Advanced Computing

Starting-up:
How to setup the initial conditions for a Molecular Dynamic Simulation

Javier Junquera

UNIVERSIDAD DE CANTABRIA

CITIMAC

The initial configuration

In molecular dynamic simulations it is necessary to design a **starting configuration** for the first simulation

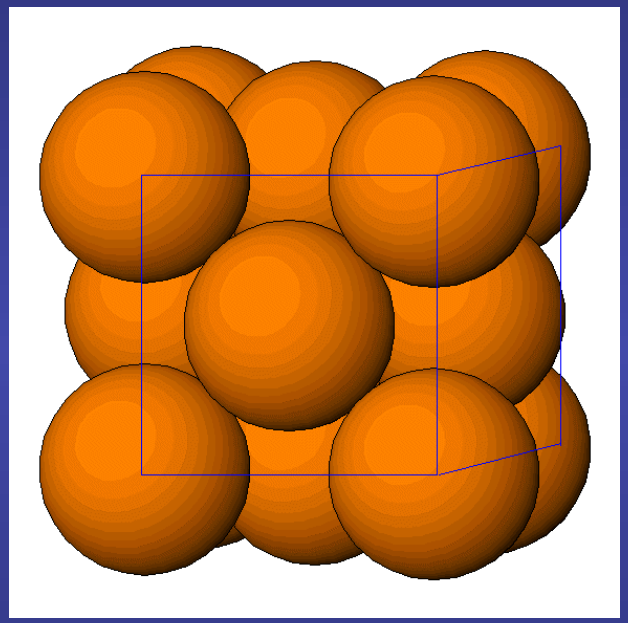
- Initial molecular positions and orientations
- Initial velocities and angular velocities

For the first run, it is important to choose a configuration that can relax quickly to the structure and velocity distribution appropriate to the fluid

This period of equilibration must be monitored carefully, since the disapperance of the initial structure might be quite slow

The initial configuration: more usual approach, start from a lattice

Almost any lattice is suitable.
Historically, the face-centered cubic structure has been the starting configuration



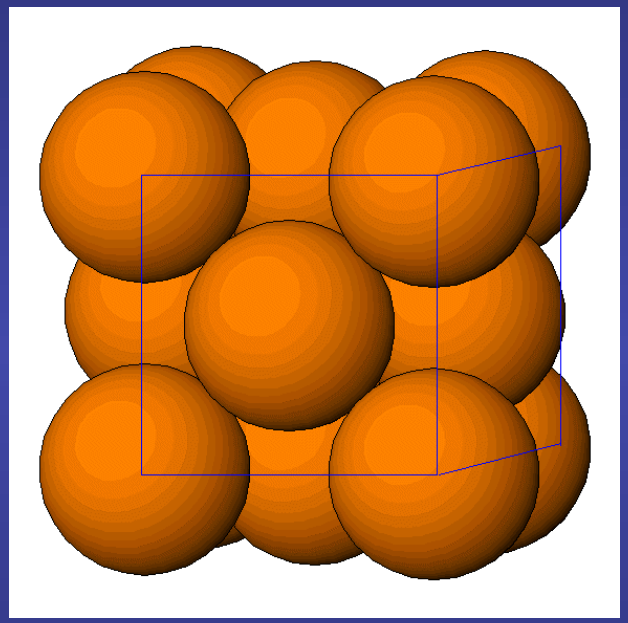
The lattice spacing is chosen so the appropriate liquid state density is obtained

During the course of the simulation, the lattice structure will disappear, to be replaced by a typical liquid structure

This process of “melting” can be enhanced by giving each molecule a small random displacement from its initial lattice point

The initial configuration: more usual approach, start from a lattice

Almost any lattice is suitable.
Historically, the face-centered cubic structure has been the starting configuration



A supercell is constructed repeating the conventional cubic unit cell of the FCC lattice N_c times along each direction

The number of atoms in the simulation box, N, is an integer of the form $N = 4N_c^3$, where N_c is the number of FCC unit cells in each direction

Units for the density

For systems consisting of just one type of atom or molecule, it is sensible to use the mass of the molecule as a fundamental unit

$$m_i = 1 \ \text{dimensionless}$$

With this convention:

- **Particle momenta and velocities become numerically identical**
- **Forces and accelerations become numerically identical**

In systems interacting via a Lennard-Jones potential, the density is oftenly quoted in reduced units

$$\rho^* = \rho\,\sigma^3 \ \ (\text{dimensionless})$$

Assuming an FCC lattice (4 atoms in the conventional cubic), and $m_i = 1$, then we can compute the lattice constant from the reduced density

$$\rho = \frac{\rho^*}{\sigma^3} = \frac{4}{a^3} \Rightarrow a = \left(\frac{4}{\rho^*}\right)^{1/3} \sigma$$

And then, the lattice constant will come in the same units as the ones used to determine σ

Units for the length of the supercell

If the lattice constant of the conventional cubic unit cell of an FCC lattice is a, then the length of the side of the supercell is $L = N\,a$

However, the implementation of the periodic boundary conditions and the calculation of minimum image distances is simplified by the use of reduced units: the length of the box is taken to define the fundamental unit of length of the simulation,

$$L = 1$$

In particular, the atomic coordinates can be defined using this unit of length, so nominally they will be in the range

$$\left(-\frac{1}{2}, \frac{1}{2}\right)$$

Implementation of a fcc lattice

```
C     ** CALCULATE THE SIDE OF THE UNIT CELL (CELL VECTOR OF UNITY SIZE) **
         CELL  = 1.0 / REAL ( NC )

C     ** BUILD THE UNIT CELL **

C     ** SUBLATTICE A **
         RX(1) =  0.0
         RY(1) =  0.0
         RZ(1) =  0.0

C     ** SUBLATTICE B **
         RX(2) =  CELL2
         RY(2) =  CELL2
         RZ(2) =  0.0

C     ** SUBLATTICE C **
         RX(3) =  0.0
         RY(3) =  CELL2
         RZ(3) =  CELL2

C     ** SUBLATTICE D **
         RX(4) =  CELL2
         RY(4) =  0.0
         RZ(4) =  CELL2

C     ** CONSTRUCT THE LATTICE FROM THE UNIT CELL **

         M = 0

         DO 99 IZ = 1, NC
            DO 98 IY = 1, NC
               DO 97 IX = 1, NC
                  DO 96 IREF = 1, 4

                     RX(IREF+M) = RX(IREF) + CELL * REAL ( IX - 1 )
                     RY(IREF+M) = RY(IREF) + CELL * REAL ( IY - 1 )
                     RZ(IREF+M) = RZ(IREF) + CELL * REAL ( IZ - 1 )

96                CONTINUE

                  M = M + 4

97             ENDDO
98          ENDDO
99       ENDDO

C     ** SHIFT CENTRE OF BOX TO THE ORIGIN **

         DO 100 I = 1, N
            RX(I) = RX(I) - 0.5
            RY(I) = RY(I) - 0.5
            RZ(I) = RZ(I) - 0.5
100      ENDDO
```

The simulation box is a unit cube centred at the origin

The number of atoms in the simulation box, N is an integer of the form $N = 4N_c^3$, where N_c is the number of FCC unit cells in each direction

Initial velocities

For a molecular dynamic simulation, the initial velocities of all the molecules must be specified

It is usual to choose random velocities, with magnitudes conforming to the required temperature, corrected so that there is no overall momentum

$$\vec{P} = \sum_{1=1}^{N} m_i \vec{v}_i = 0$$

The distribution of molecular speeds is given by

$$\rho(v_{ix}) = \sqrt{\frac{m_i}{2\pi k_B T}}\, e^{-\frac{m_i v_{ix}^2}{2k_B T}}$$

Probability density for velocity component

For a derivation of this expression, read Feynman lectures on Physics, Volume 1, Chapter 40-4

Similar equations apply for the y and z velocities

Normal distributions

The normal distribution with mean $\langle x \rangle$ and variance σ^2 is defined as

$$\rho(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x - \langle x \rangle)^2}{2\sigma^2}\right) \quad -\infty < x < \infty$$

A random number ζ' generated from this distribution is related to a number ζ generated from the normal distribution with zero mean and unit variance by

$$\zeta' = \langle x \rangle + \sigma\zeta$$

The Maxwell-Boltzmann distribution is a normal distribution with

$$\langle x \rangle = 0 \qquad \frac{1}{\sigma\sqrt{2\pi}} = \sqrt{\frac{m_i}{2\pi k_{\mathrm{B}} T}}$$

If we take the mass of the atoms or molecules as $m_i = 1$

$$\sigma = \sqrt{k_{\mathrm{B}} T}$$

Flowchart in the generation of random numbers

Generate a random number distribution uniformly between [0 and 1]

Implementation of random numbers uniformly distributed between 0 and 1

```
      REAL FUNCTION RANF ( DUMMY )

C     ********************************************************************
C     ** RETURNS A UNIFORM RANDOM VARIATE IN THE RANGE 0 TO 1.          **
C     **                                                                **
C     **                ***************                                 **
C     **                **  WARNING  **                                 **
C     **                ***************                                 **
C     **                                                                **
C     ** GOOD RANDOM NUMBER GENERATORS ARE MACHINE SPECIFIC.            **
C     ** PLEASE USE THE ONE RECOMMENDED FOR YOUR MACHINE.               **
C     ********************************************************************

      INTEGER     L, C, M
      PARAMETER ( L = 1029, C = 221591, M = 1048576 )

      INTEGER     SEED
      REAL        DUMMY
      SAVE        SEED
      DATA        SEED / 0 /

C     ********************************************************************

      SEED = MOD ( SEED * L + C, M )
      RANF = REAL ( SEED ) / M

      RETURN
      END
```

Flowchart in the generation of random numbers

Generate a random number distribution uniformly between [0 and 1]

From this, generate random numers following a normal (Gaussian) distribution with zero mean and unit variance

Implementation of random numbers following a Gaussian distribution

```
      REAL FUNCTION GAUSS ( DUMMY )

C     ********************************************************************
C     ** RANDOM VARIATE FROM THE STANDARD NORMAL DISTRIBUTION.          **
C     **                                                                **
C     ** THE DISTRIBUTION IS GAUSSIAN WITH ZERO MEAN AND UNIT VARIANCE.**
C     **                                                                **
C     ** REFERENCE:                                                     **
C     **                                                                **
C     ** KNUTH D, THE ART OF COMPUTER PROGRAMMING, (2ND EDITION         **
C     **    ADDISON-WESLEY), 1978                                       **
C     **                                                                **
C     ** ROUTINE REFERENCED:                                            **
C     **                                                                **
C     ** REAL FUNCTION RANF ( DUMMY )                                   **
C     **    RETURNS A UNIFORM RANDOM VARIATE ON THE RANGE ZERO TO ONE   **
C     ********************************************************************

      REAL        A1, A3, A5, A7, A9
      PARAMETER ( A1 = 3.949846138, A3 = 0.252408784 )
      PARAMETER ( A5 = 0.076542912, A7 = 0.008355968 )
      PARAMETER ( A9 = 0.029899776                   )

      REAL        SUM, R, R2
      REAL        RANF, DUMMY
      INTEGER     I

C     ********************************************************************

      SUM = 0.0

      DO 10 I = 1, 12
         SUM = SUM + RANF ( DUMMY )
10    ENDDO

      R  = ( SUM - 6.0 ) / 4.0
      R2 = R * R

      GAUSS = (((( A9 * R2 + A7 ) * R2 + A5 ) * R2 + A3 ) * R2 +A1 )
     :          * R

      RETURN
      END
```

Flowchart in the generation of random numbers

Generate a random number distribution uniformly between [0 and 1]

From this, generate random numers following a normal (Gaussian) distribution with zero mean and unit variance

$$\rho(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x - \langle x \rangle)^2}{2\sigma^2}\right) \quad -\infty < x < \infty$$

A random number ζ' generated from this distribution is related to a number ζ generated from the normal distribution with zero mean and unit variance by

$$\zeta' = \langle x \rangle + \sigma\zeta$$

The Maxwell-Boltzmann distribution is a normal distribution with

$$\langle x \rangle = 0 \qquad \frac{1}{\sigma\sqrt{2\pi}} = \sqrt{\frac{m_i}{2\pi k_{\mathrm{B}} T}}$$

If we take the mass of the atoms or molecules as $m_i = 1$

$$\sigma = \sqrt{k_{\mathrm{B}} T}$$

Implementation of the initial velocities

```
C     ***************************************************************
C     ** TRANSLATIONAL VELOCITIES FROM MAXWELL-BOLTZMANN DISTRIBUTION  **
C     **                                                             **
C     ** THE DISTRIBUTION IS DETERMINED BY TEMPERATURE AND (UNIT) MASS.**
C     ** THIS ROUTINE IS GENERAL, AND CAN BE USED FOR ATOMS, LINEAR    **
C     ** MOLECULES, AND NON-LINEAR MOLECULES.                        **
C     **                                                             **
C     ** ROUTINE REFERENCED:                                         **
C     **                                                             **
C     ** REAL FUNCTION GAUSS ( DUMMY )                               **
C     **    RETURNS A UNIFORM RANDOM NORMAL VARIATE FROM A           **
C     **    DISTRIBUTION WITH ZERO MEAN AND UNIT VARIANCE.           **
C     ***************************************************************

         RTEMP = SQRT ( TEMP )

         DO 100 I = 1, N
            VX(I) = RTEMP * GAUSS ( DUMMY )
            VY(I) = RTEMP * GAUSS ( DUMMY )
            VZ(I) = RTEMP * GAUSS ( DUMMY )
100      ENDDO

C     ** REMOVE NET MOMENTUM **

         SUMX = 0.0
         SUMY = 0.0
         SUMZ = 0.0

         DO 200 I = 1, N
            SUMX = SUMX + VX(I)
            SUMY = SUMY + VY(I)
            SUMZ = SUMZ + VZ(I)
200      ENDDO

         SUMX = SUMX / REAL ( N )
         SUMY = SUMY / REAL ( N )
         SUMZ = SUMZ / REAL ( N )

         DO 300 I = 1, N
            VX(I) = VX(I) - SUMX
            VY(I) = VY(I) - SUMY
            VZ(I) = VZ(I) - SUMZ
300      ENDDO
```

Units of temperature and velocity

The temperature is usually given in reduced units

$$T^* = \frac{k_\mathrm{B} T}{\varepsilon} = \frac{T}{\varepsilon / k_\mathrm{B}} \quad (\mathrm{adimensional})$$

(remember that ε is usually tabulated as $\varepsilon / k_\mathrm{B}$, in K)

In this system of units, the velocities are given in units of square root of the temperature

Typical system sizes

The size of the system is limited by the:

- available storage on the host computer
- speed of execution of the program

The double loop used to evaluate the forces or the potential energy is proportional to N^2

Special techniques may reduce this dependency to $\mathcal{O}(N)$,
but the force /energy loop almost inevitably dictates the overall speed.

Boundary conditions

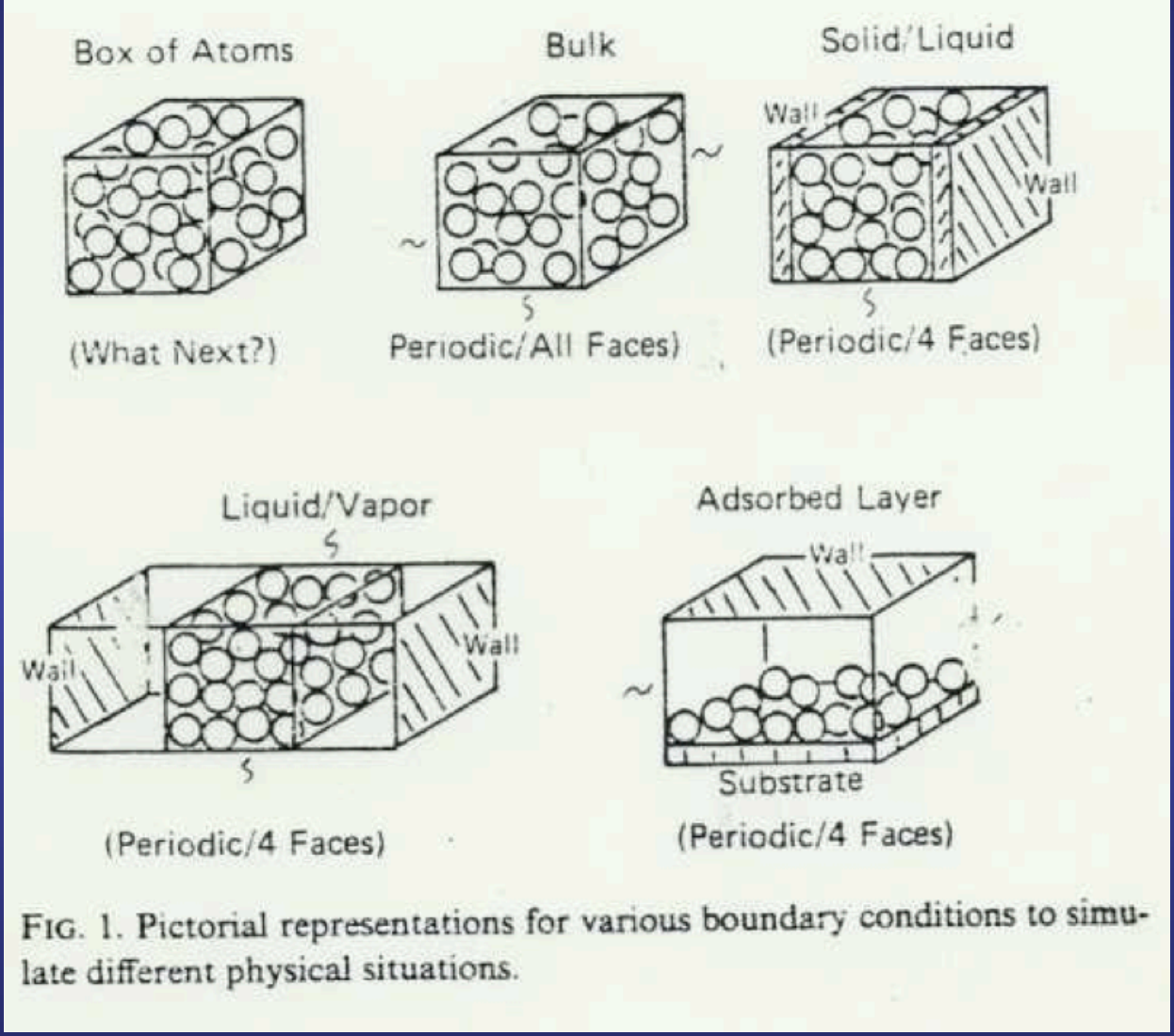
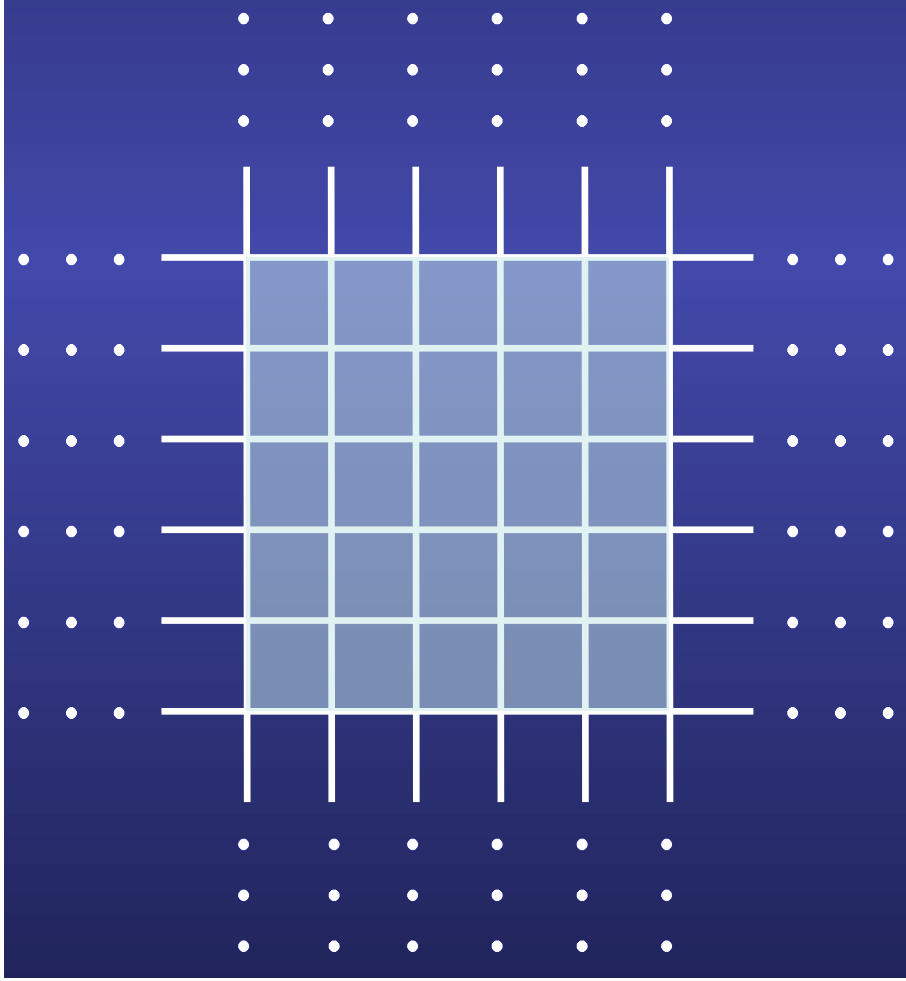


FIG. 1. Pictorial representations for various boundary conditions to simulate different physical situations.

How can we simulate an infinite periodic solid?
Periodic (Born-von Karman) boundary conditions

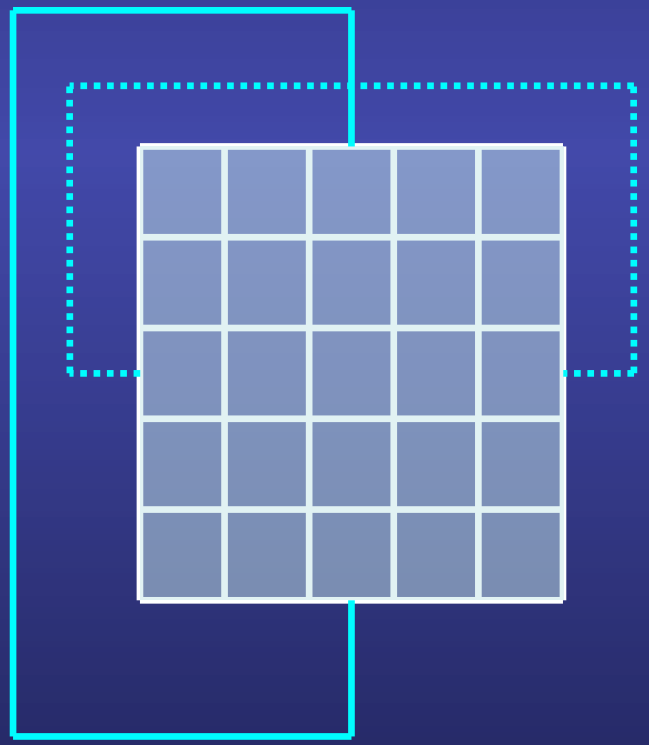
We should expect that the bulk properties to be unaffected by the presence of its surface.

A natural choice to emphasize the inconsequence of the surface by disposing of it altogether



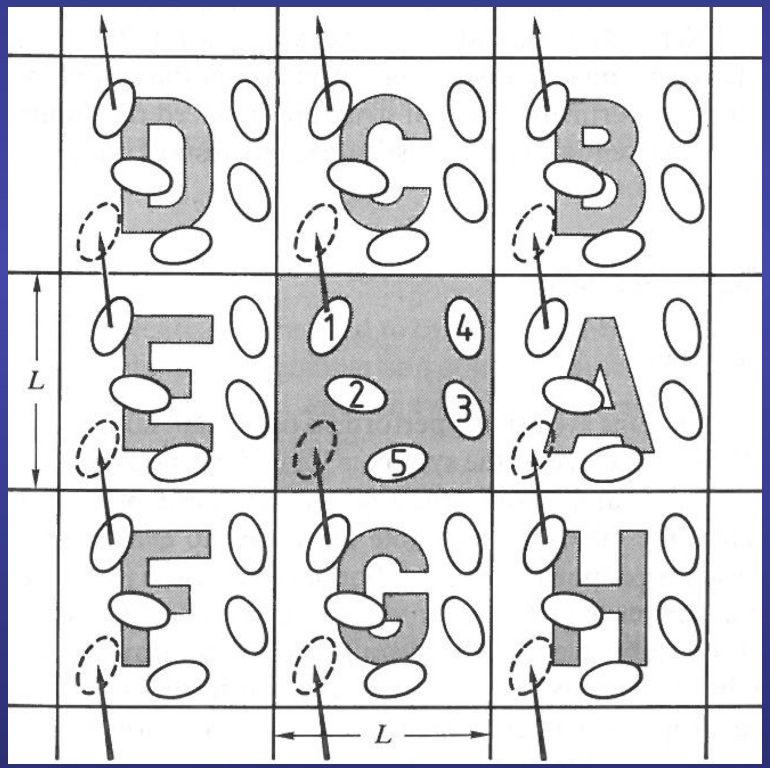
Supercell +

Born-von Karman boundary conditions



Periodic (Born-von Karman) boundary conditions

The box is replicated throughout space to form an infinite lattice



In the course of the simulation, as a molecule moves in the original box, its periodic image in each of the neighboring boxes moves in exactly the same way

Thus, as a molecule leaves the central box, one of its images will enter through the opposite face

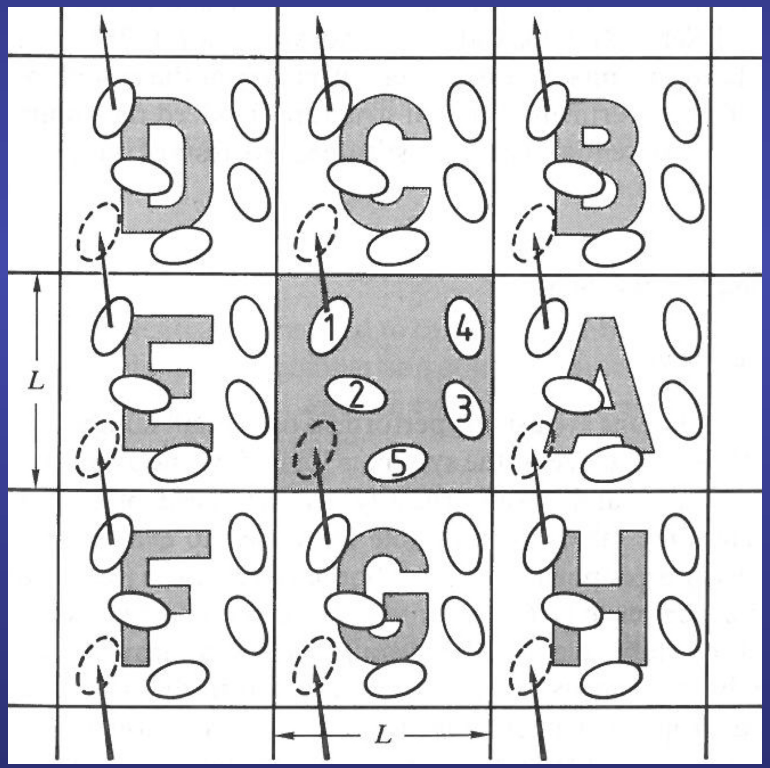
There are no walls at the boundary of the central box, and no surface molecules. This box simply forms a convenient axis system for measuring the coordinates of the N molecules

The density in the central box is conserved

It is not necessary to store the coordinates of all the images in a simulation, just the molecules in the central box

Periodic (Born-von Karman) boundary conditions: influence on the properties

It is important to ask if the properties of a small, infinitely periodic, system and the macroscopic system which it represents are the same



For a fluid of Lennard-Jones atoms, it should be possible to perform a simulation in a cubic box of side

$$L \approx 6\sigma$$

without a particle being able to "sense" the symmetry of the periodic lattice

Periodic (Born-von Karman) boundary conditions: drawbacks and benefits

Drawbacks

- Inhibits the occurrence of long-wave length fluctuations. For a cube of side L , the periodicity will suppress any density wave with a wave length greater than L .

- Be careful in the simulation of phase transitions where the range of critical fluctuations is macroscopic, and of phonons in solids.

Benefits

- Common experience in simulation work is that periodic boundary conditions have little effect on the equilibrium of thermodynamic properties and structures of fluids:
  - away from phase transitions
  - where the interactions are short-range

If the resources are available, it is always sensible to increase the number of molecules (and the box size, so as to maintain constant density) and rerun the simulations

Periodic (Born-von Karman) boundary conditions and external potentials

Up to now, we have assumed that there is no external potential
(i.e. no $v_1(\vec{r}_i)$ term in the expansion of the potential energy)

If such a potential is present:
- it must have the same periodicity as the simulation box
or
- the periodic boundary conditions must be abandoned

Periodic (Born-von Karman) boundary conditions in 2D and 1D

In some cases, it is not appropriate to employ periodic boundary conditions in each of the three coordinate directions

In the study of surfaces (2D)

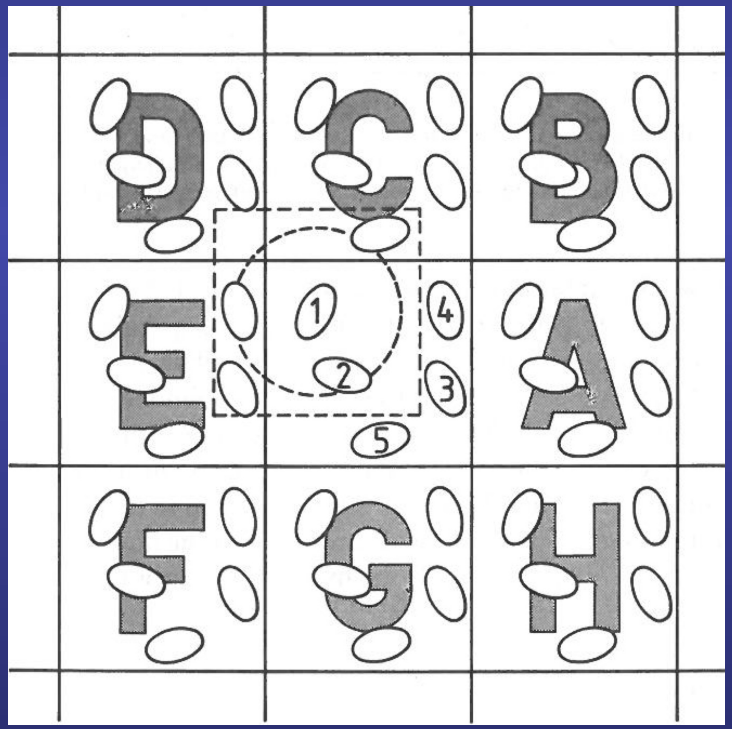
The system is periodic only in the planes parallel to the surface layer

In the study of wires or tubes (1D)

The system is periodic only along the direction of the wire or tube

Calculating properties of systems subject to periodic boundary conditions

Heart of a Molecular Dynamic or Monte Carlo program:
- calculation of the potential energy of a particular configuration
- in Molecular Dynamics, compute the forces acting on all molecules



How would we compute these for molecule 1

We must include interactions between molecule 1 and every other molecule (or periodic image)

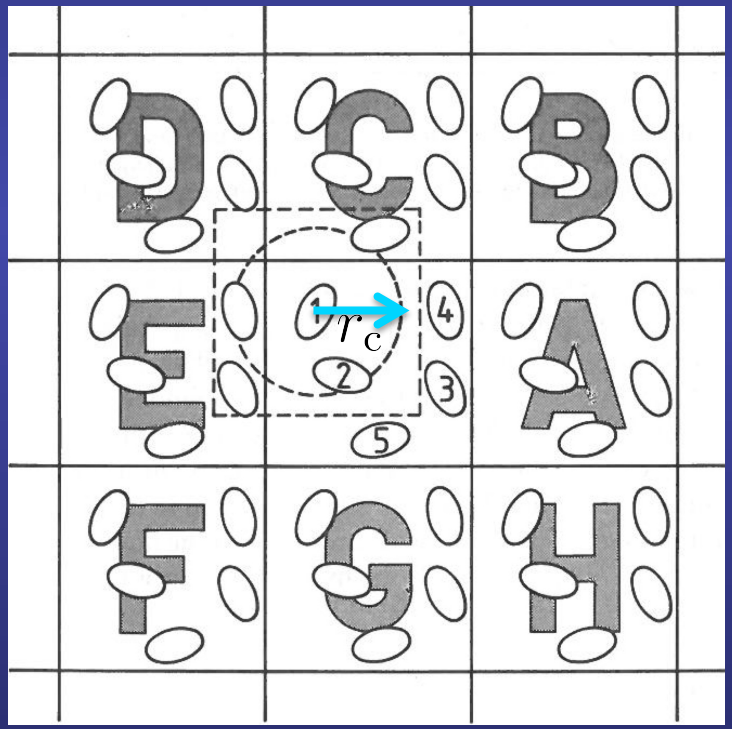
This is an infinite number of terms!!

Impossible to calculate in practice!!

For a short-range potential energy function, we must restrict this summation by making an approximation

Cutting the interactions beyond a given radius to compute the potential energy and forces

The largest contribution to the potential and forces comes from neighbours close to the molecule of interest, and for short-range forces we normally apply a spherical cutoff



That means:

$$v(r) = 0 \qquad r \geq r_c$$

Molecules 2 and 4_E contribute to the force on 1, since their centers lie inside the cutoff
Molecules 3_E and 5_F do not contribute

In a cubic simulation box of side L, the number of neighbours explicitly considered is reduced by a factor of approximately

$$\frac{4\pi r_c^3}{3L^3}$$

The introduction of a spherical cutoff should be a small perturbation, and the cutoff distance should be sufficiently large to ensure this.
Typical distance in a Lennard-Jones system: $r_c = 2.5\sigma$

Difficulties in defining a consistent POTENTIAL in MD method with the truncation of the interatomic pot

The function $v(r_{ij})$ used in a simulation contains a discontinuity at $r_{ij} = r_c$: Whenever a pair of molecules crosses this boundary, the total energy will not be conserved

We can avoid this by shifting the potential function by an amount $v_c = v(r_c)$

$$v^S(r_{ij}) = \begin{cases} v(r_{ij}) - v_c & r_{ij} \leq r_c \\ 0 & r_{ij} > r_c \end{cases}$$

The small addition term is constant for any pair interaction, and does not affect the forces

However, its contribution to the total energy varies from time step to time step, since the total number of pairs within cutoff range varies

Difficulties in defining a consistent FORCE in the MD method with the truncation of the interatomic potent

The force between a pair of molecules is still discontinuous at $r_{ij} = r_c$

For a Lennard-Jones case, the force is given by

$$\vec{f}_{ij} = \begin{cases} \frac{24\varepsilon}{r_{ij}^2}\left[2\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right]\vec{r}_{ij} & r_{ij} \le r_c \\ 0 & r_{ij} > r_c \end{cases}$$

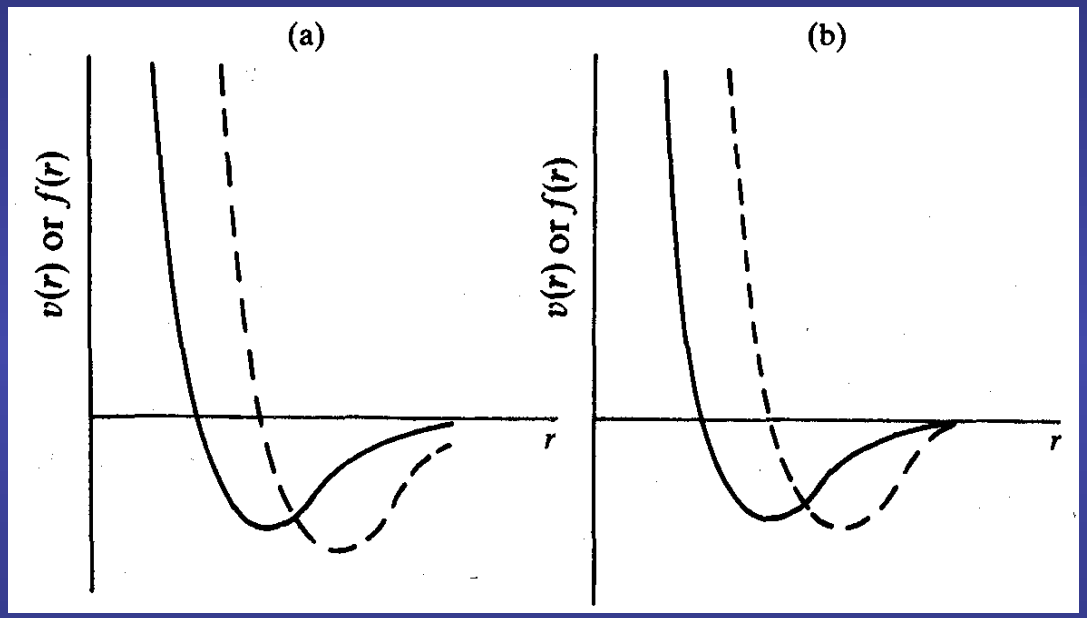
And the magnitude of the discontinuity is $0.039\frac{\varepsilon}{\sigma}$ for $r_c = 2.5\sigma$

It can cause instability in the numerical solution of the differential equations.
To avoid this difficulty, a “shifted force potential” has been introduced

$$v^{SF}(r_{ij}) = \begin{cases} v(r_{ij}) - v_c - \left(\frac{dv(r_{ij})}{dr_{ij}}\right)_{r_{ij}=r_c}(r_{ij} - r_c) & r_{ij} \le r_c \\ 0 & r_{ij} > r_c \end{cases}$$

The discontinuity now appears in the gradient of the force, not in the force itself

Difficulties in defining a consistent FORCE in the MD method with the truncation of the interatomic potent



The difference between the shifted-force potential and the original one means that the simulation no longer corresponds to the desired model liquid

However, the thermodynamic properties with the unshifted potential can be recovered using a simple perturbation scheme

Computer code for periodic boundaries

Let us assume that, initially, the N molecules in the simulation lie within a cubic box of side L, with the origin at its centre.

As the simulation proceeds, these molecules move about the infinite periodic system.

When a molecule leaves the box by crossing one of the boundaries, it is usual to switch the attention to the image molecule entering the box, simply adding or subtracting L from the appropriate coordinate

Computer code for periodic boundaries

```
       SUBROUTINE FOLD ( N, RX, RY, RZ )

C    ********************************************************************
C    ** SUBROUTINE TO FOLD TRAJECTORIES IN PERIODIC BOUNDARIES.       **
C    **                                                               **
C    ** THE FOLDING ROUTINE IS SIMPLY THE USUAL APPLICATION OF        **
C    ** BOUNDARY CONDITIONS.  WE TAKE THE UNIT CUBE AS AN EXAMPLE.    **
C    **                                                               **
C    ** PRINCIPAL VARIABLES:                                          **
C    **                                                               **
C    ** INTEGER N                  NUMBER OF MOLECULES                **
C    ** REAL    RX(N),RY(N),RZ(N) MOLECULAR POSITIONS                 **
C    **                                                               **
C    ** USAGE:                                                        **
C    **                                                               **
C    ** THE ROUTINE IS CALLED FOR EVERY CONFIGURATION GENERATED IN A  **
C    ** SIMULATION.                                                   **
C    ********************************************************************
       INTEGER N
       REAL    RX(N), RY(N), RZ(N)

       INTEGER I
C    ********************************************************************

C
C The function ANINT(X) returns the nearest integer to X,
C converting the result back to type REAL:
C ANINT(-0.49) has the value  0.0
C ANINT(-0.51) has the value -1.0
C
C In this example, we are using reduced units,
C where the length of the box is taken to define the fundamental unit
C of length in the simulation
C
       DO 100 I = 1, N
          RX(I) = RX(I) - ANINT ( RX(I) )
          RY(I) = RY(I) - ANINT ( RY(I) )
          RZ(I) = RZ(I) - ANINT ( RZ(I) )
100    ENDDO

       RETURN
       END
```

Loops in Molecular Dynamic (MD) and Monte Carlo (MC) programs

Loop on all the molecules (from $1 \leq i \leq N$)

For a given molecule i , loop over all molecules j to calculate the minimum image separation r_{ij}

If molecules are separated by distances smaller than the potential cutoff

$$r_{ij} \leq r_{\mathrm{c}}$$

Compute potential energy and forces

If molecules are separated by distances greater than the potential cutoff

$$r_{ij} > r_{\mathrm{c}}$$

Skip to the end of the inner loop, avoiding expensive calculations

Time required to examine all pair separations is proportional to N^2

Neighbour lists: improving the speed of a program

Inner loops of the MD and MC programs scale proportional to N^2

Verlet: maintain a list of the neighbours of a particular molecule, which is updated at intervals

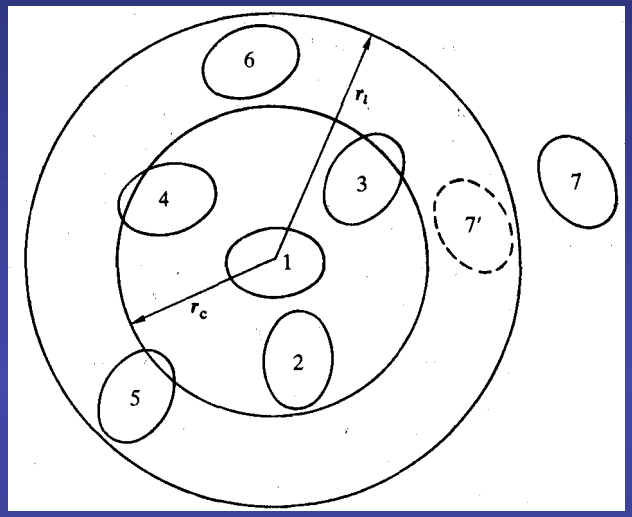
Between updates, the program does not check through all the molecules, but just those appearing on the list

The number of pais separations explicitly considered is reduced.
This saves time in:

- Looping through j
- Minimum imaging,
- Calculating r_{ij}^2
- Checking against the cutoff

The Verlet neighbour list

The potential cutoff sphere, of radius r_c , around a particular molecule is surrounded by a **"skin"**, to give a **larger sphere of radius** r_l



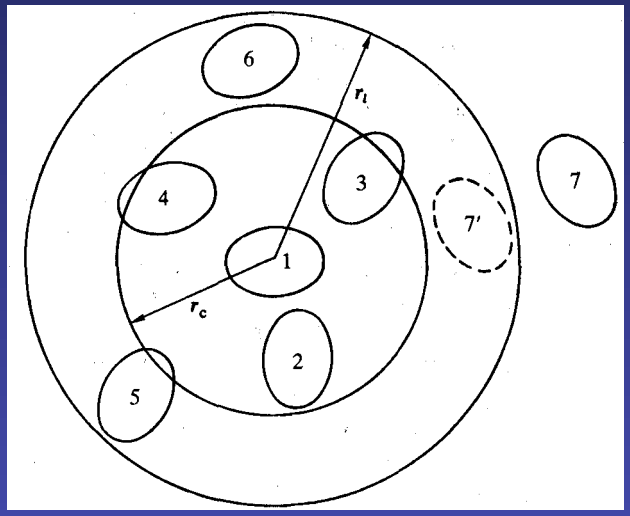
At first step, a list is constructed of all the neighbours of each molecule, for which the pair separation is within r_l

These neighbours are stored in a large one-dimensional array, LIST
The dimension of LIST is roughly $4\pi r_l^3 \rho N/6$

At the same time, a second indexing array of size N, POINT, is constructed:

- POINT (I) points to the position in the array LIST where the first neighbour of molecule I can be found.
- Since POINT(I+1) points to the first neighbour of molecule I+1, then POINT(I+1)-1 points to the last neighbour of molecule I.
- Thus, using POINT, we can readily identify the part of the last LIST array which contains neighbours of I.

The Verlet neighbour list



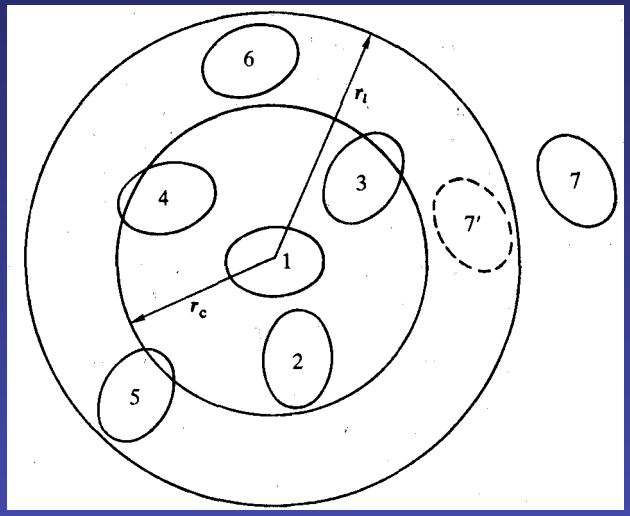
Over the next few steps, the list is used in force/energy evaluation routine

For each molecule I, the program identifies the neighbours J, by running over LIST from point to POINT(I) to POINT(I+1)-1

It is essential to check that POINT(I+1) is actually greater then POINT(I). If it is not the case, then molecule I has no neighbours

From time to time, the neighbour list is reconstructed and the cycle is repeated.

The Verlet neighbour list



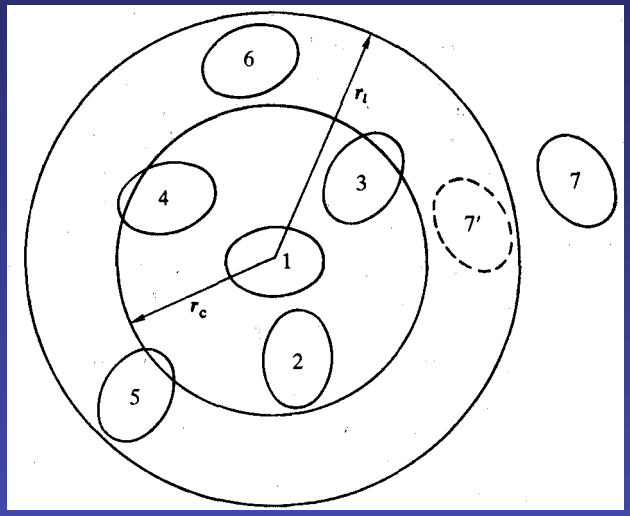
Over the next few steps, the list is used in force/energy evaluation routine

For each molecule I, the program identifies the neighbours J, by running over LIST from point to POINT(I) to POINT(I+1)-1

It is essential to check that POINT(I+1) is actually greater then POINT(I). If it is not the case, then molecule I has no neighbours

From time to time, the neighbour list is reconstructed and the cycle is repeated.

The Verlet neighbour list



The algorithm is successful if the skin around r_c is chosen to be thick enough so that between reconstructions:

A molecule, such as 7, which is not on the list of molecule 1, cannot penetrate through the skin into the important r_c sphere

Molecules, such as 3 and 4 can move in and out of this sphere, but since they are on the list of molecule 1, they are always considered until the list is next updated.

Parameters of the Verlet neighbour list: the interval between updates

Often fixed at the beginning of the program

Intervals of 10-20 steps are quite common

An important refinement: allow the program to update the list automatically:

- When the list is constructed, a vector for each molecule is set to zero

- At subsequent steps, the vector is incremented with the displacement of each molecule.

- Thus, it stores, the displacement of each molecule since the last update

- When the sum of the magnitudes of the two largest displacements exceeds $r_c - r_l$, the neighbour list should be updated again

Parameters of the Verlet neighbour list: the list sphere radius r_l

Is a parameter that we are free to choose

As r_l is increased, the frequency of updates of the neighbour list will decrease

However, with a large list, the efficiency of the non-updated steps will decrease

Table 5.1 *Time saving using a Verlet neighbour list* [*Thompson* 1983].

List Radius	Update[a] interval	Time[b] $N = 256$	Time[b] $N = 500$
no list	—	3.33	10.00
2.60	5.78	2.24	4.93
2.70	12.50	2.17	4.55
2.90	26.32	2.28	4.51
3.10	43.48	2.47	4.79
3.43	83.33	2.89	—
3.50	100.00	—	5.86

[a]Update interval is the average number of steps between updates. It is essentially independent of system size.
[b]Time is CPU time per step, in seconds. The runs were performed on a PDP 11/70.

The larger the system, the more dramatic the improvement

$$r_c = 2.5\sigma$$

$$\rho^* = 0.8 \qquad T^* = 0.76$$

Algorithm to compute the Verlet neighbour list

```
        RLSTSQ = RLIST * RLIST

        NLIST = 0

        DO 100 I = 1, N - 1

          POINT(I) = NLIST + 1

          RXI      = RX(I)
          RYI      = RY(I)
          RZI      = RZ(I)

          DO 99 J = I + 1, N

             RXIJ  = RXI - RX(J)
             RYIJ  = RYI - RY(J)
             RZIJ  = RZI - RZ(J)

             RXIJ  = RXIJ - ANINT ( RXIJ )
             RYIJ  = RYIJ - ANINT ( RYIJ )
             RZIJ  = RZIJ - ANINT ( RZIJ )
             RIJSQ = RXIJ * RXIJ + RYIJ * RYIJ + RZIJ * RZIJ

             IF ( RIJSQ .LT. RLSTSQ ) THEN

                NLIST = NLIST + 1
                LIST(NLIST) = J

             ENDIF
99         ENDDO

100     ENDDO
        POINT(N) = NLIST + 1
```

Algorithm to check whether the update of the list is required

```
      SUBROUTINE CHECK ( RCUT, RLIST, UPDATE )

      COMMON / BLOCK1 / RX, RY, RZ, FX, FY, FZ
      COMMON / BLOCK3 / RX0, RY0, RZ0

C     ********************************************************************
C     ** DECIDES WHETHER THE LIST NEEDS TO BE RECONSTRUCTED.            **
C     **                                                                **
C     ** PRINCIPAL VARIABLES:                                           **
C     **                                                                **
C     ** REAL     RX(N),RY(N),RZ(N)     ATOM POSITIONS                 **
C     ** REAL     RX0(N),RY0(N),RZ0(N)  COORDINATES AT LAST UPDATE     **
C     ** REAL     RLIST                 RADIUS OF VERLET LIST          **
C     ** REAL     RCUT                  CUTOFF DISTANCE FOR FORCES     **
C     ** REAL     DISPMX                LARGEST DISPLACEMENT           **
C     ** INTEGER  N                     NUMBER OF ATOMS                **
C     ** LOGICAL  UPDATE                IF TRUE THE LIST IS UPDATED    **
C     **                                                                **
C     ** USAGE:                                                         **
C     **                                                                **
C     ** CHECK IS CALLED TO SET UPDATE BEFORE EVERY CALL TO FORCE.      **
C     ********************************************************************

      INTEGER     N
      PARAMETER ( N = 108 )

      REAL        RX(N), RY(N), RZ(N), FX(N), FY(N), FZ(N)
      REAL        RX0(N), RY0(N), RZ0(N)
      REAL        RCUT, RLIST
      LOGICAL     UPDATE

      REAL        DISPMX
      INTEGER     I

C     ********************************************************************

C     ** CALCULATE MAXIMUM DISPLACEMENT SINCE LAST UPDATE **

      DISPMX = 0.0

      DO 30 I = 1, N

         DISPMX = MAX ( ABS ( RX(I) - RX0(I) ), DISPMX )
         DISPMX = MAX ( ABS ( RY(I) - RY0(I) ), DISPMX )
         DISPMX = MAX ( ABS ( RZ(I) - RZ0(I) ), DISPMX )

30    CONTINUE

C     ** A CONSERVATIVE TEST OF THE LIST SKIN CROSSING **

      DISPMX = 2.0 * SQRT  ( 3.0 * DISPMX ** 2 )

      UPDATE = ( DISPMX .GT. ( RLIST - RCUT ) )

      RETURN
      END
```

Algorithm to save the list for future checkings

```
      SUBROUTINE SAVE

      COMMON / BLOCK1 / RX, RY, RZ, FX, FY, FZ
      COMMON / BLOCK3 / RX0, RY0, RZ0

C     ********************************************************************
C     ** SAVES CURRENT CONFIGURATION FOR FUTURE CHECKING.                **
C     **                                                                  **
C     ** PRINCIPAL VARIABLES:                                            **
C     **                                                                  **
C     ** REAL     RX(N),RY(N),RZ(N)     ATOM POSITIONS                  **
C     ** REAL     RX0(N),RY0(N),RZ0(N)  STORAGE LOCATIONS FOR SAVE      **
C     ** INTEGER  N                     NUMBER OF ATOMS                 **
C     **                                                                  **
C     ** USAGE:                                                          **
C     **                                                                  **
C     ** SAVE IS CALLED WHENEVER THE NEW VERLET LIST IS CONSTRUCTED.     **
C     ********************************************************************

      INTEGER     N
      PARAMETER ( N = 108 )

      REAL        RX(N), RY(N), RZ(N), FX(N), FY(N), FZ(N)
      REAL        RX0(N), RY0(N), RZ0(N)

      INTEGER     I

C     ********************************************************************

      DO 100 I = 1, N

         RX0(I) = RX(I)
         RY0(I) = RY(I)
         RZ0(I) = RZ(I)

100   CONTINUE

      RETURN
      END
```

Algorithm of the inner loop to compute forces and potential energy

```
        DO 10 I = 1, N
           FX(I) = 0.0
           FY(I) = 0.0
           FZ(I) = 0.0
10      ENDDO
        V = 0.0
        W = 0.0
C       ** USE THE LIST TO FIND THE NEIGHBOURS **
        DO 200 I = 1, N - 1
           JBEG = POINT(I)
           JEND = POINT(I+1) - 1
        ** CHECK THAT ATOM I HAS NEIGHBOURS **
           IF( JBEG .LE. JEND ) THEN
              RXI  = RX(I)
              RYI  = RY(I)
              RZI  = RZ(I)
              FXI  = FX(I)
              FYI  = FY(I)
              FZI  = FZ(I)
              DO 199 JNAB = JBEG, JEND
                 J     = LIST(JNAB)
                 RXIJ  = RXI - RX(J)
                 RYIJ  = RYI - RY(J)
                 RZIJ  = RZI - RZ(J)
                 RXIJ  = RXIJ - ANINT( RXIJ )
                 RYIJ  = RYIJ - ANINT( RYIJ )
                 RZIJ  = RZIJ - ANINT( RZIJ )
                 RIJSQ = RXIJ * RXIJ + RYIJ * RYIJ + RZIJ * RZIJ
                 IF ( RIJSQ .LT. RCUTSQ ) THEN
                    SR2   = SIGSQ / RIJSQ
                    SR6   = SR2 * SR2 * SR2
                    VIJ   = SR6 * ( SR6 - 1.0 )
                    WIJ   = SR6 * ( SR6 - 0.5 )
                    V     = V + VIJ
                    W     = W + WIJ
                    FIJ   = WIJ / RIJSQ
                    FXIJ  = RXIJ * FIJ
                    FYIJ  = RYIJ * FIJ
                    FZIJ  = RZIJ * FIJ
                    FXI   = FXI + FXIJ
                    FYI   = FYI + FYIJ
                    FZI   = FZI + FZIJ
                    FX(J) = FX(J) - FXIJ
                    FY(J) = FY(J) - FYIJ
                    FZ(J) = FZ(J) - FZIJ
                 ENDIF
199           ENDDO
              FX(I) = FXI
              FY(I) = FYI
              FZ(I) = FZI
           ENDIF
200     ENDDO
        V =  4.0 * V
        W = 48.0 * W / 3.0
        DO 300 I = 1, N
           FX(I) = 48.0 * FX(I)
           FY(I) = 48.0 * FY(I)
           FZ(I) = 48.0 * FZ(I)
300     ENDDO
```

Long range forces

By definition, a long range force is defined as one in which the spatial interaction falls off no faster than r^{-d} where d is the dimensionality of the system

Charge-charge interaction between ions $v^{zz}(r) \sim r^{-1}$

Dipole-dipole interaction between molecules $v^{\mu\mu}(r) \sim r^{-3}$

Their range is greater than half the box length for a typical simulation

Brute force solution: increase the length of the central box L to hundrehts of nanometers, so that the screening by neighbours would diminish the effective range of the potentials

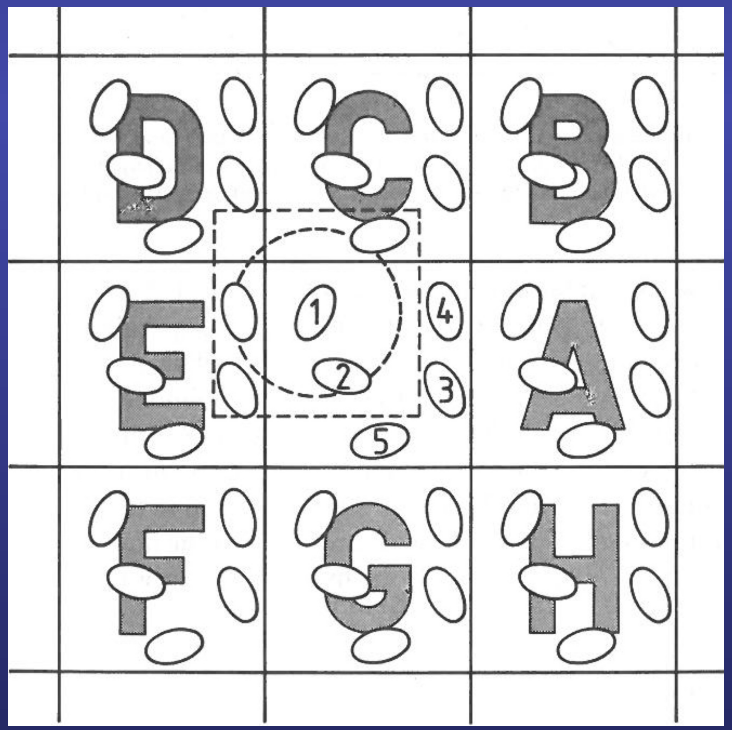
Impractical

Computational cost scales as N^2, i.e. as L^6

Statement of the problem when dealing with long-range forces

Ion 1 interacts with ions 2, 2_A, 2_B, and all the other images of 2

$$\mathcal{V}^{zz} = \frac{1}{2} \sum_{\vec{n}}{}' \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \frac{z_i z_j}{|\vec{r}_{ij} + \vec{n}|} \right)$$



z_i and z_j are the charges

For simplicity, we are ommiting all the factors $4\pi\varepsilon_0$

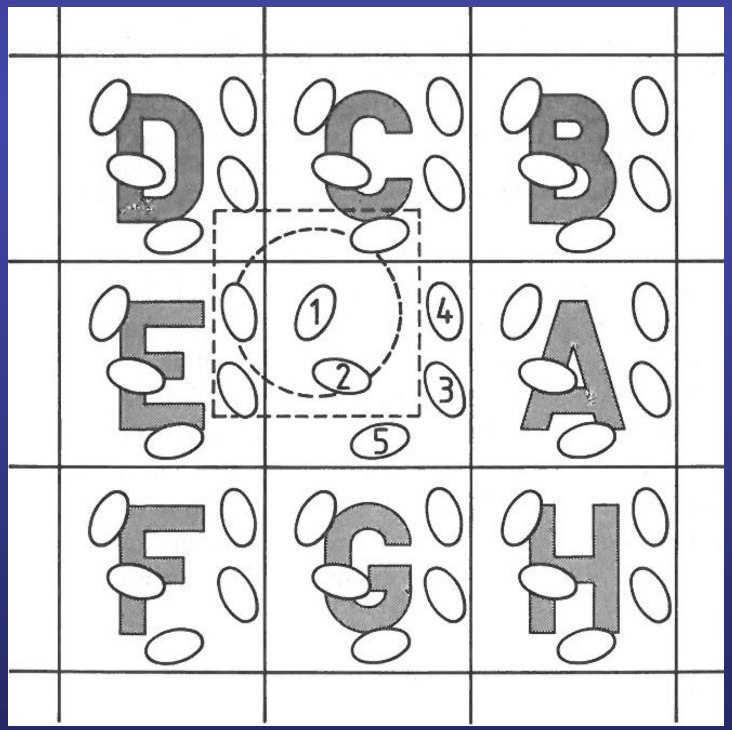
The sum over $\vec{n}$ is the sum over all simple cubic lattice points, $\vec{n} = (n_x L, n_y L, n_z L)$ with n_x, n_y, n_z integers

The prime indicates that we omit $i = j$ for $\vec{n} = 0$

Statement of the problem when dealing with long-range forces

Ion 1 interacts with ions 2, 2_A, 2_B, and all the other images of 2

$$\mathcal{V}^{zz} = \frac{1}{2} \sum_{\vec{n}}{}' \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \frac{z_i z_j}{|\vec{r}_{ij} + \vec{n}|} \right)$$



For long-range potentials, this sum is conditionally convergent, i. e. the results depends on the order in which we add up terms

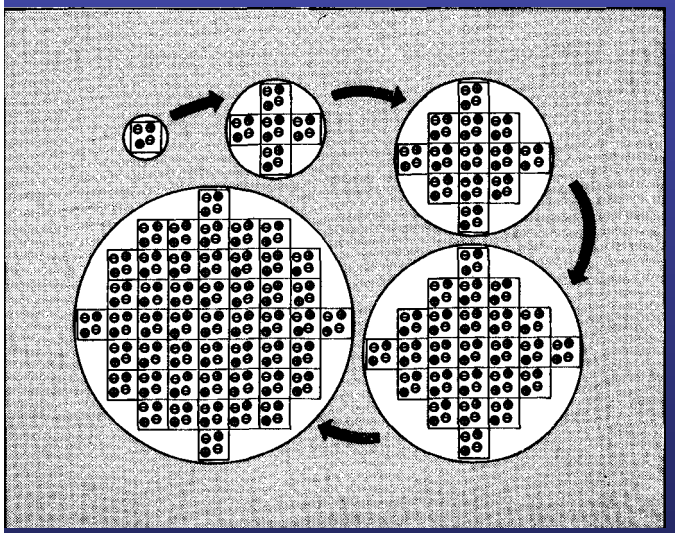
As we shall see below, the Ewald method evaluate this energy by transforming it into summations that converges not only rapidly but also absolutely

The infinite sum is conditionally convergent

$$\mathcal{V}^{zz} = \frac{1}{2} \sum_{\vec{n}}{}' \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \frac{z_i z_j}{|\vec{r}_{ij} + \vec{n}|} \right)$$

For long-range potentials, this sum is conditionally convergent, i. e. the results depends on the order in which we add up terms

Natural choice:
Take boxes in order of their proximity to the central box



First term $|\vec{n}| = 0 \qquad \vec{n} = (0,0,0)$

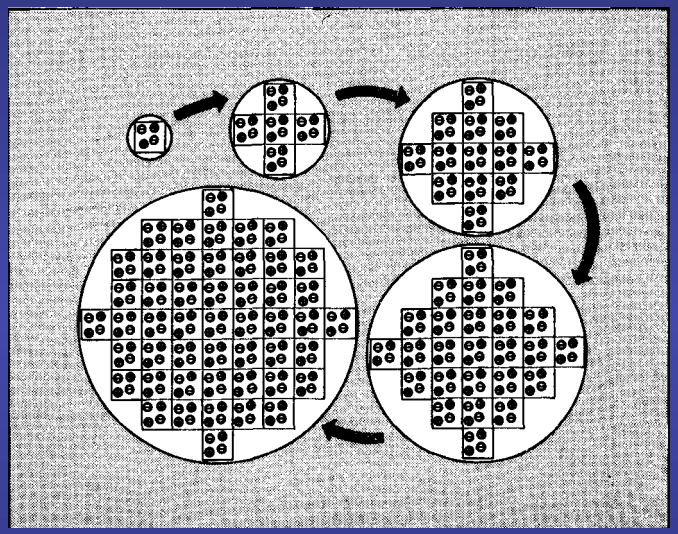
Second term $|\vec{n}| = L \quad \vec{n} = (\pm L, 0, 0), (0, \pm L, 0), (0, 0, \pm L)$

As we add further terms to the sum, we are building up our infinite system in roughly spherical layers

In vacuum, the sphere has a dipolar layer on its surface, and there would be a jump in the potential energy

The infinite sum is conditionally convergent

$$\mathcal{V}^{zz} = \frac{1}{2} \sum_{\vec{n}}{}' \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \frac{z_i z_j}{|\vec{r}_{ij} + \vec{n}|} \right)$$

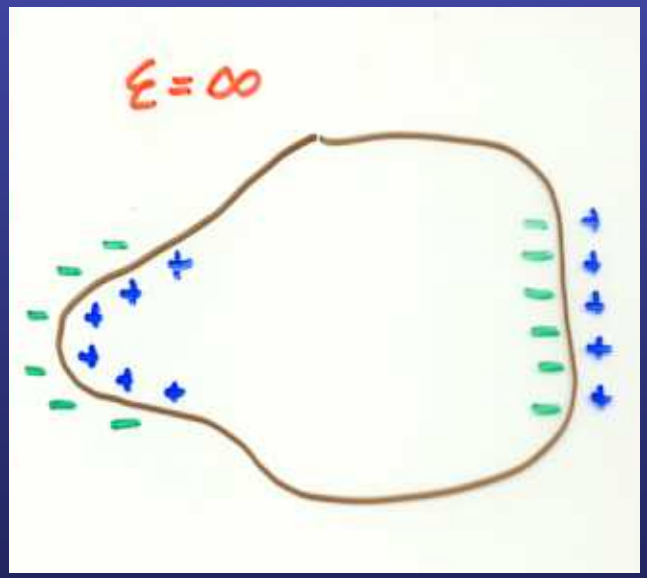


Good conductor
(perfect metal)

$$\varepsilon_s = \infty$$

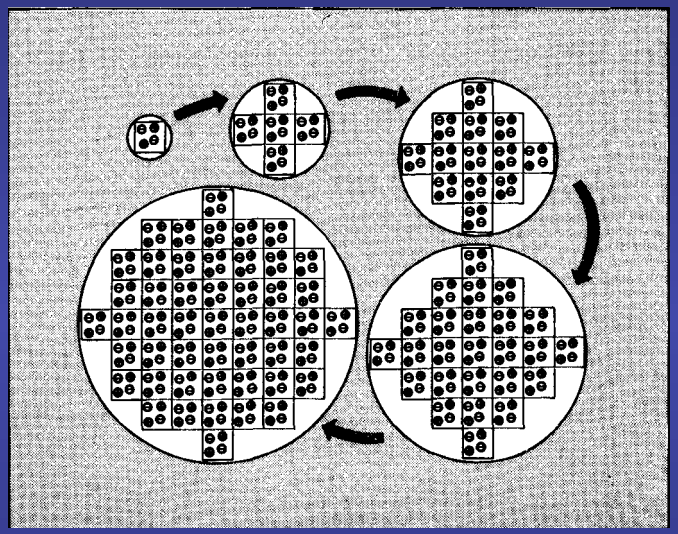
For a sphere in a conductor, there is no such layer (al the surface charges are screened)

We must specify the nature of the medium surrounding the sphere, in particular, its dieletric permittivity (the dielectric constant) ε_s



The infinite sum is conditionally convergent

$$\mathcal{V}^{zz} = \frac{1}{2}\sum_{\vec{n}}{}' \left(\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{z_i z_j}{|\vec{r}_{ij}+\vec{n}|}\right)$$



We must specify the nature of the medium surrounding the sphere, in particular, its dieletric permittivity (the dielectric constant) ε_s

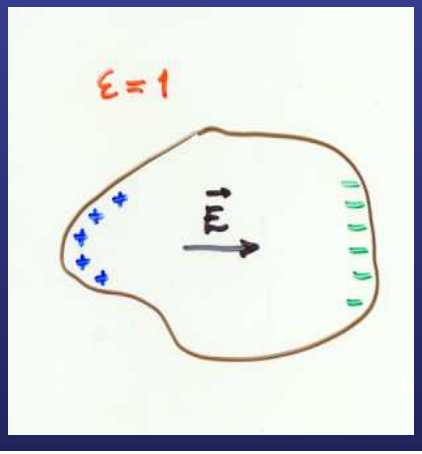
Vacuum

$$\varepsilon_s = 1$$

In vacuum, the sphere has a dipolar layer on its surface

$$\mathcal{E}_{\mathrm{d}} = -4\pi n P$$

For a sphere, the depolarization factor $n = \frac{1}{3}$

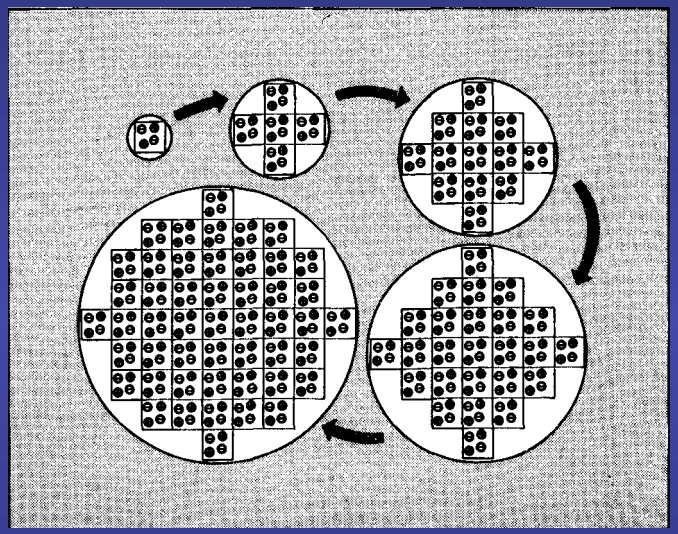


And the associated energy

$$\delta\mathcal{V} = -\frac{\Omega}{2}\vec{P}\cdot\vec{\mathcal{E}}_{\mathrm{d}} = \frac{2\pi\Omega}{3}\vec{P}^2 = \frac{2\pi\Omega}{3}\left(\frac{|\sum_{i=1}^{N} z_i\vec{r}_i|}{\Omega}\right)^2 = \frac{2\pi}{3\Omega}|\sum_{i=1}^{N} z_i\vec{r}_i|^2$$

The infinite sum is conditionally convergent

$$\mathcal{V}^{zz} = \frac{1}{2}\sum_{\vec{n}}{}' \left(\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{z_i z_j}{|\vec{r}_{ij}+\vec{n}|}\right)$$



We must specify the nature of the medium surrounding the sphere, in particular, its dieletric permittivity (the dielectric constant) ε_s

Good conductor (perfect metal)

$$\varepsilon_s = \infty$$

For a sphere in a conductor, there is no such layer (al the surface charges are screened)

Vacuum

$$\varepsilon_s = 1$$

In vacuum, the sphere has a dipolar layer on its surface

$$\mathcal{V}_{zz}(\varepsilon_s = \infty) = \mathcal{V}_{zz}(\varepsilon_s = 1) - \frac{2\pi}{3L^3}\left|\sum_i z_i \vec{r}_i\right|^2$$

Equation applies in the limit of very large sphere boxes

Splitting the charge distribution

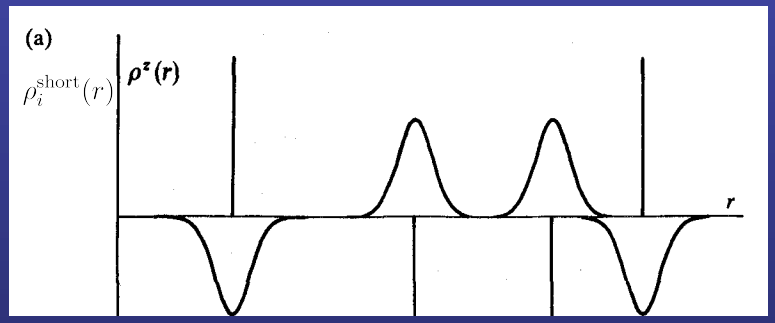
In the original problem, the charge distributions are described by delta functions. We can split it into two terms, adding a subtracting a Gaussian distribution

$$\rho_i^z(\vec{r}) = \frac{z_i \kappa^3}{\pi^{3/2}} \exp\left(-\kappa^2 r^2\right)$$

κ **Determines the width of the distribution**

$\vec{r}$ **Position relative to the center of the dist**

$$\begin{aligned}
\rho_i(\vec{r}) &= z_i \delta(\vec{r} - \vec{r}_i) + \rho_i^z(\vec{r}) - \rho_i^z(\vec{r}) \\
&= \left[z_i \delta(\vec{r} - \vec{r}_i) - \rho_i^z(\vec{r})\right] + \rho_i^z(\vec{r}) \\
&= \rho_i^{\text{short}}(\vec{r}) + \rho_i^{\text{long}}(\vec{r})
\end{aligned}$$



In what we call $\rho_i^{\text{short}}(r)$

Each point charge is surrounded by a charge distribution of equal magnitude and opposite sign, which spreads out radially from the charge.

The screened charge density produces a short-range singular potential that can be summed in real space

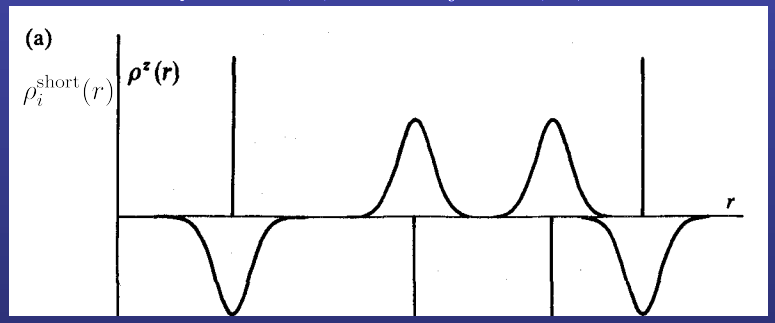
Each point charge is surrounded by a charge distribution of equal magnitude and opposite sign, which spreads out radially from the charge.
The distribution is conveniently taken to be Gaussian

$$\rho_i^z(\vec{r}) = \frac{z_i \kappa^3}{\pi^{3/2}} \exp\left(-\kappa^2 r^2\right)$$

κ Determines the width of the distribution

$\vec{r}$ Position relative to the center of the dist

$$\begin{aligned}\rho_i(\vec{r}) &= z_i\delta(\vec{r}-\vec{r}_i) + \rho_i^z(\vec{r}) - \rho_i^z(\vec{r}) \\ &= \left[z_i\delta(\vec{r}-\vec{r}_i) - \rho_i^z(\vec{r})\right] + \rho_i^z(\vec{r}) \\ &= \rho_i^{\text{short}}(\vec{r}) + \rho_i^{\text{long}}(\vec{r})\end{aligned}$$

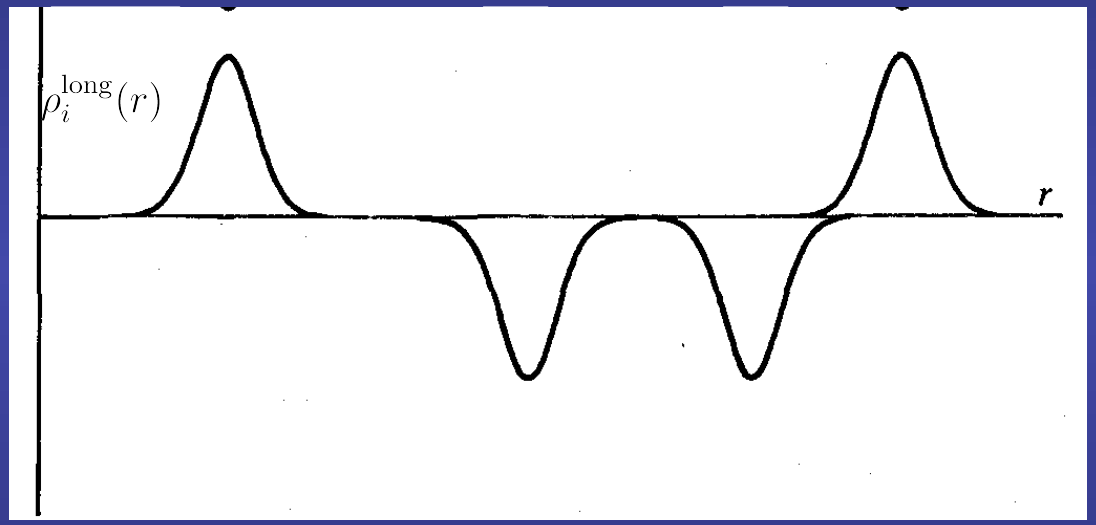


This distribution acts like an ionic atmosphere to screen the interaction between neighboring charges, that are now short ranged and singular

The total screened potential is calculated by summing over all the molecules in the central cube and all their images in the real space lattice of image boxes

The cancelling distribution produces a long-range no-singular potential that can be summed in reciprocal space

A charge distribution of the same sign as the original charge, and the same shape distribution $\rho_i^z(\vec{r})$ is also added



This cancelling distribution reduces the overall potential to that due to the original set of charges

The contribution from this cancelling charges is summed in reciprocal space

The Ewald sum: a technique for efficiently summing the interaction between an ion an all it periodic images

The final potential energy will contain a sum in real space plus a reciprocal space sum minus a self-energy term plus the surface term

$$\mathcal{V}^{zz}(\varepsilon_s = 1) = \boxed{\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\sum_{|\vec{n}|=0}^{\infty}{}' \frac{z}{i}\, z_j \frac{\operatorname{erfc}(\kappa|\vec{r}_{ij}+\vec{n}|)}{|\vec{r}_{ij}+\vec{n}|}\right.}$$

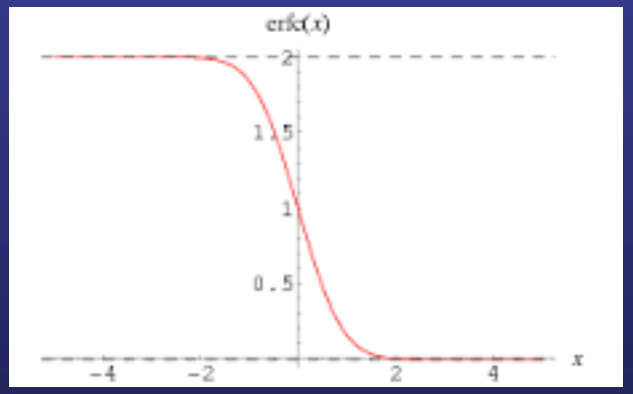
$$\left. + \left(\frac{1}{\pi L^3}\right)\sum_{\vec{k}\neq 0} z_i z_j \left(\frac{4\pi^2}{k^2}\right)\exp\left(-\frac{k^2}{4\kappa^2}\right)\cos(\vec{k}\cdot\vec{r}_{ij})\right)$$

$$- \left(\frac{\kappa}{\sqrt{\pi}}\right)\sum_{i=1}^{N} z_i^2 + \left(\frac{2\pi}{3L^3}\right)|\sum_{i=1}^{N} z_i \vec{r}_i|^2$$

A complete derivation can be found in the excellent notes by H. Lee and W. Cai
http://micro.stanford.edu/mediawiki/images/4/46/Ewald_notes.pdf

$\operatorname{erfc}(x)$ **Is the complementary error function**

Falls to zero with increasing x



If κ **is chosen to be large enough, the only terms which contributes to the sum in real space is that with** $n = 0$ **(the first term reduces to the minimum image)**

The Ewald sum: a technique for efficiently summing the interaction between an ion an all it periodic images

The final potential energy will contain a sum in real space plus a reciprocal space sum minus a self-energy term plus the surface term

$$\mathcal{V}^{zz}(\varepsilon_s = 1) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\sum_{|\vec{n}|=0}^{\infty}{}' z_i z_j \frac{\mathrm{erfc}(\kappa|\vec{r}_{ij}+\vec{n}|)}{|\vec{r}_{ij}+\vec{n}|}\right.$$

$$\left. + \left(\frac{1}{\pi L^3}\right)\sum_{\vec{k}\neq 0} z_i z_j \left(\frac{4\pi^2}{k^2}\right)\exp\left(-\frac{k^2}{4\kappa^2}\right)\cos(\vec{k}\cdot\vec{r}_{ij})\right)$$

$$- \left(\frac{\kappa}{\sqrt{\pi}}\right)\sum_{i=1}^{N} z_i^2 + \left(\frac{2\pi}{3L^3}\right)|\sum_{i=1}^{N} z_i \vec{r}_i|^2$$

Sum over reciprocal lattice vectors $\vec{k} = \dfrac{2\pi\vec{n}}{L}$

A large value of κ corresponds to a shape distribution of charge, so we need to include many terms in the k point summation to model it.

Extension of the Ewald sum to dipolar system

z_i **is replaced by** $\vec{\mu}_i \cdot \nabla_{\vec{r}_i}$

$$\mathcal{V}^{\mu\mu}(\varepsilon_s = 1) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\sum_{|\vec{n}|=0}^{\infty}{}'(\vec{\mu}_i\cdot\vec{\mu}_j)B(\vec{r}_{ij}+\vec{n}) - (\vec{\mu}_i\cdot\vec{r}_{ij})(\vec{\mu}_j\cdot\vec{r}_{ij})C(\vec{r}_{ij}+\vec{n})\right.$$

$$\left. + \left(\frac{1}{\pi L^3}\right)\sum_{\vec{k}\neq 0}(\vec{\mu}_i\cdot\vec{k})(\vec{\mu}_j\cdot\vec{k})\left(\frac{4\pi^2}{k^2}\right)\exp\left(-\frac{k^2}{4\kappa^2}\right)\cos(\vec{k}\cdot\vec{r}_{ij})\right)$$

$$- \sum_{i=1}^{N}\frac{2\kappa^3\mu_i^2}{3\pi^{1/2}} + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\frac{4\pi}{3L^3}\right)\vec{\mu}_i\cdot\vec{\mu}_j$$

Sum on i **and** j **are for dipoles in the central box**
Factor $1/(4\pi\varepsilon_0)$ **is omitted**

$$B(r) = \mathrm{erfc}(\kappa r)/r^3 + \left(\frac{2\kappa}{\sqrt{\pi}}\right)\exp(-\kappa^2 r^2)/r^2$$

$$C(r) = 3\mathrm{erfc}(\kappa r)/r^5 + \left(\frac{2\kappa}{\sqrt{\pi}}\right)\left(2\kappa^2 + \frac{3}{r^2}\right)\exp(-\kappa^2 r^2)/r^2$$