plane, then the second data plane and finally from the first data plane. At every stage we ensure that the chlorines are on the lowest possible data planes. The whole procedure is repeated with the y coordinates of the chlorines and finally the z coordinates. If a particle finds it impossible to move to a new cell because that cell already contains four chlorines there are no errors introduced; the program still calculates the Coulombic forces correctly as long as the particle data is stored in a PP cell which is a nearest neighbour of the correct cell. In test runs on $SrCl_2$ above the superionic transition temperature it was found that on average 48% of the chlorine ions were on the first data plane, 34% were on the second data plane, 13% were on the third data plane and 5% were on the fourth data plane.

We find that there is a price to be paid for treating the mobile ions in this way. The number of short range interactions that must be calculated between two neighbouring cells goes as the square of the number of ions in the cells.

In calculating the short range forces we have to assume that each cell contains 5 ions (one strontium and four chlorines) as opposed to 3 ions (one strontium and two chlorines). This means that the calculation of the short range forces takes 25/9 times as long as it would if the chlorines were not diffusing.

3.3. Potentials and the Short Range Force Calculation

3.3.1. The Model Potentials

Realistic simulations of ionic compounds rely on good interionic potentials which can reproduce important properties of the real crystal in the simulation. The interaction potentials are taken to have the well established Born-Mayer-Huggins form (Born and Mayer 1932, Huggins and Mayer 1933, Mayer 1933). The potential between two ions of species α and β is taken to be:

$$V_{\alpha\beta}(r) = z_\alpha z_\beta e^2/r + A_{\alpha\beta}\exp(-r/\rho_{\alpha\beta}) - C_{\alpha\beta}/r^6 \qquad (3.1)$$

where e is the charge on the electron and z_α and z_β are the charges on ions α and β in units of |e|. The first term in this expression represents the Coulombic interaction, the second term the repulsive interaction from the overlap of electron clouds and the third term the Van der Waals interaction.

This form of the potential neglects an important term, the interactions caused by the electronic polarisability of the ions. Molecular dynamics simulations of alkali halides that include the effect of polarisabilities through the shell model (Dick and Overhauser 1958) have been performed (Sangster and Dixon 1976). These simulations suggested that the inclusion of polarisabilities did not affect the spatial distribution of the ions or qualitatively affect the dynamics. These simulations also showed that including polarisabilities is computationally very expensive; it is mainly for this reason that they have been neglected in these simulations. The only justification for this simplification can be if the rigid ion potentials used can reproduce most of the important features of the superionic systems.

This simplification will inevitably lead to inaccuracies. The effect of polarisabilities is known to be important in determining defect energies (Hardy and Flocken 1970, Faux and Lidiard 1971). Similarly the high frequency dielectric constant, ε_∞, must be equal to 1 in a rigid ion model whereas in CaF_2 ε_∞ has been measured to be 2.05 (Lowndes 1969). The inaccuracy of ε_∞ will lead to inaccuracies in the phonon dispersion curves, particularly in the optic modes. The Lyddane-Sachs-Teller relationship relates the ratio between the longitudinal and transverse optical modes at the zone centre to the ratio between the high frequency and low frequency dielectric constants, ε_∞ and ε_0. We therefore know that if the value of ε_∞ differs from the experimental value then the ratio between the two optic modes at the zone centre in the simulation will be different from that obtained experimentally. It can be shown via lattice dynamics techniques that the effect of non-rigid-ion type behaviour is to lower the frequency of the LO branch of the phonon dispersion curve (Hardy and Karo 1979). However the effect of this discrepancy in the frequencies of the LO mode on the equation of state of the system will be small as the LO branch is only weakly volume dependent. This means that quantities derivable from the equation of state, such as the transition temperature, will be similar both for systems with rigid ion potentials and systems which include polarisability (Boyer 1981). Also the time period of the LO mode is several orders of magnitude smaller than the time periods of many of the phenomena of interest in superionics, such as the hopping time. This suggests that the two processes will not be strongly coupled. Although we must recognise these limitations in the rigid ion model, particularly when using the rigid-ion potentials to describe the highly polarisable lead cation, the model can still provide an adequate description of a superionic conductor (for a comparison between the rigid ion and shell model simulation results for CaF_2 see Dixon and Gillan 1980)

Since the fluorite structure superionics are very close to being fully ionic (Hodby 1974), we can set $z_+ = 2$ and $z_- = -1$, where the subscript + refers to the cations and the subscript - to the anions. The other potential parameters are determined empirically. The double charge on the cations ensures that they will always be well separated so that their Van der Waals and electron shell repulsion interactions will be negligible. We can therefore put $C_{++} = A_{++} = 0$. Similarly the anions and cations are sufficiently close together to ensure that their Van der Waals interactions are quenched; $C_{+-} = 0$ (Catlow

et al 1982). For $SrCl_2$ the chlorine-chlorine potentials are taken from the shell model potentials of Bendall (1976), which in turn were based on an electron gas calculation (Kim and Gordon 1974). For CaF_2 the fluorine-fluorine potentials used are those given by Catlow and Norgett (1973) obtained from a study of point defects in alkali fluorites. When these fluorine-fluorine potentials were tried in lead fluoride they were found to give an instability in the Δ_2' mode (Walker *et al* 1982). This instability was removed by further refining the Catlow and Norgett potentials. Once the anion-anion parameters have been fixed the remaining two cation-anion parameters, ρ_{+-} and A_{+-} must also be determined. For all the above systems these parameters were obtained by fitting them to the zero temperature lattice parameter and the dielectric constant ε_0 (For an example of these calculations see Axe 1965). They were also fitted to E_F, the energy of formation of a Frenkel defect, using the HADES lattice statics program (Norgett 1974, Catlow and Mackrodt 1982). It is important to get E_F correct as it has been shown that there is an empirical relationship between E_F and the transition temperature (March *et al* 1980). If we want to compare the simulation results with experiments then the simulation has to exhibit superionic behaviour in the right temperature regime. Table 3.1 shows the rigid ion parameters used. Table 3.2 gives a comparison between experimental values of various bulk and defect properties and those obtained from the rigid ion potentials. The overall agreement between experimental and calculated values seems to be reasonable. As a final check figure 3.4 shows the experimental and calculated phonon dispersion curves for CaF_2 (Gillan 1986a). Although, as expected, there are inaccuracies in the calculated optic modes, the overall agreement seems to be good. If the dispersion curves of the model are close to the dispersion curves of the real crystal then we can be fairly confident that the dynamics of the simulation and the real crystal are similar.

TABLE 3.1 *Parameters of the Short Range Potential used in the Simulation.*

	$SrCl_2$	PbF_2	CaF_2
A_{--} (eV)	1227.2	10225	1808
ρ (Å)$_{--}$	0.3214	0.225	0.293
C_{--} (eVÅ^6)	1.69	107.3	109.1
A_{+-} (eV)	774.14	122.7	674.3
ρ (Å)$_{+-}$	0.3894	0.516	0.336

TABLE 3.2 *Perfect crystal and anion defect quantities calculated from the potential model s and compared with the experimental values.* ε_0 *and* ε_∞ *are the static and high frequency dielectric constants,* c_{ij} *are the elastic constants, U is the cohesive energy per unit cell,* E_F *is the Fenkel defect formation energy,* ΔE_v *and* ΔE_i *the vacancy and interstitial migration energies.*

	$SrCl_2$		PbF_2		CaF_2	
	Calc.	Exp.	Calc.	Exp.	Calc.	Exp
ε_0	-		31.7	31.7	5.20	6.47
ε_∞	-		-		1.00	6.47
$c_{11}/10^{11}$ dyn/cm^2	6.63	7.55	5.56	9.3	15.8	17.12
c_{12}	1.44	1.72	1.44	4.4	4.15	4.68
c_{44}	1.44	1.03	1.26	2.06	3.96	3.62
U/eV	-	-	25.5	23.3	27.1	26.76
E_F	2.14	1.7	1.12	1.0	2.71	2.71
Δ_v/eV	0.1	0.35	0.08	0.18	0.19	0.42
Δ_i/eV	0.46	1.0	0.08	0.62	0.73	0.79

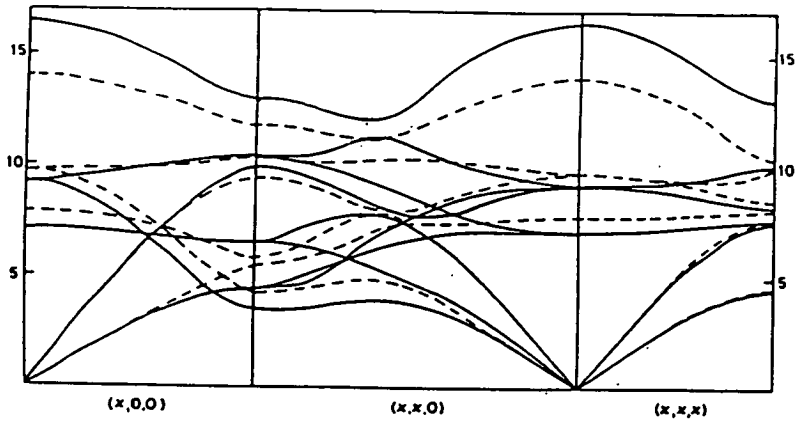


Figure 3.4 The solid line show the harmonic phonon frequencies calculated for the CaF_2 rigid ion potentials at a_0=5.645A (Gillan 1986). The dashed lines show the experimental phonon frequencies of Elcombe and Pryor (1970)

3.3.2. The Calculation of the Short Range Forces

With good potentials the calculation of the short range forces is a straightforward procedure. As the Van der Waals and electron cloud repulsion potentials decay so rapidly we only have to calculate interactions between ions whose separation is less than some cut-off radius r_e ; for particle separations greater than r_e the short range forces will be insignificant when compared to the Coulombic forces.

The short range force between two particles α and β at positions r_α and r_β respectively is given by the differential of $V'_{\alpha\beta}(r)$ with respect to r where:

$$V'_{\alpha\beta}(r_{\alpha\beta}) = A_{\alpha\beta}\exp(-r_{\alpha\beta}/\rho_{\alpha\beta}) - C_{\alpha\beta}/r_{\alpha\beta}^6 \qquad (3.2)$$

and $r_{\alpha\beta} = |r_\alpha - r_\beta|$

Differentiating (3.2) gives the force on α due to β:

$$F_{SR}(r_{\alpha\beta}) = (-A_{\alpha\beta}/\rho_{\alpha\beta}\exp(-r_{\alpha\beta}/\rho_{\alpha\beta}) + 6C_{\alpha\beta}/r_{\alpha\beta}^{7})r_{\alpha\beta}/r_{\alpha\beta} \quad (3.3)$$

3.4. Coulombic Forces and the PPPM Algorithm

Now that we have dealt with the Van der Waals and electron cloud repulsion terms in the Born-Mayer-Huggins interionic potential we are still left with the problem of the Coulombic interaction. As this interaction is long-ranged every particle will interact electrostatically with every other particle, we can not introduce a cut-off radius as with the short-ranged forces. The traditional method of calculating the Coulombic interactions has been via the Ewald Sum technique. However the operation count for an Ewald Sum calculation goes as N^2, where N is the number of particles in the system. This behaviour is acceptable for systems of only a few hundred particles. For large simulations, however, the computational cost is simply too high, we must find an algorithm whose operation count goes as some lower power of N.

The Ewald summation technique computes the Coulombic force by splitting the calculation into two parts; one part in reciprocal space and one part in real space. The more of the calculation that is done in real space the more efficient the algorithm (Born and Huang 1954). The PPPM algorithm can be viewed in some ways as an extreme form of the Ewald sum in which all the calculations are done in reciprocal space.

In the PPPM algorithm (Hockney and Eastwood, 1981) the inter-particle force is split into two parts; a short range part F_{PP}^{SR} which is zero for particle separations greater than some cut-off radius r_e, and a smoothly varying long range part R. The short range forces are calculated explicitly for all pairs of particles whose separation is less than r_e (the particle-particle part of the algorithm) and the long range forces are approximated by the particle-mesh (PM) calculation. F_{PP}^{SR} is not quite the same as the force F_{SR} which was discussed in the previous section; the PM part of the algorithm introduces an extra short range force which has to be included in the Particle-Particle calculations.

The PM algorithm solves the Coulombic force calculations on a 32^3 mesh

which is superimposed on the 11x11x10 PP cells (See section 3.2). The main stages in the PM algorithm are:

1. Assign charges to the mesh
2. Solve for the potential on the mesh
3. Difference the potentials to find the mesh defined electric field
4. Interpolate from the forces on the mesh points to forces on the particles

The PM algorithm attempts to estimate **R** as accurately as possible, it is the accuracy of the interparticle forces that determine the accuracy of the simulation. In order to achieve this accuracy correction terms are intentionally introduced into the potential in order to cancel out some of the errors arising from the charge assignment, potential differencing and interpolation steps.

3.4.1. Charge Assignment and Force Interpolation Functions

Any function chosen for mapping the charges from the computational box to the mesh must satisfy three criteria:

1. Spatial localisation of errors: at particle separations that are large compared to the mesh spacing the fluctuations induced by ion motion should be small.
2. Smoothness: the charge assigned to the mesh should vary smoothly as the particle moves across the mesh, the fluctuations of spatially localised errors should be small.
3. Momentum conservation: any charge assignment scheme should avoid self forces.

Consider first a one-dimensional scheme in which the charge on a particle is distributed over the m nearest mesh points. Let $W_p(x)$ [= $W(x-x_p)$] be the fraction of the charge assigned to mesh point p at x_p and let $G(x'-x_p)$ be the potential at x' due to a unit charge at x_p.

The potential at x′ due to a unit charge at x will be given by:

$$\phi(x') = \sum_{p} W_p(x)G(x'-x_p) \tag{3.4}$$

The correct potential, $\phi_c(x')$, is a function only of $|x-x'|$ whereas $\phi(x')$ is a function both of $|x-x'|$ and $(x-x_p)$. As we can assign the charge from a particle to m different mesh points the function $W_p(x)$ has m degrees of freedom. We can therefore choose $W_p(x)$ to satisfy certain constraints. Charge conservation gives us the first constraint on $W_p(x)$:

$$\sum_{p} W_p(x) = 1 \tag{3.5}$$

We can use the other $m-1$ degrees of freedom of W_p to kill off some of the mesh dependence of $\phi(x')$. Taylor expanding equation (3.4) about $(x-x')$ we obtain:

$$\phi(x') = \sum_{p} W_p(x)G(x-x') + \sum_{p} W_p(x) \sum_{n} (x-x_p)^n/n![d^nG(x-x')/dx^n] \tag{3.6}$$

The mesh dependence of $\phi(x')$ comes in through the second term on the right hand side of equation (3.6). In general $G(x'-x) \sim 1/|x-x'|$ therefore:

$$(x-x_p)^n/n![d^nG(x-x')/dx^n] \sim 1/n![(x-x_p)/|x-x'|]^n \tag{3.7}$$

For $|x-x'| > |x-x_p|$ the largest mesh dependent term in equation (3.6) will be:

$$\sum_{n} W_p(x)(x-x_p)[dG(x-x')/dx] \tag{3.8}$$

The next largest term will be the n=2 term etc. We can therefore reduce the mesh dependence of $\phi(x')$ by choosing $W_p(x)$ such that

$$\sum_{n} W_p(x)(x-x_p)^n = \text{constant}$$
$$n < m \tag{3.9}$$

As ϕ_c and G are even functions of their parameters then the value of the constant must be zero for n odd. Equations (3.9) now give us a way of generating a hierarchy of charge assignment functions up to some arbitrary constants, the more mesh points over which the original charge is distributed the weaker the dependence of $\phi(x')$ on the mesh points. The freedom we have in choosing the values of the constants in equations (3.9) can be used to minimise the size of the local fluctuations (the second constraint).

As a particle moves through space the force on it should vary as smoothly as possible. The smoothness of the force can be defined as the number of derivatives of the force that are continuous as the particle crosses a cell boundary. The maximum number of derivatives that can be continuous is determined by m, the number of mesh points to which the charge from an ion is assigned. By examining equation (3.9) we can see that the highest order polynomial that can be uniquely fitted to m mesh points is of order m−1 i.e. if we assign the charge from a particle to three mesh points we should be able to make the first derivative of the force on the particle continuous as the particle crosses a mesh cell boundary. Using equations (3.9) and the smoothness constraint we can now construct a hierarchy of unique charge assignment functions. Assume that a particle of unit charge is at position x. The particles 3 nearest mesh points are at x_{p-1}, x_p and x_{p+1} (See fig. 3.5). Let W_{p-1}, W_p and W_{p+1} be the fractions of the charge assigned to each of the mesh points respectively. Equation (3.5) gives us that:

$$W_{p-1}(x) + W_p(x) + W_{p+1}(x) = 1 \tag{3.10}$$

From equations (3.9) we know that the function W_p must be a polynomial in x. Using the symmetry of the problem we therefore set

$$W_p(x) = ax^2 + b$$
$$W_{p-1}(x) = W_{p+1}(x) = cx^2+dx+e \tag{3.11}$$

As the particle crosses the cell boundary at H/2 its nearest mesh points become (0,1,2) and so for continuity of value we must have:

$$W_{p-1}(H/2) = 0$$

$$W_p(H/2) = W_{p+1}(H/2) \tag{3.12}$$

and for continuity of derivative:

$$d/dxW_{p+1}(H/2) = 0$$

$$d/dxW_{p-1}(H/2) = -d/dxW_{p+1}(H/2) \tag{3.13}$$

Solving equations (3.12) and (3.13) gives us the Triangular Shaped Cloud (TSC) charge assignment function:-

$$W_p(x) = .75-(x/H)^2$$

$$W_{p+1}(x) = W_{p-1}(x) = .5(.5+x/H)^2 \tag{3.14}$$

Using displacement invariance, equations (3.14) can be rewritten as:-

$$W(x) = \begin{cases} .75-(x/H)^2 & |x| < H/2 \\ .5(1.5-|x|/H)^2 & H/2 < |x| < 3H/2 \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

Substituting equations (3.15) into equations (3.9) we see that the TSC charge assignment scheme is the second-order-accurate three point scheme with the constant for n=2 equal to $H^2/4$. Figure 3.5 shows how this scheme can be thought of as assigning a triangularly shaped charge distribution to a particle, the amount of charge assigned to a mesh point is simply the amount of the cloud that overlaps the cell containing the mesh point.

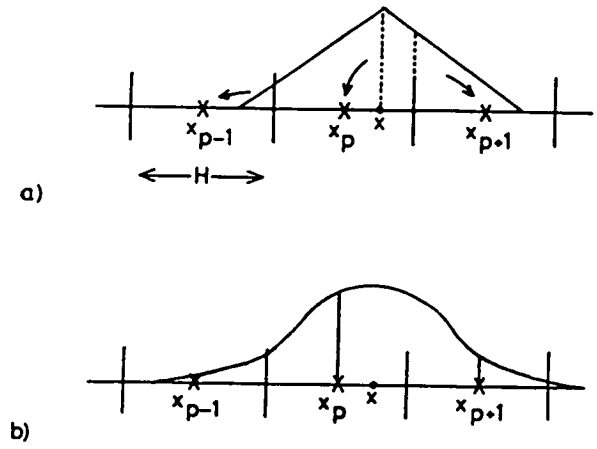


Figure 3.5 a) The cloud shape interpretation of charge assignment. The fraction of charge assigned from an ion at position x to a given mesh point is given by the area of the overlap between the cloud shape S and the cell containing the mesh point. b) The assignment function interpretation of charge assignment. The fraction of chare assigned to a mesh point is equal to the value of the assignment function W at the point.

The generalisation from a one-dimensional charge assignment scheme to a three dimensional scheme is straightforward. If $\mathbf{x} = (x,y,z)$ then define:

$$W(\mathbf{x}) = W'(x)W'(y)W'(z) \tag{3.16}$$

where W' is a one dimensional charge assignment function such as the TSC function mentioned above. If W' is the TSC function then W assigns the charge from a particle over the 27 neighbouring mesh points. Again this charge assignment scheme is second-order-accurate with respect to the three dimensional version of equation (3.9).

If the PPPM algorithm is to conserve momentum then the force on a particle of charge q_1 at $\mathbf{x}_1$ due to a particle of charge q_2 at $\mathbf{x}_2$ must be equal and opposite to the force on the particle at $\mathbf{x}_2$ due to the particle at $\mathbf{x}_1$. Consider the calculation of the force on the particle at $\mathbf{x}_1$ due to the particle at $\mathbf{x}_2$.

The charge assigned to a mesh point $\mathbf{p}$, $\delta\rho(\mathbf{x}_p)$, from the particle at $\mathbf{x}_2$ is given by:

$$\delta\rho(\mathbf{x_p}) = q_1 W(\mathbf{x_2}-\mathbf{x_p}) \tag{3.17}$$

We now formally define an operator $\mathbf{d}(\mathbf{x_p}:\mathbf{x_{p'}})$ which calculates the electric field at mesh point $\mathbf{p}$, $E(\mathbf{x_p})$, using the values of the charge density at mesh points $\mathbf{x_{p'}}$. In practice $\mathbf{d}(\mathbf{x_p}:\mathbf{x_{p'}})$ will be made up of a combination of Green functions and finite difference operators (see next subsection). Using this operator we can define:

$$E(\mathbf{x_p}) = \sum_{p'} \mathbf{d}(\mathbf{x_p}:\mathbf{x_{p'}})\rho(\mathbf{x_{p'}}) \tag{3.18}$$

Interpolating the force at $\mathbf{x_1}$ from the forces on the mesh points using the interpolation function $W'(\mathbf{x_1}-\mathbf{x_{p'}})$ we can write:

$$F(\mathbf{x_1}) = q_2 \sum_{p} W'(\mathbf{x_1}-\mathbf{x_p})E(\mathbf{x_p}) \tag{3.19}$$

Substituting equations (3.17) and (3.18) into equation (3.19) we get:

$$F(\mathbf{x_1}) = q_1 q_2 \sum_{p} \sum_{p'} W'(\mathbf{x_1}-\mathbf{x_{p'}})\mathbf{d}(\mathbf{x_p}:\mathbf{x_{p'}})W(\mathbf{x_2}-\mathbf{x_p}) \tag{3.20}$$

Similarly the force on the particle at $\mathbf{x_2}$ due to the particle at $\mathbf{x_1}$ is given by:

$$F(\mathbf{x_2}) = q_1 q_2 \sum_{p} \sum_{p'} W'(\mathbf{x_2}-\mathbf{x_{p'}})\mathbf{d}(\mathbf{x_{p'}}:\mathbf{x_p})W(\mathbf{x_1}-\mathbf{x_p}) \tag{3.21}$$

Defining W, W_p and $\mathbf{d}(\mathbf{x_{p'}}:\mathbf{x_p})$ so as to obey:

$$W \equiv W'$$

$$\mathbf{d}(\mathbf{x_p}:\mathbf{x_{p'}}) = -\mathbf{d}(\mathbf{x_{p'}}:\mathbf{x_p}) \tag{3.22}$$

then:

$$F(x_1) + F(x_2) = 0 \tag{3.23}$$

i.e if we use the same function to map charges onto the mesh as we use to interpolate the forces on the particles from the forces on the mesh and if the operator $d(x_p:x_{p'})$ has the correct symmetry properties then momentum will be conserved (assuming that all the arithmetic is done at infinite precision). $d(x_p:x_{p'})$ will have the required symmetry so long as the finite difference operator is properly balanced (see next subsection).

3.4.2. Solving the Potential and Finite Difference Operators

Now that we have a sensible scheme for assigning charges from the particles onto the mesh an efficient method must be found forcalculating the electric potential on the mesh. The potential is given by:

$$\nabla^2\phi(x) = \rho(x)/\varepsilon_0 \tag{3.24}$$

This equation is solved on the PM mesh using a Green function approach. We define a Green function $G(x_p-x_{p'})$ such that $G(x_p-x_{p'})$ is the potential at mesh point x_p due to a unit charge at mesh point $x_{p'}$. At this stage we will not attempt to find the form of G as it will eventually be calculated using a functional differentiation technique (see section 3.4.4).

Given G we can use the linearity of equation (3.24) to obtain the mesh defined electric potential. Convoluting G with the mesh defined charge density we obtain:

$$\phi(x_p) = \sum_{p'} G(x_p-x_{p'})\rho(x_{p'}) \tag{3.25}$$

Once the mesh defined potential has been calculated the mesh defined electric fields can be calculated using the relation:

$$\mathbf{E}(\mathbf{x}) = -\nabla\phi(\mathbf{x}) \tag{3.26}$$

Defining **D** to be a finite difference operator then the mesh defined electric field is given by:

$$\mathbf{E}(\mathbf{x}_p) = -\mathbf{D}\phi(\mathbf{x}_p) \tag{3.27}$$

In one dimension the simplest finite difference operator is:

$$D(x) = (\delta(x+H) - \delta(x-H))/2H, \tag{3.28}$$

the two point operator. We can improve the accuracy of the calculated derivative by including more points in our calculation. The four point operator is given by:

$$D(x_p) = \alpha(\delta(x_p+H)-\delta(x_p-H))/2H + (1-\alpha)(\delta(x_p+2H)-\delta(x_p-2H))/4H \tag{3.29}$$

The parameter α is again determined later via a minimisation procedure. We obtain the mesh defined electric fields by convoluting the electric field with D. The generalisation of D to three dimensions is straightforward, we simply define $\mathbf{D} = (D_x, D_y, D_z)$ where D_x is the two or four point finite difference operator acting in the x direction etc.

From equation (3.27) we can see that:

$$\mathbf{E}(\mathbf{x}_p) = -\mathbf{D}\phi(\mathbf{x}_p) = -\mathbf{D}\sum_{p'} G(\mathbf{x}_p-\mathbf{x}_{p'})\rho(\mathbf{x}_{p'}) \tag{3.30}$$

Comparing this with equation (3.18) we can now see the form of $\mathbf{d}(\mathbf{x}_p;\mathbf{x}_{p'})$. As G is an even function of its parameters, $\mathbf{d}(\mathbf{x}_p;\mathbf{x}_{p'})$ will have the transformation property defined in (3.22) provided that **D** transforms in the same way. Both the two point finite difference operator and the four point finite difference operator have this property.

3.4.3. Transform Space Analysis

The steps in the PM algorithm as described in the previous two subsections are summarised below.

1. Charge Assignment

$$\rho(\mathbf{x}_p) = q\sum_i W(\mathbf{x}_i-\mathbf{x}_p) \tag{3.31}$$

where $\{\mathbf{x}_p\}$ are the position of the mesh points and $\{\mathbf{x}_i\}$ are the positions of the particles.

2. Solving the Potential on the Mesh

$$\phi(\mathbf{x}_p) = \sum_{p'} G(\mathbf{x}_p-\mathbf{x}_{p'})\rho(\mathbf{x}_{p'}) \tag{3.32}$$

3. Calculation of the Electric Field

$$\mathbf{E}(\mathbf{x}_p) = -\mathbf{D}\phi(\mathbf{x}_p) \tag{3.33}$$

4. Interpolate from Mesh Defined Forces to Particle Forces

$$\mathbf{F}(\mathbf{x}_i) = q\sum_p W(\mathbf{x}_i-\mathbf{x}_p)\mathbf{E}(\mathbf{x}_p) \tag{3.34}$$

Equations (3.31) to (3.34) now give us an approximation scheme for calculating long range forces on the mesh. However we still have no idea as to the errors introduced by the mesh approximation or how these errors can be minimised. In order to investigate these two questions we can investigate equations (3.31) to (3.34) in reciprocal space.

The symbol $\Rightarrow$ is used to define "transforms to" or "whose transform is". A circumflex above a function indicates that the quantity is the Fourier transform of the real space function defined by the same symbol. The superscript ' denotes continuous functions, the superscript † is used to denote functions

defined only at mesh points. The symbol $\oplus$ is used to denote convolution.

First consider the case of an infinite one dimensional system with mesh spacing H.

3.4.3.1. The Charge Assignment Step

The density of the particle centres is given by:

$$n(x) = \sum_i \delta(x-x_i) \tag{3.35}$$

where x_i is the position of the i*th* particle.
Define the function $\zeta(x)$ to be an infinite row of δ-functions at unit spacing, the 'comb' function.

$$\zeta(x) = \sum_n \delta(x-nH) \tag{3.36}$$

Using equations (3.35) and (3.36) equations (3.31) can be rewritten as:

$$\rho^{\dagger}(x) = \zeta(x/H)n(x)\oplus G(x) = \zeta(x/H)\rho'(x) \tag{3.37}$$

The mesh defined particle charge density $\rho^{\dagger}(x)$ is calculated by convoluting the particle density function with the charge assignment function and then sampling this function at mesh points. Using the convolution theorem equation (3.37) can be Fourier transformed:

$$\hat{\rho}^{\dagger}(k) = H\zeta(kH/2\pi)\oplus\hat{\rho}'(k) \tag{3.38}$$

where $\zeta(x/H) \Rightarrow \zeta(kH/2\pi)$. Explicitly evaluating the convolution in equation (3.37) we obtain:

$$\hat{\rho}^{\dagger}(k) = \int dk' \sum_n \delta(k-nk_g)\hat{\rho}'(k-k') \tag{3.39}$$

where $k_g = 2\pi/H$

Integrating out the δ-functions we get:

$$\hat{\rho}^{\dagger}(k) = \sum_{n} \hat{W}(k-nk_g)\hat{n}(k-nk_g) \tag{3.40}$$

3.4.3.2. The Potential Solver

For an infinite one dimensional system, equation (3.32) becomes:

$$\phi(x_p) = H\sum_{p'} G(x_p-x_{p'})\rho(x_{p'}) \tag{3.41}$$

This equation can be recognised as the convolution sum for a system which is discrete in x space and periodic in k space. Therefore

$$\phi(x_p) \Rightarrow \hat{\phi}(k) = \hat{G}(k)\hat{\rho}(k) \tag{3.42}$$

where periodicity implies that for n integer

$$\hat{\phi}(k+nk_g) = \hat{\phi}(k). \tag{3.43}$$

3.4.3.3. Force Interpolation

From equation (3.28) we can see that the electric field on the mesh, $E^{\dagger}(x)$, is obtained by convoluting the finite difference operator with the mesh defined potential and then sampling this function at the mesh points.

$$E^{\dagger}(x) = \zeta(x/H)D(x)*\phi(x) \tag{3.44}$$

The force on the particles at positions $\{x_i\}$ is then obtained by taking the convolution of $E^{\dagger}(x)$ with the charge assignment function. Therefore

$$\hat{F}(k) = q/H\hat{W}(k)\hat{E}^{\dagger}(k) \tag{3.45}$$

where

$$\hat{E}^{\dagger}(k) = H\zeta(k/k_g)\oplus\hat{E}'(k) \tag{3.46}$$

$$= \sum_n \hat{E}'(k-nk_g) \tag{3.47}$$

$$= -\hat{D}(k)\hat{\phi}(k) \tag{3.48}$$

In equation (3.48) the sum over n serves only to periodically repeat $\hat{E}'$ outside the interval $|k| < k_g/2$ since by construction E'(x) is a band limited function. We can replace the sum over n using equation (3.43) ($\hat{\phi}(k+nk_g)=\hat{\phi}(k)$).

3.4.3.4. The Interparticle Force

Using equations (3.40),(3.41) and (3.48) we can now write down the Fourier transform of the mesh calculated force on a particle of charge q at x_2 due to a charge q at x_1.

$$F(x_2) = \int dk/(2\pi)\hat{F}(k)\exp(ikx_2) \tag{3.49}$$

where:

$$\hat{F}(k) = -(q/H)\hat{W}(k)\hat{E}^{\dagger}(k)$$

by equation (3.45)

$$= -(q/H)\hat{W}(k)\hat{D}(k)\hat{\phi}(k)$$

by equation (3.48)

$$= -(q/H)\hat{W}(k)\hat{D}(k)\hat{G}(k)\hat{\rho}(k)$$

by equation (3.42)

$$= (q^2/H^2)\hat{W}(k)\hat{D}(k)\hat{G}(k)\sum_n \hat{W}(k-nk_g)\hat{n}(k-nk_g)$$

by equation (3.40). Using the identity

$$n(x) = \delta(x-x_1) \Rightarrow \hat{n}(k) = \exp(-ikx_1) \tag{3.50}$$

we finally get:

$$\hat{F}(k) = (q^2/H^2)\hat{W}(k)\hat{D}(k)\hat{G}(k)\sum_n \hat{W}(k-nk_g)\exp[-i(k-nk_g)x_1] \tag{3.51}$$

Substituting equation (3.51) into equation (3.49) we get the particle-mesh calculated interparticle force.

$$F(x;x_1) = \int dk/(2\pi)[(q^2/H^2)\hat{W}(k)\hat{D}(k)\hat{G}(k)\sum_n \hat{W}(k-nk_g)\exp(-ink_gx_1)]\exp(ikx) \tag{3.52}$$

where $x = x_2-x_1$ is the particle separation. The sum over n in this equation arises from the charge assignment term and is the sole cause of any loss of displacement invariance. The n = 0 term in the sum introduces a periodicity of period length H in real space. We can now generalise the results obtained in paragraphs 3.4.3.1 to 3.4.3.4 to handle the case of finite three dimensional systems.

3.4.3.5. Generalisation to Three Dimensions and Finite Systems

In moving from an infinite system to a finite system we now have a discrete set of equally spaced allowed wave vectors, the separation being $k_0 = 2\pi/L$ where L is the period box length. This means that terms such as $\int dk$ become $1/L\sum$ (In three dimensions substitute **l** for L where $\mathbf{l} = (l_1,l_2,l_3$ and replace 1/L by $1/V_b$ where V_b is the volume of the computational box). In three dimensions equations (3.40), (3.42), (3.45) and (3.48) become:

Charge density

$$\rho'(\mathbf{x}) = (q/V_c)W(\mathbf{x})\circledast n(\mathbf{x}) \Rightarrow \hat{\rho}'(\mathbf{k}) = (q/V_c)\hat{W}(\mathbf{k})\hat{n}(\mathbf{k}) \tag{3.53}$$

$$\rho^{\dagger}(\mathbf{x}) = \zeta(\mathbf{x};\mathbf{H})\rho'(\mathbf{x}) \Rightarrow \rho'(\mathbf{k}) = (q/V_c)\sum_{\mathbf{n}} \hat{W}(\mathbf{k}_n)\hat{n}(\mathbf{k}_n) \tag{3.54}$$

Potential

$$\phi'(\mathbf{x}) = G'(\mathbf{x}) \oplus \rho^{\dagger}(\mathbf{x}) \Rightarrow \hat{\phi}'(\mathbf{k}) = \hat{G}'(\mathbf{k})\hat{\rho}(\mathbf{k}) \tag{3.55}$$

$$\phi^{\dagger}(\mathbf{x}) = \zeta(\mathbf{x};\mathbf{H})\phi'(\mathbf{x}) \Rightarrow \hat{\phi}(\mathbf{k}) = \hat{G}(\mathbf{k})\hat{\rho}(\mathbf{k}) \tag{3.56}$$

Electric field

$$\mathbf{E}'(\mathbf{x}) = -\mathbf{D}(\mathbf{x}) \oplus \phi'(\mathbf{x}) \Rightarrow \hat{\mathbf{E}}'(\mathbf{k}) = -\hat{\mathbf{D}}(\mathbf{k})\hat{\phi}'(\mathbf{k}) \tag{3.57}$$

$$\mathbf{E}^{\dagger}(\mathbf{x}) = \zeta(\mathbf{x};\mathbf{H})\mathbf{E}'(\mathbf{x}) \Rightarrow \hat{\mathbf{E}}(\mathbf{k}) = -\hat{\mathbf{D}}(\mathbf{k})\hat{\phi}(\mathbf{k}) \tag{3.58}$$

Force

$$\mathbf{F}(\mathbf{x}) = (q/V_c)W(\mathbf{x}) \oplus \mathbf{E}^{\dagger}(\mathbf{x}) \Rightarrow \hat{\mathbf{F}}(\mathbf{k}) = q/V_c \hat{W}(\mathbf{k})\hat{\mathbf{E}}^{\dagger}(\mathbf{k}) \tag{3.59}$$

where

$$V_c = H_1H_2H_3 = \text{computational box volume} \tag{3.60}$$

$$\mathbf{H} = (H_1,H_2,H_3) = \text{cell vector} \tag{3.61}$$

$$\zeta(\mathbf{x};\mathbf{H}) = \zeta(x/H_1)\zeta(y/H_2)\zeta(z/H_3) \tag{3.62}$$

$$\mathbf{n} = (n_1,n_2,n_3) \tag{3.63}$$

$$\mathbf{k_g} = 2\pi(1/H_1,1/H_2,1/H_3) = \text{grid wavenumber} \tag{3.64}$$

$$\mathbf{k_n} = \mathbf{k} - 2\pi(n_1/H_1,n_2/H_2,n_3/H_3) \tag{3.65}$$

We can now use all the above results to write down the finite, three dimensional version of equation (3.52). The mesh calculated force on a unit positive charge at $\mathbf{x_2}$ due to a unit negative charge at $\mathbf{x_1}$ is:

$$\mathbf{F}(\mathbf{x_2};\mathbf{x_1}) = (1/V_b)\sum_{\mathbf{k}} (\hat{W}(\mathbf{k})/V_c)\hat{\mathbf{D}}(\mathbf{k})\hat{G}(\mathbf{k})\sum_{\mathbf{n}} (\hat{W}(\mathbf{k_n})/V_c)\exp(-i\mathbf{k_n}\cdot\mathbf{x_1})\exp(i\mathbf{k}\cdot\mathbf{x_2}) \tag{3.66}$$

Using equation (3.66) we can calculate the errors induced by the particle-mesh approximation and choose the Green function G and the four-point finite

difference operator parameter α so as to minimise this error.

3.4.4. Optimisation of the PM algorithm

Now that we have a scheme for calculating long range forces on a mesh we must find a way of efficiently calculating all the interparticle forces for several thousand ions interacting via a Born-Huggins-Mayer potential (equation (3.1)). Denote the total Born-Mayer-Huggins by $\mathbf{F}^{TOT}$ and split it into two parts, a smoothly varying long range force **R** which will be calculated on the mesh and a short range force $\mathbf{F}_{PP}^{SR}$ which is calculated explicitly for all pairs of particles separated by less than some cut-off radius, r_e. The obvious choice for **R** is the Coulombic term in the Born-Mayer-Huggins inter-particle force, however there are problems involved in such a straightforward choice for **R**. As we are modelling **R** on a finite, periodic, discrete lattice we can not represent any of the harmonic content of $\hat{\mathbf{R}}(\mathbf{k})$ beyond the Brillouin zone boundary i.e. for $|k_i| > \pi/H_i$. In order to model accurately the long range forces $|\hat{\mathbf{R}}(\mathbf{k})|$ for $|k_i| > \pi/H_i$ must be negligible. If we choose $\mathbf{R}(\mathbf{r}) \sim \mathbf{r}/|\mathbf{r}|^3$ then $\hat{\mathbf{R}}(\mathbf{k}) \sim |\mathbf{k}|^{-1}$, $\hat{\mathbf{R}}(\mathbf{k})$ does not decay fast enough, we need to choose a form for **R** such that $\hat{\mathbf{R}}(\mathbf{k})$ decays as a much higher power of $1/|\mathbf{k}|$.

If we choose $\mathbf{R}(\mathbf{r})$ to be the force between two charge distributions, not the force between two point charges, in effect if we assign a finite size to the ions then $\hat{\mathbf{R}}(\mathbf{k})$ does decay much more rapidly. If we give each ion a density (charge) profile determined by:

$$\mathbf{S}(\mathbf{r}) = (48/\pi a^4)(a/2-r) \quad r < a/2$$

$$\mathbf{S}(\mathbf{r}) = 0 \qquad \text{otherwise} \tag{3.67}$$

where a is the diameter assigned to a particle, then the force between two such S shaped particles is given by:

$$\mathbf{R}(\mathbf{r}) = (1/4\pi\varepsilon_0)\mathbf{r}/|\mathbf{r}|^3 \oplus \mathbf{S}(\mathbf{r}) \oplus \mathbf{S}(\mathbf{r}) \tag{3.68}$$

which can be expressed in polynomial form as:

$$R(r) = 1/(4\pi\varepsilon_0)(224y-224y^3+70y^4+48y^5-21y^6)/(35a^2)$$

$$(0<=y<=1)$$

$$R(r) = 1/(4\pi\varepsilon_0)(12/y^2-224-896y-840y^2+224y^3+70y^4-48y^5+7u^6)/(35a^2)$$

$$(1<=y<=2)$$

$$R(r) = 1/(4\pi\varepsilon_0)(1/r^2) \qquad (3.69)$$

$$(y>=2)$$

where y = 2r/a. Fourier transforming equation (3.68) we get:

$$\hat{R}(k) = -ik\hat{S}^2/(|k|^2 4\pi\varepsilon_0) \qquad (3.70)$$

where

$$\hat{S}(k) = 12/(ka/2)^4\{2-2\cos(ka/2)-(ka/2)\sin(ka/2)\} \qquad (3.71)$$

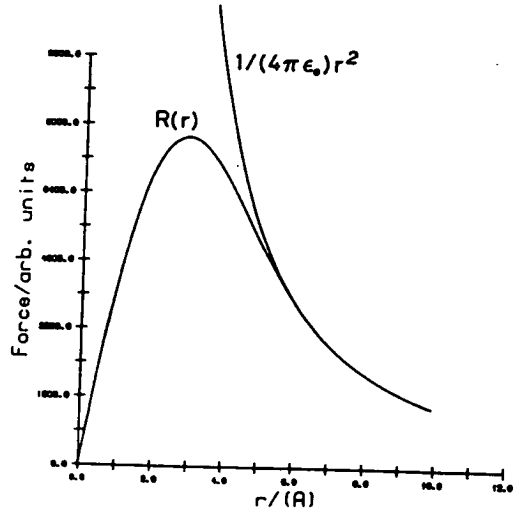


Figure 3.6 The force between two S shaped charges in arbitrary units compaired with the force between two point charges.

Thus, when R(r) is the force between two S shaped charges, $\hat{R}(k) \sim |k|^{-9}$. By assigning shapes to the charges we solve the problem of the high harmonics

of $\mathbf{R}(r)$. Figure 3.6 shows the force between two S shaped charges as a function of their separation. By comparing $\mathbf{R}$ with the equivalent force between point charges we see that for $r > a$, $\mathbf{R}$ follows the Coulombic force and for $r < a$ decays smoothly to zero. As we really want to model the forces between point charges then we must correct the forces given by the PM algorithm for particles whose separation is less than a, we must add to the value of the PM force the difference between the Coulombic force and $\mathbf{R}$, this correction term is included in the calculation of the short range forces. The short range force calculated over all pairs of particles within r_e of each other, $\mathbf{F}_{SR}^{PP}$, contains both the electron shell repulsion and Van der Waals forces discussed in section 3.3.2 and this extra correction term.

Define $\mathbf{F}(x;x_1)$ to be the mesh calculated force between two particles at x_1 and x_2 ($x = x_2 - x_1$). The displacement averaged total squared deviation of $\mathbf{F}$ from $\mathbf{R}$ is given by:

$$Q = (1/V_c)\int_{V_c} dx_1 \int_{V_b} dx |\mathbf{F}(x;x_1) - \mathbf{R}(x)|^2 \tag{3.72}$$

If we define the dimensionless quantity $Q^\dagger$ by:

$$Q^\dagger = Q/\{(4/3\pi\varepsilon_0 V_c)(1/4\pi\varepsilon_0 H_k^2)^2\} \tag{3.73}$$

where H_k is the largest of H_1, H_2 and H_3, then $\sqrt{Q^\dagger}$ gives a weak estimate on the upper bound of the deviation of the PM interparticle force from the reference force $\mathbf{R}$ for a pair of particles at any separation and any location with respect to the mesh.

Using the results of section 3.4.3 Q can now be computed. Applying the power theorem to equation (3.72) we obtain:

$$Q = (1/V_c)\int_{V_c} (1/V_b)\sum_{k} [|\hat{\mathbf{F}}(k;x_1)| - 2\hat{\mathbf{R}}(k)^*.\hat{\mathbf{F}}(k;x_1) + |\hat{\mathbf{R}}(k)|^2] \tag{3.74}$$

where * represents complex conjugation. From equation (3.65) we have:

$$\hat{F}(\mathbf{k};\mathbf{x}_1) = \hat{W}(\mathbf{k})/V_c\hat{D}(\mathbf{k})\hat{G}(\mathbf{k})\sum \hat{W}(\mathbf{k}_n)/V_c\exp(-i(\mathbf{k}-\mathbf{k}_n).\mathbf{x}_1) \tag{3.75}$$

Therefore

$$Q = (1/V_c)\int_{V_c} d\mathbf{x}_1 \sum_{\mathbf{k}} \Big[\Big|\hat{W}(\mathbf{k})/V_c\hat{D}(\mathbf{k})\hat{G}(\mathbf{k})\sum_{n} \hat{W}(\mathbf{k}_n)/V_c\exp(-i(\mathbf{k}-\mathbf{k}_n).\mathbf{x}_1)\Big|^2 - 2\hat{R}(\mathbf{k})^*.\hat{W}(\mathbf{k})/V_c\hat{D}(\mathbf{k})\hat{G}(\mathbf{k})\sum_{n} \hat{W}(\mathbf{k}_n)/V_c\exp(-i(\mathbf{k}-\mathbf{k}_n.\mathbf{x}_1)) + |\hat{R}(\mathbf{k})|^2\Big] \tag{3.76}$$

Performing the integral over V_c we get

$$Q = (1/V_b)\sum_{\mathbf{k}} \Big[\hat{U}^2(\mathbf{k})|\hat{D}(\mathbf{k})|^2\hat{G}^2(\mathbf{k})\sum_{n} \hat{U}^2(\mathbf{k}_n) - 2\hat{R}^*(\mathbf{k}).\hat{D}(\mathbf{k})\hat{G}(\mathbf{k})\hat{U}^2(\mathbf{k}) + |\hat{R}(\mathbf{k})|^2\Big] \tag{3.77}$$

We can now use the periodicity of $\hat{D}(\mathbf{k})$ and $\hat{G}(\mathbf{k})$ rewrite equation (3.77) in terms of wave-vectors in the first Brillouin zone.

$$Q = (1/V_b)\sum_{\underline{k}} \Big[|\hat{D}(\mathbf{k})|^2\hat{G}^2(\mathbf{k})\Big\{\sum_{\underline{n}} \hat{U}^2(\mathbf{k}_n)\Big\}^2 - 2\hat{D}(\mathbf{k})\hat{G}(\mathbf{k}).\sum_{\underline{n}} \hat{R}(\mathbf{k}_n)^*\hat{U}^2(\mathbf{k}_n) + \sum_{\underline{n}} |\hat{R}(\mathbf{k}_n)|^2\Big] \tag{3.78}$$

Q is a quadratic functional of $\hat{G}$. Mininising Q with respect to $\hat{G}$ by functionally differentiating Q with respect to $\hat{G}$ we obtain the optimal Green function and minimum value of $Q = Q_{opt}$:

$$\hat{G}(\underline{k}) = \frac{\underline{D}(\underline{k}).\sum_{\underline{n}} \hat{U}^2(\underline{k}_n)\,\hat{\underline{R}}^*(\underline{k}_n)}{|\underline{D}(\underline{k})|^2\left(\sum_{\underline{n}} \hat{U}^2(\underline{k}_n)\right)^2} \tag{3.79}$$

and

$$Q_{opt} = \frac{1}{V_b}\sum_{\underline{k}}\left(\sum_{\underline{n}} |\hat{\underline{R}}(\underline{k}_n)|^2 - \frac{|\underline{D}(\underline{k}_n).\sum_{\underline{n}} \hat{U}^2(\underline{k}_n)\,\hat{\underline{R}}^*(\underline{k}_n)|^2}{|\hat{\underline{D}}(\underline{k})|^2\left(\sum_{\underline{n}} \hat{U}^2(\underline{k}_n)\right)^2}\right) \tag{3.80}$$

Even though both these functions contain triply infinite sums they can be evaluated exactly. All the sums either converge very rapidly or can be written in closed form. First we obtain expressions for each of the terms mentioned in equations (3.79) and (3.80). Choosing **D** to be the four-point finite difference operator (equation (3.29)) then:

$$\hat{D}_j = i\alpha\sin(k_jH_j)/H_j + i(1-\alpha)\sin(2k_jH_j)/(2H_j) \qquad (3.81)$$

$$j = 1,2,3$$

W is chosen to be the TSC assignment function (equation (3.15)). On Fourier transforming W and dividing by V_c we get:

$$\hat{U}(\mathbf{k}) = [\sin(k_1H_1/2)\sin(k_2H_2/2)\sin(k_3H_3/2)/\{(k_1H_1/2)(k_2H_2/2)(k_3H_3/2)\}]^3 \qquad (3.82)$$

The sum $\sum_{\mathbf{n}} \hat{U}^2(\mathbf{k_n})$ in equations (3.79) and (3.80) can be written in closed form. Substituting equation (3.82) into the sum we obtain:

$$\sum_{\mathbf{n}} \hat{U}^2(\mathbf{k_n}) = \prod_{i=1}^{3} \left(\sin^6(k_iH_i/2) \sum (k_iH_i/2-\pi n_i)^{-6} \right) \qquad (3.83)$$

The identity

$$\sin^{-6}x(1-\sin^2x+(2/15)\sin^4x) = \sum_{n} (x-n\pi)^6 \qquad (3.84)$$

reduces equation (3.83) to:

$$\sum_{\mathbf{n}} \hat{U}^2(\mathbf{k_n}) = \prod_{i=1}^{3} \{1-\sin^2(k_iH_i/2) +(2/15)\sin^4(k_iH_i/2)\} \qquad (3.85)$$

$\hat{\mathbf{R}}(\mathbf{k})$ is given by equations (3.70) and (3.71). Using these two equations we see that $|\hat{\mathbf{R}}(\mathbf{k})| \sim |\mathbf{k}|^{-18}$ therefore we need only calculate a few terms of $\sum|\hat{\mathbf{R}}(\mathbf{k_n})|^2$ for it to converge within the desired accuracy.

Using equations (3.81) to (3.85) we can exactly calculate equations (3.79) and (3.80) and so obtain both the optimal Green function and a measure for the difference between the mesh calculated force and the reference force **R**. Using equation (3.80) we can investigate the effect of varying the various parameters in the PM algorithm and so choose the most cost effective form for the parameters before we start writing any of the simulation code. There are two parameters in the algorithm that have to be fixed; a, the diameter of the

charge distribution that we place on the ions and α, the free parameter in the four point finite difference operator (equation (3.29)) The system being simulated is of 11x11x10 $SrCl_2$ units (see section 3.1). If d is the spacing between two strontium ions then we have:

$$H_1 = 11d/32$$
$$H_2 = 11d/32$$
$$H_3 = 10d/(32\sqrt{2})$$

A 32^3 point mesh is used in these simulations. We define the quantity $Q'=100/Q^{\dagger}$. The value of α is found by minimising Q' with respect to α for various values of a/H_1. Figure 3.7 shows the variation of α_{min} with a/H_1. Figure 3.8 shows the variation of Q' with α for a/H_1=4.15505.

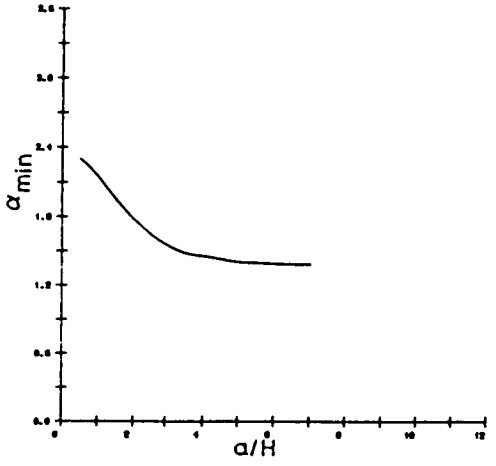


Figure 3.7 The variation of α_{min} with a/H.

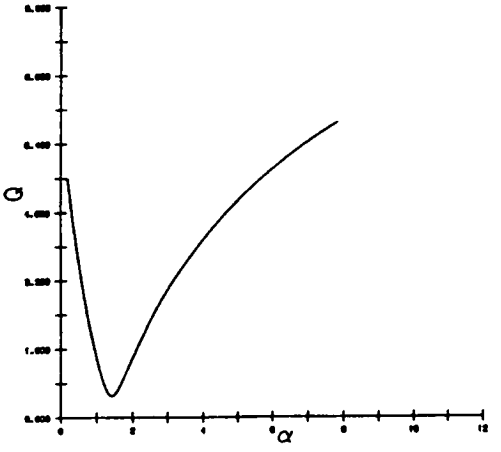


Figure 3.8 The variation of Q with α, a/H = 4.155.

Using the techniques described above we can now calculate the minimum value of Q', Q'_{opt}, for given values of a/H_1 (figure 3.9). Looking at figure 3.9 we can see that by increasing the value of $(a)/H_1$ we can make Q'_{opt} arbitrarily small. However, a is related to r_e, the cut-off radius of the short range force, a large value for a implies a large r_e and an expensive simulation. Hockney and Eastwood have found that a PM calculation of an ionic system with Q' less than or equal to 1% gives good energy stability, therefore we require that $a/H_1 >= 3.4$. At first it would seem that we should set r_e= a, however r_e can be set to be significantly smaller than a as the reference force **R** follows the point

particle Coulombic force for some $r < a$. It has been found empirically (Hockney and Eastwood) that a good measure of the lower bound on r_e is given by the cube root of the auto-correlation volume of the charge shapes, i.e.

$$r_e^3 >= \int dx \int dx' \, S(x)S(x+x')/\int S^2(x)dx \tag{3.86}$$

giving

$$r_e >= 0.7a \tag{3.87}$$

So in choosing a value for a/H_1 such that $Q' < 1\%$ we also determine the value of r_e.

The simulation was run with a=7.302A, r_e=6.0A, α=1.44, giving Q'=0.4896%

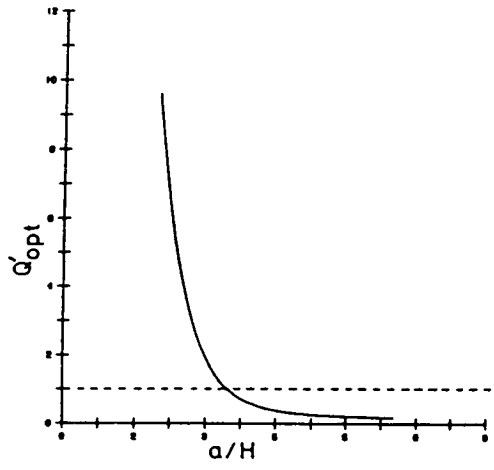


Figure 3.9 The variation of Q_{opt} with a/H.

3.5. Implementation of the PM Algorithm on the DAP

There are six stages to calculating the PM algorithm on the mesh:

1. Map particles from their 11x11x10 PP positions to their nearest 32^3 PM grid point.

2. Calculate the charge density on the 32^3 PM grid.

3. Fourier transform the charge density, multiply by $\hat{G}(\mathbf{k})$ and Fourier transform back.

4. Calculate the mesh-defined electric field.

5. Calculate the forces on the particles on the 32^3 PM grid.

6. Map the forces back from the 32^3 PM grid to the 11x11x10 PP grid.

Steps 2 - 5 are all calculated on the 32^3 PM grid, steps 1 and 6 both involve mappings between the 11x11x10 PP grid and the 32^3 PM grid.

3.5.1. The Charge Assignment Step

Calculation of the charge density is done in two stages; first we map particle data from the PP to the PM mesh and second we calculate the TSC distribution for particles on the PM mesh.

3.5.1.1. Mapping from the PP Grid to the PM Grid

In this subsection three different grids will be discussed; the 64x64 grid of PEs on the DAP, the 11x11x10 PP grid used to calculate the short range forces (see fig. 3.2), and the 32^3 PM mesh which is superimposed on the PP mesh in order to calculate the long range forces. Figure (3.3) shows how the PP grid is mapped onto the PE grid, figure (3.10) shows how the 32^3 PM grid is mapped onto the PE grid. There are two ways that positional data from the particles can be stored on the PEs. The first is the PP data-representation in which all the coordinates of one type of particle, e.g. the strontium ions, is stored on one DAP `REAL*4` matrix as shown in figure 3.3. The other way that data can be stored is in the PM data-representation in which particle data is spread over three matrix arrays `WX(,,8)`, `WY(,,8)` and `WZ(,,8)` as shown in fig 3.10.

Note that every PM mesh cell volume maps directly onto one, and only one, PP mesh cell and that there is a direct mapping between PP mesh cells and PEs. If we have a particle whose data in the PP data-representation is stored on three given PEs (one PE for the x-data, one for the y-data etc, see figure

3.3) then, in general, that particle's position will be contained within the PP mesh cell that maps directly onto the same PEs; the particle is in the volume controlled by its 'local PEs' (when we talk about 'local PEs' we will always mean the PEs containing a particles coordinates in the PP data-representation). The coordinates of a particle in the PP data-representation are given with respect to the minimum x, y and z coordinates of the PP mesh cell which maps onto the PEs containing the data for that particle. The origin of the PP cell will be contained within one of the PM cells, we define this PM cell to be at the origin of its local PP cell (the distribution of the PM cells at the origins of their local PP cells is shown in figure 3.11). When the coordinates are mapped from the PP data-representation to the PM data-representation, the particle's coordinates will be mapped onto one of the PM mesh cells whose volume is contained within the original PP mesh cell.

It is possible for a particle to have coordinates large enough that it is not contained within the volume controlled by its local PEs, for example a strontium ion undergoing a particularly large excursion from its lattice site. When the coordinates of such a particle are mapped from the PP to the PM data-representation the coordinates of the particle will be mapped onto a PM cell which does not map onto the particle's local PP cell, but onto a PP cell which is a nearest neighbour to the local PP cell.

The first problem faced in computing the PM algorithm is in moving between the PP and the PM data-representations. Given a data-plane of particle coordinates as shown in fig. (3.3) how do we map all of the x, y and z coordinates of the particles onto their nearest grid points as stored on the DAP matrix arrays `WX(,,8)`, `WY(,,8)` and `WZ(,,8)`? Consider first the strontium ion contained in the PP cell at the origin of the molecular dynamics cell, i.e. the strontium whose x-coordinate ,X, is stored on the PE at (1,1), whose y-coordinate, `Y`, is stored on the PE at (1,22) and whose z-coordinate, `Z`, is stored on the PE at (1,43) on the PE array. Any particle whose coordinates are stored on these three PEs must be closest to one of the 36 PM mesh points enclosed in the volume mapped onto the three PEs, i.e on one of the PM grid cells (i,j,k) where i = 1,2,3; j = 1,2,3; k = 1,2,3,4. We start by assigning the x coordinate of the strontium contained in the (1,1,1) PP cell to the element (1,1,1) of the array `WX` (similarly the y coordinate to `WY(1,1,1)` and the z coordinate to `WZ(1,1,1)`). To calculate which of the four possible PM x-y

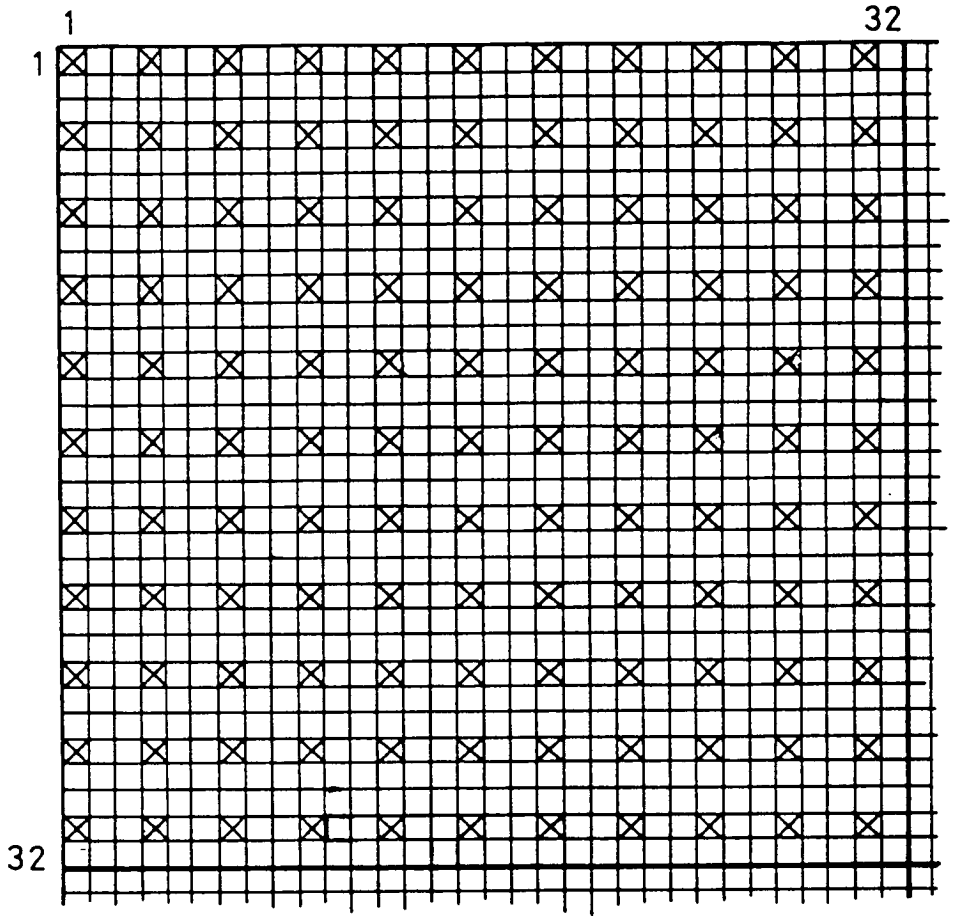


Figure 3.11 The PM cells in the first PM z-plane containing the origins of the PP cells. This same patern is repeated on PM planes 5, 8, 11, 14, 17, 20, 24, 27 and 30 (see figures 3.2 and 3.10).

planes the particle lies on we calculate an integer NZ,

$$NZ = INT(Z/H_3)+1$$

where NZ is the number of the PM x-y plane containing the strontium ion. We can now zero the elements of WZ(,,1) for which the particle data is on the wrong PM x-y plane, we also zero the equivalent entries in the arrays WX(,,1) and WY(,,1).

Now that the particle data is on the correct x-y plane we must shift the data onto the correct PM cell on this plane. We can determine the correct position by looking at the two integers NX and NY where:

$$NX = INT(X/H_1)+1$$
$$NY = INT(Y/H_2)+1$$

The value of NX tells us how many places we should shift the data to the South, if NX equals zero then the particle data is already on the correct row, if NX equals 1 then the particle data must be shifted one place to the South etc. We shift the data on WX(,,1), WY(,,1) and WZ(,,1) by the same amount. This process is then repeated with the y-coordinate to find out how far to shift the data to the East. We now have the data from the (1,1,1) position on the PP grid stored in the correct position on the PP grid.

It would be very inefficient on a parallel computer to shift the data one particle at a time, we must find a way of assigning data to four PM x-y planes simultaneously. Consider the mapping of the first x-y plane of the PP mesh onto the PM grid. All the particles stored on the first PP plane must be found somewhere on the first four planes of the PM grid, we therefore map all the coordinates from the first PP plane onto the positions on the PP plane shown in figure 3.12. The algorithm used to achieve this mapping will be discussed later on in this subsection. As before we start by discovering which of the x-y PM planes each particle lies on and zeroing that particle's data stored on the other x-y PM planes. For a particle to be on the first quadrant of the PE grid, i.e on the first PM plane, it must have a z-coordinate between 0 and H_3. For it to be on the second quadrant it should have a value between H_3 and $2H_3$ etc. We can discover which of the quadrants each of the particles belongs to by

constructing a matrix NZ(,):

```
NZ(,) = INT(WZ(,,1)/H3)
```

We can use this matrix to discover which of the elements of WZ(,,1) are on the correct x-y quadrant. If we construct an integer matrix NTESTZ(,) which is set equal to 0 for values on the first quadrant, 1 for values on the second quadrant, etc., then we can construct a logical matrix LZ(,) which is .TRUE. at the positions at which the particle data is on the correct x-y plane.

```
LZ(,) = NTESTZ(,).EQ.NZ(,)
```

We can now use LZ(,) to eliminate all the data not on the correct PM quadrant.

```
WX(.NOT.LZ(,),1) = 0.
WY(.NOT.LZ(,),1) = 0.
WZ(.NOT.LZ(,),1) = 0.
```

We now have matrices WX(,,1), WY(,,1) and WZ(,,1) in which all the data from the first plane of the PP grid is on the correct data plane on the PM grid. We can now use a similar technique to calculate the shifts to the South. We calculate a integer matrix NX(,) which tells us how far we must shift each data-value in the South direction:

```
NX(,) = INT(WX(,,1)/H1)
```

At this stage we assume that all the particles are still within range of their local PEs, i.e., that all the values in NX(,) are either 0, 1 or 2. We now calculate all the shifts to the North by 1 in parallel, followed by the shifts by 2. Because no data is being shifted out of range of the PP grid point it came from, there is no danger of accidentally overwriting any data. If we repeat this process with the y coordinates then we now have all the data from the first strontium PP plane on the correct PM grid points.

We can repeat this process for the rest of the matrices in WX(,,8), WY(,,8) and WZ(,,8) always filling four PM data planes at one time. Note that data for

the first three PM planes stored in WX(,,2) comes from the second PP plane whereas data for the fourth plane must come from the third PP plane (see fig. 3.2).

We now look at the problem of mapping the particle coordinates from the PP mesh into the positions shown in fig. 3.11. First consider mapping of the x-coordinate data from the first PP xy plane onto these positions. Figure 3.13a shows the initial data-configuration on the DAP. To do the mapping we can use the long-vector shift facility on the DAP (See appendix I). Using this facility it is possible to achieve the mapping in 8 long vector shift operations. Figure 3.12 shows how the algorithm works. Once the data has been shifted into the top quadrant of the matrix this quadrant can be copied into the appropriate locations.

So far we have only looked at the case where particle data is stored on the PP cell whose volume contains the particle's position. However it is possible, especially when considering the strontiums, that even though the particle's data is stored on one PP cell the particle itself has moved into the a part of the crystal contained within a different PP cell. The problem that this creates is not that we could now get two particles trying to occupy the same cell, the Coulomb repulsion term ensures that two particles can never get too close, but rather that in one of the intermediate stages data from one particle can overwrite data from another. We solve this problem in two stages.

The first stage is to consider particles whose z coordinate is within range of the local PP cell but whose x and y coordinates are large enough to move them into neighbouring PP cells. We have to allow for values of NX(,) and NY(,), the arrays that give us the particle shifts in the x and y directions, that can range from -3 to 5 i.e for particles being on any of the nearest neighbour PP cells. (Note that a value of 0 in NX can now represent two different situations, values of 0.1 or -0.1 in WX both give values in NX of 0; we must be careful to check any value of 0 in NX to see whether the corresponding value of WX is positive, in which case the data can be left alone, or negative, in which case we must shift the data in the positive x-direction). We set up a logical matrix whose elements are set to .TRUE. at PM grid positions containing particle data. The entries in this matrix are kept fully updated; any time we shift the coordinates of a particle onto a new PM cell then the

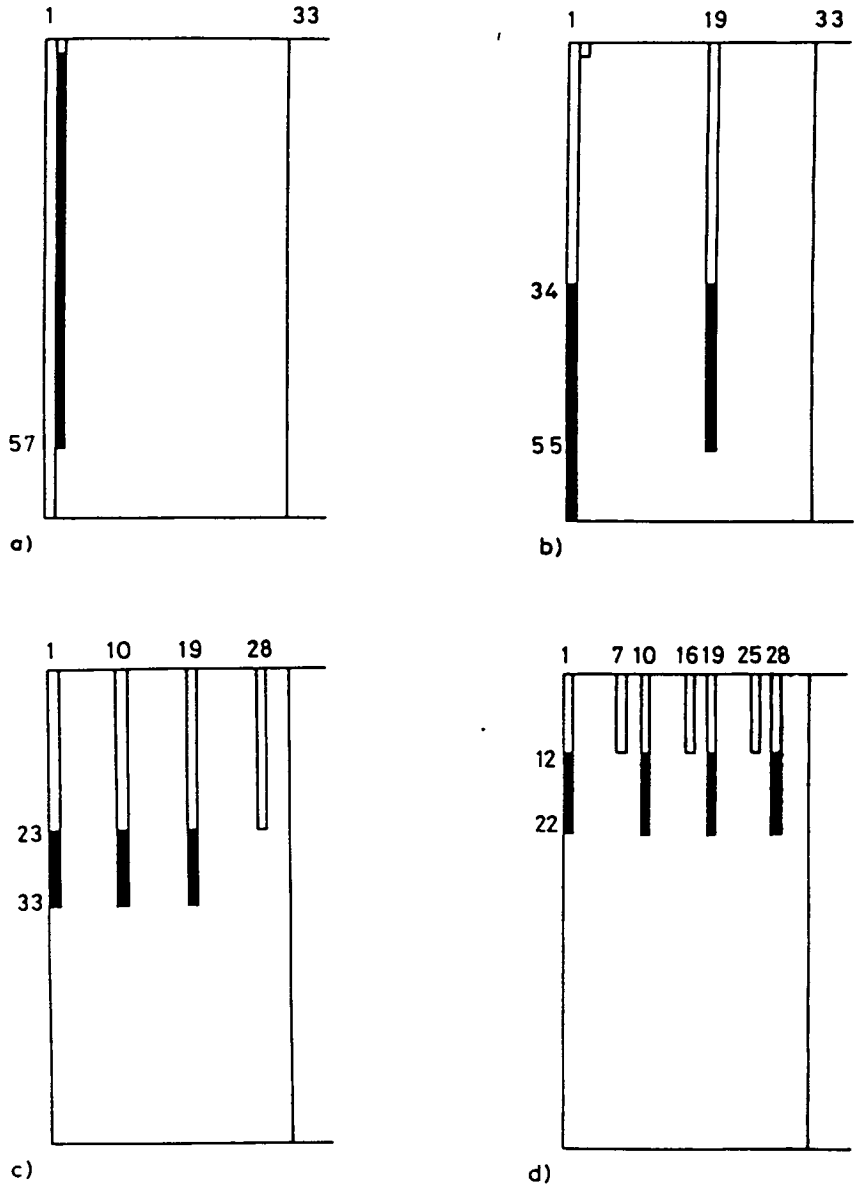
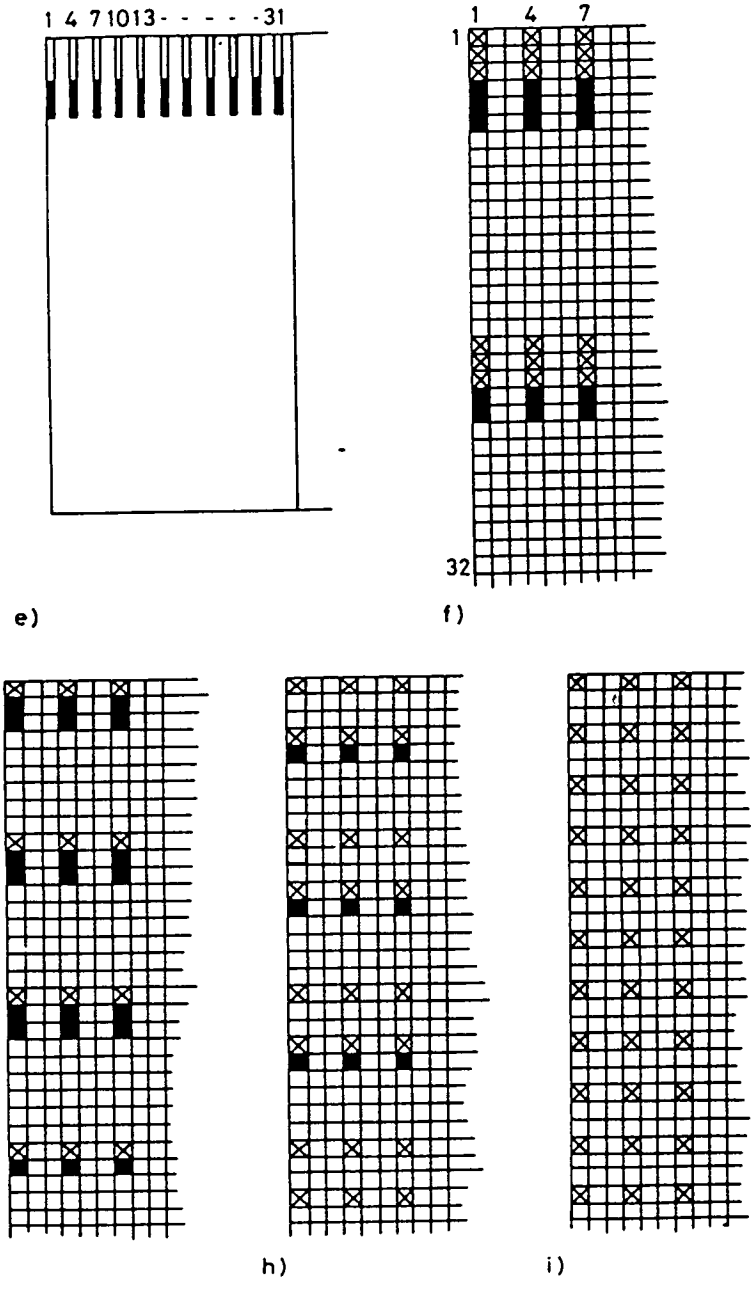


Figure 3.12 Mapping particle data from the PP cells to the PM cells containing the origins of PP cells (see figure 3.11)

Figure 3.12 a) shows the position of the x-coordinates from the first PP z-plane under the 64x64 grid of PEs (see also figure 3.3). We start by simaltaneously shifting the shaded data elements to the right by 905 places using the DAP long-vector shift facility (see Appendix A). Figure 3.12 b) shows the result of this operation. We shift the shaded elements in figure 3.12 b) to the right by 543 places to obtain the result shown in figure 3.12 c). The above process is repeated twice more, shifting the shaded data-elements in figures 3.12 c) and 3.12 d) to the right by 362 and 181 places respectively. The data is now in the positions illustrated in figure 3.12 e); all the data elements are now in the correct columns, i.e. all the data from the original PP plane is in the column it will occupy in the PM data-representation. We now begin the process of shifting the data onto the appropriate rows. For simplicity figures 3.12 f) to 3.12 i) show only the first three columns of particle data. We start by shifting the shaded elements in figure 3.12 e) to the South by 12 places. The



shaded elements in figure 3.12 f) are then shifted South by 6 places, in figure 3.12 g) by 3 places and in figure 3.12 h) by 1 place. At the end of this process we now have the PP data mapped the appropriate PM cells, figure 3.12 i). By using the above algorithm we can move the original 121 data elements from the PP to the PM data-representation in only 8 MERGE commands, the logical matrix in the MERGE command being used to mark the data-elements we wish to shift at each stage (see Appendix A). Note that as we have only used the first 32 columns of PEs in this algorithm we are free to use the remaining 32 columns to perform another mapping without adding any extra operations. In fact we can perform 4 mappings simaltaneously if we notice that for most of the algorithm we only use columns 1,4,7,10, etc. By carefully organising the data we can use the spare columns to do extra mappings, use the above algorithm data from four PP planes can be mapped onto the PM cells containing the PM cell origins using only eight DAP FORTRAN comands.

corresponding element of the logical matrix is moved to the same cell. Using this logical matrix we always know which of the PM cells contain particle data and which are empty. We proceed in the same way as before, shifting the data by the amounts specified in NX and NY, except that we now check to see that the locations that we intend to shift the data to are vacant before we actually move the data. We first move the data in the positive and negative x-directions (corresponding to shifting the data to the North and South on the PE grid) If the new location is empty then we move the data into this position, if the new location is occupied then the location one place to the East will always be empty (see fig. 3.11). Any shifted data that will overwrite existing data can be placed in these locations. As these particle have been shifted one place to the East we subtract H_1 off the corresponding element in NY. We now have all the particle data in the correct rows. Next we move the data onto the correct columns. Again we calculate NY(,) and loop over the different NY(,) values. At each stage we check to see whether the new location is vacant, if it is then move the new data into the vacant position. When, however, the new position is occupied then we move the data onto an overflow plane such that the particle data is stored on the correct position on this overflow data-plane. After we have looped over all the possible values of NY(,) and finished moving all the data either onto the WX, WY and WZ arrays or the overflow array, we can simply add the overflow array to the rest of the shifted data. We know from the physics that once particles are in the correct positions on the PM mesh then they cannot share the same position, by adding the arrays together we will not overwrite any data.

The second stage is to consider particles whose z-coordinates are out of range of their local PP cell. We map the data onto WX, WY and WZ in three stages. We first make a copy of all the entries whose z coordinates are within range of the local PE, these values are mapped onto WX, WY and WZ as described above. We then take a copy of all the data whose z-coordinate is greater than the top of the volume controlled by the local PP cell. We shift all the data on the copied plane up one PP plane and calculate the new particle positions with respect to the new PP cells. All the particle data on the copied plane is now on the correct PP plane, map the plane onto WX, WY and WZ as before. A similar technique is used for particles whose z-coordinate is out of range of the bottom of the local PP cells.

3.5.1.2. Calculation of the Charge Density

Once we have placed the particle coordinates in the correct positions with respect to the PM mesh we must then calculate the particle densities as defined by equation (3.15) i.e. for particles of unit charge we must calculate:

$$W(\mathbf{x}) = W'(x)W'(y)W'(z)$$

where:

$$W(x) = \begin{cases} .75-(x/H)^2 & |x| < H/2 \\ .5(1.5-|x|/H)^2 & H/2 < |x| < 3H/2 \\ 0 & \text{otherwise} \end{cases}$$

and x, y and z are the positions of the particles with respect to the bottom left hand corners of the PM cell that they have been mapped on to. We assign the particle densities to the 27 nearest and next-nearest neighbours around each particle's PM cell. The only problem that can arise in the calculation of the density is when two charge clouds overlap; we must be careful that the two distributions are added together where they coincide and that no data is inadvertently overwritten. The real space charge density is stored on the matrix array `WXYZ(,,8)`.

We start the calculation of the charge density by mapping the strontium coordinate data from the PP grid to the three arrays `WX`, `WY`, `WZ`. From `WX`, `WY` and `WZ` we can calculate the strontium charge density as given by equation (3.15) (remembering to multiply equation (3.15) by +2, the charge on the strontium). The arrays `WX`, `WY` and `WZ` are then set to zero. We next map the particle data from the first PP chlorine data plane (see section 3.2) onto the `WX`, `WY` and `WZ` arrays and then add the chlorine charge density (this time multiplying equation (3.15) by -1) to the array `WXYZ`. We repeat the process for all the PP chlorine data planes. At the end of this process we now have the PM TSC charge density for all the particles in the system stored on the array `WXYZ(,,8)`.

3.5.2. Calculation of the Electric Potential

As we have seen in equation (3.58) we calculate the electric potential by convoluting the charge density with the optimised Green function (equation (3.79)). This convolution is best done in reciprocal space as the convolution then becomes a simple product. We therefore Fourier transform the charge density, multiply the resultant array by the optimised Green function and Fourier transform the product back into real space to give the electric potential at all points on the PM mesh. We can calculate the optimal Green function outside the simulation program (using equations (3.58) to (3.84)) and read it in as input data for the simulation program. Three dimensional Fast Fourier Transforms (FFTs) can be done very efficiently on the DAP. An algorithm for doing the FFT has had to be specially developed as there were no multi-dimensional FFT programs available. The technique used is described in Appendix II. The values of the electric potential on mesh points is stored on the array `PHI(,,8)`.

3.5.3. Calculation of the Electric Fields

The values of the electric field components on the PM grid points are obtained by differencing the potential fields in the appropriate directions using equation (3.29). This calculation can be done completely in parallel. The array `EPOT` stores the values of one component of the electric field at the mesh points.

3.5.4. Calculation of the Forces on the Ions

3.5.4.1. Forces on the PM grid

Once we have the value of some component of the electric field at every point on the PM grid we can then begin to calculate the forces on the ions in the PM configuration. Consider, for example, the calculation of the x-component of the reference force, **R**, on the strontium ions.

As the force on the particles is calculated by interpolating forces from mesh points to particle positions we need to know the strontium ion positions with respect to the PM grid. We therefore map the strontium ion positions from the

PP grid onto the PM grid using the technique described in section 3.5.1. We now have the strontium coordinates stored on the 3 arrays WX, WY and WZ and the electric field in the x-direction stored on the array EPOT. We calculate the value of the force at particles by interpolating the forces on the PM mesh to the particle positions using the same interpolation function we used to map the charges onto the PM grid thus ensuring that the system will conserve momentum (see section 3.4.1). This prescription now gives us the forces on the particles stored on the PM grid point closest to the particle position in the array EPOT. We must now map the forces from the PM grid back onto the appropriate positions on the PP grid.

3.5.4.2. Mapping from the PM grid to the PP grid

First we consider the case of particles that are within range of their local PE, i.e. particles whose coordinates are such that they are stored in PM grid cells contained in the volume controlled by the PE on which they are stored. Consider force values stored on the first four planes of the PM mesh, i.e. in the matrix EPOT(,,1). From the way that the PM and PP grids are set up we know the data from these four planes will eventually end up on the first PP plane and be stored in the first two columns of the matrix we use to store the forces on the strontiums (see figures 3.2 and 3.3). If we can map the data from these particles back onto the PM grid points containing the origins of their PP cells, then we can use a similar technique to the one described in figures 3.12 to map the data back onto the PP grid (If we have done the mapping from the PP to the PM grid correctly then there is never any danger of overwriting data in this process, as we started with one strontium in each PP grid cell we must finish up with one strontium on every grid point with no overlaps on any of the intermediate steps). Figures 3.13 describe how this mapping can be achieved once the particle data has been moved to the PM cells at the origins of their local PEs.

Data from a particle on a particular PP grid point could be stored on any of the points on the PM grid contained in the same volume. To use the algorithm described in figures 3.13 we must find a way of moving data from anywhere in this volume onto the PM cell containing the origin of the associated PP grid cell. Start by constructing two logical matrices LUP(,) and LACROSS(,) (figure 3.14). Consider data stored on the first four PP planes, this data will all finish

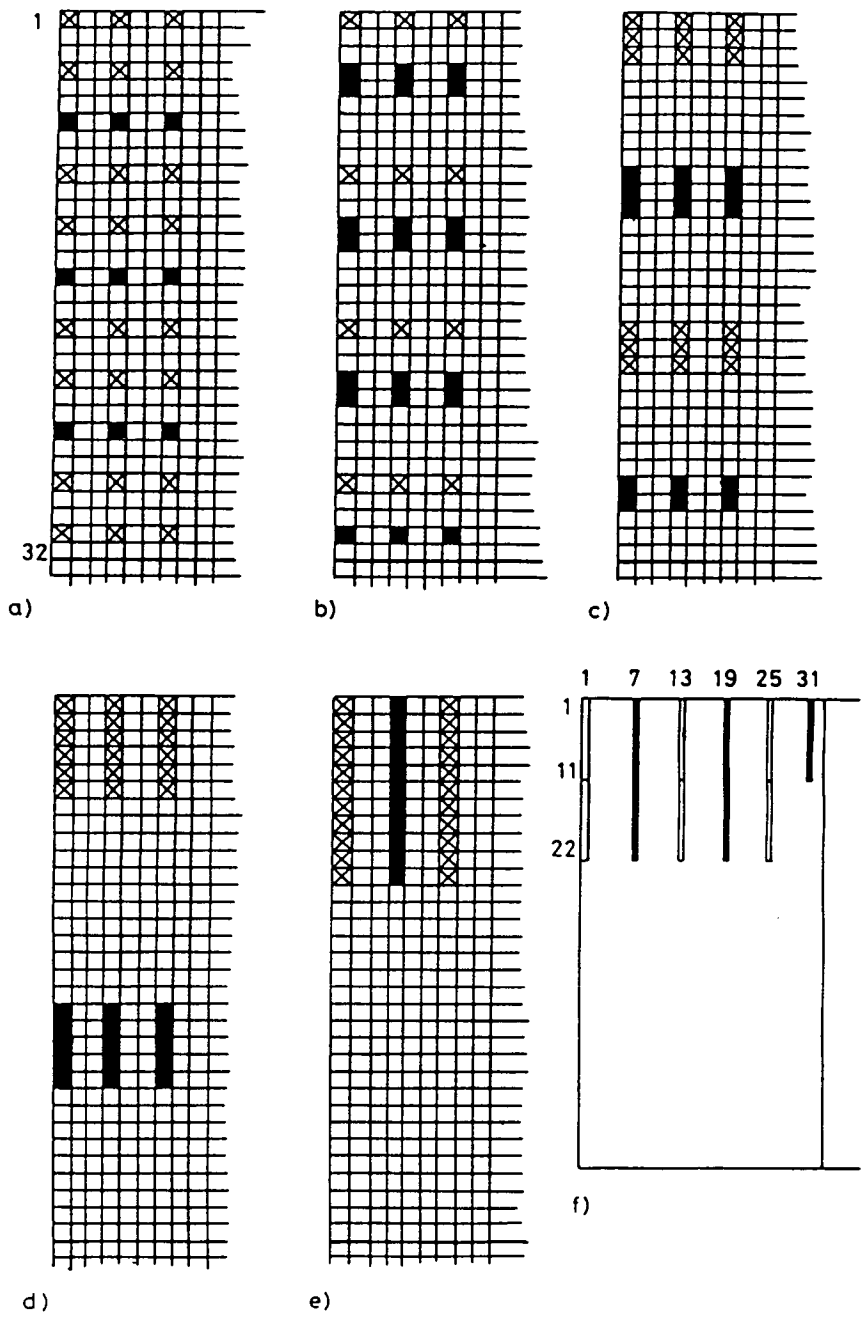
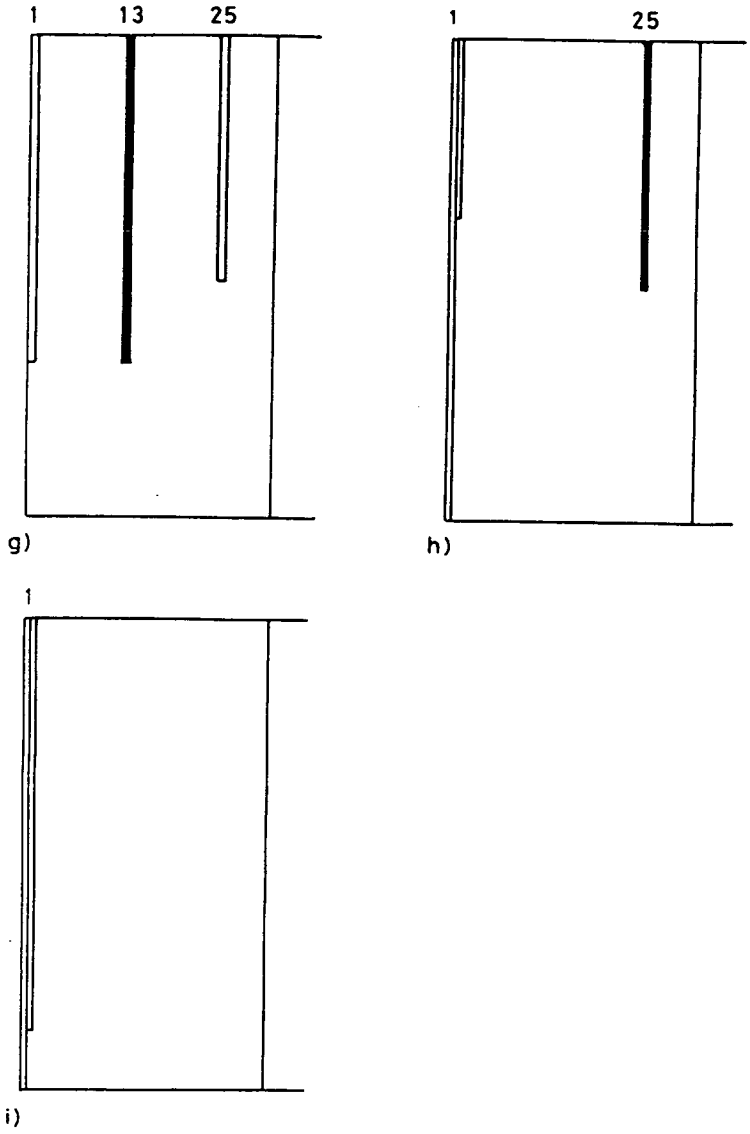


Figure 3.13 Mapping particle data from the PM cells containing the origins of PP cells (see figure 3.11) to the PP data-representation.

This algorithm is essentially the reverse of the algorithm described in figure 3.12. Figure 3.13 a) shows the starting positions of the particle data (crossed boxes) in the PM representation on the 64x64 grid of PEs. We start by gathering up the data into the first 11 rows of each column. We do this gathering in four stages. We start by shifting the shaded elements in figure 3.13 a) and moving them 2 places to the North to reach the position shown in figure 3.13 b). The shaded elements in figure 3.13 b) are then shifted 2 places to the North, by 6 places in figure 3.13 c) and by 12 places in figure 3.13 d). At the end of this process all the data is gathered into the first 11 rows, figure 3.13 e). We must now gather the rows together. We start by using the long-vector shift facility to move the data from the 4th column to the rows 12-22 on the 1st column, the data in the 10th row to rows 12-22 on the 7th column etc. to reach the position shown in figure 3.13f), this whole procedure



being done using on DAP MERGE command. We now shift the shaded elements in figure 3.13f to the left by 362, and then shift the shaded elements in figure 3.13g) to the left by 724 and the shaded elements in figure 3.13h) to the left by 1448 to reach the final position as shown in figure 3.13i). As before we have shifted 121 data-elements using only 8 MERGE commands. Similarly it is possible to map four PM planes from the PM to the PP data-representaation simaltaneously.

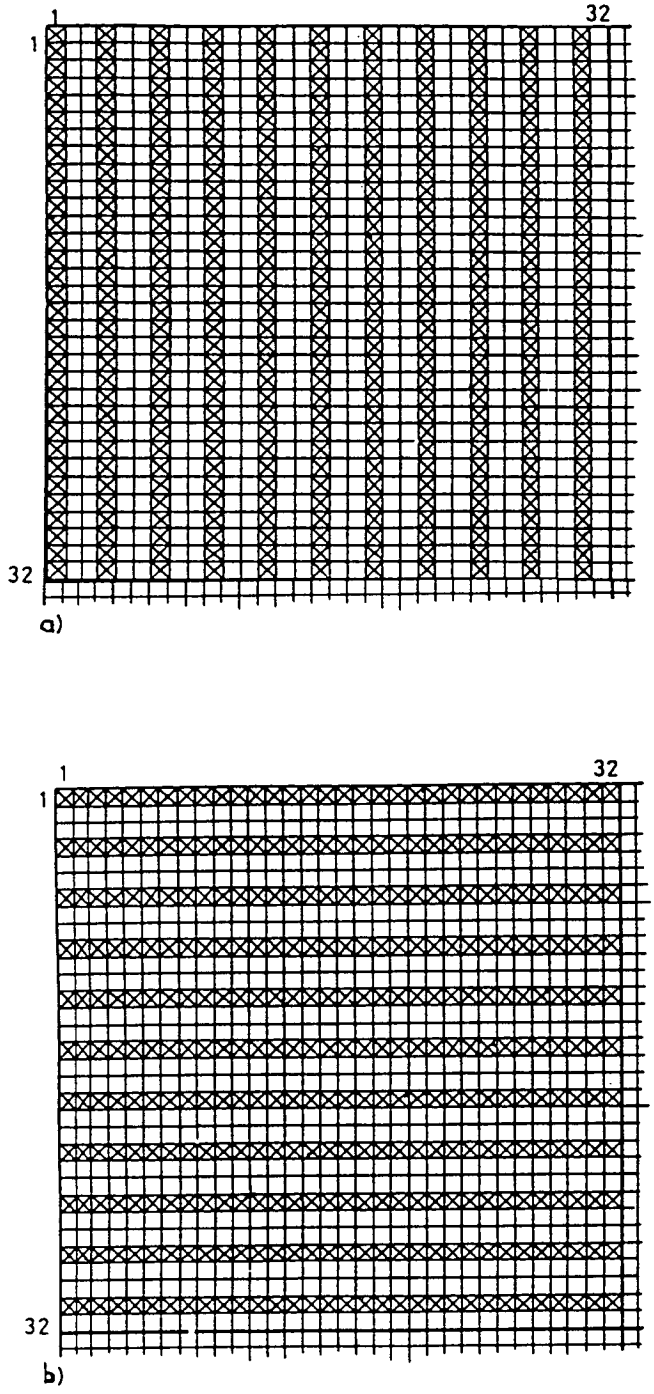


Figure 3.14 a) The logical matrix LUP(,) b) the matrix LACROSS(,)

up on the first PP plane. The entire process can be completed in four shift operations (no particle can be further than four shifts away from the PM cell containing the corresponding PP cell origin). The code to achieve this mapping is given below:

```
      LOGICAL LACROSS(,),LUP(,),L(,)
      REAL X(,)
C     SET L(,) TO BE .TRUE. WHERE X IS NON-ZERO
      L = X.NE.0.
C     MOVE ALL ELEMENTS UP TOWARDS THE LACROSS
C     ROWS, LEAVE ELEMENTS ALREADY
C     ON LACROSS ALONE, THERE IS NO DANGER OF
C     OVERWRITING ANY DATA
C     SHIFT ALL THE ELEMENTS UP BY ONE AND ADD
C     THEM TO THE ORIGONAL MATRIX X
      X = X + SHNC(X,1)
C     IF AN ELEMENT WAS ALREADY ON AN LACROSS
C     ROW IT WILL HAVE MOVED TOO FAR UP
C     AND ABOVE THE APPROPRIATE LACROSS ROW, ZERO SUCH ELEMENTS
      X(SHNC(LACROSS,1)) = 0
C     ZERO THE OLD PARTICLE POSITIONS
      X(SHSC(LACROSS,1).AND.L) = 0.
```

At the end of this process all the data on X will have been moved up to the nearest LACROSS row. If we repeat the above process using LUP instead of LACROSS and shifting in the East and West directions then we can move all the data onto the correct columns. If we then zero all the elements of X not on the intersection of LUP and LACROSS:

```
      X(.NOT.(LACROSS.AND.LUP)) = 0.
```

then we find that we have shifted all the data on X onto the PM grid cells containing origins of the PP cell controlled by the particle's local PEs.

A similar technique can be applied to all the matrices in the array EPOT so that eventually we can map the x-forces on the strontium ions from the PM data representation to the PP data representation.

The technique described above only works for particles that are still in the volume controlled by the PEs which contain their positional data. However it is possible for a particle to move out of range of its local PE, for example a chlorine ion that wants to move into the volume controlled by a different set

of PEs but can not because the new PEs already store the data for four chlorines (see section 3.2). We must find a way of mapping such data back onto the PEs that originally contained the particles coordinates in the PP data-representation and not onto the PEs controlling the PP cell containing the particles actual position. If we can find a way of tagging the force data in the PM representation from particles not in range of their local PEs, and if we can find a way of moving the force data from such particles from their present positions on the PM grid to grid points contained in the volume controlled by their local PEs without overwriting other data in the process, then we can use the above algorithm to map the force data back into the appropriate positions in the PP representation.

It is relatively straightforward to tag the force data corresponding to particles that have moved out of range of their local PEs. For simplicity we shall work in units such the the PM grid spacing is unity in all directions. As in the previous mapping from the PP to the PM data-representation we will deal with the problem in two stages; we will deal the particle's x and y coordinates separately from their z coordinates.

First we deal with particles whose z coordinate is such that the PM cell containing their data is mapped onto a PP cell on the same x-y plane as the PP cell containing the particle's coordinates in the PP data-representation. For this type of particle we must consider five possibilities for the position of the particle on the PM mesh. First there will be particles whose PM cell will map directly onto correct PP cell, then there are particles whose x coordinate is too large for their local PE, particles whose x coordinate is negative and similarly for the y coordinates and all the permutations of these cases. The arrays `WX`, `WY` and `WZ` contain the positions of all the strontium ions stored on the correct PM grid cell (x coordinates on the array `WX`, y coordinates on the array `WZ` etc.). We can examine these three arrays to look to see which of the above categories each of the particles belongs to. We can use these arrays to construct logical masks which are set .TRUE. at positions on the PM mesh where, for example, particles have an x coordinate which is larger than the maximum x coordinate of their local PP cell. We move all force data values which correspond to particles on the PM mesh having x coordinate greater than 3 by 3 grid cells in the -x direction. This should ensure that the force data is stored on a PM mesh cell which will map onto the correct PP cell.

Before moving the data we check to see if the new location already contains force data, if it does then we shift the data to an empty neighbouring PM cell (as there are at most 11x11x10 particles on the 32^3 mesh at any time there will always be an available empty site). We can repeat this process for particles with negative x coordinates, negative y coordinates etc. At the end of this process we have all the particle force data stored on PM cells which map onto the appropriate PP cells. As before there will only be one force value stored on the PM cells corresponding to one PP grid cell by virtue of the fact that we started with one particle in every PP cell and when we reverse the mapping process we must finish with one data value in every PP cell. We can now apply the mapping algorithm described at the beginning of this section to map the force data from the PM data-representation onto the PP data-representation.

Once we have mapped the force data into the PP representation for particles whose z coordinate is within the range of their local PE we can consider mapping data from particles with either a negative z-coordinate or a z coordinate which is greater than the z coordinate of the top of their local PP cell. If we take particles with a negative z coordinate to start with, then we can map them back onto some DAP matrix by assuming that their local PP cell is the one directly below their real PP cell (this will ensure positive values for the z coordinate). We will have a set of force values in the PP representation which are on the PP plane below the correct plane. We can place the force data in the correct position by shifting it all up one PP plane in the z direction. This data can now be added to the previously calculated force data. A similar technique can be used for force data corresponding to particles with z coordinates which are out of range of their local PEs.

At the end of this process we have calculated the reference force **R** on all the strontiums in the PP data-representation, by repeating the above stages we can similarly calculate the y and z forces on the strontiums and the forces on the chlorines on all four chlorine data-planes.

3.5.5. Application of the PM algorithm to the simulation of $SrCl_2$

The PM algorithm as implemented on the DAP consists of three main stages. If the DAP matrix `STRONTIUM(,)` contains the data for the positions of the

strontium ions stored in the PP data-representation (see figure 3.3), the matrix array CHLORINE(,,4) contains the data for the positions of the chlorine ions and the matrices STRFORCE(,) and CHLFORCE(,,4) store the forces on the strontium and chlorine ions respectively, then the steps needed to calculate the reference force **R** on all the ions are given below.

First we calculate the charge density for the spherically distributed 'smeared' charges on the 32^3 PM grid. This is done by taking the strontium positional data in the PP data-representation as stored on the array STRONTIUM and transferring it to the relevant grid points on the PM grid (the array WX(,,8) storing the x data, WY(,,8) storing the y data etc.) Once we have the strontium positions stored in the PM data-representation we can use this information to calculate the charge density for the strontium ions. These values are stored in the array WXYZ(,,8). This process is then repeated for each of the four chlorine data-planes, always adding the new charge density to WXYZ(,,8). At the end of this process we have calculated the charge density of the configuration (sections 3.5.1.1 and 3.5.1.2).

Secondly we calculate the PM defined electric potential at every point by convoluting the charge density with an optimised Green function. The Green function used is pre-calculated and read in at the start of the run. The convolution is performed by Fourier transforming the charge density, multiplying it by the transform of the Green function and Fourier transforming the product back into real space. The value of the electric potential at mesh points is stored in the array POT(,,8) (section 3.5.2).

Finally we calculate the forces on each all of the strontiums and chlorines in turn. We start by calculating the force on the strontium ions. We map the positions of the strontium ions from the matrix STRONTIUM(,) to the matrix arrays WX, WY and WZ. We then calculate the electric field in the x direction on each of the mesh points. This data is stored in the array EPOT(,,8). Using the particle coordinates stored in WX, WY and WZ these values can then be interpolated from mesh points to particle positions using the same interpolation function we used to map from particle positions to mesh points (section 3.5.3). Once we have the forces in the x direction on the PM mesh these values can be mapped back onto the array STRFORCE(,) which stores the value of the forces on the strontiums in the PP data-representation

(section 3.5.4). Once we have calculated the forces on the strontiums in the x direction we calculate the value of the electric field in the y direction and repeat the above process to give us the y forces stored on the array STRFORCE. The entire process is then repeated with the four chlorine data planes CHLORINE(,,4).

This algorithm gives the reference force **R** on all the particles in the simulation. However we must remember that **R** is not the Coulombic force between point charges, but the Coulombic force between spherically symmetric charge distributions. For particle separations large enough such that the charge distributions do not overlap then **R** is equivalent to the Coulombic force between point charges, for particles close enough together to allow the charge distributions to overlap then **R** will not be equivalent to the point charge Coulombic force. For particle separations smaller than a, the radius of the charge distribution, we must calculate a short-range correction term which we can add to the short-range forces (the electron shell repulsion and Van der Waals terms) so as to convert **R** into the correct form.

3.5.6. Short Range Corrections to the Reference Force

To get correct the form for the Coulombic force for closely separated particles we must subtract off the value of reference force **R** that we calculated via the PM algorithm and explicitly add in the $1/r^2$ Coulombic term. In order to do this we must identify particles whose separation is less than a, the diameter of the particle charge distribution. Provided that a is less than r_e, the cut-off radius of the Van der Waals and electron-shell repulsion forces (section 3.3), then we have already solved the problem of identifying such particles; we can include the corrections to the PM force in the routine that calculates the short-range forces. When we calculate the Van der Waals and electron shell repulsion forces we can add in the true Coulombic force and subtract off the reference force **R**. Equation (3.69) gives the analytic form for **R**.

As we must calculate the short range force terms for many thousands of interactions every timestep, it is not computationally efficient to calculate a seventh order polynomial in the innermost loop of the force calculation. On a serial computer one could store the value of the polynomial at several hundred points as a look-up table, this being a very much faster operation

than calculating the seven multiplies and seven adds needed to calculate the polynomial. However on a parallel computer a simple look-up table technique is not practicable, whereas it is possible to look up entries in a table one at a time it is not possible to look at many entries simultaneously. For example, consider calculating the value of the reference force on the chlorines stored at the same PEs on the matrices CHLORINE(,,1) and CHLORINE(,,2). When we calculate the distance between such chlorines we will obtain a matrix containing 1210 different values of the particle separation. On the DAP it would be very inefficient to calculate the value of R(r) for each entry in the matrix in turn, we must find an algorithm that will let us find the value of the force for many different r values simultaneously. The technique used is to loop over the different sets of r values rather than over the different r entries.

As the value of R should be accurate to 1%, we calculate a set of values of r, (r^n), such that $|R(r^{i+1})-R(r^i)|/|R(r^i)|<.01$ for $1<=i<=n$. These values of r along with the corresponding values of R are then stored in two vectors, RDATA(200) and FDATA(200) respectively. Once we have calculated these two vectors then the code to calculate the value of R(r) to 1% accuracy for any value of r between 0 and r_e is simply:-

```
        DO 1 I = 1,200
         FSRCOUL((RMOD.GT.RDATA(I)).AND.(RMOD.LT.RDATA(I+1))) = FDATA(I)
1       CONTINUE
```

where RMOD is the matrix storing the values of r and FSRCOUL stores the corresponding values of R. Note that this routine requires no algebraic operations, only logical manipulations and data assignment. Even though we require a 200 step DO loop the code still runs many times faster than the code to calculate a seventh order polynomial.

3.6. Program Tests and Timings

The program used to do the MD simulations is described schematically in appendix III. The test procedure for the program consists of three stages; first we must check that the code treating the mobile ions works as intended (this can be checked by inspection), secondly we must then check that the routines to calculate the interparticle forces are giving us the correct forces (especially

the PM part of the force calculation) and finally we must check that the entire MD program is working correctly.

3.6.1. Testing the Force Calculation Routines

We first test the simulation with only the short-range force routines, neglecting the long-range Coulombic force part of the program for the time being. Although it is highly unlikely that the crystal will be stable without these long-range force contributions, we can use this restricted simulation to test that the coordinate updating routines, the short-range force routines and the mobile ions routines when put together produce a system whose centre of mass and total energy are both stable.

We must next test the part of the program that calculates the Coulombic forces. This is done in several stages. First we test the routines that shift particle data from the PP mesh to the PM mesh and vice versa. Testing the data shifting routines is once again done by inspection. After checking that the mapping routines are working satisfactorily we check that the PM algorithm correctly calculates the force between two ions and conserves the total momentum of the system. This can be done by placing two ions on the mesh and comparing the values of the force obtained via the program with the analytic values. The values for the force between the two ions as calculated by the program was found to be within 1% of the analytic value.

The final tests to check that the entire program is working correctly are described in the next chapter.

3.6.2. Program Timings

One complete MD timestep takes 54s to calculate on the DAP. The majority of this time is taken up in calculating the short-range inter-particle forces, the time taken to calculate the short-range forces being 39s and the time taken to calculate the reference force being 15s. The time required to update the particle coordinates is insignificant compared with the time required to calculate the forces.

In this chapter the MD simulations of the fluorite structure superionic compounds are described and the results from these simulations are discussed.

4.1. Description of the Simulation Runs

MD simulations have been performed on three different fluorite structure superionic compounds; $SrCl_2$, CaF_2 and PbF_2. The interatomic potentials used in these simulations have been described in chapter 3, the MD simulation technique has been described in chapter 2 and the method used for calculating the Coulombic forces has been described in chapter 3. Several different MD runs were performed for each of the three superionic compounds at temperatures both above and below the superionic transition temperature.

For each of the MD runs the centre of masses of the 1210 cations and 2420 anions were placed on the equilibrium fluorite lattice sites. These ions were then given random velocities using values chosen from a Gaussian distribution. The equations of motion for the system were then integrated forward in time using the Beeman algorithm with an initial timestep, δt, of 0.01ps. For the first few hundred timesteps the particle velocities were rescaled so as to give a total kinetic energy equivalent to the desired temperature of the system (calculated via the equipartition theorem). After these first few hundred timesteps the velocity rescaling was switched off and the system was allowed to evolve for another few hundred timesteps with δt=0.0075. In order to ensure that the simulation had equilibrated properly the size of the temperature fluctuations was monitored from the point that the velocity rescaling was turned off, the system being defined to be in equilibrium when the size of the temperature fluctuations reached a constant value. Unfortunately it was not possible to use the function $\varepsilon(t)$ described in chapter 2 to determine that the simulation was ergodic and in the micro-canonical ensemble. In using the PPPM algorithm to calculate the Coulombic forces we cannot accurately calculate the Coulombic potential

energy of the system and therefore have no accurate value for the total energy of the system to substitute in the expression for $\mathcal{E}$ (t).

No data was collected from the simulation during the equilibration period. Once the system had equilibrated it was run for several hundred more timesteps during which time data on the ionic positions was periodically written onto disc for later analysis. Because of the large size of the system it proved impractical to write the coordinates onto disc every timestep, the disc would have been completely filled before the simulation had time to move forward any significant amount in simulation time. As a compromise data from the simulation was written onto disc every 14 timesteps (equivalent to writing out the data once every 0.105ps) for 672 timesteps (equivalent to 5.04ps in simulation time). As typical hopping times have been reported to be of the order 0.6ps, this should be a perfectly adequate data-density.

Table 4.1 gives details of the MD runs performed. The temperature quoted in this table is the average temperature of the simulated system over the period in which the data was collected. Note that every simulation was started with the ions in equilibrium positions. This was done to ensure that at low temperature, where the defects can have lifetimes of many picoseconds, we were not simply watching the same defects at different temperatures in the various simulations.

Table 4.1 Description of the Runs

$SrCl_2$		PbF_2		CaF_2	
Temp	Timesteps	Temp	Timesteps	Temp	Timesteps
770	1092	417	1092	852	1072
823	1240	571	1012	975	1120
928	1012	781	1800	1222	1046
1084	1072	841	1582	1366	1052
1216	1072	894	1112	1495	1018
1341	1092				
1438	1072				
1525	1072				

The lattice parameters (equivalent to the cation-cation nearest neighbour distances) used in the simulations where 7.23A for the $SrCl_2$ simulations, 6.03A for the PbF_2 simulations and 5.712A for the CaF_2 simulations.

The data from the DAP MD simulations is not in a particularly useful format for use in any analysis programs. It is possible for an anion in one of the PP cells to be closest to any one of 16 anion lattice sites (figure 4.1), making it difficult, for example, to spot anions jumping between different lattice sites. To make the job of analysing the data easier it is more convenient to calculate the anion positional coordinates with respect to the anion lattice site closest to the ions, rather than storing their positions with respect to the origin of some arbitrary box in space (see section 3.2). In order to do this we need some way of defining the equilibrium lattice sites in the simulated solid. Since the cations do not diffuse we can use their average positions over the simulation run to define an underlying regular lattice. Using this underlying lattice we can then define the anion lattice sites to be at the centre of mass of the appropriate cation tetrahedron. Using these lattice sites it is then possible to produce two arrays, one containing the cation positions with respect to the nearest cation lattice site and the other containing the anion coordinates with respect to the nearest cation lattice site. The anion data array also contains an identification number for each of the anions (using the identification numbers it is possible to follow the path of an individual anion as it diffuses through the lattice). Each MD simulation run produces 48 such arrays, one for every 14 timesteps of the data-collection runs (equivalent to one set of arrays every .105ps), and it is these arrays which are used for the subsequent data analysis. The analysis of the configurations generated by the DAP was performed using an ICL 1972 and an Amdahl V/7 mainframe.

Before we start any detailed examination of these configurations we need to know whether the simulated model systems are showing the required superionic behaviour, and also whether the structure of the crystals obtained using the PPPM algorithm to model the Coulombic forces corresponds to the structures obtained from similar simulations of smaller systems using the Ewald algorithm for the calculation of the long-range forces.

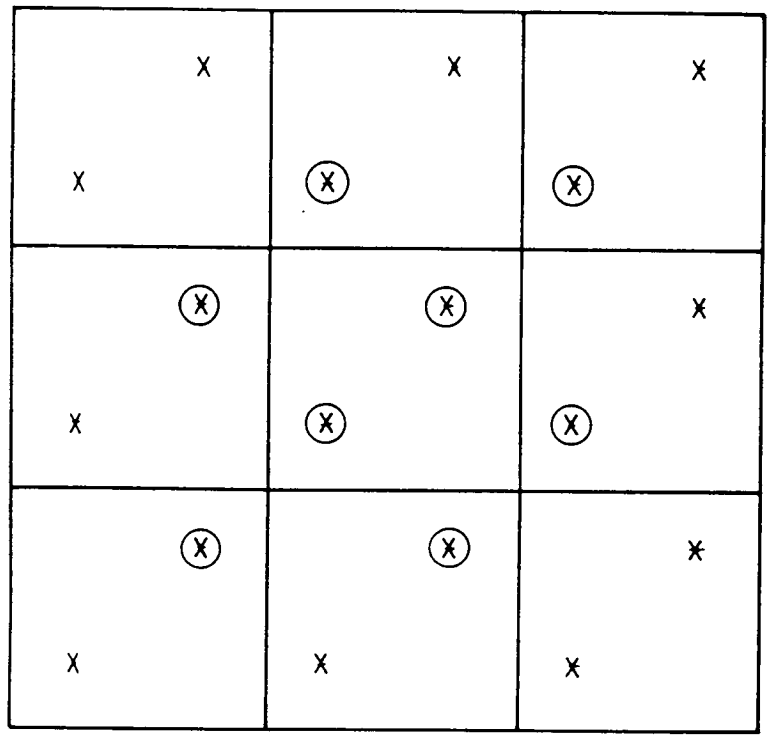


Figure 4.1 If an anion is in the centre PP cell then it could be closest to any one of 16 anion lattice sites; the 8 sites in the same xy plane of PP cells (the circled sites in the diagram) and the equivalent 8 sites in the plane below.

4.2.1. Calculation of the Self-Diffusion Coefficients

A superionic conductor is a compound which exhibits significant diffusion of at least one of its sub-species whilst still in the solid phase. In order to investigate the simulations to determine whether they are exhibiting superionic behaviour we therefore need to calculate the self-diffusion coefficients of the various ionic species in the compounds. The self-diffusion coefficient for species α can be conveniently calculated by examining the time-dependant mean square displacement, $<r_\alpha(t)^2>$. The self diffusion coefficient can be obtained from $<r_\alpha(t)^2>$ using the relation:

$$<r_\alpha(t)^2> \rightarrow B_\alpha+6D_\alpha|t| \tag{4.1}$$

for large $|t|$ (Sangster and Dixon 1976) where D_α is the self diffusion coefficient for species α and B_a is a constant.

The quantity $<r_\alpha(t)^2>$ is measured in the simulation by calculating the average distance travelled by each of the ions travelled in a time t. As we only have 48 configurations, each separated in time by .105ps we can only calculate $<r_\alpha(t)^2>$ for values of t which are multiples of .105ps and for t less than 47*.105ps.

$$<r_\alpha(t)^2>=(1/N_\tau)\sum_i (1/N_\alpha)[r_{i\alpha}(t+\tau)-r_{i\alpha}(\tau)]^2 \tag{4.2}$$

where t and τ are both multiples of .105ps, N_α is the number of ions of species α and N_τ is the number of time origins used ($N_\tau=(48-(t/.105))$.

Figure 4.2 shows the graphs of $<r_\alpha(t)^2>$ calculated for all the runs described in table 4.1. It can be seen from the graphs that the gradient of the cation mean-square displacement curves is always zero. This means that the cations do not diffuse through the lattice, even at the highest temperatures used in the simulations. Therefore, as the cation lattice must still be well defined, the simulated crystals are in the solid phase. It can also be seen from the graphs that by increasing the temperature of the simulated crystals we go from a regime in which the anions do not diffuse into a regime in which there is

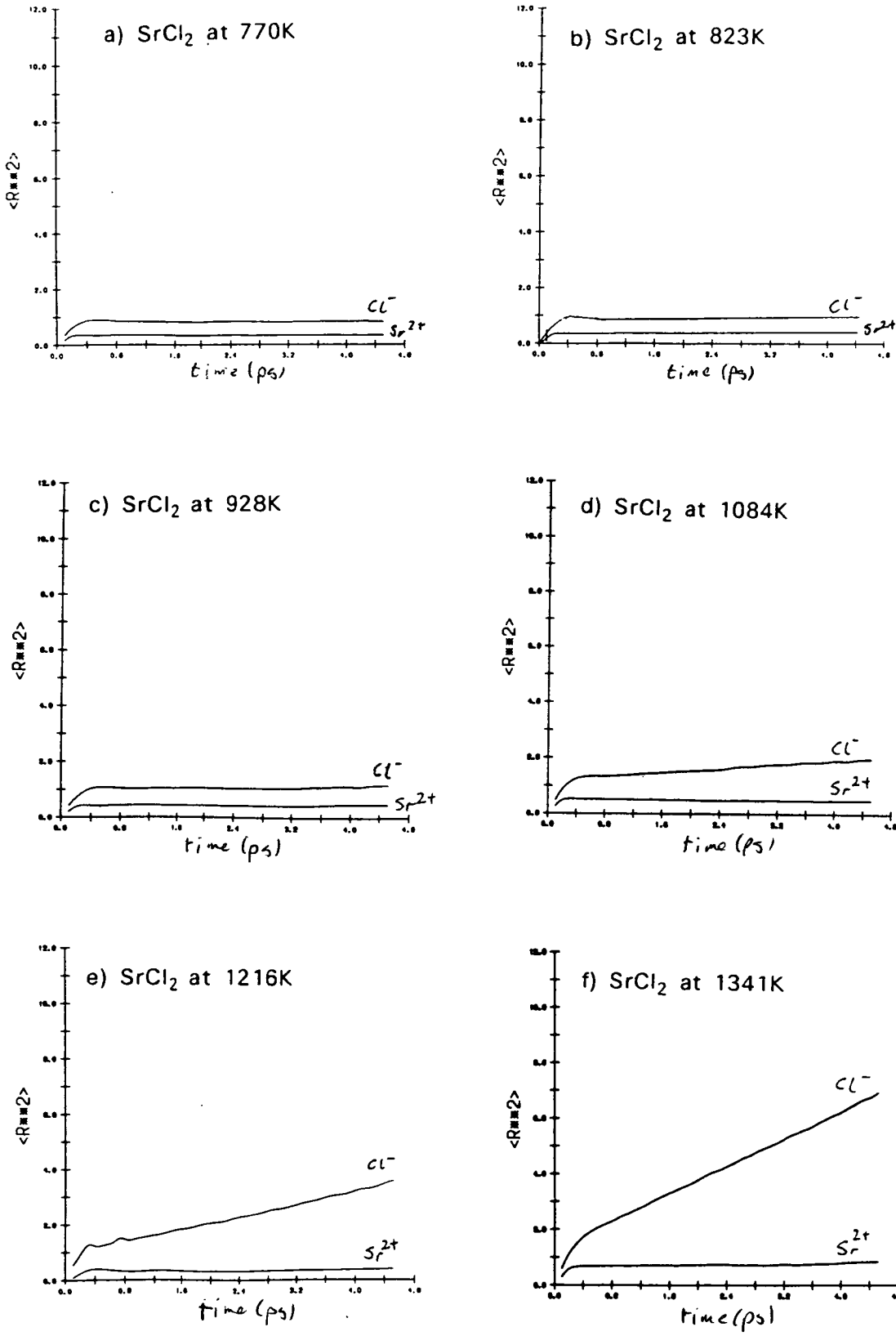
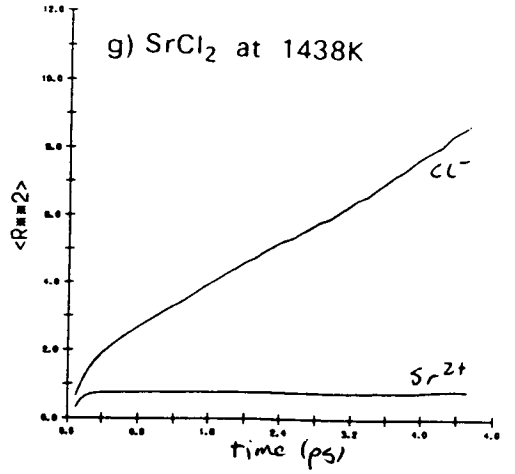
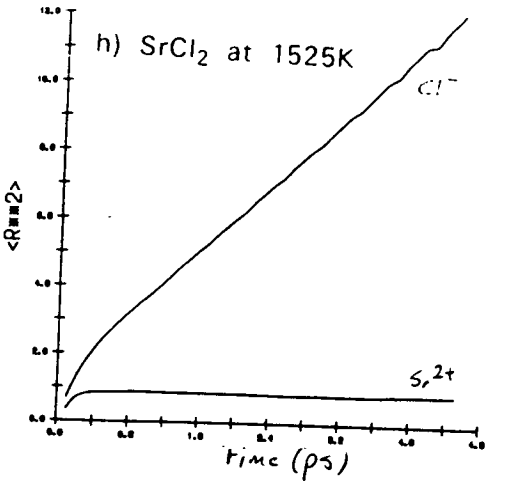
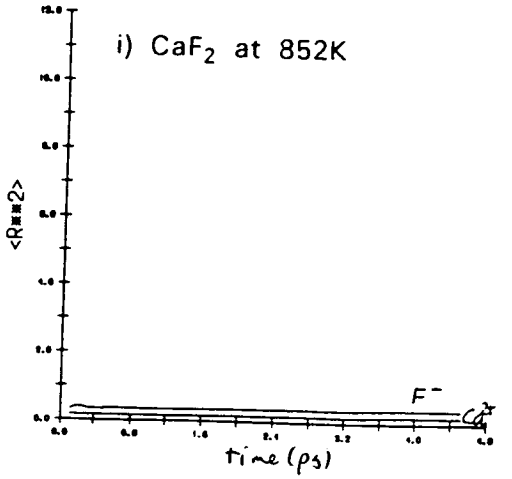
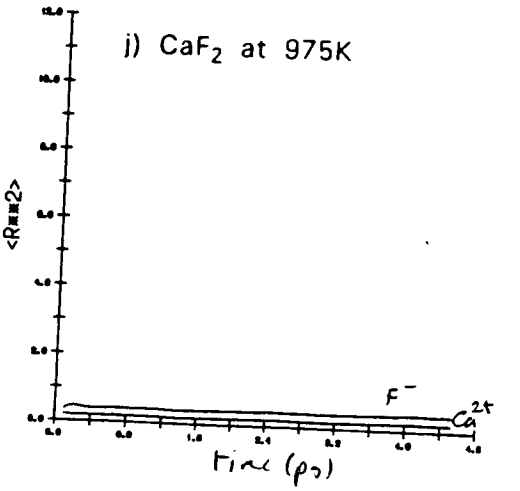


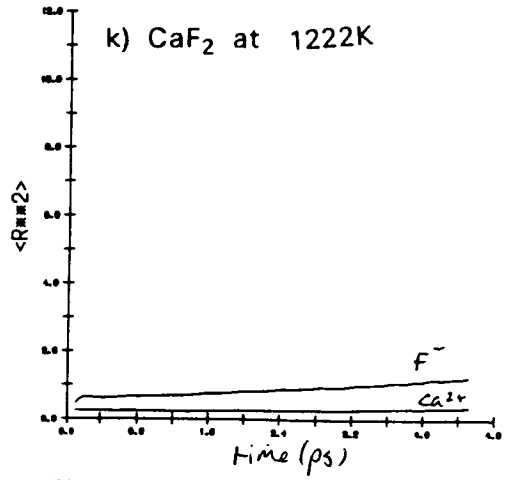
Figure 4.2 The mean square displacement, $<r_\alpha(t)^2>$ plotted versus time for each of the simulation runs described in table 4.1.

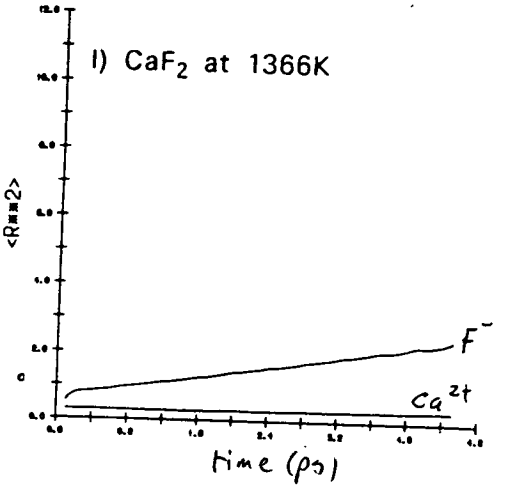

g) $SrCl_2$ at 1438K
<Rmm2>
Cl^-
Sr^{2+}
time (ps)

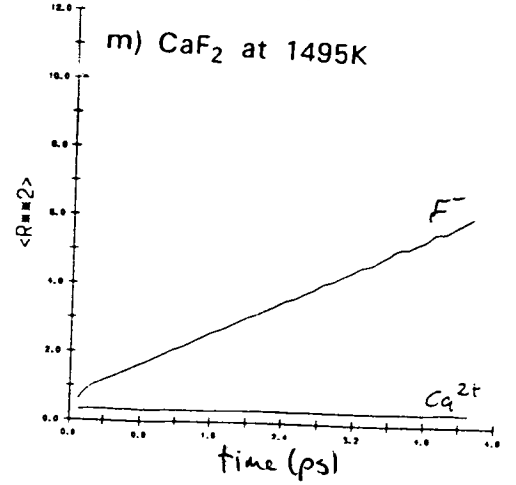

h) $SrCl_2$ at 1525K
<Rmm2>
Cl^-
Sr^{2+}
time (ps)

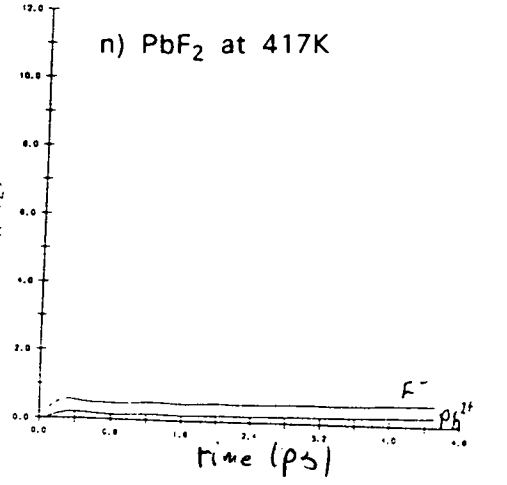

i) CaF_2 at 852K
<Rmm2>
F^-
Ca^{2+}
time (ps)


j) CaF_2 at 975K
<Rmm2>
F^-
Ca^{2+}
time (ps)

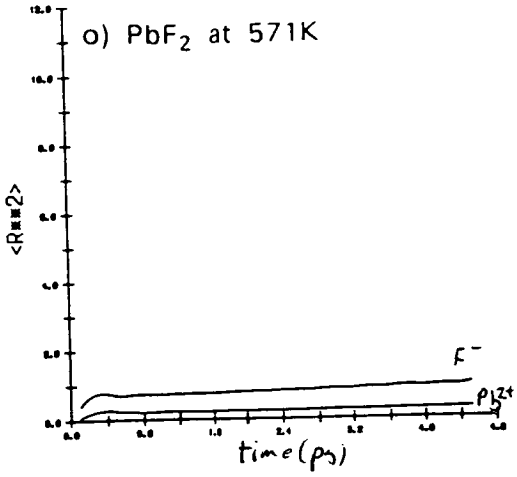

k) CaF_2 at 1222K
<Rmm2>
F^-
Ca^{2+}
time (ps)


l) CaF_2 at 1366K
<Rmm2>
F^-
Ca^{2+}
time (ps)

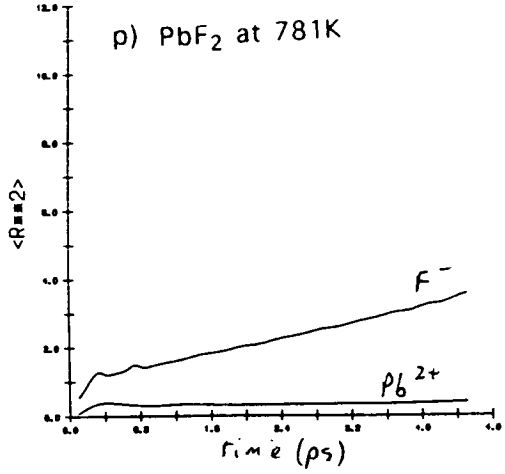

m) CaF_2 at 1495K
<Rmm2>
F^-
Ca^{2+}
time (ps)


n) PbF_2 at 417K
<Rmm2>
F^-
Pb^{2+}
time (ps)

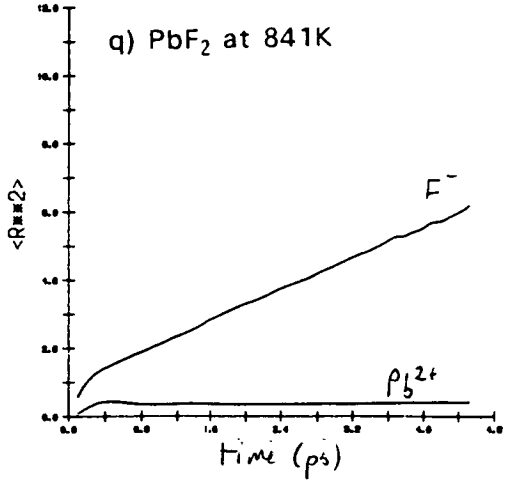
o) PbF_2 at 571K



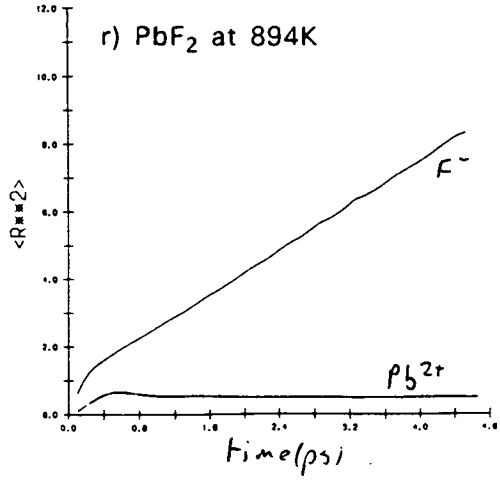
p) PbF_2 at 781K



q) PbF_2 at 841K



r) PbF_2 at 894K



significant anion diffusion. The values of D_α can be calculated from the gradient of the graphs and can be compared with the values of D_α obtained in other MD simulations of fluorite structure superionics. We can also use the Nernst-Einstein relation (Corish and Jacobs 1973) to convert the experimentally measured electrical conductivities, σ, of the fluorite structure superionics into values for the self diffusion coefficients, D. The Nernst-Einstein equation relates D to the electrical conductivity via:

$$D=(fk_B/Ne^2)\sigma T \qquad (4.3)$$

where N is the number density of the conducting ions and f is the correlation factor which is taken to be approximately unity (Corish and Jacobs 1973, Hansen and McDonald 1976). In figure 4.3 the values of D obtained in the present simulations are compared with those obtained in other MD simulations and those obtained experimentally from measurements of σ. Examining figure 4.3 it can be seen that the values of D obtained in these simulations are in good agreement with the values of D obtained in other simulations. As the present simulations are of systems an order of magnitude larger than those used in previous simulation work, these results suggest that we should not expect significant new conduction mechanisms to be present in the simulations on the large system. Examining figure 4.3 we can see that the values of D obtained from the simulations follow the same trends as those obtained experimentally. Note that the values of D obtained from the simulation of $SrCl_2$ show an upward shift of about 250K compared with the the experimental results. This shift can be explained as being the result of the Frenkel defect formation energy obtained using the interionic potentials used in the simulation being higher than those obtained experimentally (Gillan and Dixon 1978). If we rescale the simulation temperature by 250K then the values of D obtained from experiment and simulation are in reasonable agreement. The values of D obtained from the simulation of CaF_2 seem to be in very good agreement with experiment, probably because the defect Frenkel formation energy obtained using the interionic potentials used in the simulation is the same as the formation energy as measured experimentally (see table 3.1). The simulation results for PbF_2 show an upward shift of about 10% from the experimental results, again this can be put down to the Frenkel defect formation energy being 12% higher in the simulation than has been measured

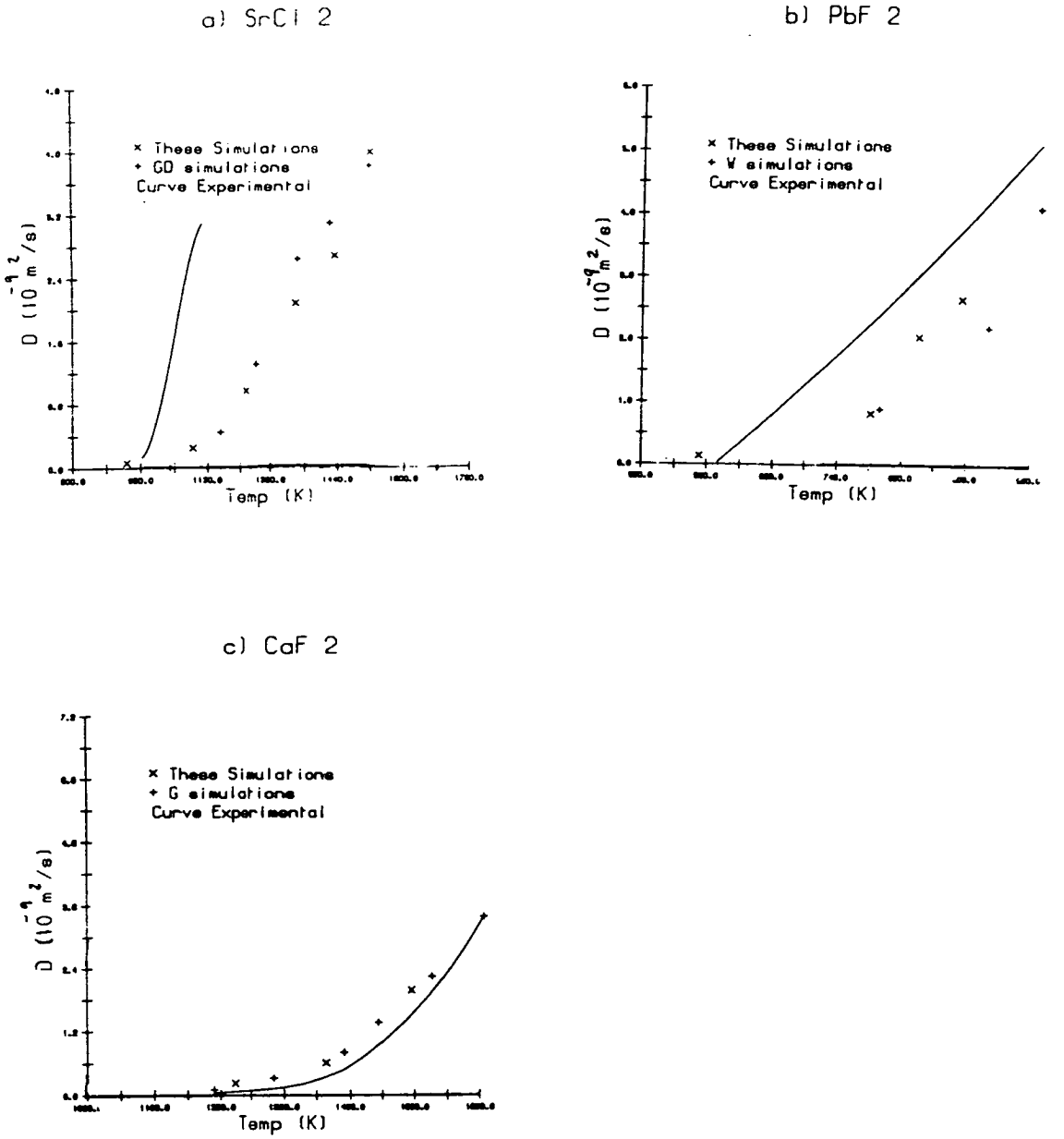


Figure 4.3 A comparison of the values of D obtained from the prsent simulations with those obtained in other MD simulations and from experiment.

a) $SrCl_2$. The other MD results are from the simulations of Gillan and Dixon (1979) and the experimental results are from Carr *et al* 1978.

b) PbF_2. The other MD results are taken from the MD simulations of Walker *et al* 1982 and the experimental results are taken from the NMR studies of Gordon and Strange (1978).

c) CaF_2. The other MD results are taken from Gillan 1986. The experimental results are taken from Derrington *et al* 1975.

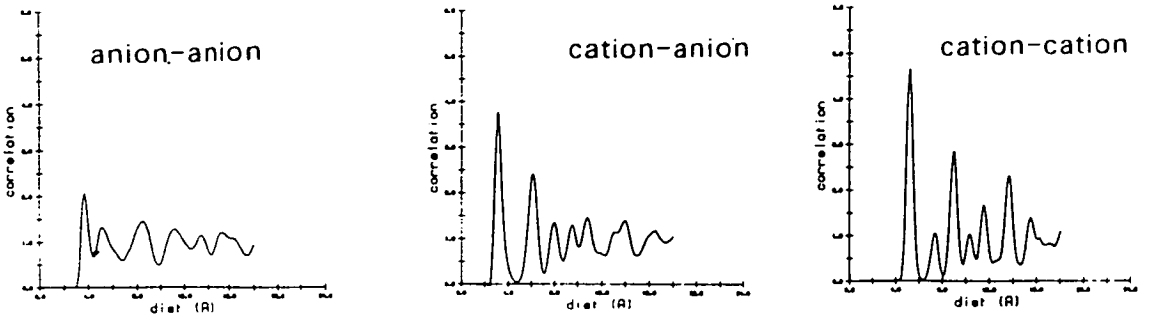
experimentally. Allowing for the differences between the Frenkel defect formation energies between the simulations and experiments we can see that the simulated systems are a good model for the superionic behaviour in fluorite structure superionic crystals.

4.2.2. Radial Distribution Functions

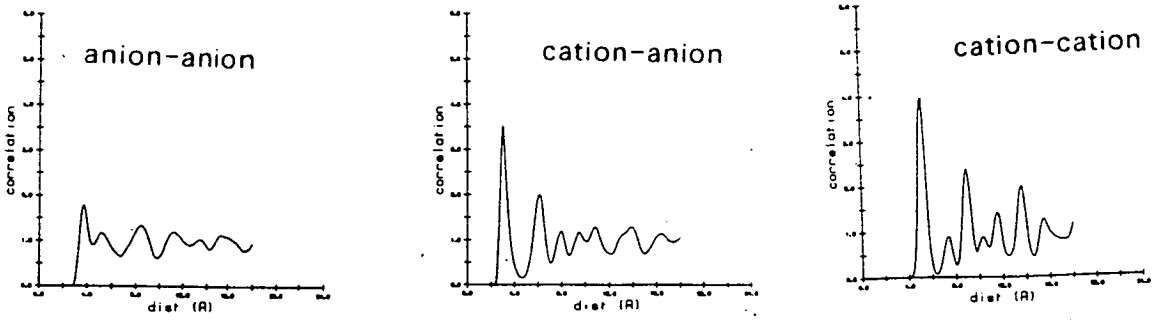
Once we know that the system is showing the appropriate superionic behaviour we need to test that the crystal structure obtained using the PPPM algorithm is consistent with the structures obtained in other MD simulations for which the Ewald summation was used to calculate the Coulombic forces. We can best compare the structures obtained in the various simulations by examining the radial distribution functions (rdfs) for the anion-anion, cation-cation and cation-anion interactions in the various simulations. Using the methods described in section 2.1.5 it is a simple procedure to calculate these functions for each of the simulation runs. Given the large number of ions in the present simulations it was found that it was only necessary to examine 1 of the 48 configurations from each of the simulation runs to obtain accurate rdfs. Figure 4.4 shows a selection of rdfs obtained from the present simulations. These rdfs can be compared with rdfs obtained in other simulations, for example the rdfs obtained for $SrCl_2$ by Gillan and Dixon. The rdfs obtained in these simulations are very similar to those shown in the other simulations both in the position and height of the peaks. It can also be seen from the rdfs in figure 4.4 that all three superionic fluorites have very similar structures (allowing for a rescaling of the lattice parameter between the different compounds).

Examining the rdfs several points become immediately obvious. At temperatures below the superionic transition temperature all three rdfs show many well defined peaks, a characteristic of a well ordered solid. As the temperature is increased the peaks, as expected, become flatter and the anion-anion rdf becomes similar to the rdf obtained from a liquid, indicating that the correlations on the anion sublattice decay rapidly with increasing temperature. At these high temperatures, however, the cation-cation rdf still has many well resolved peaks suggesting that the cation sublattice is still well-ordered in the superionic phase. The rdfs therefore support the results

a) $SrCl_2$ at 770K



b) $SrCl_2$ at 1084K



c) $SrCl_2$ at 1525K

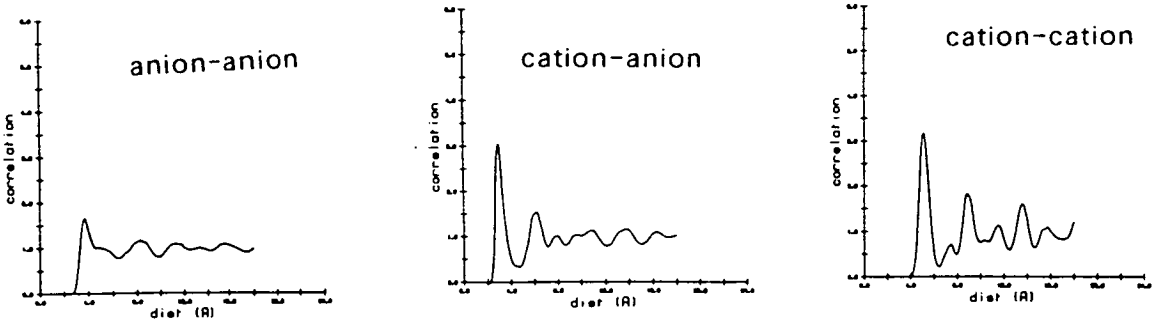
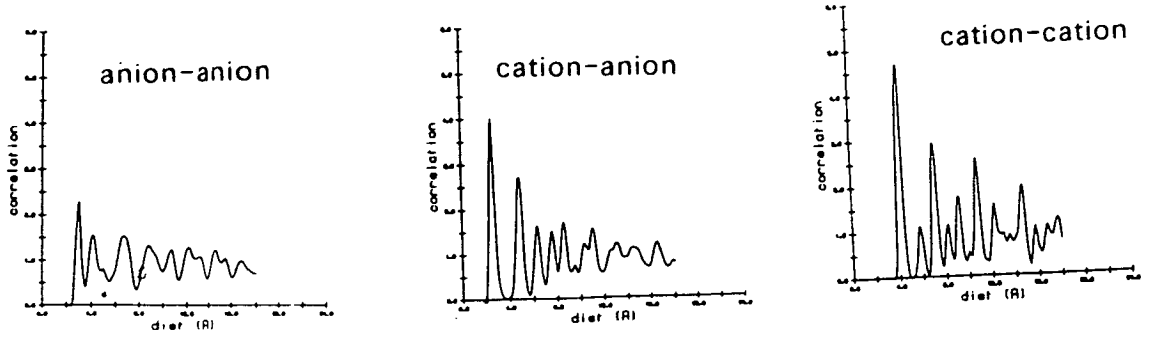
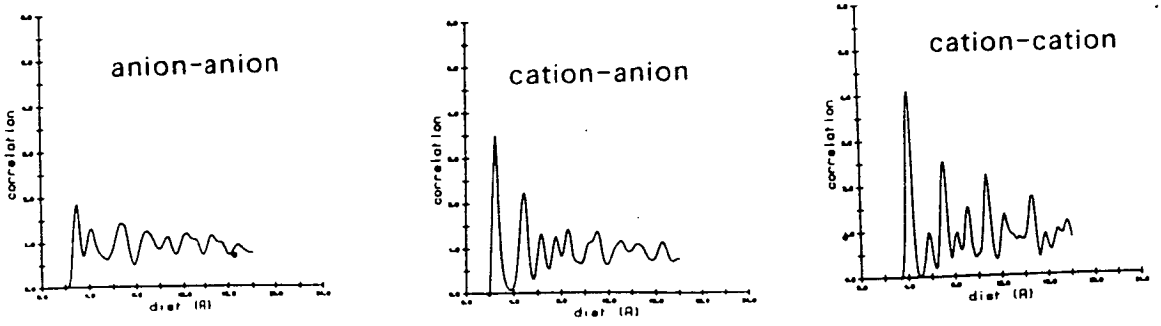


Figure 4.4 The anion-anion, cation-anion and cation-cation rdfs.

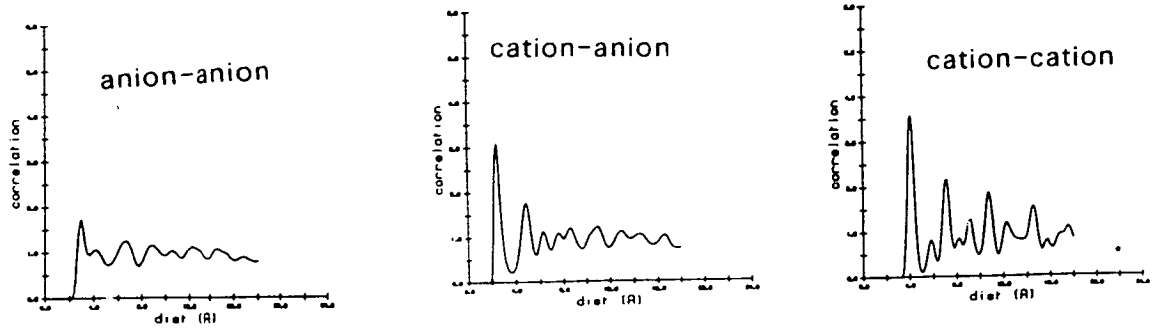
d) CaF_2 at 852K



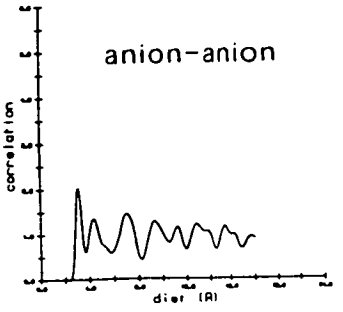
e) CaF_2 at 1222 K

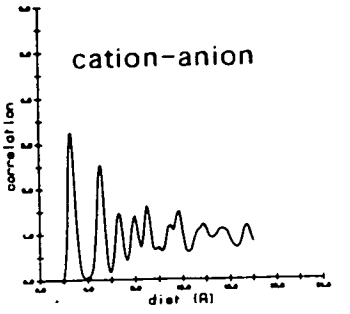


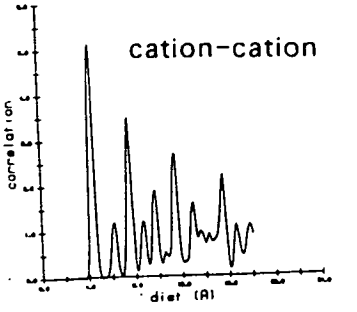
f) CaF_2 at 1495K



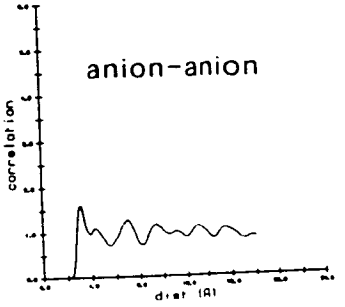
g) PbF_2 at 417K

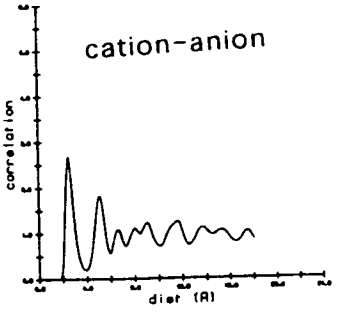


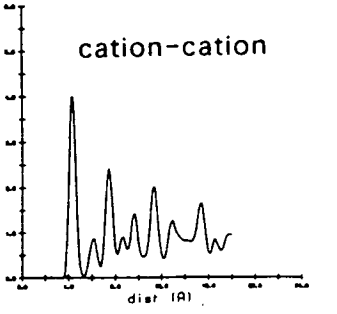




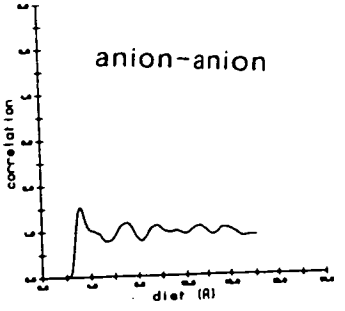
h) PbF_2 at 781K

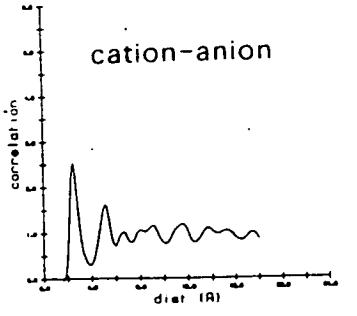


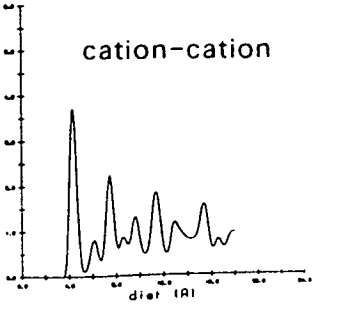




i) PbF_2 at 894K.







obtained from the diffusion coefficients in suggesting that the disorder in the superionic phase is confined to the anion sublattice and that even at high temperatures the cation lattice remains well defined.

4.3. Low Temperature Behaviour

At temperatures below the superionic transition temperature theoretical calculations have predicted that the dominant type of defect in the fluorite structure superionic crystals will be Frenkel defects on the anion lattice (Catlow and Norgett 1973, Catlow *et al* 1977), the interstitial residing in the empty cube-centre site. Both theoretical calculations and analysis of ionic conductivity measurements (Catlow *et al* 1977, Bendall and Catlow 1980, Jacobs *et al* 1984, Jacobs and Ong 1976, 1980, Carr *et al* 1978) have also shown that the anion vacancies should be more mobile than the anion interstitials. Catlow *et al* (1973,1977) have predicted from their lattice statics calculations the probable motion of the Frenkel vacancy and Frenkel interstitial through the crystal. They predict that the vacancy will move along the 100 direction on the anion lattice travelling directly between lattice sites (figure 4.5). The predicted interstitial motion is more complicated, rather than travelling directly between empty cube centre sites in a 110 direction, the interstitial moves in a 111 direction to replace a lattice anion which is then displaced into an adjacent cube centre site (see figure 4.6).

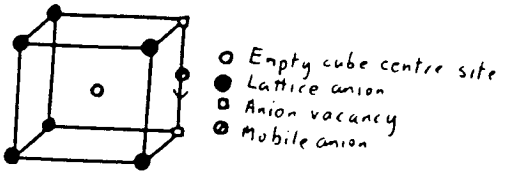


Figure 4.5 The vacancy motion.

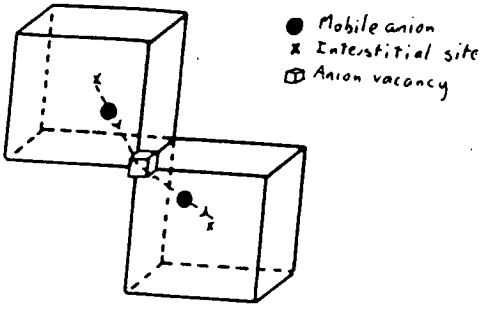


Figure 4.6 The interstitialcy mechanism.

In addition to Frenkel defect motion Catlow *et al* also predict that at higher temperatures pairs of anions will swap positions without the intervention of

any of the Frenkel defects (see figure 4.7).

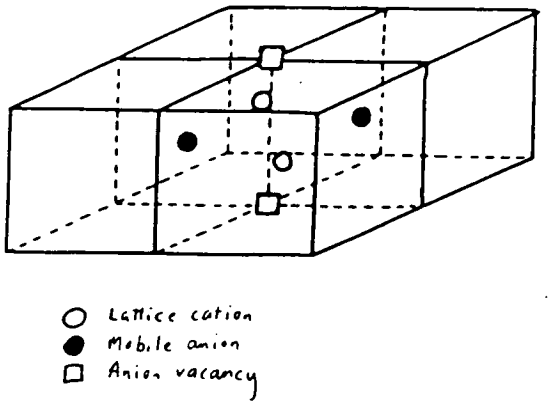


Figure 4.7 The anion swapping mechanism.

At temperatures below the superionic transition temperature the concentration of such Frenkel defects is low and previous MD simulations have been on systems which are too small for such defects to form spontaneously. Some MD simulations have attempted to investigate the behaviour of these defects by specifically introducing a Frenkel defect pair into the simulation and following its subsequent evolution (Moscinski and Jacobs 1984b). The size of the present simulations (over an order of magnitude larger than the systems used by Moscinski and Jacobs) has turned out to be sufficient to enable the properties of spontaneously created defects to be determined without the necessity of artificially creating them.

In order to investigate Frenkel defects in the MD simulations we need a way of analysing the data generated by the DAP so that the Frenkel vacancies and interstitials can be identified. The interstitials can be readily identified by examining all the ions in turn and looking to see the amount of time spent by each in the vicinity of the cube centre interstitial site. Any ion spending an appreciable amount of time near the cube centre site is a candidate for a Frenkel interstitial. The Frenkel vacancies can be easily seen when they change lattice sites. We define an anion to be on a regular lattice site if it is within a sphere of radius d centred on that site, d being chosen to be $a_0/3$ where a_0 is the anion-anion nearest neighbour distance. When a Frenkel vacancy moves

on the anion lattice then this motion will involve an anion moving from one anion lattice site to another (see fig 4.5). By examining the particle data it is possible to identify these moving anion events and so identify the candidates for the Frenkel vacancy. A hopping anion could be due to a mobile interstitial, however mobile interstitial ions will spend time near the cube centre site whereas the anions moving because of vacancy motion will travel directly along a 100 direction. By examining the particle trajectories it is possible to uniquely assign anion hopping events to either mobile vacancies or mobile interstitials.

4.3.1. Analysis of the Low Temperature MD runs

Examining figures 4.2 it can be seen that 6 of the MD simulation runs are of systems below the superionic transition temperature; $SrCl_2$ at 770K, 823K and 928K, CaF_2 at 852K and 975K, and PbF_2 at 417K.

The simulation of $SrCl_2$ at 770K was found to contain one Frenkel defect. The interstitial remained centred on the cube centre site for 5.04ps over which the simulation was observed (figure 4.8) whereas the associated Frenkel vacancy changed lattice site four times in the same period. During each hop the moving anion travelled directly between anion lattice sites as predicted by Catlow *et al* (figure 4.9). The average time taken for the vacancy to complete a move between lattice sites was only .2ps compared with an average residence time of the vacancy on an anion site of approximately 1.2ps, it therefore seems reasonable to describe the vacancy as 'hopping' between the anion sites.

The simulation of $SrCl_2$ at 823K was also found to contain only one Frenkel defect, the defect being created .63ps after the start of the data collection run. At this increased temperature both the interstitial and vacancy are more mobile, the interstitial executing one jump and the vacancy seven in the 5.04ps. The trajectories of the moving interstitial followed the path path predicted by Catlow *et al.* As before the vacancy hops directly along the 100 direction, the average hopping time being .2ps. The average time spent by the interstitial in the cube centre site was about 2.3ps.

The simulation of $SrCl_2$ at 928K initially contained three Frenkel defects. 0.6ps

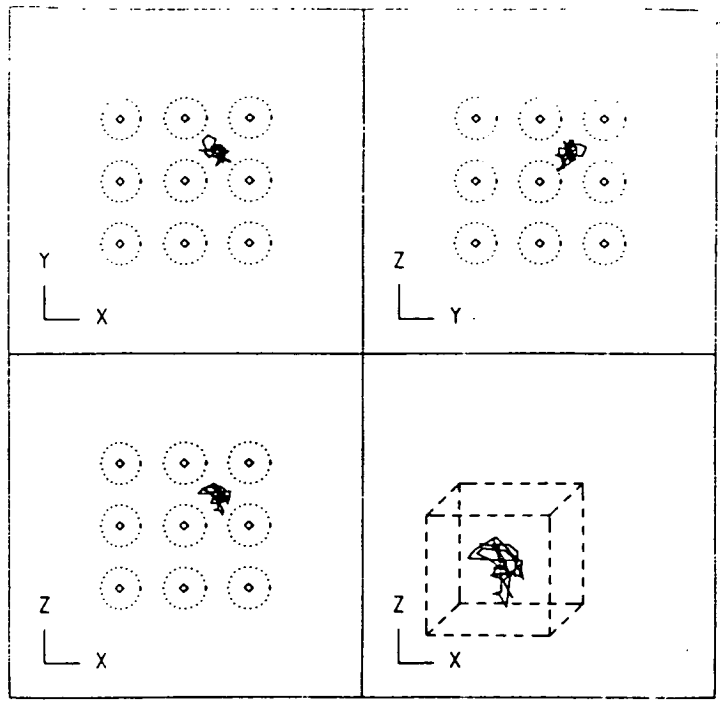


Figure 4.8 The interstitial defect in $SrCl_2$ at 770K. The diagram shows the three projections of the anion motion onto the XY, ZY and XZ planes and a representation of the motion in three dimensions. The diamonds represent anion lattice sites and the dashed circles have a radius $a_0/3$.

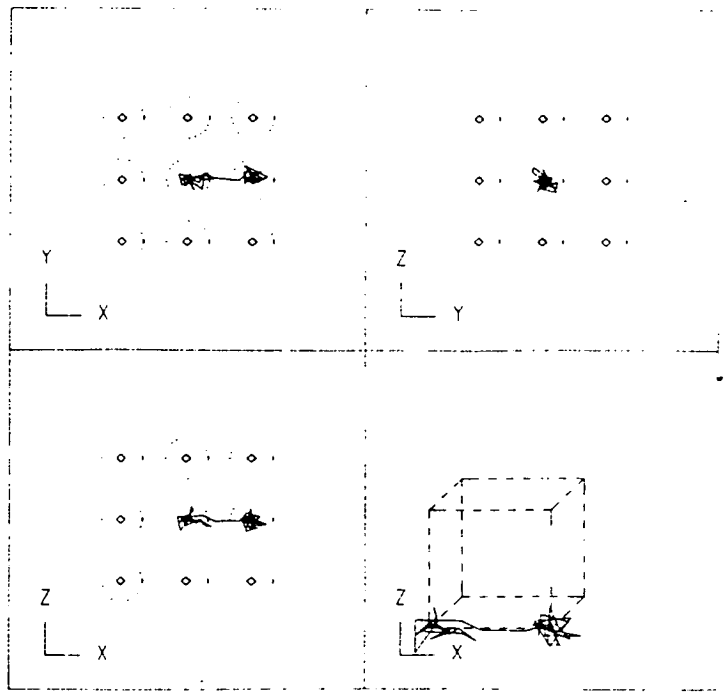


Figure 4.9 Motion of a typical vacancy defect in $SrCl_2$ at 770K. Note that the direction of motion of the vacancy defect is in the opposite direction to the motion of the associated hopping anion.

after the start of the data collection run one of the vacancy-interstitial pairs was annihilated, another pair annihilating after 3.6ps. During the run no Frenkel defect creation events were seen other than the creation of some short lived defects in which the vacancy annihilated the interstitial after only one hop. During the observation period of 5.04ps the interstitials completed six jumps and the vacancies completed 13 hops. As before the interstitial hops were via the 111 interstitialcy mechanism and the vacancy hops were via the the direct 100 hop mechanism. In this run the average time spent by the interstitial in the cube centre site was 1.6ps. At no point in this run were the direct anion exchanges predicted by Catlow *et al* observed.

The simulations of CaF_2 below the superionic transition temperature showed similar behaviour to the simulations of $SrCl_2$. The simulation of CaF_2 at 852K contained one Frenkel defect. During the run the interstitial jumped once via the interstitialcy mechanism and the vacancy completed 6 hops. As before the moving anion in the vacancy hop took on average only .2ps to move between anion lattice sites and the interstitials spent over 3ps in the cube centre sites between hops. No Frenkel defect creation or destruction events were seen over the 5.04ps observation period.

The simulation of CaF_2 at 975K again contained a single Frenkel defect, both the vacancy and interstitial completing four hops in the 5.04ps. The average time spent by the interstitial in the cube centre site was approximately .8ps at this temperature compared to 3ps at 852K. Figure 4.10 shows three connected interstitial hops. The figure shows three separate anions undergoing hops. The first anion (right hand cube) started its hop at t=2.415ps and completed it at t=3.465ps. The next anion (centre cube) started its hop at 3.15ps and completed it at t=3.99ps. The last anion in the hopping chain (left hand cube) started its hop at t=3.675ps and completed it at t=4.62ps (where t=0 corresponds to the start of the data-collection run after the system had properly equilibrated). It can be seen from the times of hopping given above that the various interstitial hops overlap each other, one jump starts before the previous one has been completed. Again no sign was seen of the direct anion exchange mechanism predicted by Catlow *et al*

The final simulation of a crystal below the superionic transition temperature was of PbF_2 at 417K. Once again the crystal was found to contain a single

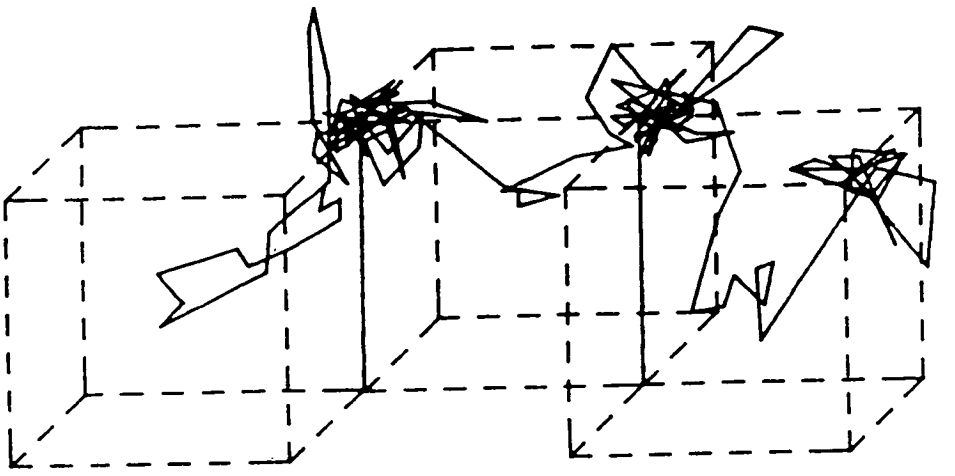


Figure 4.10 Three interstitial jumps from the simulation of CaF_2 at 975K. Note that the interstitials move via the interstitialcy mechanism described by Catlow *et al*, 1977 (see figure 4.6)

Frenkel defect, the vacancy completing 11 hops and the interstitial completing 5 hops in the 5.04ps observation period. As before the interstitial and vacancy hops were via the mechanisms predicted by Catlow *et al.* In this simulation the average amount of time spent by the interstitial in the vicinity of the cube centre site was only .6ps. Analysing the interstitial trajectories (figure 4.11) it can be seen that although the mobile anion does pass close to the cube centre site it is not trapped by the site, the anion simply passes straight through the centre of the empty anion cube.

4.3.2. Conclusions

Analysing the MD simulation runs at temperatures below the superionic transition temperature it can be seen that the MD results support the conclusions of the theoretical lattice statics calculations. Below the superionic transition temperature the predominant defect is the Frenkel defect, the interstitial and vacancy moving through the lattice using the same mechanisms predicted by the same calculations and the vacancies being significantly more mobile than the interstitials. There is no evidence in the low temperature runs for the direct anion swap mechanism although this mechanism could become important at higher temperatures. Increasing the temperature of the simulation to temperatures close to the superionic transition temperature it does become evident, however, that a simple Frenkel defect model may not be adequate to describe the high temperature behaviour of the fluorite structure crystals. Although at low temperatures the Frenkel interstitial is trapped by the cube centre site (see figure 4.8 for example), as the temperature is increased the time spent by the interstitial in the cube centre site decreases until a point is reached at which the mobile interstitial ions are not trapped by this site but simply pass close to it. At this point it is not obvious that the cube centre location should be called an 'interstitial site', it is simple a point on the trajectory of one of the mobile ions. Without a stable cube centre site it becomes difficult to see how a simple Frenkel defect model could work. The behaviour of the vacancies, however, remains consistent through all of the simulations. In all the runs described above the vacancies travel in a 100 direction, the associated hopping anion moving rapidly between the anion sites. The only effect of increasing the temperature is to increase the mobility of the vacancy defects, the vacancies can still be

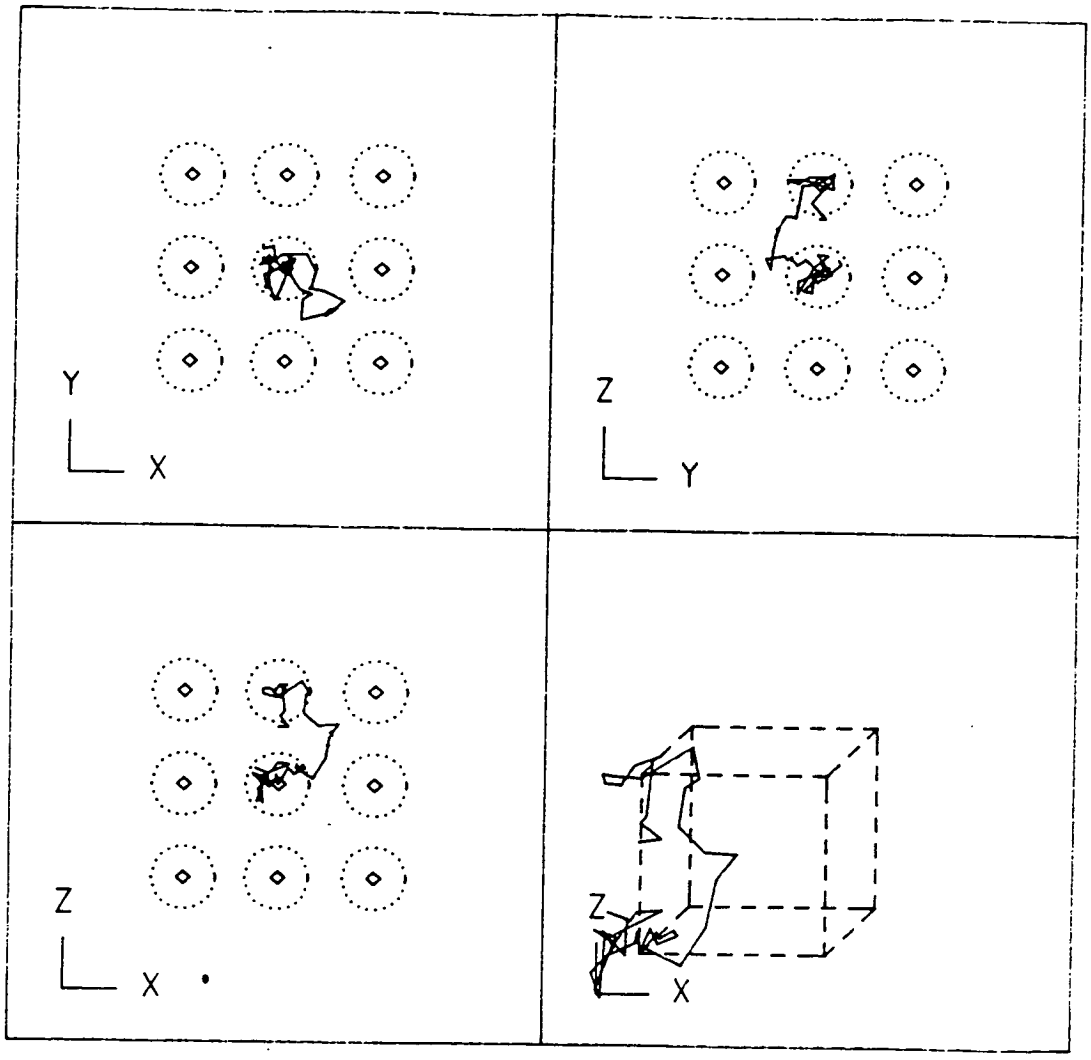


Figure 4.11 Interstitial motion in PbF_2 at 417K.

thought of as residing on the anion lattice sites.

4.4. High Temperature Behaviour

As the temperature of the fluorite structure crystals is increased above the superionic transition temperature the disorder on the anion sublattice greatly increases. At temperatures below the transition temperature the average number of anions not on anion sites (i.e. not within $a_0/3$ of an equilibrium anion lattice site where a_0 is the nearest neighbour anion-anion separation) at any instant is about 3% whereas in the superionic phase up to 30% of the anions are not on anion sites (figure 4.12 shows how this disorder varies with temperature for $SrCl_2$). These results are similar to the neutron scattering results of Shapiro and Reidinger (1975) who found that in the superionic phase of the fluorite structure crystals, only 70% of the anions were found near anion lattice sites.

At first sight it might appear that the anion sublattice has in some way melted in the superionic phase (Dworkin and Bredig 1968), especially if we note that both the anion-anion rdfs and the form of the anion mean-square displacements (figures 4.3 and 4.4) resemble those obtained from liquids. This increasing disorder with increasing temperature can be seen in the crystal configurations. Figure 4.13 shows slices of the different simulations at a range of temperatures. At low temperatures it can be seen that the configurations are well ordered, the anions and cations being close to their regular lattice sites. At higher temperatures the anion lattice becomes more disordered whilst the cations still remain close to their equilibrium lattice sites. It is now well established, both from MD simulations and neutron scattering results, that the anion sublattice remains solid (see chapter 1 for a discussion of the relevant results). Most of the disorder apparent in the rdfs etc. arises as a result of anions undergoing large anharmonic vibrations about lattice sites. If the anion density in the unit cell of the fluorite structure crystals is calculated it can be seen that the anion density is still strongly centred about the equilibrium anion lattice sites. Some of the anions do diffuse through the lattice, but their motion is via discrete hops between these lattice sites. By analysing the anion diffusion in terms of discrete particle hops much can be learnt about the properties of the fluorite structure crystal in the superionic

phase.

4.4.1. Single Particle Hopping Analysis

We define a hop as an anion moving from within $a_0/3$ of one anion lattice site to within $a_0/3$ of another anion lattice site. Using this definition of a particle hop we can produce a hopping list for each of the runs; an entry in the list being composed of the initial and final sites of the hopping particle, the time the particle leaves the initial site, the time it arrives at the new site and the identification number of the hopping anion. Using this list we can easily determine the direction and time of flight of each of the hops in each of the simulation runs. Table 4.2 gives the number of completed hops in each of the simulations and the proportion of these hops in the various hop directions.

It can be seen from this table that the majority of the hops are along a cube edge in the 100 directions. There are also particle hops to sites on the opposite corner of the same face in the anion cube in the 110 direction, a few particle hops to the anion site on the opposite corner of the cube in the 111 direction, and a very small percentage are longer hops, in the 200, 210 and 211 directions (figure 4.14).

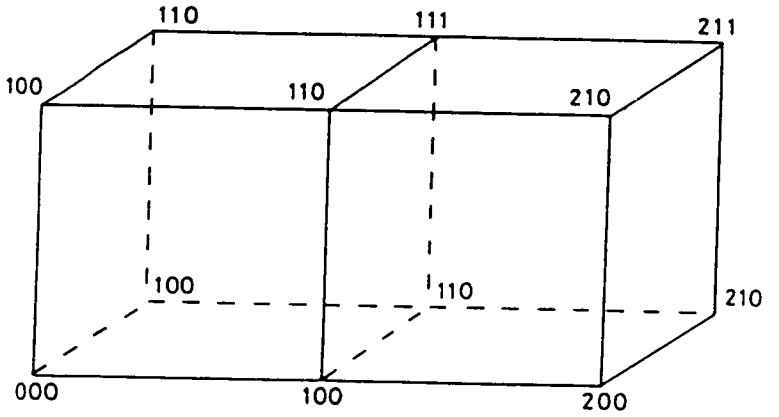


Figure 4.14 A selection of 100, 110, 111 etc hops from the site 000.

The 200, 210 and 211 hops have not been reported from other simulations, probably because they are so infrequent that the chance of observing them in a simulation of a small system is remote. However, as these hops are so

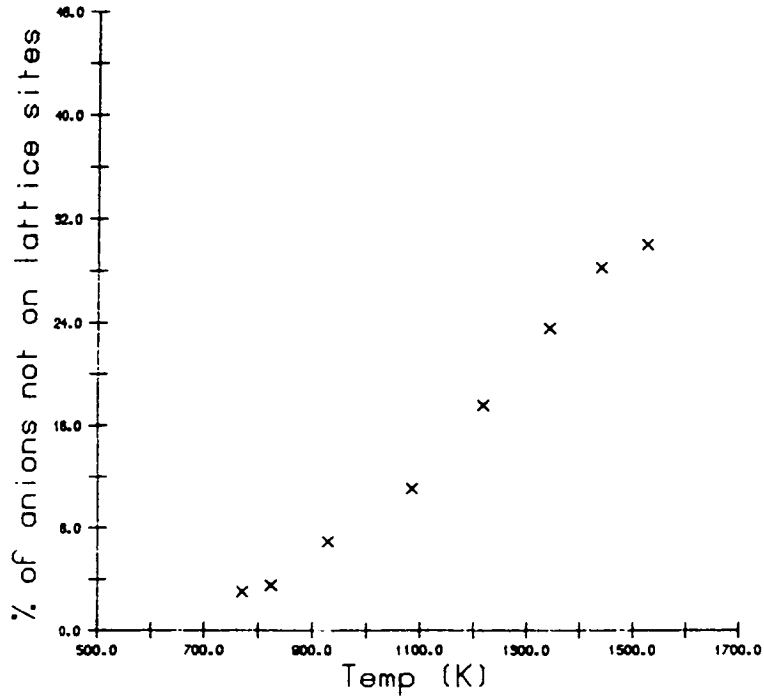
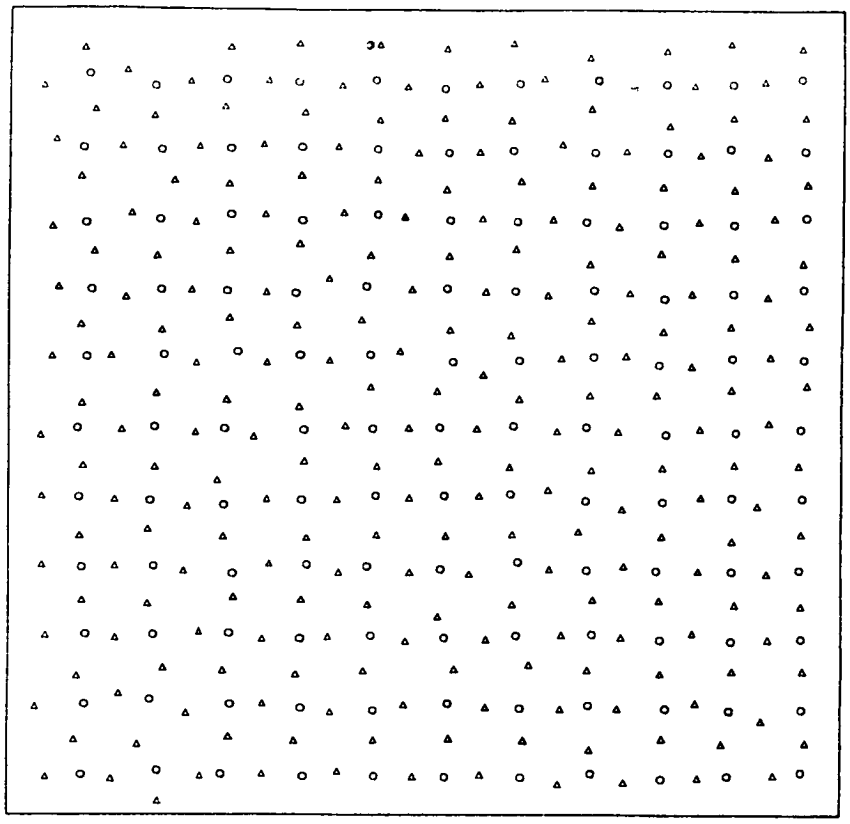
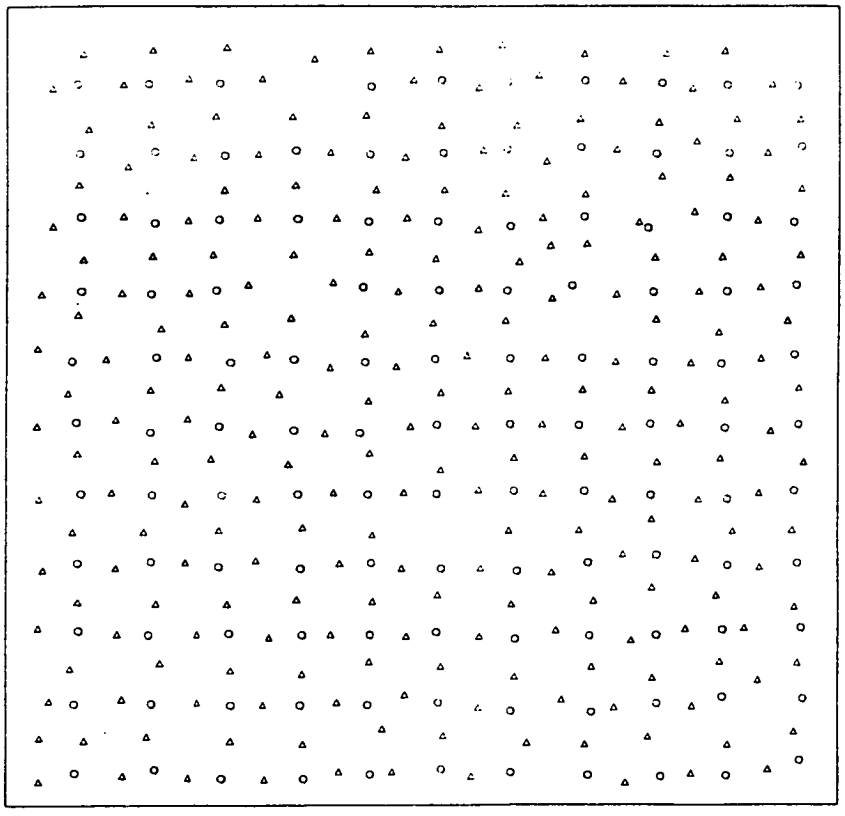


Figure 4.12 The percentage of anions not within $a_0/3$ of an equilibrium anion lattice site as a function of temperature.

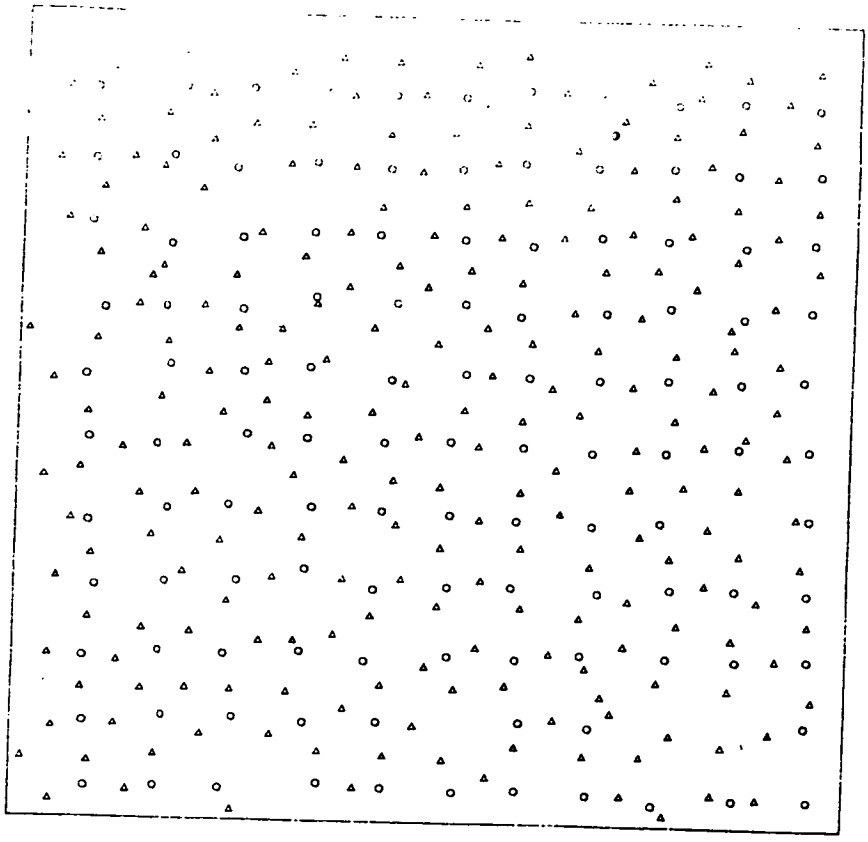


a) $SrCl_2$ at 700K

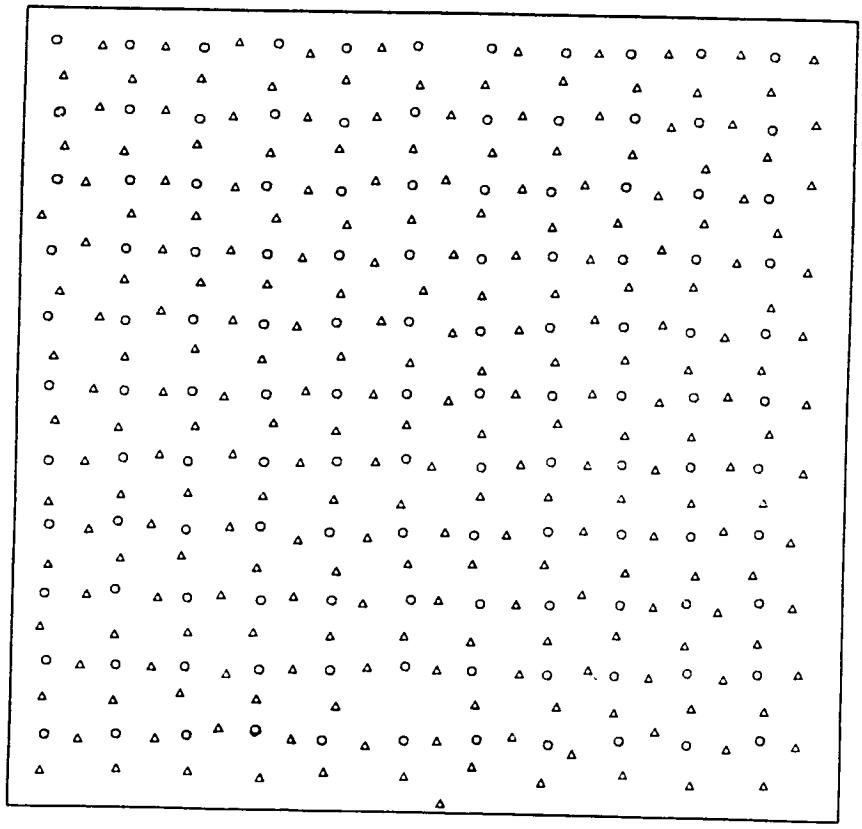


b) $SrCl_2$ at 1028K

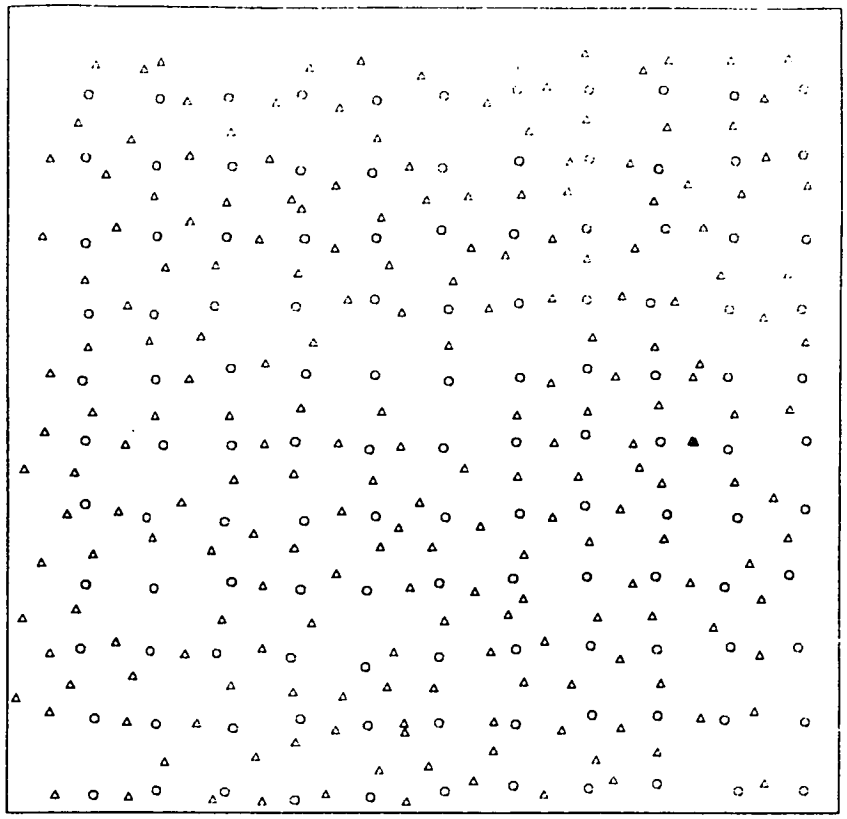
Figure 4.13 Pictures of the instantaneous positions of the ions in some of the MD simulation runs. Each picture shows the ions found in one given xy plane of PP cells (see figure 3.2). In some cases ions whose equilibrium lattice sites are contained within this volume have temporarily oscillated onto neighbouring xy PP cell planes, similarly ions from neighbouring xy planes of PP cells may have moved onto the plane.



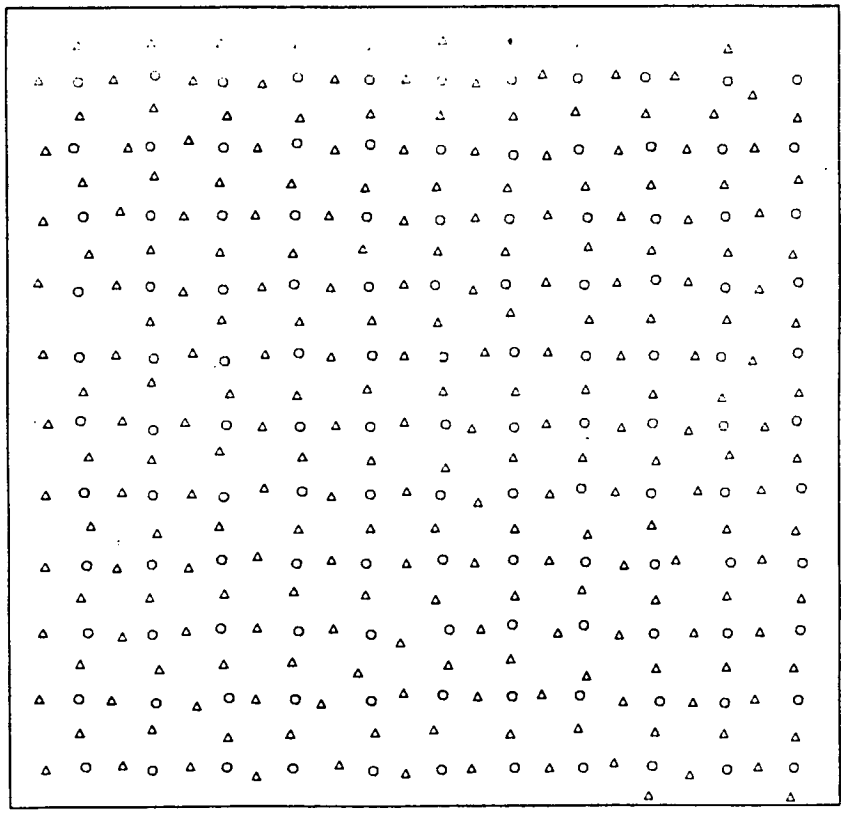
c) $SrCl_2$ at 1525K



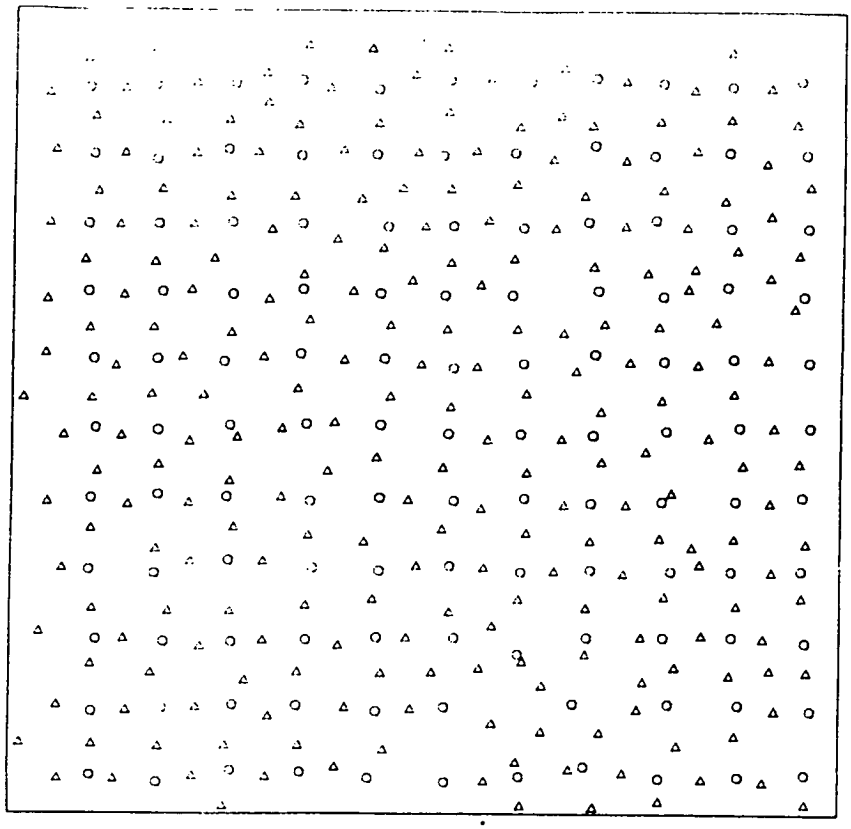
d) PbF_2 at 471K



e) PbF_2 at 894K



f) CaF_2 at 852K



g) CaF_2 at 1495K

Table 4.2a Percentage of the total number of hops which are in each of the various hopping directions.

Compound	Temp (K)	No of hops	%100	%110	%111	%other
$SrCl_2$	770	4	100.0	-	-	-
$SrCl_2$	823	11	90.9	9.1	-	-
$SrCl_2$	923	23	95.7	4.3	-	-
$SrCl_2$	1084	155	93.5	6.5	-	-
$SrCl_2$	1216	506	90.7	7.9	1.4	-
$SrCl_2$	1341	900	85.6	11.2	3.1	0.1
$SrCl_2$	1438	1233	87.8	10.1	2.1	-
$SrCl_2$	1525	1712	84.0	13.8	2.1	0.1
PbF_2	417	16	100.0	-	-	-
PbF_2	571	63	87.3	7.9	4.8	-
PbF_2	781	718	87.8	9.9	1.7	0.6
PbF_2	841	1245	85.4	12.3	2.0	0.3
PbF_2	894	1637	86.0	11.6	2.1	0.3
CaF_2	852	7	85.7	14.3	-	-
CaF_2	975	8	75.0	25.0	-	-
CaF_2	1222	151	87.4	8.6	4.0	-
CaF_2	1366	493	88.1	8.1	3.6	0.2
CaF_2	1495	1272	84.5	12.3	3.1	0.1

Table 4.2b Average time of flight for anion hops in each of the different hopping directions.

Compound	Temp(K)	100(ps)	110 (ps)	111 (ps)	Average for all hops (ps)
$SrCl_2$	770	0.289	-	-	0.289
$SrCl_2$	823	0.431	2.84	-	0.649
$SrCl_2$	923	0.649	0.84	-	0.657
$SrCl_2$	1084	0.622	1.43	-	0.674
$SrCl_2$	1216	0.639	1.35	1.37	0.705
$SrCl_2$	1341	0.573	1.28	1.38	0.680
$SrCl_2$	1438	0.606	1.13	1.39	0.678
$SrCl_2$	1525	0.598	1.16	1.13	0.689
PbF_2	417	0.669	-	-	0.669
PbF_2	571	0.645	1.79	0.735	0.740
PbF_2	781	0.463	0.805	0.910	0.506
PbF_2	841	0.500	0.917	1.00	0.565
PbF_2	894	0.522	0.885	1.01	0.577
CaF_2	852	0.245	3.89	-	0.765
CaF_2	975	0.490	0.789	-	0.564
CaF_2	1222	0.408	1.01	0.892	0.479
CaF_2	1366	0.424	0.845	1.03	0.483
CaF_2	1495	0.463	0.789	1.02	0.521

infrequent, they will not have a significant contribution to the superionic behaviour of the crystals. Examining the table it would appear that these 'long' hops are more likely to occur in PbF_2 than in $SrCl_2$ or CaF_2.

Examining table 4.2 we can see that in the superionic phases the ratios of (100), (110) and (111) hops remains remarkably constant. The ratio of 100:110:111 hops is approximately 83:10:3 for all the simulation runs, independent of the temperature of the simulation or the compound being simulated. Also the average time of flight, (τ_f), would appear to be independent of the temperature of the simulation. If the average value for τ_f for each of the three compounds is divided by the appropriate value of a_0, then it can be seen that the average speed of a hopping anion is approximately the same for all three compounds. Examining table 4.2 it is obvious that all three compounds show very similar single particle hopping behaviour, the only slight difference between them being the larger number of 'long' hops in the PbF_2 simulations.

Using the hopping list it is also possible to calculate the self-diffusion coefficients. We can calculate the hopping rate, s, by counting the number of anion hops completed in some time interval. Assuming that the direction of the hops of any given anion is statistically independent, then the anion self-diffusion coefficient is given by:

$$D = (1/6)b_0^2 s \qquad (4.4)$$

(Corish and Jacobs 1973) where b_0^2 is the average hopping distance (which can be calculated from the proportions of the different hops). The values of D obtained from the hopping list are compared with those obtained from the gradient of $<r_\alpha(t)^2>$ in figure 4.15. The two methods give similar results for D, but the values of D obtained via the hopping analysis are consistently about 10% higher than those obtained via the gradient. This can be explained if the anion hops are not independent. Examining the hopping list we can see that in some cases an anion hop is immediately followed by a return anion hop giving a value for D which is consistently higher from the hopping analysis than from the gradient of $<r_\alpha(t)^2>$.

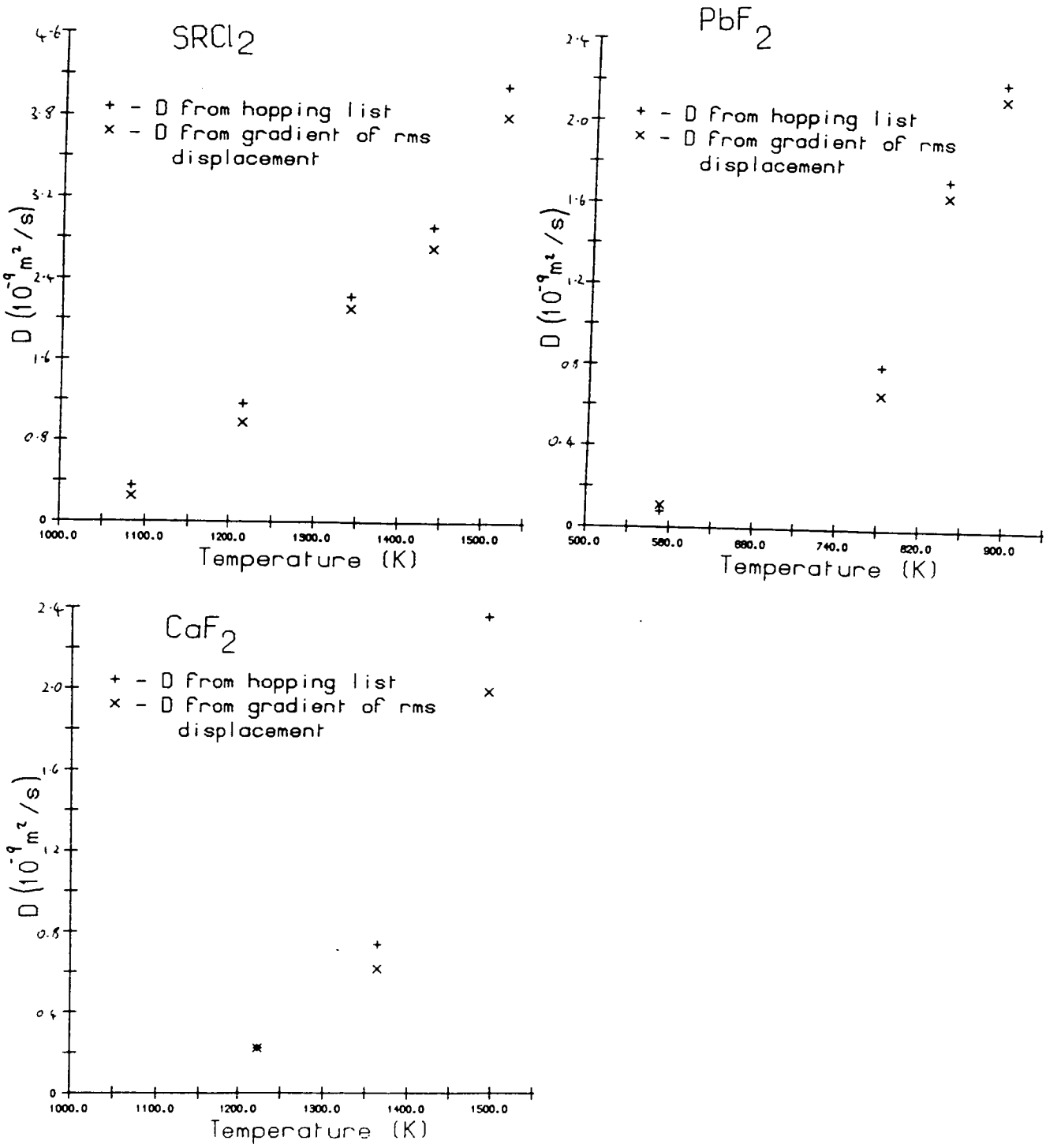


Figure 4.15 A comparison of the values of the anion self diffusion coeeficients obtained via the hopping list and the gradient of the rms anion displacement graphs (figure 4.2)

4.4.1.1. Analysis of Hopping-times

It is possible to examine the distribution of τ_f for each of the three hop directions in the various simulations. Figure 4.16 shows the distribution of hopping-times obtained in some of the high temperature simulation runs (the high temperature runs being chosen simply because the large number of anion hops at high temperatures results in better statistics). Examining the graphs several interesting features are readily apparent. The first is that the average hop times shown in table 4.2 do not correspond to any peaks in the hopping time distribution, the hopping-time distribution is not symmetrically distributed about the mean hopping time. Secondly, the 100 and 110 distributions have very different forms. The 100 distributions all have a large, narrow peak at 0.2-0.4ps, with some of the distributions showing signs of a second peak or shoulder at around 1ps. The 0.2-0.4ps peak is not present in any of the 110 or 111 distributions, these distributions consisting of a single, wide peak centred on about 1ps. Note also that the tails of the 100 and 110 distributions are approximately coincident.

Extending the ideas used to explain the low temperature defect behaviour of the fluorite structure crystals it is possible to provide an explanation for the form of the hopping-time distribution functions. In the low temperature phase we would expect to see two peaks in the 100 hopping-time distributions; one peak at about .3ps, corresponding to the vacancy hops and the other at about 2-3ps corresponding to anions involved in interstitial hops. As the vacancies are more mobile than the interstitials at low temperature, we would expect the peak due to anions involved in vacancy hops to be larger than the peak due to anions involved in interstitial hops. The 110 and 111 distributions would also contain the peak due to interstitials motion, but not the peak due to vacancy motion, the vacancies moving exclusively in the 100 directions. The interstitialcy mechanism involves anions moving from a regular anion lattice site, up to the cube centre site and then back to one of the regular anion sites again. From the cube centre all the anion sites on the corners of the empty cube will be equivalent (except, perhaps, for the site which has just been vacated). If we therefore assume that an anion in the cube centre site is equally likely to move to any of the cube-corner sites then the ratio of 100:110:111 hops performed by the mobile anions will be 3:3:1. We therefore

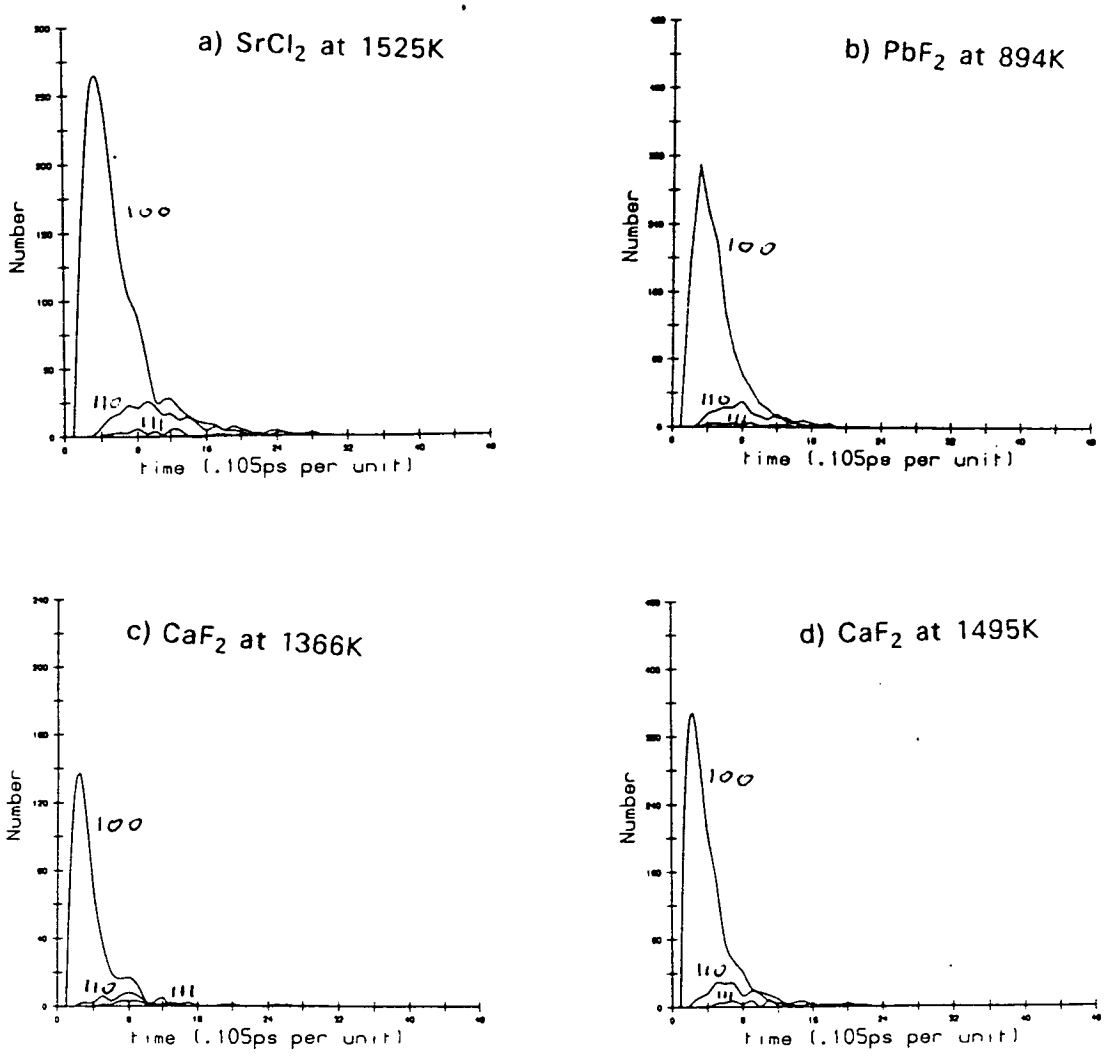


Figure 4.16 The distribution of hopping times for anion hops in the 100, 110 and 111 directions.

assume that the number of 100 hops that are the results of interstitial motion will equal the total number of 110 hops (which should all be interstitial hops). At low temperatures the peak in the 100 hop time distribution due to interstitial motion should be equivalent to the single peak in the 110 hopping-time distribution. We have seen that as the temperature of the crystal is increased, the average time taken for an anion to complete an interstitial hop decreases (see section 4.3). We would therefore expect the 100 interstitial peak to move towards the 100 vacancy peak as the temperature of the simulation reaches the superionic transition temperature. At temperatures above the superionic transition temperature these two distributions could well overlap.

Using these ideas it is possible to interpret the form of the hopping-time distributions. The main peak in the 100 distributions is at 0.2ps-0.3ps, the average time taken for a vacancy hop. We can therefore conclude that this peak is due to the motion of anions involved in vacancy hops. Similarly, as we expect the majority of 110 hops to be due to interstitial motion, we can assume that the 110 distributions show the hopping-time distribution of anions involved in interstitial motion. If we now examine the 100 distributions which show signs of two peaks (the $SrCl_2$ and CaF_2 distributions) we can see that the position and shape of the subsidiary peak corresponds to the position of the 110 peak. We therefore assume that the second peak or shoulder in the 100 distribution is due to the motion of interstitials along the 100 direction. Figure 4.17 shows the decomposition of the 100 hopping-time plots into two separate distributions, one corresponding to interstitial motion (which is assumed to be equivalent to the 110 hopping-time distribution) and the other corresponding to vacancy motion. It is now possible to divide the anion hops into interstitial hops and vacancy hops. If there are N_{TOT} hops recorded in a simulation run, with N_{100}, N_{110}, N_{111} hops in the (100), (110) and (111) directions respectively, then the number of interstitial hops will be $2N_{110}+N_{111}$ with $N_{TOT}-2N_{110}-N_{111}$ vacancy hops. Calculating these sums for the various simulations we obtain the results shown in table 4.3. It can be seen from this table that if we assign the defect types using the above prescription then the vacancies are four or five times as mobile as the interstitials.

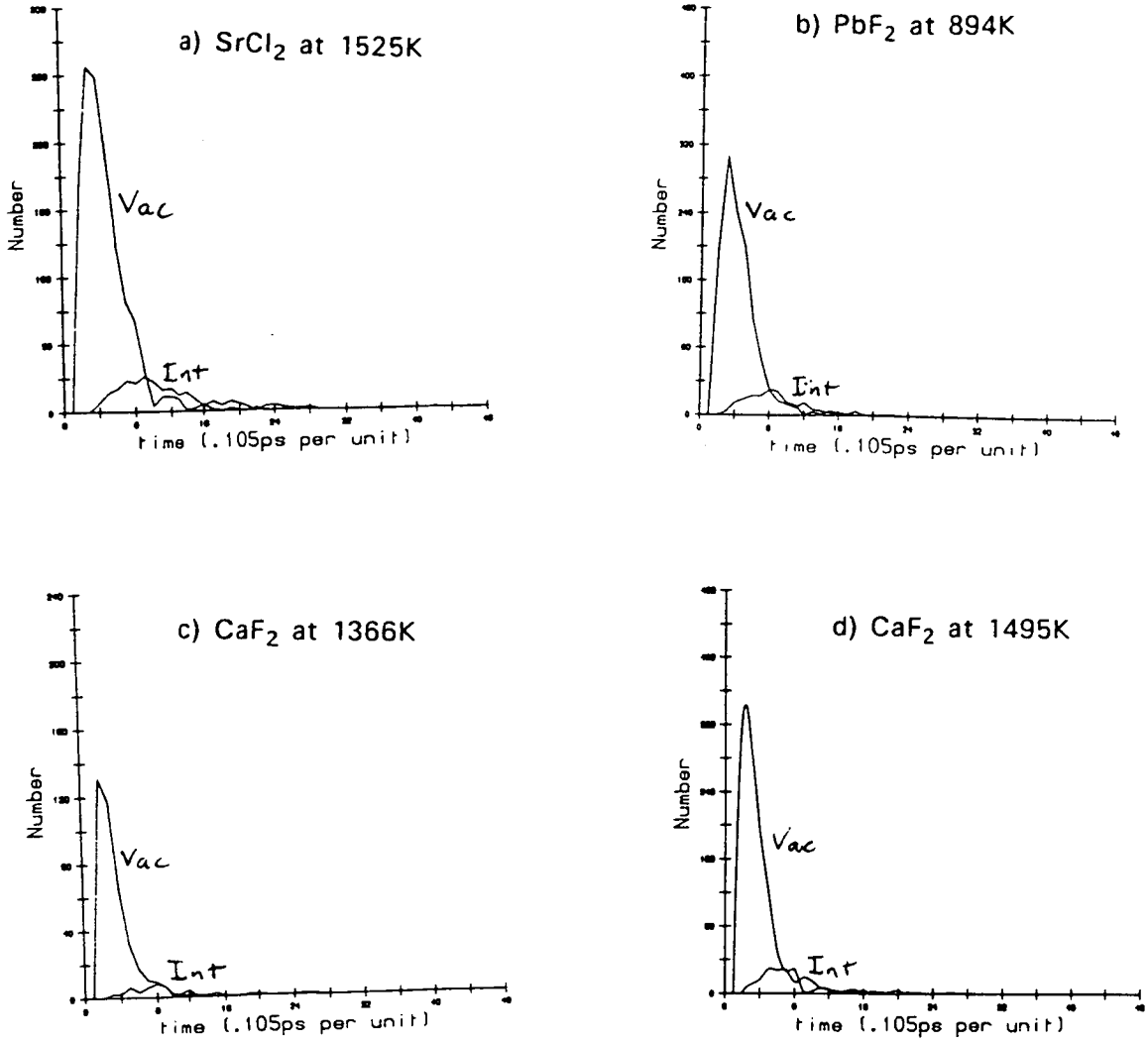


Figure 4.17 The decomposition of the 100 hopping-time distributions into two peaks; one corresponding to vacancy motion and the other to interstitial motion.

Table 4.3 The number of vacancy and interstitial hops determined from the analysis of the hopping-time distribution functions. All the results are for the simulations of $SrCl_2$

Temp (K)	No of hops	No of vacancy hops	No of interstitial hops
1084	155	115	40
1216	506	345	161
1341	900	632	268
1438	1233	874	359
1525	1712	1132	580

There are several weak points in the above argument. The first is a simple lack of data. As the argument relies on the examination of the second peak in the 100 data we need to test that this feature is not simply a statistical quirk, but is indeed a real feature. We are also assuming that it is possible to interpolate the low temperature behaviour of the crystals into the superionic phase. The above argument also relies on the decomposition of the 100 plot into two separate distributions. If we are to do this decomposition properly then we need to calculate, *a priori*, the expected form of the interstitial and vacancy hopping-time distributions and attempt to fit these distributions to the observed data. Fitting the data to simple Gaussian or Lorentzian functions will not be adequate as the distributions are asymmetric, some more complicated distribution function is called for. One possibility would be to attempt to fit the distributions to some form of Boltzmann function. Certainly it might be expected that the vacancy hopping-time distribution could be modelled using some form of Boltzmann distribution. If the time taken for an anion vacancy to hop between lattice sites is compared with the time that a free anion in a gas would take to cover the same distance at the same temperature then the two times are similar. The time of flight for the anion hop, assuming the flight to be gas-like, is given by (Gillan and Dixon 1978):

$$(1/6)a_0/\langle v_x^2\rangle^{.5} = 0.2\text{ps at } 1522\text{K} \qquad (4.5)$$

where $\langle v_x^2\rangle$ is the r.m.s. thermal velocity. As this time is similar to the average time required for a vacancy hop we might expect the distribution of hopping-times for vacancy motion to be similar to the distribution of flight

times for gas-like particles covering the same distance. This work has not yet been attempted, we would want data from many more hops before attempting to fit to the observed distribution and the longer runs required to obtain this data have not yet been performed.

4.4.1.2. Defects and Hopping Trajectories

In the previous subsection we attempted to determine whether an anion hop was the result of vacancy or interstitial motion on the basis of the time taken for the hop to be completed. An alternative method of assigning defect types is via an analysis of the trajectories of the mobile ions. In section 4.3 we saw that in the low temperature phase vacancies move directly through the anion lattice in a 100 direction whereas interstitials move through the empty cube centre sites travelling along 111 directions. If a hopping anion trajectory passes within $a_0/3$ of the empty cube centre site then we can label the hop as an interstitial hop whereas if the trajectory of the mobile ion does not enter this region then we label it as a vacancy hop. Table 4.4 shows the results of this analysis as applied to the $SrCl_2$ simulation runs. Although the numbers of interstitial and vacancy hops are not identical to those shown in table 4.3, they are only different by approximately 10 to 20%, reasonable agreement considering the crude nature of both the analysis methods. Whichever of the analysis techniques we use, the final conclusion is that the vacancy defects are significantly more mobile than the interstitial defects.

Table 4.4 The number of vacancy and interstitial hops determined from the analysis of the particle trajectories. All the results are for the simulations of $SrCl_2$

Temp (K)	No of hops	No of vacancy hops	No of interstitial hops
1084	155	135	20
1216	506	419	87
1341	900	670	230
1438	1233	958	275
1525	1712	1201	511

4.4.2. Correlated Hopping Analysis

As the number of anions and anion sites is equivalent, a particle can only hop from site 1 to site 2 if the particle originally at site 2 hops to a third site, 3. The particle hops can therefore be placed into correlated sequences. By studying these sequences it is possible to represent the disorder in terms of vacancy and interstitial motion in yet another way (Dixon and Gillan 1978). The correlated hopping analysis assumes that both the interstitial and vacancy defects can be placed on lattice sites (see figure 4.18) so avoiding the problem of deciding when a particle has passed close enough to the empty cube centre site to be classed as an interstitial.

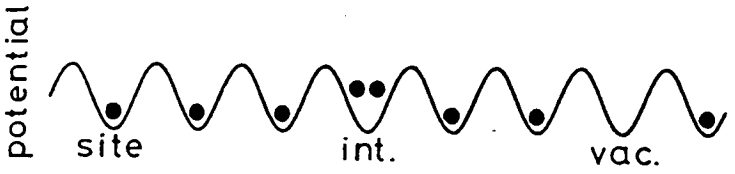


Figure 4.18 The definition of interstitials and vacancies in the correlated hopping interpretration of the defects in the fluorite structure superionic crystals.

Consider a given site, n. As the simulation proceeds some anions will hop onto the site n whilst some anions originally on site n will hop onto new sites. Using the original hopping list we can produce a new array with 2420 elements, each element of the array corresponding to one of the anion sites and storing all the relevant information on the hops involving the particular site. We can then go through this array and reorder the hops involving a given site by the time at which the hop occurs (this time being defined as the average of the time the anion left the original site and the time it arrived at the new site).

Using this new array we can begin to piece together the correlated hopping sequences. For example, consider the entry in the new array corresponding to

the site n. Suppose that there are two hopping events involving this site; one a hop occurring at time t_1 in which an anion moves from site n_1 onto n, and the other involving an anion moving from site n onto site n_2 at time t_2. This process can be represented diagrammatically (see figure 4.19). If t_1 is smaller than t_2, i.e. the anion from n_1 arrives at n before the anion from n hops to site n_2 then we define both hops to be interstitial hops. If, on the other hand, t_1 is greater than t_2, i.e. the anion hops from the site n hops to n_2 before an anion has hopped from site n_1 to n, then we define both hops to be vacancy hops. Figure 4.20 shows how these two alternatives can be represented, the arrows on the hops pointing in the direction of increasing time. Using this representation vacancy hops are represented by upward pointing arrows and interstitial hops being represented by downward pointing lines.

Figure 4.19 The single-site hopping diagram for the site n. The diagram represents an anion hopping from site n_1 to site n at time t_1 with another anion hopping from site n to site n_2 at time t_2.

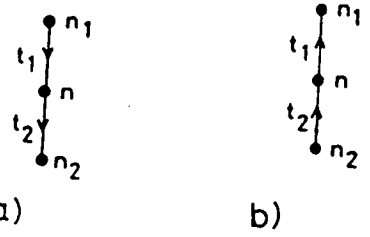


Figure 4.20 a) The single-site hopping diagram representing an interstitial defect hop. b) The single-site hopping diagram representing a vacancy hop.

Now consider the case of a site which is only involved in one anion hop. Although the hops will normally occur in pairs, it is possible that the other hop in the hopping pair either started before we started collecting data or was not completed by the finish of the run. In either case we will only see one hop of the hopping pair. If we observe an anion hopping onto a site n but we do not observe the associated hop of an anion away from the site, we can represent the event diagrammatically as shown in figure 4.21. If, on the other hand, we observe an anion hopping away from a site n without seeing the

associated hop of an anion onto the site n, we can represent the event as shown in figure 4.22.

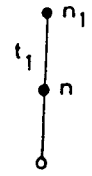


Figure 4.21 A single-site hopping diagram represnting an unpaired anion hop from the site n_1 onto the site n.

Figure 4.22 A single-site hopping diagram represnting an unpaired anion hop from the site n onto the site n_2.

If there are more than two events involving a given site, then the hops can be split up into combinations of the diagrams shown in figures 4.20, 4.21 and 4.22. Consider, for example, the case in which one site is involved in three hops. If we represent particles hopping into the site by '+', and particles hopping away from the site by '−', then there are six possibilities. The hop sequence must be either +−+, +−−, ++−, −++, −+− or −−+ (where the position of the + or − shows the time ordering, i.e. +−+ represents a particle hopping onto the site followed by a particle hopping away from the site and finally another particle hopping onto the site). If we assume that two interstitials or two vacancies can not occupy the same site simultaneously then the sequences +++ or −−− are impossible. The three hops will be split into two diagrams, one of the type shown in figure 4.20 and one of the type shown in figures 4.21 and 4.22. The problem is to determine which two of the three hops should be combined together to form a hopping pair and which hop should be considered as the isolated hop. If the sequence of hops is ++−, −++, −−+ or +−− then the decomposition is obvious, the hopping pair must consist of a +− or a −+, and in each of the above sequences there is only one way either a +− or a −+ sequence can be obtained, eg ++− must split + +−. The +−+ and −+− sequences are more difficult to decompose. Take for example the +−+ sequence, the decomposition could be either + −+ or +− +. We need some way

of choosing the most sensible option. One possible method is to calculate the time difference between the two hops in the hopping pair for both choices of hopping pair, and then choose the decomposition which has the smallest time separation between the two hops forming the hopping pair.

Using similar techniques it is possible to produce a combination of single site hopping graphs such as 4.20, 4.21 and 4.22 for all the 2420 sites in the simulation. Note that a given hop will always appear in two graphs. The top of any single site hopping graph represents a particle entering the given site whilst the bottom of the graph represents a particle leaving the given site. Any hop will therefore appear in the diagram representing the site for which the given hop represents a particle leaving the site and also in the diagram for which the hop represents a particle entering the site. For example consider a hop in which a particle leaves site 1 and enters site 2. There will be a graph for site 1 in which this hop will appear as the bottom half of the graph and there will also be a graph for site 2 in which the given hop will appear as the top half of the graph. In order to construct the correlated hopping chains we need therefore only join together all such pairs of graphs.

Once we have split all the hops into hopping pairs we need to be able to place them together into correlated chains. Consider, for example, the six hopping graphs shown in figure 4.23.

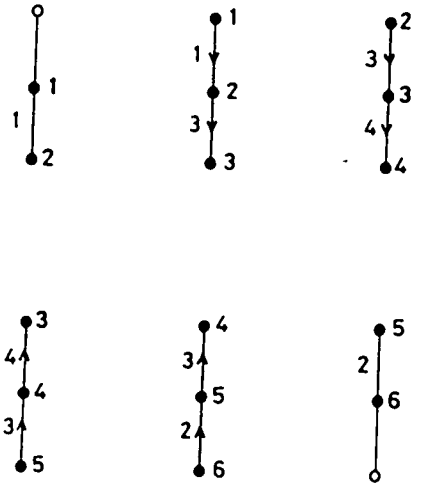


Figure 4.23 Six correlated single-site hopping diagrams

Figure 4.24 shows how these graphs can be joined together to form a correlated hopping sequence. Each bond in the correlated hopping graph corresponds to a single anion hop, each of the hops appearing in two single site hopping graphs. If both these single site hopping graphs show this hop to be a vacancy hop, then we can define this hop to be a vacancy in the correlated hopping graph (see for example the hop between site 5 and site 6 at time 2), similarly if both the single site hopping graphs show this hop to be an interstitial hop then we can define this hop to be an interstitial hop in the correlated hopping graph (see for example the hop between site 2 and site 3 at time 3). However, how do we decide what type of hop we should assign to the anion hop between site 3 and site 4 at time 4. In one of the single site hopping graphs we have defined this hop to be an interstitial hop, whereas in the other we have defined it to be a vacancy hop. As this hop involves a vacancy and an interstitial interacting and annihilating each other, we define these types of hops to be 'destruction' hops. The destruction hops can be identified by the downward 'V' shape they make in the correlated hopping diagram. We can also identify defect 'creation' events from the correlated hopping diagram. As with the defect destruction hops, these hops are characterised by the fact that in one of the single site hopping graphs it appears as an interstitial hop whereas in the other it appears as a vacancy hop. The difference between the destruction and creation events is that the creation events appear as an upward, rather than downward, pointing 'V' shape in the correlated hopping graph. If one of the single site hopping graphs corresponding to a given hop is one of the graphs involving only a single anion hop (see figure 4.24) then we assign the hop type (i.e. interstitial or vacancy) by the hop definition in the other single site hopping graph involving the given site (for example, we define the hop between site 1 and site 2 in figure 4.24 to be an interstitial hop).

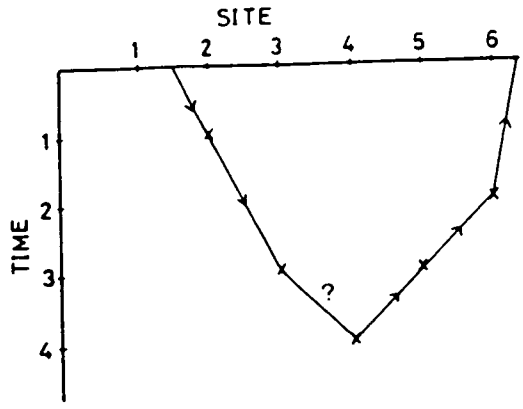


Figure 4.24 The correlated hopping graph for the six correlated single-site hopping diagrams shown in figure 4.21.

Using this analysis technique we are liable to over-estimate the number of creation and destruction hops. For example consider the situation shown in figure 4.25.

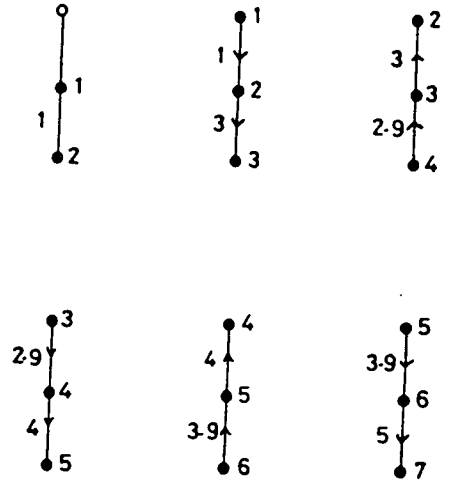


Figure 4.25 Six correlated single site-hopping graphs.

If we use the analysis methods described above then we assign 3 interstitial hops, 2 destruction events and a defect creation event to the hops in this diagram. However, if we look at the third and fifth single site hopping graphs we can see that the time difference between the incoming and outgoing hops

for these sites is small, much less than the average time required for a particle to complete a hop (0.6ps in the case of $SrCl_2$). Given the arbitrary nature in which the time at which a hop occurs was decided it therefore seems sensible to look again at the hop type assigned to the single site hopping graphs when the time difference between the incoming and outgoing hops is small. For example, in the case described above we might decide that it would be more sensible to reassign the third and fifth single site hopping graphs as being interstitial hopping graphs and so describe all 6 hops in figure 4.25 as being interstitial hops.

A simple rule was used to determine whether the single site hopping graph assignments should be changed. All the graphs in which the incoming and outgoing hop times were less than the average hop time were found. The original hopping assignments from these graphs were then compared with the hopping assignments given to the single site hopping graphs on either side of the original graphs in the correlated hopping graph. For graphs in which both the hopping graphs on either side were different from the one given to the test graph, then the hopping assignment given to the test graph was changed. As an example consider the third single site hopping graph shown in figure 4.25. This graph is originally defined to be a vacancy graph whereas the graphs on either side of it are defined to be interstitial graphs (graphs 2 and 4). We therefore define graph 3 to represents a hopping interstitial (similarly for graph 5). This whole process can then be repeated until no more changes in hop assignment occur.

Using the techniques described above we can now create the correlated hopping graph for the simulation by joining together all the single site hopping graphs. Using the correlated hopping we can calculate the numbers of hops which are due to interstitial or vacancy motion and the number of defect creation and destruction events in the simulation run. The results of this analysis are shown in table 4.5.

Table 4.5 Percentages of vacancy hops, interstitial hops, creation and annihilations events in the various MD simulations.

Compound	Temp(K)	%interstitial hops	%vacancy hops	%annihilation	%creation
$SrCl_2$	923	25	57	11	7
$SrCl_2$	1084	40	46	6	8
$SrCl_2$	1216	35	45	10	10
$SrCl_2$	1341	38	44	9	9
$SrCl_2$	1438	39	44	9	8
$SrCl_2$	1525	40	41	10	9
PbF_2	571	37	52	6	6
PbF_2	781	42	39	9	10
PbF_2	841	42	42	8	8
PbF_2	894	41	42	8	9
CaF_2	1222	37	49	5	9
CaF_2	1366	36	44	10	10
CaF_2	1495	43	41	8	8

It is immediately obvious from this table that the hopping analysis predicts that the mobilities of the vacancy and interstitial defects are approximately equal. This is in contrast to the results of the two other analysis methods applied earlier in this section (the analysis of the anion hopping times and the analysis of the hopping trajectories), both of which giving the result that the vacancy defects were significantly more mobile than the interstitial defects.

Examining table 4.5 it can be seen that the results from $SrCl_2$, PbF_2 and CaF_2 are all very similar, the ratio of interstitial:vacancy:creation:destruction hops being typically 41:41:9:9 for simulations in the superionic phase. These results differ slightly from those obtained by Gillan and Dixon, 1978, for $SrCl_2$ at 1522K, who obtained hopping ratios of 28:34:19:19. The discrepancy between these results arises solely from the single site hopping graph reassignment step described above which tends to redefine creation and destruction hops as vacancy or interstitial hops. If this reassignment step is removed then the two sets of results come into close agreement.

By examining the correlated hopping diagram it is also possible to calculate the number of interstitial and vacancy defects present in the simulation at any given time. If we draw a horizontal line across the diagram at some time t, then the number of lines in the hopping diagram which intersect this

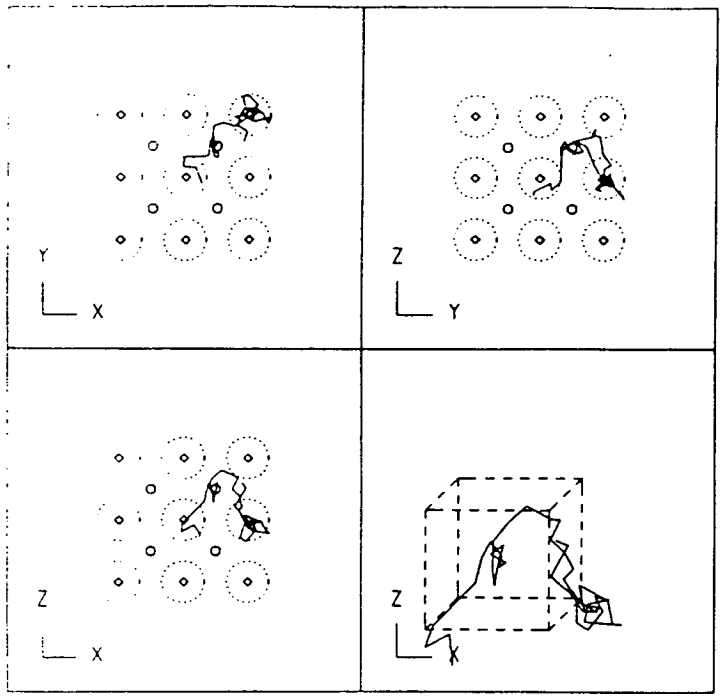
horizontal line gives us the number of defects at time t; if the line has an upward pointing arrow then the defect is a vacancy, if the line has a downward pointing arrow then it is an interstitial. We would expect that the number of vacancies and interstitials counted in this way should be equal, however this is not found to be the case. In general there were found to be a few more vacancies than interstitials. This is because it is possible for the hopping analysis to miss some of the interstitial defects. If we examine the hopping-time distributions (figure 4.16) then we can see that the tails on these distributions are quite long. In fact, even for the simulations of crystals close to there melting points, some of the anion hops can take several picoseconds.

Table 4.6 The maximum time spent by an anion in the vicinity of the empty cube centre site.

$SrCl_2$		PbF_2		CaF_2	
Temp	Lifetime(ps)	Temp	Lifetime(ps)	Temp	Lifetime(ps)
770	5.04	417	0.63	852	3.47
823	2.63	571	1.05	975	0.84
928	2.63	781	1.78	1222	2.31
1084	2.31	841	1.16	1366	2.62
1216	2.21	894	1.05	1495	2.00
1341	2.00				
1438	3.05				
1525	2.00				

The trajectories of anions performing these long lifetime hops bear a striking resemblance to the trajectories of the low-temperature interstitial defects. Some of these trajectories are shown in figure 4.26. Occasionally an anion will leave a lattice site to spend several picoseconds oscillating about the cube centre site before returning to its original lattice site. When the particle leaves the lattice site, the resultant vacancy is free to migrate through the lattice and will be included in the hopping list. The interstitial, however, will not be included in the hopping list, hence the possibility of an imbalance between the number of *observed* vacancies and interstitials. The number of such long lived, cube centre interstitials at high temperature is however very small compared with the total number of interstitials found via the correlated hopping diagram (although the number of such defects is comparable with the number of such defects found at lower temperatures).

a) $SrCl_2$ at 1525K



b) CaF_2 at 1495K

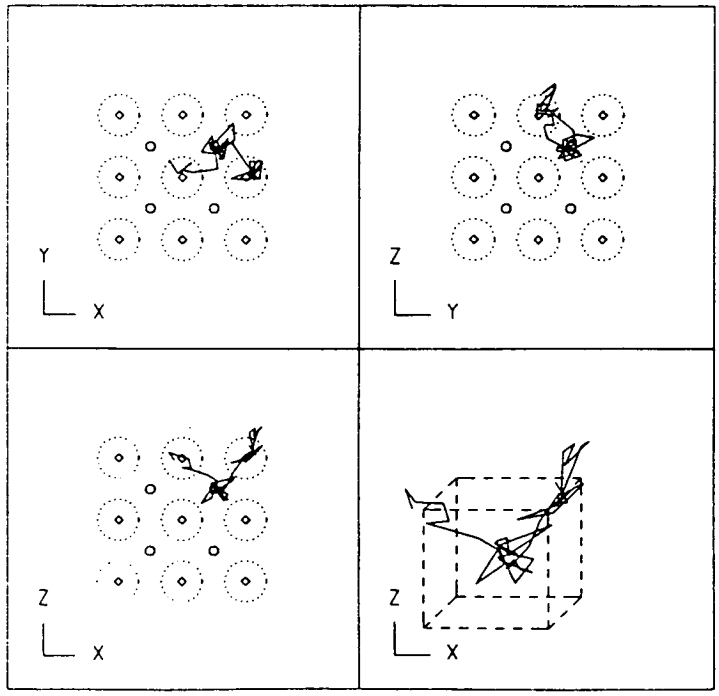


Figure 4.26 The trajectories of some anions from the superionic phases of the three crystals which show behaviour similar to that of low-temperature interstitial anions.

c) PbF_2 at 894K

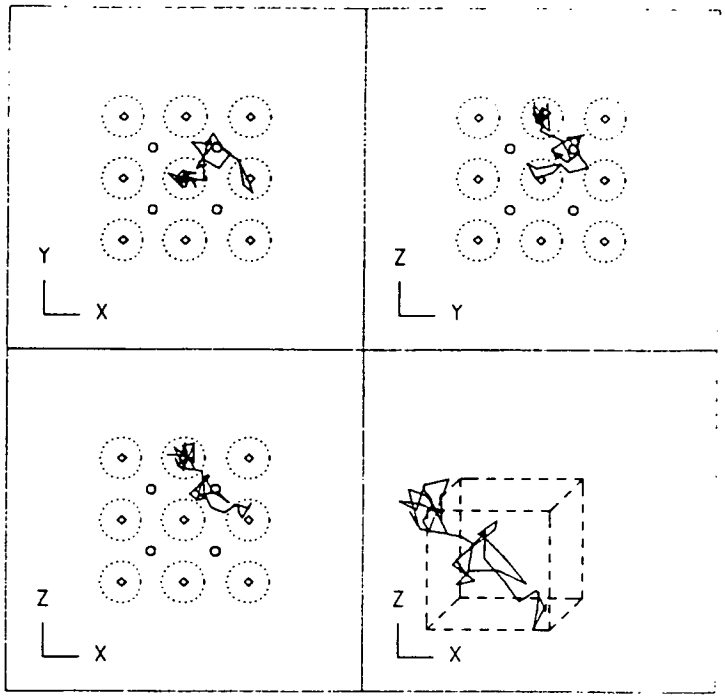


Table 4.7 Percentage of vacancy defects per site in the various simulations

$SrCl_2$		PbF_2		CaF_2	
Temp(K)	% of defects	Temp(K)	% of defects	Temp(K)	% of defects
928	0.08	571	0.12	1222	0.37
1084	0.41	781	1.20	1366	0.41
1216	1.40	841	1.98	1495	2.23
1341	2.11	894	3.26		
1438	2.44				
1525	3.55				

Table 4.7 shows the average number of defects found in the various simulations It can be seen from table 4.7 that although the apparent disorder of the superionic fluorites is high (up to 30% of the anions are not within $a_0/3$ of a lattice site), the percentage of true defects per lattice site is very low, even at the highest temperatures the proportion of vacancies per lattice site is only 2 or 3%. A vast majority of the anions in the superionic phase are simply executing anharmonic vibrations about the equilibrium anion lattice sites.

There are several other quantities that can be calculated from the correlated hopping diagram, for example the mean-free paths and average lifetimes of the vacancy and interstitial defects. These results are shown in table 4.8. It can be seen that as the total number of defects increases with increasing temperature, the tendency is for the mean free path and mean lifetime of the individual defects to decrease. As the concentration of defects increases, the probability that two defects will interact and annihilate each other increases.

Table 4.8 The mean free paths and average lifetimes for the vacancies and interstitials as defined from the correlated hopping diagrams.

	Temp(K)	vacancy mean free path (A)	interstitial mean free path (A)	mean vacancy lifetime (ps)	mean interstitial lifetime (ps)
$SrCl_2$	923	10.8	7.6	13.9	14.5
$SrCl_2$	1084	12.9	11.8	16.9	12.8
$SrCl_2$	1216	11.3	10.6	12.8	10.4
$SrCl_2$	1341	13.2	12.5	12.9	11.9
$SrCl_2$	1438	12.0	12.4	10.6	10.0
$SrCl_2$	1525	11.6	12.7	10.7	9.8
PbF_2	571	12.4	13.0	15.4	15.3
PbF_2	781	9.6	10.1	8.2	9.0
PbF_2	841	12.0	13.3	9.9	9.4
PbF_2	894	10.9	12.5	8.9	8.5
CaF_2	1366	8.7	8.5	8.1	7.6
CaF_2	1495	9.8	11.8	9.1	9.1

4.4.3. The Anion Swapping Mechanism

In section 4.3 an anion swapping mechanism was discussed in which neighbouring anions exchange positions without the intervention of a vacancy or interstitial defect (figure 4.7). These defects are straightforward to pick up in the correlated hopping diagram; they appear as loops connecting two sites in which the two hops are performed by different particles (figure 4.27).

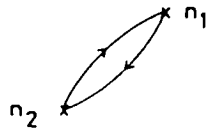


Figure 4.27 The appearance of the anion swapping mechanism in the correlated hopping diagram.

In order to count the number of such anion swapping events we need therefore only look for the two site loops in the correlated hopping diagram and count the number of occasions on which the two jumps are made by different particles (as opposed to one particle making a jump which is immediately followed by a return jump). Although no swapping events were

found in the crystals below the superionic transition temperature, it has been predicted that these swapping events will become important at higher temperatures (Catlow *et al* 1977). However, even at the highest temperatures, the number of such defects was found to be very small. In the simulation of $SrCl_2$ at 1525K only 5 events were found that could be candidates for the anion swapping mechanism. It is therefore unlikely that this mechanism plays an important role in the dynamical behaviour of the fluorite structure superionic conductors.

4.4.4. Conclusions

At temperatures below the superionic transition temperature the defect behaviour of the superionic conductors is straightforward. All the defects generated in the simulations are Frenkel defects; the interstitials residing on the empty cube centre sites and moving via the 111 interstitialcy mechanism, the vacancies residing on the anion lattice sites and moving directly along the 100 directions. The crystal has well defined defects. At higher temperatures the system becomes from much more complicated. Firstly, the time spent by interstitial ions in the empty cube centre sites decreases until it no longer becomes sensible to talk of interstitials residing on cube centre sites. Secondly, interstitial and vacancy motion begin to look very similar. It becomes increasingly difficult to determine whether the hop of any given anion should be regarded as being the result of vacancy or interstitial motion.

Different methods of discriminate between vacancy and interstitial motion can give very different results. If we determine whether a hop is an interstitial or vacancy hop on the basis of its trajectory (which follows naturally from an extension of the low temperature behaviour of the crystal), then we find that the vacancy defects are significantly more mobile than the interstitial defects. If, on the other hand, we analyse the hops in terms of the order of hops into and out of anion lattice sites then we obtain interstitials which are just as mobile as the vacancies. As the results on defect mobility depend critically on the method of analysis employed, we have to be very careful in interpreting the experimental or theoretical results. If we are to compare simulation results with experiments then we must compare like with like. For example, the lattice statics calculations on defect mobilities (see the references in section 4.3)

essentially use the 'hopping trajectory' definition of an interstitial. All the calculations are done assuming that the anion interstitial will spend at least some time at the empty cube centre site. Using this definition of the Frenkel defects the calculations predict that the vacancy defects will be more mobile than the interstitial defects. If, in the simulation, we also use the 'hopping trajectory' defect definition, then the simulations predict a similar result. We should not compare the lattice statics results with those obtained from the correlated hopping analysis of the MD results, the two methods have different definitions of Frenkel interstitial defects, definitions that are not equivalent.

What is clear from the simulations, however, is that even at high temperatures, the anion lattice is still well ordered. Although the number of anions not within $a_0/3$ of an anion lattice site can be very large (over 30% in some cases), the number of true defects per lattice site is very small, of the order 2 or 3%. Most of the anions are simply undergoing anharmonic oscillations about their lattice sites. The time spent by an anion on about a lattice site is always at least an order of magnitude larger than the average hopping time. The simulations also show just similar is the dynamical behaviour of the three compounds being simulated.

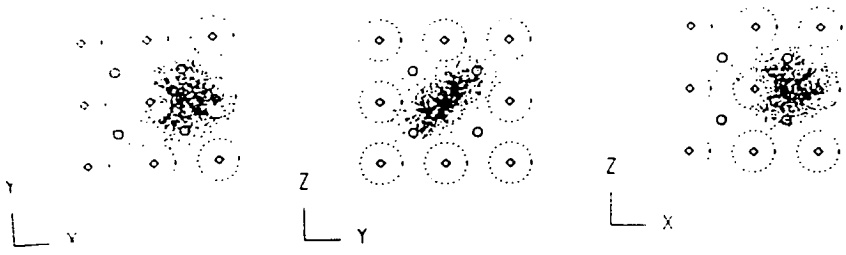
4.5. Hopping Anion Trajectories and the Neutron Scattering Results

Using the hopping list it is possible to find all the hopping ions and examine the trajectories they follow through the empty anion cubes. Taking the 48 configurations we obtain from each MD simulation run (see section 4.1) we can plot the positions in the anion sublattice of all the ions executing hops in the various directions. The positions are only plotted for anions that are not within $a_0/3$ of a regular anion lattice site. The hopping anion densities are shown in figure 4.28. Examining these graphs we can see that the hopping anions travel in well defined regions of the crystal, they are not uniformly distributed throughout the empty anion cubes. The mobile anions are typically restricted to move on the surface of the triangles made up of two adjacent anion lattice sites and the centre of the cube (see figure 4.29).

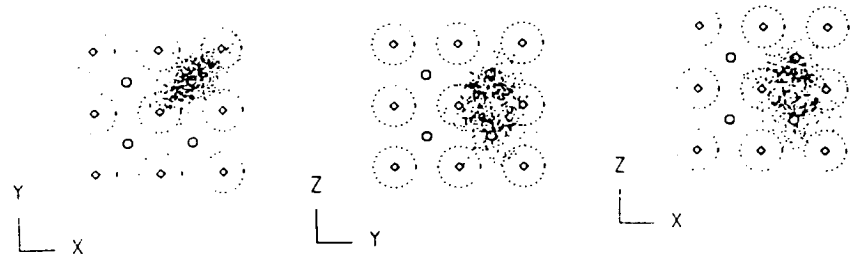
We can examine the hopping anion density more closely by plotting the density contours in the 110 plane connecting opposite cube edges. The contours obtained are shown in figure 4.30. Several features are apparent from

a) $SrCl_2$ at 1525K

100 hops



110 hops



111 hops

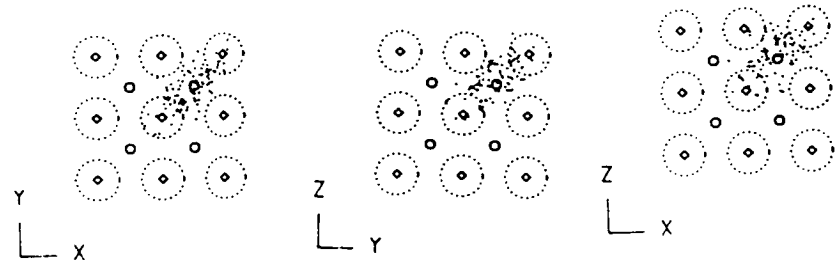
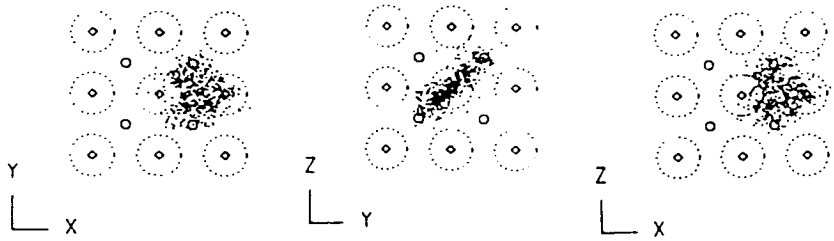


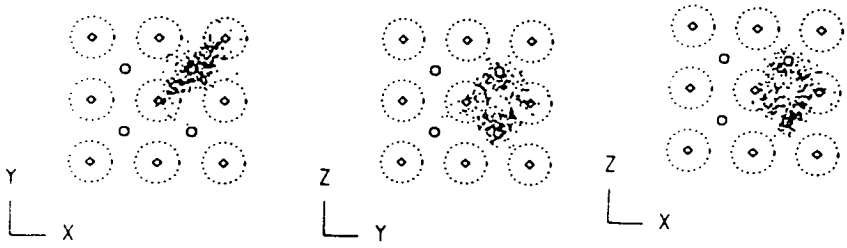
Figure 4.28 The hopping anion density in the anion lattice. The open circles show the projections of the positions of the anion cube-centres.

b) CaF_2 at 1495K

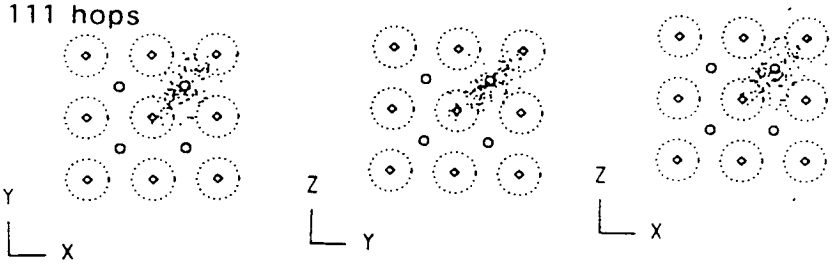
100 hops



110 hops

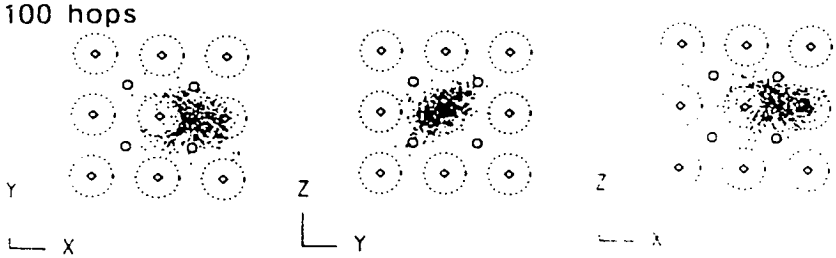


111 hops

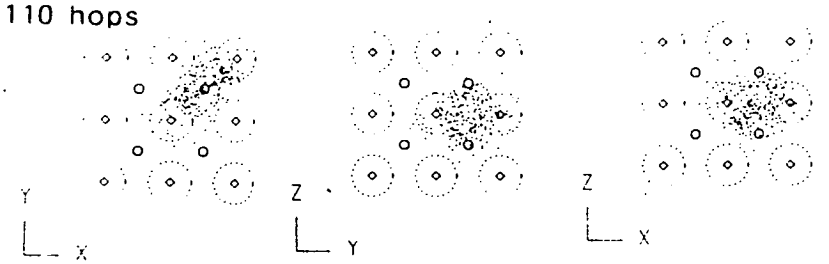


c) PbF_2 at 894K

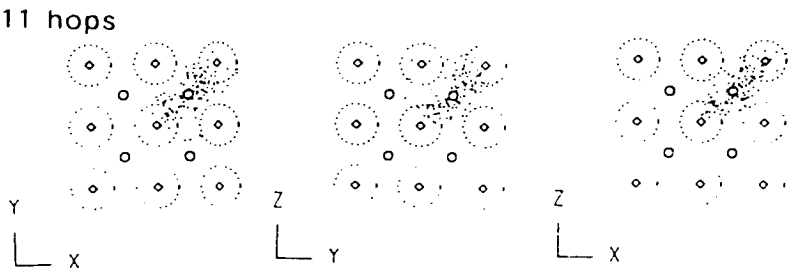
100 hops



110 hops



111 hops



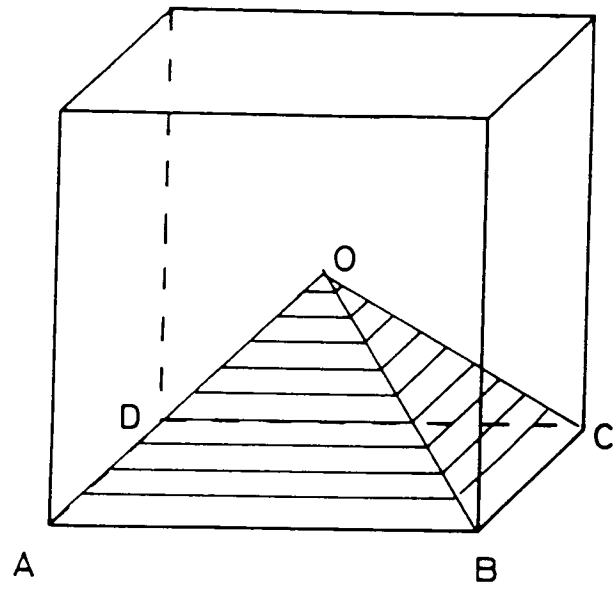
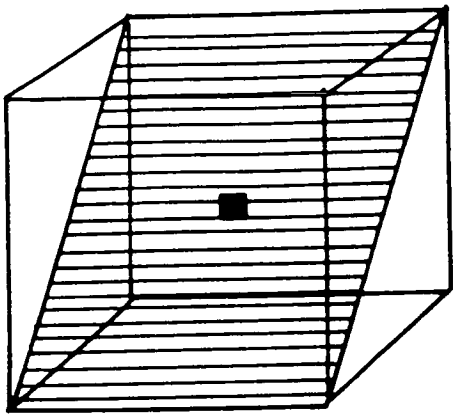


Figure 4.29 The trajectory of an anion hopping from site A to site B will typically lie on the surface of the triangle AOB.

these contours. Firstly we can see that there is no peak in the hopping anion density at the centre of the anion cube. Although there is a significant anion density in this region, the anions pass straight through. Very few mobile anions are trapped in this site, as they would be at temperatures below the superionic transition temperature. Examining the mobile anion density there is no evidence for any position in the anion lattice which could be defined to be the 'interstitial site'. We can also see from the contour plots that most anions undergoing 100 hops travel directly between lattice sites, but there is a significant probability that their trajectories will be relaxed towards the cube centre. Particles hopping in a 110 direction tend to follow a trajectory in which they move towards the cube centre site in a 111 direction and then back down to their new site, again in a 111 direction. There is no maximum in the anion distribution at the cube centre site, if anything the density has a minimum at this point. The anion density for the particles hopping in the 110 direction is, however, fairly large over most of the centre of the cube centre area. The 111 hopping path appears to be a path which directly connects the two sites. The number of 111 hops in any of the simulations is small compared with the numbers of other types of hop and so the density contour plots are not as useful as the 100 and 110 contour plots. More data would be needed before much else could be said about the 111 hopping trajectories.

These results on the trajectories of the hopping anions can be compared with the quasi-elastic coherent neutron scattering results reported by Hutchings *et al* 1984 (see chapter 1). Hutchings *et al* attempted to fit various types of defect cluster to the observed neutron diffraction data. The feature to which they attempted to fit their cluster models was only observed above the superionic transition temperature and so must be a consequence, in some way, of the behaviour of the fluorite structure crystals in the superionic phase. One feature common to all the defect clusters they used in their fitting procedures was the presence of an interstitial site close to the mid-anion position, but relaxed a distance y in the 110 direction towards the empty cube centre site. We can calculate the average position of an anion undergoing a 100 hop an compare it with the position of this 110 interstitial defect in the proposed defect clusters. In table 4.9 the average anion positions as calculated from the simulation are shown along side the positions of the 110 interstitial defects as calculated from the defect clusters.

a) The 110 plane



■ Empty cube centre site

b) $SrCl_2$ at 1438K

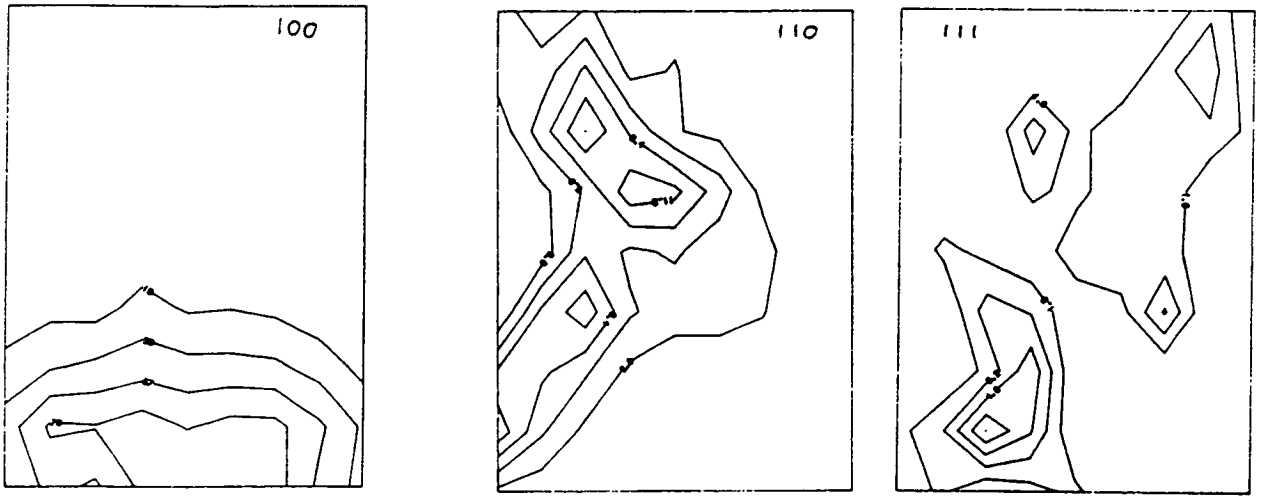
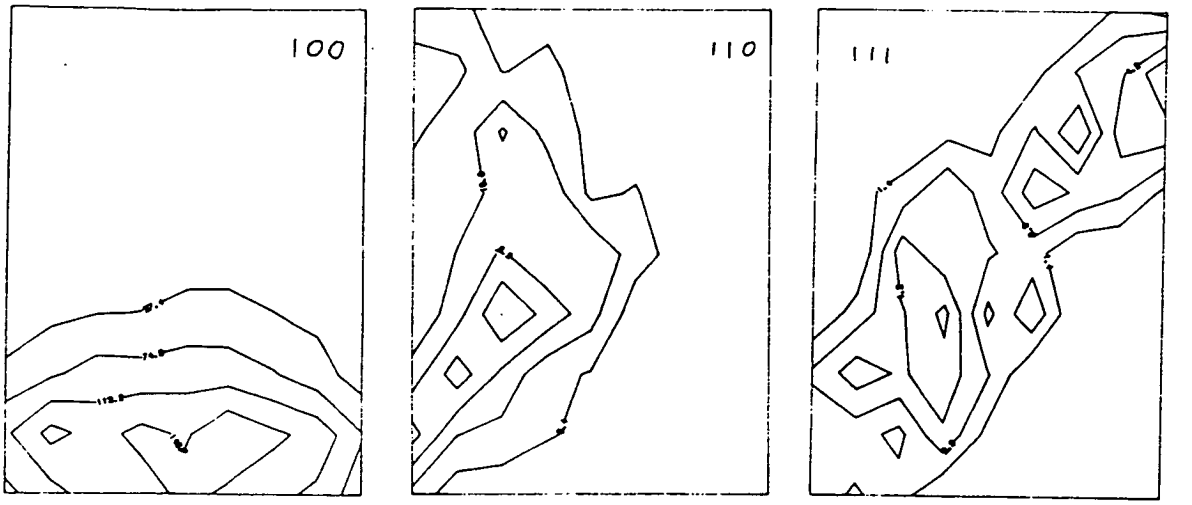
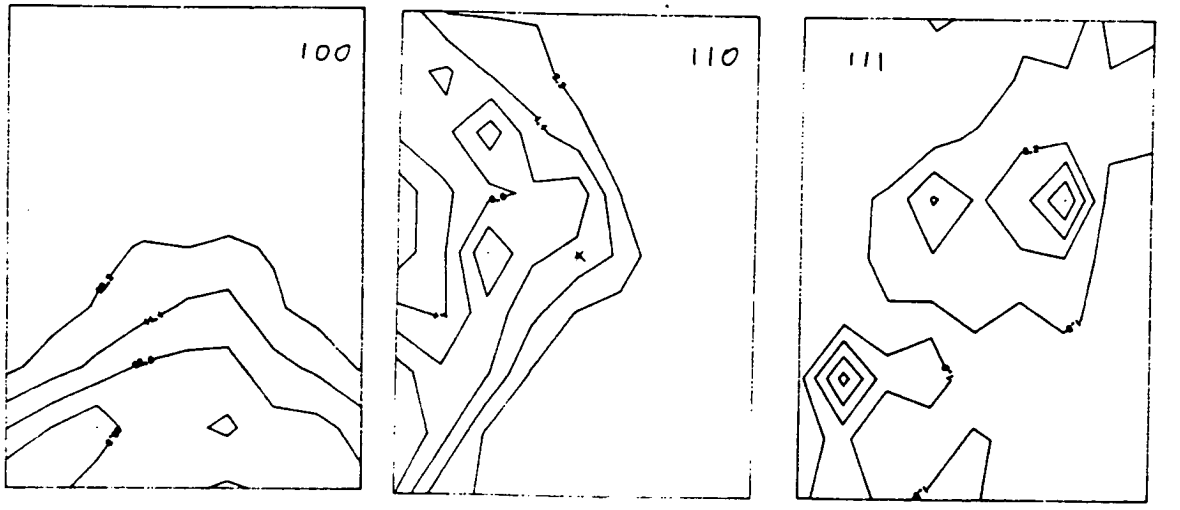


Figure 4.30 Contour plots of the hopping anion density projected onto the 110 plane. The contours are defined such that there are always 4 contours defined between the highest and lowest densities seen in the 110 plane for the various simulations. As the hopping anions are only recorded for anions not within $a_0/3$ of anion lattice sites, the contours will not be accurate within this radius of the corners of the contour plots.

c) $SrCl_2$ at 1525K



d) CaF_2 at 1495K



e) PbF_2 at 894K

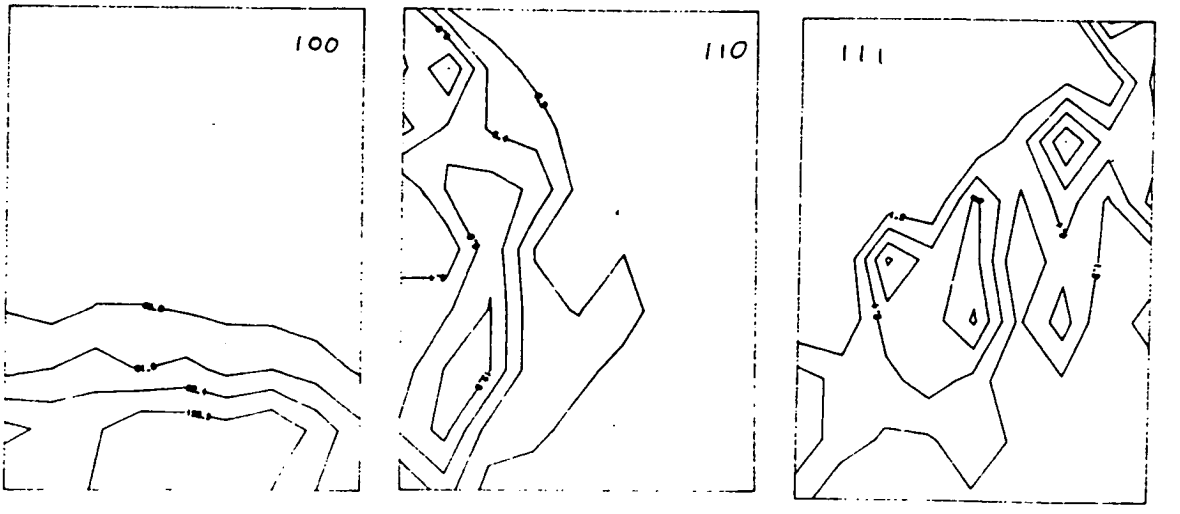


Table 4.9 The displacement of the average anion position from the mid-anion position towards the empty cube centre site in units of a_0 as calculated from the simulations.

Compound	Temp(K)	Disp (yy0)a
$SrCl_2$	1216	0.152±0.003
$SrCl_2$	1341	0.148±0.002
$SrCl_2$	1438	0.158±0.002
$SrCl_2$	1525	0.158±0.002

Experimental: For $SrCl_2$ at 1078K y = 0.16-0.34; best fit y = 0.16

PbF_2	571	0.113±0.007
PbF_2	781	0.142±0.003
PbF_2	841	0.153±0.002
PbF_2	894	0.145±0.002

Experimental: For PbF_2 at 829K y = 0.14-0.34; best fit y = 0.28

CaF_2	1222	0.139±0.004
CaF_2	1366	0.149±0.003
CaF_2	1495	0.160±0.002

Experimental: For CaF_2 at 1473K y = 0.18-0.34; best fit y = 0.22

If we compare the simulation results with the experimental results we can see that the average position of a hopping anion and the position of the 110 interstitial in the defect clusters are in reasonable agreement. The agreement between the average position for the hopping anions and the position of the interstitial defect in the best-fit model is exact for the case of $SrCl_2$ at 1084K, and the MD results from PbF_2 at 841K and CaF_2 at 1495K are within the range of positions obtained from fitting the neutron scattering data, although they are not in agreement with the positions obtained from the best-fit cluster (the 9:1:8 defect cluster).

If we examine the hopping anion density we can see that there is no maximum in the hopping anion density at the position predicted by the defect cluster approach. It therefore seems very misleading to talk of the 110 interstitial position as being an 'interstitial site'; it is not a position in the unit cell at which we would expect to find an excess of anions. Nor are all the anion hops responsible for producing this average position due to interstitial hops, a majority of the anion hops in the 100 direction are, in fact, due to the motion of vacancies directly along a 100 direction. What the neutron scattering

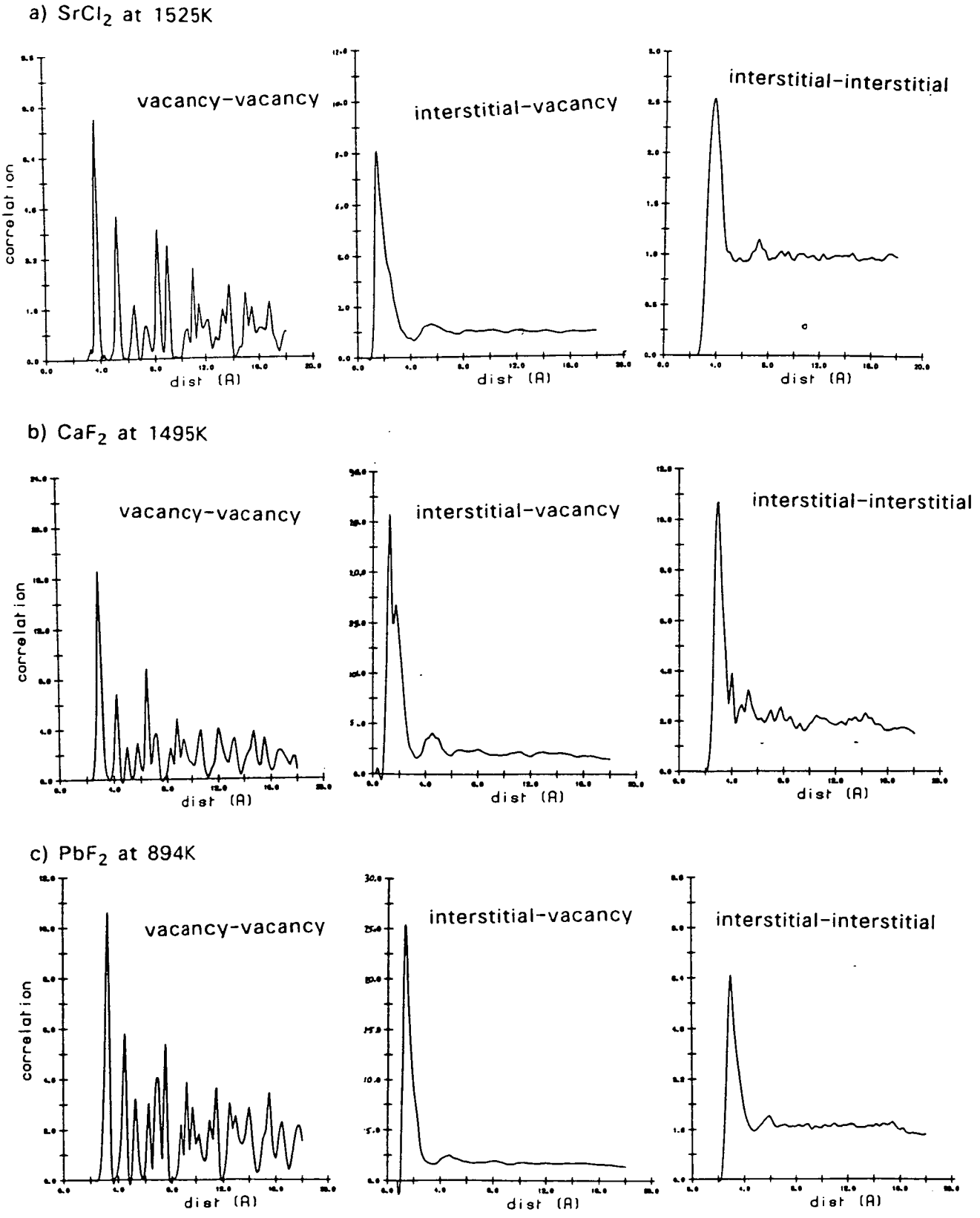


Figure 4.31 The vacancy-vacancy, interstitial-vacancy and interstitial-interstitial rdfs for the highest temperature simulations of the three compounds.

experiments are measuring is the *average* position of the mobile anions and not the position of an interstitial in any Frenkel defect pair.

It should be possible to test the 'average position' interpretation of the neutron scattering data experimentally. If we examine the average positions of the hopping ions in the simulation we can see that as the temperature of the simulation is increased, the average position of a hopping anion moves in towards the empty cube centre site. At present the published neutron scattering results are only available at one temperature for each of the compounds CaF_2, $SrCl_2$ and PbF_2 and so we can not see whether this trend can be seen experimentally. If the neutron scattering experiments could be repeated at a range of temperatures it should be possible to see whether this trend can also be seen in the real crystals.

The defect cluster model predicts the relative positions of the vacancy and interstitial defects of the Frenkel defect pair (see figure 1.8). If we examine the interstitial-vacancy rdfs we can see whether this function has a peak at the required radius (where an interstitial is defined to be any anion that has not been within $a_0/3$ of an equilibrium lattice site for at least .3ps). Figure 4.31 shows the vacancy-interstitial rdf obtained from the simulation of $SrCl_2$ at 1525K. There is a strong peak in this function at $a_0/2$, another weaker peak at $a_0/3$ with the rest of the function being constant. The $a_0/2$ peak can be explained in terms of vacancy motion, the peak corresponding to the position shown in figure 4.5 in which the mobile anion is at a distance of $a_0/2$ from the two vacancies. The peak at $a_0/3$ could well be due to correlated chains of vacancy hops in which two adjacent anions are hopping simultaneously. There is no sign of any peak in this rdf corresponding to the interstitial-vacancy separation predicted by Hutchings *et al* at $2a_0$.

4.6. Lattice Instabilities and the Superionic Transition

Recent lattice dynamics calculations on the alkali halides and alkali earth halides have provided strong evidence for a correlation between the melting transition in these compounds and lattice instabilities in the low temperature phases (a correlation first noted by Hertzfeld and Goeppert-Mayer (1934) in model calculations on solid argon). For example it is argued that the onset of the acoustic instability $c_{44} \rightarrow 0$ is correlated with the melting transition in the

alkali halides; in this picture a liquid is simply defined as a 'solid' which cannot support a shear (Boyer 1981). This correlation has led to the belief that such phase transitions may actually be driven by intrinsic instabilities in the low temperature phases. Given that there is evidence for the role of lattice instabilities in driving phase transitions it seems natural that we should look to see whether similar instabilities are implicated in the superionic transition in fluorite structure compounds.

The techniques of lattice dynamics (Born and Huang 1954) provide an efficient way of looking for such lattice instabilities. Using lattice dynamics we can calculate the frequency of the normal modes of the system as a function of the lattice parameter and see whether any of these modes show anomalous behaviour. Although these calculations do not explicitly include the temperature of the system, the temperature of the system can be implicitly included via the lattice parameter. We can define the temperature of the system to be the temperature at which the effect of thermal expansion is such as to give us a lattice parameter equivalent to the one being used in the lattice dynamics calculation, the larger the lattice parameter the higher the temperature. In treating temperature in this way we are making many approximations, in particular we are neglecting the effect of increased anharmonicity and the formation of defects at high temperatures. Because of these approximations any numbers obtained from the lattice dynamics for large values of the lattice parameter can only be used as rough estimates of the true values we would obtain experimentally.

Boyer (1981) has performed the lattice dynamics calculations for two fluorite structure superionic crystals, $SrCl_2$ and CaF_2 using the Gordon-Kim potentials (Gordon and Kim 1972) to model the inter-particle forces. The main result of these calculations was the discovery of an apparent correlation between the onset of the superionic transition and the softening and ultimate instability of the (100) zone boundary vibration with X_3 symmetry (See figure 4.32). Note that this mode involves correlated motion of chains of anions but involves no motion of the cations (figure 4.33), this motion should be compared with that observed in the MD simulations of fluorite structure superionics (section 4.5) in which it was shown that anion diffusion in the superionic phase is characterised by correlated anion motion predominantly along the 100 directions.

It is possible to calculate the static energy of the X_3 symmetry displacement as a function of the mode amplitude. Figure 4.34 shows two graphs of $\Delta U=U-U_0$ where U_0 is the static energy of the undistorted fluorite lattice versus the amplitude of the X_3 mode. The two graphs show the energy versus amplitude for two different values of the lattice spacing; the first for a lattice spacing of 5.4A corresponding to a temperature of 300K (well below the temperature of the superionic phase) and the second for a lattice parameter of 5.76A corresponding to a temperature of 1400K (well into the superionic phase). By examining these two graphs it is possible to see that increasing the lattice parameter dramatically reduces the barrier for anion hopping as well as reducing the value of the curvature in ΔU at the origin from a positive value for a lattice parameter corresponding to a temperature below the superionic transition to a negative value for a lattice parameter corresponding to a temperature above the superionic transition temperature. The barrier height at T=1400K is reduced to 8400K, sufficiently large however that we would not expect to see unrestricted anion hopping.

In order to investigate these lattice instabilities further we extended the lattice dynamics calculations to include several other superionic systems and other forms for the inter-ionic potentials. The extra systems considered were SrF_2, again using the Gordon-Kim potentials and $SrCl_2$, PbF_2 and CaF_2 using the potentials derived in chapter 3. In all these cases the zone boundary mode with the X_3 symmetry was found to soften and ultimately become unstable as the lattice parameter was increased from values comparable to the low temperature phase to values corresponding to the superionic phase. From these calculations it would appear that the softening of the zone boundary X_3 mode is an inherent feature of compounds with the fluorite structure.

In calculating the frequencies for the X_3 symmetry mode at the zone-boundary for the new compounds mentioned in the previous paragraph it soon became apparent that the situation as regards unstable modes was a lot more complicated than had been previously reported. The frequency squared of every mode on the 100 (or 010 or 001) face of the Brillouin zone-boundary, except for those at the corners (see figure 4.35), was found to become negative at a lattice parameter corresponding to the equilibrium lattice parameter of the superionic phase. In the superionic phase all the zone boundary modes on the 100 faces become heavily damped. Table 4.10 gives

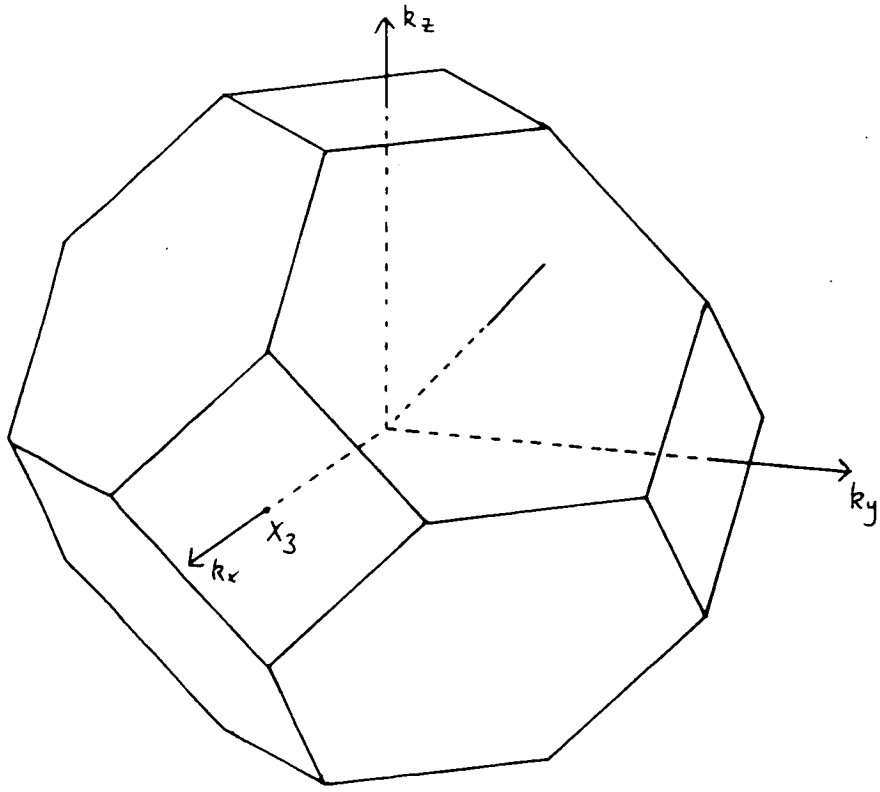


Figure 4.32 The Brillouin zone for a fluorite structure crystal. The diagam shows the positions of the 100 faces and the X_3 points.

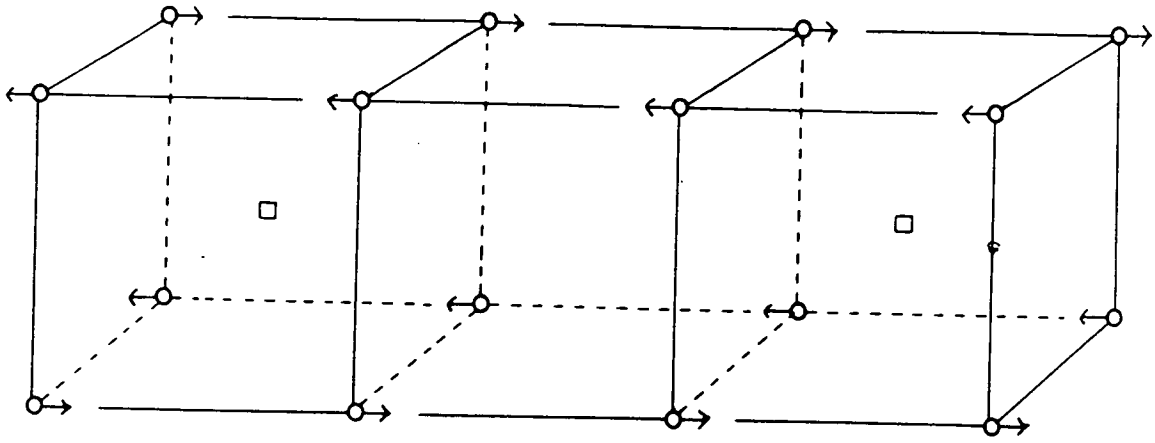


Figure 4.33 Displacements associated with the X_3 mode. Cirles denote the anions and squares the cations.

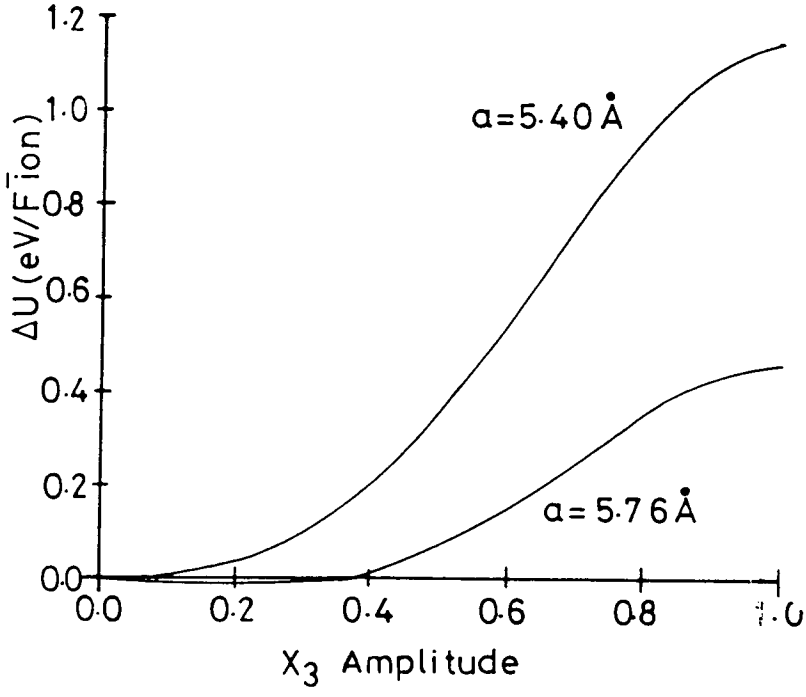


Figure 4.34 The change in the static energy, ΔU, as a function of the X_3 amplitude: solid lines are for y=z=0.

typical values of frequency squared for these modes as a function lattice parameter. When the eigenvectors of these modes where examined it was discovered that all these modes involve motion of chains of anions in the 100 directions with either no motion or only very negligible motion of the cations. From figure 4.35 it can be seen that although it is the X_3 mode at the centre of the 100 faces which has the largest negative value of ν^2, it is not significantly more negative than the values obtained at neighbouring points on this face. We would not expect the X_3 mode to dominate the dynamics in the superionic phase.

Table 4.10 The value of ν^2 as a function of the lattice parameter. The values given are for $SrCl_2$ calculated using the inter-atomic potentials described in section 3.3.

Lattice parameter(A)	ν^2
6.61	77.66
6.88	50.75
6.93	44.78
6.98	38.33
7.04	31.12
7.09	22.36
7.14	7.82
7.20	-18.65
7.23	-24.32

Examining the results of the lattice dynamics calculations it would appear that there might be some correlation between instabilities in certain zone-boundary modes and the onset of the superionic transition. The evidence for this claim is summarised below.

Firstly as the temperature of the fluorite structure crystal becomes comparable to the superionic transition temperature the lattice dynamics calculations predict that all the lowest frequency longitudonal modes on the 100 faces of the Brillouin-zone soften and then become unstable. The lattice parameter at which the instability occurs is similar to the lattice parameter at which the system becomes superionic (relating lattice parameter to temperature as described above). The eigenvectors of all these modes has no component for motion of the cations but does involve the motion of chains of anions, a similar behaviour to that shown by the hopping ions in the superionic phase.

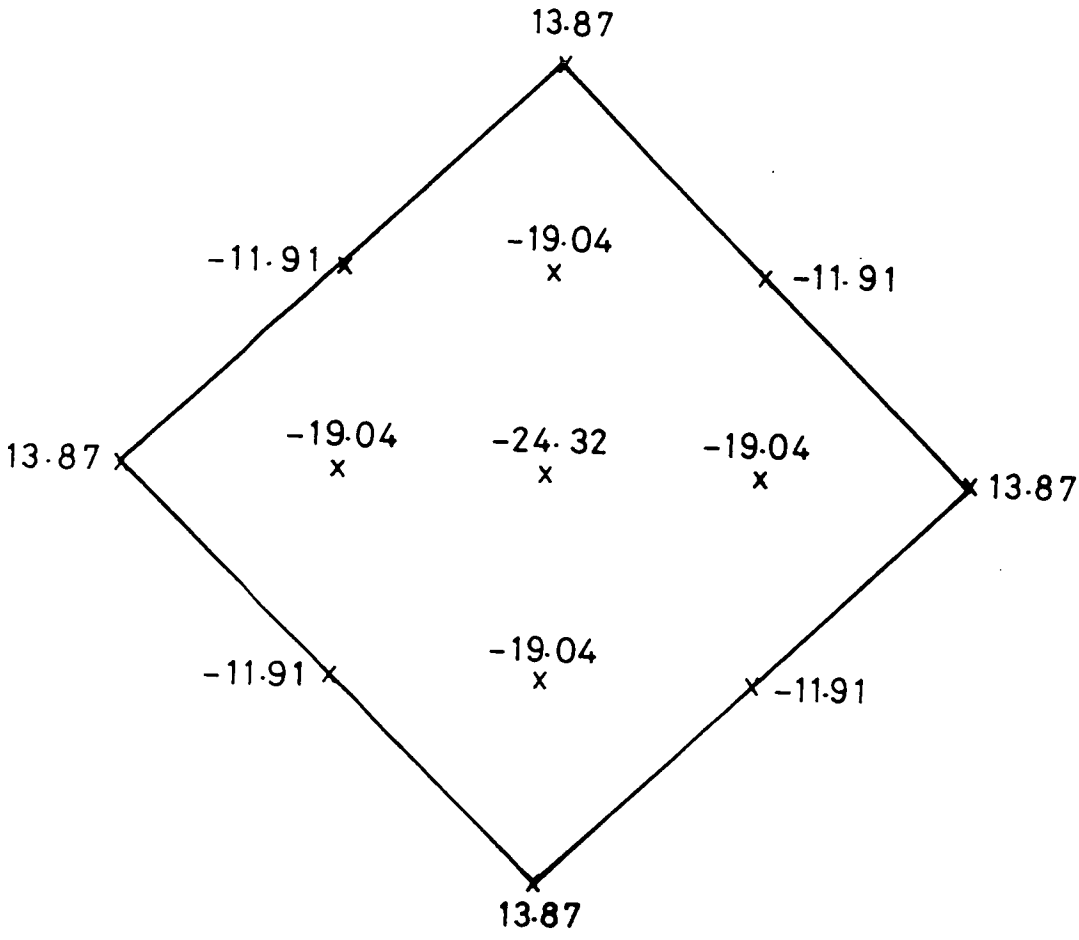


Figure 4.35 The values of ν^2 on the 100 face of the Brillouin-zone for $SrCl_2$ with a lattice parameter of 7.23A.

Secondly there is experimental evidence that some modes become heavily damped as the temperature of the fluorite structure crystals approach the superionic transition temperature. The softening and subsequent instability of the longitudonal modes on the X_3 system is not just a feature of a particular choice of potential, but can also be observed in the real crystals (Kjems private communication).

4.7. Final Conclusions

There are several conclusions that can be drawn from the work presented in this thesis. The first is that the PPPM algorithm works well. It is an efficient, accurate and well-controlled way of modelling the long-range forces in large systems. The algorithm has been used here to model Coulombic forces but it can easily be adapted to model other forms of long-range forces, for example dipole-dipole interactions, simply by changing the form of the Green function used in the calculations. Previously this algorithm has been used for simulations on serial computers (Hockney and Eastwood 1981), the present work has shown that this algorithm can be readily adapted for use on parallel computers (and hence also for use on vector computers). For MD simulations of several thousand particles interacting via long-range forces, the PPPM algorithm provides a way of efficiently calculating the inter-particle forces thereby making simulations of such large systems feasible with present day super-computers, of whatever architecture.

The results obtained from the large MD simulations of the superionic solids have shown that the results obtained from the large systems do not differ markedly from previous results obtained from simulations of systems an order magnitude smaller than those used here. The conduction mechanisms, diffusion coefficients and crystal structure obtained from the different simulations are all very similar. However the simulations of the large systems can provide results that could not be obtained from the smaller systems; in going to large scale simulations we can do more than just improve statistics. For example, using the large systems it has been possible to investigate the defect behaviour of the fluorite structure crystals at temperatures below the superionic transition temperature.

The defect concentration of the crystals at these temperatures is so small that

it is not possible to see these low temperature defects without introducing them into the system artificially. By examining the behaviour of these low temperature defects it has been possible to gain some understanding of the changes involved in the defect dynamics as the crystals moves into the superionic phase. At temperatures below the superionic transition temperature the Frenkel defects are well defined. The interstitials and vacancies have very different behaviour; the interstitials residing for several picoseconds on the empty cube centre sites and moving via an interstitialcy mechanism, the vacancies residing on the anion lattice sites and moving via a rapid hop directly along an anion cube edge. As the temperature of the crystal is increased the time spent by an interstitial in the cube centre site decreases until at temperatures just below the transition temperature, the interstitials no longer 'reside' at the cube centre site, they simply pass through it. At this point it becomes difficult to know exactly how to define 'interstitials' and vacancy defects. There is no longer a unique way of deciding which anion hops are due to interstitial motion and which to vacancy motion. Different ways of assigning hops to defect types can give very different numbers of each type of hop, hence leading to the differences in the defect mobilities given by the different analysis techniques.

Examining the results given in the thesis it can be seen that the behaviour of all three fluorite structure compounds studied is very similar. The form of the rdfs, the anion hopping behaviour and the low temperature defect behaviour are all very similar. This suggests that the behaviour of the fluorite structure superionics is very much a function of the fluorite lattice rather than the exact form of the interionic potentials.

The results presented here can also throw light on the neutron scattering results of Hutchings *et al* 1984. Hutchings *et al* interpret the neutron scattering results as providing evidence for 'defect clusters' in the superionic ~~clusters~~ crystals. However, the results in section 4.5 cast some doubt on this intrepretation in favour of the interpretation suggested by Gillan and Wolf (Gillan 1986, Gillan and Wolf 1985). Gillan suggests that the neutron scattering data can be described totally adequately in terms of a hopping model, in which all the defects reside on lattice sites and the observed neutron diffraction spectrum arises from the relaxation of the anion lattice about such defects. Examining the hopping anion densities produced from the simulations it can be seen that

there are no peaks in the density distributions in the positions predicted by Hutchings *et al* for the position of the interstitial site; the only peaks in the anion density distributions are actually at the lattice sites. What is also obvious from the present results, is that the mid-anion interstitial site predicted by Hutchings *et al* is not a true site, but simply the average position of the hopping anions.

If we interpret this mid-anion site as being the average position of the hopping anions, rather than being an interstitial site, then we get a very different view of the conduction mechanism in superionic solids. It is no longer to correct to think of the interstitials being tied up in clusters whilst the vacancies carry the current (Moscinski and Jacobs 1985). The model which interprets the coherent scattering in terms of interstitial motion and incoherent scattering in terms of vacancy motion has been shown to be too simplistic. The interstitial position in the cluster models has been shown to be the average position of hopping anions, both anions hopping as a result of vacancy motion and anions hopping as a result of interstitial motion (independent of the criteria used to define interstitials and vacancies), the quasi-elastic peak is not due simply to interstitial motion. We have instead a picture of anions hopping between well-defined lattice sites, some of the hops being defined as vacancy hops, some as interstitial hops (although the distinction between vacancies and interstitial hops is very blurred), with the anion lattice relaxing around the mobile anions. Given the difficulty in distinguishing between vacancy and interstitial hops, it does not seem to be very sensible to be too concerned about the relative mobilities of vacancy and interstitial point defects, the relative mobility is very much a function of the analysis technique used to measure it.

The only difference between this interpretation and the neutron scattering results is in the location of the anion vacancy with respect to the hopping anion, this vacancy being given a distinct location in either the 3:1:2 or 9:1:8 clusters described by Hutchings *et al*. There is no evidence in the present simulation that this vacancy is in any particular location with respect to the mobile anion (see the defect-defect rdfs in section 4.5). It has been suggested however (Gillan 1986), that this additional vacancy will have little effect on the calculated scattering from the 3:1:2 and 9:1:8 clusters and so will not effect how well the defect clusters fit the observed neutron scattering data.

This present work also suggests a possible mechanism for driving the superionic transition in the fluorite structure superionic compounds. Most compounds with the fluorite structure seem to exhibit some form of superionic behaviour. If we examine the dynamics of the fluorite lattice then we find, at least for all the different potentials tried so far, that there is an inherent instability in the boundary of the Brillouin zone. As the lattice parameter of the lattice is increased (corresponding to increasing the temperature of the crystal) then the entire 100 face of the Brillouin zone softens and eventually becomes unstable. The lattice parameter at which this lattice instability occurs corresponds closely to the lattice parameter at which the real crystals become superionic. Also the eigenvectors of all the unstable modes involve only anion motion along chains in a 100 direction. The evidence for the role of these unstable modes in driving the superionic phase transition is, at the moment, purely circumstantial. For the theory to have a more solid grounding some form of scattering experiment will need to be performed.

4.7.1. Further Work

The present work does not include any neutron scattering results from the simulation. One of the main reasons for setting up the large scale simulation was to improve the resolution of the neutron scattering data, large systems enable many more points in reciprocal space to be examined. Neutron scattering data could through much light on the behaviour of the simulation, in particular it could provide support for the defect interpretation of the quasi-elastic coherent scattering results and the unstable mode picture of the phase transition. A major priority for further work on these simulations must therefore be to calculate the various scattering cross-sections for the simulated systems.

References

Alder B and Wainwright TE,1959,J. Chem. Phys.,**31**,459-466
Anderson H C,1980,J. Chem. Phys.,**72**,2384-93
Arnold VI,1963, Upsekhi Matem. Nauk,**18**,13
Axe J D ,1965,Phys. Rev.,**139**,A1215
Beeman D,1976,J. Comput. Phys.,**20**,130
Beevers Ca and Ross MAS,1937,Z Krist,**97**,59
Bendall P J,1976,Thesis,Oxford University
Bendall PJ and Catlow CRA,1980,J Phys, Paris,**41**,C6-61
Beniere M, Chemla M and Beniere F,1979, J Phys Chem Solids,**40**,729-37
Berne BJ,1971,*Physical Chemistry, an Advanced Treatise* vol. VIIIB, (Acedemic Press, London) p 621
Berry MV in *Topics in Non-Linear Dynamics*, ed. S. Jorna, AIP Conference Procedings (No. 46)
Born M and Mayer J E,1932,Z. Phys.,**75**,1
Born M and Huang K,1954,*Dynamical Theory of Crystal Lattices*, (Oxford University Press, London)
Bowler KC and Pawley GS,1984, Proc. IEEE,**71**(1)
Bowler KC, Chalmers DL, Kenway RD, Kenway A, Pawley GS, Wallace DJ, 1984,Nucl. Phys.,**B240**,213
Boyer LL,1980,Phys Rev Lett,**45**,1858-1862
Boyer LL,1981,Phys. Rev. B,**23**,3673
Boyer LL,1984,in *Thermal Expansion 8* ed T A Hahn,(Plenum)
Bradley JN and Greene PD,1967, Trans Faraday Soc,**63**,424
Brigham OE,1974, *The Fast Fourier Transform*, (Prentice-Hall, Englewood Cliffs, NJ)
Brown D and Clarke JHR,1984,CCP5 Quarterly, No. 11,11
Carr VM, Chadwick AV and Saghafian R,1978, J Phys C, **11**,L637
Catlow CRA and Norgett MJ,1973,J Phys C,**6**,1325
Catlow CRA and Mackcrodt WC,*Computer Simulation of Solids*, eds. CRA Catlow and WC Mackrodt (Springer-Verlag,Berlin,1982)
Catlow CRA and Hayes W,1982,J Phys C,**15**,L9-13
Catlow CRA, Norgett MJ, Ross TA,1977,1977,J Phys C,**10**,1063
Catlow C R A,Dixon M and Mackcrodt W C,*Computer Simulation of Solids*, eds. C R A Catlow and W C Mackrodt (Springer-Verlag,Berlin,1982)
Chandra S,1981,*Superionic Solids, Principles and Applications*, (Elesevier North Holland, Amsterdam)
Chudley CT and Elliott RJ,1961,Proc Phys Soc,**77**,353-61
Clausen K,Hayes W,Hutchins M T,Kjems J K,Schnabel P and Smith C, 1981,Solid St. Ionics,**5**,589
Corbin N,1985,CCP5 Quarterly, No. 16,86
Corish J and Jacobs PWM,1973,*Surface and Defect Properties of Solids*, Vol 2, Ch 7 (London: Chemical Society
Davies ST,1984, in *Supercomputers and Parallel Computation*, ed. DJ Paddon, (Clarendon Press, Oxford)
De Leeuw SW and Thorpe MF,1986,J. Non-Cryst. Solids,**75**,393-398
De Raedt B, Sprik M, Klein ML,1984,J. Chem. Phys.,**80**,5719
Derrington CE, Lindner A, O'Keefe M,1975, Solid State Chem,**15**,171
Dick B G and Overhauser A W ,1958,Phys. Rev.,**112**,90
Dickens M H,Hutchins M T,Kjems J and Lechner R E,1978, J Phys C,**11**,L583
Dickens MH, Hayes W,Smith C and Hutchings MT,1979, in *Fast Ion*

Transport in Solids, ed. P Vashishta, JN Mundy and GK Shenoy (Elsvier North Holland, Amsterdam)
Dickens MH, Hayes W, Hutchings MT and Smith C,1979,J Phys C, **12**,L97-102
Dickens MH, Hayes W,Smith C,Hutchings MT and Kjems JK,1979, in *Fast Ion Transport in Solids*, ed. P Vashishta, JN Mundy and GK Shenoy (Elsvier North Holland, Amsterdam)
Dickens HH, Hayes W, Schabel P, Hutchings MT, Lechner RE and Renker B,1983,J Phys C,**16**,L1
Dixon M and Gillan MJ,1978,J Phys C,**11**,L1901-17
Dixon M and Gillan M J,1980,J. de Phys.,C6,supp. to No 7,**41**,C6-C24
Dove MT and Pawley GS,1983,J. Phys. C:Solid St. Phys.,**16**,5969
Dove MT, Pawley GS, Dolling G, Powell BM,1986, Mol. Phys.,**57**,865
Duane S and Kogut JB,1985,Phys Rev Lett,**55**,2774-2777
Dworkin AS and Bredig MA,1968,J Phys Chem,**72**,1277
Etsall TH and Flengas SN,1970, Chem Rev,**70**,339
Ewald PP,1921,Ann. Phys. (Ger),**64**,253
Faux I and Lidiard A B,1971,Z. Naturf.,**A26**,62
Fitterer GR,1966, J Metals,**18**,960
Fukugita M, Kaneko T, Ukawa A to appear in Nuclear Physics B
Funke K,1976, Prog in Solid State Chem,**11**,345
Geller S,1973, in *Fast Ion Transport in Solids*, ed. W Van Gool (North Holland, Amsterdam)
Gillan MJ,1986a, to be published in J. Phys. C
Gillan MJ,1986b, to be published in J. Phys. C
Gillan MJ and Richardson DD,1979,J Phys C,**12**,L61-5
Gillan MJ and Dixon M,1980,J. Phys C,**13**,1901-17
Gillan MJ and Dixon M,1980,J. Phys C.,**13**,L835
Gillan MJ and Wolf D,1986,Phys Rev Lett,**55**,1299-302
Gordon RE and Strange JH,1978,J Phys C,**11**,3213
Gordon RG and Kim YS,1972,J Chem Phys,**56**,3122
Gross S,1976, Energy Conversion **15**,95
Hansen JP McDonald IR,1976, *Theory of Simple Liquids*,(Acedemic Press,London) p45
Hansen JP and McDonald IR,1976,J Phys C: Solid St Phys,**7**,L384-6
Hardy J P and Flocken J W ,1970,CRC Crit. Rev. Solid State Sci., **1**,605
Hardy J R and Karo A M, *Lattice Dynamics and Statics of Alkali Halide Crystals* (Plenum,New York,1979)
Hayes W,1978,Contemp Physics,**19**,469-86
Hertzfeld KF, Goeppart-Mayer M,1934, Phys Rev,**46**,995
Heyes DM,1983,JCS Faraday Trans.,**79**,1741
Heyes DM,1985,CCP5 Quarterly, No. 19,43
Hockney RW and Eastwood JW,1981, *Computer Simulation Using Particles*, (McGraw-Hill,New York)
Hodby J W, *Crystals with the Fluorite Structure*, ed. W Hayes (Clarendon,Oxford,1974)
Hood GM and Morrison JA,1967,J Appl Phys,**38**,4796-802
Huggins M L and Mayer J E,1933,J. Chem. Phys.,**1**,643
Huggins RA,1977,Electrochim Acta,**22**,773
Hurditch RJ,1976,Extended Abstracts - Electrochem Soc. Meeting, Las Vegas, Abstract No 198 p. 524
Hutchings MT,Clausen K,Dickens MH,Hayes W,Kjems JK,Schnabel PG and Smith C,1984,J. Phys. C:Solid St. Phys.,**17**,3903

International Computers Ltd.,1980, DAP: Introduction to FORTRAN programming,Technical Publication 6755, Software Distribution Centre, Putney, London
International Computers Ltd.,1980, DAP: FORTRAN language, Technical Publication 6918, Software Distribution Centre, Putney, London
Jacobs PWM and Ong SH,1976, J Phys, Paris,**37**,C7-331
Jacobs PWM and Ong SH,1980, Cryst Lattice Defects **8**,177
Jacobs PWM, Sahni VC, Vempati CS,1984,Phil Mag A,**49**,301
Jacucci G and Rahman A,1978,J Chem Phys,**69**,4117-25
Kim Y S and Gordon R G,1974,J. Chem. Phys.,**60**,4332
Kiukkola K and Wagner C,1957,J Electrochemical Soc,**104**,308
Kolmogorov AN,1954,Dokl. Akad. Nauk SSSR,**98**,527
Koto K, Shultz H and Huggins RA,1980,Solid State Ionics,**2**,43-6
Lebowitz AJ, Percic JK, Verlet L,1967,Phys. Rev.,**153**(1),250-254
Lowndes R P ,969,J. Phys. C: Solid State Phy.,**2**,1595
March N H,Richardson D D and Tosi M P,1980,Solid State Commun., **35**,903
Mayer J E,1933,J. Chem. Phys.,**1**,270
Metropolis N, Rosenbluth AW and MN, Teller AH and E,1953, J Chem Phys,**21**,1087
Moscinski J and Jacobs P W M,1985,Proc. R. Soc.,A**398**,173
Moser J,1962, Nachr. Acad. Wiss. Gottingen, Math. Phys.,K1,11a
Murch GE and Thorn RG,1977,Phil Mag,**35**,493,517,529
Norgett M J,1974,Harewell Report AERE-R7650
Nose S and Klein M L, J. Chem. Phys.,**81**,511-519
O'Keefe M,1976,in *Proc Int Conf on Fast Ion Conductors, Schenectady,* ed. GD Mahan and WL Roth (Plenum, New York)
O'Keefe M and Hyde BG,1976,Phil Mag,**33**,219-24
Pancharatnam S, Huggins RA, Mason DM,1975,J Electrochem Soc,**122**,869
Parinello M and Rahman A,1980,Phys. Rev. Lett.,**45**,1196-99
Parinello M and Rahman A,1981,J. Appl. Phys.,**52**,7182-90
Pawley GS,1981,Mol. Phys.,**43**,1321
Pawley GS and Dove MT,1983,Chem. Phys. Lett.,**99**,45
Pawley GS, Swendson RH, Wallace DJ, Wilson KG,1984,Phys Rev,**B29**,4030
Peters C and Klein ML,1985,Mol. Phys.,**54**,895
Piazzini S and Bianchi G,1973,Chim et Indust,**54**,994
Powles JG,1986,CCP5 Quarterly, No. 21,25
Rahman A,1964,Phys. Rev.,**A136**,405
Raleigh DO,1976, in *Electrode Processes in Solid State Ionics,* ed M Kleitz and J Dupoy (D Reidel Pub Co. Holland)
Refson K,1985,Physica **131B**,256
Sangster M J and Dixon M,1976,Adv. Phys.,**25**,247
Schnabel P,Hayes W,Hutchings MT,Lechner R and Renker B,1983, Radiation Effects,**75**,73-77
Schofield P,1973,Comput. Phys. Commun.,**5**,17
Shapiro SM,1976,in *Superionic Conductors,* ed. GD Mahan and WL Roth, (Plenum, New York)
Shapiro SM and Reidinger F,1979, in *Physics of Superionic Conductors, Topics in Current Physics No 15,* ed. MB Salamon (Springer, Berlin)
Sproule RT,1974, Mater Engg,**79**,48
Swartztrauber PN,1984,Parallel Comp,**1**,45-63
Takahashi T, 1973, in *Fast Ion Transport in Solids,*ed W van Gool (North Holland, Amsterdam)

Tubandt C,1932,in *Handbuch der Exp Physik XII Part 1*,ed. W Wien and F Harms (Akadem. Verlag, Leipzig)
Tubandt C and Lorenz E,1913,Z Phys Chem,**87**,513-43
Vesely F,1978,*Computerexperimente an Flussigkeitmodellen*, (Physik Verlag,Weinheim) p 45
Walker A B,Dixon M and Gillan M J,1982,J Phys C,**15**,4061
Weber N,1974, Energy Conversion, **14**,1
Weissbart J and Smart WH,1970, in *Extended Abstracts - Electrochem. Soc. Fall Meeting, Atlantic City*
Whittingham MS and Huggins RA,1972,NBS Special Publication,**364**,139
Wolf ML, Walker JR and Catlow CRA,1984,Solid State Ionics,**13**,33-38
Yao YFY and Kummer JT,1967, J Inorg Nucl Chem,**29**,2453

I. Introduction to the DAP and DAPFORTRAN

I.I. The Computer

The Distributed Array Processor (DAP) is an array processor made up of a number of processing elements (PEs) each with its own local storage. Each PE obeys the same instruction simultaneously but uses its own local data, thus operating in a Single Instruction Multiple Data (SIMD) mode. The N_p^2 (N_p=64) processing elements are considered as lying on a square N_pxN_p grid with the local store of each PE being regarded as a third dimension to give a structure of the form as shown in figure 1.

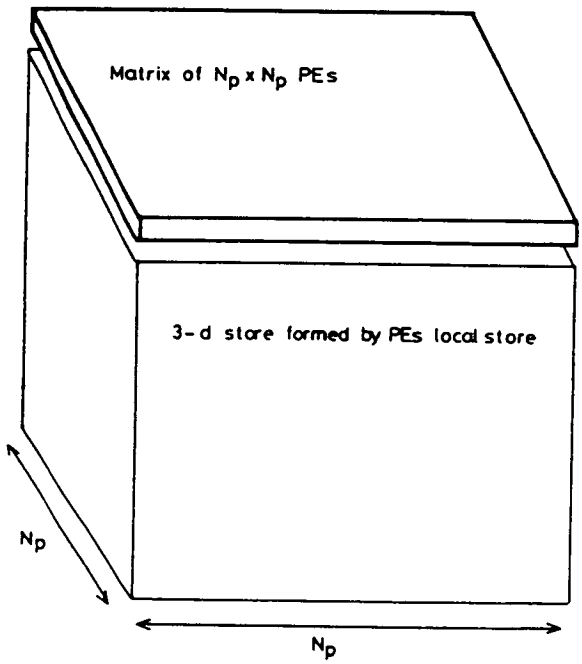


Fig. 1. PE matrix and DAP storage.

Each PE has connections with its four nearest neighbours . These connections allow for planar data movements in which all the data in one plane is simultaneously shifted in any one of the four nearest neighbour directions. The algorithms require that there be two choices for the geometry of the data plane. With "planar" geometry in operation data shifted off the edge of the plane disappears and data entering an edge is set to zero. With "cyclic" geometry in operation data shifted off one edge reappears at the opposite edge. By using a bit plane of 'activity control' bits (the activity control plane

can be thought of as a logical mask) the PEs can be given limited local control. By using such a logical mask, operations at PEs whose corresponding mask bit is .FALSE. can be suppressed. Logical masks can also be used to "merge" two data planes. Where a PE has a mask bit set to .TRUE. it takes data from the first data plane and where it is set to .FALSE. it takes data from the second data plane.

I.II. The DAPFORTRAN programming language

DAP FORTRAN is similar in many ways to FORTRAN IV but with extended data structures and instructions to handle the parallelism of the DAP. Algebraic and logical data is organised into three main forms; scalars, vectors and matrices. Scalars can be treated essentially as scalar variables in FORTRAN IV. Vectors are 1-d arrays of 64 data items stored horizontally on single bit-planes. Matrices are 64x64 data arrays stored vertically (see figure 2).

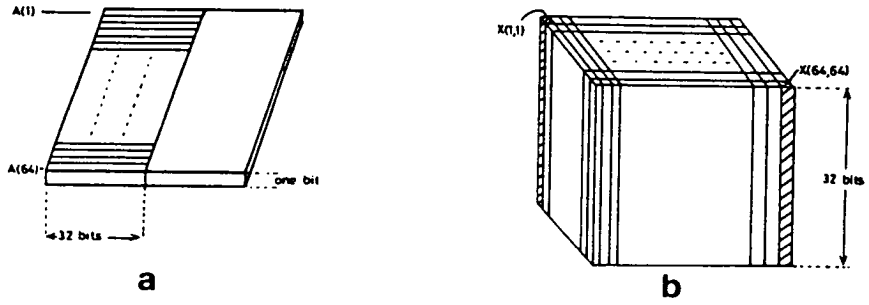


Fig. 2. (a) Storage of the REAL * 4 vector $A()$ on the DAP; (b) Storage of the REAL * 4 matrix $X(,)$ on the DAP.

In DAP FORTRAN a real 64x64 matrix would be declared using the statement

```
REAL X(,)
```

where the (,) means, by definition, that X is a 64x64 matrix. If X(,), Y(,) and Z(,) are declared as real matrix variables then a statement such as

```
X(,) = Y(,) + Z(,)
```

would cause all 4096 elements in Y(,) to be added to all 4096 corresponding

elements of Z(,) simultaneously, the results being placed in X(,), thus 4096 additions are performed simultaneously by a single add instruction which has been broadcast to all the PEs. It is not necessary to write a matrix variable as X(,), for example, writing it simply as X is also permissible in DAP FORTRAN providing it has been declared as being of type matrix. However writing it as X(,) does make it obvious that the variable being considered is of type matrix. If L(,) is defined to be a logical matrix (one bit-plane) then the instruction

```
X(,) = MERGE(Y(,),Z(,),L(,))
```

has the effect of assigning to the matrix X(,) the data in Y(,) where the corresponding mask bit is .TRUE. and the data in Z(,) where the corresponding mask bit is .FALSE. As described earlier, logical masks can also be used to suppress operations on certain PEs. The instruction

```
X(L(,)) = Y(,) + Z(,)
```

would first calculate all 4096 elements of the matrix Y(,) + Z(,), however these calculated values will only be assigned to elements of X(,) where the corresponding mask bit is .TRUE. Where the mask bit is .FALSE. the corresponding element of X(,) will be unchanged. The other useful instructions are those that shift data on a data plane. The instruction

```
X(,) = SHNC(Y(,),1)
```

(**SH**ift **N**orth **C**yclic) would assign to X(i,j) the value of Y(i+1,j), the coordinates being calculated modulo 64 as cyclic geometry is selected.

The DAPFORTRAN language also allows for a matrix to be treated as a 4096 vector. A long-vector is stored in the same way as a matrix, but is considered as being a vector of 4096 elements as opposed to a 2-d 64x64 array. The first column of 64x64 PEs stores the first 64 elements of the vector, the second column stores the next 64 elements etc. There are two DAPFORTRAN shift instructions provided to operate on the long-vector data-types. The instructions SHRC and SHLC (**SH**ift **R**ight **C**yclic and **SH**ift **L**eft **C**yclic) enable the elements in the long vector to be shuffled one place to the left or right, eg the instruction:

```
X(,) = SHRC(Y(,),1)
```

would assign to X(i) the value of Y(i-1). The long-vectors are declared in exactly the same way as matrices and can be treated in exactly the same way as matrices; long-vector instructions work on matrices and vice versa. For example we can refer to the element X(1,1) simply as X(1), similarly X(1,2) is equivalent to X(65). An instruction such as:

```
X(,) = SHRC(SHNC(Y(,),1),1)
```

is perfectly valid DAPFORTRAN. The DAPFORTRAN language is described in more detail in two ICL technical manuals, DAP: Introduction to FORTRAN programming (1980) and DAP:FORTRAN language (1980).

II. Multi-dimensional FFTs on Highly Parallel Computers

II.I. Introduction

There are many examples of problems in several different fields where it is necessary to be able to compute quickly multi-dimensional Fourier transforms (FTs). Examples of the use of multi-dimensional FTs. range from scattering theory to fluid dynamics, solution of Poisson's equation on a mesh and image processing via filtering techniques. The computation of these transforms should be a problem well suited to Highly Parallel Computers (HPCs), however to obtain the greatest efficiency from using this type of computer it is important that the calculation should be organised in such a way as to utilise the parallelism implicit in the problem. To simplify the writing of the code, use is made of readily available Fast Fourier Transform (FFT) routines. It is assumed that for a HPC with N_p^2 processing elements, there will be a routine available to compute the 2-d FT of a set of N_p^2 complex data points (for an example of a 2-d FFT routine on the Distributed Array Processor (DAP) see Davies 1984). This appendix is divided into two sections. In the first section 2-dimensional transforms are discussed, and in particular a method for calculating several NxN transforms simultaneously using one call to an N_pxN_p 2-d FT routine (applicable when $N_p/N=1,2,4,8,...$). This is achieved by 'interleaving' the input matrices in such a way that their transforms can be easily unscrambled. An efficient algorithm for doing the interleaving is also given. In the second section 3-dimensional FTs will be discussed, and in particular a method of using the results from the first section in combination with base "m + n" FTs so as to maximise the parallelism in the problem to obtain 3-d FFT routines that run as quickly as possible. A method for using an algorithm to write a general 3-d FFT program (as opposed to a program for specific values of N and N_p will also be discussed.

The algorithm developed below was specifically designed for use in a program to solve Poisson's equation on a mesh in a molecular dynamics simulation. As this routine has to be called twice every timestep and as a molecular dynamics program can run for several tens of thousands of timesteps the major concern was to develop a program that ran as quickly as possible rather than one which used the minimum amount of memory. By putting the extra

memory used by the 3-d FT routine into a common block, the workspace is made available for use in other parts of the molecular dynamics program. If the amount of memory used is important, then a different FFT should be used for doing the 1-d N-point transform in the second section. A wide range of such transforms is available (Swarztrauber 1984).

II.II. Two-dimensional FFTs

This section covers the problem of efficiently computing a 2-d FT on an NxN set of complex data points using a HPC with N_p^2 PEs. Obviously if $N = N_p$ there is no problem, simply use the library routine, but what happens if $N = N_p$.One could attempt to write the complete 2-d FT routine from scratch, but writing FFT routines can be a long, intricate and laborious job. In addition a new routine would have to be written for each value of N_p/N. There is however a fairly straightforward way of using the library 2-d FFT routine to do this task, provided that the ratio N_p/N is a power of 2. The technique presented here for doing this can be generalised to cope with ratios that are powers of numbers other than 2, but as for all HPCs currently available N_p is a power of two, N_p/N=1,2,4,8,.. should cover most cases encountered in practice.

To use the library routine efficiently we must find a way of doing several NxN FTs with one call of the library $N_p x N_p$ FT routine; for example if $N_p/N = 2$ we would like to be able to compute four NxN FT's simultaneously. It can be shown that once we have solved the problem for $N_p/N = 2$ then we have effectively solved the problem for any value of this ratio that is a power of 2.

Figure 1a represents the initial situation. On one data plane of the HPC we have placed four of the NxN matrices to be transformed. In the notation used capital letters refer to transformed data and lower case letters refer to the original data, so **a** would refer to an NxN matrix and A(i,j) would refer to the element in the ith row and jth column of the FT of **a**. We would like a routine that will take us from the situation in figure 1a, to the situation in figure 1b with only one call to the library FT routine. The problem is to find a way of organising the data from the four NxN matrices such that on calling the library FT routine the matrix shown in figure 1b can be readily recovered.

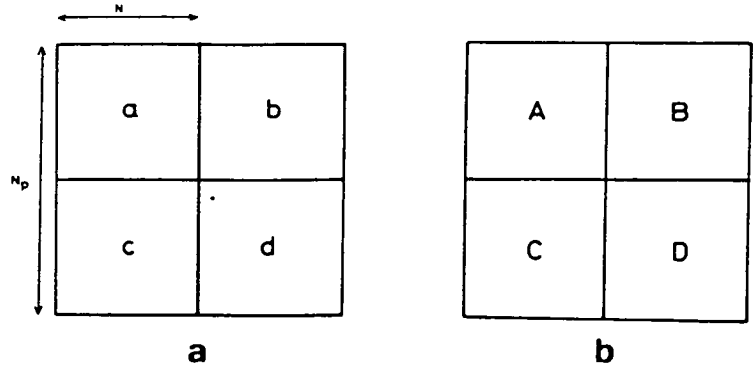


Fig. 1 (a) How the four $N \times N$ matrices are stored on an $N_p \times N_p$ word-plane, $N_p = 2N$; (b) The storage of the FTs of the four $N \times N$ matrices in (a).

This can be achieved by 'interleaving' the four matrices in such a way that the desired transforms appear modified only by known phase factors. Start by restructuring the data as shown in figure 2. This restructuring is a non-trivial computational problem and will be discussed later.

Let x be the N_pxN_p matrix representing the numbers on the data plane after the interleaving of the four NxN matrices has occurred. The 2-d N_pxN_p FT of this matrix is defined as:-

$$X(k_x, k_y) = \sum_{n_x, n_y=0}^{N_p-1}\sum x(n_x, n_y) \exp\left[-2\pi i(n_x k_x + n_y k_y)/N_p\right]. \qquad (2.1)$$

Equation (2.1) can now be split into four terms; n_x and n_y both even, n_x even and n_y odd, n_x odd and n_y even, n_x and n_y both odd.

$$\begin{aligned} X(k_x, k_y) = \sum_{n_x, n_y=0}^{N-1}\sum \{ & x(2n_x, 2n_y) \exp\left(-2\pi i\left[n_x k_x + n_y k_y\right]/N\right) \\ & + x(2n_x, 2n_y+1) \exp\left(-2\pi i\left[n_x k_x + n_y k_y\right]/N\right)\phi_y(k_y) \\ & + x(2n_x+1, 2n_y) \exp\left(-2\pi i\left[n_x k_x + n_y k_y\right]/N\right)\phi_x(k_x) \\ & + x(2n_x+1, 2n_y+1) \exp\left(-2\pi i\left[n_x k_x + n_y k_y\right]/N\right)\phi_x(k_x)\phi_y(k_y)\} \end{aligned} \qquad (2.2)$$

where

$$\phi_x(k_x) = \exp(-2\pi i k_x/N_p), \qquad \phi_y(k_y) = \exp(-2\pi i k_y/N_p). \qquad (2.3)$$

Note that an expression such as the first term of (2.2) is simply :

$$A(k_x, k_y) + B(k_x, k_y) + C(k_x, k_y) + D(k_x, k_y) \qquad (2.4)$$

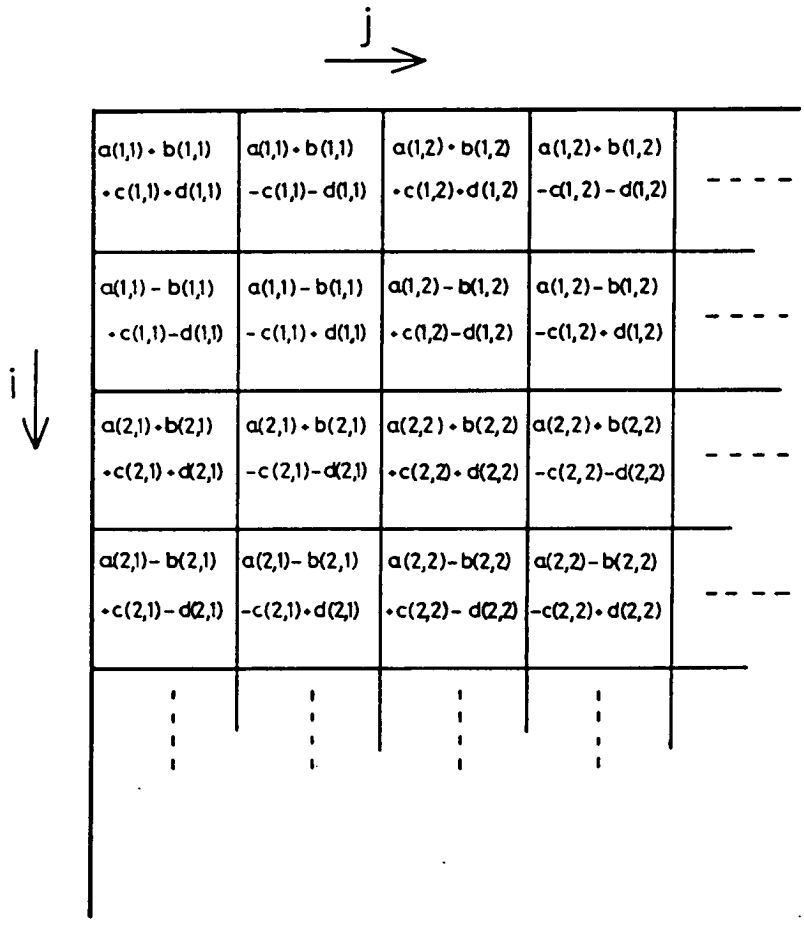


Fig. 2. The form of the restructured data.

by the way that the interleaved matrix is set up, so by defining

$$\begin{aligned} A' &= A + B + C + D, & C' &= A - B + C - D, \\ B' &= A + B - C - D, & D' &= A - B - C + D, \end{aligned} \tag{2.5}$$

it is possible to rewrite equation (2.2) in the form:-

$$\begin{aligned} X(k_x, k_y) = A'(k_x, k_y) &+ B'(k_x, k_y)\phi_y(k_y) + C'(k_x, k_y)\phi_x(k_x) \\ &+ D'(k_x, k_y)\phi_x(k_x)\phi_y(k_y). \end{aligned} \tag{2.6}$$

Now split the matrix X into four sections

$$\begin{aligned} X_{00} &= \{X(k_x, k_x): 0 \leqslant k_x, k_y \leqslant N-1\}, \\ X_{10} &= \{X(k_x + N, k_y): 0 \leqslant k_x, k_y \leqslant N-1\}, \\ X_{01} &= \{X(k_x, k_y + N): 0 \leqslant k_x, k_y \leqslant N-1\}, \\ X_{11} &= \{X(k_x + N, k_y + N): 0 \leqslant k_x, k_y \leqslant N-1\}. \end{aligned} \tag{2.7}$$

Note that as N_p = 2N then using equation (2.3)

$$\phi_x(k_x + N) = -\phi_x(k_x), \quad \phi_y(k_y + N) = -\phi_y(k_y) \tag{2.8}$$

so that in terms of X_{00}, X_{10},X_{01} and X_{11} equation (2.6) can be rewritten as

$$\begin{aligned} X_{00} &= A' + B'\phi_y + C'\phi_x + D'\phi_x\phi_y, & X_{01} &= A' - B'\phi_y + C'\phi_x - D'\phi_x\phi_y, \\ X_{10} &= A' + B'\phi_y - C'\phi_x - D'\phi_x\phi_y, & X_{11} &= A' - B'\phi_y - C'\phi_x + D'\phi_x\phi_y \end{aligned} \tag{2.9}$$

where

$$\phi_x = \{\phi_x(k_x): 0 \leqslant k_x \leqslant N-1\}, \qquad \phi_y = \{\phi_y(k_y): 0 \leqslant k_y \leqslant N-1\}. \tag{2.10}$$

Equations (2.9) now give us four simultaneous equations for A', B', C' and D' which can be solved to give:-

$$\begin{aligned} 4A' &= X_{00} + X_{10} + X_{01} + X_{11}, & 4C' &= (X_{00} - X_{10} + X_{01} - X_{11})\phi_x^*, \\ 4B' &= (X_{00} + X_{10} - X_{01} - X_{11})\phi_y^*, & 4D' &= (X_{00} - X_{10} - X_{01} + X_{11})\phi_x^*\phi_y^* \end{aligned} \tag{2.11}$$

where * represents complex conjugation. Equations (2.11) can then be substituted into equations (2.5) to give

$$\begin{aligned} 4A &= (A' + B' + C' + D'), & 4C &= (A' - B' + C' - D'), \\ 4B &= (A' + B' - C' - D'), & 4D &= (A' - B' - C' + D'). \end{aligned} \tag{2.12}$$

The calculation of the inverse transform is done using the result

$$x(n_x, n_y) = \left\{ \sum_{n_x, n_y=0}^{N-1} \sum X^*(k_x, k_y) \exp\left[-2\pi i (k_x n_x + k_y n_y)/N \right] / N^2 \right\}^* . \qquad (2.13)$$

The inverse transform is performed by taking the complex conjugate of the input matrices, taking the normal 2-d FT and finally taking the complex conjugate once more and normalising by $1/N^2$.

The first problem encountered in coding this algorithm is how to change a matrix in the form of figure 1(a) into one in the form of figure 2 efficiently. If we can construct a matrix in the form of figure 3 then the problem is solved; once we have a matrix in this form then a matrix of the form in figure 2 can be be constructed by using the shift facilities.

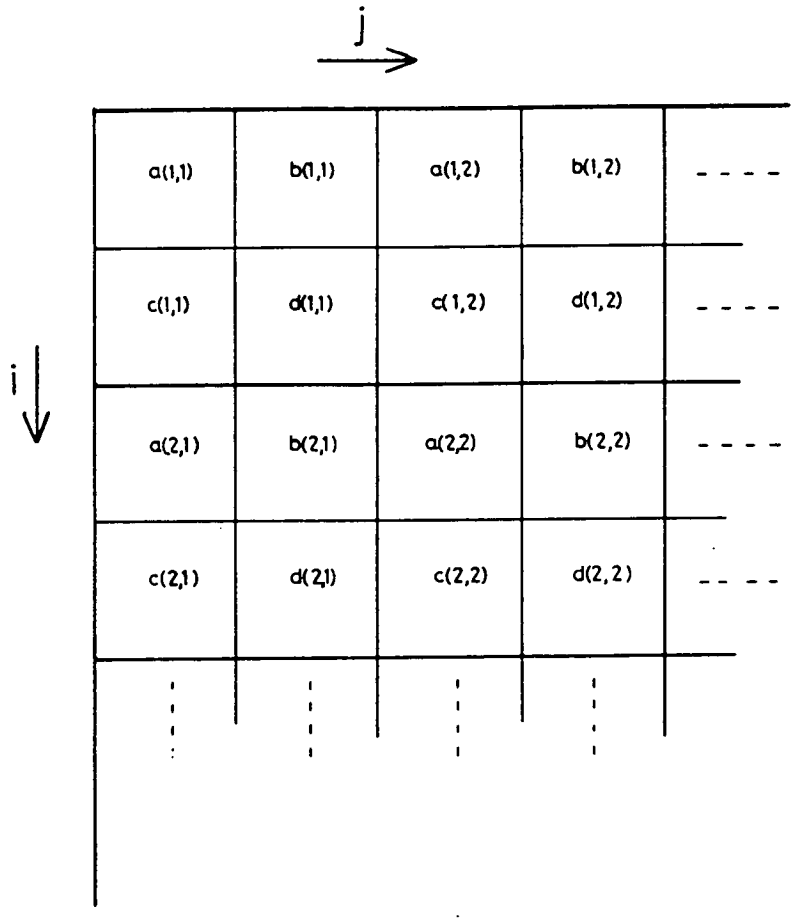


Fig.3 The first stage in restructuring the data.

Take the case N_p = 16. Consider first the problem of turning a matrix of the form in fig. 4a into a matrix of the form in fig. 4d.

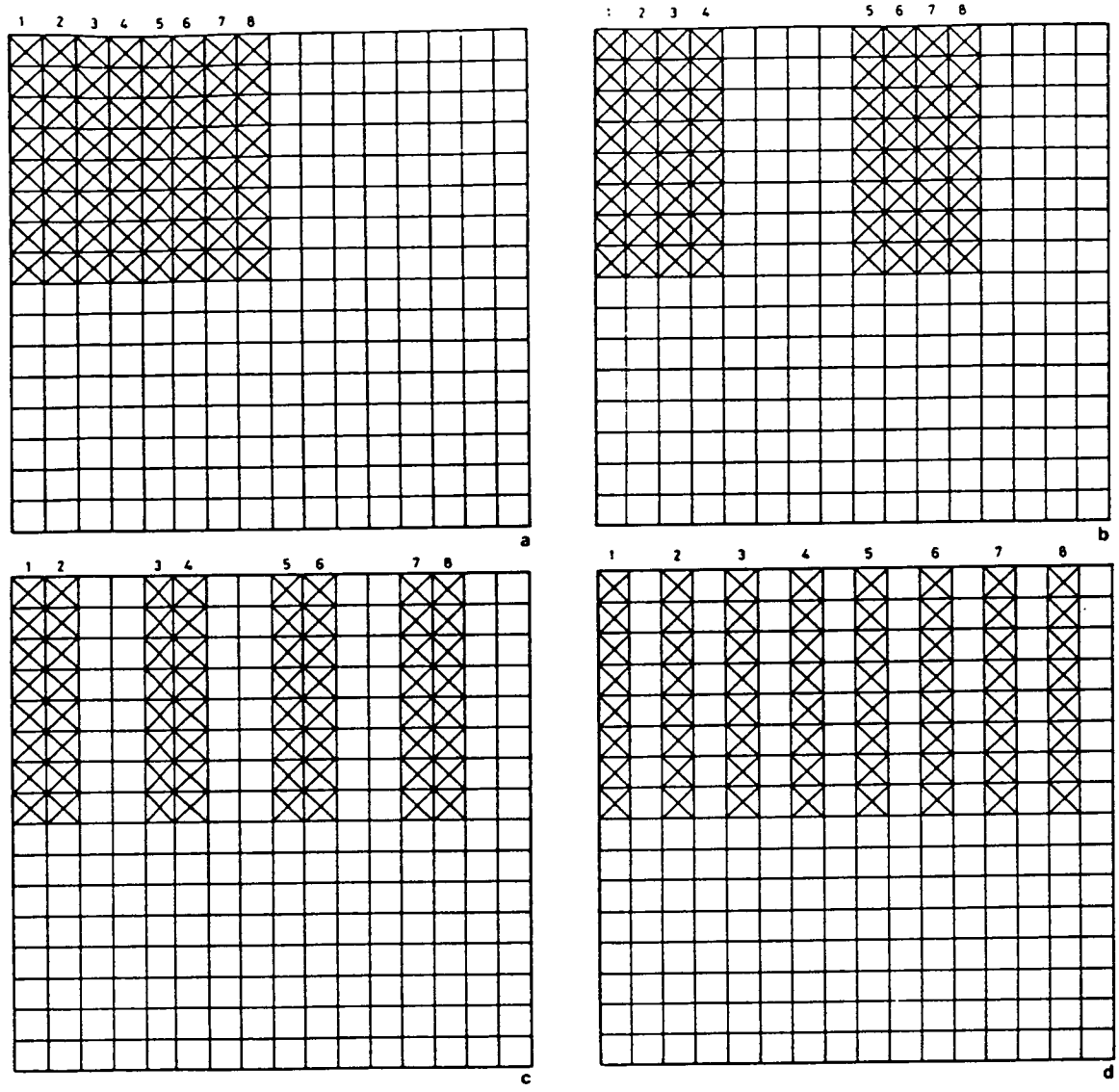


Fig. 4. The stages involved in spacing out the columns of an $N \times N$ matrix on an $N_p \times N_p$ word-plane; $N = 8$, $N_p = 16$.

We could get the latter by shifting each column independently, making a total of 7 operations. However using a parallel computer one would like to be able to reduce this to 3 (= $\log_2 8$) operations using some form of binary algorithm. On inspection of the interleaved matrix it can be seen that the first four columns of the original matrix are transferred onto the first 8 columns of the interleaved matrix and that the next 4 columns are transferred onto the last 8 columns of the interleaved matrix. The first step in the algorithm is to shift the east half of a 4 columns to the east (fig. 4b). The next step is to shift the east halves of the new columns to the east by 2. After repeating this procedure once more, reducing the shift to one column, the required matrix is obtained (figs 4c and 4d). This algorithm can be used for any value of N_p that

is a power of 2.

It is now not too difficult to adapt this algorithm for the job required. Start by interleaving the matrix from figure 1a in the E-W direction, **a** and **c** to the east, **b** and **d** to the west. Then repeat the procedure in the N-S direction. The code for doing this on the DAP where N = 32, N_p = 64 is shown below.

```
1.          REAL X(,),X1(,)
2.          X1 = X
3.          N = 32
4.    C     this loop does the interleaving in the E-W direction
5.          DO 1 I = 1,5
6.             N2 = N
7.             N = N/2
8.             X = MERGE(SHEP(X,N),X,ALTC(N2))
9.             X1 = MERGE(X1,SHWP(X1,N),ALTC(N2))
10.   1     CONTINUE
11.         X = MERGE(X1,X,ALTC(1))
```

Similarly for the N-S direction

The very fast instruction ALTC(N) used in lines 8,9 and 11 produces a logical mask of which the first N columns are .FALSE., the next N columns are .TRUE., the next N columns are .FALSE. etc. The command ALTR(N) has the analogous effect on rows. The interleaving is done in lines 5-10. Line 8 spaces out the matrices **a** and **c** to the east and line 10 spaces out matrices **b** and **d** to the west. The two matrices constructed, X(,) and X1(,), are then interleaved together using the MERGE on line 11. The division by 2 on line 7 could be done more efficiently by equivalencing N to a logical vector and replacing the division by a shift. This algorithm can be written much more compactly in the form

```
          N = 32
          DO 1 I = 1,5
             NH = N/2
C            E-W interleaving
             X(SHWC(ALTC(N),NH)) = MERGE(SHWP(X,NH),SHEP(X,NH),ALTC(NH))
C            N-S interleaving
             X(SHNC(ALTR(N),NH)) = MERGE(SHNP(X,NH),SHSP(X,NH),ALTR(NH))
             N=NH
1         CONTINUE
```

For N ≠ 32 the program is easily changed. Replace the 32s by N and the 5s by $\log_2 N$. Once the matrix has been got into the form of fig. 3 then the interleaving procedure can be completed by using shifts and logical masks. For instance if X(,) is a matrix in the form of fig. 3 then the code

```
X = X+MERGE(-SHEP(X,1),SHWP(X,1),ALTC(1))
X(ALTR(1).LNEQ.ALTC(1)) =
*       -MERGE(SHSP(SHWP(X,1),1),SHNP(SHEP(X,1),1),ALTR(1))
X = X+MERGE(SHEP(X,1),-SHWP(X,1),ALTC(1))
```

will produce a matrix in the form of fig. 2.

On the DAP the 2-d 64x64 REAL*4 FFT routine runs in 30ms. When the above algorithm was coded with N_p = 64 and N = 32 the program computed four 32x32 complex FTs in 52ms.

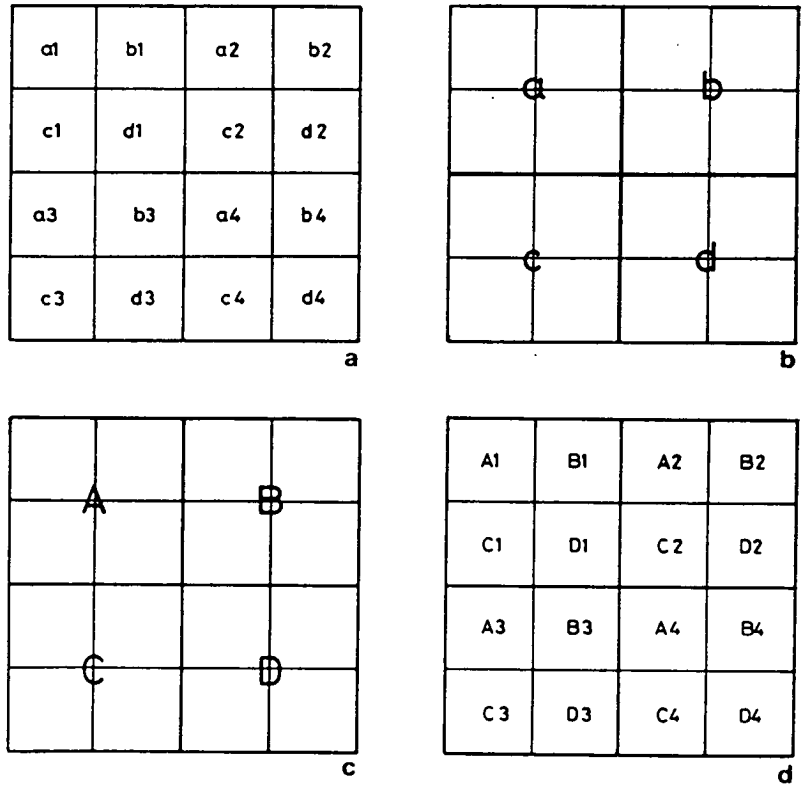


Fig. 5. (a) The storage of the 16 $N \times N$ matrices on the word-plane; (b) The situation after one call to the interleaving routine. The $2N \times 2N$ matrix a is made up of the four $N \times N$ matrices $a1$, $b1$, $c1$, $d1$ in an analogous way to the matrix in Fig. 4, similarly for the matrices b c, d; (c) The situation after one call to the $N_p/2 \times N_p/2$ FT routine and the unscrambling routine; (d) The situation after one more call to the unscrambling routine.

We have now covered the cases where N_p/N = 1 (trivial) and N_p/N = 2. What

happens when $N_p/N = 4$ (fig. 5a)? The problem can be split into two parts. First use the interleaving routine with N_p replaced by $N_p/2$. This will produce the situation as in fig. 5b. Now use the interleave routine a second time with N_p as the parameter and then call the library FT routine. Using the unscrambling algorithm described above we get to the position shown in fig. 5c, i.e. we have the $N_p/2 \times N_p/2$ Fourier transforms of the four quarters of the matrix in fig. 5b. By repeating the unscrambling, this time with N_p replaced by $N_p/2$, we obtain the 16 transformed matrices in their correct positions (fig. 5d). The case $N_p/N=8$ can be similarly computed, this time using 3 calls to both the interleaving and unscrambling routines.

II.III. Three-dimensional FFTs

Consider a 3-d FT on a set of N^3 complex data points. The transform is defined by:-

$$X(k_x, k_y, k_z) = \sum_{n_x, n_y, n_z=0}^{N-1} x(n_x, n_y, n_z) \exp[-2\pi i k_x n_x/N] \times \exp[-2\pi i k_y n_y/N] \exp[-2\pi i k_z n_z/N], \quad (2.14)$$

i.e. the 3-d FT can be regarded as a set of three nested one dimensional transforms. Define (2.14) as

$$X = \{ X(k_x, k_y, k_z): 0 \leq k_x, k_y, k_z \leq N-1 \} \quad (2.15)$$

and

$$X(k_x, k_y, n_z) = \left\{ \sum_{n_x, n_y=0}^{N-1} x(n_x, n_y, n_z) \exp[-2\pi i n_x k_x/N] \times \exp[-2\pi i n_y k_y/N]: 0 \leq k_x, k_y \leq N-1 \right\}. \quad (2.16)$$

Then using this notation the 3-d FT (2.14) can be written as:-

$$X = \sum_{n_z=0}^{N-1} X(k_x, k_y, n_z) \exp[-2\pi i n_z k_z/N]. \quad (2.17)$$

The term $X(k_x, k_y, n_z)$ can be regarded as the 2-d FT of the data set

$$\{ X(n_x, n_y, n_z): 0 \leq n_x, n_y \leq N-1 \}. \quad (2.18)$$

Thus the 3-d transform of a set of N^3 complex data points can be thought of as the N-point 1-d transform of a set of N NxN data sets. The 2-d transforms

can be computed efficiently using the results of the previous section and the 1-d N-point transform for the third dimension can be calculated using what is called a base "r_1 + r_2" FFT as follows.

In calculating the 2-d transform we did not have to code any FFT routines explicitly as they were available in the library FFT routine. However, to calculate the 1-d FT efficiently we must code such a two base algorithm. Consider an N-point FT where $N = r_1 r_2$:

$$X(k) = \sum_{n=0}^{N-1} x(n) \exp[-2\pi i k n/N]. \qquad (2.19)$$

Rewrite k and n in equation (2.20) as :-

$$\begin{aligned} k &= k_1 r_1 + k_0, \quad 0 \leqslant k_0 \leqslant r_1 - 1, \quad 0 \leqslant k_1 \leqslant r_2 - 1, \\ n &= n_1 r_2 + n_0, \quad 0 \leqslant n_0 \leqslant r_2 - 2, \quad 0 \leqslant n_1 \leqslant r_1 - 1. \end{aligned} \qquad (2.20)$$

Then on substituting for n, equation (2.20) can be rewritten as

$$X(k) = \sum_{n_0=0}^{r_2-1} \left[\sum_{n_1=0}^{r_1-1} x(n_1, n_0) W^{k n_1 r_2} \right] W^{k n_0} \qquad (2.21)$$

where

$$W = \exp(-2\pi i/N). \qquad (2.22)$$

But W^N = 1 therefore

$$W^{k n_1 r_2} = W^{(k_1 r_1 + k_0) n_1 r_2} = W^{k_0 n_1 r_2}. \qquad (2.23)$$

So substituting for k in equation (2.21)

$$X(k_1, k_0) = \sum_{n_0=0}^{r_2-1} \left(\sum_{n_1=0}^{r_1-1} x(n_1, n_0) W^{k_0 n_1 r_2} \right) W^{(k_1 r_1 + k_0) n_0}. \qquad (2.24)$$

Regrouping the terms in W (known as 'twiddle' factoring), equation (2.24) can be written in the form

$$X(k_1, k_0) = \sum_{n_0=0}^{r_2-1} \left(\sum_{n_1=0}^{r_1-1} x(n_1, n_0) W^{k_0 (n_1 r_2 + n_0)} \right) W^{k_1 n_0 r_1}, \qquad (2.25)$$

or in recursive form as

$$\begin{aligned} x_1(k_0, n_0) &= \left[\sum_{n_1=0}^{r_1-1} x(n_1, n_0) W^{k_0 n_1 r_2} \right] W^{k_0 n_0}, \\ x_2(k_0, k_1) &= \sum_{n_0=0}^{r_2-1} x_1(k_0, n_0) W^{k_1 n_0 r_1}, \\ X(k_1, k_0) &= x_2(k_0, k_1). \end{aligned} \qquad (2.26)$$

i.e. the final results come out in 'base' reversed form.

Consider the specific example of calculating a 3-d FT on an N^3 set of complex data points with N_p = 64 and N = 32. Consider how the data is stored in the computer. The data is stored over two arrays, $X(,,N_Z)$ for the real data parts and $Y(,,N_Z)$ for the imaginary parts, where $N_Z=N^3/N_p^2$ which equals 8 in the example. For the sake of compactness the expression z(a,b) is used to represent the 32x32 matrix x(a,b) + iy(a,b). Capital letters refer to arrays on the computer, for instance X(,,2) is the second data plane of the real data and, bold type letters refer to 32x32 matrices, for example **x**(a,b) is the $(r_2a+b)^{th}$ 32x32 matrix. Similarly Z(,,2) represents X(,,2) + iY(,,2) (fig. 6).

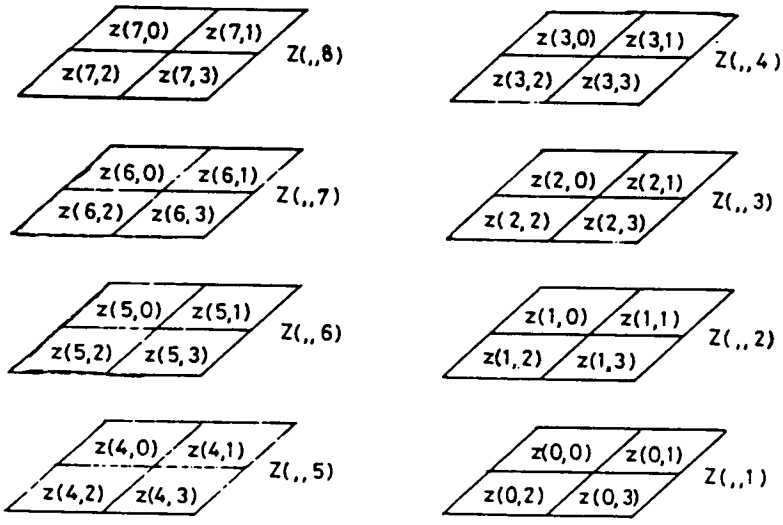


Fig. 6 The storage of the 32 32×32 matrices on the complex array $Z(,,8)$. $z(a, b)$ refers to the 32×32 matrix $z(n_x, n_y, (r_2a+b))$.

We can calculate the terms

$$z(k_x, k_y, n_z), \quad n_z = 0,\ldots,31 \qquad (2.27)$$

by using the 2-d FT algorithm developed in the first section, simply by calling

the routine 8 times, once for each data plane. We now have to perform a 32-point transform on the 32 32x32 complex data points produced by using the 2-d routine. Substituting $r_1=N_Z=8$, $r_2=4$, N=32 into (2.26)

$$z(k_1, k_0) = \left(\sum_{n_1=0}^{r_1-1} z(n_1, n_0) W^{4k_0n_1} \right) W^{k_0n_0}. \qquad (2.28)$$

Define

$$z_1'(k_1, n_0) = \sum_{n_1=0}^{r_1-1} z(n_1, n_0) W^{4k_0n_1}. \qquad (2.29)$$

Calculate all the z_1' terms first and then multiply each by $W^{n_0k_0}$. The calculation of the $z_1'(k_1,n_0)$ terms fits nicely onto the HPC (this being the original reason for choosing this FFT algorithm). Look at the calculation of $z_1'(k_0,i)$ $0 <= i <= 3$

(note that the four 32x32 matrices $z_1'(k_0,i)$ all lie on the k_0 data plane)

$$z_1'(k_0, i) = z(0, i) + W^{4k_0}z(1, i) + W^{8k_0}z(2, i) + W^{12k_0}z(3, i) + W^{16k_0}z(4, i) + W^{20k_0}z(5, i) + W^{24k_0}z(6, i) + W^{28k_0}z(7, i). \qquad (2.30)$$

All the matrices $z_1'(k_0,i)$ can be calculated simultaneously for $0 <= k_0 <= 7$.

The next stage in the calculation is to multiply each of the $z_1'(k_0,n_0)$ by $W^{n_0k_0}$. This again can be done efficiently in parallel. Set up a complex matrix Q:-

$$Q(.) = \begin{cases} 1, & 1 \leqslant i, j \leqslant 32, \\ W, & 1 \leqslant i \leqslant 32, 33 \leqslant j \leqslant 64, \\ W^2, & 33 \leqslant i \leqslant 64, 1 \leqslant j \leqslant 32, \\ W^3, & 33 \leqslant i, j \leqslant 64. \end{cases} \qquad (2.31)$$

Then the multiplication of each $z_1'(k_0,n_0)$ by $W^{n_0k_0}$ is equivalent to multiplying $Z(,,k_0)$ by Q^{k_0}. Having calculated the z_1 matrices, continue with the the base 4 part of the algorithm.

$$z_2(k_0, k_1) = \sum_{n_0=0}^{r_2-1} z_1(k_0, n_0) W^{8k_1n_0}$$

$$= z_1(k_0, 0) + W^{8k_1}z_1(k_0, 1) + W^{16k_1}z_1(k_0, 2) + W^{24k_1}z_1(k_0, 3).$$

Therefore (2.32)

$$z_2(k_0, 0) = z_1(k_0, 0) + z_1(k_0, 1) + z_1(k_0, 2) + z_1(k_0, 3),$$
$$z_2(k_0, 1) = z_1(k_0, 0) + W^8 z_1(k_0, 1) + W^{16} z_1(k_0, 2) + W^{24} z_1(k_0, 3),$$
$$z_2(k_0, 2) = z_1(k_0, 0) + W^{16} z_1(k_0, 1) + z_1(k_0, 2) + W^{16} z_1(k_0, 3),$$
$$z_1 2(k_0, 3) = z_1(k_0, 0) + W^{24} z_1(k_0, 1) + W^{16} z_1(k_0, 2) + W^8 z_1(k_0, 3). \quad (2.33)$$

but as the coefficients have a simple form,

$$W^8 = \exp(-\pi i/2) = -i, \qquad W^{16} = \exp(-\pi i) = -1, \qquad W^{24} = \exp(-3\pi i/2) = i. \quad (2.34)$$

the equations (2.33) can be rewritten as

$$z_2(k_0, 0) = z_1(k_0, 0) + z_1(k_0, 1) + z_1(k_0, 2) + z_1(k_0, 3),$$
$$z_2(k_0, 1) = z_1(k_0, 0) - iz_1(k_0, 1) - z_1(k_0, 2) + iz_1(k_0, 3),$$
$$z_2(k_0, 2) = z_1(k_0, 0) - z_1(k_0, 1) + z_1(k_0, 2) - z_1(k_0, 3),$$
$$z_2(k_0, 3) = z_1(k_0, 0) + iz_1(k_0, 1) - z_1(k_0, 2) - iz_1(k_0, 3). \quad (2.35)$$

Table 1

k_z coordinates of z matrices	k_1	k_0	k_0	k_1	k_z', base reversed k_z coordinates of z_2 matrices
0	0	0	0	0	0
1	0	1	1	0	4
2	0	2	2	0	8
3	0	3	3	0	12
4	0	4	4	0	16
5	0	5	5	0	20
6	0	6	6	0	24
7	0	7	7	0	28
8	1	0	0	1	1
⋮	⋮	⋮	⋮	⋮	⋮
27	3	3	3	3	15
28	3	4	4	3	19
29	3	5	5	3	23
30	3	6	6	3	27
31	3	7	7	3	31

$k_z = 8k_1 + k_0$, $k_z' = 4k_0 + k_1$.

Note that the calculation of the base 4 transform requires no multiplication operations, this being why the 'twiddle' factoring was done. The calculation of the base 4 transform can now be done data-plane by data-plane using shifts and the operations + and −. The final stage in the calculation of the transform is 'base' reversal (2.27). If figure 7a represents the ordering of the z_2 matrices on the data planes after the base 4 and base 8 transforms have been made, figure 7b represents the order in which we must place them to achieve the 'base' reversal (it is not the matrices that are base reversed but their z

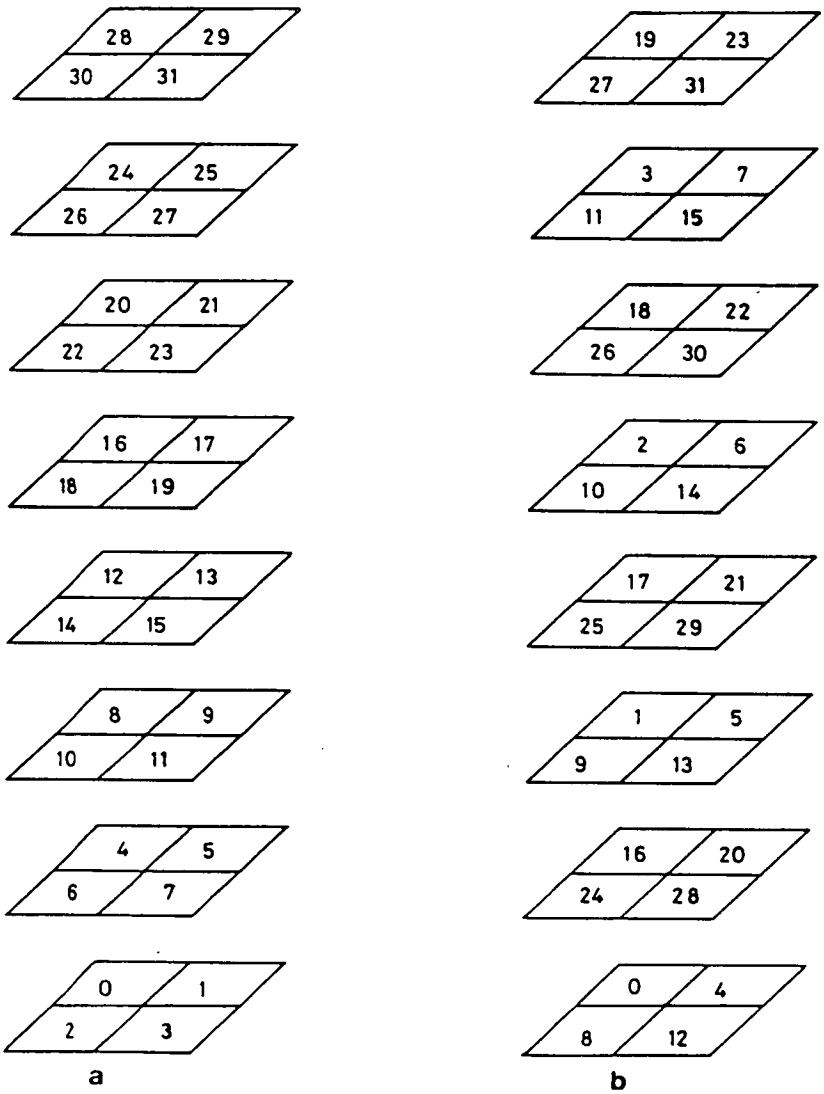


Fig. 7. (a) The 32 32×32 matrices $z_2(k_x, k_y, k_z)$, the numbers refer to the k_z coordinate; (b) The reordering of these matrices after 'base' reversal of the k_z label, see Table 1.

coordinate). Table 1 shows how the positions in 7b were calculated. This process cannot be coded easily in parallel, the quickest way that it can be done is simply to shift every 32x32 matrix into its required position individually. While the code to do this is tedious to write it does run quickly, the time taken to achieve the data shifting is insignificant compared with the time taken to perform the 2-d FTs.

When the above algorithm was coded for the DAP, the 3-d FT of a set of 32^3 data points was calculated in 476ms, of which 416ms was used in the calculation of the 2-d transforms. To check that the program was working correctly a random complex matrix was compared with the inverse FT of the FT of the same matrix.

This algorithm can be generalised to calculate 3-d FFTs on a set of N^3 data points using an HPC with N_p^2 PEs where N and N_p can take any values such that N_p/N is a power of two. If the data is stored on a series of r_1 complex data planes each containing r_2 NxN matrices then the procedure to calculate this transform is given below.

The 2-d transform can be done quite simply, requiring r_3 calls to both the interleaving and unscrambling routines where $r_3=\sqrt{2}$

The equation for the base "r_1" transform is given in equation (2.27)

$$x_2(k_0, k_1) = \sum_{n_0=0}^{r_2-1} x_1(k_0, n_0) W^{k_1 n_0 r_1}. \qquad (2.36)$$

Set up a complex vector of length r_1, WFAC(NR1) where

$$\mathrm{WFAC}(k) = W^{r_1}. \qquad (2.37)$$

The code

```
      M = 0
      DO 2 K = 0,NR2-1
         DO 1 N = 0,NR1-1
            M = M + K
            IF (M.GE.NR1) M = M - NR1
            Z(,,K) = Z(,,K) + WFAC(M)*Z1(,,N)
1        CONTINUE
         Z(,,K) = Z(,,K)*Q
         Q = Q*Q
2     CONTINUE
```

calculates the z_2 matrices in the previous equation. This code would not work in DAP FORTRAN as the language does not allow array indexing from zero or complex variables; Z, Z1, W and Q are all complex arrays. The same will be true of all the examples of code given in this section. The multiplication by $W^{n_0 k_0}$ is done, as before, by setting up an $N_p x N_p$ matrix Q set equal to W^{n_0} at PEs where the NxN matrix label is n_0 and simply multiplying the k^{0th} data-plane by Q^{k_0}.

Calculate the base r_2 transform. In the example given this was straightforward as r_2 was small. If r_2 is greater than four then the excessive data shifting involved in the calculation of the transform makes the computation inefficient. By splitting the base "r_2" transform into two parts the computation of the transform can be made much more efficient as follows.

The base "r_2" transform is defined by

$$z_2(k_0, k_1) = \sum_{n_0=0}^{r_2-1} z_1(k_0, n_0) W^{k_1 n_0 r_1} \qquad (2.38)$$

but for all the transforms considered here r_2 can be factorised, $r_2=r_3^2$, r_3 an integer. Rewrite k_0 and n_0 as base r_3 numbers

$$\begin{aligned} k_1 &= r_3 K_1 + K_0, \quad 0 \leqslant K_1, K_0 \leqslant r_3, \\ n_0 &= r_3 N_1 + N_0, \quad 0 \leqslant N_1, N_0 \leqslant r_3. \end{aligned} \qquad (2.39)$$

Equation (2.36) can now be rewritten as:-

$$z_2(k_0, k_1) = \sum_{N_0=0}^{r_3-1} \sum_{N_1=0}^{r_3-1} z_1(k_0, N_1, N_0) W^{r_1(r_3K_1 + K_0)(r_3N_1 + N_0)} \qquad (2.40)$$

or recursively as

$$\begin{aligned} z_2^{(1)}(k_0, K_0, N_0) &= \left\{ \sum_{N_1=0}^{r_3-1} z_1(k_0, N_1, N_0) W^{r_1 r_3 N_0 K_1} \right\} W^{r_1 N_0 K_0}, \\ z_2^{(2)}(k_0, K_0, K_1) &= \sum_{N_0=0}^{r_3-1} z_2^{(1)}(k_0, K_0, N_0) W^{r_1 r_3 N_1 K_0}, \\ z_2(k_0, K_1, K_0) &= z_2^{(2)}(k_0, K_0, K_1) \end{aligned} \qquad (2.41)$$

using $r_3^2=r_2$.

Note how the arguments of the $z_2^{(1)}$, $z_2^{(2)}$ etc. relate to the way that the NxN matrices are stored on the data planes of the HPC. The first argument

determines the complex data plane on which the NxN matrix is stored and the second and third arguments determine the position of the matrix on the data plane (fig. 8). The reason for splitting up the base "r_2" part of the algorithm becomes obvious when we look at the calculation of equations (2.39). The $z_2^{(2)}$ matrices are stored in the data-planes Z(,,i) and the $z_2^{(1)}$ matrices in the data-planes Z1(,,i).

Consider the calculation of $z_2^{(2)}(k_0,K_0,i)$ for $0 <= i <= r_3$, the K_0^{th} row of NxN matrices on the k_0^{th} data plane. The K_0^{th} row of NxN $z_2^{(1)}$ matrices is obtained by multiplying the n^{th} row of NxN z_1 matrices by $W^{r_1 r_3 K_1 (n-1)}$ and then summing the r_3 NxN matrices along a column and storing the resultant row of matrices in the K_0^{th} row of the $z_2^{(2)}$ data-plane. The multiplication by $W^{r_3 K_0 N_2}$ follows in a similar way to the case shown above: set up an N_pxN_p matrix QFAC(,) which has the value $W^{r_3 K_0 N_2}$ at positions corresponding to the NxN matrix whose block coordinates are (K_0,N_0) for all N_0 and K_0. The multiplication by $W^{r_3 K_0 N_2}$ simply becomes a multiplication of each of the data-planes by QFAC(,). This transform operates on one data-plane at a time and so can be done inside a loop over data-planes.

Initialise an N_pxN_p matrix Q(,) such that the n^{th} set of rows of Q(,) are $W^{r_1 r_3 n}$, form a logical mask LMASK(,) = ALTR(N) and set QFAC(,) = $W^{r_1 N_2 K_0}$ Then the code to calculate the transform for data-plane M is:-

```
      DO 1 K0 = 0,NR3-1
           Z(LMASK,M) = SUMROW((Z1(,,M)*Q),N)
           LMASK = SHSC(LMASK,N)
1     CONTINUE
      Z(,,M) = Z(,,M)*QFAC
```

where SUMROW(A,N) is the function:-

```
      REAL MATRIX FUNCTION SUMROW(A,N)
      REAL A(,)
      RLOG2R3 = ALOG(NR3)/LOG(2.)
      DO 1 I = 1,LOG2R3
           N1 = NP/(2*I)
           A = A + SHSC(A,N1)
1     CONTINUE
      SUMROW = A
      RETURN
      END
```

and NR3 = r_3. The calculation of the $z_2^{(2)}$ matrices follows similarly, except that here it is the columns which are multiplied by $W^{r_1 r_2 K_0 N_0}$ and the NxN matrices are summed in the E-W direction using SHWC in place of SHSC in the above routine.

To obtain the final z_2 matrices we must base reverse the $z_2^{(2)}$ matrix labels. If A(,) is the data-plane storing the r_3^2 NxN matrices and LMASK(,) is a logical mask set to .TRUE. down the block diagonal then the following code will base reverse the matrix labels (equivalent to interchanging the matrix row and column block coordinates N_0 and K_0)

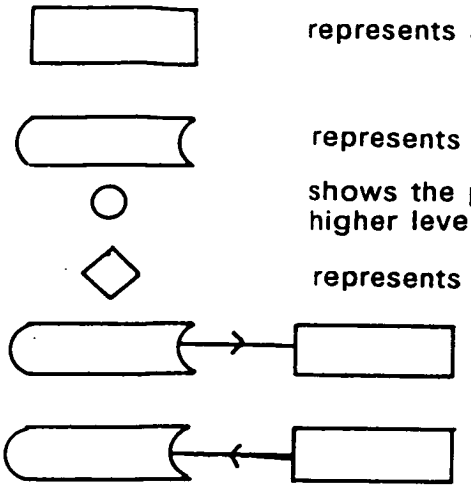
```
1          A(.NOT.LMASK) = TRAN(A)
2          DO 1 I=0,NR3-1
3               A = SHEC(A,N)
4               A(LMASK) = TRAN(A)
5    1     CONTINUE
6          A = SHEC(A,N)
```

Line 1 takes the transpose of A using a library function TRAN leaving the block diagonal unchanged. This leaves the other NxN matrices in the correct positions but individually transposed. In lines 2 to 5 these matrices are transposed back. When the matrix A is transposed NxN matrices on the block diagonal are transposed without their block coordinates being changed, thus each of the NxN matrices are shifted in turn to the block diagonal (line 3) and then transposed (line 4). This code will work for any value of N such that r_3 is an integer.

After the r_1 and r_2 point transforms have been done there still remains the problem of 'base' reversing the N NxN matrices ('base' reversal is defined by equation 2.27). The problem is not as simple as in the previous section as the matrices are spread over r_1 data-planes and r_1 need not be equal to r_2. There does not seem to be an elegant way of doing this base reversal as there was above. The simplest method is to shift each of the NxN matrices individually into the correct position. Although the code to do this is to write it should run quickly, data shift operations being significantly faster than arithmetic operations. The easiest method is to write a subroutine for each possibility and to call the appropriate subroutine at run time, as the number of possibilities is quite limited.

Using the code given above it is possible to write one program to calculate 3-d FFTs for any values of N and N_p such that the ratio N_p/N is a power of two. The calculation of the inverse transforms is done using the method described by equation (2.14).

III. Structure of the MD simulation program



represents a process

represents data in a common block

shows the point at which the diagram connects with a higher level process

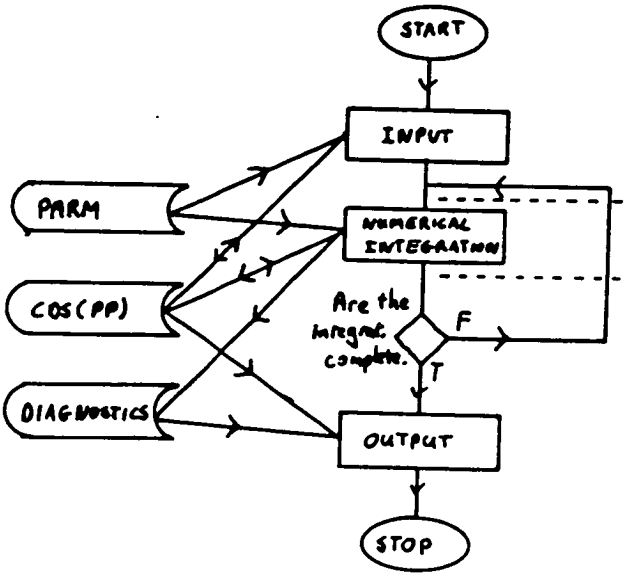
represents a decision process

data passing from a common block to a process

data passing from a process to a common block

III a)

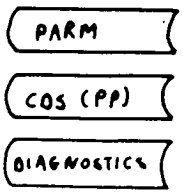
DATA FLOW PROGRAM PROCESSES COMMENTS



input an old configuration or initialise a new one

numerically integrate the equations of motion using the Beeman algorithm, expanded in figure III b)

output the new configuration and diagnostics from the run

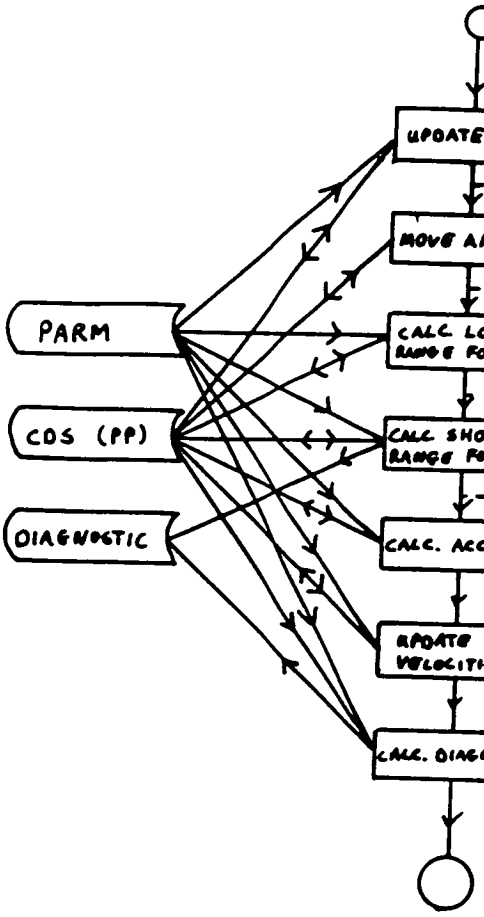


PARM contains the parameters for the program and the control data eg No of timesteps, timestep size etc.

CDS (PP) stores the particle coordinates, velocities and accelerations in the PP data representation

DIAG contains the diagnostic data from the runs such as temperatures and potential energy

III b)



update all the particle postions from time t to t+δt using the positions, velocities and accelerations of the particles from the previous timesteps

expanded in figure III c)

expanded in figure III d)

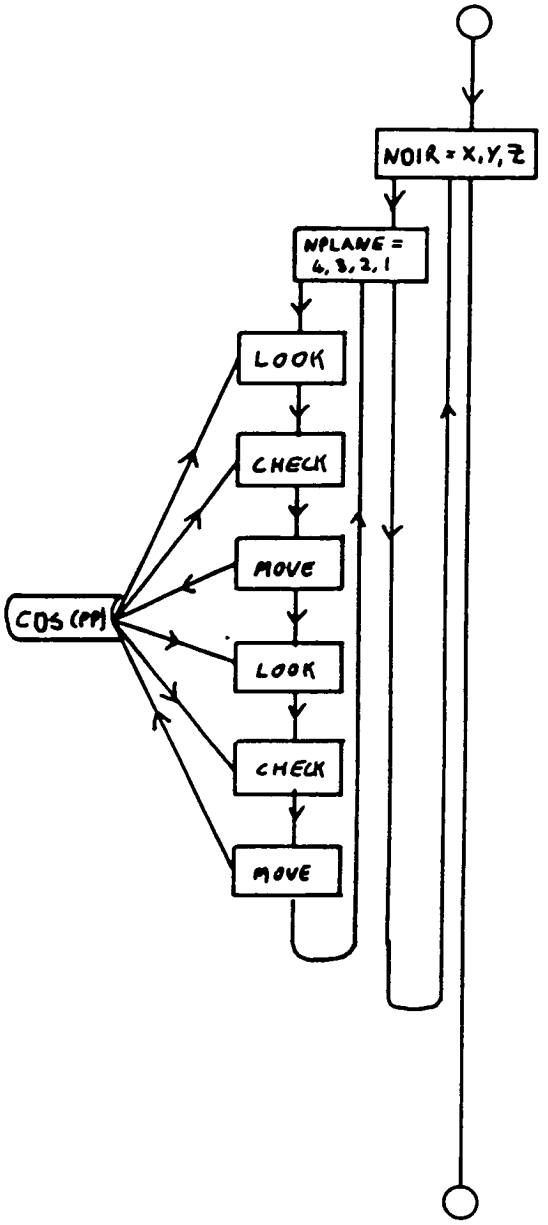
expanded in figure III e)

calculate the accelerations on every particle using a=F/m

calculate the particle velocities at time t+δt

calculate the temperature, potentials etc. of the system at time t+δt

III c)



loop over the x,y and z directions

loop over the four anion data-planes

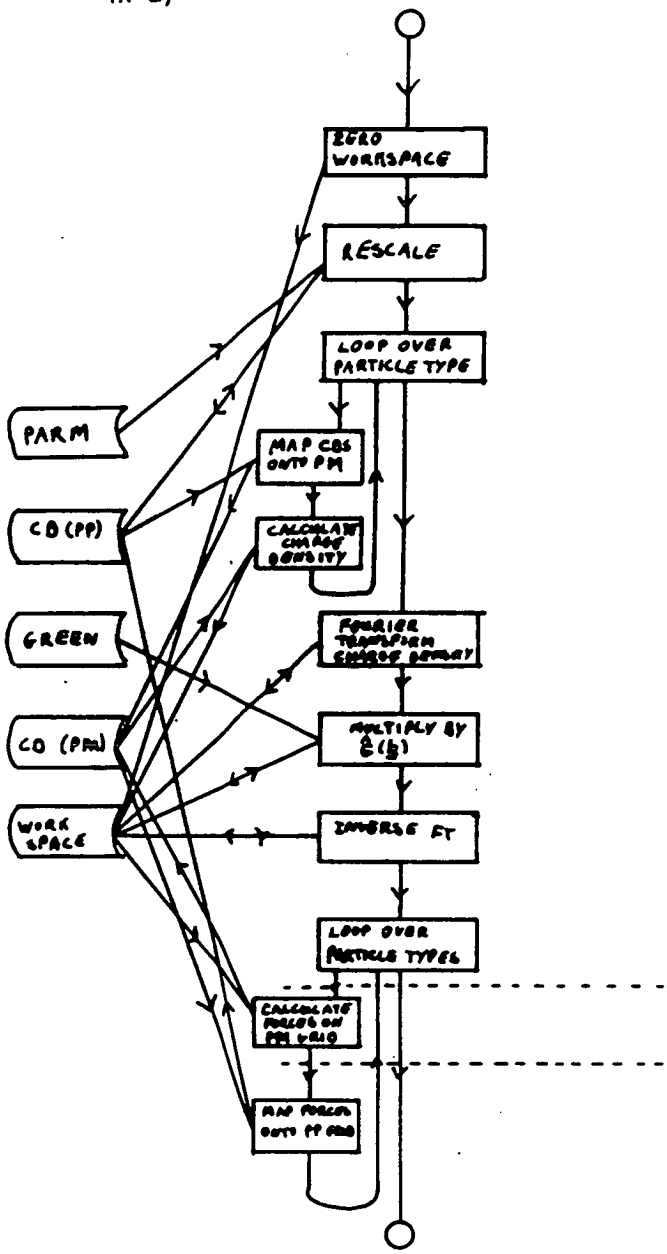
look for anions on plane NPLANE that have moved out of range of their present PEs in the +NDIR direction

look to see whether the fourth anion data-plane is empty at the positions where the anions should be moved to

move the anions out of range onto the positions where the fourth data-plane is empty, if there are any conflicts then leave those particular anions where they are. After the anions have been moved, shuffle them down to the lowest possible anion data plane.

Repeat the process in the -NDIR direction

III d)



Rescale the coordinates into units such that $h_1=h_2=h_3=1$

Inverse Fourier transform to obtain the electric potential on the mesh

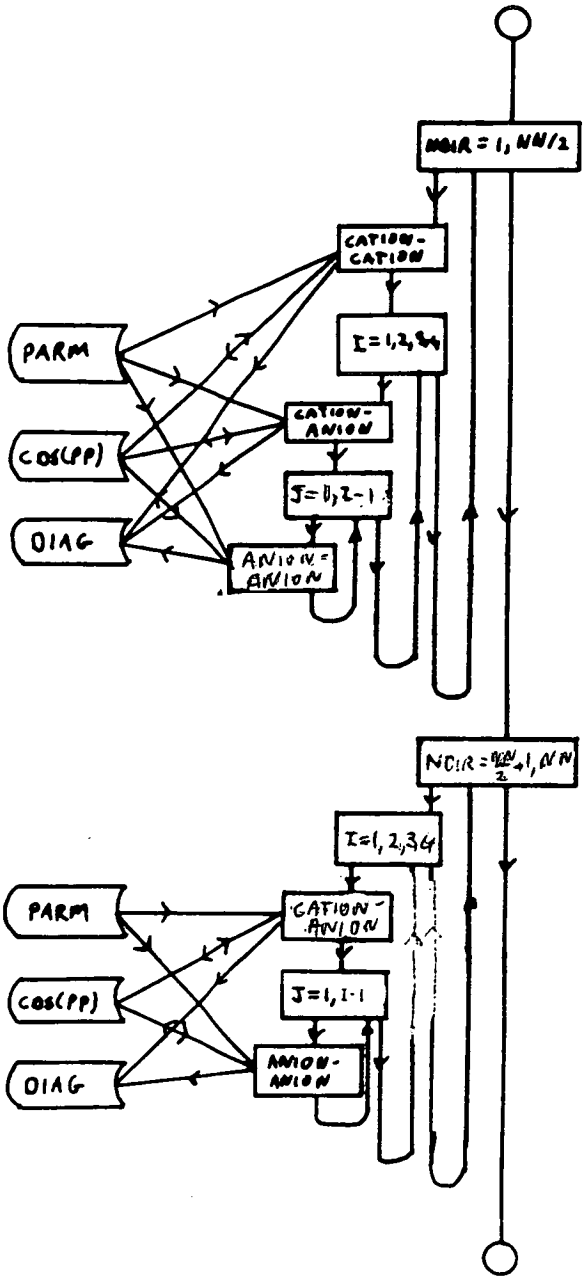
Expanded in figure III g)

WORK SPACE is a common block used to store quantities such as the mesh defined electric potential

CD (PM) stores particle coordinates on the PM mesh.

GREEN stores the precalculated values of $\hat{G}(\underline{k})$

III e)



Each PP cell interacts with NN other cells.
Loop over one half of these cells

calculate the forces that the cations exert on each other

loop over the anion data planes

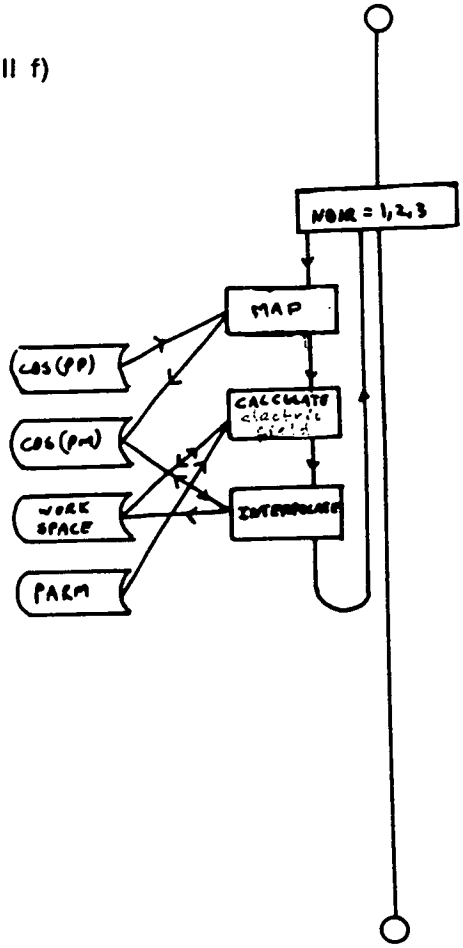
calculate the forces that the anions exert on the cations

loop over the anion data planes

calculate the forces the anions exert on the anions

Repeat the process for the other half of the PP cells (note that we do not have to calculate the cation-cation interactions these are given from applying Newton's third law to the cation-cation interactions calculated above

III f)



Map the NDIR component of the particles coordinates onto the PM grid

calculate the NDIR component of the electric field

interpolate the forces on the particles from the electric fields on the mesh points

Parallel Computing 3 (1986) 167–184
North-Holland

Two and three dimensional FFTs on highly parallel computers

A. BRASS and G.S. PAWLEY

Department of Physics, University of Edinburgh, Edinburgh EH9 3JZ, United Kingdom

Received September 1985
Communicated by Professor D.J. Evans

Abstract. An algorithm is described for computing two and three dimensional Fourier transforms on computers of SIMD architecture. The algorithm assumes the existence of a library routine for the calculation of a 2-d Fourier transform on a set of N_p^2 data points where N_p^2 is the number of processing elements. The paper discusses how to use this routine to calculate 2-d Fourier transforms on a set of N^2 data points where N_p/N is a power of two, using an interleaving technique. The paper also discusses the use of the 2-d results in conjunction with base '$r_1 + r_2$' FFT algorithms to calculate 3-d Fourier transforms on a set of N^3 complex data points. In the final section a general program is described to calculate 3-d Fourier transforms for any values of N and N_p such that N_p/N is a power of two. Timings are given for the algorithms run on an ICL Distributed Array Processor.

Introduction

There are many examples of problems in several different fields where it is necessary to be able to compute quickly multi-dimensional Fourier transforms (FTs). Examples of the use of multi-dimensional FTs range from scattering theory to fluid dynamics, solution of Poisson's equation on a mesh [3] and image processing via filtering techniques. The computation of these transforms should be a problem well suited to Highly Parallel Computers (HPCs), however to obtain the greatest efficiency from using this type of computer it is important that the calculation should be organised in such a way as to utilise the parallelism implicit in the problem. To simplify the writing of the code, use is made of readily available Fast Fourier Transform (FFT) routines. It is assumed that for a HPC with N_p^2 processing elements, there will be a routine available to compute the 2-d FT of a set of N_p^2 complex data points (for an example of a 2-d FFT routine on the Distributed Array Processor (DAP) see [2]). The present paper is divided into three sections. In the first section we will discuss exactly what is meant by a HPC, and in particular the features needed on such a computer to be able to use the algorithms given here. As the specific examples given are for the ICL DAP there is also a brief discussion of some features of the high level language DAP FORTRAN.

In the second section 2-dimensional transforms are discussed, and in particular a method for calculating several $N \times N$ transforms simultaneously using one call to an $N_p \times N_p$ 2-d FT routine (applicable when $N_p/N = 1, 2, 4, 8, \ldots$). This is achieved by 'interleaving' the input matrices in such a way that their transforms can be easily unscrambled. An efficient algorithm for doing the interleaving is also given.

In the final section 3-dimensional FTs will be discussed, and in particular a method of using the results from Section 2 in combination with base '$m + n$' FFTs so as to maximise the parallelism in the problem to obtain 3-d FFT routines that run as quickly as possible. A method for using an algorithm to write a general 3-d FFT program (as opposed to a program for specific values of N and N_p) will also be discussed.

The algorithm developed below was specifically designed for use in a program to solve Poisson's equation on a mesh in a molecule dynamics simulation. As this routine has to be called twice every timestep and as a molecular dynamics program can run for several tens of thousands of timesteps the major concern was to develop a program that ran as quickly as possible rather than one which used the minimum amount of memory. By putting the extra memory used by the 3-d FT routine into a common block, the workspace is made available for use in other parts of the molecular dynamics program. If the amount of memory used is important, then a different FFT should be used for doing the 1-d N-point transform in Section 3. A wide range of such transforms is available [6].

1. The computer

The type of computer considered is an array processor made up of a number of processing elements (PEs) each with its own local storage. Each PE obeys the same instruction simultaneously but uses its own local data, thus operating in a Single Instruction Multiple Data (SIMD) mode. The N_p^2 processing elements are considered as lying on a square $N_p \times N_p$ grid with the local store of each PE being regarded as a third dimension to give a structure of the form as shown in Fig. 1. Each PE has connections with its four nearest neighbours. These connections allow for planar data movements in which all the data in one plane is simultaneously shifted in any one of the four nearest neighbour directions. The algorithms require that there be two choices for the geometry of the data plane. With 'planar' geometry in operation data shifted off the edge of the plane disappears and data entering an edge is set to zero. With 'cyclic' geometry in operation data shifted off one edge reappears at the opposite edge. By using a bit plane of 'activity control' bits (the activity control plane can be thought of as a logical mask) the PEs can be given limited local control. By using such a logical mask,

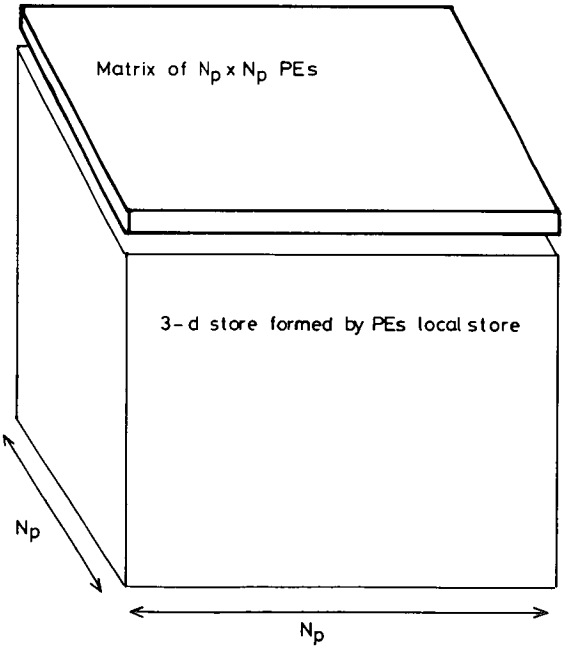


Fig. 1. PE matrix and DAP storage.

operations at PEs whose corresponding mask bit is .FALSE. can be suppressed. Logical masks can also be used to 'merge' two data planes. Where a PE has a mask bit set to .TRUE. it takes data from the first data plane and where it is set to .FALSE. it takes data from the second data plane. An example of such a machine is the ICL Distributed Array Processor (DAP) which has 64^2 PEs. (Goodyear Aerospace's MPP and GEC's GRID are based on similar designs). As the example given in this paper is written in DAP FORTRAN, the language designed by ICL to perform parallel computations on the DAP, it might be useful to review some of the relevant features of that language.

DAP FORTRAN is similar in many ways to FORTRAN IV but with extended data structures and instructions to handle the parallelism of the DAP. Algebraic and logical data is organised into three main forms: scalars, vectors and matrices. Scalars can be treated essentially as scalar variables in FORTRAN IV. Vectors are 1-d arrays of 64 data items stored horizontally on single bit-planes. Matrices are 64×64 data arrays stored vertically (see Fig. 2). In DAP FORTRAN a real 64×64 matrix would be declared using the statement

REAL $X(,)$

where the (,) means, by definition, that X is a 64×64 matrix. If $X(,)$, $Y(,)$ and $Z(,)$ are declared as real matrix variables then a statement such as

$$X(,) = Y(,) + Z(,)$$

would cause all 4096 elements in $Y(,)$ to be added to all 4096 corresponding elements of $Z(,)$ simultaneously, the results being placed in $X(,)$, thus 4096 additions are performed simultaneously by a single add instruction which has been broadcast to all the PEs. It is not necessary to write a matrix variable as $X(,)$, for example, writing it simply as X is also permissible in DAP FORTRAN providing it has been declared as being of type matrix. However writing it as $X(,)$ does make it obvious that the variable being considered is of type matrix. If $L(,)$ is defined to be a logical matrix (one bit-plane), then the instruction

$$X(,) = \text{MERGE}(Y(,), Z(,), L(,))$$

has the effect of assigning to the matrix $X(,)$ the data in $Y(,)$ if the corresponding mask bit is .TRUE. and the data in $Z(,)$ if the corresponding mask bit is .FALSE. As described earlier, logical masks can also be used to suppress operations on certain PEs. The instruction

$$X(L(,)) = Y(,) + Z(,)$$

would first calculate all 4096 elements of the matrix $Y(,) + Z(,)$, however these calculated values will only be assigned to elements of $X(,)$ if the corresponding mask bit is .TRUE. If the mask bit is .FALSE., the corresponding element of $X(,)$ will be unchanged. The other useful instructions are those that shift data on a data plane. The instruction

$$X(,) = \text{SHNC}(Y(,), 1)$$

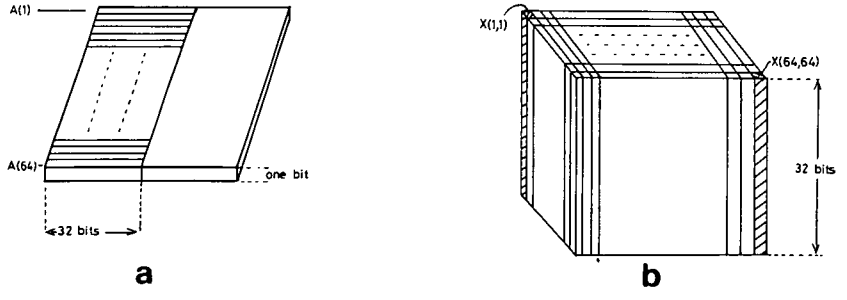


Fig. 2. (a) Storage of the REAL * 4 vector $A()$ on the DAP; (b) Storage of the REAL * 4 matrix $X(,)$ on the DAP.

(*SH*ift *N*orth *C*yclic) would assign to $X(i, j)$ the value of $Y(i+1, j)$, the coordinates being calculated modulo 64 as cyclic geometry is selected. Further details of the DAP FORTRAN language are given in [4,5].

2. The 2-dimensional transform

This section covers the problem of efficiently computing a 2-d FT on a $N \times N$ set of complex data points using a HPC with N_p^2 PEs. Obviously if $N = N_p$ there is no problem, simply use the library routine, but what happens if $N \neq N_p$? One could attempt to write the complete 2-d FT routine from scratch, but writing FFT routines can be a long, intricate and laborious job. In addition a new routine would have to be written for each value of N_p/N. There is however a fairly straightforward way of using the library 2-d FFT routine to do this task, provided that the ratio N_p/N is a power of 2. The technique presented here for doing this can be generalised to cope with ratios that are powers of numbers other than 2, but as for all HPCs currently availabe N_p is a power of two, $N_p/N = 1, 2, 4, 8, \ldots$ should cover most cases encountered in practice.

To use the library routine efficiently we must find a way of doing several $N \times N$ FTs with one call of the library $N_p \times N_p$ FT routine; for example if $N_p/N = 2$ we would like to be able to compute four $N \times N$ FT's simultaneously. It can be shown that once we have solved the problem for $N_p/N = 2$, then we have effectively solved the problem for any value of this ratio that is a power of 2.

2.1. *Theory for the ratio* $N_p/N = 2$

Figure 3(a) represents the initial situation. On one data plane of the HPC we have placed four of the $N \times N$ matrices to be transformed. In the notation used capital letters refer to transformed data and lower case letters refer to the original data, so $\boldsymbol{a}$ would refer to an $N \times N$ matrix and $A(i, j)$ would refer to the element in the ith row and jth column of the FT of $\boldsymbol{a}$. We would like a routine that will take us from the situation in Fig. 3(a), to the situation in Fig. 3(b) with only one call to the library FT routine. The problem is to find a way of organising the data from the four $N \times N$ matrices such that on calling the library FT routine the matrix shown in Fig. 3(b) can be readily recovered. This can be achieved by 'interleaving' the four matrices in such a way that the desired transforms appear modified only by known phase factors. Start by

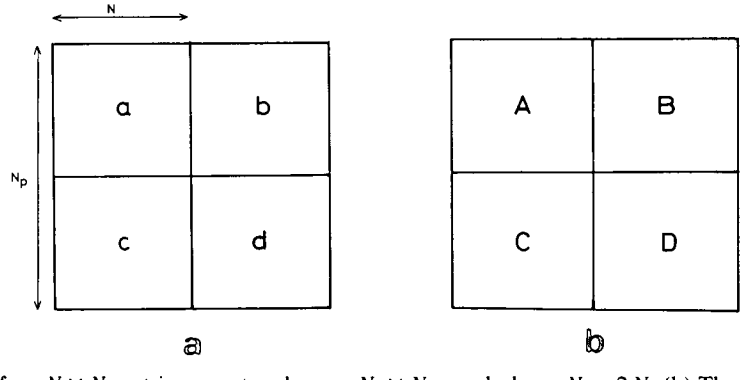


Fig. 3. (a) How the four $N \times N$ matrices are stored on an $N_p \times N_p$ word-plane, $N_p = 2N$; (b) The storage of the FTs of the four $N \times N$ matrices in (a).

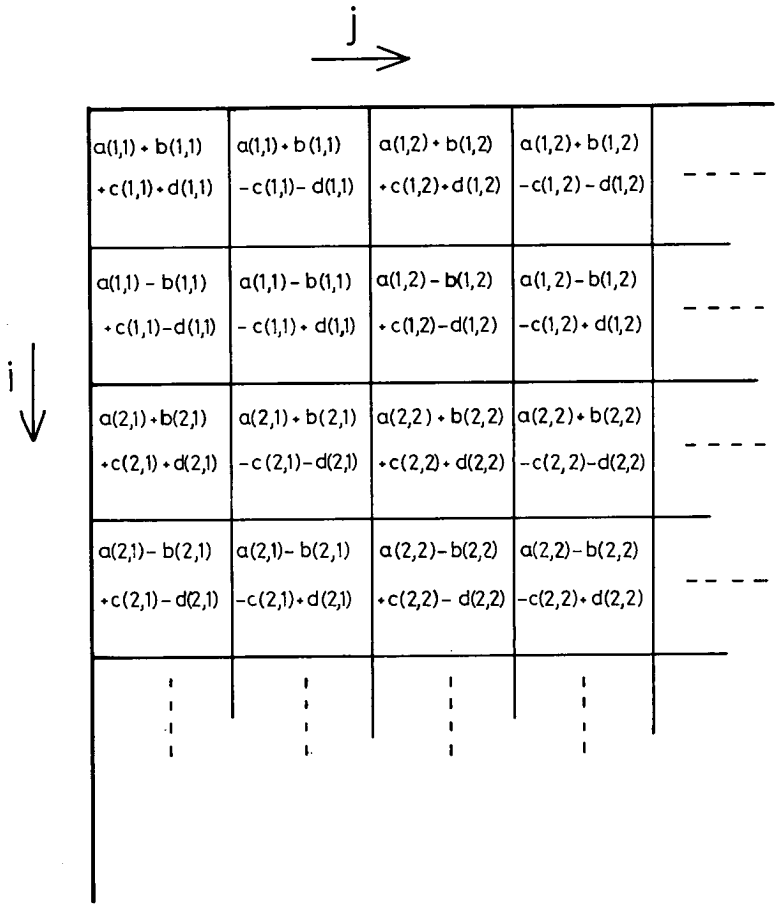


Fig. 4. The form of the restructured data.

restructuring the data as shown in Fig. 4. This restructuring is a non-trivial computational problem and in the next sub-section an algorithm will be discussed for achieving this 'interleaving'.

Let $\boldsymbol{X}$ be the $N_p \times N_p$ matrix representing the numbers on the data plane after the interleaving of the four $N \times N$ matrices has occured. The 2-d $N_p \times N_p$ FT of this matrix is defined as

$$X(k_x, k_y) = \sum_{n_x, n_y=0}^{N_p-1} x(n_x, n_y) \exp\left[-2\pi i(n_x k_x + n_y k_y)/N_p\right]. \tag{2.1}$$

Equation (2.1) can now be split into four terms; n_x and n_y both even, n_x even and n_y odd, n_x odd and n_y even, n_x and n_y both odd:

$$\begin{aligned} X(k_x, k_y) = \sum_{n_x, n_y=0}^{N-1} \{ & x(2n_x, 2n_y) \exp\left(-2\pi i\left[n_x k_x + n_y k_y\right]/N\right) \\ & + x(2n_x, 2n_y+1) \exp\left(-2\pi i\left[n_x k_x + n_y k_y\right]/N\right)\phi_y(k_y) \\ & + x(2n_x+1, 2n_y) \exp\left(-2\pi i\left[n_x k_x + n_y k_y\right]/N\right)\phi_x(k_x) \\ & + x(2n_x+1, 2n_y+1) \exp\left(-2\pi i\left[n_x k_x + n_y k_y\right]/N\right)\phi_x(k_x)\phi_y(k_y)\} \end{aligned} \tag{2.2}$$

where

$$\phi_x(k_x) = \exp(-2\pi i k_x/N_p), \qquad \phi_y(k_y) = \exp(-2\pi i k_y/N_p). \tag{2.3}$$

Note that an expression such as the first term of (2.2) is simply

$$A(k_x, k_y) + B(k_x, k_y) + C(k_x, k_y) + D(k_x, k_y) \tag{2.4}$$

by the way that the interleaved matrix is set up, so by defining

$$\begin{aligned} A' &= A + B + C + D, & C' &= A - B + C - D, \\ B' &= A + B - C - D, & D' &= A - B - C + D, \end{aligned} \tag{2.5}$$

it is possible to rewrite (2.2) in the form

$$\begin{aligned} X(k_x, k_y) = A'(k_x, k_y) &+ B'(k_x, k_y)\phi_y(k_y) + C'(k_x, k_y)\phi_x(k_x) \\ &+ D'(k_x, k_y)\phi_x(k_x)\phi_y(k_y). \end{aligned} \tag{2.6}$$

Now split the matrix X into four sections

$$\begin{aligned} X_{00} &= \{ X(k_x, k_x): 0 \leqslant k_x, k_y \leqslant N-1 \}, \\ X_{10} &= \{ X(k_x + N, k_y): 0 \leqslant k_x, k_y \leqslant N-1 \}, \\ X_{01} &= \{ X(k_x, k_y + N): 0 \leqslant k_x, k_y \leqslant N-1 \}, \\ X_{11} &= \{ X(k_x + N, k_y + N): 0 \leqslant k_x, k_y \leqslant N-1 \}. \end{aligned} \tag{2.7}$$

Note that as $N_p = 2N$, then using (2.3)

$$\phi_x(k_x + N) = -\phi_x(k_x), \quad \phi_y(k_y + N) = -\phi_y(k_y) \tag{2.8}$$

so that in terms of X_{00}, X_{10}, X_{01} and X_{11} eq. (2.6) can be rewritten as

$$\begin{aligned} X_{00} &= A' + B'\phi_y + C'\phi_x + D'\phi_x\phi_y, & X_{01} &= A' - B'\phi_y + C'\phi_x - D'\phi_x\phi_y, \\ X_{10} &= A' + B'\phi_y - C'\phi_x - D'\phi_x\phi_y, & X_{11} &= A' - B'\phi_y - C'\phi_x + D'\phi_x\phi_y \end{aligned} \tag{2.9}$$

where

$$\phi_x = \{ \phi_x(k_x): 0 \leqslant k_x \leqslant N-1 \}, \qquad \phi_y = \{ \phi_y(k_y): 0 \leqslant k_y \leqslant N-1 \}. \tag{2.10}$$

Equations (2.9) now give us four simultaneous equations for A', B', C' and D' which can be solved to give

$$\begin{aligned} 4A' &= X_{00} + X_{10} + X_{01} + X_{11}, & 4C' &= (X_{00} - X_{10} + X_{01} - X_{11})\phi_x^*, \\ 4B' &= (X_{00} + X_{10} - X_{01} - X_{11})\phi_y^*, & 4D' &= (X_{00} - X_{10} - X_{01} + X_{11})\phi_x^*\phi_y^* \end{aligned} \tag{2.11}$$

where * represents complex conjugation. Equations (2.11) can then be substituted ionto (2.5) to give

$$\begin{aligned} 4A &= (A' + B' + C' + D'), & 4C &= (A' - B' + C' - D'), \\ 4B &= (A' + B' - C' - D'), & 4D &= (A' - B' - C' + D'). \end{aligned} \tag{2.12}$$

The calculation of the inverse transform is done using the result

$$x(n_x, n_y) = \left\{ \sum_{n_x, n_y=0}^{N-1} X^*(k_x, k_y) \exp\left[-2\pi i(k_x n_x + k_y n_y)/N\right]/N^2 \right\}^*. \tag{2.13}$$

The inverse transform is performed by taking the complex conjugate of the input matrices, taking the normal 2-d FT and finally taking the complex conjugate once more and normalising by $1/N^2$.

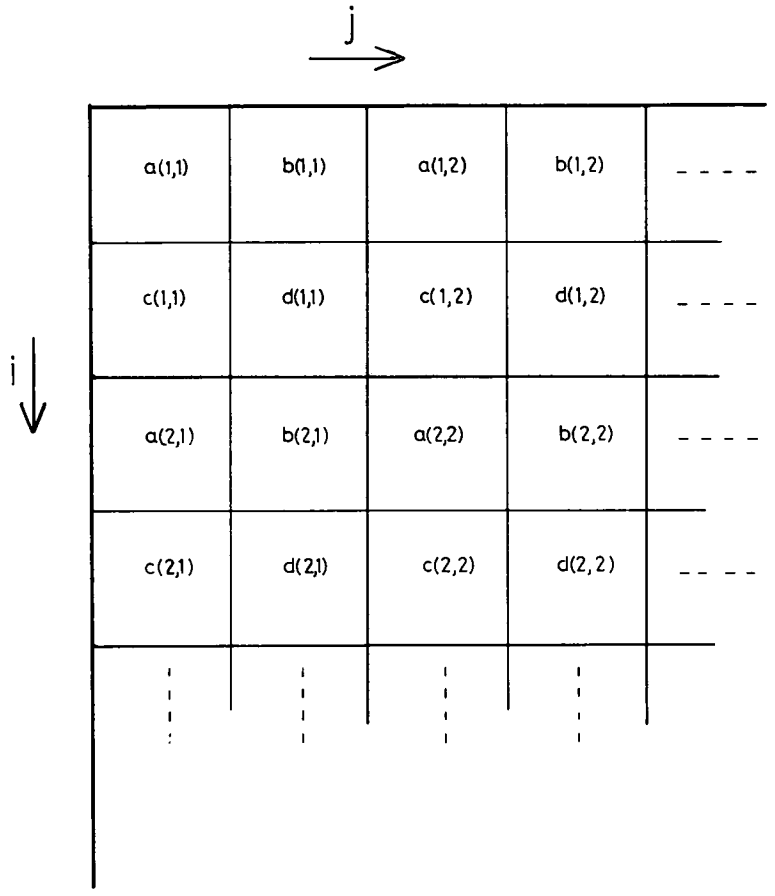


Fig. 5. The first stage in restructuring the data.

2.2. Computational details

The first problem encountered in coding this algorithm is how to change a matrix in the form of Fig. 3(a) into one in the form of Fig. 4 efficiently. If we can construct a matrix in the form of Fig. 5, then the problem is solved; once we have a matrix in this form, a matrix of the form in Fig. 4 can be be constructed by using the shift facilities. Take the case $N_p = 16$. Consider first the problem of turning a matrix of the form of Fig. 6(a) into a matrix of the form in Fig. 6(c). We could get the latter by shifting each column independently, making a total of 7 operations. However using a parallel computer one would like to be able to reduce this to 3 ($= \log_2 8$) operations using some form of binary algorithm. On inspection of the interleaved matrix it can be seen that the first four columns of the original matrix are transferred onto the first 8 columns of the interleaved matrix and that the next 4 columns are transferred onto the last 8 columns of the interleaved matrix. The first step in the algorithm is to shift the east half of ***a*** 4 columns to the east (Fig. 6(b)). The next step is to shift the east halves of the new columns to the east by 2. After repeating this procedure once more, reducing the shift to one columns, the required matrix is obtained (Figs. 6(c) and (d)). This algorithm can be used for any value of N_p that is a power of 2.

It is now not too difficult to adapt this algorithm for the job required. Start by interleaving the matrix from Fig. 3(a) in the E–W direction, ***a*** and ***c*** to the east, ***b*** and ***d*** to the west. Then repeat the procedure in the N–S direction. The code for doing this on the DAP with $N = 32$,

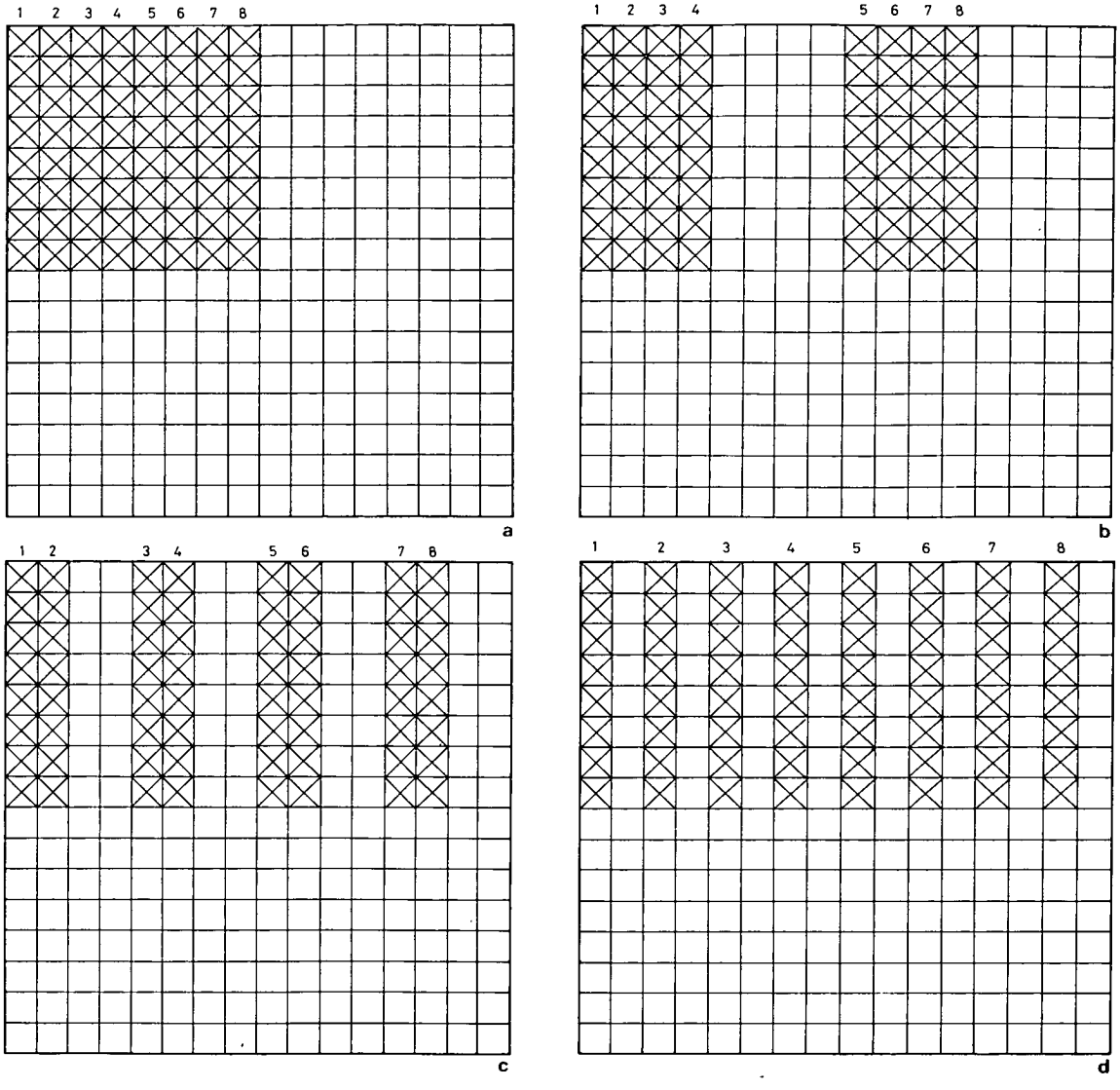


Fig. 6. The stages involved in spacing out the columns of an $N \times N$ matrix on an $N_p \times N_p$ word-plane; $N = 8$, $N_p = 16$.

$N_p = 64$ is shown below:

```
1.      REAL X(,), X1(,)
2.      X1 = X
3.      N = 32
4.  C   this loop does the interleaving in the E–W direction
5.      DO 1 I = 1, 5
6.        N2 = N
7.        N = N/2
8.        X = MERGE(SHEP(X,N), X,ALTC(N2))
9.        X1 = MERGE(X1,SHWP(X1,N),ALTC(N2))
10.  1  CONTINUE
11.     X = MERGE(X1, X,ALTC(1))
```

Similarly for the N–S direction.

The very fast instruction ALTC(N) used in lines 8, 9 and 11 produces a logical mask of which the first N columns are .FALSE., the next N columns are .TRUE., the next N columns are .FALSE. etc. The command ALTR(N) has the analogous effect on rows. The interleaving is done in lines 5–10. Line 8 spaces out the matrices ***a*** and ***c*** to the east and line 10 spaces out matrices ***b*** and ***d*** to the west. The two matrices constructed, $X(,)$ and $X1(,)$, are then interleaved together using the MERGE on line 11. The division by 2 on line 7 could be done more efficiently by equivalencing N to a logical vector and replacing the division by a shift. This algorithm can be written much more compactly in the form

```
      N = 32
      DO 1 I = 1, 5
        NH = N/2
C       E–W interleaving
      X(SHWC(ALTC(N),NH)) = MERGE(SHWP(X,NH),SHEP(X,NH),ALTC(NH))
C       N–S interleaving
      X(SHNC(ALTR(N),NH)) = MERGE(SHNP(X,NH),SHSP(X,NH),ALTR(NH))
      N = NH
1     CONTINUE
```

For $N \neq 32$ the program is easily changed. Replace the 32s by N and the 5s by $\log_2 N$. Once the matrix has been got into the form of Fig. 5, the interleaving procedure can be completed by

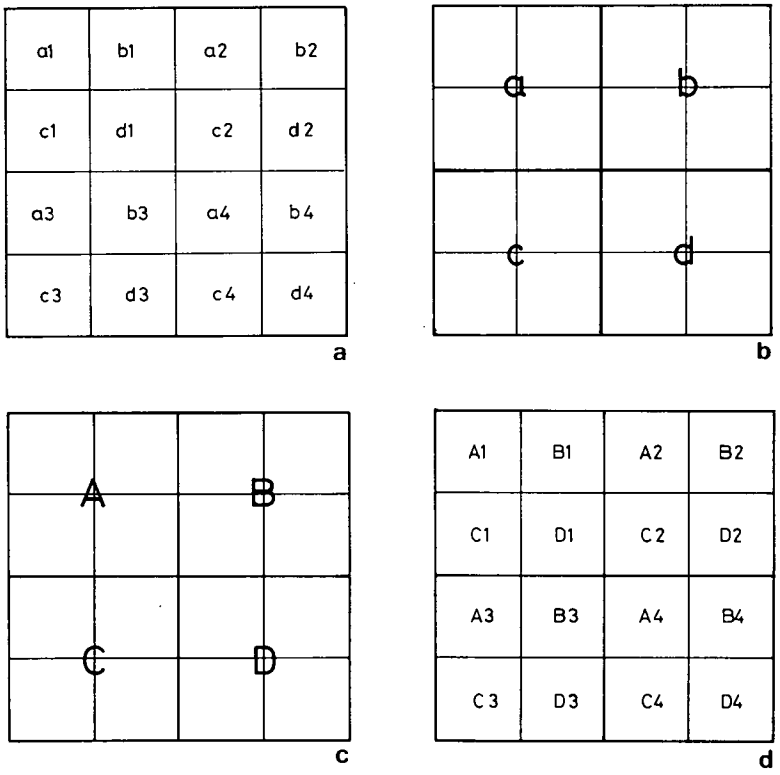


Fig. 7. (a) The storage of the 16 $N \times N$ matrices on the word-plane; (b) The situation after one call to the interleaving routine. The $2N \times 2N$ matrix ***a*** is made up of the four $N \times N$ matrices ***a*1**, ***b*1**, ***c*1**, ***d*1** in an analogous way to the matrix in Fig. 4, similarly for the matrices ***b*** ***c***, ***d***; (c) The situation after one call to the $N_p/2 \times N_p/2$ FT routine and the unscrambling routine; (d) The situation after one more call to the unscrambling routine.

using shifts and logical masks. For instance if $X(,)$ is a matrix in the form of Fig. 5, then the code

```
  X = X + MERGE(-SHEP(X,1),SHWP(X,1),ALTC(1))
  X(ALTR(1) .LNEQ. ALTC91)) =
*   - MERGE(SHSP(SHWP(X,1),1),SHNP(SHEP(X,1),1),ALTR(1))
  X = X + MERGE(SHEP(X,1), - SHWP(X,1),ALTC(1))
```

will produce a matrix in the form of Fig. 4.

On the DAP the 2-d 64×64 REAL * 4 FFT routine runs in 30 ms. When the above algorithm was coded with $N_p = 64$ and $N = 32$ the program computed four 32×32 complex FTs in 52 ms.

2.3. Adapting the $N_p/N = 2$ theory for N_p/N equal to any power of 2

We have now covered the cases where $N_p/N = 1$ (trivial) and $N_p/N = 2$. What happens when $N_p/N = 4$ (Fig. 7(a)? The problem can be split into two parts. First use the interleaving routine with N_p replaced by $N_p/2$. This will produce the situation as in Fig. 7(b). Now use the interleave routine a second time with $N_p/2$ as the parameter and then call the library FT routine. Using the unscrambling algorithm described in Section 2.1 we get to the position shown in Fig. 7(d), i.e. we have the $N_p/2 \times N_p/2$ Fourier transforms of the four quarters of the matrix in Fig. 7(b). By repeating the unscrambling, this time with N_p replaced by $N_p/2$, we obtain the 16 transformed matrices in their correct positions (Fig. 7(d)). The case $N_p/N = 8$ can be similarly computed, this time using 3 calls to both the interleaving and unscrambling routines.

3. The 3-dimensional transform

Consider a 3-d FT on a set of N^3 complex data points. The transform is defined by

$$X(k_x, k_y, k_z) = \sum_{n_x, n_y, n_z = 0}^{N-1} x(n_x, n_y, n_z) \exp[-2\pi i k_x n_x/N] \times \exp[-2\pi i k_y n_y/N] \exp[-2\pi i k_z n_z/N], \tag{3.1}$$

i.e. the 3-d FT can be regarded as a set of three nested one dimensional transforms. Define (3.1) as

$$\boldsymbol{X} = \{ X(k_x, k_y, k_z) : 0 \leqslant k_x, k_y, k_z \leqslant N-1 \} \tag{3.2}$$

and

$$X(\boldsymbol{k}_x, \boldsymbol{k}_y, n_z) = \left\{ \sum_{n_x, n_y = 0}^{N-1} x(n_x, n_y, n_z) \exp[-2\pi i n_x k_x/N] \times \exp[-2\pi i n_y k_y/N] : 0 \leqslant k_x, k_y \leqslant N-1 \right\}. \tag{3.3}$$

Then using this notation the 3-d FT (3.1) can be written as

$$\mathbb{X} = \sum_{n_z = 0}^{N-1} X(\boldsymbol{k}_x, \boldsymbol{k}_y, n_z) \exp[-2\pi i n_z k_z/N]. \tag{3.4}$$

The term $X(k_x, k_y, n_z)$ can be regarded as the 2-d FT of the data set

$$\{X(n_x, n_y, n_z): 0 \leqslant n_x, n_y \leqslant N-1\}. \tag{3.5}$$

Thus the 3-d transform of a set of N^3 complex data points can be thought of as the N-point 1-D transform of a set of N $N \times N$ data sets. The 2-d transforms can be computed efficiently using the results of the previous section and the 1-d N-point transform for the third dimension can be calculated using what is called a base '$r_1 + r_2$' FFT.

3.1. Base '$r_1 + r_2$' Fast Fourier Transforms

In calculating the 2-d transform we did not have to code any FFT routines explicitly as they were available in the library FFT routine. However, to calculate the 1-d FT efficiently we must code such a two base algorithm. Consider an N-point FT where $N = r_1 r_2$:

$$X(k) = \sum_{n=0}^{N-1} x(n) \exp[-2\pi i kn/N]. \tag{3.6}$$

Rewrite k and n in eq. (3.6) as

$$\begin{aligned} k &= k_1 r_1 + k_0, \quad 0 \leqslant k_0 \leqslant r_1 - 1, \quad 0 \leqslant k_1 \leqslant r_2 - 1, \\ n &= n_1 r_2 + n_0, \quad 0 \leqslant n_0 \leqslant r_2 - 2, \quad 0 \leqslant n_1 \leqslant r_1 - 1. \end{aligned} \tag{3.7}$$

Then substituting n, eq. (3.6) can be rewritten as

$$X(k) = \sum_{n_0=0}^{r_2-1} \left[\sum_{n_1=0}^{r_1-1} x(n_1, n_0) W^{kn_1 r_2} \right] W^{kn_0} \tag{3.8}$$

where

$$W = \exp(-2\pi i/N). \tag{3.9}$$

But $W^N = 1$ therefore

$$W^{kn_1 r_2} = W^{(k_1 r_1 + k_0) n_1 r_2} = W^{k_0 n_1 r_2}. \tag{3.10}$$

So substituting k in eq. (3.8)

$$X(k_1, k_0) = \sum_{n_0=0}^{r_2-1} \left(\sum_{n_1=0}^{r_1-1} x(n_1, n_0) W^{k_0 n_1 r_2} \right) W^{(k_1 r_1 + k_0) n_0}. \tag{3.11}$$

Regrouping the terms in W (known as 'twiddle' factoring), eq. (3.11) can be written in the form

$$X(k_1, k_0) = \sum_{n_0=0}^{r_2-1} \left(\sum_{n_1=0}^{r_1-1} x(n_1, n_0) W^{k_0 (n_1 r_2 + n_0)} \right) W^{k_1 n_0 r_1}, \tag{3.12}$$

or in recursive form as

$$\begin{aligned} x_1(k_0, n_0) &= \left[\sum_{n_1=0}^{r_1-1} x(n_1, n_0) W^{k_0 n_1 r_2} \right] W^{k_0 n_0}, \\ x_2(k_0, k_1) &= \sum_{n_0=0}^{r_2-1} x_1(k_0, n_0) W^{k_1 n_0 r_1}, \\ X(k_1, k_0) &= x_2(k_0, k_1), \end{aligned} \tag{3.13}$$

i.e. the final results come out in 'base' reversed form.

3.2. Computational details

Consider the specific example of calculating a 3-d FT on an N^3 set of complex data points with $N_p = 64$ and $N = 32$. Consider how the data is stored in the computer. The data is stored over two arrays, $X(,, N_z)$ for the real data parts and $Y(,, N_z)$ for the imaginary parts, where $N_z = N^3/N_p^2$ which equals 8 in the example. For the sake of compactness the expression $z(a, b)$ is used to represent the 32×32 matrix $x(a, b) + i\,y(a, b)$. Capital letters refer to arrays on the computer, for instance $X(,, 2)$ is the second data plane of the real data and, bold type letters refer to 32×32 matrices, for example $\boldsymbol{x}(a, b)$ is the $(r_2 a + b)$th 32×32 matrix. Similarly $Z(,, 2)$ represents $X(,, 2) + iY(,, 2)$ (Fig. 8).

We can calculate the terms

$$z(k_x, k_y, n_z), \quad n_z = 0, \ldots, 31$$

by using the 2-d FT algorithm developed in Section 2, simply by calling the routine 8 times, once for each data plane. We now have to perform a 32-point transform on the 32 32×32 complex data points produced by using the 2-d routine. Substituting $r_1 = N_z = 8$, $r_2 = 4$, $N = 32$ into (3.13)

$$z(k_1, k_0) = \left(\sum_{n_1=0}^{r_1-1} z(n_1, n_0) W^{4k_0 n_1} \right) W^{k_0 n_0}. \tag{3.14}$$

Define

$$z_1'(k_1, n_0) = \sum_{n_1=0}^{r_1-1} z(n_1, n_0) W^{4k_0 n_1}. \tag{3.15}$$

Calculate all the z_1' terms first and then multiply eachy by $W^{n_0 k_0}$. The calculation of the $z_1'(k_1, n_0)$ terms fits nicely onto the HPC (this being the original reason for choosing this FFT

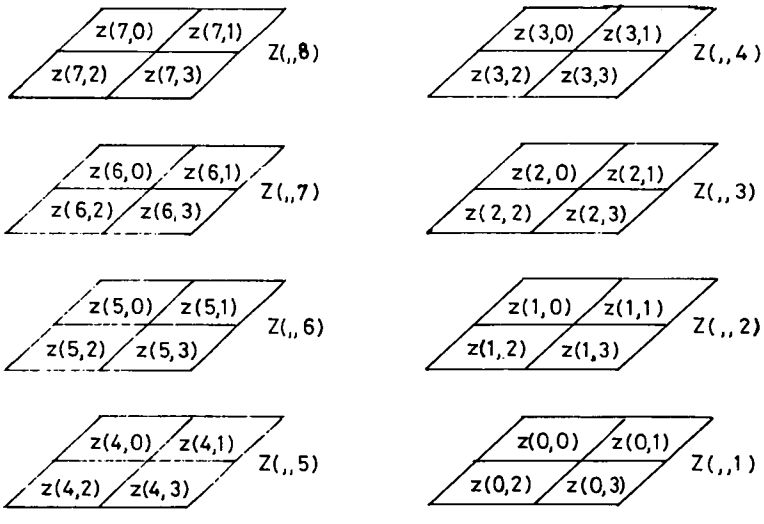


Fig. 8. The storage of the 32 32×32 matrices on the complex array $Z(,,8)$. $z(a, b)$ refers to the 32×32 matrix $z(n_x, n_y, (r_2 a + b))$.

algorithm). Look at the calculation of $z_1'(k_0, i)$, $0 \leqslant i \leqslant 3$ (note that the four 32×32 matrices $z(k_1, i)$ all lie on the k_0 data plane)

$$z_1'(k_0, i) = z(0, i) + W^{4k_0}z(1, i) + W^{8k_0}z(2, i) + W^{12k_0}z(3, i) + W^{16k_0}z(4, i) + W^{20k_0}z(5, i) + W^{24k_0}z(6, i) + W^{28k_0}z(7, i). \tag{3.16}$$

All the matrices $z_1'(k_1, i)$ can be calculated simultaneously for $0 \leqslant k_0 \leqslant 7$.

The next stage in the calculation is to multiply each of the $z_1'(k_0, n_0)$ by $W^{n_0k_0}$. This again can be done efficiently in parallel. Set up a complex matrix Q:

$$Q(,) = \begin{cases} 1, & 1 \leqslant i, j \leqslant 32, \\ W, & 1 \leqslant i \leqslant 32, 33 \leqslant j \leqslant 64, \\ W^2, & 33 \leqslant i \leqslant 64, 1 \leqslant j \leqslant 32, \\ W^3, & 33 \leqslant i, j \leqslant 64. \end{cases} \tag{3.17}$$

Then the multiplication of each $z_1'(k_0, n_0)$ by $W^{n_0k_0}$ is equivalent to multiplying $Z(,, k_0)$ by Q^{k_0}. Having calculated the z_1 matrices, continue with the base 4 part of the algorithm:

$$z_2(k_0, k_1) = \sum_{n_0=0}^{r_2-1} z_1(k_0, n_0)W^{8k_1n_0} = z_1(k_0, 0) + W^{8k_1}z_1(k_0, 1) + W^{16k_1}z_1(k_0, 2) + W^{24k_1}z_1(k_0, 3), \tag{3.18}$$

Therefore,

$$\begin{aligned} z_2(k_0, 0) &= z_1(k_0, 0) + z_1(k_0, 1) + z_1(k_0, 2) + z_1(k_0, 3), \\ z_2(k_0, 1) &= z_1(k_0, 0) + W^8z_1(k_0, 1) + W^{16}z_1(k_0, 2) + W^{24}z_1(k_0, 3), \\ z_2(k_0, 2) &= z_1(k_0, 0) + W^{16}z_1(k_0, 1) + z_1(k_0, 2) + W^{16}z_1(k_0, 3), \\ z_12(k_0, 3) &= z_1(k_0, 0) + W^{24}z_1(k_0, 1) + W^{16}z_1(k_0, 2) + W^8z_1(k_0, 3), \end{aligned} \tag{3.19}$$

but as the coefficients have a simple form,

$$W^8 = \exp(-\pi i/2) = -i, \qquad W^{16} = \exp(-\pi i) = -1, \qquad W^{24} = \exp(-3\pi i/2) = i, \tag{3.20}$$

the eqs. (3.19) can be rewritten as

$$\begin{aligned} z_2(k_0, 0) &= z_1(k_0, 0) + z_1(k_0, 1) + z_1(k_0, 2) + z_1(k_0, 3), \\ z_2(k_0, 1) &= z_1(k_0, 0) - iz_1(k_0, 1) - z_1(k_0, 2) + iz_1(k_0, 3), \\ z_2(k_0, 2) &= z_1(k_0, 0) - z_1(k_0, 1) + z_1(k_0, 2) - z_1(k_0, 3), \\ z_2(k_0, 3) &= z_1(k_0, 0) + iz_1(k_0, 1) - z_1(k_0, 2) - iz_1(k_0, 3). \end{aligned} \tag{3.21}$$

Note that the calculation of the base 4 transform requires no multiplication operations, this being why the 'twiddle' factoring was done. The calculation of the base 4 transform can now be done data plane by data plane using shifts and the operations + and −. The final stage in the calculation of the transform is 'base' reversal (3.13). If Fig. 9(a) represents the ordering of the z_2 matrices on the data planes after the base 4 and base 8 transforms have been made, Fig. 9(b) represents the order in which we must place them to achieve the 'base' reversal (it is not the matrices that are base reversed but their z coordinate). Table 1 shows how the positions in Fig. 9(b) were calculated. This process cannot be coded easily in parallel, the quickest way that it can be done is simply to shift every 32×32 matrix into its required position individually. While

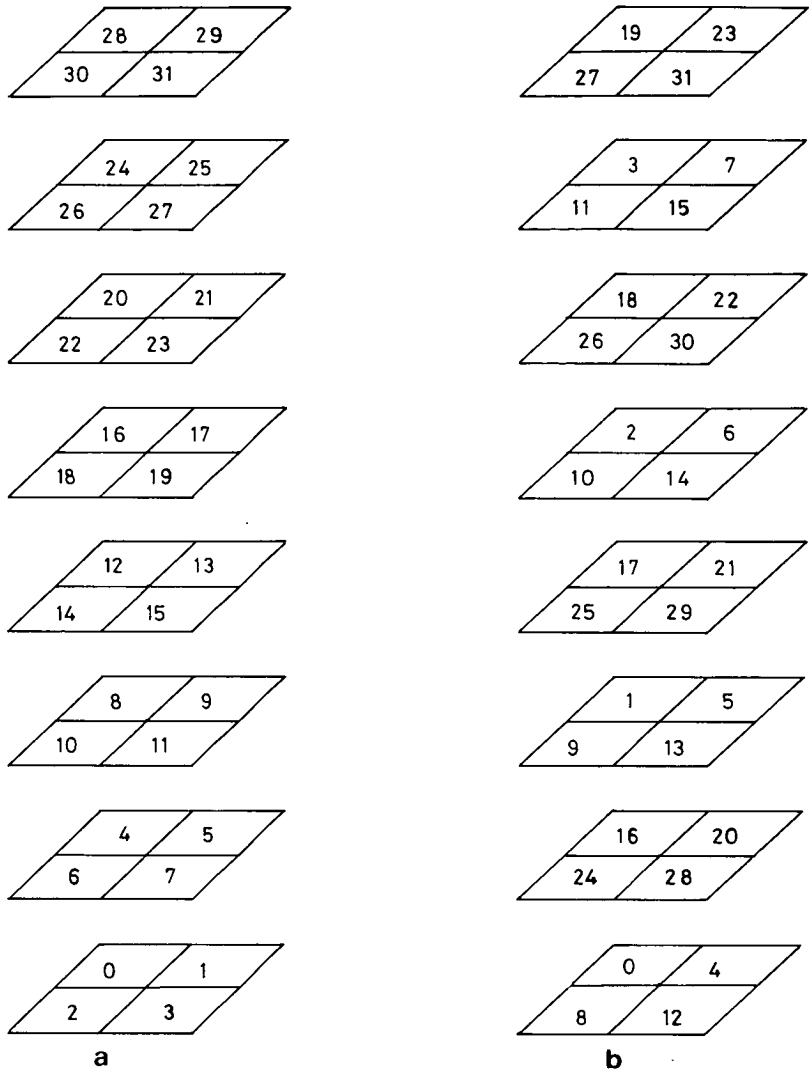


Fig. 9. (a) The 32 32×32 matrices $z_2(k_x, k_y, k_z)$, the numbers refer to the k_z coordinate; (b) The reordering of these matrices after 'base' reversal of the k_z label, see Table 1.

the code to do this is tedious to write it does run quickly, the time taken to achieve the data shifting is insignificant compared with the time taken to perform the 2-d FTs.

When the above algorithm was coded for the DAP, the 3-d FT of a set of 32^3 data points was calculated in 476 ms, of which 416 ms was used in the calculation of the 2-d transforms. To check that the program was working correctly a random complex matrix was compared with the inverse FT of the FT of the same matrix.

3.3. The general 3-dimensional transform

This subsection describes how to write one program to calculate 3-d FTs on a set of N^3 data points using an HPC with N_p^2 PEs where N and N_p can take any values such that N_p/N is a power of two. If the data is stored on a series of r_1 complex data planes each containing r_2 $N \times N$ matrices, then the procedure to calculate this transform is given below.

Table 1

k_z coordinates of z matrices	k_1	k_0	k_0	k_1	k'_z, base reversed k_z coordinates of z_2 matrices
0	0	0	0	0	0
1	0	1	1	0	4
2	0	2	2	0	8
3	0	3	3	0	12
4	0	4	4	0	16
5	0	5	5	0	20
6	0	6	6	0	24
7	0	7	7	0	28
8	1	0	0	1	1
⋮	⋮	⋮	⋮	⋮	⋮
27	3	3	3	3	15
28	3	4	4	3	19
29	3	5	5	3	23
30	3	6	6	3	27
31	3	7	7	3	31

$k_z = 8k_1 + k_0$, $k'_z = 4k_0 + k_1$.

(1) The 2-d transform can be done quite simply, requiring r_3 calls to both the interleaving and unscrambling routines (see Sections 2.2 and 2.3) where $r_3 = \sqrt{r_2}$.

(2) The equation for base "r_1" transform is given in eq. (3.13)

$$x_2(k_0, k_1) = \sum_{n_0=0}^{r_2-1} x_1(k_0, n_0) W^{k_1 n_0 r_1}.$$

Set up a complex vector of length r_1, WFAC(NR1) where

$$\mathrm{WFAC}(k) = W^{r_1}.$$

The code

```
      M = 0
      DO 2 K = 0,NR2-1
        DO 1 N = 0,NR1-1
          M = M + K
          IF (M . GE. NR1) M = M - NR1
          Z(,,K) = Z(,,K) + WFAC(M) * Z1(,,N)
1       CONTINUE
        Z(,,K) = Z(,,K) * Q
        Q = Q * Q
2     CONTINUE
```

calculates the z_2 matrices in the previous equation.

This code would not work in DAP FORTRAN as the language does not allow array indexing from zero or complex variables; Z, $Z1$, W and Q are all complex arrays. The same will be true of all the examples of code given in this section. The multiplication by $W^{n_0 k_0}$ is done, as before, by setting up an $N_p \times N_p$ matrix Q set equal to W^{n_0} at PEs where the $N \times N$ matrix label is n_0 and simply multiplying the k_0^{th} data plane by Q^{k_0}.

(3) Calculate the base "r_2" transform. In the example given this was straightforward as r_2 was small. If r_2 is greater than four, then the excessive data shifting involved in the calculation

of the transform makes the computation inefficient. By splitting the base "r_2" transform into two parts the computation of the transform can be made much more efficient as follows.

The base "r_2" transform is defined by

$$z_2(k_0, k_1) = \sum_{n_0=0}^{r_2-1} z_1(k_0, n_0) W^{k_1 n_0 r_1} \tag{3.22}$$

but for all the transforms considered here r_2 can be factorised, $r_2 = r_3^2$, r_3 an integer. Rewrite k_0 and n_0 as base r_3 numbers:

$$\begin{aligned} k_1 &= r_3 K_1 + K_0, \quad 0 \leqslant K_1, K_0 \leqslant r_3, \\ n_0 &= r_3 N_1 + N_0, \quad 0 \leqslant N_1, N_0 \leqslant r_3. \end{aligned} \tag{3.23}$$

Equation (3.22) can now be rewritten as

$$z_2(k_0, k_1) = \sum_{N_0=0}^{r_3-1} \sum_{N_1=0}^{r_3-1} z_1(k_0, N_1, N_0) W^{r_1(r_3K_1+K_0)(r_3N_1+N_0)} \tag{3.24}$$

or recursively as

$$\begin{aligned} z_2^{(1)}(k_0, K_0, N_0) &= \left\{ \sum_{N_1=0}^{r_3-1} z_1(k_0, N_1, N_0) W^{r_1 r_3 N_0 K_1} \right\} W^{r_1 N_0 K_0}, \\ z_2^{(2)}(k_0, K_0, K_1) &= \sum_{N_0=0}^{r_3-1} z_2^{(1)}(k_0, K_0, N_0) W^{r_1 r_3 N_1 K_0}, \\ z_2(k_0, K_1, K_0) &= z_2^{(2)}(k_0, K_0, K_1) \end{aligned} \tag{3.25}$$

using $r_3^2 = r_2$.

Note how the arguments of the $z_2^{(1)}, z_2^{(2)}$ etc. relate to the way that the $N \times N$ matrices are stored on the data planes of the HPC. The first argument determines the complex data plane on which the $N \times N$ matrix is stored and the second and third arguments determine the position of the matrix on the data plane (Fig. 10). The reason for splitting up the base "r_2" part of the algorithm becomes obvious when we look at the calculation of eqs. (3.25). The $z_2^{(2)}$ matrices are stored in the data planes $Z(,,i)$ and the z_1 matrices in the data planes $Z1(,,i)$.

Consider the calculation of $z_2^{(2)}(k_0, K_0, i)$ for $0 \leqslant i \leqslant r_3$, the K_0th row of $N \times N$ matrices on the k_0th data plane. The K_0th row of $N \times N$ $z_2^{(1)}$ matrices is obtained by multiplying the nth row of $N \times N$ z_1 matrices by $W^{r_1 r_3 K_1 (n-1)}$ and then summing the r_3 $N \times N$ matrices along a column and storing the resultant row of matrices in the K_0th row of the $z_2^{(2)}$ data plane. The multiplication by $W^{r_3 K_0 N_0}$ follows in a similar way to the case in Section 2.3: set up an $N_p \times N_p$ matrix QFAC(,) which has the value $W^{r_3 K_0 N_0}$ at positions corresponding to the $N \times N$ matrix whose block coordinates are (K_0, N_0) for all N_0 and K_0. The multiplication by $W^{r_3 K_0 N_0}$ simply becomes a multiplication of each of the data planes by QFAC(,). This transform operates on one data plane at a time and so can be done inside a loop over data planes.

Initialise an $N_p \times N_p$ matrix $Q(,)$ such that the nth set of rows of $Q(,)$ are $W^{r_1 r_3 n}$, form a logical mask LMASK(,) = ALTR(N) and set QFAC(,) = $W^{r_1 N_0 K_0}$. Then the code to calculate the transform for data plane M is

```
    DO 1 K0 = 0,NR3 - 1
      Z(LMASK, M) = SUMROW((Z1(,,M) * Q), N)
      LMASK = SHSC(LMASK, N)
1   CONTINUE
    Z(,,M) = Z(,,M) * QFAC
```

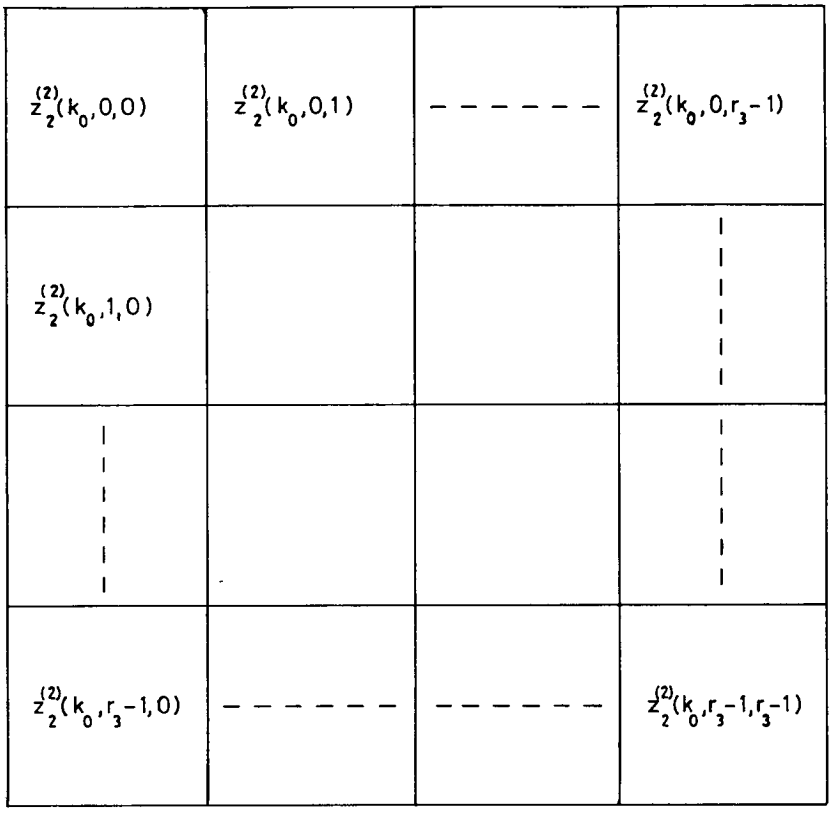


Fig. 10. Storage of the $r_2(=r_3^2)$ $z_2^2(k_0, i, j)$ matrices; $0 \leqslant i \leqslant r_3-1, 0 \leqslant j \leqslant r_3-1$.

where SUMROW(A,N) is the function

```
      REAL MATRIX FUNCTION SUMROW(A,N)
      REAL A(,)
      RLOG2R3 = ALOG(NR3)/LOG(2.)
      DO 1 I = 1,LOG2R3
        N1 = NP/(2 * I)
        A = A + SHSC(A,N1)
1     CONTINUE
      SUMROW = A
      RETURN
      END
```

and NR3 = r_3.

The calculation of the $z_2^{(2)}$ matrices follows similarly, except that here it is the columns which are multiplied by $W^{r_1 r_3 K_0 N_1}$ and the $N \times N$ matrices are summed in the E–W direction using SHWC in place of SHSC in the above routine.

To obtain the final z_2 matrices we must base reverse the $z_2^{(2)}$ matrix labels. If A(,) is the data-plane storing the r_3^2 $N \times N$ matrices and LMASK(,) is a logical mask set to .TRUE. down the block diagonal, then the following code will base reverse the matrix labels (equivalent to

interchanging the matrix row and column block coordinates N_0 and K_0)

```
1      A(.NOT. LMASK) = TRAN(LMASK)
2      DO 1 I = 0,NR3-1
3        A = SHEC(A, N)
4        A(LMASK) = TRAN(A)
5    1 CONTINUE
6      A = SHEC(A, N)
```

Line 1 takes the transpose of A using a library function TRAN leaving the block diagonal unchanged. This leaves the other $N \times N$ matrices in the correct positions but individually transposed. In lines 2 to 5 these matrices are transposed back. When the matrix A is transposed $N \times N$ matrices on the block diagonal are transposed without their block coordinates being changed, thus each of the $N \times N$ matrices are shifted in turn to the block diagonal (line 3) and then transposed (line 4). This code will work for any value of N such that r_3 is an integer.

(4) After the r_1 and r_2 point transforms have been done there still remains the problem of 'base' reversing the $N\,N \times N$ matrices ('base' reversal is defined by eq. (3.13)). The problem is not as simple as in the previous section as the matrices are spread over r_1 data planes and r_1 need not be equal to r_2. There does not seem to be an elegant way of doing this base reversal as there was in the previous subsection. The simplest method is to shift each of the $N \times N$ matrices individually into the correct position. Although the code to do this is tedious to write it should run quickly,data shift operations being significantly faster than arithmetic operations. The easiest method is to write a subroutine for each possibility and to call the appropriate subroutine at run time, as the number of possibilities is quite limited.

Using the code given above it is possible to write one program to calculate 3-d FFTs for any values of N and N_p such that the ratio N_p/N is a power of two. The calculation of the inverse transforms is done using the method described by eq. (2.14).

Acknowledgment

The authors would like to thank Mr. K. Refson for useful suggestions and discussions. The authors would also like to thank the SERC for their support of the two ICL DAPs at Edinburgh. One of the authors (AB) would also like to acknowledge the financial support of an SERC Information Technology grant.

References

[1] E.O. Brigham, *The Fast Fourier Transform* (Prentice-Hall, Englewood Cliffs, NJ, 1974).
[2] S.T. Davies, The implementation of the FFT on the DAP, in: D.J. Paddon, ed., *Supercomputers and Parallel Computation* (Clarendon Press, Oxford, 1984) 195–207.
[3] R.W. Hockney and J.W. Eastwood, *Computer Simulation Using Particles* (McGraw-Hill, New York, 1981).
[4] International Computers Ltd., DAP: Introduction to FORTRAN programming, Technical Publication 6755, Software Distribution Centre, Putney, London (1980).
[5] International Computers Ltd., DAP: FORTRAN language, Technical Publications 6918, Software Distribution Centre, Putney, London (1980).
[6] P.N. Swarztrauber, FFT algorithms for vector computers, *Parallel Comput.* 1 (1984) 45–63.