Constraints in molecular dynamics simulations

I. Lagrangian formulation of classical mechanics

The laws of classical mechanics can be expressed using the so called Lagrangian formulation. This formalism is based on the notion of action *S*, which is defined as an integral over the trajectory fragment between the time moments t_1 and t_2

$$S = \int_{t_1}^{t_2} \left(E_{kin} - E_{pot}\right) dt = \int_{t_1}^{t_2} L(\vec{r}, \dot{\vec{r}}) dt \,, \tag{1}$$

where $L(\vec{r}, \dot{\vec{r}}) \equiv E_{kin}(\dot{\vec{r}}) - E_{pot}(\vec{r})$ is the Lagrangian. The Newton's equations of motions can be derived for any actual trajectory followed in MD by requiring that S reaches extremum for the actual trajectory. This requirement results in the equation

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\vec{r}}} = \frac{\partial L}{\partial \vec{r}}, \tag{2}$$

from which the familiar Newton's equation of motion follows. It follows from the Lagrangian approach that forces can be computed from the derivatives of the potential as $\vec{f} = -\frac{\partial E_{pot}}{\partial \vec{r}}$.

II. Incorporating constraints into molecular dynamics simulations

Because fluctuations in bond lengths cannot be correctly described by classical physics, the bond potential is often replaced with the constraints, which maintain the constant bond lengths. Consider a polypeptide chain and set the requirement that all backbone bond lengths are equal to *d*. This is equivalent to introducing *N*-1 constraints $\sigma_k = r_{i,i+1}^2 - d^2$. The Lagrangian, which takes into account the constraints, is

$$L' = L - \sum_k \lambda_k \sigma_k \,. \tag{3}$$

The second term in Eq. (3) can be viewed as an additional potential due to the presence of constraints with unknown Lagrangian factors λ_j. By using Eq. (3) one can write the equations of motions as

$$m\ddot{\vec{r}}_i = \frac{\partial L'}{\partial \vec{r}_i} = \vec{F}_i - \sum_k \lambda_k \frac{\partial \sigma_k}{\partial \vec{r}_i} \,. \tag{4}$$

Denote the additional forces (second term in Eq. (4)) as

$$\vec{G}_{ik} = -\lambda_k \frac{\partial \sigma_k}{\partial \vec{r}_i} \qquad (5)$$

Let us now consider a singe constraint for a single bond in a diatomic molecule. For the constraint $\sigma = \frac{1}{2}(r^2 - d^2)$, the corresponding force (Eq. (5)) is a centripetal force $\vec{G} = -\lambda \vec{r}$. In order to determine λ one can treat one of the constrained atoms as rotating on a circular orbit around the other one. From this analogy $\lambda = m\omega^2$ (where ω is the angular frequency) and $\vec{G} = -m\omega^2 \vec{r}$. Note that the force *G* is the only force acting on the rotating atom (*F*=0). Applying the Verlet algorithm and assuming that the constraint is satisfied at *t* and *t*-Δt, one can show that the constraint at $t+\Delta t$ is maintained with the accuracy

$$r^2(t+\Delta t) - d^2 \approx -\frac{(\omega \Delta t)^4}{6} d^2 \ . \qquad (6)$$

The computed trajectory will exponentially diverge from the circular orbit. Therefore, the constraints must be satisfied exactly (not simply within the accuracy of Verlet algorithm).

Consider a polypeptide chain of *N* atoms and assume that *N-1* bond lengths are constrained to the distance *d*. Using Verlet algorithm one can write the coordinates of an atom *i* at *t*+*D*t as

$$\vec{r}_i(t+\Delta t) = 2\vec{r}_i(t) - \vec{r}_i(t-\Delta t) - \frac{\Delta t^2}{m_i} \sum_k \lambda_k \frac{\partial \sigma_k(t)}{\partial \vec{r}_i}, \qquad (7)$$

where $\sigma_k = r_{i,i+1}^2 - d^2$ is the *k*th constraint imposed between the atoms *i* and *i*+1. Let us assume that the deviation between unconstrained trajectory $\vec{r}_i'(t+\Delta t) = 2\vec{r}_i(t) - \vec{r}_i(t-\Delta t)$ and the one satisfying the constraint $\vec{r}_i(t+\Delta t)$ is small (prime indicates unconstrained trajectory). Using $\vec{r}_i(t+\Delta t) - \vec{r}_i'(t+\Delta t)$ as a small parameter we expand $\sigma_k(t+\Delta t)$ keeping only linear terms as

$$\sigma_k(t+\Delta t) = \sigma_k'(t+\Delta t) + \sum_i \frac{\partial \sigma'(t+\Delta t)}{\partial \vec{r}_i} \left(\vec{r}_i(t+\Delta t) - \vec{r}_i'(t+\Delta t) \right). \qquad (8)$$

Setting the requirement that the constraints must be satisfied at $t+\Delta t$, we rewrite Eq. (8) as

$$\sigma_k'(t+\Delta t) = \sum_i \sum_l \frac{\Delta t^2}{m_i} \frac{\partial \sigma_k'(t+\Delta t)}{\partial \vec{r}_i} \lambda_l \frac{\partial \sigma_l(t)}{\partial \vec{r}_i}. \qquad (9)$$

Eq. (9) is a set of *N-1* linear equations with respect to unknown Lagrangian factors λ_l. Although these equations can be solved analytically, in practice inverting the matrix for the system of *N-1* linear equations is computationally expensive and approximate methods should be used. For a small protein of about 50 amino acids, which contains roughly about 500 bonds, 500 λ_l factors must be determined.

III. SHAKE algorithm

To devise a numerical method for adjusting the conformations of protein to satisfy bond-length constraints, assume that constraints are applied iteratively in order from k=1 to k=N-1. Consider a single constraint $\sigma_k = r_{i,i+1}^2 - d^2$ and rewrite Eqs. (7,9) accordingly

$$\vec{r}_i(t+\Delta t) = 2\vec{r}_i(t) - \vec{r}_i(t-\Delta t) - \frac{\Delta t^2}{m_i}\lambda_k \frac{\partial \sigma_k(t)}{\partial \vec{r}_i} \tag{10}$$

and

$$\sigma_k'(t+\Delta t) = \sum_i \frac{\partial \sigma_k'(t+\Delta t)}{\partial \vec{r}_i}\lambda_k \frac{\partial \sigma_k(t)}{\partial \vec{r}_i}\frac{\Delta t^2}{m_i}. \tag{11}$$

Therefore, the Lagrangian multiplier for the single constraint k is

$$\lambda_k = \frac{\sigma_k'(t+\Delta t)}{\sum_i \frac{\partial \sigma_k'(t+\Delta t)}{\partial \vec{r}_i}\frac{\partial \sigma_k(t)}{\partial \vec{r}_i}\frac{\Delta t^2}{m_i}}. \tag{12}$$

Generally, because the kth constraint depends on the coordinates of atoms i and i+1, only two terms are present in the sum in the denominator of Eq. (12). The sign of λ_k is determined by the sign of $\sigma_k'(t+\Delta t)$, i.e, if the bond length is stretched, the force G is negative acting to reduce the bond length. Once λ_k is computed, the corrected positions of atoms i and i+1 at $t+\Delta t$ are obtained using Eq. (10).

In practice, the iterative procedure, which adjusts the coordinates of atoms after each "unconstrained" iteration of Verlet algorithm consists of the following steps:

1. Iterate equation of motions using Verlet algorithm to get the unconstrained coordinates $\vec{r}_i'(t+\Delta t)$ for all atoms.
2. Select kth constraint (acting on atoms i and i+1) and compute the factor λ_k using Eq. (12), then adjust the coordinates of atoms i and i+1 using Eq. (10).
3. Select a new constraint k+1 (acting on the atoms i+1 and i+2) and use the coordinates $\vec{r}_{i+1}(t+\Delta t)$ (step 2) and $\vec{r}_{i+2}'(t+\Delta t)$ to compute the factor λ_{k+1}. Both coordinates are considered as unconstrained by the k constraint. Adjust the positions of atoms i+1 and i+2 using Eq. (10).

4. Repeat the procedure for other constraints, considering one constraint at a time.
5. Start a new iteration from the first constraint using the positions of atoms computed during the previous iteration (steps 2-4). Several iterations are usually needed, because each application of the k+1 constraint partially undoes the adjustment performed by the kth constraint.

6. The number of iterations is determined by the desired accuracy in keeping the bond lengths constant.

This iterative scheme is referred to as SHAKE algorithm developed by Berendsen and coworkers (*J. Comp. Phys.* **23**, 327 (1977)). NAMD uses SHAKE algorithm to constrain bond lengths.

The following keywords specify the performance of SHAKE (Box 1). If rigidBonds is set to all, all bonds involving light atoms (hydrogens) are constrained. The keyword rigidTolerance specifies the tolerance of bond length convergence (the default value is 10^{-8} Å). If the difference between the bond length adjusted by SHAKE and the target length d is less than the tolerance, SHAKE iterations are stopped. The keyword rigidIterations (the default value is 100) determines the maximum number of SHAKE iterations. Typically SHAKE converges within about 10 iterations.

An important advantage of using SHAKE is the possibility to select longer integration steps as compared to the molecular dynamics without constrained bonds.

Box 1. Example of the NAMD configuration file for Aβ11-42 dimer

```
# input system.....
structure  ab_m2_dimer_solv.psf
coordinates      ./output/ab_quench04149.coor
bincoordinates   ./output/abi_quench04149.coor
binvelocities    ./output/abi_quench04149.vel

#..force field.....................
paratypecharmm on
parameters  par_all22_na.inp
parameters  par_all22_prot.inp
exclude  scaled1-4
1-4scaling       1.0
dielectric       1.0
switching  on
switchdist  8.0
cutoff  12.0
pairlistdist 13.5
margin  0.0
stepspercycle 20
```

```
rigidBonds      all                 # apply SHAKE to all hydrogens
rigidTolerance  0.00001             # maximum difference between adjusted and
assigned
                                    # bond lengths in Å
rigidIterations 100                 # maximum number of SHAKE iterations

# Ewald EL.....................
PME             on                  #  turning on PME
PMETolerance    0.000001            #  accuracy of computing real-space term
PMEGridSizeX    32                  # grids for fast evaluation in Fourier sum
PMEGridSizeY    32                  # note that 32=2^5
PMEGridSizeZ    32                  #

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

#output.....................
outputenergies 1000
outputtiming 1000
binaryoutput no
outputname output/ab_quench04150
restartname     output/abi_quench04150
restartfreq     10000
binaryrestart   yes
DCDfile         output/ab_quench04150.dcd
dcdfreq  1000

#MD protocol..............
seed            32204150
numsteps        40000
#temperature 300
#rescaleFreq     1
#rescaleTemp     runtemp
#reassignFreq 1
#reassignTemp   300
#reassignIncr 0.001
#reassignHold 300

# periodic boundary conditions......
cellBasisVector1   57.8   0.0  0.0          # components of the unit cell basis
vectors
cellBasisVector2    0.0  57.8  0.0          #
cellBasisVector3    0.0   0.0 57.8          #
cellOrigin              0.0   0.0  0.0      # position of the unit cell center
wrapWater           on
```