3.4 Molecular Dynamics

A global flow scheme for MD is given in Fig. 3.3. Each MD or EM run requires as input a set of initial coordinates and – optionally – initial velocities of all particles involved. This chapter does not describe how these are obtained; for the setup of an actual MD run check the online manual at www.gromacs.org.

THE GLOBAL MD ALGORITHM

1. Input initial conditions

Potential interaction V as a function of atom positions
Positions $\boldsymbol{r}$ of all atoms in the system
Velocities $\boldsymbol{v}$ of all atoms in the system
$\Downarrow$

repeat 2,3,4 for the required number of steps:

2. Compute forces

The force on any atom

$$\boldsymbol{F}_i = -\frac{\partial V}{\partial \boldsymbol{r}_i}$$

is computed by calculating the force between non-bonded atom pairs:

$$\boldsymbol{F}_i = \sum_j \boldsymbol{F}_{ij}$$

plus the forces due to bonded interactions (which may depend on 1, 2, 3, or 4 atoms), plus restraining and/or external forces.
The potential and kinetic energies and the pressure tensor may be computed.
$\Downarrow$

3. Update configuration

The movement of the atoms is simulated by numerically solving Newton's equations of motion

$$\frac{\mathrm{d}^2\boldsymbol{r}_i}{\mathrm{d}t^2} = \frac{\boldsymbol{F}_i}{m_i}$$

or

$$\frac{\mathrm{d}\boldsymbol{r}_i}{\mathrm{d}t} = \boldsymbol{v}_i; \quad \frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = \frac{\boldsymbol{F}_i}{m_i}$$

$\Downarrow$

4. if required: **Output step**
write positions, velocities, energies, temperature, pressure, etc.

Figure 3.3: The global MD algorithm

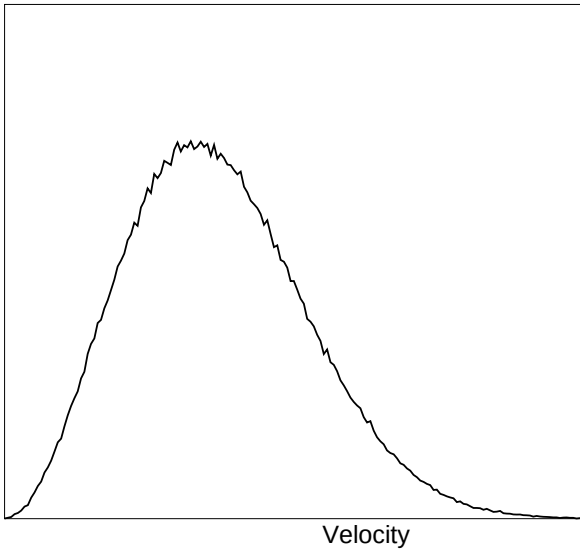

Figure 3.4: A Maxwell-Boltzmann velocity distribution, generated from random numbers.

3.4.1 Initial conditions

Topology and force field

The system topology, including a description of the force field, must be read in. Force fields and topologies are described in chapter 4 and 5, respectively. All this information is static; it is never modified during the run.

Coordinates and velocities

Then, before a run starts, the box size and the coordinates and velocities of all particles are required. The box size and shape is determined by three vectors (nine numbers) $\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3$, which represent the three basis vectors of the periodic box.

If the run starts at $t = t_0$, the coordinates at $t = t_0$ must be known. The *leap-frog algorithm*, the default algorithm used to update the time step with Δt (see 3.4.4), also requires that the velocities at $t = t_0 - \frac{1}{2}\Delta t$ are known. If velocities are not available, the program can generate initial atomic velocities $v_i, i = 1 \ldots 3N$ with a (Fig. 3.4) at a given absolute temperature T:

$$p(v_i) = \sqrt{\frac{m_i}{2\pi kT}} \exp\left(-\frac{m_i v_i^2}{2kT}\right) \tag{3.7}$$

where k is Boltzmann's constant (see chapter 2). To accomplish this, normally distributed random numbers are generated by adding twelve random numbers R_k in the range $0 \leq R_k < 1$ and subtracting 6.0 from their sum. The result is then multiplied by the standard deviation of the velocity distribution $\sqrt{kT/m_i}$. Since the resulting total energy will not correspond exactly to the required temperature T, a correction is made: first the center-of-mass motion is removed and then all velocities are scaled so that the total energy corresponds exactly to T (see eqn. 3.18).

Center-of-mass motion

The center-of-mass velocity is normally set to zero at every step; there is (usually) no net external force acting on the system and the center-of-mass velocity should remain constant. In practice, however, the update algorithm introduces a very slow change in the center-of-mass velocity, and therefore in the total kinetic energy of the system – especially when temperature coupling is used. If such changes are not quenched, an appreciable center-of-mass motion can develop in long runs, and the temperature will be significantly misinterpreted. Something similar may happen due to overall rotational motion, but only when an isolated cluster is simulated. In periodic systems with filled boxes, the overall rotational motion is coupled to other degrees of freedom and does not cause such problems.

3.4.2 Neighbor searching

As mentioned in chapter 4, internal forces are either generated from fixed (static) lists, or from dynamic lists. The latter consist of non-bonded interactions between any pair of particles. When calculating the non-bonded forces, it is convenient to have all particles in a rectangular box. As shown in Fig. 3.1, it is possible to transform a triclinic box into a rectangular box. The output coordinates are always in a rectangular box, even when a dodecahedron or triclinic box was used for the simulation. Equation 3.1 ensures that we can reset particles in a rectangular box by first shifting them with box vector $\mathbf{c}$, then with $\mathbf{b}$ and finally with $\mathbf{a}$. Equations 3.3, 3.4 and 3.5 ensure that we can find the 14 nearest triclinic images within a linear combination that does not involve multiples of box vectors.

Pair lists generation

The non-bonded pair forces need to be calculated only for those pairs i, j for which the distance r_{ij} between i and the nearest image of j is less than a given cut-off radius R_c. Some of the particle pairs that fulfill this criterion are excluded, when their interaction is already fully accounted for by bonded interactions. GROMACS employs a *pair list* that contains those particle pairs for which non-bonded forces must be calculated. The pair list contains atoms i, a displacement vector for atom i, and all particles j that are within `rlist` of this particular image of atom i. The list is updated every `nstlist` steps, where `nstlist` is typically 10. There is an option to calculate the total non-bonded force on each particle due to all particle in a shell around the list cut-off, *i.e.* at a distance between `rlist` and `rlistlong`. This force is calculated during the pair list update and retained during `nstlist` steps.

To make the neighbor list, all particles that are close (*i.e.* within the neighbor list cut-off) to a given

particle must be found. This searching, usually called neighbor search (NS) or pair search, involves periodic boundary conditions and determining the *image* (see sec. 3.2). The search algorithm is $O(N)$, although a simpler $O(N^2)$ algorithm is still available under some conditions.

Cut-off schemes: group versus Verlet

From version 4.6, GROMACS supports two different cut-off scheme setups: the original one based on atom groups and one using a Verlet buffer. There are some important differences that affect results, performance and feature support. The group scheme can be made to work (almost) like the Verlet scheme, but this will lead to a decrease in performance. The group scheme is especially fast for water molecules, which are abundant in many simulations, but on the most recent x86 processors, this advantage is negated by the better instruction-level parallelism available in the Verlet-scheme implementation. The group scheme is deprecated in version 5.0, and will be removed in a future version.

In the group scheme, a neighbor list is generated consisting of pairs of groups of at least one atom. These groups were originally charge groups (see sec. 3.4.2), but with a proper treatment of long-range electrostatics, performance is their only advantage. A pair of groups is put into the neighbor list when their center of geometry is within the cut-off distance. Interactions between all atom pairs (one from each charge group) are calculated for a certain number of MD steps, until the neighbor list is updated. This setup is efficient, as the neighbor search only checks distance between charge group pair, not atom pairs (saves a factor of $3 \times 3 = 9$ with a three-atom water model) and the non-bonded force kernels can be optimized for, say, a water molecule "group". Without explicit buffering, this setup leads to energy drift as some atom pairs which are within the cut-off don't interact and some outside the cut-off do interact. This can be caused by

- atoms moving across the cut-off between neighbor search steps, and/or
- for charge groups consisting of more than one atom, atom pairs moving in/out of the cut-off when their charge group center of geometry distance is outside/inside of the cut-off.

Explicitly adding a buffer to the neighbor list will remove such artifacts, but this comes at a high computational cost. How severe the artifacts are depends on the system, the properties in which you are interested, and the cut-off setup.

The Verlet cut-off scheme uses a buffered pair list by default. It also uses clusters of atoms, but these are not static as in the group scheme. Rather, the clusters are defined spatially and consist of 4 or 8 atoms, which is convenient for stream computing, using e.g. SSE, AVX or CUDA on GPUs. At neighbor search steps, a pair list is created with a Verlet buffer, ie. the pair-list cut-off is larger than the interaction cut-off. In the non-bonded force kernels, forces are only added when an atom pair is within the cut-off distance at that particular time step. This ensures that as atoms move between pair search steps, forces between nearly all atoms within the cut-off distance are calculated. We say *nearly* all atoms, because GROMACS uses a fixed pair list update frequency for efficiency. An atom-pair, whose distance was outside the cut-off, could possibly move enough during this fixed number of steps that its distance is now within the cut-off. This small chance results in a small energy drift, and the size of the chance depends on the temperature. When temperature coupling is used, the buffer size can be determined automatically, given a certain tolerance on the energy drift.

The mdp file settings specific to the Verlet scheme are:

```
cutoff-scheme            = Verlet
verlet-buffer-tolerance = 0.005
```

The Verlet buffer size is determined from the latter option, which is by default set to 0.005 kJ/mol/ps pair energy error per atom. Note that errors in pair energies cancel and the effect on the total energy drift is usually at least an order of magnitude smaller than the tolerance. Furthermore, the drift of the total energy is affected by many other factors; often, the contribution from the constraint algorithm dominates.

For constant-energy (NVE) simulations, the buffer size will be inferred from the temperature that corresponds to the velocities (either those generated, if applicable, or those found in the input configuration). Alternatively, the tolerance can be set to -1 and a buffer set manually by specifying `rlist` > `max(rcoulomb, rvdw)`. The simplest way to get a reasonable buffer size is to use an NVT mdp file with the target temperature set to what you expect in your NVE simulation, and transfer the buffer size printed by `grompp` to your NVE mdp file.

The Verlet cut-off scheme is implemented in a very efficient fashion based on clusters of particles. The simplest example is a cluster size of 4 particles. The pair list is then constructed based on cluster pairs. The cluster-pair search is much faster searching based on particle pairs, because $4 \times 4 = 16$ particle pairs are put in the list at once. The non-bonded force calculation kernel can then calculate all 16 particle-pair interactions at once, which maps nicely to SIMD units which can perform multiple floating operations at once (e.g. SSE, AVX, CUDA on GPUs, BlueGene FPUs). These non-bonded kernels are much faster than the kernels used in the group scheme for most types of systems, except for water molecules on processors with short SIMD widths when not using a buffered pair list. This latter case is common for (bio-)molecular simulations, so for greatest speed, it is worth comparing the performance of both schemes.

As the Verlet cut-off scheme was introduced in version 4.6, not all features of the group scheme are supported yet. The Verlet scheme supports a few new features which the group scheme does not support. A list of features not (fully) supported in both cut-off schemes is given in Table 3.2.

Energy drift and pair-list buffering

For a canonical (NVT) ensemble, the average energy error caused by the finite Verlet buffer size can be determined from the atomic displacements and the shape of the potential at the cut-off. The displacement distribution along one dimension for a freely moving particle with mass m over time t at temperature T is Gaussian with zero mean and variance $\sigma^2 = t\,k_B T/m$. For the distance between two atoms, the variance changes to $\sigma^2 = \sigma_{12}^2 = t\,k_B T(1/m_1 + 1/m_2)$. Note that in practice particles usually interact with other particles over time t and therefore the real displacement distribution is much narrower. Given a non-bonded interaction cut-off distance of r_c and a pair-list cut-off $r_\ell = r_c + r_b$, we can then write the average energy error after time t for pair interactions between one particle of type 1 surrounded by particles of type 2 with number density ρ_2, when the inter particle distance changes from r_0 to r_t, as:

$$\langle \Delta V \rangle \;=\; \int_0^{r_c} \int_{r_\ell}^{\infty} 4\pi r_0^2 \rho_2 V(r_t) G\left(\frac{r_t - r_0}{\sigma}\right) dr_0\, dr_t \tag{3.8}$$

Non-bonded interaction feature	group	Verlet
unbuffered cut-off scheme	√	not by default
exact cut-off	shift/switch	√
shifted interactions	force+energy	energy
switched potential	√	√
switched forces	√	√
non-periodic systems	√	Z + walls
implicit solvent	√	
free energy perturbed non-bondeds	√	√
group energy contributions	√	CPU (not on GPU)
energy group exclusions	√	
AdResS multi-scale	√	
OpenMP multi-threading	only PME	√
native GPU support		√
Lennard-Jones PME	√	√

Table 3.2: Differences (only) in the support of non-bonded features between the group and Verlet cut-off schemes.

$$\approx \int_{-\infty}^{r_c}\int_{r_\ell}^{\infty} 4\pi r_0^2\rho_2\Big[V'(r_c)(r_t-r_c)+ V''(r_c)\frac{1}{2}(r_t-r_c)^2\Big]G\Big(\frac{r_t-r_0}{\sigma}\Big)\,dr_0\,dr_t \tag{3.9}$$

$$\approx 4\pi(r_\ell+\sigma)^2\rho_2\int_{-\infty}^{r_c}\int_{r_\ell}^{\infty}\Big[V'(r_c)(r_t-r_c)+ V''(r_c)\frac{1}{2}(r_t-r_c)^2+ V'''(r_c)\frac{1}{6}(r_t-r_c)^3\Big]G\Big(\frac{r_t-r_0}{\sigma}\Big)\,dr_0\,dr_t \tag{3.10}$$

$$= 4\pi(r_\ell+\sigma)^2\rho_2\Big\{\frac{1}{2}V'(r_c)\Big[r_b\sigma G\Big(\frac{r_b}{\sigma}\Big)-(r_b^2+\sigma^2)E\Big(\frac{r_b}{\sigma}\Big)\Big]+ \frac{1}{6}V''(r_c)\Big[\sigma(r_b^2+2\sigma^2)G\Big(\frac{r_b}{\sigma}\Big)-r_b(r_b^2+3\sigma^2)E\Big(\frac{r_b}{\sigma}\Big)\Big]+ \frac{1}{24}V'''(r_c)\Big[r_b\sigma(r_b^2+5\sigma^2)G\Big(\frac{r_b}{\sigma}\Big)-(r_b^4+6r_b^2\sigma^2+3\sigma^4)E\Big(\frac{r_b}{\sigma}\Big)\Big]\Big\} \tag{3.11}$$

where G is a Gaussian distribution with 0 mean and unit variance and $E(x) = \frac{1}{2}\mathrm{erfc}(x/\sqrt{2})$. We always want to achieve small energy error, so σ will be small compared to both r_c and r_ℓ, thus the approximations in the equations above are good, since the Gaussian distribution decays rapidly. The energy error needs to be averaged over all particle pair types and weighted with the particle counts. In GROMACS we don't allow cancellation of error between pair types, so we average the absolute values. To obtain the average energy error per unit time, it needs to be divided by the neighbor-list life time $t = (\texttt{nstlist}-1)\times\texttt{dt}$. This function can not be inverted analytically, so we use bisection to obtain the buffer size r_b for a target drift. Again we note that in practice the error we usually be much smaller than this estimate, as in the condensed phase particle

displacements will be much smaller than for freely moving particles, which is the assumption used here.

When (bond) constraints are present, some particles will have fewer degrees of freedom. This will reduce the energy errors. The displacement in an arbitrary direction of a particle with 2 degrees of freedom is not Gaussian, but rather follows the complementary error function:

$$\frac{\sqrt{\pi}}{2\sqrt{2}\sigma}\,\mathrm{erfc}\left(\frac{|r|}{\sqrt{2}\,\sigma}\right) \tag{3.12}$$

where σ^2 is again $k_B T/m$. This distribution can no longer be integrated analytically to obtain the energy error. But we can generate a tight upper bound using a scaled and shifted Gaussian distribution (not shown). This Gaussian distribution can then be used to calculate the energy error as described above. We consider particles constrained, i.e. having 2 degrees of freedom or fewer, when they are connected by constraints to particles with a total mass of at least 1.5 times the mass of the particles itself. For a particle with a single constraint this would give a total mass along the constraint direction of at least 2.5, which leads to a reduction in the variance of the displacement along that direction by at least a factor of 6.25. As the Gaussian distribution decays very rapidly, this effectively removes one degree of freedom from the displacement. Multiple constraints would reduce the displacement even more, but as this gets very complex, we consider those as particles with 2 degrees of freedom.

There is one important implementation detail that reduces the energy errors caused by the finite Verlet buffer list size. The derivation above assumes a particle pair-list. However, the GROMACS implementation uses a cluster pair-list for efficiency. The pair list consists of pairs of clusters of 4 particles in most cases, also called a 4×4 list, but the list can also be 4×8 (GPU CUDA kernels and AVX 256-bit single precision kernels) or 4×2 (SSE double-precision kernels). This means that the pair-list is effectively much larger than the corresponding 1×1 list. Thus slightly beyond the pair-list cut-off there will still be a large fraction of particle pairs present in the list. This fraction can be determined in a simulation and accurately estimated under some reasonable assumptions. The fraction decreases with increasing pair-list range, meaning that a smaller buffer can be used. For typical all-atom simulations with a cut-off of 0.9 nm this fraction is around 0.9, which gives a reduction in the energy errors of a factor of 10. This reduction is taken into account during the automatic Verlet buffer calculation and results in a smaller buffer size.

In Fig. 3.5 one can see that for small buffer sizes the drift of the total energy is much smaller than the pair energy error tolerance, due to cancellation of errors. For larger buffer size, the error estimate is a factor of 6 higher than drift of the total energy, or alternatively the buffer estimate is 0.024 nm too large. This is because the protons don't move freely over 18 fs, but rather vibrate.

Cut-off artifacts and switched interactions

With the Verlet scheme, the pair potentials are shifted to be zero at the cut-off, which makes the potential the integral of the force. This is only possible in the group scheme if the shape of the potential is such that its value is zero at the cut-off distance. However, there can still be energy drift when the forces are non-zero at the cut-off. This effect is extremely small and often not noticeable, as other integration errors (e.g. from constraints) may dominate. To completely avoid cut-off artifacts, the non-bonded forces can be switched exactly to zero at some distance smaller

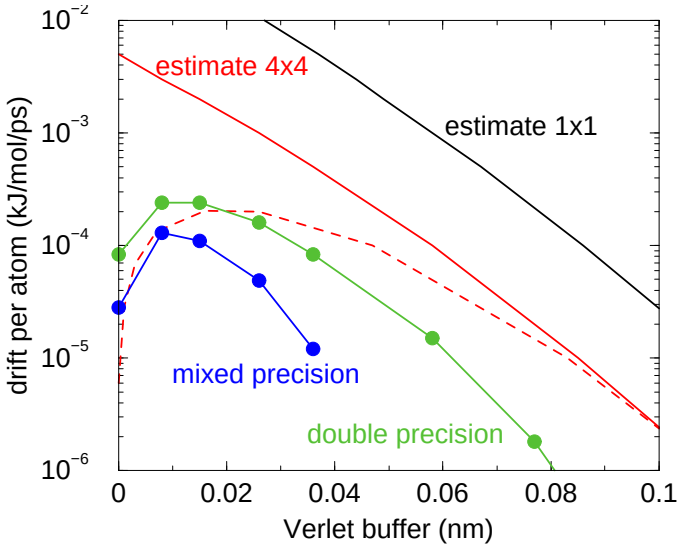

Figure 3.5: Energy drift per atom for an SPC/E water system at 300K with a time step of 2 fs and a pair-list update period of 10 steps (pair-list life time: 18 fs). PME was used with `ewald-rtol` set to 10^{-5}; this parameter affects the shape of the potential at the cut-off. Error estimates due to finite Verlet buffer size are shown for a 1×1 atom pair list and 4×4 atom pair list without and with (dashed line) cancellation of positive and negative errors. Real energy drift is shown for simulations using double- and mixed-precision settings. Rounding errors in the SETTLE constraint algorithm from the use of single precision causes the drift to become negative at large buffer size. Note that at zero buffer size, the real drift is small because positive (H-H) and negative (O-H) energy errors cancel.

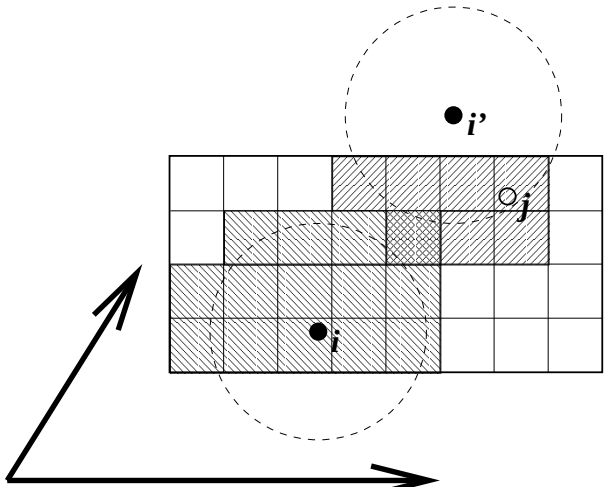

Figure 3.6: Grid search in two dimensions. The arrows are the box vectors.

than the neighbor list cut-off (there are several ways to do this in GROMACS, see sec. 4.1.5). One then has a buffer with the size equal to the neighbor list cut-off less the longest interaction cut-off.

With the group cut-off scheme, one can also choose to let `mdrun` only update the neighbor list when required. That is when one or more particles have moved more than half the buffer size from the center of geometry of the charge group to which they belong (see sec. 3.4.2), as determined at the previous neighbor search. This option guarantees that there are no cut-off artifacts. **Note** that for larger systems this comes at a high computational cost, since the neighbor list update frequency will be determined by just one or two particles moving slightly beyond the half buffer length (which not even necessarily implies that the neighbor list is invalid), while 99.99% of the particles are fine.

Simple search

Due to eqns. 3.1 and 3.6, the vector $\boldsymbol{r}_{ij}$ connecting images within the cut-off R_c can be found by constructing:

$$\begin{aligned} \boldsymbol{r}''' &= \boldsymbol{r}_j - \boldsymbol{r}_i & (3.13)\\ \boldsymbol{r}'' &= \boldsymbol{r}''' - \mathbf{c} * \texttt{round}(r'''_z/c_z) & (3.14)\\ \boldsymbol{r}' &= \boldsymbol{r}'' - \mathbf{b} * \texttt{round}(r''_y/b_y) & (3.15)\\ \boldsymbol{r}_{ij} &= \boldsymbol{r}' - \mathbf{a} * \texttt{round}(r'_x/a_x) & (3.16) \end{aligned}$$

When distances between two particles in a triclinic box are needed that do not obey eqn. 3.1, many shifts of combinations of box vectors need to be considered to find the nearest image.

Grid search

The grid search is schematically depicted in Fig. 3.6. All particles are put on the NS grid, with the smallest spacing $\geq R_c/2$ in each of the directions. In the direction of each box vector, a particle

i has three images. For each direction the image may be -1,0 or 1, corresponding to a translation over -1, 0 or +1 box vector. We do not search the surrounding NS grid cells for neighbors of i and then calculate the image, but rather construct the images first and then search neighbors corresponding to that image of i. As Fig. 3.6 shows, some grid cells may be searched more than once for different images of i. This is not a problem, since, due to the minimum image convention, at most one image will "see" the j-particle. For every particle, fewer than 125 (5^3) neighboring cells are searched. Therefore, the algorithm scales linearly with the number of particles. Although the prefactor is large, the scaling behavior makes the algorithm far superior over the standard $O(N^2)$ algorithm when there are more than a few hundred particles. The grid search is equally fast for rectangular and triclinic boxes. Thus for most protein and peptide simulations the rhombic dodecahedron will be the preferred box shape.

Charge groups

Charge groups were originally introduced to reduce cut-off artifacts of Coulomb interactions. When a plain cut-off is used, significant jumps in the potential and forces arise when atoms with (partial) charges move in and out of the cut-off radius. When all chemical moieties have a net charge of zero, these jumps can be reduced by moving groups of atoms with net charge zero, called charge groups, in and out of the neighbor list. This reduces the cut-off effects from the charge-charge level to the dipole-dipole level, which decay much faster. With the advent of full range electrostatics methods, such as particle-mesh Ewald (sec. 4.8.2), the use of charge groups is no longer required for accuracy. It might even have a slight negative effect on the accuracy or efficiency, depending on how the neighbor list is made and the interactions are calculated.

But there is still an important reason for using "charge groups": efficiency with the group cut-off scheme. Where applicable, neighbor searching is carried out on the basis of charge groups which are defined in the molecular topology. If the nearest image distance between the *geometrical centers* of the atoms of two charge groups is less than the cut-off radius, all atom pairs between the charge groups are included in the pair list. The neighbor searching for a water system, for instance, is $3^2 = 9$ times faster when each molecule is treated as a charge group. Also the highly optimized water force loops (see sec. B.2.1) only work when all atoms in a water molecule form a single charge group. Currently the name *neighbor-search group* would be more appropriate, but the name charge group is retained for historical reasons. When developing a new force field, the advice is to use charge groups of 3 to 4 atoms for optimal performance. For all-atom force fields this is relatively easy, as one can simply put hydrogen atoms, and in some case oxygen atoms, in the same charge group as the heavy atom they are connected to; for example: CH_3, CH_2, CH, NH_2, NH, OH, CO_2, CO.

With the Verlet cut-off scheme, charge groups are ignored.

3.4.3 Compute forces

Potential energy

When forces are computed, the potential energy of each interaction term is computed as well. The total potential energy is summed for various contributions, such as Lennard-Jones, Coulomb, and

bonded terms. It is also possible to compute these contributions for *energy-monitor groups* of atoms that are separately defined (see sec. 3.3).

Kinetic energy and temperature

The temperature is given by the total kinetic energy of the N-particle system:

$$E_{kin} = \frac{1}{2}\sum_{i=1}^{N} m_i v_i^2 \tag{3.17}$$

From this the absolute temperature T can be computed using:

$$\frac{1}{2} N_{\mathrm{df}} kT = E_{\mathrm{kin}} \tag{3.18}$$

where k is Boltzmann's constant and N_{df} is the number of degrees of freedom which can be computed from:

$$N_{\mathrm{df}} \;=\; 3N - N_c - N_{\mathrm{com}} \tag{3.19}$$

Here N_c is the number of *constraints* imposed on the system. When performing molecular dynamics $N_{\mathrm{com}} = 3$ additional degrees of freedom must be removed, because the three center-of-mass velocities are constants of the motion, which are usually set to zero. When simulating in vacuo, the rotation around the center of mass can also be removed, in this case $N_{\mathrm{com}} = 6$. When more than one temperature-coupling group is used, the number of degrees of freedom for group i is:

$$N_{\mathrm{df}}^i \;=\; (3N^i - N_c^i)\frac{3N - N_c - N_{\mathrm{com}}}{3N - N_c} \tag{3.20}$$

The kinetic energy can also be written as a tensor, which is necessary for pressure calculation in a triclinic system, or systems where shear forces are imposed:

$$\mathbf{E}_{\mathrm{kin}} = \frac{1}{2}\sum_{i}^{N} m_i \boldsymbol{v}_i \otimes \boldsymbol{v}_i \tag{3.21}$$

Pressure and virial

The pressure tensor $\mathbf{P}$ is calculated from the difference between kinetic energy E_{kin} and the virial $\boldsymbol{\Xi}$:

$$\mathbf{P} = \frac{2}{V}(\mathbf{E}_{\mathrm{kin}} - \boldsymbol{\Xi}) \tag{3.22}$$

where V is the volume of the computational box. The scalar pressure P, which can be used for pressure coupling in the case of isotropic systems, is computed as:

$$P = \mathrm{trace}(\mathbf{P})/3 \tag{3.23}$$

The virial $\boldsymbol{\Xi}$ tensor is defined as:

$$\boldsymbol{\Xi} = -\frac{1}{2}\sum_{i<j} \boldsymbol{r}_{ij} \otimes \boldsymbol{F}_{ij} \tag{3.24}$$

The GROMACS implementation of the virial computation is described in sec. B.1.

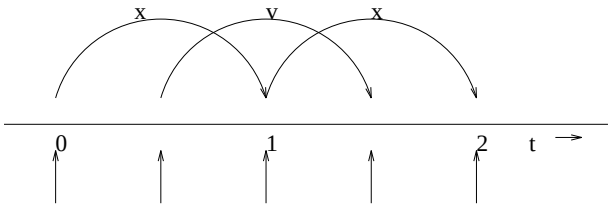

Figure 3.7: The Leap-Frog integration method. The algorithm is called Leap-Frog because $\boldsymbol{r}$ and $\boldsymbol{v}$ are leaping like frogs over each other's backs.

3.4.4 The leap-frog integrator

The default MD integrator in GROMACS is the so-called *leap-frog* algorithm [20] for the integration of the equations of motion. When extremely accurate integration with temperature and/or pressure coupling is required, the velocity Verlet integrators are also present and may be preferable (see 3.4.5). The leap-frog algorithm uses positions $\boldsymbol{r}$ at time t and velocities $\boldsymbol{v}$ at time $t - \frac{1}{2}\Delta t$; it updates positions and velocities using the forces $\boldsymbol{F}(t)$ determined by the positions at time t using these relations:

$$\boldsymbol{v}(t + \frac{1}{2}\Delta t) = \boldsymbol{v}(t - \frac{1}{2}\Delta t) + \frac{\Delta t}{m}\boldsymbol{F}(t) \tag{3.25}$$

$$\boldsymbol{r}(t + \Delta t) = \boldsymbol{r}(t) + \Delta t \boldsymbol{v}(t + \frac{1}{2}\Delta t) \tag{3.26}$$

The algorithm is visualized in Fig. 3.7. It produces trajectories that are identical to the Verlet [21] algorithm, whose position-update relation is

$$\boldsymbol{r}(t + \Delta t) = 2\boldsymbol{r}(t) - \boldsymbol{r}(t - \Delta t) + \frac{1}{m}\boldsymbol{F}(t)\Delta t^2 + O(\Delta t^4) \tag{3.27}$$

The algorithm is of third order in $\boldsymbol{r}$ and is time-reversible. See ref. [22] for the merits of this algorithm and comparison with other time integration algorithms.

The equations of motion are modified for temperature coupling and pressure coupling, and extended to include the conservation of constraints, all of which are described below.

3.4.5 The velocity Verlet integrator

The velocity Verlet algorithm [23] is also implemented in GROMACS, though it is not yet fully integrated with all sets of options. In velocity Verlet, positions $\boldsymbol{r}$ and velocities $\boldsymbol{v}$ at time t are used to integrate the equations of motion; velocities at the previous half step are not required.

$$\boldsymbol{v}(t + \frac{1}{2}\Delta t) = \boldsymbol{v}(t) + \frac{\Delta t}{2m}\boldsymbol{F}(t) \tag{3.28}$$

$$\boldsymbol{r}(t + \Delta t) = \boldsymbol{r}(t) + \Delta t\, \boldsymbol{v}(t + \frac{1}{2}\Delta t) \tag{3.29}$$

$$\boldsymbol{v}(t + \Delta t) = \boldsymbol{v}(t + \frac{1}{2}\Delta t) + \frac{\Delta t}{2m}\boldsymbol{F}(t + \Delta t) \tag{3.30}$$

or, equivalently,

$$\boldsymbol{r}(t+\Delta t) = \boldsymbol{r}(t) + \Delta t\, \boldsymbol{v} + \frac{\Delta t^2}{2m}\boldsymbol{F}(t) \tag{3.31}$$

$$\boldsymbol{v}(t+\Delta t) = \boldsymbol{v}(t) + \frac{\Delta t}{2m}\left[\boldsymbol{F}(t) + \boldsymbol{F}(t+\Delta t)\right] \tag{3.32}$$

With no temperature or pressure coupling, and with *corresponding* starting points, leap-frog and velocity Verlet will generate identical trajectories, as can easily be verified by hand from the equations above. Given a single starting file with the *same* starting point $\boldsymbol{x}(0)$ and $\boldsymbol{v}(0)$, leap-frog and velocity Verlet will *not* give identical trajectories, as leap-frog will interpret the velocities as corresponding to $t = -\frac{1}{2}\Delta t$, while velocity Verlet will interpret them as corresponding to the timepoint $t = 0$.

3.4.6 Understanding reversible integrators: The Trotter decomposition

To further understand the relationship between velocity Verlet and leap-frog integration, we introduce the reversible Trotter formulation of dynamics, which is also useful to understanding implementations of thermostats and barostats in GROMACS.

A system of coupled, first-order differential equations can be evolved from time $t = 0$ to time t by applying the evolution operator

$$\begin{aligned} \Gamma(t) &= \exp(iLt)\Gamma(0) \\ iL &= \dot{\Gamma}\cdot\nabla_{\Gamma}, \end{aligned} \tag{3.33}$$

where L is the Liouville operator, and Γ is the multidimensional vector of independent variables (positions and velocities). A short-time approximation to the true operator, accurate at time $\Delta t = t/P$, is applied P times in succession to evolve the system as

$$\Gamma(t) = \prod_{i=1}^{P} \exp(iL\Delta t)\Gamma(0) \tag{3.34}$$

For NVE dynamics, the Liouville operator is

$$iL = \sum_{i=1}^{N} \boldsymbol{v}_i \cdot \nabla_{\boldsymbol{r}_i} + \sum_{i=1}^{N} \frac{1}{m_i}\boldsymbol{F}(r_i)\cdot\nabla_{\boldsymbol{v}_i}. \tag{3.35}$$

This can be split into two additive operators

$$\begin{aligned} iL_1 &= \sum_{i=1}^{N} \frac{1}{m_i}\boldsymbol{F}(r_i)\cdot\nabla_{\boldsymbol{v}_i} \\ iL_2 &= \sum_{i=1}^{N} \boldsymbol{v}_i\cdot\nabla_{\boldsymbol{r}_i} \end{aligned} \tag{3.36}$$

Then a short-time, symmetric, and thus reversible approximation of the true dynamics will be

$$\exp(iL\Delta t) = \exp(iL_2\frac{1}{2}\Delta t)\exp(iL_1\Delta t)\exp(iL_2\frac{1}{2}\Delta t) + \mathcal{O}(\Delta t^3). \tag{3.37}$$

This corresponds to velocity Verlet integration. The first exponential term over $\frac{1}{2}\Delta t$ corresponds to a velocity half-step, the second exponential term over Δt corresponds to a full velocity step, and the last exponential term over $\frac{1}{2}\Delta t$ is the final velocity half step. For future times $t = n\Delta t$, this becomes

$$\begin{aligned}\exp(iLn\Delta t) &\approx \left(\exp(iL_2\frac{1}{2}\Delta t)\exp(iL_1\Delta t)\exp(iL_2\frac{1}{2}\Delta t)\right)^n \\ &\approx \exp(iL_2\frac{1}{2}\Delta t)\Big(\exp(iL_1\Delta t)\exp(iL_2\Delta t)\Big)^{n-1} \\ &\quad \exp(iL_1\Delta t)\exp(iL_2\frac{1}{2}\Delta t)\end{aligned} \tag{3.38}$$

This formalism allows us to easily see the difference between the different flavors of Verlet integrators. The leap-frog integrator can be seen as starting with Eq. 3.37 with the $\exp(iL_1\Delta t)$ term, instead of the half-step velocity term, yielding

$$\exp(iLn\Delta t) = \exp(iL_1\Delta t)\exp(iL_2\Delta t) + \mathcal{O}(\Delta t^3). \tag{3.39}$$

Here, the full step in velocity is between $t - \frac{1}{2}\Delta t$ and $t + \frac{1}{2}\Delta t$, since it is a combination of the velocity half steps in velocity Verlet. For future times $t = n\Delta t$, this becomes

$$\exp(iLn\Delta t) \approx \Big(\exp(iL_1\Delta t)\exp(iL_2\Delta t)\Big)^n. \tag{3.40}$$

Although at first this does not appear symmetric, as long as the full velocity step is between $t-\frac{1}{2}\Delta t$ and $t + \frac{1}{2}\Delta t$, then this is simply a way of starting velocity Verlet at a different place in the cycle.

Even though the trajectory and thus potential energies are identical between leap-frog and velocity Verlet, the kinetic energy and temperature will not necessarily be the same. Standard velocity Verlet uses the velocities at the t to calculate the kinetic energy and thus the temperature only at time t; the kinetic energy is then a sum over all particles

$$\begin{aligned}KE_{\text{full}}(t) &= \sum_i \left(\frac{1}{2m_i}\boldsymbol{v}_i(t)\right)^2 \\ &= \sum_i \frac{1}{2m_i}\left(\frac{1}{2}\boldsymbol{v}_i(t-\frac{1}{2}\Delta t) + \frac{1}{2}\boldsymbol{v}_i(t+\frac{1}{2}\Delta t)\right)^2,\end{aligned} \tag{3.41}$$

with the square on the *outside* of the average. Standard leap-frog calculates the kinetic energy at time t based on the average kinetic energies at the timesteps $t+\frac{1}{2}\Delta t$ and $t-\frac{1}{2}\Delta t$, or the sum over all particles

$$KE_{\text{average}}(t) = \sum_i \frac{1}{2m_i}\left(\frac{1}{2}\boldsymbol{v}_i(t-\frac{1}{2}\Delta t)^2 + \frac{1}{2}\boldsymbol{v}_i(t+\frac{1}{2}\Delta t)^2\right), \tag{3.42}$$

where the square is *inside* the average.

A non-standard variant of velocity Verlet which averages the kinetic energies $KE(t+\frac{1}{2}\Delta t)$ and $KE(t-\frac{1}{2}\Delta t)$, exactly like leap-frog, is also now implemented in GROMACS (as `.mdp` file option `md-vv-avek`). Without temperature and pressure coupling, velocity Verlet with half-step-averaged kinetic energies and leap-frog will be identical up to numerical precision. For

temperature- and pressure-control schemes, however, velocity Verlet with half-step-averaged kinetic energies and leap-frog will be different, as will be discussed in the section in thermostats and barostats.

The half-step-averaged kinetic energy and temperature are slightly more accurate for a given step size; the difference in average kinetic energies using the half-step-averaged kinetic energies (*md* and *md-vv-avek*) will be closer to the kinetic energy obtained in the limit of small step size than will the full-step kinetic energy (using *md-vv*). For NVE simulations, this difference is usually not significant, since the positions and velocities of the particles are still identical; it makes a difference in the way the the temperature of the simulations are *interpreted*, but *not* in the trajectories that are produced. Although the kinetic energy is more accurate with the half-step-averaged method, meaning that it changes less as the timestep gets large, it is also more noisy. The RMS deviation of the total energy of the system (sum of kinetic plus potential) in the half-step-averaged kinetic energy case will be higher (about twice as high in most cases) than the full-step kinetic energy. The drift will still be the same, however, as again, the trajectories are identical.

For NVT simulations, however, there *will* be a difference, as discussed in the section on temperature control, since the velocities of the particles are adjusted such that kinetic energies of the simulations, which can be calculated either way, reach the distribution corresponding to the set temperature. In this case, the three methods will not give identical results.

Because the velocity and position are both defined at the same time t the velocity Verlet integrator can be used for some methods, especially rigorously correct pressure control methods, that are not actually possible with leap-frog. The integration itself takes negligibly more time than leap-frog, but twice as many communication calls are currently required. In most cases, and especially for large systems where communication speed is important for parallelization and differences between thermodynamic ensembles vanish in the $1/N$ limit, and when only NVT ensembles are required, leap-frog will likely be the preferred integrator. For pressure control simulations where the fine details of the thermodynamics are important, only velocity Verlet allows the true ensemble to be calculated. In either case, simulation with double precision may be required to get fine details of thermodynamics correct.

3.4.7 Twin-range cut-offs

To save computation time, slowly varying forces can be calculated less often than rapidly varying forces. In GROMACS such a multiple time step splitting is possible between short and long range non-bonded interactions. In GROMACS versions up to 4.0, an irreversible integration scheme was used which is also used by the GROMOS simulation package: every n steps the long range forces are determined and these are then also used (without modification) for the next $n-1$ integration steps in eqn. 3.25. Such an irreversible scheme can result in bad energy conservation and, possibly, bad sampling. Since version 4.5, a leap-frog version of the reversible Trotter decomposition scheme [24] is used. In this integrator the long-range forces are determined every n steps and are then integrated into the velocity in eqn. 3.25 using a time step of $\Delta t_{\mathrm{LR}} = n\Delta t$:

$$\boldsymbol{v}(t+\frac{1}{2}\Delta t) = \begin{cases} \boldsymbol{v}(t-\frac{1}{2}\Delta t) + \frac{1}{m}\left[\boldsymbol{F}_{\mathrm{SR}}(t) + n\boldsymbol{F}_{\mathrm{LR}}(t)\right]\Delta t & , \quad \text{step } \% \; n = 0 \\ \boldsymbol{v}(t-\frac{1}{2}\Delta t) + \frac{1}{m}\boldsymbol{F}_{\mathrm{SR}}(t)\Delta t & , \quad \text{step } \% \; n \neq 0 \end{cases} \tag{3.43}$$

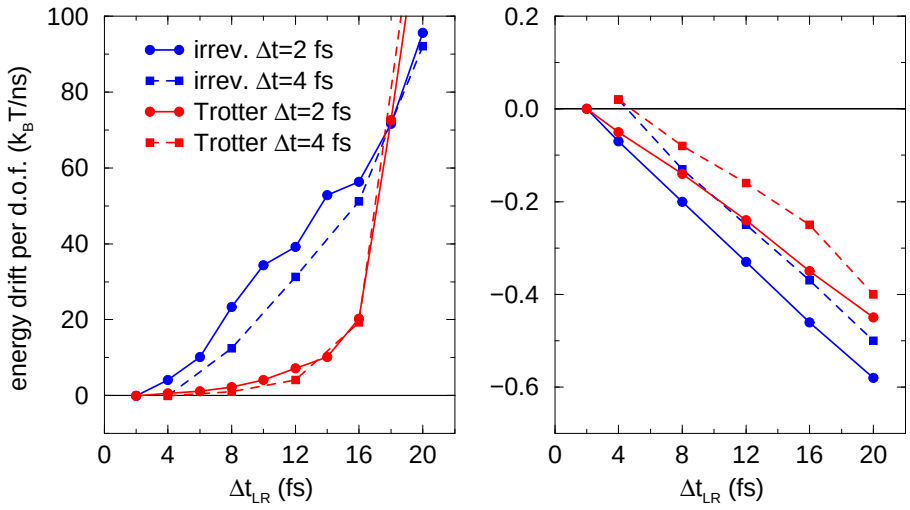

Figure 3.8: Energy drift per degree of freedom in SPC/E water with twin-range cut-offs for reaction field (left) and Lennard-Jones interaction (right) as a function of the long-range time step length for the irreversible "GROMOS" scheme and a reversible Trotter scheme.

The parameter n is equal to the neighbor list update frequency. In 4.5, the velocity Verlet version of multiple time-stepping is not yet fully implemented.

Several other simulation packages uses multiple time stepping for bonds and/or the PME mesh forces. In GROMACS we have not implemented this (yet), since we use a different philosophy. Bonds can be constrained (which is also a more sound approximation of a physical quantum oscillator), which allows the smallest time step to be increased to the larger one. This not only halves the number of force calculations, but also the update calculations. For even larger time steps, angle vibrations involving hydrogen atoms can be removed using virtual interaction sites (see sec. 6.8), which brings the shortest time step up to PME mesh update frequency of a multiple time stepping scheme.

As an example we show the energy conservation for integrating the equations of motion for SPC/E water at 300 K. To avoid cut-off effects, reaction-field electrostatics with $\epsilon_{RF} = \infty$ and shifted Lennard-Jones interactions are used, both with a buffer region. The long-range interactions were evaluated between 1.0 and 1.4 nm. In Fig. 3.7 one can see that for electrostatics the Trotter scheme does an order of magnitude better up to Δt_{LR} = 16 fs. The electrostatics depends strongly on the orientation of the water molecules, which changes rapidly. For Lennard-Jones interactions, the energy drift is linear in Δt_{LR} and roughly two orders of magnitude smaller than for the electrostatics. Lennard-Jones forces are smaller than Coulomb forces and they are mainly affected by translation of water molecules, not rotation.