3.13 Replica exchange

Replica exchange molecular dynamics (REMD) is a method that can be used to speed up the sampling of any type of simulation, especially if conformations are separated by relatively high energy barriers. It involves simulating multiple replicas of the same system at different temperatures and randomly exchanging the complete state of two replicas at regular intervals with the probability:

$$P(1 \leftrightarrow 2) = \min\left(1, \exp\left[\left(\frac{1}{k_B T_1} - \frac{1}{k_B T_2}\right)(U_1 - U_2)\right]\right) \tag{3.140}$$

where T_1 and T_2 are the reference temperatures and U_1 and U_2 are the instantaneous potential energies of replicas 1 and 2 respectively. After exchange the velocities are scaled by $(T_1/T_2)^{\pm 0.5}$ and a neighbor search is performed the next step. This combines the fast sampling and frequent barrier-crossing of the highest temperature with correct Boltzmann sampling at all the different temperatures [59, 60]. We only attempt exchanges for neighboring temperatures as the probability decreases very rapidly with the temperature difference. One should not attempt exchanges for all possible pairs in one step. If, for instance, replicas 1 and 2 would exchange, the chance of exchange for replicas 2 and 3 not only depends on the energies of replicas 2 and 3, but also on the energy of replica 1. In GROMACS this is solved by attempting exchange for all "odd" pairs on "odd" attempts and for all "even" pairs on "even" attempts. If we have four replicas: 0, 1, 2 and 3, ordered in temperature and we attempt exchange every 1000 steps, pairs 0-1 and 2-3 will be tried at steps 1000, 3000 etc. and pair 1-2 at steps 2000, 4000 etc.

How should one choose the temperatures? The energy difference can be written as:

$$U_1 - U_2 = N_{df} \frac{c}{2} k_B (T_1 - T_2) \tag{3.141}$$

where N_{df} is the total number of degrees of freedom of one replica and c is 1 for harmonic potentials and around 2 for protein/water systems. If $T_2 = (1 + \epsilon)T_1$ the probability becomes:

$$P(1 \leftrightarrow 2) = \exp\left(-\frac{\epsilon^2 c\, N_{df}}{2(1+\epsilon)}\right) \approx \exp\left(-\epsilon^2 \frac{c}{2} N_{df}\right) \tag{3.142}$$

Thus for a probability of $e^{-2} \approx 0.135$ one obtains $\epsilon \approx 2/\sqrt{c\,N_{df}}$. With all bonds constrained one has $N_{df} \approx 2\,N_{atoms}$ and thus for $c = 2$ one should choose ϵ as $1/\sqrt{N_{atoms}}$. However there is one problem when using pressure coupling. The density at higher temperatures will decrease, leading to higher energy [61], which should be taken into account. The GROMACS website features a so-called "REMD calculator," that lets you type in the temperature range and the number of atoms, and based on that proposes a set of temperatures.

An extension to the REMD for the isobaric-isothermal ensemble was proposed by Okabe *et al.* [62]. In this work the exchange probability is modified to:

$$P(1 \leftrightarrow 2) = \min\left(1, \exp\left[\left(\frac{1}{k_B T_1} - \frac{1}{k_B T_2}\right)(U_1 - U_2) + \left(\frac{P_1}{k_B T_1} - \frac{P_2}{k_B T_2}\right)(V_1 - V_2)\right]\right) \tag{3.143}$$

where P_1 and P_2 are the respective reference pressures and V_1 and V_2 are the respective instantaneous volumes in the simulations. In most cases the differences in volume are so small that the second term is negligible. It only plays a role when the difference between P_1 and P_2 is large or in phase transitions.

Hamiltonian replica exchange is also supported in GROMACS. In Hamiltonian replica exchange, each replica has a different Hamiltonian, defined by the free energy pathway specified for the simulation. The exchange probability to maintain the correct ensemble probabilities is:

$$P(1 \leftrightarrow 2) = \min\left(1, \exp\left[\left(\frac{1}{k_B T} - \frac{1}{k_B T}\right)((U_1(x_2) - U_1(x_1)) + (U_2(x_1) - U_2(x_2)))\right]\right) \tag{3.144}$$

The separate Hamiltonians are defined by the free energy functionality of GROMACS, with swaps made between the different values of λ defined in the mdp file.

Hamiltonian and temperature replica exchange can also be performed simultaneously, using the acceptance criteria:

$$P(1 \leftrightarrow 2) = \min\left(1, \exp\left[\left(\frac{1}{k_B T} -\right)\left(\frac{U_1(x_2) - U_1(x_1)}{k_B T_1} + \frac{U_2(x_1) - U_2(x_2)}{k_B T_2}\right)\right]\right) \tag{3.145}$$

Gibbs sampling replica exchange has also been implemented in GROMACS [63]. In Gibbs sampling replica exchange, all possible pairs are tested for exchange, allowing swaps between replicas that are not neighbors.

Gibbs sampling replica exchange requires no additional potential energy calculations. However there is an additional communication cost in Gibbs sampling replica exchange, as for some permutations, more than one round of swaps must take place. In some cases, this extra communication cost might affect the efficiency.

All replica exchange variants are options of the `mdrun` program. It will only work when MPI is installed, due to the inherent parallelism in the algorithm. For efficiency each replica can run on a separate rank. See the manual page of `mdrun` on how to use these multinode features.

3.14 Essential Dynamics sampling

The results from Essential Dynamics (see sec. 8.10) of a protein can be used to guide MD simulations. The idea is that from an initial MD simulation (or from other sources) a definition of

the collective fluctuations with largest amplitude is obtained. The position along one or more of these collective modes can be constrained in a (second) MD simulation in a number of ways for several purposes. For example, the position along a certain mode may be kept fixed to monitor the average force (free-energy gradient) on that coordinate in that position. Another application is to enhance sampling efficiency with respect to usual MD [64, 65]. In this case, the system is encouraged to sample its available configuration space more systematically than in a diffusion-like path that proteins usually take.

Another possibility to enhance sampling is flooding. Here a flooding potential is added to certain (collective) degrees of freedom to expel the system out of a region of phase space [66].

The procedure for essential dynamics sampling or flooding is as follows. First, the eigenvectors and eigenvalues need to be determined using covariance analysis (`g_covar`) or normal-mode analysis (`g_nmeig`). Then, this information is fed into `make_edi`, which has many options for selecting vectors and setting parameters, see `gmx make_edi -h`. The generated `edi` input file is then passed to `mdrun`.

3.15 Expanded Ensemble

In an expanded ensemble simulation [67], both the coordinates and the thermodynamic ensemble are treated as configuration variables that can be sampled over. The probability of any given state can be written as:

$$P(\vec{x}, k) \propto \exp\left(-\beta_k U_k + g_k\right), \tag{3.146}$$

where $\beta_k = \frac{1}{k_B T_k}$ is the β corresponding to the kth thermodynamic state, and g_k is a user-specified weight factor corresponding to the kth state. This space is therefore a *mixed*, *generalized*, or *expanded* ensemble which samples from multiple thermodynamic ensembles simultaneously. g_k is chosen to give a specific weighting of each subensemble in the expanded ensemble, and can either be fixed, or determined by an iterative procedure. The set of g_k is frequently chosen to give each thermodynamic ensemble equal probability, in which case g_k is equal to the free energy in non-dimensional units, but they can be set to arbitrary values as desired. Several different algorithms can be used to equilibrate these weights, described in the mdp option listings.

In GROMACS, this space is sampled by alternating sampling in the k and $\vec{x}$ directions. Sampling in the $\vec{x}$ direction is done by standard molecular dynamics sampling; sampling between the different thermodynamics states is done by Monte Carlo, with several different Monte Carlo moves supported. The k states can be defined by different temperatures, or choices of the free energy λ variable, or both. Expanded ensemble simulations thus represent a serialization of the replica exchange formalism, allowing a single simulation to explore many thermodynamic states.

3.16 Parallelization

The CPU time required for a simulation can be reduced by running the simulation in parallel over more than one core. Ideally, one would want to have linear scaling: running on N cores makes the simulation N times faster. In practice this can only be achieved for a small number of cores.

The scaling will depend a lot on the algorithms used. Also, different algorithms can have different restrictions on the interaction ranges between atoms.

3.17 Domain decomposition

Since most interactions in molecular simulations are local, domain decomposition is a natural way to decompose the system. In domain decomposition, a spatial domain is assigned to each rank, which will then integrate the equations of motion for the particles that currently reside in its local domain. With domain decomposition, there are two choices that have to be made: the division of the unit cell into domains and the assignment of the forces to domains. Most molecular simulation packages use the half-shell method for assigning the forces. But there are two methods that always require less communication: the eighth shell [68] and the midpoint [69] method. GROMACS currently uses the eighth shell method, but for certain systems or hardware architectures it might be advantageous to use the midpoint method. Therefore, we might implement the midpoint method in the future. Most of the details of the domain decomposition can be found in the GROMACS 4 paper [5].

3.17.1 Coordinate and force communication

In the most general case of a triclinic unit cell, the space in divided with a 1-, 2-, or 3-D grid in parallelepipeds that we call domain decomposition cells. Each cell is assigned to a particle-particle rank. The system is partitioned over the ranks at the beginning of each MD step in which neighbor searching is performed. Since the neighbor searching is based on charge groups, charge groups are also the units for the domain decomposition. Charge groups are assigned to the cell where their center of geometry resides. Before the forces can be calculated, the coordinates from some neighboring cells need to be communicated, and after the forces are calculated, the forces need to be communicated in the other direction. The communication and force assignment is based on zones that can cover one or multiple cells. An example of a zone setup is shown in Fig. 3.12.

The coordinates are communicated by moving data along the "negative" direction in x, y or z to the next neighbor. This can be done in one or multiple pulses. In Fig. 3.12 two pulses in x are required, then one in y and then one in z. The forces are communicated by reversing this procedure. See the GROMACS 4 paper [5] for details on determining which non-bonded and bonded forces should be calculated on which rank.

3.17.2 Dynamic load balancing

When different ranks have a different computational load (load imbalance), all ranks will have to wait for the one that takes the most time. One would like to avoid such a situation. Load imbalance can occur due to three reasons:

- inhomogeneous particle distribution
- inhomogeneous interaction cost distribution (charged/uncharged, water/non-water due to GROMACS water innerloops)

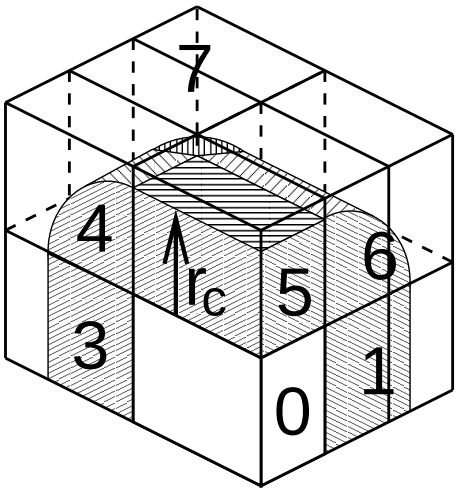

Figure 3.12: A non-staggered domain decomposition grid of 3×2×2 cells. Coordinates in zones 1 to 7 are communicated to the corner cell that has its home particles in zone 0. r_c is the cut-off radius.

- statistical fluctuation (only with small particle numbers)

So we need a dynamic load balancing algorithm where the volume of each domain decomposition cell can be adjusted *independently*. To achieve this, the 2- or 3-D domain decomposition grids need to be staggered. Fig. 3.13 shows the most general case in 2-D. Due to the staggering, one might require two distance checks for deciding if a charge group needs to be communicated: a non-bonded distance and a bonded distance check.

By default, `mdrun` automatically turns on the dynamic load balancing during a simulation when the total performance loss due to the force calculation imbalance is 5% or more. **Note** that the reported force load imbalance numbers might be higher, since the force calculation is only part of work that needs to be done during an integration step. The load imbalance is reported in the log file at log output steps and when the `-v` option is used also on screen. The average load imbalance and the total performance loss due to load imbalance are reported at the end of the log file.

There is one important parameter for the dynamic load balancing, which is the minimum allowed scaling. By default, each dimension of the domain decomposition cell can scale down by at least a factor of 0.8. For 3-D domain decomposition this allows cells to change their volume by about a factor of 0.5, which should allow for compensation of a load imbalance of 100%. The minimum allowed scaling can be changed with the `-dds` option of `mdrun`.

3.17.3 Constraints in parallel

Since with domain decomposition parts of molecules can reside on different ranks, bond constraints can cross cell boundaries. Therefore a parallel constraint algorithm is required. GROMACS uses the P-LINCS algorithm [48], which is the parallel version of the LINCS algorithm [47] . (see 3.6.2). The P-LINCS procedure is illustrated in Fig. 3.14. When molecules cross the cell boundaries, atoms in such molecules up to (`lincs_order + 1`) bonds away are communicated

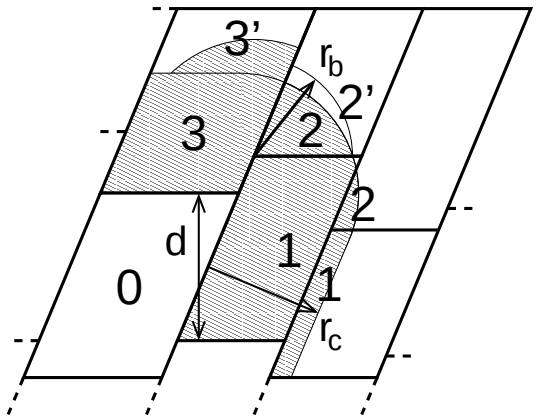

Figure 3.13: The zones to communicate to the rank of zone 0, see the text for details. r_c and r_b are the non-bonded and bonded cut-off radii respectively, d is an example of a distance between following, staggered boundaries of cells.

interaction	range	option	default
non-bonded	r_c = max(r_{list},r_{VdW},r_{Coul})	`mdp` file	
two-body bonded	max(r_{mb},r_c)	`mdrun -rdd`	starting conf. + 10%
multi-body bonded	r_{mb}	`mdrun -rdd`	starting conf. + 10%
constraints	r_{con}	`mdrun -rcon`	est. from bond lengths
virtual sites	r_{con}	`mdrun -rcon`	0

Table 3.3: The interaction ranges with domain decomposition.

over the cell boundaries. Then, the normal LINCS algorithm can be applied to the local bonds plus the communicated ones. After this procedure, the local bonds are correctly constrained, even though the extra communicated ones are not. One coordinate communication step is required for the initial LINCS step and one for each iteration. Forces do not need to be communicated.

3.17.4 Interaction ranges

Domain decomposition takes advantage of the locality of interactions. This means that there will be limitations on the range of interactions. By default, `mdrun` tries to find the optimal balance between interaction range and efficiency. But it can happen that a simulation stops with an error message about missing interactions, or that a simulation might run slightly faster with shorter interaction ranges. A list of interaction ranges and their default values is given in Table 3.3.

In most cases the defaults of `mdrun` should not cause the simulation to stop with an error message of missing interactions. The range for the bonded interactions is determined from the distance between bonded charge-groups in the starting configuration, with 10% added for headroom. For the constraints, the value of r_{con} is determined by taking the maximum distance that (`lincs_order + 1`) bonds can cover when they all connect at angles of 120 degrees. The actual constraint communication is not limited by r_{con}, but by the minimum cell size L_C, which has the following lower

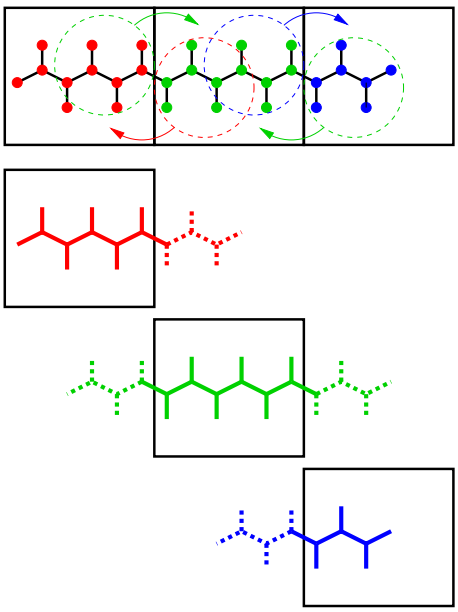

Figure 3.14: Example of the parallel setup of P-LINCS with one molecule split over three domain decomposition cells, using a matrix expansion order of 3. The top part shows which atom coordinates need to be communicated to which cells. The bottom parts show the local constraints (solid) and the non-local constraints (dashed) for each of the three cells.

limit:

$$L_C \geq \max(r_{\mathrm{mb}}, r_{\mathrm{con}}) \tag{3.147}$$

Without dynamic load balancing the system is actually allowed to scale beyond this limit when pressure scaling is used. **Note** that for triclinic boxes, L_C is not simply the box diagonal component divided by the number of cells in that direction, rather it is the shortest distance between the triclinic cells borders. For rhombic dodecahedra this is a factor of $\sqrt{3/2}$ shorter along x and y.

When $r_{\mathrm{mb}} > r_c$, `mdrun` employs a smart algorithm to reduce the communication. Simply communicating all charge groups within r_{mb} would increase the amount of communication enormously. Therefore only charge-groups that are connected by bonded interactions to charge groups which are not locally present are communicated. This leads to little extra communication, but also to a slightly increased cost for the domain decomposition setup. In some cases, *e.g.* coarse-grained simulations with a very short cut-off, one might want to set r_{mb} by hand to reduce this cost.

3.17.5 Multiple-Program, Multiple-Data PME parallelization

Electrostatics interactions are long-range, therefore special algorithms are used to avoid summation over many atom pairs. In GROMACS this is usually . PME (sec. 4.8.2). Since with PME all particles interact with each other, global communication is required. This will usually be the limiting factor for scaling with domain decomposition. To reduce the effect of this problem, we have come up with a Multiple-Program, Multiple-Data approach [5]. Here, some ranks are selected to do only the PME mesh calculation, while the other ranks, called particle-particle (PP) ranks, do all the rest of the work. For rectangular boxes the optimal PP to PME rank ratio is usually 3:1,

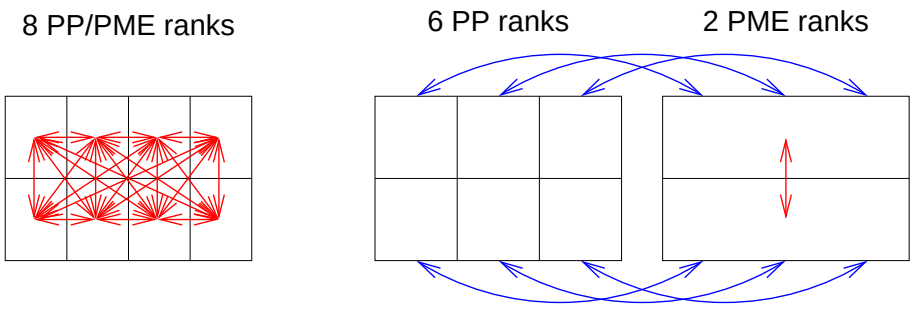

Figure 3.15: Example of 8 ranks without (left) and with (right) MPMD. The PME communication (red arrows) is much higher on the left than on the right. For MPMD additional PP - PME coordinate and force communication (blue arrows) is required, but the total communication complexity is lower.

for rhombic dodecahedra usually 2:1. When the number of PME ranks is reduced by a factor of 4, the number of communication calls is reduced by about a factor of 16. Or put differently, we can now scale to 4 times more ranks. In addition, for modern 4 or 8 core machines in a network, the effective network bandwidth for PME is quadrupled, since only a quarter of the cores will be using the network connection on each machine during the PME calculations.

`mdrun` will by default interleave the PP and PME ranks. If the ranks are not number consecutively inside the machines, one might want to use `mdrun -ddorder pp_pme`. For machines with a real 3-D torus and proper communication software that assigns the ranks accordingly one should use `mdrun -ddorder cartesian`.

To optimize the performance one should usually set up the cut-offs and the PME grid such that the PME load is 25 to 33% of the total calculation load. `grompp` will print an estimate for this load at the end and also `mdrun` calculates the same estimate to determine the optimal number of PME ranks to use. For high parallelization it might be worthwhile to optimize the PME load with the `mdp` settings and/or the number of PME ranks with the `-npme` option of `mdrun`. For changing the electrostatics settings it is useful to know the accuracy of the electrostatics remains nearly constant when the Coulomb cut-off and the PME grid spacing are scaled by the same factor. **Note** that it is usually better to overestimate than to underestimate the number of PME ranks, since the number of PME ranks is smaller than the number of PP ranks, which leads to less total waiting time.

The PME domain decomposition can be 1-D or 2-D along the x and/or y axis. 2-D decomposition is also known as pencil decomposition because of the shape of the domains at high parallelization. 1-D decomposition along the y axis can only be used when the PP decomposition has only 1 domain along x. 2-D PME decomposition has to have the number of domains along x equal to the number of the PP decomposition. `mdrun` automatically chooses 1-D or 2-D PME decomposition (when possible with the total given number of ranks), based on the minimum amount of communication for the coordinate redistribution in PME plus the communication for the grid overlap and transposes. To avoid superfluous communication of coordinates and forces between the PP and PME ranks, the number of DD cells in the x direction should ideally be the same or a multiple of the number of PME ranks. By default, `mdrun` takes care of this issue.

3.17.6 Domain decomposition flow chart

In Fig. 3.16 a flow chart is shown for domain decomposition with all possible communication for different algorithms. For simpler simulations, the same flow chart applies, without the algorithms and communication for the algorithms that are not used.

3.18 Implicit solvation

Implicit solvent models provide an efficient way of representing the electrostatic effects of solvent molecules, while saving a large piece of the computations involved in an accurate, aqueous description of the surrounding water in molecular dynamics simulations. Implicit solvation models offer several advantages compared with explicit solvation, including eliminating the need for the equilibration of water around the solute, and the absence of viscosity, which allows the protein to more quickly explore conformational space.

Implicit solvent calculations in GROMACS can be done using the generalized Born-formalism, and the Still [70], HCT [71], and OBC [72] models are available for calculating the Born radii.

Here, the free energy G_{solv} of solvation is the sum of three terms, a solvent-solvent cavity term (G_{cav}), a solute-solvent van der Waals term (G_{vdw}), and finally a solvent-solute electrostatics polarization term (G_{pol}).

The sum of G_{cav} and G_{vdw} corresponds to the (non-polar) free energy of solvation for a molecule from which all charges have been removed, and is commonly called G_{np}, calculated from the total solvent accessible surface area multiplied with a surface tension. The total expression for the solvation free energy then becomes:

$$G_{\mathrm{solv}} = G_{\mathrm{np}} + G_{\mathrm{pol}} \tag{3.148}$$

Under the generalized Born model, G_{pol} is calculated from the generalized Born equation [70]:

$$G_{\mathrm{pol}} = \left(1 - \frac{1}{\epsilon}\right) \sum_{i=1}^{n} \sum_{j>i}^{n} \frac{q_i q_j}{\sqrt{r_{ij}^2 + b_i b_j \exp\left(\frac{-r_{ij}^2}{4 b_i b_j}\right)}} \tag{3.149}$$

In GROMACS, we have introduced the substitution [73]:

$$c_i = \frac{1}{\sqrt{b_i}} \tag{3.150}$$

which makes it possible to introduce a cheap transformation to a new variable x when evaluating each interaction, such that:

$$x = \frac{r_{ij}}{\sqrt{b_i b_j}} = r_{ij} c_i c_j \tag{3.151}$$

In the end, the full re-formulation of 3.149 becomes:

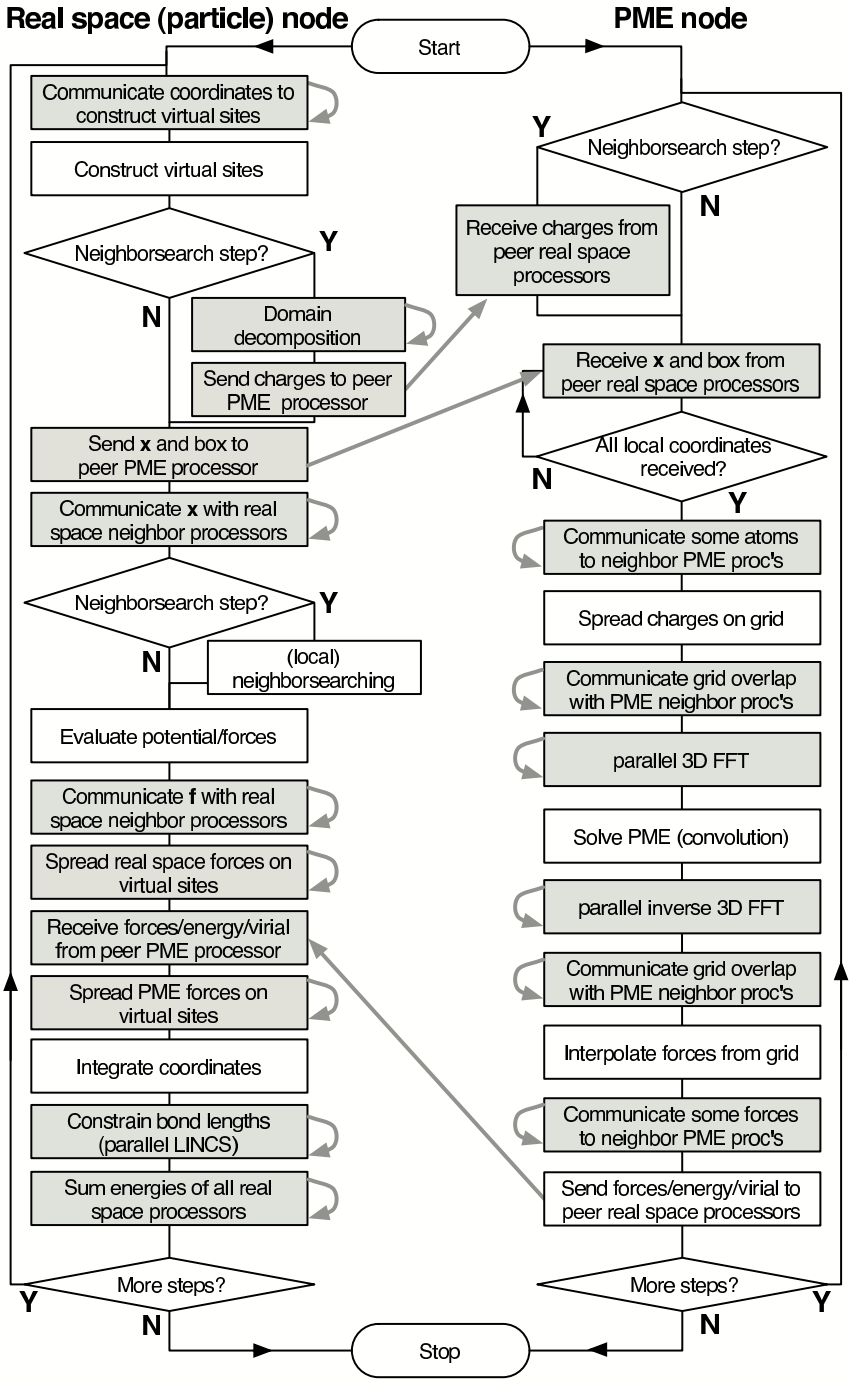

Figure 3.16: Flow chart showing the algorithms and communication (arrows) for a standard MD simulation with virtual sites, constraints and separate PME-mesh ranks.

Verlet does allow the calculation, at the cost of an extra round of global communication, and can compute, mod any integration errors, the true NPT ensemble.

The full equations, combining both pressure coupling and temperature coupling, are taken from Martyna *et al.* [34] and Tuckerman [39] and are referred to here as MTTK equations (Martyna-Tuckerman-Tobias-Klein). We introduce for convenience $\epsilon = (1/3)\ln(V/V_0)$, where V_0 is a reference volume. The momentum of ϵ is $v_\epsilon = p_\epsilon/W = \dot{\epsilon} = \dot{V}/3V$, and define $\alpha = 1 + 3/N_{dof}$ (see Ref [39])

The isobaric equations are

$$\begin{aligned}
\dot{\boldsymbol{r}}_i &= \frac{\boldsymbol{p}_i}{m_i} + \frac{p_\epsilon}{W}\boldsymbol{r}_i \\
\frac{\dot{\boldsymbol{p}}_i}{m_i} &= \frac{1}{m_i}\boldsymbol{F}_i - \alpha\frac{p_\epsilon}{W}\frac{\boldsymbol{p}_i}{m_i} \\
\dot{\epsilon} &= \frac{p_\epsilon}{W} \\
\frac{\dot{p}_\epsilon}{W} &= \frac{3V}{W}(P_{\text{int}} - P) + (\alpha - 1)\left(\sum_{n=1}^{N}\frac{\boldsymbol{p}_i^2}{m_i}\right), \qquad (3.71)
\end{aligned}$$

(3.72)

where

$$P_{\text{int}} = P_{\text{kin}} - P_{\text{vir}} = \frac{1}{3V}\left[\sum_{i=1}^{N}\left(\frac{\boldsymbol{p}_i^2}{2m_i} - \boldsymbol{r}_i\cdot\boldsymbol{F}_i\right)\right]. \qquad (3.73)$$

The terms including α are required to make phase space incompressible [39]. The ϵ acceleration term can be rewritten as

$$\frac{\dot{p}_\epsilon}{W} = \frac{3V}{W}\left(\alpha P_{\text{kin}} - P_{\text{vir}} - P\right) \qquad (3.74)$$

In terms of velocities, these equations become

$$\begin{aligned}
\dot{\boldsymbol{r}}_i &= \boldsymbol{v}_i + v_\epsilon\boldsymbol{r}_i \\
\dot{\boldsymbol{v}}_i &= \frac{1}{m_i}\boldsymbol{F}_i - \alpha v_\epsilon\boldsymbol{v}_i \\
\dot{\epsilon} &= v_\epsilon \\
\dot{v_\epsilon} &= \frac{3V}{W}(P_{\text{int}} - P) + (\alpha - 1)\left(\sum_{n=1}^{N}\frac{1}{2}m_i\boldsymbol{v}_i^2\right) \\
P_{\text{int}} &= P_{\text{kin}} - P_{\text{vir}} = \frac{1}{3V}\left[\sum_{i=1}^{N}\left(\frac{1}{2}m_i\boldsymbol{v}_i^2 - \boldsymbol{r}_i\cdot\boldsymbol{F}_i\right)\right] \qquad (3.75)
\end{aligned}$$

For these equations, the conserved quantity is

$$H = \sum_{i=1}^{N}\frac{\boldsymbol{p}_i^2}{2m_i} + U\left(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_N\right) + \frac{p_\epsilon}{2W} + PV \qquad (3.76)$$

$$G_{\mathrm{pol}} = \left(1 - \frac{1}{\epsilon}\right) \sum_{i=1}^{n} \sum_{j>i}^{n} \frac{q_i q_j}{\sqrt{b_i b_j}} \, \xi(x) = \left(1 - \frac{1}{\epsilon}\right) \sum_{i=1}^{n} q_i c_i \sum_{j>i}^{n} q_j c_j \, \xi(x) \tag{3.152}$$

The non-polar part (G_{np}) of Equation 3.148 is calculated directly from the Born radius of each atom using a simple ACE type approximation by Schaefer *et al.* [74], including a simple loop over all atoms. This requires only one extra solvation parameter, independent of atom type, but differing slightly between the three Born radii models.