```
-----------------------------------------------------------------
Traj  T (K)     <E(T)> (kJ/mol)    Cv(T) (kJ/mol/K)
  1     70.0      -2.927478E+02        1.901182E-01
  2     75.2      -2.921061E+02        1.918446E-01
  3     80.7      -2.907020E+02        1.947924E-01
  4     86.6      -2.899388E+02        1.956457E-01
  5     93.0      -2.883251E+02        1.993989E-01
  6     99.9      -2.874375E+02        2.012598E-01
  7    107.2      -2.854636E+02        2.073695E-01
  8    115.1      -2.844192E+02        2.103462E-01
  9    123.6      -2.832991E+02        2.130148E-01
 10    132.7      -2.807932E+02        2.240865E-01
 11    142.4      -2.794164E+02        2.294507E-01
 12    152.9      -2.744829E+02        2.737941E-01
 13    164.2      -2.695909E+02        3.837099E-01
 14    176.3      -2.662145E+02        4.779797E-01
 15    189.3      -2.616999E+02        6.111249E-01
 16    203.2      -2.487103E+02        7.997212E-01
 17    218.2      -2.410640E+02        7.489722E-01
 18    234.2      -2.274456E+02        5.589124E-01
 19    251.5      -2.216274E+02        5.079860E-01
 20    270.0      -2.099786E+02        4.759778E-01
-----------------------------------------------------------------
 Energy Histrogram Analysis within
    [  -3.055179E+02,  -1.136750E+02 ] kJ/mol
 Number of Configuration with Energy
    Below:   -3.055179E+02 kJ/mol,          0 Config
    Above:   -1.136750E+02 kJ/mol,          6 Config
 Energy Histogram Resolution:   500
-----------------------------------------------------------------
 Date and time : 2018-03-07, 23:59:29
 WALL time used:    0 days   2 hours  40 minutes  3.2574 seconds
```

File 2.2: Last lines of `mPTMC`'s output from a Monte Carlo simulation of $(H_2O)_8$ cluster (TIP4P) using 20 parallel tempering trajectories geometrically progressed between 70 and 270 K, and 10^6 Monte Carlo cycles.

2.3 Displacement and Rotation

The primary idea of `mPTMC` is to use randomness by generating an enormous number of possible solid or liquid-like arrangements. While in the atomic codes only random displacements of atoms are needed, in a molecular code we also need to care of the relative orientation of the molecule. At present, we still keep the individual molecule rigid. Those arrangements are generated according to a Markov chain, starting from the initial coordinates (r_o) of the system of interest. Then, the subroutine `displace-rotate` randomly chooses an atom or molecule and displaces and rotates its coordinates, $r_o \rightarrow$

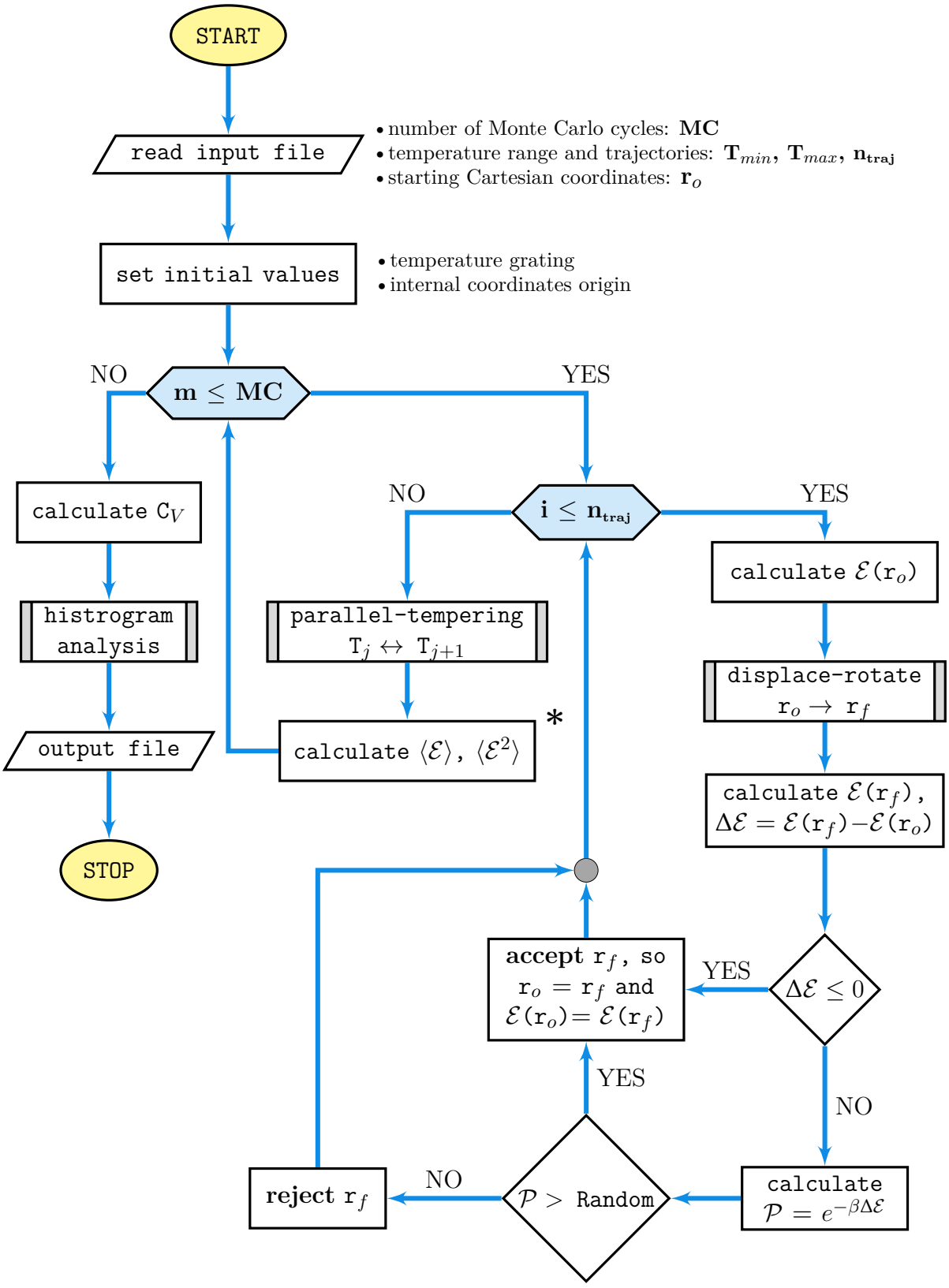

Figure 2.2: Flowchart describing the general Parallel Tempering Monte Carlo algorithm. The blue shapes indicates counting loops and the double-lined rectangles indicates a reference to a subroutine. The computation of average quantities (*) is only performed after the equilibrium has been reached. The connection circle (●) only helps to show the direction of program flow. We denote by `Random` a random number in the range [0, 1).

r_f (Fig. 2.2). For a many-body system, the easiest way to do this is using Jacobi coordinates;[35] that is, to displace the center of mass and rotate the principal axes.[36]

Although, `mPTMC` can do a simulation of atomic, molecular, or mixed clusters; obviously, it is not necessary to rotate a single atom or calculate its center of mass (here, atoms are considered as points). In the following, our explanations will apply to molecules. The `displace-rotate` procedure displaces and rotates an atom or a molecule in a three steps process. This subroutine calculates the molecular center of mass and randomly moves it, and then the rigid molecule is rotated. A simplified Fortran algorithm for each of these steps is described below.

First, a random molecule is selected and its center of mass is compute, using Jacobi coordinates, the center of mass for a molecule of N atoms is

$$\mathbf{R}_{\mathrm{CM}} = \frac{1}{W}\sum_{k}^{N} A_k \mathbf{r}_k \tag{2.3}$$

where $W = \sum_k A_k$ is the total molecular mass; A_k is the atomic mass and $\mathbf{r}_k$ are the Cartesian coordinates for the k-th atom, respectively. In total we have `NM` molecules and each molecule consists of `NA` atoms. File 2.3 shows a basic Fortran algorithm to compute the center of mass. The function `AtomicMass` returns the atomic mass, `REAL(8)`, for an atom with atomic number `Z` (line 10, File 2.3). `ro` is a three indices array. The first index, `sm`, specifies a molecule the set of `NM` molecules; the second, `j`, defines an atom from a system of `NA` atoms in the molecule; and last, `k`, defines each Cartesian component x (`k` $= 1$), y (`k` $= 2$), or z (`k` $= 3$). Line 11, File 2.3.

```
1  !selecting a random molecule
2  sm=FLOOR(1+NM*Random(seed))       !NM: total number of molecules
3  !center of mass (RCM)
4  DO k=1,3                          !coordinate x(k=1),y(k=2),z(k=3)
5     !Initialization
6     TotalWeight=0.0                !Total molecular mass
7     RCM(k)=0.0
8     !Loop: Atom inside a Molecule
9     DO j=1,NA(sm)                  !NA: number of atoms
10       TotalWeight=TotalWeight+AtomicMass(Z(sm,j))
11       RCM(k)=RCM(k)+ro(sm,j,k)*AtomicMass(Z(sm,j))
12    ENDDO
13    RCM(k)=RCM(k)/TotalWeight
14 ENDDO
```

File 2.3: Simplified Fortran code visualization of the center of mass calculation for a random molecule (sm). `RCM` are the coordinates of the center of mass.

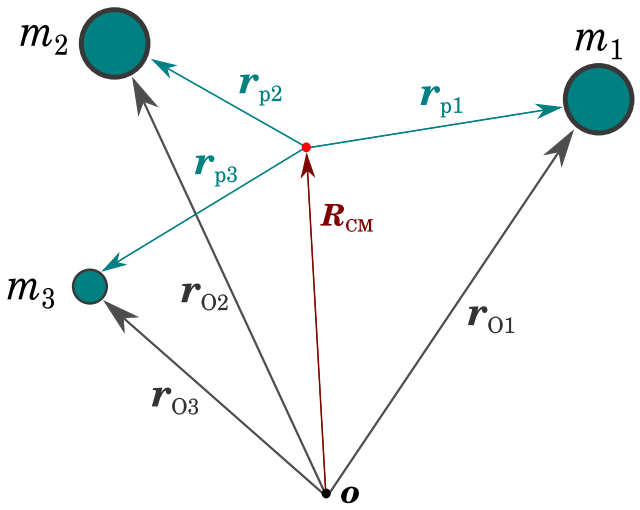

Figure 2.3: Set of principal axes for three-body problem; the principal axes are $\mathbf{r}_{p1}$, $\mathbf{r}_{p2}$, $\mathbf{r}_{p3}$ (blue-green), the center of mass is $\mathbf{R}_{\text{CM}}$ (red), and the initial coordinates are $\mathbf{r}_{o1}$, $\mathbf{r}_{o2}$, $\mathbf{r}_{o3}$ (gray).

After calculating the center of mass, the second step is to displace it, randomly. The line 5 and 6 in File 2.4 show a random move of the center of mass, where `Random` is a uniform pseudo-random number generated in the range [0, 1)[37] and `maxDisplace` is the maximum displacement allowed (line 7, File 2.1). The movement must be inside a sphere of defined radius (line 6, File 2.1). A displacement of $\mathbf{R}_{\text{CM}}$ is in the sphere with center (h_1, h_2, h_3) and radius r_{SBC} if and only if

$$r_{\text{SBC}}^2 > \left[(R_{\text{CM}}^{(x)} - h_1)^2 + (R_{\text{CM}}^{(y)} - h_2)^2 + (R_{\text{CM}}^{(z)} - h_3)^2\right] \tag{2.4}$$

(lines 9 and 10, File 2.4). Based on the initial coordinates `ro`, `mPTMC` computes the coordinates origin (h_1, h_2, h_3). By default, to move the center of mass within the sphere, a maximum of one hundred thousand attempts can be made.

In the same fashion as the displacement of the center of mass, the rotation is also performed randomly. In line 3 File 2.5, `maxRotat` is the maximum rotation allowed for the molecular principal axes (defined in the input file, line 7 File 2.1). Those axes represent the position of any atom in the molecular frame of reference, when the coordinates of the center of mass define the origin. The principal axes can be found by simple vectorial analysis; by definition, the principal axes are $\mathbf{r}_{pi} = \mathbf{r}_{oi} - \mathbf{R}_{\text{CM}}$ $(i = 1, 2, 3)$, $\mathbf{r}_o$ are the initial coordinates (Fig. 2.3). The random displacement (`displace`), accepted for the center of mass in Eq. 2.4, is explicitly shown in line 6, File 2.5 to calculate the principal axes (`rp`).

Any orientation can be reached, starting from the reference orientation of principal axes,

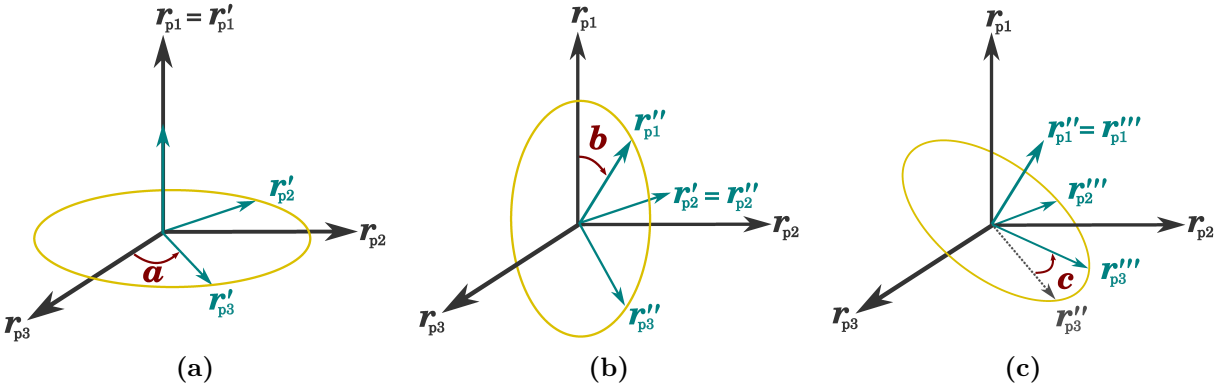

Figure 2.4: Euler angles geometrical definition. The fixed system, for the principal axes, is shown in black. The rotated systems is shown in blue-green. (a) first rotation around $\mathbf{r}_{p1}$, (b) second rotation around $\mathbf{r}'_{p2}$, and (c) third rotation around $\mathbf{r}''_{p1}$. The rotation through the angles a and c are counterclockwise.

using a specific sequence of intrinsic rotations, whose magnitudes are the Euler angles $(a, b, c,$ Fig. 2.4). The Euler angles describe the orientation of a rigid molecule with respect to a fixed internal coordinate system.

```
counter=0
DO
   !attempt to displace
   DO k=1,3
      Displace(k)=(2.0*Random(seed)-1.0)*maxDisplace
      RCM(k)=RCM(k)+Displace(k)
   ENDDO
   !spherical boundary conditions, origin (h1,h2,h3)
   r2=( (RCM(1)-h(1))**2 + (RCM(2)-h(2))**2 + (RCM(3)-h(3))**2  )
   !a good move?
   IF ( rsbc**2 > r2 ) THEN
      !Exiting from the first loop
      EXIT
   ELSE
      !Recovering the old coordinates
      DO k=1,3
         RCM(k)=RCM(k)-Displace(k)
      ENDDO
      counter=counter+1
      !max number of attempts 1E5
      IF ( counter > 100000 ) CALL ABORT
   ENDIF
ENDDO
```

File 2.4: Simplified Fortran code visualization of the random displacement of a center of mass (`RCM`) inside a sphere of radius `rsbc`.

```
DO
   DO k=1,3
      !principal axes for rotation
      DO j=1,NA(sm)
         rp(sm,j,k)=ro(sm,j,k)+Displace(k)-RCM(k)
      ENDDO
      !euler rotation angles
      Rotate(k)=(2.0*Random(seed)-1.0)*MaxRotat
   ENDDO
   a=Rotate(1) ; b=Rotate(2) ; c=Rotate(3) !a(k=1), b(k=2), c(k=3)
   !Rotating principal axes
   DO j=1,NAtoms(sm)
      rp(sm,j,1)=rp(sm,j,1)*( COS(a)*COS(b)*COS(c) - SIN(a)*SIN(c) )+&
                 rp(sm,j,2)*(-COS(a)*COS(b)*SIN(c) - SIN(a)*COS(c) )+&
                 rp(sm,j,3)*( COS(a)*SIN(b)                        )
      !Final coordinates (x, k=1) after move and rotate
      rf(sm,j,1)=rp(sm,j,1)+RCM(1)

      rp(sm,j,2)=rp(sm,j,1)*( SIN(a)*COS(b)*COS(c) + COS(a)*SIN(c) )+&
                 rp(sm,j,2)*(-SIN(a)*COS(b)*SIN(c) + COS(a)*COS(c) )+&
                 rp(sm,j,3)*( SIN(a)*SIN(b)                        )
      !Final coordinates (y, k=2) after move and rotate
      rf(sm,j,2)=rp(sm,j,2)+RCM(2)

      rp(sm,j,3)=rp(sm,j,1)*(-SIN(b)*COS(c)                        )+&
                 rp(sm,j,2)*( SIN(b)*SIN(c)                        )+&
                 rp(sm,j,3)*( COS(b)                               )
      !Final coordinates (z, k=3) after move and rotate
      rf(sm,j,3)=rp(sm,j,3)+RCM(3)
   ENDDO
   Overlap=0
   DO i=1,NM                          !loop over all molecules
      IF ( i == sm ) CYCLE            !avoiding self-counting
      DO jsm=1,NA(sm)                 !jsm is for all atoms in sm
         DO ji=1,NA(i)                !ji is for all atoms in i
            d(jsm,ji,i)=( (rf(sm,jsm,1) - ro(i,ji,1))**2 + &
                          (rf(sm,jsm,2) - ro(i,ji,2))**2 + &
                          (rf(sm,jsm,3) - ro(i,ji,3))**2    )
            IF( cutoff > d(jsm,ji,i) )  Overlap=Overlap+1
         ENDDO
      ENDDO
   ENDDO
   IF ( Overlap == 0 ) EXIT   !Exiting from the first loop
ENDDO
```

File 2.5: Simplified Fortran code visualization of the random rotation about the axes of a coordinate system; that is, rotations of principal axes (`rp`) according to the three Euler angles `a`, `b` and `c`.

We can obtain the matrix representations of these transformation, and apply them sequentially as a matrix product to obtain the overall effect of the rotation. As a result, the rotation of the principal axes coordinates is as follows

1. The coordinates are rotated about the $\mathbf{r}_{p1}$ axis counterclockwise through an angle a in the range $0 \leq a < 2\pi$, onto new axes denoted by $\mathbf{r}_{p1}$, $\mathbf{r}'_{p2}$, $\mathbf{r}'_{p3}$. Fig. 2.4 (a). Where the first rotation matrix is

$$\mathcal{S}_1(a) = \begin{pmatrix} \cos(a) & \sin(a) & 0 \\ -\sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.5}$$

   causes $\mathbf{r}'_{p2}$ and $\mathbf{r}'_{p3}$ to remain in the same plane, and the $\mathbf{r}_{p1}$ ($=\mathbf{r}'_{p1}$) axis is not changed.

2. The coordinates are then rotated about the $\mathbf{r}'_{p2}$ axis clockwise through an angle b in the range $0 \leq b \leq \pi$, onto new axes denoted by $\mathbf{r}''_{p1}$, $\mathbf{r}'_{p2}$, $\mathbf{r}''_{p3}$. Fig. 2.4 (b). The second rotation matrix is

$$\mathcal{S}_2(b) = \begin{pmatrix} \cos(b) & 0 & -\sin(b) \\ 0 & 1 & 0 \\ \sin(b) & b & \cos(b) \end{pmatrix}. \tag{2.6}$$

   It is applied to the coordinate system as it exists after the first rotation. The $\mathbf{r}'_{p2}$ ($=\mathbf{r}''_{p2}$) axis is not changed under this rotation.

3. The third rotation is like the first, but with rotation around c. The coordinates are now rotated about the $\mathbf{r}''_{p1}$ axis counterclockwise through an angle c in the range $0 \leq c < 2\pi$, into the final axes, denoted $\mathbf{r}''_{p1}$, $\mathbf{r}'''_{p2}$, $\mathbf{r}'''_{p3}$. Fig. 2.4 (c). The $\mathbf{r}''_{p1}$ ($=\mathbf{r}'''_{p1}$) axis is not changed. Finally, the third rotation matrix is

$$\mathcal{S}_3(c) = \begin{pmatrix} \cos(c) & \sin(c) & 0 \\ -\sin(c) & \cos(c) & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.7}$$

Equivalently, the total rotation matrix is described by the triple matrix product

$$\mathcal{S}(a,b,c) = \mathcal{S}_1(a) \cdot \mathcal{S}_2(b) \cdot \mathcal{S}_3(c) \tag{2.8}$$

File 2.5 show a simplified Fortran code visualization of the random rotation (lines 13–15, 19–21, and 25–27). The rotated coordinates of the i-th principal axis is

$$\mathbf{r}_{pi}^{\text{rot}} = \mathcal{S} \cdot \mathbf{r}_{pi} \tag{2.9}$$

where $i = 1, 2, 3$, and $\mathbf{r}_{pi}^{\text{rot}} = \mathbf{r}_{pi}'''$ (Fig. 2.4). As a result, after displace and rotation, the final coordinate are (lines 17, 23 and 29 File 2.5)

$$\mathbf{r}_f = \mathbf{r}_{pi}^{\text{rot}} + \mathbf{R}_{\text{CM}}. \tag{2.10}$$

Finally, lines 32–42 (File 2.5) include a loop to guarantee there is not atomic overlapping; that is,

$$|\mathbf{r}_f - \mathbf{r}_o^{(m)}| > r_{\text{cutoff}} \tag{2.11}$$

where $\mathbf{r}_o^{(m)}$ are the coordinates for all the molecules ($\{1, 2, \ldots, m, \ldots, \texttt{NM}\}$) and r_{cutoff} is the maximum approach distance. For a point nucleus model, $r_{\text{cutoff}} = 0$ and any value larger than zero means a hard-sphere atomic radius. In terms of interaction, the pair potentials between atoms converge rapidly at distances larger than zero (`cutoff` in line 39, File 2.5).

2.4 Parallel Tempering

Parallel tempering, also known as replica exchange, is a procedure aimed at improving the thermodynamic behavior of the Monte Carlo method. At higher temperatures, Monte Carlo simulations are able to sample phase space more efficiently than at lower temperature. At lower temperatures, a simulation can early get trapped in local regions resultingÂ in a non-ergodic sampling of phase space. Parallel tempering improves the sampling by allowing to exchange configurations. This enable that the lower temperature trajectories can "jump" between different local regions.[38]

The essence of the method consists of the following. Our Monte Carlo simulation is propagated simultaneously for a number of trajectories (n_{traj}), so we have n_{traj} replicas (copies of the system), each at a different temperature T_i. Then, the exchange of configurations at different neighboring temperatures are attempted and accepted based on a Metropolis criterion.

Exchanges between neighboring trajectories with temperature T_i and T_j are accepted with the probability[39]

$$\mathcal{P}(T_i \leftrightarrow T_j) = MIN\left[1, e^{(\beta_i - \beta_j)(\mathcal{E}_i - \mathcal{E}_j)}\right] \tag{2.12}$$

where β is the inverse temperature $1/k_B T$. File 2.6 shows a simplified Fortran code visualizing the parallel tempering scheme.

As a rule of thumb, an exchange is attempted 10% of the time (Eq. 2.12). The exchange is efficient when the energy histograms of neighboring temperatures overlap. Therefore, the geometric distribution of temperatures is usually taken.[40,41]