Linked-List Cell Molecular Dynamics

This chapter explains the linked-list cell MD algorithm, the computational time of which scales as $O(N)$ for N atoms. We will replace function `computeAccel()` in the $O(N^2)$ program `md.c` by this algorithm.

Recall that the naive double-loop implementation to compute pair interactions scales as $O(N^2)$. In fact, with a finite cut-off length, r_c (see `#define RCUT 2.5` in `md.c`), an atom interacts with only a limited number of atoms ~ $(4\pi/3)r_c^3(N/V)$, where V is the volume of the simulation box. The linked-list cell algorithm explained below computes the entire interaction with $O(N)$ operations.

CELLS

First divide the simulation box into small cells of equal size. The edge lengths of each cell, (r_{cx}, r_{cy}, r_{cz}) (`double rc[3]`), must be at least r_c; we use $r_{c\alpha} = L_\alpha/L_{c\alpha}$, where $L_{c\alpha} = \lfloor L_\alpha/r_c \rfloor$ (α = x, y, z) and L_α is the simulation box length in the α direction (`double Region[3]`). Here $\lfloor x \rfloor$ is the floor function, i.e., the largest integer that is less than or equal to x. An atom in a cell interacts with only the other atoms in the same cell and its 26 neighbor cells. The number of cells to accommodate all these atoms is $L_{cx}L_{cy}L_{cz}$, where $L_{c\alpha} = L_\alpha/r_{c\alpha}$ (α = x, y, z) (`int lc[3]`). We identify a cell with a vector cell index, $\vec{c} = (c_x, c_y, c_z)$ ($0 \le c_x \le L_{cx}-1$; $0 \le c_y \le L_{cy}-1$; $0 \le c_z \le L_{cz}-1$), and a serial cell index (see the figure below),

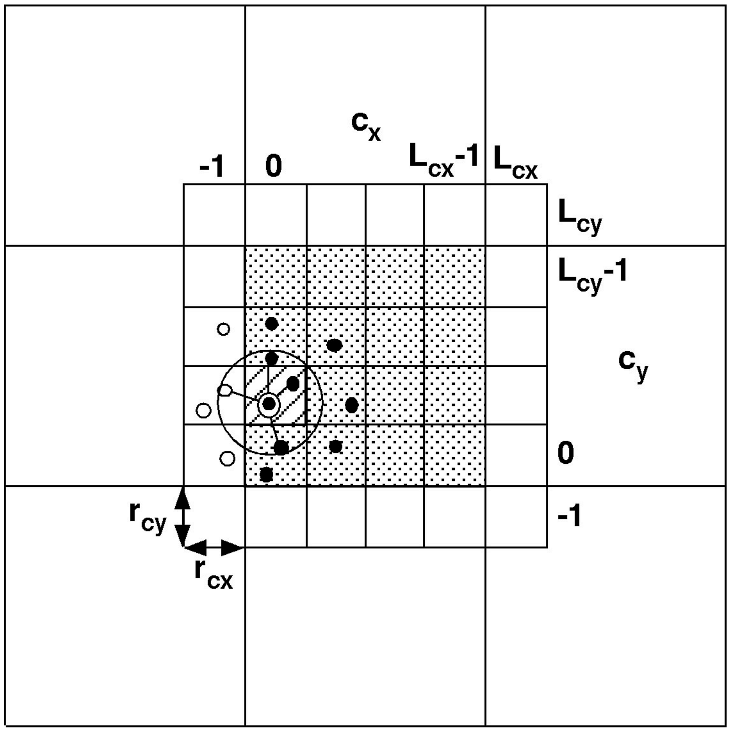
$$c = c_xL_{cy}L_{cz} + c_yL_{cz} + c_z$$

or

$$c_x = c/(L_{cy}L_{cz})$$
$$c_y = (c/L_{cz}) \bmod L_{cy}$$
$$c_z = c \bmod L_{cz}.$$

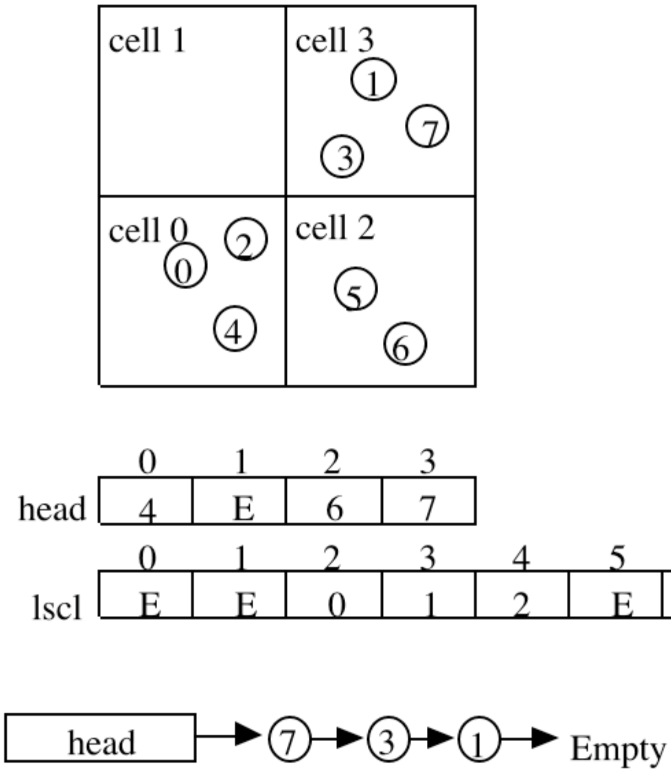
An atom with coordinate $\vec{r}$ belongs to a cell with the vector cell index,

$$c_\alpha = \lfloor r_\alpha/r_{c\alpha} \rfloor \ (\alpha = \text{x, y, z}).$$



LISTS

The atoms belonging to a cell is organized using linked lists. The following describes the data structures and algorithms to construct the linked lists and compute the interatomic interaction using them.



DATA STRUCTURES

`lscl[NMAX]`: An array implementation of the linked lists. `lscl[i]` holds the atom index to which the *i*-th atom points.
`head[NCLMAX]`: `head[c]` holds the index of the first atom in the *c*-th cell, or `head[c]` = `EMPTY` (= –1) if there is no atom in the cell.

ALGORITHM 1: LIST CONSTRUCTOR

```
/* Reset the headers, head */
for (c=0; c<lcxyz; c++) head[c] = EMPTY;
/* Scan atoms to construct headers, head, & linked lists, lscl */
for (i=0; i<nAtom; i++) {
  /* Vector cell index to which this atom belongs */
  for (a=0; a<3; a++) mc[a] = r[i][a]/rc[a];
  /* Translate the vector cell index, mc, to a scalar cell index */
  c = mc[0]*lcyz+mc[1]*lc[2]+mc[2];
  /* Link to the previous occupant (or EMPTY if you're the 1st) */
  lscl[i] = head[c];
  /* The last one goes to the header */
  head[c] = i;
}
```

where `lcyz = lc[1]*lc[2]; lcxyz = lcyz*lc[0].`

ALGORITHM 2: INTERACTION COMPUTATION

```
/* Scan inner cells */
for (mc[0]=0; mc[0]<lc[0]; (mc[0])++)
for (mc[1]=0; mc[1]<lc[1]; (mc[1])++)
for (mc[2]=0; mc[2]<lc[2]; (mc[2])++) {
  /* Calculate a scalar cell index */
  c = mc[0]*lcyz+mc[1]*lc[2]+mc[2];
  /* Scan the neighbor cells (including itself) of cell c */
  for (mc1[0]=mc[0]-1; mc1[0]<=mc[0]+1; (mc1[0])++)
  for (mc1[1]=mc[1]-1; mc1[1]<=mc[1]+1; (mc1[1])++)
  for (mc1[2]=mc[2]-1; mc1[2]<=mc[2]+1; (mc1[2])++) {
    /* Periodic boundary condition by shifting coordinates */
    for (a=0; a<3; a++) {
      if (mc1[a] < 0)
        rshift[a] = -Region[a];
      else if (mc1[a]>=lc[a])
        rshift[a] = Region[a];
      else
        rshift[a] = 0.0;
    }
    /* Calculate the scalar cell index of the neighbor cell */
    c1 = ((mc1[0]+lc[0])%lc[0])*lcyz
        +((mc1[1]+lc[1])%lc[1])*lc[2]
        +((mc1[2]+lc[2])%lc[2]);
    /* Scan atom i in cell c */
    i = head[c];
    while (i != EMPTY) {
      /* Scan atom j in cell c1 */
      j = head[c1];
      while (j != EMPTY) {
        if (i < j) { /* Avoid double counting of pair (i, j) */
          $r_{ij}$ = $r_i$-($r_j$+$r_{shift}$); /* Image-corrected relative pair position */
          if ($r_{ij}$ < $r_c^2$)
            Compute forces on pair (i, j)
          ...
        }
        j = lscl[j];
      }
      i = lscl[i];
    }
  }
}
```

A neighbor cell can be outside of the central simulation box, i.e., the cell-index vector component c_α can be -1 or L_α (α = x, y, z). To find the atoms that belong to such an outside cell, the modulo operator (%) pulls back the cell index into the range $[0, L_\alpha-1]$ (α = x, y, z):

```
c1 = ((mc1[0]+lc[0])%lc[0])*lcyz
   + ((mc1[1]+lc[1])%lc[1])*lc[2]
   + ((mc1[2]+lc[2])%lc[2]);
```

The atomic positions in an outside cell, on the other hand, must be treated as they are and must not be pulled back into the central simulation box. The shift vector $\vec{r}_{\text{shift}}$ (`double rshift[3]`) to shift the central image of such an atom to the appropriate image position by a simulation box length, if necessary:

```
if (mc1[a] < 0) rshift[a] = -Region[a]; else if (mc1[a]>=lc[a]) rshift[a] =
Region[a]; else rshift[a] = 0.0;
```