Computation of non-bonded interactions: Part 1

Computation of non-local (non-bonded) interactions in the potential function $E_p^{non\text{-}local}$ scales with the number of atoms as N^2. For the large molecular systems which contain thousands of atoms this scaling makes the computation of $E_p^{non\text{-}local}$ prohibitively slow. To reduce the computational costs several methods have been developed. These include (i) modifications of potential functions or forces (cut-off methods), (ii) generation of neighbor lists, and (iii) Ewald or multipole methods (for electrostatic interactions). In this lecture the first two approaches are discussed.

I. Cut-off methods

Generally the non-bonded potentials, such as van-der-Waals (represented by the Lennard-Jones function) or electrostatic interactions are assumed spherically symmetric. Therefore, it is possible to modify the original non-bonded potential in such a way that it would decay faster at large distances *r* and become zero at some finite *r*. Three modification techniques are commonly used, which are based on truncation, switch, and shift functions. All three must satisfy several requirements:

1. potential (or force) should remain minimally perturbed by modifying function at small distances;

2. Modified potential (or force) must remain a smooth function. This requirement is crucial for molecular dynamics, Langevin dynamics, or minimization procedures. Violation of this requirement may result in catastrophic instability of the integration of equation of motions due to sudden variation of forces;

3. Because energy is conserved in standard molecular dynamics simulations (NVE ensemble), modification of potentials or forces should not lead to noticeable energy drift.

The general implementation of cut-off methods takes on the form

$$V_{ij}(r_{ij}) = S(r_{ij})\left[\frac{B_{ij}}{r_{ij}^{12}} - \frac{A_{ij}}{r_{ij}^{6}} + \frac{q_i q_j}{r_{ij}}\right], \qquad (1)$$

where *S(r)* is a function, which modifies the pairwise non-bonded potential V_{ij} for a pair of atoms *i* and *j* (The meaning of the terms in Eq. (1) is discussed in the previous lecture).

Truncation function: The most simple implementation of cut-off method is based on truncation of non-bonded interactions at $r=a$. In this case $S(r)=1$ for $r \le a$, and $S(r)=0$ at $r > a$. In general, truncation creates a discontinuity in the potential, which, strictly speaking, results in the infinite force at $r=a$. Because truncation violates all the conditions outlined above (except (1)), its use is generally not recommended. However, if the potential function is already close to zero at $r=a$, the numerical error associated with the truncation is small. MOIL molecular dynamics package uses the truncation method for van-der-Waals interactions with the typical value of a=9 Å. The ratio $V_{LJ}(r=a)/V_{LJ}(r=r_{min})$ (where r_{min} is the location of minimum in Lennard-Jones potential) for OPLS force field used in MOIL is about 0.03 for a=9Å (assuming the interactions between C_β carbons).

<u>*Switching function:*</u> One can use a smooth function, which gradually brings the potential to zero in the interval (a,b). One possible choice for $S(r)$ is

$$S(r) = 1,\ r < a$$
$$S(r) = 1 + \left(\frac{r^2 - a^2}{b^2 - a^2}\right)^2 \left(2\frac{r^2 - a^2}{b^2 - a^2} - 3\right),\ a \le r \le b \tag{2}$$
$$S(r) = 0,\ r > b$$

The first derivative of $S(r)$ becomes zero at $r=a$ and $r=b$. NAMD utilizes multistep integration, therefore the switching function in Eq. (2) is applied to modify van-der-Waals forces (not the potentials itself). Because $S(r)$ is a smooth function, it gradually reduces the amplitude of van-der-Waals interactions from "full strength" at $r=a$ to zero at $r=b$. Similar to van-der-Waals interactions $S(r)$ is applied to electrostatic forces and not to the potential itself. Using $S(r)$ the electrostatic forces F_{EL} in NAMD are partitioned into local and non-local contributions as

$$F_{EL}(r) = S(r)F_{EL}(r) + (1 - S(r))F_{EL}(r). \tag{3}$$

The first term in Eq. (3) represents the local part of electrostatic forces, and the second term gives the long-range part. The separation of electrostatic forces in NAMD is used in multistep algorithm. Local part of F_{EL} is computed every integration step, but the non-local part is evaluated periodically using Ewald summation. The typical values for a and b are 8 and 12 Å. The switching functionality in NAMD is controlled by the keyword `switching` (see the NAMD configuration file in Box 1). The keywords `switchdist` and `cutoff` specify a and b values, respectively. It is important to remember that switching of electrostatic forces is performed in conjunction with the application of particle mesh Ewald technique (key word `PME` must be set `on`). If `PME` is off, then the keyword `switching` actually implies shifting procedure for electrostatic interactions.

Even though the switching function $S(r)$ is smooth, it still introduces a transient increase in the absolute value of the derivative of the potential function that leads to artificial forces. To minimize their impact the width of the switching interval b-a must be sufficiently large (e.g., > 4 Å). CHARMM uses the same switching function for both, van-der-Waals and electrostatic interactions. In contrast, MOIL utilizes switching function only for partitioning the electrostatic interactions.

Shift functions: As an alternative to switching functions, one may use shift functions, which slightly shift the potential in the entire range of *r* to intersect it with the *x* axis. The specific implementation of shift function differs for electrostatic and van-der-Waals interactions. For electrostatic interactions two shift functions are introduced

$$S_1(r) = \left(1 - \frac{r^2}{b^2}\right)^2 \text{ (r} \le \text{b)} \tag{4}$$

and

$$S_2(r) = \left(1 - \frac{r}{b}\right)^2 \text{ (r} \le \text{b)} \tag{5}$$

For $r > b$ both shift functions are zero. However, the force computed for $S_1(r)$ contains a term, which depends on *r*, whereas $S_2(r)$ adds a constant (independent on *r*) term to the force. Both shifted potential functions and their first derivatives become zero at $r=b$. NAMD uses $S_1(r)$ function to shift electrostatic interactions, when Ewald method is not selected (PME off). In this case the keyword switching implies the application of shift function. CHARMM uses both $S_1(r)$ and $S_2(r)$ functions, while MOIL does not use any of the shift functions.

Another form of shift function is used for van-der-Waals interactions. The potential is modified as

$$V_{LJ}(r_{ij}) = \frac{B_{ij}}{r_{ij}^{12}} - \frac{A_{ij}}{r_{ij}^{6}} + C_{ij} r_{ij}^{6} + D_{ij} \text{ for } r \le b. \tag{6}$$

The coefficients C_{ij} and D_{ij} are chosen from the conditions $V_{LJ}(r=b)=0$ and $\left.\frac{\partial V_{LJ}}{\partial r}\right|_{r=b} = 0$. These conditions result in $C_{ij} = 2B_{ij}b^{-18} - A_{ij}b^{-12}, D_{ij} = -3B_{ij}b^{-12} + 2A_{ij}b^{-6}$. Note that when only a constant is added in Eq. (6), the derivative $\frac{\partial V_{LJ}}{\partial r}$ would not become zero at $r=b$. Shifting of van-der-Waals interactions is not used in NAMD or MOIL, but is implemented in CHARMM.

II. Neighbor lists

Cut-off methods alone do not save computational time, because they require calculation of the distances between all the pairs of atoms in order to make an appropriate modification of the non-bonded potential. Therefore, the computations still scale as N^2. The additional, essentially bookkeeping technique, which significantly reduces the CPU time, is based on Verlet lists and consists of the following steps.

1. For each particle *i* a list is generated, which includes all the particles that are found within the sphere $r_v > r_c$, where r_c is the cut-off distance for non-bonded interactions. This list is called a Verlet list for a particle *i*. There are *N* Verlet lists created, and the positions of particles at the time of Verlet lists' generation are saved.

2. At each conformational update the check is made on whether a maximum displacement of any particle *j* (with respect to the stored position of the particle *j* at the time of Verlet list generation) exceeds r_v -r_c. If yes, one of the particles previously located outside the Verlet list may have moved within the cut-off distance r_c of another particle. Note that this is necessary, but not sufficient condition for the appearance of a new particle in the sphere of r_c radius around any other particle.

As long as the displacement of particles remains small, the Verlet list is used to compute the pairwise interaction energy. Implementation of Verlet list is given in Algorithm 34 in Frenkel and Smit book. Note that the arrays nlist and xv contain the number of particles in the Verlet lists and the positions of particles at the time of generation of Verlet lists. The array list contains the particle indexes in the Verlet lists of all particles.

The Verlet lists quickly become incomplete, because many particles enter or leave the buffer between the spheres of the radii r_c and r_v without triggering the new generation of the lists. However, the new lists are always created when a particle appears in the cut-off sphere of any other particle. The computation of non-bonded interactions scales as *N* when Verlet lists are used and as N^2 when new Verlet lists are created.

In NAMD the generation of Verlet lists is performed periodically without computations of actual particle displacements. The interval between Verlet lists generations is specified by the keyword stepspercycle. The default value is 20. The radius of the Verlet list sphere is given by pairlistdist (the default value is 13.5 Å). Therefore, NAMD tolerates certain inaccuracies in keeping track the particles in the cut-off spheres. To minimize these errors the optional pairlistspercycle parameter can used to force more frequent Verlet list generation.

Another type of lists created to reduce N^2 scaling in the computation of non-bonded interactions is the cell lists. The cell lists are generated by dividing the system into the cells of the size $r_n \geq r_c$. For each cell the list of particles is made, which belong to a given cell. Then computation of non-bonded interactions for a particle *i* can be restricted only to the particles in a given and immediately adjacent cells. Algorithm 37 in Frenkel and Smit book shows the generation of cell lists for one dimensional system (Box 3) and utilizes two arrays. The array hoc registers the last particle put into the list of particles for a given cell. The array ll (linked-list array) gives for a given particle the index of the previous particle that has been put into the list of the same cell. It is important to note that the generation of cell lists and the corresponding computation of non-bonded energies scale as N^1 (Box 4).

In practice, Verlet lists speed up the computations only if the number of particles in a Verlet list $N_v << N$, the total number of particles. For typical values of Lennard-Jones potential this condition is met for N exceeding approximately 100. The powerful approach is to create first cell lists and then generate Verlet lists based on existing cell lists that allows to reduce the overall scaling from N^2 to N^1. Practical implementation of cell and Verlet lists can use atom or group based approach. In atom based approach all atoms are treated individually, when they are assigned to lists. This approach potentially leads to instabilities, because atoms from the same group (e.g., belonging to the amide group, water molecule etc) may be assigned to different lists causing sudden force jumps. The group based approach takes into account the distribution of atoms over groups and allocates groups (not atoms) to cell or Verlet lists. In NAMD the groups are referred to as patches and hydrogens are always allocated to the same patch to which the donor atom is assigned. Note that CHARMM topology file top_all22_prot.inp defines the atom groups for each amino acid.

The use of neighbor lists requires the application of interaction cut-offs. In NAMD neighbor lists are applied for short-range (van-der-Waals) non-bonded interactions, for which the scaling of computations becomes linear with N. For long-range (electrostatic) non-bonded interactions Ewald sums are typically used.

Box 1. Example of the NAMD configuration file

```
# input system......
structure ala2_solv.psf
coordinates ./output/ala2_quench0100.coor
bincoordinates ./output/ala2i_quench0100.coor
binvelocities ./output/ala2i_quench0100.vel

#..force field.......................
paratypecharmm on
parameters par_all22_na.inp
parameters par_all22_prot.inp
exclude scaled1-4
1-4scaling 1.0                        #for EL1-4 only
dielectric 1.0

switching on
switchdist 8.0                        # =a
cutoff 12.0                           # =b, applies to both for VdW and EL
```

```
pairlistdist 13.5
margin 0.0
stepspercycle 20                    # frequency of updating non-bonded list
rigidBonds all                      # shake all Hs
rigidTolerance 0.00001              # default value
rigidIterations 100


# Ewald EL.........................
PME on                              # turn on Ewald sum
PMETolerance 0.000001               # default value
PMEGridSizeX 32                     # as in Moil 2^5
PMEGridSizeY 32                     # as in Moil 2^5
PMEGridSizeZ 32                     # as in Moil 2^5

#...peptide CM restraint....
#constraints on


#integrator ............
timestep 1.0
fullElectFrequency 4

#output...................
outputenergies 1000                 # frequency of writing E to stdout
outputtiming 1000                   # frequency of writing time to stdout
binaryoutput no                     # don't use binary files for final output
outputname output/ala2_quench0101   # where to save final coord andvel
restartname output/ala2i_quench0101
restartfreq 10000                   # frequency of writing restart coord
                                    # and vel out
binaryrestart yes                   # no binary files for restart
DCDfile output/ala2_quench0101.dcd  # trajectory filename
dcdfreq 1000                        # frequency of writing to dcdfile

#MD protocol..............
seed 3190101
numsteps 1000000                    #steps in simulations
#temperature 300                    # generate initial distr. of vel for T
#rescaleFreq 1
#rescaleTemp 300
#reassignFreq 1                     # frequency of reassigning of temp
#reassignTemp 300
#reassignIncr 0.001                 # T increment
#reassignHold 300                   # final T after heating
```

```
# periodic boundary conditions......
cellBasisVector1 34.7 0.0 0.0
cellBasisVector2 0.0 34.7 0.0
cellBasisVector3 0.0 0.0 34.7
cellOrigin 0.0 0.0 0.0
wrapWater on
```

Box 2. Generation of Verlet lists

```
subroutine new_vlist
do i=1,npart
        nlist(i) = 0                    #initializing the Verlet lists
        xv(i) = x(i)                    # storing the positions of particles in xv
enddo
do i=1,npart-1                                  # loop over all pairs of particles
 do j=i+1,npart                                 #
        xr = x(i)-x(j)
        if(abs(xr) < rv)then            # determining if the particle j belongs to the
                nlist(i) = nlist(i) + 1 # Verlet list of i and vice versa
                nlist(j) = nlist(j) + 1 # advancing the counters in Verlet lists
                list(i,nlist(i)) = j            # storing the particles i and j in
respective
                list(j,nlist(j)) = I            # Verlet lists
        endif
 enddo
enddo
return
end subroutine
```

Here, npart is the number of particles, and the array x contains the current positions of particles.

Box 3. Generation of the cell lists

```
subroutine new_nlist(rc)

rn=box/int(box/rc)                              # setting the size of cells to rn ≥ rc
do icel=0,ncel-1
        hoc(icel) = 0                           # initializing hoc arrays
enddo
```

```
do i=1,npart                                        # assigning the particles to cell lists
        icel = int(x(i)/rn)                         # based on hoc and linked-list array ll
        ll(i)=hoc(icel)
        hoc(icel)=i
enddo
return
end subroutine
```

Here, box is the system size, rc is the cut-off distance, ncel is the number of cells, and npart is the number of particles.

Box 4. Calculation of energy using cell lists

```
subroutine ennlist

en=0
icel = int(xi/rn)                           # cell number for a particle i
do ncel = 1, nneigh                         # loop over adjacent cells
        jcel = neigh(icel,ncel)             # adjacent cell number from the list neigh()
        j=hoc(jcel)                         # first particle in the cell
        do while (j > 0)
                if(i/=j) en=en+enij(i,xi,j,xj)
                j=ll(j)                             # get a new partcle from the linked-list ll
        enddo
enddo
return
end subroutine
```

***Homework**: User guide for NAMD is at http://www.ks.uiuc.edu/Research/namd/current/ug/*