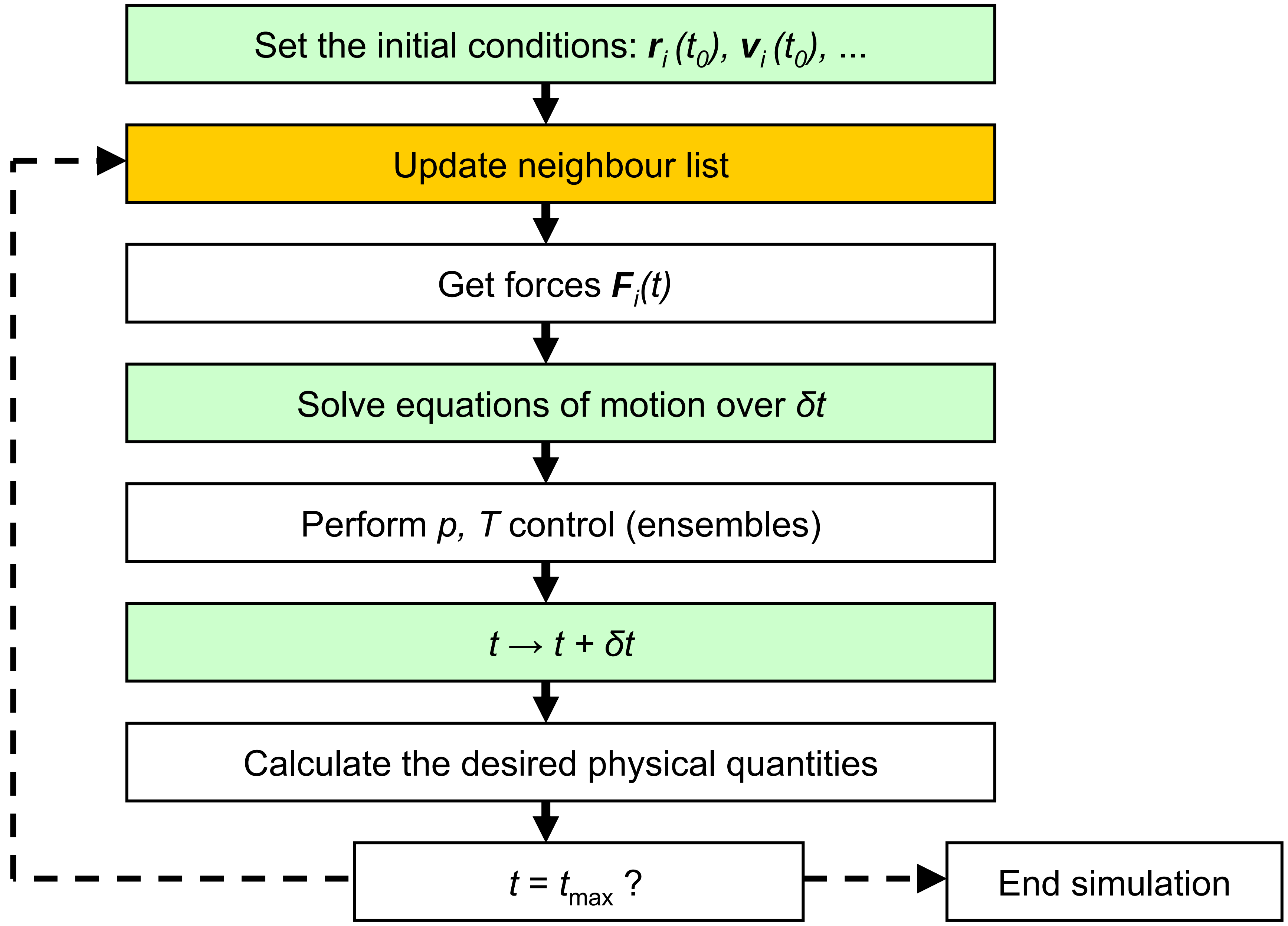
4: CUT-OFFS AND NEIGHBOUR LISTS


Set the initial conditions: $\boldsymbol{r}_i(t_0)$, $\boldsymbol{v}_i(t_0)$, ...
Update neighbour list
Get forces $\boldsymbol{F}_i(t)$
Solve equations of motion over δt
Perform p, T control (ensembles)
$t \rightarrow t + \delta t$
Calculate the desired physical quantities
$t = t_{\max}$?
End simulation

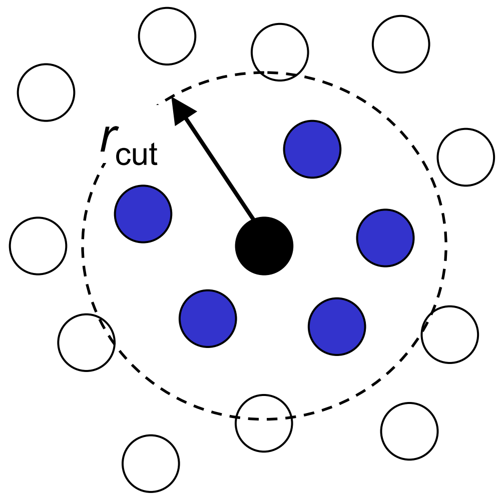
Cut-off radius

An important factor for the performance of an MD simulation code is its scalability with the number of simulated particles N.

If we calculated all the atomic interactions, we would have to carry out ½ N (N – 1) force calculations. (Not good!)

Fortunately, some interatomic forces decrease strongly with distance (van der Waals, covalent interactions etc.) We can thus limit the interactions to be considered within a certain distance, the cut-off radius r_{cut}.

Setting a certain cut-off radius in an interatomic potential requires us also to ensure that the interactions considered produce the correct cohesive energies etc.

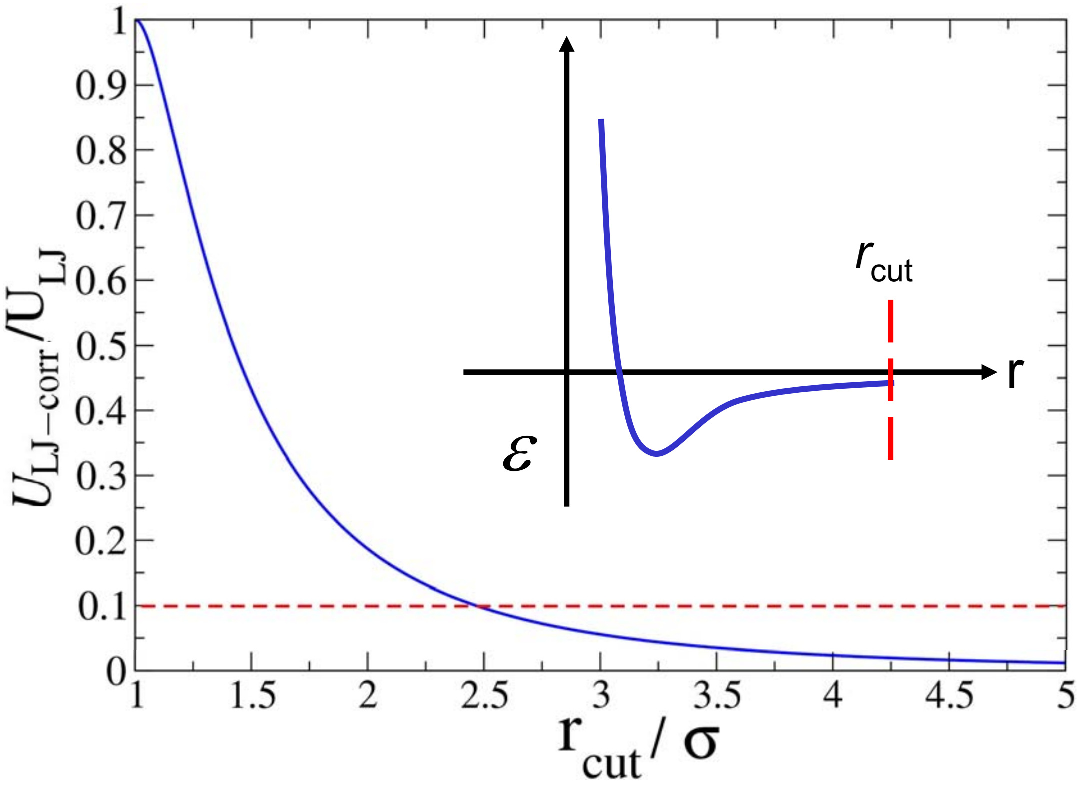


Truncation of the LJ-potential

Let us consider the 12-6 Lennard-Jones potential.

The most straightforward way of doing the cut-off is the simple truncation at $r = r_{\text{cut}}$

This induces an error in the energy calculations which can be taken care of with a correction term:



$$U_{\text{LJ-corr}} = -\frac{8}{9}\pi\rho\sigma^3\varepsilon\left[\left(\frac{\sigma}{r_{\text{cut}}}\right)^9 - 3\left(\frac{\sigma}{r_{\text{cut}}}\right)^3\right]$$

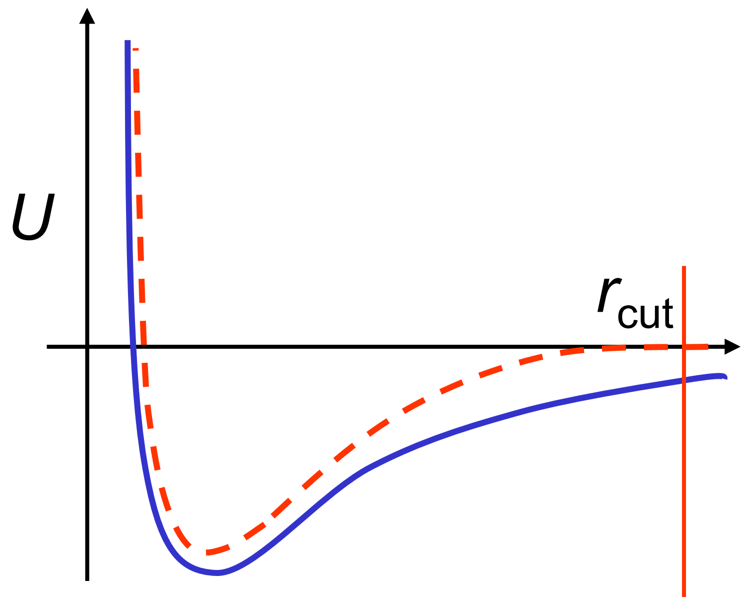
assuming homogeneous density ρ

For cut-off radii > 2.5σ the error in energy is less than 10%

Cut-off methods (1)

Simple truncation is not acceptable: discontinuity in the potential at the cut-off distance means an infinite force for an atom pair crossing the discontinuity!

MD algorithms cannot deal with this, resulting in poor energy conservation.



So what we need to do is modify the potential somehow, so that the potential and force are continuous at the cut-off distance; e.g. by shifting

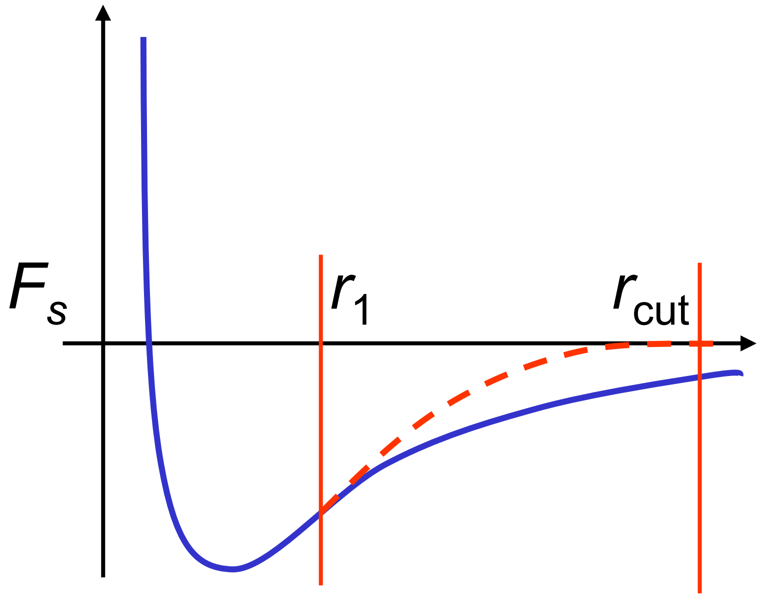
$$u_{sf}(r) = \begin{cases} u(r) - \underbrace{u(r_c)}_{\text{energy}} - \underbrace{\frac{du}{dr}(r - r_c)}_{\text{force}} & r \le r_c \\ 0 & r > r_c \end{cases}$$

Note that if we do the cut-off for an already parametrized potential we have to ensure that the potential still has the desired properties.

Cut-off methods (2)

Another way is to ”switch off” the potential between a chosen distance r_1 and r_{cut}

$$F_s(r) = \begin{cases} F_\alpha(r), & \text{if} \quad r < r_1 \\ F_\alpha(r) + S(r), & \text{if} \quad r_1 < r < r_{\text{cut}} \\ 0, & \text{if} \quad r_{\text{cut}} \le r \end{cases}$$



With the conditions for the switching function

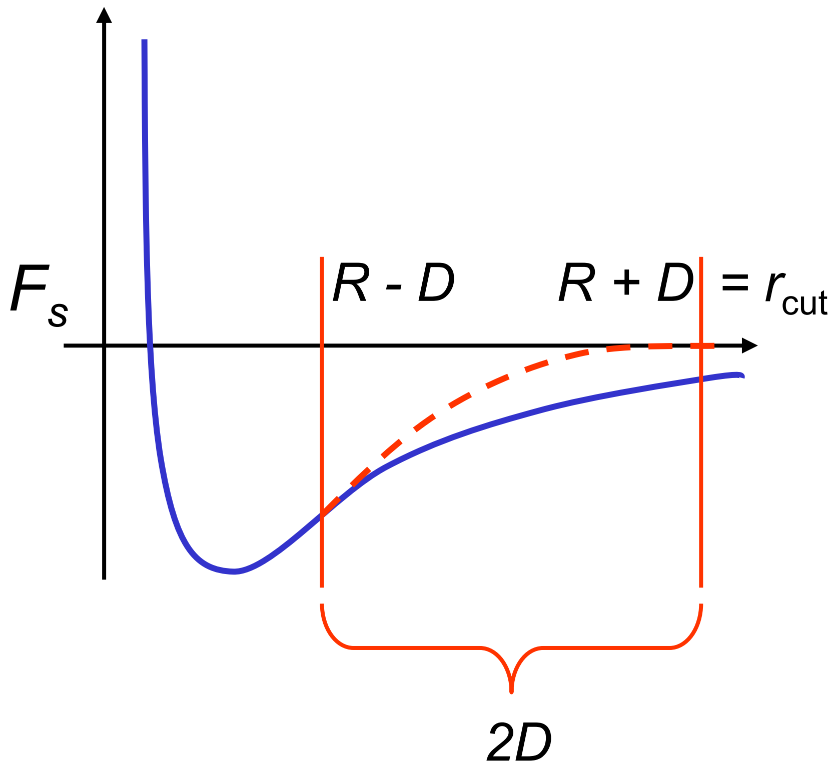
$$\begin{aligned} S(r_1) &= 0 \\ S'(r_1) &= 0 \\ S(r_{\text{cut}}) &= -F_\alpha(r_{\text{cut}}) \\ S'(r_{\text{cut}}) &= F'_\alpha(r_{\text{cut}}) \end{aligned}$$

S can be a polynomial, based on trigonometric functions etc.

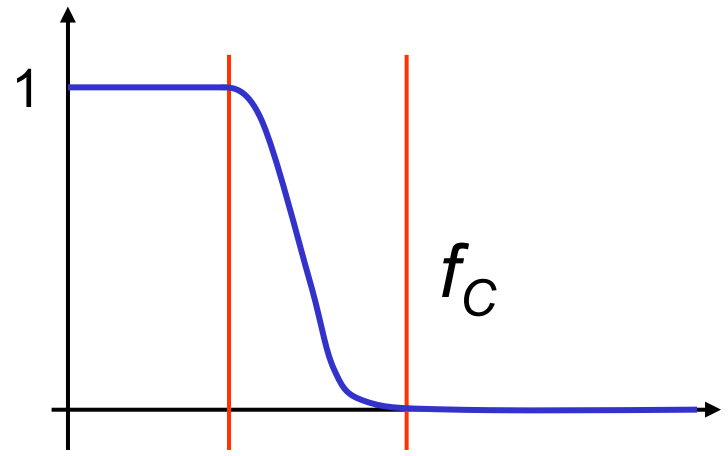
Cut-off methods (3)

The switch function used in, e.g., Tersoff-type potentials

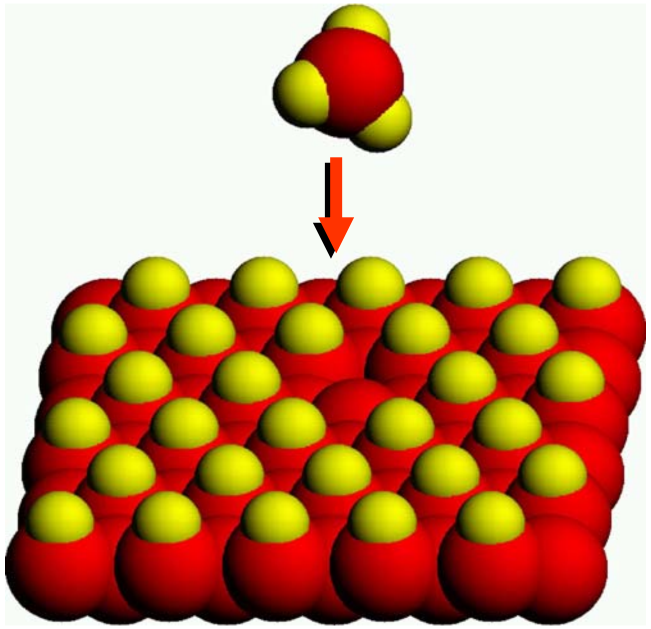
$$f_C(r) = \begin{cases} 1, & \text{if} \quad r \le R - D \\ \frac{1}{2} - \frac{1}{2}\sin\left(\frac{\pi}{2}\frac{(r-R)}{D}\right), & \text{if} \quad R - D < r < R + D \\ 0, & \text{if} \quad R + D \le r \end{cases}$$

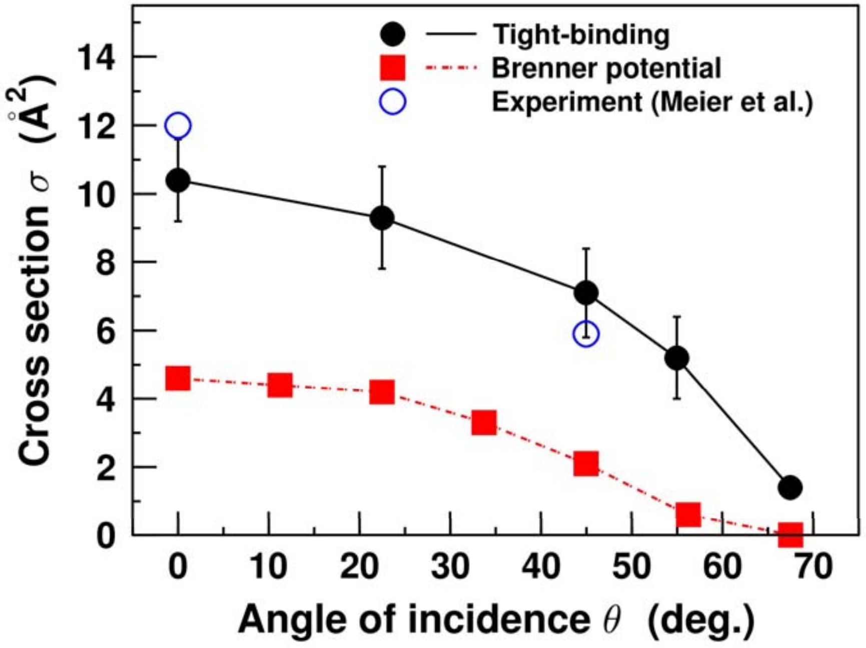


In this case, instead of adding the switch function to the potential F_s(r), we multiply it by f_C(r)



Example: Changing the Brenner potential cutoff





Tight-binding
Brenner potential
Experiment (Meier et al.)
Cross section σ (Å^2)
0
2
4
6
8
10
12
14
0
10
20
30
40
50
60
70
Angle of incidence θ (deg.)

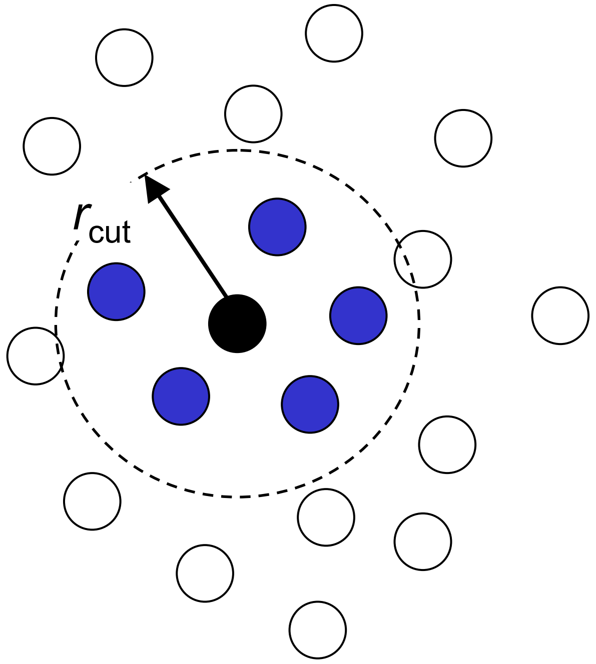
- Simulations of CH_3 sticking on diamond (111) surface; aim to obtain sticking cross-sections in specific case studies.
- Original parametrization of the Brenner C-H potential (more on this tomorrow) has a C-C cut-off radius of 2.0 Å.
- Increasing the cut-off to 2.46 Å (maximum) results in an increase of the cross-section by a factor of ~3, but the values remain still too low.

Träskelin et al., J. Appl. Phys. 93 (2002) 1826

Neighbour list

- Looking for atoms within the cutoff distance is quite time-consuming in MD:

```
do i=1,N
        do j=i,N
          if(i == j) cycle
          dx = x(j) – x(j)
          dy = x(j) – y(i)
          dz = z(j) – z(i)
          rsq = dx*dx + dy*dy + dz*dz
          r = sqrt(rsq) ! Don’t do this at home!
        enddo
enddo
```



- However, since atoms move within a time step only < 0.2 Å, the local neighbours of a given atom remain the same for many time steps.

- We can tabulate the neighbours of each atom and focus our search for interaction pairs within that table.

Verlet neighbour list

- Construct a list which contains, for each atom *i*, the indices of all atoms *j* whose distance from atom *i* is smaller than some chosen distance r_m

  with the condition that $r_\mathrm{m} > r_\mathrm{cut}$

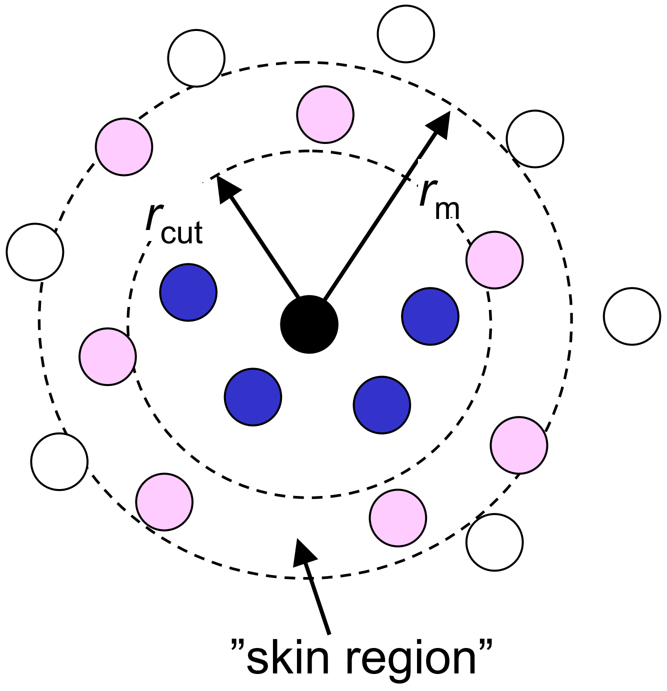
- Update the list every N_m time steps

- N_m and r_m are chosen so that

  $$r_m - r_c > N_m v_{typ}\, \delta t$$

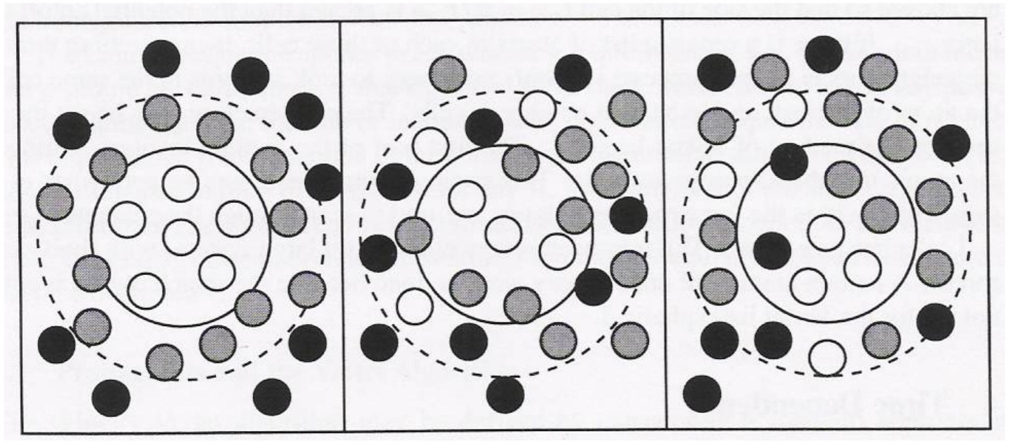
  where v_{typ} is a typical speed of an atom.

- Only the atoms inside the neighbour list are taken into account in calculations of the short-range forces.



When to update the list?

So, the idea is to update the list soon enough to prevent the situation where atoms coming from outside r_m should be interacting with the central atom *i,* while according to the neighbour list they are *not*.

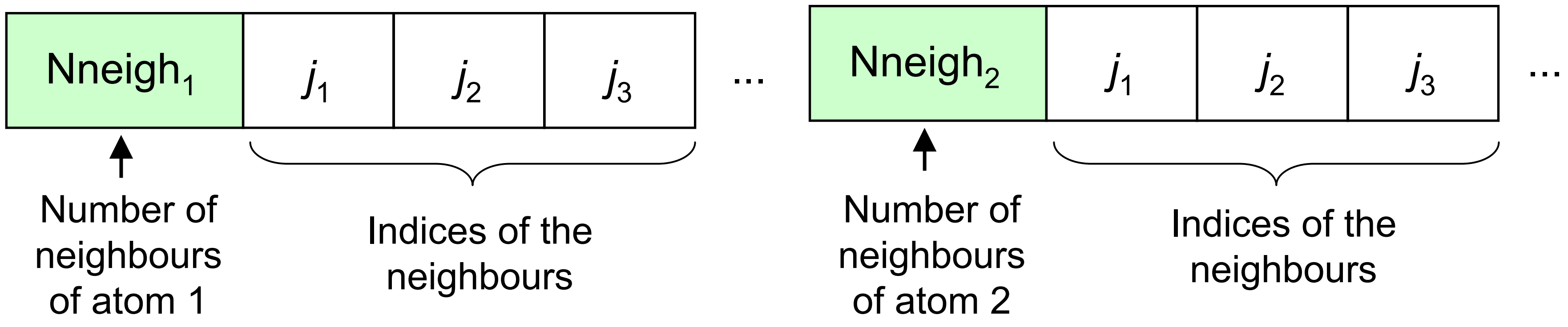

Construction ... Later ... Too late!

A static value of N_m may lead to problems when there can be energetic processes in the system so that $v_j >> v_{typ}$

A better way is to have a dynamic way of determining when to update the neighbour list. For example:

1) Keep track of the two largest atom displacements from the time when the neighbour list was updated the last time.
2) Update the list when $r_{max,1} + r_{max,2} > r_m - r_c$

Verlet neighbour list in practice



Nneigh_1
j_1
j_2
j_3
…
Nneigh_2
j_1
j_2
j_3
…
Number of neighbours of atom 1
Indices of the neighbours
Number of neighbours of atom 2
Indices of the neighbours

```
startofneighbourlist = 1
do i=1,N
        nneighboursi = 0
        do j=1,N
                if(i == j) cycle
                dx = x(j) – x(j)
                dy = x(j) – y(i)
                dz = z(j) – z(i)
                rsq = dx*dx + dy*dy + dz*dz
                if(rsq < rmsq) then
                        nneighboursi = nneighboursi + 1
                        neighlist(startofneighbourlist + nneighboursi) = j
                endif
        enddo
        neighbourlist(startofneighbourlist) = nneighboursi
        startofneighbourlist = startofneighbourlist + nneighboursi + 1
enddo
```

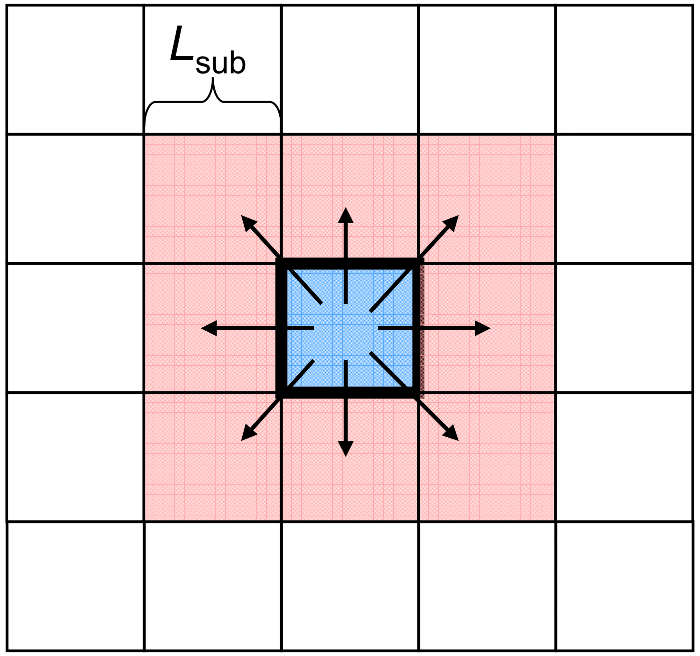
Linked list and cell division

The Verlet list decreases the amount of calculating distances to other atoms by a factor of N_m. However, our algorithm is still $O(N^2)$.

Using a cellular division of the simulation cell and a linked list we can make the algorithm truly $O(N)$.

We first divide the MD cell into smaller subcells, say $MxMxM$ of them (for a cubic cell). The size of one subcell is chosen so that $L_{sub} = L_{cell}/M > r_m$

Now, as we look for neighbours for an atom in the subcell highlighted in blue, we will limit the search only to the neighbouring subvolumes shown in pink.



Average number of atoms in a subcell

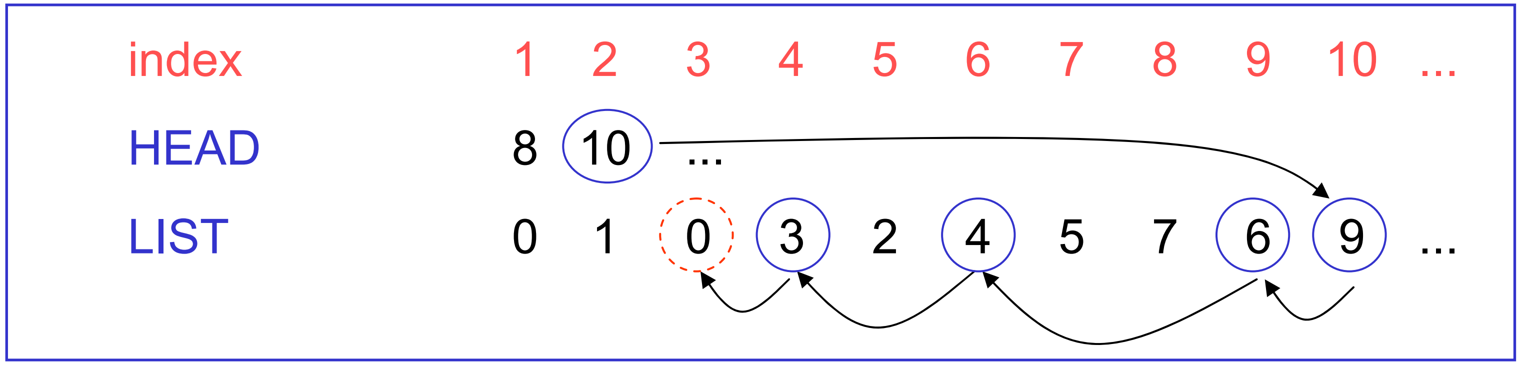
$$N_{sub} = N/M^3$$

Linked list and cell division (2)

Instead of the original $N(N\text{-}1) \sim N^2$ atom pairs, we now have to go through only 27 x N_{sub} x N atom pairs. (Small systems?)

For simple, symmetric potential energy functions (e.g. Lennard-Jones) we further have to calculate only every second neighbour pair in the force calculations.

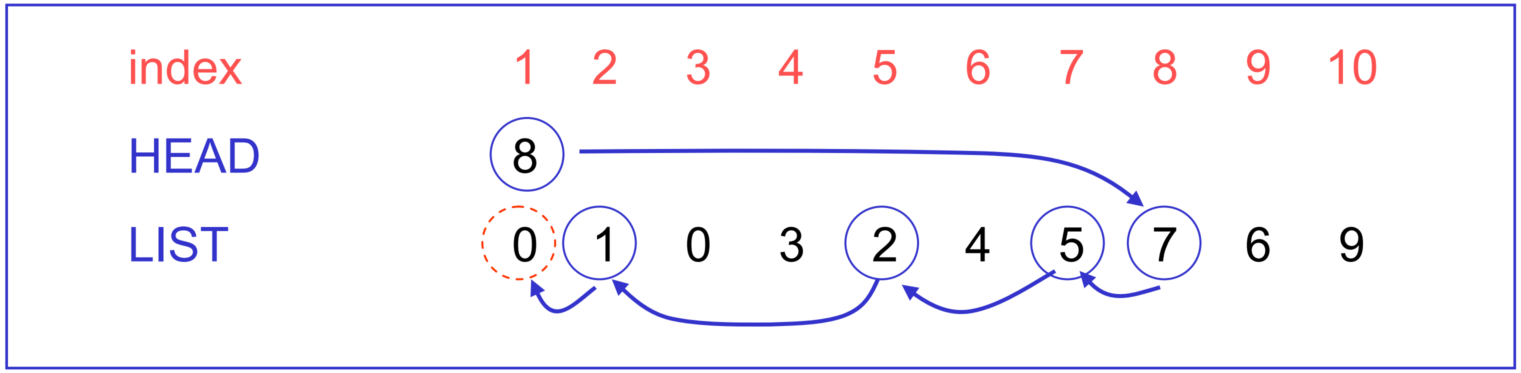
In practice:



HEAD (size MxMxM): tells us where the neighbours in a subcell start

LIST (size N): element i tells us where the next atom index of this cell is

Linked list and cell division (3)



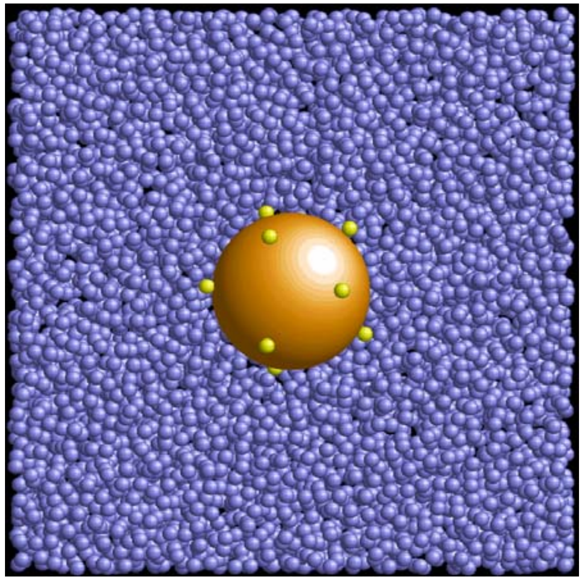
This algorithm requires periodic boundaries (it can be formulated for open boundaries as well, but things get a bit tricky there).

```
do i=1,N
        head(i) = 0
enddo
do i = 1,N
        icell = 1 + int((x(i)+xsize/2)/xsize*M) &  !Cell center at origin
                    int((y(i)+ysize/2)/ysize*M)*M &
                    int((z(i)+zsize/2)/zsize*M)*M*M
        list(i) = head(icell)
        head(icell) = i

enddo
```

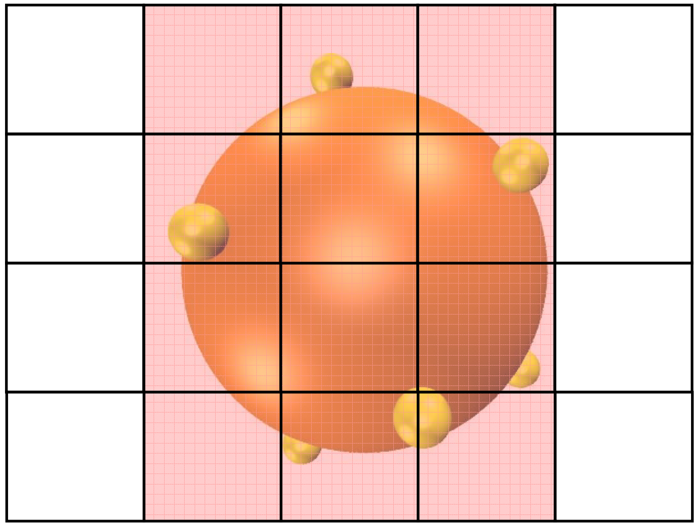
Different particle sizes

Consider a large solute embedded in a solvent consisting of much smaller particles.



Constructing a cell list for the solvent particles is ok. However, the solute is much larger than a single (or several!) subcells.

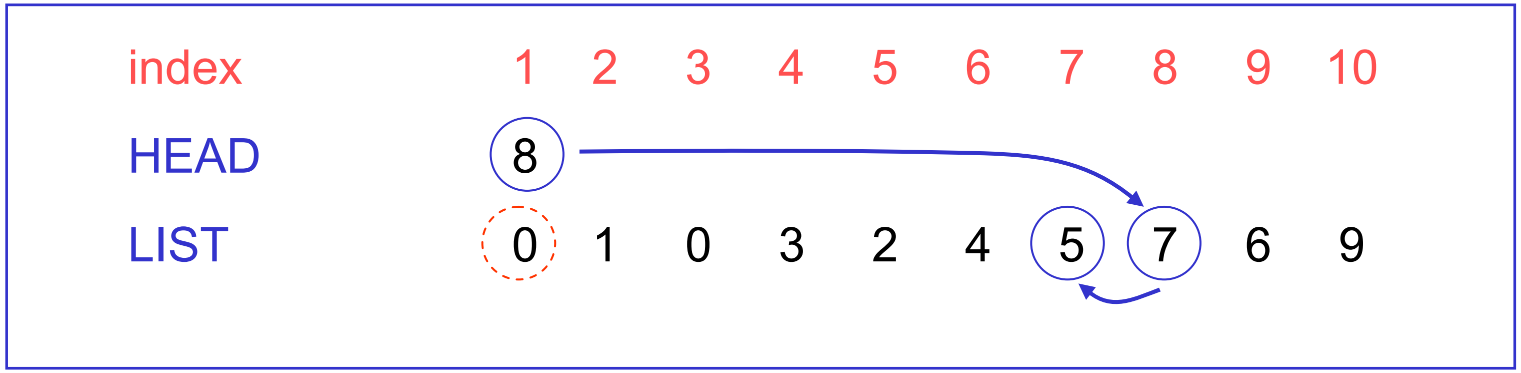
One option is to construct a standard Verlet list for the solute. The linked list of the solvent particles is then decoded into a similar Verlet list.



While we could envision other ways of coping with this, in practice one generally decodes the cell lists to Verlet lists (regardless of variations in particle size) in order to simplify the force calculations.

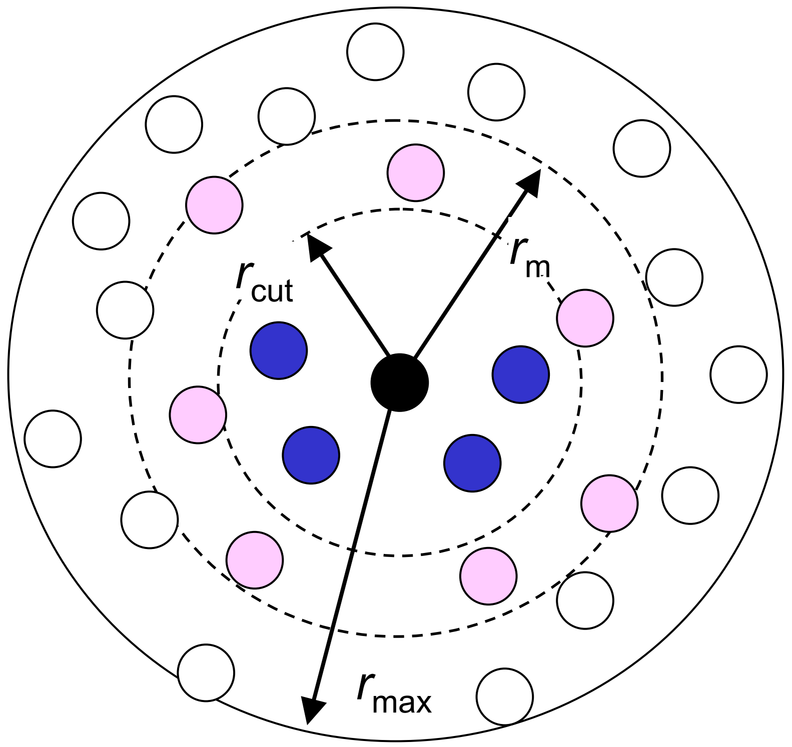
Linked list to a Verlet list

```
do i=1,N
        do (loop over the 27 neighbour cells)
                icell = (inz*My + iny)*Mx + inx
                ! Get the first atom in the cell
                j = head(icell)
                do
                        if(j==0) ! Exit from the innermost loop
                        (... Get the distance r between atoms i and j ...)
                        if(r < rmsq) then
                                (... Accept neighbour ...)
                        endif
                        j = list(j)
                enddo
        enddo
enddo
```



Twin-range approach

We can include long-range energy corrections to the neighbour list – cut-off scheme with the so-called twin range approach.



When a neighbour list is constructed, calculate the interaction energies and forces of the central atom with the ones at

$r_m < r_j < r_{max}$ i.e. ○

After the construction of the neighbour list carry out the force and energy calculations as usual by using the neighbour list...

... But add also the long-range corrections (according to a Taylor expansion of the forces) that were calculated when the neighbour list was constructed, every single time step until the neighbour list is constructed again.

In principle one thus obtains slightly more accurate forces and interactions, but the neighbour list has to be updated frequently enough.

Summary

- Short-range interaction cut-offs can be done in a variety of ways: direct truncation, shifting, and switching of the potential energy functions.
- Important things are to get rid of discontinuities at the cut-off distance and ensure that the potential still has the desired properties.
- Verlet neighbour list is straightforward to construct, but this makes the algorithm $O(N^2)$.
- Linked list and cell division make the algorithm truly $O(N)$, so their use is strongly recommended. (However, note the prefactors in the scaling!)