LENDING COPY

DARESBURY LABORATORY

INFORMATION QUARTERLY for MD & MC SIMULATIONS

An Informal Newsletter associated with Collaborative Computational Project No. 5 on Molecular Dynamics and Monte Carlo Simulations of Macroscopic systems.

Number 10 September 1983

Contents

Editor: Dr. William Smith

Science & Engineering Research Council, Daresbury Laboratory, Daresbury, Warrington WA4 4AD, England

Deputy Editor: Dr. David Heyes

Royal Holloway College, University of London, Egham TW20 OEX England

Editorial

Our contributors this month are old friends who have written many stimulating articles for past issues of the newsletter. Hopefully, they have become 'household names' in laboratories throughout the world for their efforts, as they thoroughly deserve. Your humble editor offers them his sincere thanks on behalf of our readers.

Instant fame and gratitude is also available to any of our readers who may be moved to follow the example of our regular contributors and express their ideas and comments through these pages.

Contributors:

D. Adams	Department of Chemistry, The University of Southampton, Southampton, SO9 5NH.
D. Fincham	D.A.P. Support Unit, The Computer Centre, Queen Mary College, Mile End Road, London, E1 4NS.
D.M. Heyes	Department of Chemistry, Royal Holloway College, Egham, Surrey, TW20 OEX.
N. Quirke	School of Chemical Engineering, Cornell University, Ithaca, New York, U.S.A.
W. Smith	T.C.S. Division, S.E.R.C., Daresbury Laboratory, Warrington, WA4 4AD.
M.L. Wolf & C.R.A. Catlow	Department of Chemistry, University of London, 20 Gordon Street, London, WC1H OAJ.

C.C.P.5 MEETING ON PHASE TRANSITIONS

19th - 20th December 1983
SOUTHAMPTON UNIVERSITY

Invited Lecturers

Professor G.R. Luckhurst Southampton University	Simulation of Phase Transitions in Liquid Crystals.
Professor D. Landau University of Georgia, U.S.A.	Finite Size Effects In Monte Carlo Simulations of Phase Transitions.
Dr M.L. Klein National Research Council of Canada, Ottawa, Canada	Phase Transitions using new methods in Molecular Dynamics.
Dr G.S. Pawley Edinburgh University	Molecular Dynamics of Plastic Crystalline Phases in SF_6.

Contributed Papers of ten to twenty minutes duration are now invited. Authors should send titles and abstracts to either Dr. D. Adams or Dr D.J. Tildesley, Department of Chemistry, The University of Southampton, Southampton SO95NH, United Kingdom by 31st October so that a preliminary programme may be drawn up.

Accommodation and meals will cost £32 per person and cheques should be made payable to

"C.C.P.5. Southampton Meeting"

and sent to Dr D. Adams. A further circular will be sent when a Draft Programme has been made.

General News

1. The most important news this month concerns the renewal of CCP5 for a further three years commencing in October 1983. This news is very welcome indeed following a somewhat lengthy application to the S.E.R.C. There are, however, to be some changes in the structure of the renewed Project, which will have a significant impact and briefly, they are as follows.

   Firstly, the scope of the Project is to be enlarged from its current area of molecular dynamics and Monte Carlo simulations to include the computer simulation of lattices by 'energy minimization' techniques. It is understood that such simulation methods have a strong following in the U.K. and a fairly comprehensive suite of related computer programs exists, for this purpose, at Daresbury Laboratory. In future therefore, the Project is expected to support this interest through the newsletter, conferences etc. and with the usual resources. The Project henceforth will be known by the title "The Computer Simulation of Condensed Phases".

   Secondly, the Project will not support a postdoctoral research associate as it has done in the past. (This post is currently held by Dr. D.M. Heyes). Instead the services of one additional member of the Daresbury T.C.S. Division will be made available. It is understood that this person will be Dr. Maurice Leslie, who has appropriate expertise in the area of lattice simulations.

2. Dr. David Heyes will cease to be CCP5 Secretary on 1st October 1983. He would like to thank the people, too numerous to mention, who have contributed to the activities of CCP5. It is through their efforts that the newsletter, program library and conferences have been such a success.

   It is appropriate here for the editor of the CCP5 newsletter to thank Dr. Heyes for his very worthy efforts in supporting CCP5 in all its activities. The success of CCP5, in no small way, owes much to him. It is a great pleasure to thank him for his participation.

Dr. Heyes will continue in the field of computer simulation at Royal Holloway College, where he will remain for a further five years on a Royal Society Fellowship.

3. The following announcement comes from Dr. M. Leslie from Daresbury Laboratory:-

A number of solid state simulation programs are available for use at Daresbury. These were written primarily to deal with ionic materials, although some of the programs can deal with non-ionic substances. Ionic materials may be treated either using the rigid ion model or the shell model. Various options are available for the short range repulsive interaction between ions. Programs are available at Daresbury for the following:-

(i) Calculation of the properties of perfect lattices - lattice energy, elastic and dielectric constants.
(ii) Relaxation of perfect the lattices to an equilibrium structure.
(iii) Calculation of phonon dispersion curves.
(iv) Calculation of defect energies.

Further details of these programs and of any restrictions on their use may be obtained from Dr. M. Leslie, Daresbury Laboratory.

4. The next CCP5 meeting on the subject of 'Phase Transitions' is scheduled to take place in Southampton on 19/20th December 1983. The invited speakers include, G.R. Luckhurst (Southampton), D. Landau (Georgia, U.S.A.), M.L. Klein (Ottawa, Canada), and G.S. Pawley (Edinburgh). The cost of accommodation and meals will be £32 per person. Authors wishing to contribute papers (of ten to twenty minutes duration) to the meeting should send titles and abstracts to either Dr. D. Adams or Dr. D.J. Tildesley, Department of Chemistry, the University of Southampton, Southampton, SO9 5NH, U.K. before 31st October 1983.

5. The University of Manchester Regional Computer Centre (UMRCC) began its acceptance trials of the CYBER 205 in June 1983. Unfortunately, while the hardware performed well, a substantial number of deficiencies were found in the FORTRAN 77 compiler (called FORTRAN 200). Since then this

problem has compounded and now the release of the compiler has been delayed until the end of the year. This means that the acceptance test for the CYBER 205 will be delayed until January 1984. Further problems include the delay in the development of the SCOPE 2 Remote Host Facility software, which links the Amdahl to the 7600's and the ICL 1900's to the 205. This software will now be tested in November. The Amdahl 470/V7A initial acceptance test is delayed until early September. The MASSTOR mass storage system appears to be functioning. Apparently its robot selector is a sight to behold!

6. The general news from the Rutherford and Appleton Laboratory is that the newly installed ICL Atlas 10 is performing well and on last report was accounting for 60% of batch CPU hours available to users. It has also been reported that the handover of the Atlas was re-scheduled for 8th August, as opposed to 1st September as mentioned previously.

7. The development of the University of London Computing Centre (ULCC) Amdahl/Cray 1S configuration is progressing. The Amdahl 470 V/8 is required to support 120 concurrent terminal sessions, batch processing, service the automatic filestore and up to 100 remote job entry work-stations as well as front-ending the Cray 1S. The terminal access to the Amdahl is intended primarily for editing and job submission to the batch system and substantial interactive use is not anticipated. The terminal system is the IBM TSO system, with a subset of the Cambridge Phoenix system. TSO is as supplied, without enhancements by ULCC, though users will be able to introduce enhancements of their own via the TSO command package. Current terminal access is restricted to about 50 users of which 20 may be linked concurrently. The number of concurrent users will rise gradually to about 100 by the end of the year. All users of the Amdahl are allocated 25 tracks (½ M bytes) of disc space.

The CDC Cyber 72 computer was withdrawn from service on 15th August.

8. Two additional programs have been donated to the CCP5 Program Library. The first of these is the program ADMIXT by W. Smith which simulates Lennard-Jones particle mixtures. The second is SURF by D.M. Heyes, which simulates model alkalai kalide laminas.Documentation for each of

these is also available. These programs and others in the CCP5 Program Library are available free of charge to academic establishments. A list of the programs available is provided overleaf.

Anyone wishing to donate to or receive programs from the CCP5 Program Library, should contact the Librarian, Dr. W. Smith, SERC, Daresbury Laboratory, Daresbury, Warrington, WA4 4AD, U.K.

List of Programs in the CCP5 Program Library.

MDATOM by S. M. Thompson.

M.D. simulation of atomic fluids. Uses 12/6 Lennard - Jones potential function and fifth order Gear integration algorithm. Calculates system average configuration energy, kinetic energy, virial, mean square force and the associated R.M.S. deviations and also system pressure, temperature, constant volume specific heat, mean square displacement, quantum corrections and radial distribution function.

HMDIAT by S. M. Thompson.

M.D. simulation of diatomic molecule fluids. Uses 12/6 Lennard - Jones site - site potential functions and a fifth order Gear algorithm for centre - of - mass motion. Angular motion is calculated by fourth order Gear algorithm with quaternion orientation parameters. Calculates system average configuration energy, kinetic energy, virial, mean square force, mean square torque and the associated R.M.S. deviations and also system pressure, temperature, constant volume specific heat, mean square displacement and quantum corrections.

MDLIN by S. M. Thompson.

M.D. simulation of linear molecule fluids. Uses 12/6 Lennard - Jones site - site potential functions and a fifth order Gear algorithm for centre - of - mass motion. Angular motion is calculated by fourth order Gear algorithm with quaternion orientation parameters. List of calculated properties is the same as HMDIAT.

MDLINQ by S. M. Thompson.

M.D. simulation of linear molecule fluids. Uses 12/6 Lennard - Jones site - site potential functions plus a point electrostatic quadrupole. Uses a fifth order Gear algorithm for centre - of - mass motion. Angular motion is calculated by fourth order Gear algorithm with quaternion orientation parameters. List of calculated properties is the same as HMDIAT.

MDTETRA by S. M. Thompson.

M.D. simulation of tetrahedral molecule fluids. Uses 12/6 Lennard - Jones site - site potential functions and a fifth order Gear algorithm for centre - of - mass motion. Angular motion is calculated by fourth order Gear algorithm with quaternion orientation parameters. List of calculated properties is the same as HMDIAT.

MDPOLY by S. M. Thompson.

M.D. simulation of polyatomic molecule fluids. Uses 12/6 Lennard - Jones site - site potential functions and a fifth order Gear algorithm for centre - of - mass motion. Angular motion is calculated by fourth order Gear algorithm with quaternion orientation parameters. List of calculated properties is the same as HMDIAT.

ADMIXT by W. Smith.

M.D. simulation of monatomic molecule mixtures. Uses 12/6 Lennard - Jones atom - atom potential functions and a Verlet leapfrog algorithm for centre - of - mass motion. Calculates system average configuration energy, kinetic energy and virial and associated R.M.S. deviations and also pressure, temperature, mean square displacements and radial distribution functions.

MDMIXT by W. Smith.

M.D. simulation of polyatomic molecule mixtures. Uses 12/6 Lennard - Jones site - site potential functions and a Verlet leapfrog algorithm for centre - of - mass motion. Angular motion is calculated by the Fincham leapfrog algorithm using quaternion orientation parameters. Calculates system average configuration energy, kinetic energy and virial and associated R.M.S. deviations and also pressure and temperature.

MDMULP by W. Smith.

M.D. simulation of polyatomic molecule mixtures. Uses 12/6 Lennard - Jones site - site potential functions and point electrostatic multipoles (charge, dipole and quadrupole). Long range electrostatic effects are calculated using the Ewald summation method. Uses a Verlet leapfrog algorithm for centre - of - mass motion. Angular motion is calculated by the Fincham leapfrog algorithm using quaternion orientation parameters. Calculates system average configuration energy, kinetic energy and virial and associated R.M.S. deviations and also pressure and temperature.

MDMPOL by W. Smith & D. Fincham.

M.D. simulation of polyatomic molecule mixtures. Uses 12/6 Lennard - Jones site - site potential functions and fractional charges to represent electrostatic multipoles. Long range electrostatic effects are calculated using the Ewald summation method. Uses a Verlet leapfrog algorithm for centre - of - mass motion. Angular motion is calculated by the Fincham leapfrog algorithm using quaternion orientation parameters. Calculates system average configuration energy, kinetic energy and virial and associated R.M.S. deviations and also pressure and temperature.

DENCOR by W. Smith.

Calculation of density correlation functions. Processes atomic M.D. data to produce the Fourier transform of the particle density, the intermediate scattering functions and the dynamic structure factors.

CURDEN by W. Smith.

Calculation of current density correlation functions. Processes atomic M.D. data to produce the Fourier transform of the current density, the current density correlation functions and their temporal Fourier transforms.

HLJ1 by D. M. Heyes.

M.D. simulation of atomic fluids. Uses 12/6 Lennard - Jones site - site potential function and a Verlet leapfrog algorithm for centre - of - mass motion. Calculates system average configuration energy and kinetic energy and associated R.M.S. deviations and also pressure, temperature, mean square displacements and radial distribution function.

HLJ2 by D. M. Heyes.

M.D. simulation of atomic fluids. Uses 12/6 Lennard - Jones site - site potential function and a Verlet leapfrog algorithm for centre - of - mass motion. Calculates system average configuration energy and kinetic energy and associated R.M.S. deviations and also pressure, temperature, mean square displacements, radial distribution function and velocity autocorrelation function.

HLJ3 by D. M. Heyes.

M.D. simulation of atomic fluids. Uses 12/6 Lennard - Jones site - site potential function and a Verlet leapfrog algorithm for centre - of - mass motion. The link - cell method is employed to enable large simulations. Calculates system average configuration energy and kinetic energy and associated R.M.S. deviations and also pressure, temperature, mean square displacements and radial distribution function.

HLJ4 by D. M. Heyes.

M.D. simulation of atomic fluids. Uses 12/6 Lennard - Jones site - site potential function and a Verlet leapfrog algorithm for centre - of - mass motion. The algorithm allows either the temperature or the pressure to be constrained. Calculates system average configuration energy and kinetic energy and associated R.M.S. deviations and also pressure, temperature, mean square displacements and radial distribution function.

HLJ5 by D. M. Heyes.

M.D. simulation of atomic fluids. Uses 12/6 Lennard - Jones site - site shifted potential function and a Verlet leapfrog algorithm for centre - of - mass motion. This method removes the discontinuities at the potential cutoff radius. Calculates system average configuration energy and kinetic energy and associated R.M.S. deviations and also pressure, temperature, mean square displacements and radial distribution function.

HLJ6 by D. M. Heyes.

M.D. simulation of atomic fluids. Uses 12/6 Lennard - Jones site - site shifted potential function and the Toxvaerd algorithm for centre - of - mass motion. This algorithm is more accurate than the Verlet algorithm. Calculates system average configuration energy and kinetic energy and associated R.M.S. deviations and also pressure, temperature, mean square displacements and radial distribution function.

MCRPM by D. M. Heyes.

M.C. simulation of electrolytes. Monte Carlo program using restricted primitive model of an electrolyte. The potential is regarded as infinite for $r < d$ and Coulombic for $r > d$. The properties calculated are the average configuration energy and its R.M.S. deviation, the pair radial distribution function and the melting factor.

SURF by D. M. Heyes.

M.D. simulation of model alkalai halide lamina. Molecular dynamics simulation for ionic laminae using the Tosi-Fumi / Born-Mayer-Huggins potential and the Evjen method for evaluating the lattice sums. The integration algorithm used is the Verlet method. The program calculates the system potential and kinetic energies, the pressure and the final averages and R.M.S. fluctuations. The program also calculates density profiles such as number density, temperature, energy and pressure.

HSTOCH by W. F. van Gunsteren & D. M. Heyes.

S.D. or M.D. simulation of molecules in vacuo or in a rectangular cell with solvent or lattice atoms (i.e. Langevin or Brownian dynamics of large molecules).

MDATOM by D. Fincham.

M.D. simulation of atomic fluids. Uses 12/6 Lennard - Jones potential function and Verlet leapfrog integration algorithm. Calculates system average configuration energy, kinetic energy, virial and the associated R.M.S. deviations and also system pressure, temperature, mean square displacement and radial distribution function.

MDDIAT by D. Fincham.

M.D. simulation of diatomic molecule fluids. Uses 12/6 Lennard - Jones site - site potential functions and the Verlet leapfrog algorithm for centre - of - mass motion. Angular motion is calculated using the constraint algorithm. Calculates system average configuration energy, kinetic energy, virial and the associated R.M.S. deviations and also system pressure, temperature and mean square displacement.

MDDIATQ by D. Fincham.

M.D. simulation of diatomic fluids. Uses 12/6 Lennard - Jones site - site potential functions and a point quadrupole electrostatic term. Employs the Verlet leapfrog algorithm for centre - of - mass motion. Angular motion is calculated using the constraint algorithm. Calculates system average configuration energy, kinetic energy, virial and the associated R.M.S. deviations and also system pressure, temperature and mean square displacement.

MDIONS by D. Fincham & N. Anastasiou.

M.D. simulation of electrolytes. Uses exp/6/8 potential function and the Coulomb electrostatic potential. Long range interactions are calculated using the Ewald summation method. Uses the Verlet leapfrog algorithm for particle motion. Calculates system average configuration energy, kinetic energy, virial and the associated R.M.S. deviations and also system pressure, temperature, radial distribution functions, static structure factors and mean square displacements.

MDMANY by D. Fincham & W. Smith.

M.D. simulation of polyatomic molecules. Uses 12/6 Lennard - Jones site - site potential functions and fractional charges to represent electrostatic multipoles. Long range electrostatic effects are calculated using the Ewald summation method. Uses a Verlet leapfrog algorithm for centre - of - mass motion. Angular motion is calculated by the Fincham leapfrog algorithm using quaternion orientation parameters. Calculates system average configuration energy, kinetic energy and virial and associated R.M.S. deviations and also pressure and temperature. FORTRAN 77 standard program.

CARLOS by B. Jonsson & S. Romano.

M.C. simulation of a polyatomic solute molecule in an aqueous cluster. (i.e. a molecule surrounded by water molecules). The water - water potential is calculated using an analytical fit to an ab initio potential energy surface due to Matsuoka et al. The solute-solvent potential is optional. The program provides an energy and coordinate 'history' of the M.C. simulation. An analysis program CARLAN for processing the data produced by CARLOS is also available.

MCN by N. Corbin.

M.C. simulation of atomic fluids. Standard (Metropolis) Monte Carlo program for atomic fluids.

SCN by N. Corbin.

M.C. simulation of atomic fluids. Standard (Rossky,Friedman and Doll) Monte Carlo program for atomic fluids.

SMF by N. Corbin.

M.C. simulation of atomic fluids. Standard (path integral method) Monte Carlo program for atomic fluids.

PARALLEL PROGRAMMING FOR LATTICE PROBLEMS ON THE DAP

D. Fincham & N. Quirke.

The aim of this article is to give a simple but informative example of the use of DAP Fortran when solving lattice problems. We have chosen to discuss the Monte Carlo simulation of the Solid on Solid model (S.O.S). This model has been discussed in a previous Newsletter article (1) which included a Fortran program. We find the comparison between the present DAP version and the earlier Fortran version particularly illustrative of the benefits of DAP Fortran.

The S.O.S. model is a restricted Ising model of the crystal-motherphase interface. Most of the present day understanding of crystal growth is based upon results obtained from studying the S.O.S. model in either its equilibrium or kinetic forms (2). The model exhibits a phase transition, the Roughening Transition, at a temperature T_r, above which the free energy of surface features is zero. It has been demonstrated theoretically (2) that the free energy of an infinitely long step in the surface of the S.O.S. model should go to zero when $T \to T_r$ as

$$A \quad \exp\left(-\alpha / |T - T_r|^{\frac{1}{2}} \right)$$

This has recently been confirmed (3) by a direct calculation of the step free energy in a Monte Carlo simulation of the S.O.S. model.

The roughening temperature marks the transition between a sharp, well defined surface with, in the kinetic model, a nucleatian barrier to growth, and a delocalized, rough surface with divergent surface width. Since the roughening temperature is a property of the surface alone it can be studied with the equilibrium form of the model and it is this version we take as our example below. We first give some details of the S.O.S. model and then turn to the DAP programming techniques.

In the S.O.S. model space is completely filled by a lattice solid or fluid. The model is specified by the positions of the sites and their state. Using a Cartesian frame, the X,Y directions are equivalent but the Z axis stretches from completely solid at $-\infty$ to completely fluid at $+\infty$. The interface is conveniently located initially at Z = 0. So far we have done nothing more than define a three dimensional Ising model with special boundary conditions. The new feature of the S.O.S. model is the S.O.S. restriction preventing inclusions of fluid inside the solid. We insist that a solid site can only have another solid site below it. Now the state of the system can be completely specified by giving the X,Y co-ordinates of a column of sites and the height (h) of the highest solid site in each column. The Hamiltonian can be written:

$$H = \epsilon \sum_{\langle ij \rangle} | h_i - h_j |^p \quad \text{with} \quad \epsilon > 0 \text{ and we take } p = 1$$

The sum is over all columns i and the four nearest neighbour columns j of i. Using Monte Carlo simulation we shall calculate the surface energy and specific heat of this model as a function of temperature on a 64 x 64 array of columns. Within the limits imposed by the finite size of the simulation, the vanishing of the step free energy and its derivatives can be observed using the methods outlined in reference (3). The DAP is particularly suited to this sort of lattice problem, giving several times the speed of computation of a CDC 7600.

There are four basic features of the DAP program which are different from those of a conventional serial program.

A) Use of parallel updating with chessboard ordering. In the conventional program an individual site is chosen at random and updated. On the DAP the xy plane is mapped onto the 64 x 64 plane of processing elements which work in parallel. Since the energy of a site depends on the heights of its nearest neighbours it would not be correct to update all the sites simultaneously. Instead, we divide the plane into a chessboard pattern, and update first all the "black" sites (which have "white" nearest neighbours), and then all the "white" sites.

B) Use of short wordlength integer arithmetic. As the DAP processing elements are single-bit processors, all arithmetic is provided in software and short wordlength fixed point arithmetic runs very much more quickly than floating-point arithmetic. Since the column heights and energies are integer quantities we can take advantage of this if we are prepared to propose a maximum height for columns and guard against overflows.

C) Use of built-in shift functions. The heart of the calculation is the comparison of column heights with those of their nearest neighbours. In looking for neighbours we wish as usual in simulation to incorporate cyclic boundary conditions. These tasks are easily achieved in DAP Fortran using shift functions which are part of the language.

D) Use of logical masks. It is possible in DAP Fortran to use logical matrices in place of subscripts to select positions in the DAP plane where processing is to be carried out. This makes it easy to write code involving conditional statements. Several examples will be found below.

We now go on to consider the program in detail. The reader may find it helpful to refer to the listing and compare it with the Fortran program in (1).

A DAP program has two parts, a Host section written in Fortran and a DAP section written in DAP Fortran. Control is transferred from host to DAP by calling a special subroutine called an entry subroutine and values are passed through COMMON blocks. In the example, the host program reads in the number of iterations per column, NITS, and the reduced temperature TSTAR = $k_B T/\varepsilon$. After the DAP processing it prints the surface energy per site SURFEN, the specific heat SURFCV, and the fraction of accepted iterations YES. However, the host and the DAP recognise different data formats and so special conversion subroutines we called on entry to the DAP and immediately before return to the host.

The DAP subroutine declares a number of matrices, for example the matrix of column heights HEIGHTS (,). This represents a 64 x 64 matrix. The maximum height of a column is restricted by the declaration INTEGER *2 to be a 16 bit quantity. The initial section of the program involves:

a) Initialisation of the random number generator. This is one of the DAP subroutine library routines and returns a 64 x 64 plane of random numbers. It is of the exclusive-or (XOR) or shift-register type, and believed to be the highest-quality generator available on any computer (4);

b) setting up the logical mask BLACK which is .TRUE. in all the "black" positions of the chessboard pattern. This uses built-in functions and is best understood by reference to the diagram below, drawn for a 4 x 4 DAP.

Figure 1

```
      F  T  F  T                  F  F  F  F

      F  T  F  T                  T  T  T  T

      F  T  F  T                  F  F  F  F

      F  T  F  T                  T  T  T  T

        ALTC(1)                     ALTR(1)


                    .LEQ.


                 T  F  T  F

                 F  T  F  T

                 T  F  T  F

                 F  T  F  T
```

c) initialisation of accumulators to zero and, lastly,

d) the setting of all column heights and energies to zero with single statements. This sets the S.O.S. surface at Z = 0.

The iteration loop begins with the selection of two matrices of random numbers. The REAL matrix RAND will be used later. The LOGICAL matrix UP is a random set of .TRUE. and .FALSE. values. It is used as a mask to select a set of trial column heights THEIGHT which are either increased or decreased by one. Then the loop JB = 1,2 runs over "black" and "white" columns successively. The trial energies are calculated by comparing the trial heights of the columns with the <u>actual</u> heights of their nearest neighbours which are in the matrices SOUTH, NORTH, WEST and EAST. The setting up of these matrices is explained below. A matrix DELEN of energy changes is calculated, and a logical expression is used to apply the Metropolis algorithms (1) to produce the LOGICAL matrix ACCEPT. This statement illustrates very well how logical quantities are used in DAP Fortran and produce very clear code. Note that the expression EXP(-DELEN*BETA).GT.RAND automatically includes the case when DELEN is less than zero since RAND is uniformly distributed between 0 and 1.

So far processing has been carried out for every column but the results will be used only in the case of the columns labelled "black". Hence we apply the logical expression ACCEPT.AND.BLACK as a mask to control the updating of the matrices HEIGHT. The logical matrix ACCEPTED accumulates ACCEPT over the "black" and "white" parts of the iterations. The next stage is to calculate the energies of the columns from the difference between the heights of neighbouring columns. These are obtained using the built in cyclic shift functions SHNC etc. Neighbours exist to the south, north, west and east of any column. For example to calculate the bond energy between each column and its southern neighbour the matrix SOUTH is defined by shifting all heights one place to the <u>north</u>. This brings the southern neighbour of each site into coincidence with it, so that the south-north bond energies, which are just the absolute values of the difference in heights along these bonds, can be computed by the statement ABS (HEIGHT-SOUTH). This is illustrated below:

HEIGHT				SOUTH				ABS (HEIGHT-SOUTH)			
3	0	-3	2	-1	2	4	-2	4	2	7	4
-1	2	4	-2	-4	0	1	-2	3	2	3	0
-4	0	1	-2	2	1	1	-3	6	1	0	1
2	1	1	-3	3	0	-3	2	1	1	4	5

Note that since the shifts are cyclic the periodic boundary conditions are automatically obeyed. In the Fortran program (1) the handling of the periodic boundaries is much more complicated. This whole procedure is now repeated for the other set of columns using the statement BLACK = .NOT. BLACK before the next iteration of the JB loop.

At the end of each iteration, when the "black" and "white" columns have been updated, the matrices ENERGY and ACCEPTED are summed.

Finally, after initial equilibration averages over the complete run are taken before transferring variables to the host program. The program performs more than 250 complete iterations per second, even though because of the black-white ordering only half the processors are active at one time. At the cost of some complication it is possible to perform the simulation on a 64 x 128 grid so that all processors are used in updating, first the "black" columns and then the "white" columns.

```
      PROGRAM ROUGH
      COMMON/PARAMS/ NITS,NEQUIL,TSTAR
      COMMON/RESULTS/ SURFEN,SURFCV,YES
      READ*,NITS,NEQUIL,TSTAR
      CALL DAP
      PRINT*,SURFEN,SURFCV,YES
      STOP
      END


      ENTRY SUBROUTINE DAP
      COMMON/PARAMS/ NITS,NEQUIL,TSTAR
      COMMON/RESULTS/ SURFEN,SURFCV,YES
      INTEGER*2 HEIGHT(,),THEIGHT(,),ENERGY(,),TENERGY(,)
      INTEGER*2 DELEN(,)
      INTEGER*2 SOUTH(,),NORTH(,),WEST(,),EAST(,)
      REAL*4 RAND(,)
      LOGICAL BLACK(,),UP(,),ACCEPT(,),ACCEPTED(,)
      EXTERNAL REAL MATRIX FUNCTION G05XORREAL4
      EXTERNAL LOGICAL MATRIX FUNCTION G05XORPLANE
C convert input parameters
      CALL CONVFS4(NITS,3)
      BETA=1.0/TSTAR
C initialise random number generator
      CALL G05XORBEGIN
C set up black mask
      BLACK=ALTC(1).LEQ.ALTR(1)
C zero accumulators
      YES=0.0
      SURFEN=0.0
      SURFEN2=0.0
C set initial heights and energies
      HEIGHT=0
      SOUTH=SHNC(HEIGHT)
      NORTH=SHSC(HEIGHT)
      WEST =SHEC(HEIGHT)
      EAST =SHWC(HEIGHT)
      ENERGY=0
C loop over iterations
      DO 100 JIT=1,NITS
      RAND=G05XORREAL4(0.0)
      UP=G05XORPLANE(0)
      IF(ANY(ABS(HEIGHT).GT.32766)) ERROR 1
C trial heights
      THEIGHT(UP)=HEIGHT+1
      THEIGHT(.NOT.UP)=HEIGHT-1
C loop over black and white columns
      ACCEPTED=.FALSE.
      DO 101 JB=1,2
C calculate trial energies of the sites
      TENERGY=ABS(THEIGHT-SOUTH)
     1       +ABS(THEIGHT-NORTH)
     2       +ABS(THEIGHT-WEST )
     3       +ABS(THEIGHT-EAST )
```

pto.

```
C apply Metropolis algorithm
      DELEN=TENERGY-ENERGY
      ACCEPT=EXP(-DELEN*BETA).GE.RAND
      ACCEPT=ACCEPT.AND.BLACK
      HEIGHT(ACCEPT)=THEIGHT
      ACCEPTED=ACCEPTED.OR.ACCEPT
C calculate energies using new heights
      SOUTH=SHNC(HEIGHT)
      NORTH=SHSC(HEIGHT)
      WEST =SHEC(HEIGHT)
      EAST =SHWC(HEIGHT)
      ENERGY =ABS(HEIGHT-SOUTH)
     1       +ABS(HEIGHT-NORTH)
     2       +ABS(HEIGHT-WEST )
     3       +ABS(HEIGHT-EAST )
C interchange black and white
      BLACK=.NOT.BLACK
  101 CONTINUE
C form sums over columns for this iteration
      IF(JIT.LE.NEQUIL) GO TO 100
         YES=YES+SUM(ACCEPTED)
         TERMSUM=0.5*SUM(ENERGY)
         SURFEN =SURFEN +TERMSUM
         SURFEN2=SURFEN2+TERMSUM**2
  100 CONTINUE
C perform final averaging and conversions
      NCOUNT=NITS-NEQUIL
      SURFEN=SURFEN/NCOUNT
      SURFEN2=SURFEN2/NCOUNT
      SURFCV=(SURFEN2-SURFEN**2)*BETA**2
      SURFEN=SURFEN/4096.0
      SURFCV=SURFCV/4096.0
      YES=YES/NCOUNT/4096.
      CALL CONVSF4(SURFEN,3)
      RETURN
      END
```

References

1. N. Quirke, CCP5 Newsletter No.6, September 1982.

2. J.D. Weeks, G.H. Gilmer, Advances in Chemical Physics, 40 157, (1979).

3. G. Jacucci, N. Quirke, Physics Letters A (accepted for publication).

4. K. Smith, S. Pawley (in preparation).

HARD-SPHERE MOLECULAR DYNAMICS

D.M. Heyes

There is an ever-present interest in the hard-sphere system. It is perhaps the most simple model of the liquid and solid states and therefore clearly distinguishes between the effects on condensed phase properties due to the nature of the pair potential and the many-body dynamics. The many-body nature of molecular dynamics can be solved exactly on a digital computer using the method of Molecular Dynamics, MD.

Many computer simulators will be familiar with the method of MD as applied to molecules interacting via continuous potentials such as the Lennard-Jones form. The system travels through phase space in a series of time steps of constant duration. In Hard-Sphere MD, which was devised by Alder and Wainwright [1], the concept of the time step does not exist. Instead a sequence of collisions is undergone in strict chronological order. The system is aged by going directly from one hard-sphere, HS, collision (i.e., the contact and rebounding of two spheres) to the next. All other non-colliding HS are also moved in accordance with their velocities and the times between the aforementioned isolated collisions. The technique follows a sequence of binary elastic collisions. The particles move in straight lines with constant velocity between collisions. As this is a procedure which has not been described in detail before, the FORTRAN code of the essential parts of a hard-sphere program are given below.

At the beginning of each HS simulation it is necessary to go through the particle pairs to establish, for a given set of starting positions and velocities, what are the times to the first collision for each hard-sphere. Let i and j be the indices of two particles which collide at some time t'=t from the present configuration, i.e., at t'=0. The position and velocity of i is $\underline{r}_i$ and $\underline{v}_i$, respectively. Also let us define the following relative positions and velocities,

$$\underline{r}_{ij} = \underline{r}_i - \underline{r}_j, \quad (1)$$

$$\underline{v}_{ij} = \underline{v}_i - \underline{v}_j. \quad (2)$$

The MD cell has sides along the x,y, and z directions. Now consider the x,y and z position and velocity components,

$$r_{xij}(t) = r_{xij}(0) + v_{xij}(0)t, \quad (3)$$

$$r_{yij}(t) = r_{yij}(0) + v_{yij}(0)t, \quad (4)$$

$$r_{zij}(t) = r_{zij}(0) + v_{zij}(0)t. \quad (5)$$

If the diameter of each HS is σ then,

$$\sigma^2 = (r_{xij}(0) + v_{xij}(0)t)^2 + (r_{yij}(0) + v_{yij}(0)t)^2 + (r_{zij}(0) + v_{zij}(0)t)^2. \quad (6)$$

In concise vector notation then,

$$(\underline{r}_{ij} + \underline{v}_{ij}t)^2 = \sigma^2. \quad (7)$$

We need to obtain t. This involves solving the following quadratic equation for t,

$$At^2 + 2Bt + C = 0, \qquad (8)$$

where,

$$A = \mathbf{v}_{ij}^2 = v_{xij}^2 + v_{yij}^2 + v_{zij}^2, \qquad (9)$$

$$B = \mathbf{r}_{ij}\cdot\mathbf{v}_{ij} = r_{xij}v_{xij} + r_{yij}v_{yij} + r_{zij}v_{zij} \qquad (10)$$

$$C = \mathbf{r}_{ij}^2 - \sigma^2 = r_{xij}^2 + r_{yij}^2 + r_{zij}^2 - \sigma^2. \qquad (11)$$

The solution is,

$$t = - B +/- (B^2 - AC)^{1/2}/A. \qquad (12)$$

Let there be N hard-spheres at positions RX(I), RY(I) and RZ(I), where I is the generalised HS index. The velocity components are VX(I), VY(I) and VZ(I). Each HS collides with its first HS, index NJ(I), in a time TC(I). The position coordinates are in units of the MD cell sidelength. Also similarly SG2 = σ^2 is in units of the box sidelength squared. Then a typical FORTRAN code for this is:

```
      DO 1 I=1,N
      TC(I)=1.0E10
1     CONTINUE
      N=N-1
      DO 10 I=1,N1
      I1=I+1
      RXI=RX(I)
      RYI=RY(I)
      RZI=RZ(I)
      DO 20 J=I1,N
      X=RXI-RX(J)
      Y=RYI-RY(J)
      Z=RZI-RZ(J)
      IF (X.GT.0.5) X=X-1.0
      IF (Y.GT.0.5) Y=Y-1.0
      IF (Z.GT.0.5) Z=Z-1.0
      IF (X.LT.-0.5) X=X+1.0
      IF (Y.LT.-0.5) Y=Y+1.0
      IF (Z.LT.-0.5) Z=Z+1.0
      RR=X*X+Y*Y+Z*Z
      U=VX(I)-VX(J)
      V=VY(I)-VY(J)
      W=VZ(I)-VZ(J)
      B=X*U+Y*V+Z*W
C     ELIMINATE THOSE HS PAIRS GOING AWAY FROM EACH OTHER
      IF (B.GE.0.0) GOTO 30
      A=U*U+V*V+W*W
      C=RR-SG2
      AC=A*C
      BB=B*B
C     ELIMINATE THOSE HS PAIRS WHICH DO NOT COLLIDE
      IF (AC.GE.BB) GOTO 40
      Q=SQRT(BB-AC)
C     TAKE THE LEAST POSITIVE OF THE ROOTS
      T=-(B+Q)/A
      IF (T.GT.TC(I)) GOTO 50
C     THE FIRST COLLISION FOR I
      TC(I)=T
      NJ(I)=J
50    CONTINUE
      IF (T.GT.TC(J)) GOTO 60
C     THE FIRST COLLISION FOR J
      TC(J)=T
      NJ(J)=I
60    CONTINUE
40    CONTINUE
30    CONTINUE
20    CONTINUE
10    CONTINUE
```

This was called the 'long cycle' by Alder and Wainwright [1]. Note that initially all TC(I) are set to very large values which should get reduced when the above two-particle loop is executed. Even if they do not change, the following collisions will almost certainly reduce them before they 'float to the top of the stack' of collision times.

Now isolate the first colliding pair out of the possibilites TC(I),

```
      TNEXT=1.0E8
      DO 100 I=1,N
      IF (TC(I).LT.TNEXT) K1=I
      IF (TC(I).LT.TNEXT) TNEXT=TC(I)
100   CONTINUE
C     FIND THE COLLIDING PARTNER INDEX
      K2=NJ(K1)
      NCOL=0
```

NCOL is the total number of collisions accumulated. Having determined the starting collision (between hard-spheres with indices I=K1 and K2) we are able to go through the <u>entire</u> simulation without recourse to another two-particle double loop. The main program now starts. We now enter an open-ended number of 'short cycles'.

```
5     CONTINUE
      DO 200 I=1,N
      TC(I)=TC(I)-TNEXT
      RX(I)=RX(I)+VX(I)*TNEXT
      RY(I)=RY(I)+VY(I)*TNEXT
      RZ(I)=RZ(I)+VZ(I)*TNEXT
      IF (RX(I).GE.1.0) RX(I)=RX(I)-1.0
      IF (RY(I).GE.1.0) RY(I)=RY(I)-1.0
      IF (RZ(I).GE.1.0) RZ(I)=RZ(I)-1.0
      IF (RX(I).LT.0.0) RX(I)=RX(I)+1.0
      IF (RY(I).LT.0.0) RY(I)=RY(I)+1.0
      IF (RZ(I).LT.0.0) RZ(I)=RZ(I)+1.0
200   CONTINUE
      NCOL=NCOL+1
```

Above we have reduced each particle's first collision time by the time to the first collision (between K1 and K2). The HS positions have been evolved and subjected to periodic boundary conditions. Note that each sidelength of the MD cell is the same. The program unit of distance is the MD cell's sidelength.

The velocity changes after contact of the HS are governed by conservation of linear and angular momentum. The relative velocity change on contact is,

$$-\mathbf{v}_{ij}(0)\cdot\mathbf{r}_{ij}(t)/\sigma$$

i.e., the projection of the relative velocity along the vector between the HS centres. Perpendicular velocity components do not change. The HS velocity change is simply the projection of this in the 'laboratory' frame,

$$(\mathbf{r}_{ij}/\sigma)(-\mathbf{v}_{ij}\cdot\mathbf{r}_{ij}/\sigma),$$

We now perform the collision for the hard-sphere pair (K1,K2),

```
      X=RX(K1)-RX(K2)
      Y=RY(K1)-RY(K2)
      Z=RZ(K1)-RZ(K2)
C     FIND THE NEAREST IMAGE
      IF (X.GT.0.5) X=X-1.0
      IF (Y.GT.0.5) Y=Y-1.0
      IF (Z.GT.0.5) Z=Z-1.0
      IF (X.LT.-0.5) X=X+1.0
      IF (Y.LT.-0.5) Y=Y+1.0
      IF (Z.LT.-0.5) Z=Z+1.0
      U=VX(K1)-VX(K2)
      V=VY(K1)-VY(K2)
      W=VZ(K1)-VZ(K2)
      B=(X*U+Y*V+Z*W)/SG2
      X=X*B
      Y=Y*B
      Z=Z*B
      VX(K1)=VX(K1)-X
      VY(K1)=VY(K1)-Y
      VZ(K1)=VZ(K1)-Z
      VX(K2)=VX(K2)+X
      VY(K2)=VY(K2)+Y
      VZ(K2)=VZ(K2)+Z
```

The smallest new TC(I), which has already been obtained, should give the next collision partners <u>unless</u>: (a) one of the HS which was involved in the previous collision (K1 and K2) is involved in the next collision or (b) K1 or K2 were the first collision partners for other HS. It now remains to recalculate the TC(I) for K1, K2, NJ(K1) and NJ(K2),

```
      DO 300 I=1,N
      NI(I)=0
      IF (I.EQ.K1) NI(I)=1
      IF (I.EQ.K2) NI(I)=1
      IF (NJ(I).EQ.K1) NI(I)=1
      IF (NJ(I).EQ.K2) NI(I)=1
      IF (NI(I).EQ.1) TC(I)=1.0E8
300   CONTINUE
C     RECALCULATE COLLISION PARTNERS
      DO 310 I=1,N
      IF (NI(I).EQ.0) GOTO 320
      RXI=RX(I)
      RYI=RY(I)
      RZI=RZ(I)
      DO 330 J=1,N
      IF (I.EQ.J) GOTO 340
      X=RXI-RX(J)
      Y=RYI-RY(J)
      Z=RZI-RZ(J)
      IF (X.GT.0.5) X=X-1.0
      IF (Y.GT.0.5) Y=Y-1.0
      IF (Z.GT.0.5) Z=Z-1.0
      IF (X.LT.-0.5) X=X+1.0
      IF (Y.LT.-0.5) Y=Y+1.0
      IF (Z.LT.-0.5) Z=Z+1.0
      RR=X*X+Y*Y+Z*Z
      U=VX(I)-VX(J)
      V=VY(I)-VY(J)
      W=VZ(I)-VZ(J)
      B=X*U+Y*V+Z*W
      IF (B.GE.0.0) GOTO 350
      A=U*U+V*V+W*W
      C=RR-SG2
      AC=A*C
      BB=B*B
      IF (AC.GE.BB) GOTO 360
      Q=SQRT(BB-AC)
      T=-(B+Q)/A
      IF (T.GT.TC(I)) GOTO 370
      TC(I)=T
      NJ(I)=J
370   CONTINUE
      IF (T.GT.TC(J)) GOTO 380
      TC(J)=T
      NJ(J)=I
380   CONTINUE
360   CONTINUE
350   CONTINUE
340   CONTINUE
330   CONTINUE
320   CONTINUE
310   CONTINUE
      IF (NCOL.LT.NCOLT) GOTO 5
```

Here NI(I) is a 'flag' array of dimension N, which is used to select the required hard-sphere pairs. Above NCOLT is the total number of collisions allowed in the run. These short cycles are gone through until NCOL equals NCOLT.

Acknowledgements

The author is indebted to Dr. L.V. Woodcock (University of Bradford) for teaching him hard-sphere Molecular Dynamics.

References

[1] B.J. Alder and T.E. Wainwright, J. Chem. Phys., 31, 459 (1959).

IONIC FILM MOLECULAR DYNAMICS

D.M. Heyes

In the June 1983 issue of the CCP5 Newsletter the theory and units for performing MD on Tosi-Fumi/Born-Mayer-Huggins alkali halide films were outlined [1,2]. In this note the FORTRAN code (optimised for the CRAY-1S computer) for such a scheme is presented.

Let there be N ions in the MD cell. Let the component positions along the three mutually perpendicular MD cell sides be RX(I), RY(I) and RZ(I), where I is a generalised ion index which can range from 1 to N. The first N/2 are for cations and the second N/2 are for anions. FX(I), FY(I), and FZ(I) are the force components. PT(I) contains the potential energy per ion. Q(I) contains the charge: +1.0 for the first N/2 and -1.0 for the remainder. S is the sidelength of the MD cell in A. CORR is a constant for the long-range potential 'dipolar' term [1].

$$\begin{aligned}&PRR(KSP)=\beta_{ij}b\exp(\sigma_{ij}/\rho)\\ &PC(KSP)=C_{ij}\\ &PD(KSP)=D_{ij}\\ &RHOI=1/\rho\end{aligned}$$

where KSP=1, 2 and 3 for ++, +- and --, respectively [2]. The energy unit is,

$$q^2/4\pi\varepsilon_0(A^{-1}) = 23.071417\times10^{-19}\ J,$$

where q is the electron charge and ε_0 is the permittivity of free space.

```
      N2=N/2
      PI=3.1415926536
      CORR=PI*0.5/(SQRT(3.0)*S**3)
      TSI=2.0/S
      DO 1 I=1,N
      FX(I)=0.0
      FY(I)=0.0
      FZ(I)=0.0
      PT(I)=0.0
1     CONTINUE
      DO 10 KSP=1,3
      IF (KSP.NE.1) goto 20
C     DO CATION-CATION INTERACTIONS
      I1=1
      I2=N2-1
      JS=0
      J2=N2
20    CONTINUE
      IF (KSP.NE.2) GOTO 30
C     DO CATION-ANION INTERACTIONS
      I1=1
      I2=N2
      J2=N
30    CONTINUE
      IF (KSP.NE.3) GOTO 40
C     DO ANION-ANION INTERACTIONS
      I1=N2+1
      I2=N-1
      JS=0
      J2=N
40    CONTINUE
C     ASSIGN POTENTIAL COEFFICIENTS
      RP=PRR(KSP)
      CP=PC(KSP)
      DP=PD(KSP)
      DO 50 I=I1,I2
      IF (KSP.EQ.2) JS=N2-I
      QI=Q(I)
      RXI=RX(I)
      RYI=RY(I)
      RZI=RZ(I)
      M=0
C     PERFORM VECTORISED INNER J LOOP
      DO 60 J=JS+I+1,J2
      M=M+1
      QIJ=QI*Q(J)
      X=RXI-RX(J)
      Y=RYI-RY(J)
      Z=RZI-RZ(J)
      X=X-INT(TSI*X)*S
      Y=Y-INT(TSI*Y)*S
      XX=X*X
      YY=Y*Y
      ZZ=Z*Z
      RR=XX+YY+ZZ
      R=SQRT(RR)
      RI=1.0/R
      RRI=RI*RI
      R6I=RRI*RRI*RRI
      R8I=R6I*RRI
      PEXP=RP*EXP(-R*RHOI)
      PC6=CP*R6I
      PD8=DP*R8I
C     CALCULATE POTENTIAL FROM WITHIN THE MD CELL
      P1=QIJ*RI+PEXP+PC6+PD8
C     CALCULATE THE COULOMB LONG RANGE CORRECTION BY GOING
C     AROUND THE NEAREST 8 J IMAGES
      XM=X-S
      YM=Y-S
      XP=X+S
      YP=Y+S
      XMS=XM*XM
      YMS=YM*YM
      XPS=XP*XP
      YPS=YP*YP
C     FIRST THE POTENTIAL
      P2=1.0/SQRT(ZZ+XMS+YY)
      P3=1.0/SQRT(ZZ+XPS+YPS)
      P4=1.0/SQRT(ZZ+XX+YMS)
      P5=1.0/SQRT(ZZ+XMS+YMS)
      P6=1.0/SQRT(ZZ+XPS+YY)
      P7=1.0/SQRT(ZZ+XX+YPS)
      P8=1.0/SQRT(ZZ+XMS+YPS)
      P9=1.0/SQRT(ZZ+XPS+YMS)
      P=P1+QIJ*((P2+P3+P4+P5+P6+P7+P8+P9)
     1+CORR*(XX+YY-ZZ-ZZ))
C     SECONDLY THE FORCES
      F1=QIJ*RRI*RI+(R*RHOI*PEXP+6.0*PC6+8.0*PD8)*RRI
      F2=P2/(ZZ+XMS+YY)
      F3=P3/(ZZ+XPS+YPS)
      F4=P4/(ZZ+XX+YMS)
      F5=P5/(ZZ+XMS+YMS)
      F6=P6/(ZZ+XPS+YY)
      F7=P7/(ZZ+XX+YPS)
      F8=P8/(ZZ+XMS+YPS)
      F9=P9/(ZZ+XPS+YMS)
      F396=F3+F9+F6
      F47=F4+F7
      F258=F2+F5+F8
      XF=(X*F47+XP*F396+XM*F258)*QIJ+X*F1-CORR*QIJ*2.0*X
      YF=(Y*(F2+F6)+YP*(F3+F7+F8)+YM*(F4+F5+F9))*QIJ+Y*F1
     1-CORR*QIJ*2.0*Y
      ZF=Z*(F396+F47+F258)*QIJ+Z*F1+CORR*4.0*Z*QIJ
      FXI(M)=XF
      FYI(M)=YF
      FZI(M)=ZF
      FX(J)=FX(J)-XF
      FY(J)=FY(J)-YF
      FZ(J)=FZ(J)-ZF
      PTI(M)=P
      PT(J)=PT(J)+P
60    CONTINUE
      MAX=M
      FX(I)=FX(I)+SSUM(MAX,FXI,1)
      FY(I)=FY(I)+SSUM(MAX,FYI,1)
      FZ(I)=FZ(I)+SSUM(MAX,FZI,1)
      PT(I)=PT(I)+SSUM(MAX,PTI,1)
50    CONTINUE
10    CONTINUE
```

The CRAY-FORTRAN subroutine SSUM(N,X,I) sums the first N elements of array X. I is the spacing between the elements in the sum.

References

[1] D.M. Heyes, 'A New Method for Performing MD and MC on Point Charge Systems I: The Lamina', Daresbury Laboratory **Information Quarterly for MD/MC Simulations**, No. 9, 20-27, June 1983.

[2] D.M. Heyes,' Program Units for Molecular Dynamics II. Born-Mayer-Huggins Potential', Daresbury Laboratory **Information Quarterly for MD/MC Simulations**, No. 9, 35-40, June 1983.

Alternatives to the Periodic Cube in Computer Simulation

David Adams

In two dimensions, if you want periodic boundary conditions, there is a choice of a periodic cell with either four sides or six, the square or the hexagon. Any parallelogram is just a distorted square and fills space with the same packing as squares. Periodic cells have to fill space merely by translation, so triangles are no use:

In three dimensions there are five shapes which fill space in the required way, the five parallelohedra of the crystallographer E.S. Fedorov. They are:

1. The good old cube or parallelopiped which packs as simple cubic.

2. The hexagonal prism. I have never heard of this being used in computer simulation, but there is no great reason why it should not be. However, as even hexagonal close packing can be accommodated in the periodic cube[1] there has never been any need for it.

3. The "elongated" dodecahedron with 28 edges, 18 vertices, and 12 faces, eight of them with four and four with six edges. I have never heard of this being used in computer simulation and I can't think of any sound, scientific reason why it should be used.

4. The rhombic dodecahedron, with 24 edges, 14 vertices and 12 faces, each with four edges. It packs as face centred cubic. It can be produced by taking a cube and cutting off the edges, while preserving the full symmetry of the cube, until exactly one quarter of the cubes' volume is left. This periodic cell has been used[2], though hardly extensively. It has the advantage in that of all the five shapes it has the largest inscribed sphere, much larger than that of the cube, which has a very unspherical shape. So if you want the maximum possible range for a radial distribution function for a given number of particles in the periodic cell, this is the periodic shape to use. Of all possible shapes this will give you the maxium distance between a particle and its own periodic images. Its disadvantage is obvious, it isn't too easy to program, and is likely to be sufficiently slow that its

advantages are outweighed: it is probably better to use a periodic cube with a larger number of particles. However, I think some research into this could be useful.

5. The 14-hedron or cubo-octahedron or orthic tetrakai-decahedron or truncated octahedron, with 36 edges, 24 vertices and 14 faces. It packs as body centred cubic. It can be produced by taking a cube and cutting off the corners, while preserving the full symmetry of the cube, until exactly one half of the cubes' volume is left. This periodic cell is being used[3,4] and deserves serious consideration. Of the five shapes it has the smallest circumsphere and so may fairly be described as the most nearly spherical of the periodic cell-shapes available. It is compared with the cube and the rhombic dodecahedron in table 1. It has only a slightly smaller inscribed sphere than the rhombic dodecahedron. Its advantage over the rhombic dodecahedron is that truncated octahedral boundary conditions are relatively simple to program. Figure 1 is a drawing of a truncated octahedron inside a cube and figure 2 shows the FORTRAN code used in the inner, force calculating loop of a molecular dynamics program. The code assumes that the truncated octahedron is cut from a cube of unit length. The first part is identical to the code required for simple cubic periodic boundary conditions[5]. This either brings the vector (DX, DY, DZ) into the nearest-neighbour truncated octahedron or leaves it in one of eight surrounding truncated octahedra which share one of its six sided faces. The equation of the plane containing this shared surface in the positive octant is:

$$x + y + z = 3/4$$

Given this, the code to find the true nearest-neighbour (DX, DY, DZ) is fairly obvious. The code can be vectorized for the CRAY by replacing the IF statement with the CVMGM function. Both Steve Thompson and myself have experimented with various vectorized versions and we found it to be always much slower than when simple cubic periodic boundary conditions are used. However, that was only for the CRAY and the extra overhead of the truncated octahedron should not be large on other machines, or when the force calculation itself is substantial. So although these boundary conditions would be unfavourable with a simple Lennard-Jones potential on the CRAY, they might be attractive with an Ewald potential. The k-space part of the Ewald summation is then slightly different as the reciprocal lattice of body centred cubic is face centred cubic. The attraction of the truncated octahedron with the Ewald summation is that the anisotropic parts of the potential, and in particular the interactions between charges out into the corners of the cell, should be smaller and the extra time required to find the nearest images easily outweighed by a reduction in the number of reciprocal lattice vectors required. The truncated octahedron becomes doubly attractive if one is simulating a single ion in a dipolar solvent, for then the distance between periodic images of the ion is considerably increased without increasing the quantity of solvent.

Truncated octahedral periodic boundary conditions may be used with a number of crystal structures, the list of numbers available with the most common is shown in Table 2.

There is a way round having to use one of the five shapes: use a non-Euchidean space. At a liquid-state conference at the University of Canterbury in 1973 Isenberg read a paper called "Optimum Boundary Conditions in Molecular Dynamics Calculations." The final two paragraphs of the abstract of that paper say:

"The solution to the anisotropy problem in 2-D is to calculate the motions of the particles on the surface of a sphere, with particles interacting along great circles only. In 3-D the volume is the surface of a 4-D, in which interactions lie along the great circles of the 4-D sphere. This model gives complete, statistical, isotropy of directional properties and isotropy of 'image' distances. However, the coordinate system is no longer cartesian but will approach a cartesian system with short-range forces as the size of the sphere is increased.

"This 'spherical' box has the additional advantage that the total potential energy between any two particles, summed over all images, can be calculated analytically. Thus no 'cut off' in the potential energy function has to be introduced."

This method has not received much attention, though it does crop-up occasionally. There are two minor variants. The interaction between two particles may be taken along the shorter area of the great circle or both long and short routes may be included. As far as I can determine the first publication in which spherical boundary conditions were used was for the 2D one component plasma by Hansen *et al*[6]. Schreiner[7] presented some results for the 3D Lennard-Jones fluid at the CCP5 Manchester meeting; they struck me as more number dependant than one would expect with normal boundary conditions. However, Kratky[8] has shown that a rigorous correction for the number dependence might be possible.

An obvious disadvantage of the non-Euclidean space is that one does not get ordinary crystalline solid packing[9]. Quite possibly the packing of a high density fluid phase will also be distorted. The advantage of spherical boundary conditions, as seen by Isenberg, is that they avoid the considerable anisotropy of periodic cube boundary conditions. My own feeling is that periodic truncated octahedron boundary conditions offer a better compromise. The anisotropy is much smaller than with a cube and the distortions of a non-Euclidean geometry is avoided. However, there is scope for more work in this area, little is known about the effects of the shape of the periodic cell on the results obtained.

References

1. D.A. Young & B.J. Alder, J.Chem. Phys. 73, 2430 (1980)
2. S.S. Wang & J.A. Krumhand, J.Chem. Phys. 56, 4287 (1972)
3. K.R. Wilson, talk given at CCP5 Reading meeting (1982)
4. D.J. Adams, J.Chem. Phys. 78, 2585 (1983)
5. D.J. Adams, CCP5 Info. Quart. 3 (1981)
6. J.P. Hansen, D. Levesque & J.J. Weis, Phys.Rev. Lett. 43, 979 (1979)
7. W. Schreiner, talk given at CCP5 Manchester meeting (1982), also W. Schreiner & K.W. Kratky, J.Chem. Soc. Farad. Trans. II 78, 379 (1982)
8. K.W. Kratky, J. Comput. Phys. 37, 205 (1980)
9. A.L. Mackay, J.Phys. A: Maths. Gen. 13, 3373 (1980)

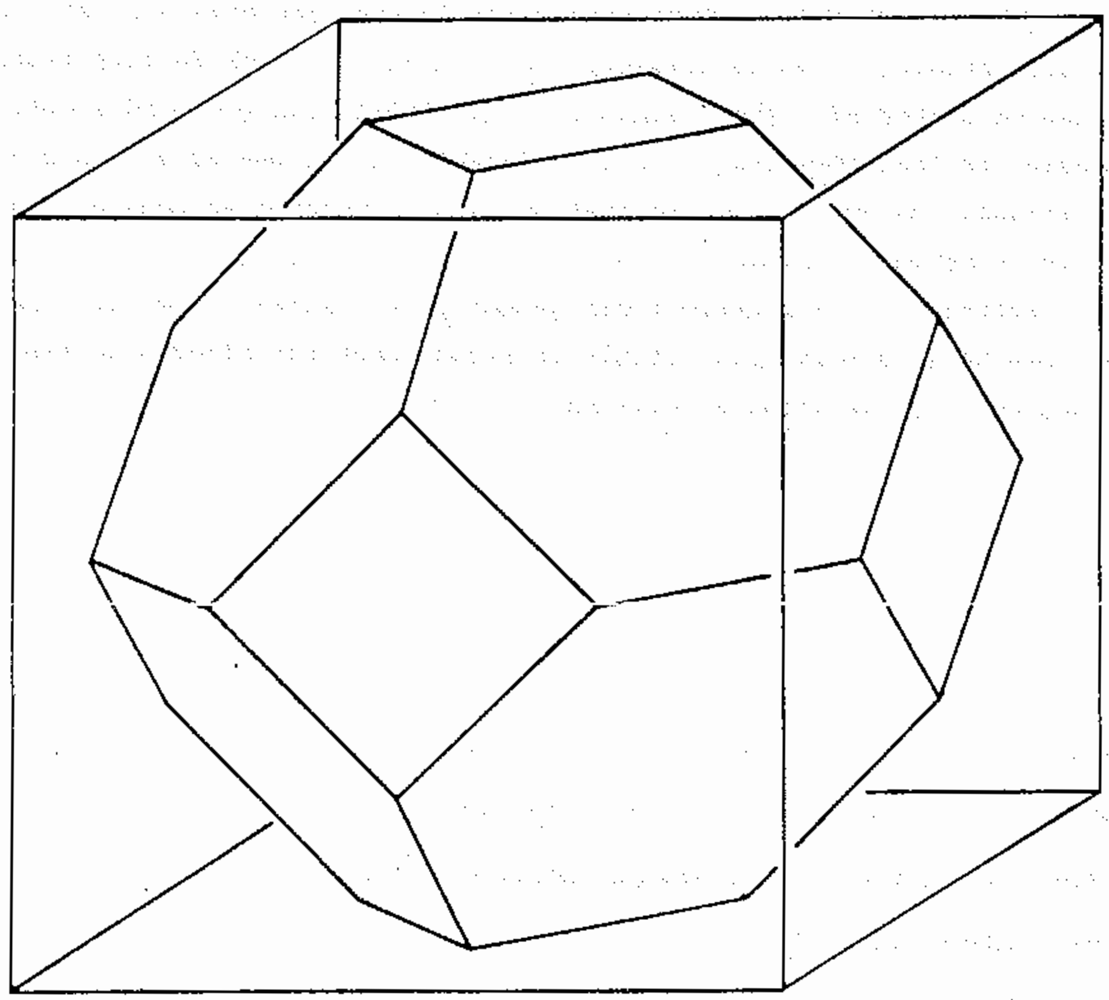


Fig. 1 The Truncated Octahedron

Table 1

Comparison of cube, truncated octahedron (TO), and rhombic dodecahedron (RD)

Shape	$\frac{\text{Circumsphere radius}}{\text{Inscribed sphere radius}}$	$\frac{\text{Inscribed sphere volume}}{\text{volume}}$	$\frac{\text{Circumsphere volume}}{\text{volume}}$
Cube	$\sqrt{3} \simeq 1.73$	$\pi/6 \simeq 0.52$	$\sqrt{3}\pi/2 \simeq 2.72$
TO	$\sqrt{5/3} \simeq 1.29$	$\sqrt{3}\pi/8 \simeq 0.68$	$5\sqrt{5}\pi/24 \simeq 1.46$
RD	$\sqrt{2} \simeq 1.41$	$\sqrt{2}\pi/6 \simeq 0.74$	$2\pi/3 \simeq 2.09$

Table 2

Numbers of primitive cells that can be used with periodic truncated-octahedral boundary conditions.

Lattice type	General Formula	1st few terms
simple cubic	$\frac{1}{2}(2n)^3$	4, 32, 108, 256, 500, 864 ...
body-centred cubic	$(2n)^3$	8, 64, 216, 512, 1000 ...
face-centred cubic	$1/4(4n)^3$	16, 128, 432, 1024 ...

```
      DX  =  X(I) - X(J)
      DX  =  DX - AINT (2 * DX)
      DY  =  Y(I) - Y(J)
      DY  =  DY - AINT (2 * DY)
      DZ  =  Z(I) - Z(J)
      DZ  =  DZ - AINT (2 * DZ)
      IF (ABS(DX) + ABS(DY) + ABS(DZ).LT. 0.75) GØTØ 1
      DX  =  DX - SIGN (0.5, DX)
      DY  =  DY - SIGN (0.5, DY)
      DZ  =  DZ - SIGN (0.5, DZ)
    1 CONTINUE
```

Fig. 2 Calculation of the vector (DX, DY, DZ) between the nearest-neighbour images of particles I & J as it might appear in the inner loop of a m.d. force calculation. The containing cube of the truncated octahedron is of unit length.

THE PERIODIC BOUNDARY CONDITION IN NON-CUBIC MD CELLS: WIGNER-SEITZ CELLS WITH REFLECTION SYMMETRY.

W. Smith

David Adams pointed out in his earlier article (1) that the great majority of computer simulations employ a periodic boundary condition (PBC) based on the simple cubic MD cell despite there being useful and sometimes preferable alternatives. My purpose in this note is to indicate how alternatives to the simple cubic PBC may be constructed and to provide coding examples for a few cases. The MD cells I shall describe form a recognisable class, which is not general, but does include some potentially useful cells such as the rhombic dodecahedron and the truncated octahedron described by David Adams. The class of MD cells I shall discuss conform to the following criteria:

(i) The MD cells are space filling (!). The space filling is accomplished through the translation of the MD cell periodically through space, without rotation, in the manner usually understood when constructing lattices.

(ii) The MD cell is symmetric with respect to reflection in various planes passing through the centre of the cell. These planes are so constructed as to establish an equivalence between faces, vertices or edges of the cell and are generally easy to identify. As a minimum however, every face of the cell is related to another through reflection symmetry. (This symmetry naturally applies only to the cells and not to the particles they contain).

The criterion (ii) above permits a simple description of the geometry of the cell; in terms of the set of vectors defining the location of the centres of the immediate neighbouring cells. Every face of the MD cell can be defined by an equation of the form:

$$d_i = \underline{r}_i \cdot \underline{u}_i \qquad (1)$$

where:

$\underline{r}_i$ is a vector defining a point in the plane containing the 'i th' cell face.

$\underline{u}_i$ is a unit vector originating at the centre of the MD cell and pointing towards the centre of the 'i th' neighbouring cell.

$2d_i$ is the distance between the cell centre and the centre of the 'i th' neighbouring cell.

Such a description is possible because each vector u associated with a cell face must, by definition, be perpendicular to that face. (It should also be remarked that while every face of the cell can be defined by a vector $\underline{u}_i$, it does not necessarily follow that every vector $\underline{u}_i$ has a face associated with it, since cells may be neighbours in the sense that they are are in contact with each other via an edge or a vertex rather than via a cell face). The similarity of the MD cells in this class to the Wigner-Seitz cells favoured by solid state physicists is obvious.

The unit vectors $\underline{u}_i$ are particularly useful in performing the particle relocations that are associated with the periodic boundary condition. Suppose that in the course of a MD simulation a particle has moved out of the simulation cell and is now in the 'i th' neighbouring cell. We naturally wish to relocate the particle to its periodic image within the original cell. This can be done by applying the following vector operation (in which the vector $\underline{r}$ locates the particle with respect to the geometric centre of the cell):

$$T_i(\underline{r}) = \underline{r} - 2*\underline{u}_i*\mathrm{AINT}(\underline{r}\cdot\underline{u}_i/d_i) \qquad (2)$$

The effect of this operation is that the particle is moved to its appropriate periodic image position, provided that the particle is in the 'i th' neighbouring cell at the instant the operation is carried out. If however, the particle is not in the 'i th' neighbouring cell, one of three results may be obtained:

(i) The particle remains in the same position (as for instance, when the particle is already in the original cell).

(ii) The particle is moved to another cell neighbouring the original cell, but at the appropriate image position.

(iii) The particle is moved to the correct position in the original cell even though the $\underline{u}_i$ vector is not appropriate to that neighbouring cell.

Examples of these effects can be seen most clearly in the two - dimensional case presented in Figure 1, in which a square cell is used. The results described are relevant only to the class of cells having the reflection symmetry described earlier. It is worth noting that the operation described by equation (2) is defined with respect to the vectors $\underline{u}_i$, which are more numerous than the cell faces. This means that when applying the periodic boundary condition, we must consider edges and vertices as well as cell faces to be sure of a correct procedure.

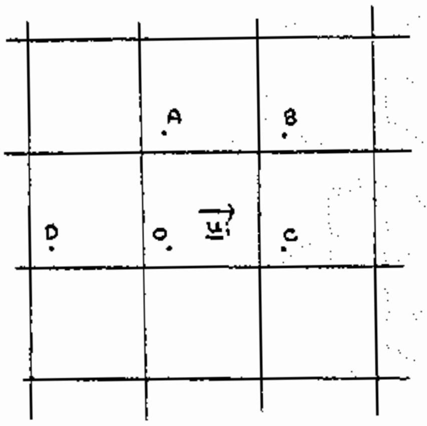


Figure 1. Image particles A,B,C,D and O are affected differently by the operation T_i ($\underline{r}$) associated with the vector $\underline{u}_i$. Both A and O are unaffected. B is moved to the site of A, while both C and D are relocated to O; the correct position.

In principle, a PBC algorithm may be constructed from the set of T_i ($\underline{r}$) operations described in equation (2). What is required is a sequence of these operations, which collectively will transform any position vector to the correct periodic image. Such a sequence is not, in general, easy to construct and some trial - and - error is required before a workable scheme emerges. There are however some simplifying features, namely the symmetry of the cells and the ability of each of the operations to affect the contents of more than one neighbouring cell. These features combine to permit PBC algorithms very much simpler than might be expected from the large number of T_i ($\underline{r}$) operations . (In practice only a small number of T_i ($\underline{r}$) operations is required).

These comments can best be illustrated by a specific example. Consider the two - dimensional hexagonal PBC displayed in Figure (2). (The hexagon is a two - dimensional equivalent of the class of cells described earlier). The effect of the operation T_i ($\underline{r}$) on a particle in any of the cells neighbouring the central MD cell, is to move the particle to the left of the line AB a distance of 2*d to the right. Similarly particles right of line CD are moved 2*d to the left. Thus all particles are brought within the parallel lines AB and CD. We may then apply yet another operation T_j ($\underline{r}$), to reduce this area yet again and thus by repeated application of different operations finally ensure that all particles are correctly accounted for. However after the first operation a more elegant alternative arises.

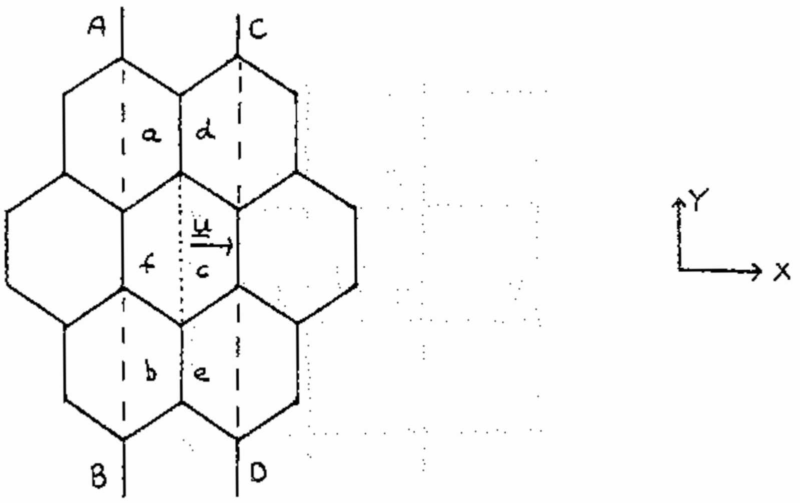


Figure (2). The 2-D Hexagonal PBC. (Width of cell = 2d).

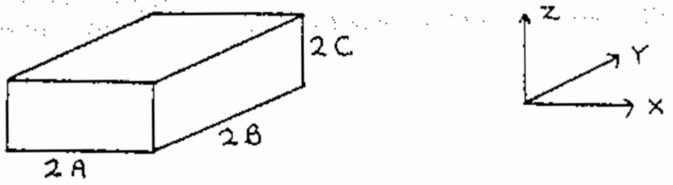
Particles in the regions 'a' and 'b' clearly require relocation to region 'c' as indicated. Similarly those from regions 'd' and 'e' are destined for region 'f'. There is, surprisingly, a simple procedure that will do this conveniently (due, I believe, to David Adams). In FORTRAN it takes the form:

```
      IF(ABS(X)+ √3.0*ABS(Y).LT.2.0*D)GO TO 10
      X=X-SIGN(D,X)
      Y=Y-SIGN( √3.0*D,Y)
10    CONTINUE
```

In this procedure, the first statement checks if the particle is outside the cell boundary (effectively using a scalar product similar to that used in equation (2)), while the two following statements apply the required translation operation. These statements, together with the single operation T_1 ($\underline{r}$) constitute the PBC algorithm for this case.

I conclude this article by listing some codes that may be used for various periodic boundary conditions. They have been constructed using the (rather vague) procedure outlined above. The intrepid reader may wish to attempt to derive the algorithms for himself, indeed he may be well advised to do so ! (The exercise is not unlike the manipulation of the infernal Rub ik cube!).

A. Rectangular Box

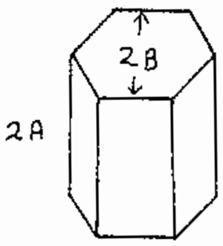


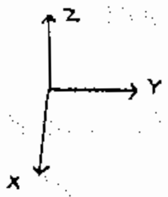
Useful unit vectors: (1,0,0),(0,1,0),(0,0,1)

Algorithm:

```
        X=X-2.0*A*AINT(X/A)
        Y=Y-2.0*B*AINT(Y/B)
        Z=Z-2.0*C*AINT(Z/C)
```

B. Hexagonal Prism



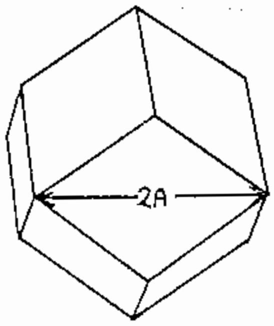


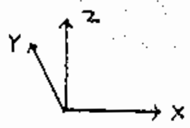
Useful unit vectors: (1,0,0),(0,0,1),(1/2,$\sqrt{3}$/2,0)

Algorithm:

```
        Z=Z-2.0*A*AINT(Z/A)
        X=X-2.0*B*AINT(X/B)
        IF(ABS(X)+ √3.0*ABS(Y).LT.2.0*B)GO TO 10
        X=X-SIGN(B,X)
        Y=Y-SIGN( √3.0*B,Y)
10      CONTINUE
```

C. Rhombic Dodecahedron



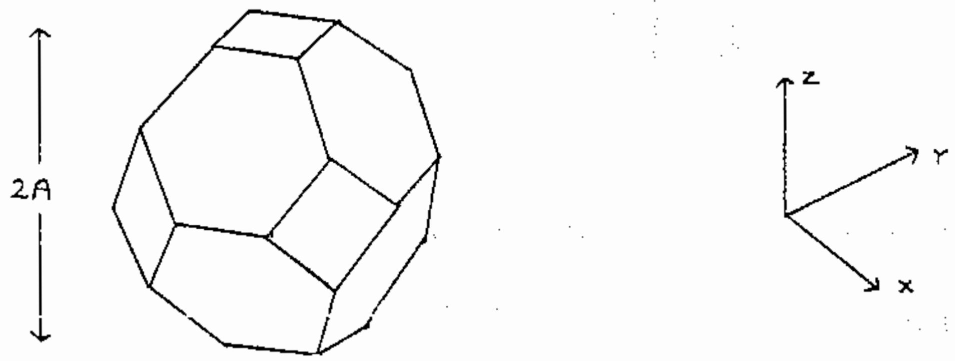


Useful unit vectors: (1,0,0),(0,1,0),(0,0,1),(1,1,$\sqrt{2}$)/2

Algorithm:

```
      Z=Z- √8.0*A*AINT(Z/( √2.0*A))
      X=X-2.0*A*AINT(X/A)
      Y=Y-2.0*A*AINT(Y/A)
      IF(ABS(X)+ABS(Y)+ √2.0*ABS(Z).LT.2.0*A)GO TO 10
      X=X-SIGN(A,X)
      Y=Y-SIGN(A,Y)
      Z=Z-SIGN( √2.0*A,Z)
10    CONTINUE
```

D. Truncated Octahedron



Useful unit vectors: (1,0,0),(0,1,0),(0,0,1),(1,1,1)/$\sqrt{3}$

Algorithm:

```
      X=X-2.0*A*AINT(X/A)
      Y=Y-2.0*A*AINT(Y/A)
      Z=Z-2.0*A*AINT(Z/A)
      IF(ABS(X)+ABS(Y)+ABS(Z).LT.1.5*A)GO TO 10
      X=X-SIGN(A,X)
      Y=Y-SIGN(A,Y)
      Z=Z-SIGN(A,Z)
10    CONTINUE
```

References.

(1) D. Adams, CCP5 Info. Quart. No.10, (Sept.) 1983.

ROTATIONAL MOTION OF LINEAR MOLECULES

David Fincham

DAP Support Unit, Queen Mary College

In a previous article (CCP5 Newsletter Number 2, September 1981) I described a simple algorithm for the rotational motion of rigid polyatomic molecules, based on a quaternion representation of the orientation.

The four quaternion parameters are related by a single constraint equation and provide a suitable representation for the orientation of non-linear molecules which have three degrees of rotational freedom. They are less suitable for use with linear molecules which have only two degrees of rotational freedom, as there would then be a second implicit constraint which numerical errors might violate. For these molecules I prefer an algorithm which represents the orientation of the molecule by the cartesian components of a vector along its axis, with a constraint on its length. I learned this algorithm from Konrad Singer, and several other people have used it, but I do not think the details have appeared in print. Furthermore the formulation given below makes clear two points that are not always realised; the algorithms can give the correct dynamics for any linear molecule, not just diatomics and the forcecentres need not necessarily correspond with the mass centres. The discussion below concerns only the rotational motion which of course can be handled completely independently of the translational motion.

We specify the orientation of the molecule by $\underline{e}$, a unit vector along its axis. Let its moment of inertia about a perpendicular axis through the centre of mass be I. If the force centres are at positions $d_\alpha \underline{e}$ relative to the COM the torque on the molecule is:

$$\underline{T} = \underline{e} \times \sum_\alpha d_\alpha \underline{f}_\alpha$$

We study the rotational motion by applying the method of constraints to an 'equivalent diatomic pseudo-molecule'. Let the pseudo-molecule have unit length with masses m at each end on which forces $\underline{G}$ and $-\underline{G}$ act. Its rotational motion will be the same as that of the actual linear molecule provided it has the same moment of inertia and the same torque acts upon it. These conditions are satisfied if:

$$m = 2I$$

and

$$\underline{G} = \sum_\alpha d_\alpha \underline{f}_\alpha - \underline{e}\,(\underline{e}.\sum_\alpha d_\alpha \underline{f}_\alpha)$$

The second term here subtracts out the component parallel to $\underline{e}$, which is irrelevant to the rotational motion. It is not essential but convenient to do this since then $G^2 = \underline{T}^2$ and we can obtain the mean square torque which is a useful number to get out of a simulation.

If a and b are the two 'atoms' of the pseudo-molecule they move under the influence of the forces $\underline{G}$ and of undetermined bond forces acting along the axis of the molecule. Applying the leapfrog algorithm to this motion gives:

$$\underline{r}_a^{n+1} = \underline{r}_a^n + \Delta t \dot{\underline{r}}_a^{n-\frac{1}{2}} + (\Delta t^2/2I)\,\underline{G}^n + \tfrac{1}{2}\lambda \underline{e}^n$$

$$\underline{r}_b^{n+1} = \underline{r}_b^n + \Delta t \dot{\underline{r}}_b^{n-\frac{1}{2}} - (\Delta t^2/2I)\,\underline{G}^n - \tfrac{1}{2}\lambda \underline{e}^n$$

or, since $\underline{e} = \underline{r}_a - \underline{r}_b$

$$\underline{e}^{n+1} = \underline{e}^n + \Delta t\, \dot{\underline{e}}^{n-\frac{1}{2}} + (\Delta t^2/I)\,\underline{G}^n + \lambda \underline{e}^n$$

$$= \hat{\underline{e}} + \lambda \underline{e}^n$$

where $\hat{\underline{e}}$ is the axis vector which would result from 'free-flight' alone. The multiplier λ is determined by the condition that the length of the axis must be preserved:

$$\text{i.e.}\quad 1 = |\underline{e}^{n+1}|^2$$

$$= |\underline{e}|^2 + 2\lambda\, \underline{e}.\underline{e}^n + \lambda^2 |\underline{e}^n|^2$$

giving, remembering that $|\underline{e}^n|^2 = 1$,

$$\lambda = -\,\underline{e}.\underline{e}^n + \left[(\underline{e}.\underline{e}^n)^2 - \underline{e}^2 + 1\right]^{\frac{1}{2}}$$

The algorithm is completed by calculating the new axis vector velocity:

$$\dot{\underline{e}}^{n+\frac{1}{2}} = (\underline{e}^{n+1} - \underline{e}^{n})/\Delta t$$

Molecular Dynamics of Superionic Conductors

M.L. Wolf and C.R.A. Catlow

Earlier M.D. work on superionics - that is solids with exceptionally high ionic conductivities - demonstrated the value of the technique in elucidating details of structure and transport in this important class of compound.(1)(2) These earlier studies were, however, confined to relatively simple cubic materials e.g. AgI and CaF_2. Recently we have applied the technique to the more complex, layer structured superionics, $\beta''Al_2O_3$ and Li_3N. Our work exploits the efficiency of the FUNGUS programs developed by Walker for M.D. studies of ionic crystals; the program is written specifically for use on the CRAY.

Our work on $\beta''Al_2O_3$ demonstrates an intriguing change with temperature in the structural properties of the material. At lower temperatures, a well defined lattice structure is found for the conducting Na^+ ions which migrate by a hopping mechanism. This gives way at higher temperatures to more liquid-like structural and transport properties. Confidence in the reliability of the calculations is encouraged by their success in reproducing observed diffusion coefficients.

In the case of Li_3N the work has revealed intriguing migration mechanisms for the cations. The simulations show that the Li^+ ions move within the layers by complex concerted migration mechanisms involving several cations. It has often been speculated that such mechanisms could be significant in superionics. These simulation studies provide the first strong evidence for their occurrence.

Our dynamical simulations on superionics are now being extended to include the highly defective high temperature Bi_2O_3 phase for which neutron scattering work is in progress.

(1) Vashishta P. and Rahman A., Phys. Rev. Lett. 40, 1337 (1978).
(2) Gillan M.J. and Dixon M., J. Phys. C. 13, 1901 (1980).