3.5 Shell molecular dynamics

GROMACS can simulate polarizability using the shell model of Dick and Overhauser [41]. In such models a shell particle representing the electronic degrees of freedom is attached to a nucleus by a spring. The potential energy is minimized with respect to the shell position at every step of the simulation (see below). Successful applications of shell models in GROMACS have been published for N_2 [42] and water [43].

3.5.1 Optimization of the shell positions

The force $\boldsymbol{F}_S$ on a shell particle S can be decomposed into two components

$$\boldsymbol{F}_S \;=\; \boldsymbol{F}_{bond} + \boldsymbol{F}_{nb} \tag{3.94}$$

where $\boldsymbol{F}_{bond}$ denotes the component representing the polarization energy, usually represented by a harmonic potential and $\boldsymbol{F}_{nb}$ is the sum of Coulomb and van der Waals interactions. If we assume that $\boldsymbol{F}_{nb}$ is almost constant we can analytically derive the optimal position of the shell, i.e. where $\boldsymbol{F}_S = 0$. If we have the shell S connected to atom A we have

$$\boldsymbol{F}_{bond} \;=\; k_b \left(\boldsymbol{x}_S - \boldsymbol{x}_A\right). \tag{3.95}$$

In an iterative solver, we have positions $\boldsymbol{x}_S(n)$ where n is the iteration count. We now have at iteration n

$$\boldsymbol{F}_{nb} \;=\; \boldsymbol{F}_S - k_b\left(\boldsymbol{x}_S(n) - \boldsymbol{x}_A\right) \tag{3.96}$$

and the optimal position for the shells $x_S(n+1)$ thus follows from

$$\boldsymbol{F}_S - k_b\left(\boldsymbol{x}_S(n) - \boldsymbol{x}_A\right) + k_b\left(\boldsymbol{x}_S(n+1) - \boldsymbol{x}_A\right) = 0 \tag{3.97}$$

if we write

$$\Delta\boldsymbol{x}_S = \boldsymbol{x}_S(n+1) - \boldsymbol{x}_S(n) \tag{3.98}$$

we finally obtain

$$\Delta\boldsymbol{x}_S = \boldsymbol{F}_S/k_b \tag{3.99}$$

which then yields the algorithm to compute the next trial in the optimization of shell positions

$$\boldsymbol{x}_S(n+1) \;=\; \boldsymbol{x}_S(n) + \boldsymbol{F}_S/k_b. \tag{3.100}$$

3.6 Constraint algorithms

Constraints can be imposed in GROMACS using LINCS (default) or the traditional SHAKE method.

3.6.1 SHAKE

The SHAKE [44] algorithm changes a set of unconstrained coordinates $\boldsymbol{r}^{'}$ to a set of coordinates $\boldsymbol{r}''$ that fulfill a list of distance constraints, using a set $\boldsymbol{r}$ reference, as

$$\mathrm{SHAKE}(\boldsymbol{r}^{'} \rightarrow \boldsymbol{r}''; \boldsymbol{r}) \tag{3.101}$$

This action is consistent with solving a set of Lagrange multipliers in the constrained equations of motion. SHAKE needs a *relative tolerance*; it will continue until all constraints are satisfied within that relative tolerance. An error message is given if SHAKE cannot reset the coordinates because the deviation is too large, or if a given number of iterations is surpassed.

Assume the equations of motion must fulfill K holonomic constraints, expressed as

$$\sigma_k(\boldsymbol{r}_1 \ldots \boldsymbol{r}_N) = 0; \quad k = 1 \ldots K. \tag{3.102}$$

For example, $(\boldsymbol{r}_1 - \boldsymbol{r}_2)^2 - b^2 = 0$. Then the forces are defined as

$$-\frac{\partial}{\partial \boldsymbol{r}_i}\left(V + \sum_{k=1}^{K}\lambda_k\sigma_k\right), \tag{3.103}$$

where λ_k are Lagrange multipliers which must be solved to fulfill the constraint equations. The second part of this sum determines the *constraint forces* $\boldsymbol{G}_i$, defined by

$$\boldsymbol{G}_i = -\sum_{k=1}^{K}\lambda_k\frac{\partial\sigma_k}{\partial\boldsymbol{r}_i} \tag{3.104}$$

The displacement due to the constraint forces in the leap-frog or Verlet algorithm is equal to $(\boldsymbol{G}_i/m_i)(\Delta t)^2$. Solving the Lagrange multipliers (and hence the displacements) requires the solution of a set of coupled equations of the second degree. These are solved iteratively by SHAKE. For the special case of rigid water molecules, that often make up more than 80% of the simulation system we have implemented the SETTLE algorithm [45] (sec. 5.5).

For velocity Verlet, an additional round of constraining must be done, to constrain the velocities of the second velocity half step, removing any component of the velocity parallel to the bond vector. This step is called RATTLE, and is covered in more detail in the original Andersen paper [46].

3.6.2 LINCS

The LINCS algorithm

LINCS is an algorithm that resets bonds to their correct lengths after an unconstrained update [47]. The method is non-iterative, as it always uses two steps. Although LINCS is based on matrices, no matrix-matrix multiplications are needed. The method is more stable and faster than SHAKE, but it can only be used with bond constraints and isolated angle constraints, such as the proton angle in OH. Because of its stability, LINCS is especially useful for Brownian dynamics. LINCS has two parameters, which are explained in the subsection parameters. The parallel version of LINCS, P-LINCS, is described in subsection 3.17.3.

The LINCS formulas

We consider a system of N particles, with positions given by a $3N$ vector $\boldsymbol{r}(t)$. For molecular dynamics the equations of motion are given by Newton's Law

$$\frac{\mathrm{d}^2\boldsymbol{r}}{\mathrm{d}t^2} = \boldsymbol{M}^{-1}\boldsymbol{F}, \tag{3.105}$$

where $\boldsymbol{F}$ is the $3N$ force vector and $\boldsymbol{M}$ is a $3N \times 3N$ diagonal matrix, containing the masses of the particles. The system is constrained by K time-independent constraint equations

$$g_i(\boldsymbol{r}) = |\boldsymbol{r}_{i_1} - \boldsymbol{r}_{i_2}| - d_i = 0 \qquad i = 1, \ldots, K. \tag{3.106}$$

In a numerical integration scheme, LINCS is applied after an unconstrained update, just like SHAKE. The algorithm works in two steps (see figure Fig. 3.10). In the first step, the projections of the new bonds on the old bonds are set to zero. In the second step, a correction is applied for the lengthening of the bonds due to rotation. The numerics for the first step and the second step are very similar. A complete derivation of the algorithm can be found in [47]. Only a short description of the first step is given here.

A new notation is introduced for the gradient matrix of the constraint equations which appears on the right hand side of this equation:

$$B_{hi} = \frac{\partial g_h}{\partial r_i} \tag{3.107}$$

Notice that $\boldsymbol{B}$ is a $K \times 3N$ matrix, it contains the directions of the constraints. The following equation shows how the new constrained coordinates $\boldsymbol{r}_{n+1}$ are related to the unconstrained coordinates

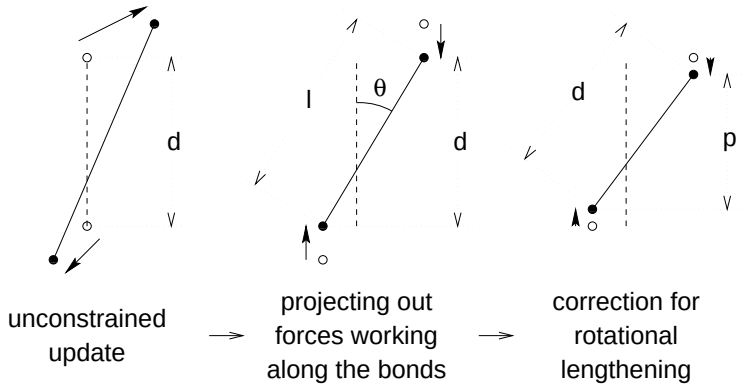

Figure 3.10: The three position updates needed for one time step. The dashed line is the old bond of length d, the solid lines are the new bonds. $l = d\cos\theta$ and $p = (2d^2 - l^2)^{\frac{1}{2}}$.

$\boldsymbol{r}_{n+1}^{unc}$ by

$$\begin{aligned} \boldsymbol{r}_{n+1} = (\boldsymbol{I} - \boldsymbol{T}_n\boldsymbol{B}_n)\boldsymbol{r}_{n+1}^{unc} + \boldsymbol{T}_n\boldsymbol{d} = \\ \boldsymbol{r}_{n+1}^{unc} - \boldsymbol{M}^{-1}\boldsymbol{B}_n(\boldsymbol{B}_n\boldsymbol{M}^{-1}\boldsymbol{B}_n^T)^{-1}(\boldsymbol{B}_n\boldsymbol{r}_{n+1}^{unc} - \boldsymbol{d}) \end{aligned} \tag{3.108}$$

where $\boldsymbol{T} = \boldsymbol{M}^{-1}\boldsymbol{B}^T(\boldsymbol{B}\boldsymbol{M}^{-1}\boldsymbol{B}^T)^{-1}$. The derivation of this equation from eqns. 3.105 and 3.106 can be found in [47].

This first step does not set the real bond lengths to the prescribed lengths, but the projection of the new bonds onto the old directions of the bonds. To correct for the rotation of bond i, the projection of the bond, p_i, on the old direction is set to

$$p_i = \sqrt{2d_i^2 - l_i^2}, \tag{3.109}$$

where l_i is the bond length after the first projection. The corrected positions are

$$\boldsymbol{r}_{n+1}^{*} = (\boldsymbol{I} - \boldsymbol{T}_n\boldsymbol{B}_n)\boldsymbol{r}_{n+1} + \boldsymbol{T}_n\boldsymbol{p}. \tag{3.110}$$

This correction for rotational effects is actually an iterative process, but during MD only one iteration is applied. The relative constraint deviation after this procedure will be less than 0.0001 for every constraint. In energy minimization, this might not be accurate enough, so the number of iterations is equal to the order of the expansion (see below).

Half of the CPU time goes to inverting the constraint coupling matrix $\boldsymbol{B}_n\boldsymbol{M}^{-1}\boldsymbol{B}_n^T$, which has to be done every time step. This $K \times K$ matrix has $1/m_{i_1} + 1/m_{i_2}$ on the diagonal. The off-diagonal elements are only non-zero when two bonds are connected, then the element is $\cos\phi/m_c$, where m_c is the mass of the atom connecting the two bonds and ϕ is the angle between the bonds.

The matrix $\boldsymbol{T}$ is inverted through a power expansion. A $K \times K$ matrix $\boldsymbol{S}$ is introduced which is the inverse square root of the diagonal of $\boldsymbol{B}_n\boldsymbol{M}^{-1}\boldsymbol{B}_n^T$. This matrix is used to convert the diagonal elements of the coupling matrix to one:

$$\begin{aligned} (\boldsymbol{B}_n\boldsymbol{M}^{-1}\boldsymbol{B}_n^T)^{-1} &= \boldsymbol{S}\boldsymbol{S}^{-1}(\boldsymbol{B}_n\boldsymbol{M}^{-1}\boldsymbol{B}_n^T)^{-1}\boldsymbol{S}^{-1}\boldsymbol{S} \\ &= \boldsymbol{S}(\boldsymbol{S}\boldsymbol{B}_n\boldsymbol{M}^{-1}\boldsymbol{B}_n^T\boldsymbol{S})^{-1}\boldsymbol{S} = \boldsymbol{S}(\boldsymbol{I} - \boldsymbol{A}_n)^{-1}\boldsymbol{S} \end{aligned} \tag{3.111}$$

The matrix $\boldsymbol{A}_n$ is symmetric and sparse and has zeros on the diagonal. Thus a simple trick can be used to calculate the inverse:

$$(\boldsymbol{I} - \boldsymbol{A}_n)^{-1} = \boldsymbol{I} + \boldsymbol{A}_n + \boldsymbol{A}_n^2 + \boldsymbol{A}_n^3 + \ldots \tag{3.112}$$

This inversion method is only valid if the absolute values of all the eigenvalues of $\boldsymbol{A}_n$ are smaller than one. In molecules with only bond constraints, the connectivity is so low that this will always be true, even if ring structures are present. Problems can arise in angle-constrained molecules. By constraining angles with additional distance constraints, multiple small ring structures are introduced. This gives a high connectivity, leading to large eigenvalues. Therefore LINCS should NOT be used with coupled angle-constraints.

For molecules with all bonds constrained the eigenvalues of A are around 0.4. This means that with each additional order in the expansion eqn. 3.112 the deviations decrease by a factor 0.4. But for relatively isolated triangles of constraints the largest eigenvalue is around 0.7. Such triangles can occur when removing hydrogen angle vibrations with an additional angle constraint in alcohol groups or when constraining water molecules with LINCS, for instance with flexible constraints. The constraints in such triangles converge twice as slow as the other constraints. Therefore, starting with GROMACS 4, additional terms are added to the expansion for such triangles

$$(\boldsymbol{I} - \boldsymbol{A}_n)^{-1} \approx \boldsymbol{I} + \boldsymbol{A}_n + \ldots + \boldsymbol{A}_n^{N_i} + \left(\boldsymbol{A}_n^* + \ldots + \boldsymbol{A}_n^{*\,N_i}\right) \boldsymbol{A}_n^{N_i} \tag{3.113}$$

where N_i is the normal order of the expansion and $\boldsymbol{A}^*$ only contains the elements of $\boldsymbol{A}$ that couple constraints within rigid triangles, all other elements are zero. In this manner, the accuracy of angle constraints comes close to that of the other constraints, while the series of matrix vector multiplications required for determining the expansion only needs to be extended for a few constraint couplings. This procedure is described in the P-LINCS paper[48].

The LINCS Parameters

The accuracy of LINCS depends on the number of matrices used in the expansion eqn. 3.112. For MD calculations a fourth order expansion is enough. For Brownian dynamics with large time steps an eighth order expansion may be necessary. The order is a parameter in the `*.mdp` file. The implementation of LINCS is done in such a way that the algorithm will never crash. Even when it is impossible to to reset the constraints LINCS will generate a conformation which fulfills the constraints as well as possible. However, LINCS will generate a warning when in one step a bond rotates over more than a predefined angle. This angle is set by the user in the `*.mdp` file.

3.7 Simulated Annealing

The well known simulated annealing (SA) protocol is supported in GROMACS, and you can even couple multiple groups of atoms separately with an arbitrary number of reference temperatures that change during the simulation. The annealing is implemented by simply changing the current reference temperature for each group in the temperature coupling, so the actual relaxation and coupling properties depends on the type of thermostat you use and how hard you are coupling it. Since we are changing the reference temperature it is important to remember that the system will

NOT instantaneously reach this value - you need to allow for the inherent relaxation time in the coupling algorithm too. If you are changing the annealing reference temperature faster than the temperature relaxation you will probably end up with a crash when the difference becomes too large.

The annealing protocol is specified as a series of corresponding times and reference temperatures for each group, and you can also choose whether you only want a single sequence (after which the temperature will be coupled to the last reference value), or if the annealing should be periodic and restart at the first reference point once the sequence is completed. You can mix and match both types of annealing and non-annealed groups in your simulation.

3.8 Stochastic Dynamics

Stochastic or velocity Langevin dynamics adds a friction and a noise term to Newton's equations of motion, as

$$m_i \frac{\mathrm{d}^2 \boldsymbol{r}_i}{\mathrm{d}t^2} = -m_i \gamma_i \frac{\mathrm{d}\boldsymbol{r}_i}{\mathrm{d}t} + \boldsymbol{F}_i(\boldsymbol{r}) + \mathring{\boldsymbol{r}}_i, \tag{3.114}$$

where γ_i is the friction constant $[1/\mathrm{ps}]$ and $\mathring{\boldsymbol{r}}_i(t)$ is a noise process with $\langle \mathring{r}_i(t)\ \mathring{r}_j(t+s) \rangle = 2 m_i \gamma_i k_B T \delta(s) \delta_{ij}$. When $1/\gamma_i$ is large compared to the time scales present in the system, one could see stochastic dynamics as molecular dynamics with stochastic temperature-coupling. The advantage compared to MD with Berendsen temperature-coupling is that in case of SD the generated ensemble is known. For simulating a system in vacuum there is the additional advantage that there is no accumulation of errors for the overall translational and rotational degrees of freedom. When $1/\gamma_i$ is small compared to the time scales present in the system, the dynamics will be completely different from MD, but the sampling is still correct.

In GROMACS there are two algorithms to integrate equation (3.114): a simple and efficient one and a more complex leap-frog algorithm [49], which is now deprecated. The accuracy of both integrators is equivalent to the normal MD leap-frog and Velocity Verlet integrator, except with constraints where the complex SD integrator samples at a temperature that is slightly too high (although that error is smaller than the one from the Velocity Verlet integrator that uses the kinetic energy from the full-step velocity). The simple integrator is nearly identical to the common way of discretizing the Langevin equation, but the friction and velocity term are applied in an impulse fashion [50]. The simple integrator is:

$$\boldsymbol{v}' = \boldsymbol{v}(t - \frac{1}{2}\Delta t) + \frac{1}{m}\boldsymbol{F}(t)\Delta t \tag{3.115}$$

$$\Delta \boldsymbol{v} = -\alpha\, \boldsymbol{v}'(t + \frac{1}{2}\Delta t) + \sqrt{\frac{k_B T}{m}(1-\alpha^2)}\, \boldsymbol{r}_i^G \tag{3.116}$$

$$\boldsymbol{r}(t + \Delta t) = \boldsymbol{r}(t) + \left(\boldsymbol{v}' + \frac{1}{2}\Delta \boldsymbol{v}\right)\Delta t \tag{3.117}$$

$$\boldsymbol{v}(t + \frac{1}{2}\Delta t) = \boldsymbol{v}' + \Delta \boldsymbol{v} \tag{3.118}$$

$$\alpha = 1 - e^{-\gamma \Delta t} \tag{3.119}$$

where $\boldsymbol{r}_i^G$ is Gaussian distributed noise with $\mu = 0, \sigma = 1$. The velocity is first updated a full time step without friction and noise to get $\boldsymbol{v}'$, identical to the normal update in leap-frog. The friction

and noise are then applied as an impulse at step $t + \Delta t$. The advantage of this scheme is that the velocity-dependent terms act at the full time step, which makes the correct integration of forces that depend on both coordinates and velocities, such as constraints and dissipative particle dynamics (DPD, not implented yet), straightforward. With constraints, the coordinate update eqn. 3.117 is split into a normal leap-frog update and a $\Delta \boldsymbol{v}$. After both of these updates the constraints are applied to coordinates and velocities.

In the deprecated complex algorithm, four Gaussian random numbers are required per integration step per degree of freedom, and with constraints the coordinates need to be constrained twice per integration step. Depending on the computational cost of the force calculation, this can take a significant part of the simulation time. Exact continuation of a stochastic dynamics simulation is not possible, because the state of the random number generator is not stored.

When using SD as a thermostat, an appropriate value for γ is e.g. 0.5 ps^{-1}, since this results in a friction that is lower than the internal friction of water, while it still provides efficient thermostatting.

3.9 Brownian Dynamics

In the limit of high friction, stochastic dynamics reduces to Brownian dynamics, also called position Langevin dynamics. This applies to over-damped systems, *i.e.* systems in which the inertia effects are negligible. The equation is

$$\frac{\mathrm{d}\boldsymbol{r}_i}{\mathrm{d}t} = \frac{1}{\gamma_i}\boldsymbol{F}_i(\boldsymbol{r}) + \mathring{\boldsymbol{r}}_i \tag{3.120}$$

where γ_i is the friction coefficient [amu/ps] and $\mathring{\boldsymbol{r}}_i(t)$ is a noise process with $\langle \mathring{r}_i(t)\ \mathring{r}_j(t+s)\rangle = 2\delta(s)\delta_{ij}k_BT/\gamma_i$. In GROMACS the equations are integrated with a simple, explicit scheme

$$\boldsymbol{r}_i(t+\Delta t) = \boldsymbol{r}_i(t) + \frac{\Delta t}{\gamma_i}\boldsymbol{F}_i(\boldsymbol{r}(t)) + \sqrt{2k_BT\frac{\Delta t}{\gamma_i}}\,\boldsymbol{r}_i^G, \tag{3.121}$$

where $\boldsymbol{r}_i^G$ is Gaussian distributed noise with $\mu = 0$, $\sigma = 1$. The friction coefficients γ_i can be chosen the same for all particles or as $\gamma_i = m_i\,\gamma_i$, where the friction constants γ_i can be different for different groups of atoms. Because the system is assumed to be over-damped, large timesteps can be used. LINCS should be used for the constraints since SHAKE will not converge for large atomic displacements. BD is an option of the `mdrun` program.

3.10 Energy Minimization

Energy minimization in GROMACS can be done using steepest descent, conjugate gradients, or l-bfgs (limited-memory Broyden-Fletcher-Goldfarb-Shanno quasi-Newtonian minimizer...we prefer the abbreviation). EM is just an option of the `mdrun` program.

3.10.1 Steepest Descent

Although steepest descent is certainly not the most efficient algorithm for searching, it is robust and easy to implement.

We define the vector $\boldsymbol{r}$ as the vector of all $3N$ coordinates. Initially a maximum displacement h_0 (*e.g.* 0.01 nm) must be given.

First the forces $\boldsymbol{F}$ and potential energy are calculated. New positions are calculated by

$$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n + \frac{\boldsymbol{F}_n}{\max(|\boldsymbol{F}_n|)} h_n, \tag{3.122}$$

where h_n is the maximum displacement and $\boldsymbol{F}_n$ is the force, or the negative gradient of the potential V. The notation $\max(|\boldsymbol{F}_n|)$ means the largest of the absolute values of the force components. The forces and energy are again computed for the new positions
If ($V_{n+1} < V_n$) the new positions are accepted and $h_{n+1} = 1.2h_n$.
If ($V_{n+1} \geq V_n$) the new positions are rejected and $h_n = 0.2h_n$.

The algorithm stops when either a user-specified number of force evaluations has been performed (*e.g.* 100), or when the maximum of the absolute values of the force (gradient) components is smaller than a specified value ϵ. Since force truncation produces some noise in the energy evaluation, the stopping criterion should not be made too tight to avoid endless iterations. A reasonable value for ϵ can be estimated from the root mean square force f a harmonic oscillator would exhibit at a temperature T. This value is

$$f = 2\pi\nu\sqrt{2mkT}, \tag{3.123}$$

where ν is the oscillator frequency, m the (reduced) mass, and k Boltzmann's constant. For a weak oscillator with a wave number of 100 cm^{-1} and a mass of 10 atomic units, at a temperature of 1 K, $f = 7.7$ kJ mol^{-1} nm^{-1}. A value for ϵ between 1 and 10 is acceptable.

3.10.2 Conjugate Gradient

Conjugate gradient is slower than steepest descent in the early stages of the minimization, but becomes more efficient closer to the energy minimum. The parameters and stop criterion are the same as for steepest descent. In GROMACS conjugate gradient can not be used with constraints, including the SETTLE algorithm for water [45], as this has not been implemented. If water is present it must be of a flexible model, which can be specified in the `*.mdp` file by `define = -DFLEXIBLE`.

This is not really a restriction, since the accuracy of conjugate gradient is only required for minimization prior to a normal-mode analysis, which cannot be performed with constraints. For most other purposes steepest descent is efficient enough.

3.10.3 L-BFGS

The original BFGS algorithm works by successively creating better approximations of the inverse Hessian matrix, and moving the system to the currently estimated minimum. The memory requirements for this are proportional to the square of the number of particles, so it is not practical

for large systems like biomolecules. Instead, we use the L-BFGS algorithm of Nocedal [51, 52], which approximates the inverse Hessian by a fixed number of corrections from previous steps. This sliding-window technique is almost as efficient as the original method, but the memory requirements are much lower - proportional to the number of particles multiplied with the correction steps. In practice we have found it to converge faster than conjugate gradients, but due to the correction steps it is not yet parallelized. It is also noteworthy that switched or shifted interactions usually improve the convergence, since sharp cut-offs mean the potential function at the current coordinates is slightly different from the previous steps used to build the inverse Hessian approximation.

3.11 Normal-Mode Analysis

Normal-mode analysis [53, 54, 55] can be performed using GROMACS, by diagonalization of the mass-weighted Hessian H:

$$R^T M^{-1/2} H M^{-1/2} R = \text{diag}(\lambda_1, \ldots, \lambda_{3N}) \tag{3.124}$$

$$\lambda_i = (2\pi\omega_i)^2 \tag{3.125}$$

where M contains the atomic masses, R is a matrix that contains the eigenvectors as columns, λ_i are the eigenvalues and ω_i are the corresponding frequencies.

First the Hessian matrix, which is a $3N \times 3N$ matrix where N is the number of atoms, needs to be calculated:

$$H_{ij} = \frac{\partial^2 V}{\partial x_i \partial x_j} \tag{3.126}$$

where x_i and x_j denote the atomic x, y or z coordinates. In practice, this equation is not used, but the Hessian is calculated numerically from the force as:

$$H_{ij} = -\frac{f_i(\mathbf{x} + h\mathbf{e}_j) - f_i(\mathbf{x} - h\mathbf{e}_j)}{2h} \tag{3.127}$$

$$f_i = -\frac{\partial V}{\partial x_i} \tag{3.128}$$

where $\mathbf{e}_j$ is the unit vector in direction j. It should be noted that for a usual normal-mode calculation, it is necessary to completely minimize the energy prior to computation of the Hessian. The tolerance required depends on the type of system, but a rough indication is 0.001 kJ mol^{-1}. Minimization should be done with conjugate gradients or L-BFGS in double precision.

A number of GROMACS programs are involved in these calculations. First, the energy should be minimized using `mdrun`. Then, `mdrun` computes the Hessian. **Note** that for generating the run input file, one should use the minimized conformation from the full precision trajectory file, as the structure file is not accurate enough. `g_nmeig` does the diagonalization and the sorting of the normal modes according to their frequencies. Both `mdrun` and `g_nmeig` should be run in double precision. The normal modes can be analyzed with the program `g_anaeig`. Ensembles of structures at any temperature and for any subset of normal modes can be generated with `g_nmens`. An overview of normal-mode analysis and the related principal component analysis (see sec. 8.10) can be found in [56].

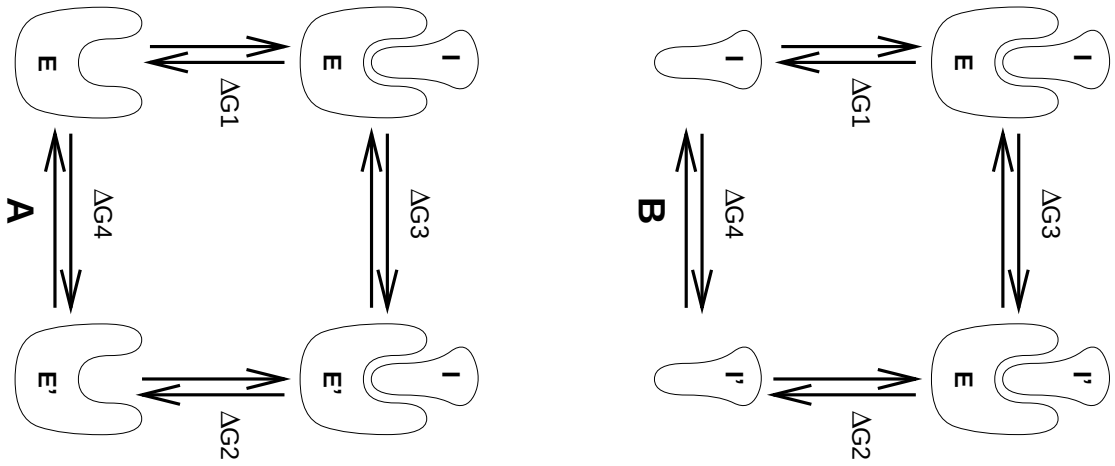

Figure 3.11: Free energy cycles. **A:** to calculate ΔG_{12}, the free energy difference between the binding of inhibitor **I** to enzymes **E** respectively **E**′. **B:** to calculate ΔG_{12}, the free energy difference for binding of inhibitors **I** respectively **I**′ to enzyme **E**.

3.12 Free energy calculations

3.12.1 Slow-growth methods

Free energy calculations can be performed in GROMACS using a number of methods, including "slow-growth." An example problem might be calculating the difference in free energy of binding of an inhibitor **I** to an enzyme **E** and to a mutated enzyme **E**′. It is not feasible with computer simulations to perform a docking calculation for such a large complex, or even releasing the inhibitor from the enzyme in a reasonable amount of computer time with reasonable accuracy. However, if we consider the free energy cycle in Fig. 3.11A we can write:

$$\Delta G_1 - \Delta G_2 = \Delta G_3 - \Delta G_4 \tag{3.129}$$

If we are interested in the left-hand term we can equally well compute the right-hand term.

If we want to compute the difference in free energy of binding of two inhibitors **I** and **I**′ to an enzyme **E** (Fig. 3.11B) we can again use eqn. 3.129 to compute the desired property.

Free energy differences between two molecular species can be calculated in GROMACS using the "slow-growth" method. Such free energy differences between different molecular species are physically meaningless, but they can be used to obtain meaningful quantities employing a thermodynamic cycle. The method requires a simulation during which the Hamiltonian of the system changes slowly from that describing one system (A) to that describing the other system (B). The change must be so slow that the system remains in equilibrium during the process; if that requirement is fulfilled, the change is reversible and a slow-growth simulation from B to A will yield the same results (but with a different sign) as a slow-growth simulation from A to B. This is a useful check, but the user should be aware of the danger that equality of forward and backward growth results does not guarantee correctness of the results.

The required modification of the Hamiltonian H is realized by making H a function of a *coupling parameter* $\lambda : H = H(p, q; \lambda)$ in such a way that $\lambda = 0$ describes system A and $\lambda = 1$ describes

system B:

$$H(p,q;0) = H^{\mathrm{A}}(p,q); \quad H(p,q;1) = H^{\mathrm{B}}(p,q). \tag{3.130}$$

In GROMACS, the functional form of the λ-dependence is different for the various force-field contributions and is described in section sec. 4.5.

The Helmholtz free energy A is related to the partition function Q of an N,V,T ensemble, which is assumed to be the equilibrium ensemble generated by a MD simulation at constant volume and temperature. The generally more useful Gibbs free energy G is related to the partition function Δ of an N,p,T ensemble, which is assumed to be the equilibrium ensemble generated by a MD simulation at constant pressure and temperature:

$$\begin{aligned} A(\lambda) &= -k_B T \ln Q & (3.131)\\ Q &= c \iint \exp[-\beta H(p,q;\lambda)]\, dp\, dq & (3.132)\\ G(\lambda) &= -k_B T \ln \Delta & (3.133)\\ \Delta &= c \iiint \exp[-\beta H(p,q;\lambda) - \beta pV]\, dp\, dq\, dV & (3.134)\\ G &= A + pV, & (3.135) \end{aligned}$$

where $\beta = 1/(k_B T)$ and $c = (N! h^{3N})^{-1}$. These integrals over phase space cannot be evaluated from a simulation, but it is possible to evaluate the derivative with respect to λ as an ensemble average:

$$\frac{dA}{d\lambda} = \frac{\iint (\partial H/\partial\lambda) \exp[-\beta H(p,q;\lambda)]\, dp\, dq}{\iint \exp[-\beta H(p,q;\lambda)]\, dp\, dq} = \left\langle \frac{\partial H}{\partial \lambda} \right\rangle_{NVT;\lambda}, \tag{3.136}$$

with a similar relation for $dG/d\lambda$ in the N,p,T ensemble. The difference in free energy between A and B can be found by integrating the derivative over λ:

$$\begin{aligned} A^{\mathrm{B}}(V,T) - A^{\mathrm{A}}(V,T) &= \int_0^1 \left\langle \frac{\partial H}{\partial \lambda} \right\rangle_{NVT;\lambda} d\lambda & (3.137)\\ G^{\mathrm{B}}(p,T) - G^{\mathrm{A}}(p,T) &= \int_0^1 \left\langle \frac{\partial H}{\partial \lambda} \right\rangle_{NpT;\lambda} d\lambda. & (3.138) \end{aligned}$$

If one wishes to evaluate $G^{\mathrm{B}}(p,T) - G^{\mathrm{A}}(p,T)$, the natural choice is a constant-pressure simulation. However, this quantity can also be obtained from a slow-growth simulation at constant volume, starting with system A at pressure p and volume V and ending with system B at pressure p_B, by applying the following small (but, in principle, exact) correction:

$$G^{\mathrm{B}}(p) - G^{\mathrm{A}}(p) = A^{\mathrm{B}}(V) - A^{\mathrm{A}}(V) - \int_p^{p^{\mathrm{B}}} [V^{\mathrm{B}}(p') - V]\, dp' \tag{3.139}$$

Here we omitted the constant T from the notation. This correction is roughly equal to $-\frac{1}{2}(p^{\mathrm{B}} - p)\Delta V = (\Delta V)^2/(2\kappa V)$, where ΔV is the volume change at p and κ is the isothermal compressibility. This is usually small; for example, the growth of a water molecule from nothing in a bath of 1000 water molecules at constant volume would produce an additional pressure of as much as 22 bar, but a correction to the Helmholtz free energy of just -1 kJ mol^{-1}.

In Cartesian coordinates, the kinetic energy term in the Hamiltonian depends only on the momenta, and can be separately integrated and, in fact, removed from the equations. When masses do not change, there is no contribution from the kinetic energy at all; otherwise the integrated contribution to the free energy is $-\frac{3}{2} k_B T \ln(m^{\mathrm{B}}/m^{\mathrm{A}})$. **Note** that this is only true in the absence of constraints.

3.12.2 Thermodynamic integration

GROMACS offers the possibility to integrate eq. 3.137 or eq. 3.138 in one simulation over the full range from A to B. However, if the change is large and insufficient sampling can be expected, the user may prefer to determine the value of $\langle dG/d\lambda \rangle$ accurately at a number of well-chosen intermediate values of λ. This can easily be done by setting the stepsize `delta_lambda` to zero. Each simulation can be equilibrated first, and a proper error estimate can be made for each value of $dG/d\lambda$ from the fluctuation of $\partial H/\partial\lambda$. The total free energy change is then determined afterward by an appropriate numerical integration procedure.

GROMACS now also supports the use of Bennett's Acceptance Ratio [57] for calculating values of ΔG for transformations from state A to state B using the program `g_bar`. The same data can also be used to calculate free energies using MBAR [58], though the analysis currently requires external tools from the external `pymbar` package, at https://SimTK.org/home/pymbar.

The λ-dependence for the force-field contributions is described in detail in section sec. 4.5.