Water

A very simple molecule that consists of 3 atoms: one oxygen and two hydrogen atoms. Some remarkable properties. **Very high melting and vaporization points** for liquid with so light molecular mass. For example, CCl4 melt at 250K, while water at 273K. Anomalous density behavior at freezing (**expansion in density** – ice lighter than liquid water, ice 0.92 g/mL, liquid water 1.0g/mL). **Very strong electrical forces** (special orientation forces?). High dielectric constant

Explain with "simple" microscopic model macroscopic behavior? Explain also microscopic data, since macroscopic observation are too few.

- ➔ Structures
- ➔ Correlation functions spatial & time.
- ➔ Spatial correlation functions (first and second peak) as determinant of interaction strengths
- ➔ Time correlations measures (for example, dielectric response).

A water molecule is neutral but it carries **a large dipole moment**. The oxygen is highly negative and the hydrogen positive. The dipole moment of the molecule (it is not linear) is 1.855 Debye unit (Debye = $3.33564 \cdot 10^{-30}$ C m). Electron charge and distance of one angstrom corresponds to 4.803 Debye.

The **geometry of a single water molecule** is determined by the following parameters (using a model potential TIP3P – position of an atom as light as hydrogen in quantum mechanics is subject to considerable uncertainty): r(OH) 0.9572 a(HOH) 109.47 Since the hydrogen atoms are symmetric, the dipole moment is an experimental indication that the molecule is not linear. The individual molecules are set to be rigid (no violations of the above internal coordinates are allowed). We shall use harmonic restraints to fix the geometry (at least until we will learn how to handle holonomic constraints). For internal water potential, we write

$$U_{\text{internal}} = k\left(r_{o-h1} - 0.9572\right)^2 + k\left(r_{o-h2} - 0.9572\right)^2 + k'\left(r_{h1-h2} - 1.5139\right)^2$$

where k and k' are constants chosen to minimize the changes in the distances compared to the ideal values, and are chosen empirically to be 600 kcal/molÅ^2.

The potential between n water molecules using the TIP3P model is

$$U_{\text{between molecules}} = \sum_{i>j}\left(\frac{A}{r_{ij}^{12}(oo)} - \frac{B}{r_{ij}^{6}(oo)}\right) + K\sum_{i>j}\sum_{k,l}\frac{c_{ik}c_{jl}}{r_{ik,jl}}$$

the indices i, j are for water molecules, the indices k, l are for the atoms of a single molecule. Note the newly inserted constant K that is used for unit consistency. In our case it is 332.0716. The first sum is over the oxygen atoms only and includes repulsion

between the different terms. This type of potential is also called Lennard Jones. The hydrogen atoms (deprived from electrons) are so small that their influence on the hard-core shape of the molecule is ignored. The hard core is set to be spherical with the oxygen atom at the center. Hydrogen atoms (and oxygen atoms too) are coming to play in the second term of electrostatic interactions.
The potential parameters are as follows

$$A = 629,400 \text{ kcal Å}^{12}/\text{mol}$$
$$B = 625.5 \quad \text{kcal Å}^{6}/\text{mol}$$
$$c_0 = -0.834$$
$$c_H = 0.417$$

To begin with we will be interested in the water dimer. The interaction between the two molecules can be written more explicitly (below) as

$$U_{\text{between molecules}} = \frac{A}{r_{o1o2}^{12}} - \frac{B}{r_{o1o2}^{6}} + \frac{c_{o1}c_{02}}{r_{o1o2}} + \frac{c_{o1}c_{h21}}{r_{o1h21}} + \frac{c_{o1}c_{h22}}{r_{o1h22}} + \frac{c_{o2}c_{h11}}{r_{o1h21}} + \frac{c_{o2}c_{h12}}{r_{o2h12}}$$
$$+ \frac{c_{h11}c_{h21}}{r_{h11h21}} + \frac{c_{h11}c_{h22}}{r_{h11h22}} + \frac{c_{h12}c_{h21}}{r_{h12h21}} + \frac{c_{h12}c_{h22}}{r_{h12h22}}$$

For the first monomer we use *o1 h11* & *h12* to denote the corresponding three atoms. For the second monomer we used instead *o2 h21* & *h22*. Of course the modeling of the interaction between molecules must come together with the internal potential keeping the geometry fixed. Note that we do not compute electrostatic and Lennard Jones interactions of atom within the same molecule.

The water dimer has an optimal geometry (a minimum energy configuration) that we now wish to determine. This is a question of how to arrange negatively charged atoms close to positively charge atoms and keep similar charges away from each other. **Together with the fix geometry of the individual molecules this is a frustrated system** (you cannot make everybody happy – a classic example is of three spins) that also has implications on the docking problem and drug design (determining the orientation of two interacting molecules) and the range of complementing interactions and shape matching

What is the dimensionality of the problem? We have three atoms in each of the water molecules making a total of eighteen degrees of freedom. The actual number relevant for the docking is much smaller. Of the eighteen six are of translation and rotation of the whole system (not changing the relative orientation) and another six are reduced to the internal constraints on the structure on the structure of the molecule. Hence, only six degrees of freedom remain when we attempt to match the two (rigid) water molecules together.

This is possible but requires considerable work for implementation and makes it necessary to write the energy in considerably less convenient coordinates (the most convenient are just the Cartesian coordinates of the individual atoms). So rather than being clever at the very first step, we will use the simplest approach. **In the simplest approach we attempt to optimize the energy of the water dimer as a function of all degrees of freedom**. This requires more work from the computer, but less work from us. Minimizing the energy (and find the best structure) in this large 18 dimensional space is something we cannot do by intuition and we must design an algorithm to do it.

The **Algorithm of steepest descent** is one way of doing it, and this is the first approach that we are going to study. Consider a guess coordinate vector for the water dimer that we denote by R (capitals denotes arrays or vectors, lower case denotes elements of a vector). It is a vector of length 18 and includes all the x,y,z coordinates. What is the best way of organizing the coordinates? In some programs the vector of coordinates is decomposed into three Cartesian vectors $R \equiv (X,Y,Z) = (x_1, x_2, ... x_{18}, y_1, ..., y_{18}, z_1, ..., z_{18})$. This set up is however less efficient from the view point of memory architecture. When we compute distances between atoms (the main computation required for the energy calculations) we require the (x,y,z) coordinates of the two atoms. By aggregating all the Cartesian components together (e.g. all the x-s are coming first) we may create cache misses. To find the Cartesian components of one atom makes it necessary to "walk" quite far along the array, every time we need one of the Cartesian components we need to step through n numbers to pick the next Cartesian component. This is not a serious problem for 18 atoms but for hundred of thousands or millions of atoms (like some simulations are) it might be.
A more efficient structure that we shall adopt is therefore:
$R = (x_1, y_1, z_1, ..., x_{18}, y_{18}, z_{18})$
where the coordinates that belong to a single particle are kept together, making it less likely to have cache misses.

Starting from an initial configuration R_0 we want to find a lower energy configuration. The argument is that low energy configurations are more likely to be relevant. This is true in general but is not always sufficient. We shall come back when discussing entropy temperature and how to simulate thermal fluctuations.

For the moment assume that we are going to make a small step a displacement $\delta R = (R - R_0)$ of norm $\Delta \equiv |\delta R| = \sqrt{\sum_i (q_i - q_{i0})^2}$. We have used the so-called norm 2 distance measure which is very common in computational biophysics. We have also used another variable q_i, which is any of the Cartesian coordinates of the system. For a system with N atoms, it has *3N* q_i. It is useful to know that a more general formulation would be $\Delta^{(n)} = \left[\sum_i \left(q_i - q_{i0} \right)^n \right]^{\frac{1}{n}}$. These different distances emphasize alternative aspects of the system. For example when $n \to \infty$ the norm approaches the largest displacement along a given Cartesian direction in the system. Another widely used distance is for $n = 1$ which is "Manhattan" distance.

The displacement δR is made in a large space of 18 degrees of freedom, so while we fixed the size of the displacement to be small there is still a lot to be done to find an optimal displacement that will reduce the energy as much as possible (for a given size of a displacement). How are we going to choose the displacement (given that we are choosing the norm of the displacement to be small). Here is a place where the Taylor's series can come to the rescue. We expand the potential $U(R)$ in the neighborhood of the current coordinate set R_0 to the first order in Δ (other higher order terms are considered small and are neglected).

$$U\left(R_0 + \delta R\right) \simeq U\left(R_0\right) + \sum_i \left. \frac{dU}{dq_i} \right|_{q_i = q_{i0}} \left(q_i - q_{i0}\right) = U\left(R_0\right) + \left(\nabla U\right)^t \cdot \delta R$$

The expression $\left(\nabla U\right)^t$ means a transpose vector of rank $3N$, which is also called the gradient of the potential

$$\left(\nabla U\right)^t = \left(\frac{dU}{dq_1}, \frac{dU}{dq_2}, \ldots, \frac{dU}{dq_N} \right)$$

Similarly we have for δR

$$\delta R = \begin{pmatrix} \left(q_1 - q_{10}\right) \\ \left(q_2 - q_{20}\right) \\ \ldots \\ \left(q_N - q_{N0}\right) \end{pmatrix}$$

The expression for the potential difference is a scalar (or inner) product between two vectors. Hence we can also write

$$U\left(R_0 + \delta R\right) - U\left(R_0\right) \equiv \delta U = \left|\nabla U\right|_2 \cdot \left|\delta R\right|_2 \cdot \cos(\theta) = \left|\nabla U\right| \cdot \Delta \cdot \cos(\theta)$$

We have omitted the "2" from the expression of the vector norm $|A|$ on the right hand side. We will always use the norm 2, unless specifically suggested otherwise, and therefore carrying the "2" around is not necessary. Note that the gradient of the potential is something that we compute at the current point, R_0, and it is not something that we can change. Similarly we fixed the norm of the displacement Δ. So **the only variable in town is the direction of the displacement with respect to the gradient of the potential (θ)**.

To minimize the difference in energies (making $U\left(R_0 + \delta R\right)$ as low as possible) we should make $\cos(\theta)$ as small as possible. The best we could do is to make $\cos(\theta) = -1$

and choose the step in the opposite direction to the gradient vector. Hence, the steepest descent algorithm for energy minimization is

$$\delta R = -\frac{\nabla U}{|\nabla U|} \cdot \Delta$$

We did not discuss so far how to choose the step length Δ, except to argue that it should be small. We expect that if the gradient is very small then we are near a minimum (actually a stationary point where $\nabla U = 0$). In that case only a small step should be used. On the other hand if the gradient is large, a somewhat larger step could be taken since we anticipate a significant change in the function (to the better). Of course the step still cannot be too large since our arguments are based on a linear expansion of the function. Based on the above arguments, it is suggestive to use the following (simpler) expression for the displacement that is directly proportional to the length of the gradient vector.

$$\delta R = -\nabla U \cdot \Delta$$

We can imagine now a numerical minimization process that goes as follows:

0. Init -- $R = R_0$
1. Compute $\nabla U(R)$
2. Compute a step $\delta R = -\nabla U \cdot \Delta$ and a new coordinate set $R = R + \delta R$
3. Check for convergence $\left(|\nabla U \le \varepsilon|\right)$

If converged, stop. Otherwise return to 1.

This procedure finds for us a local minimum, a minimum to which we can slide down directly from an initial guess. It will not provide a solution to the global optimization problem, finding the minimum that is the lowest of them all.
We can take the above expression one step further and write it down as a differential equation with the coordinates a function of a progress variable τ. This is not the most efficient way of minimizing the structure under consideration but it will help us understand the process better. We write

$$\frac{dR}{d\tau} = -\nabla U$$

where the (dummy) variable τ is varying from zero to ∞. At $\tau \to \infty$ we approach the nearby stationary point. Not that the potential $U\left(R(\tau)\right)$ is a monotonically non-increasing function of τ. This is easy to show as follows. Multiple both side of the equation by $\left(\frac{dR}{d\tau}\right)^t$, we have

$$\left(\frac{dR}{d\tau}\right)^t \cdot \left(\frac{dR}{d\tau}\right) = -\left(\frac{dR}{d\tau}\right)^t \cdot \nabla U$$

$$-\left(\frac{dR}{d\tau}\right)^t \cdot \nabla U = -\sum_i \frac{dq_i}{d\tau} \cdot \frac{dU}{dq_i}$$

Assuming that the potential does not have explicit dependence of τ (i.e. that τ is more than a dummy variable which contradicts our initial set-up) we can write the final expression in a more compact and illuminating form

$$0 \le \left(\frac{dR}{d\tau}\right)^t \left(\frac{dR}{d\tau}\right) = -\frac{dU}{d\tau}$$
$$\frac{dU}{d\tau} \le 0$$

Hence as we progress the solution of the differential equation the potential energy is decreasing in the best case and is not increasing in the worst case. There can be different variants of how to choose the norm of the step - Δ, we already mentioned two of them.

Another important variant of the steepest descent minimization is to make the step size a parameter and to optimize it for a given search direction defined by ∇U. One approach would be to perform **a search for a minimum along the line defined by the $-\Delta \cdot \nabla U$,** i.e. we seek a Δ such that $\nabla U(R)^t \cdot \nabla U(R + \Delta \cdot \nabla U) = 0$ as a function of the single scalar variable Δ. This one-dimensional minimization makes sense if the calculation of the gradient can be avoided in the one-dimensional minimization, and search steps in the one-dimensional minimization can be computed more efficiently than the determination of the direction. For example, in the one-dimensional optimization only calculation of the energy $U(R + \Delta \cdot \nabla U)$ can be used in conjunction with the approach of interval halving. We guess a given Δ_0 and if the energy is going up we try a half of the previous size $\Delta_0/2$, if it is going down we double it. We continue to evaluate the progress of halving an interval that contains a minimum until we hit a minimum with desired properties (halving on the left or halving on the right results in an increasing energy). **This scheme assumes that the calculation of the potential energy is a more efficient than the calculation of the gradient**. This is however not the case here and for the task at hand the line search option of the steepest descent algorithm is not efficient.

We note that compared to other optimization algorithms (such as conjugate gradient that we shall not discuss, and the Newton-Raphson algorithm that we will) the steepest descent algorithm is considerably slower. However, it is a lot more stable than (for example) the Newton Raphson approach. It is a common practice in molecular simulations to start with a “crude” minimizer like the Steepest Descent algorithm described above to begin with (if the initial structure is pretty bad), and then to refine the coordinates to perfection using something like the Newton Raphson algorithm, which is on our agenda.

A few words about computing potential derivatives (which is of prime importance for minimization algorithms in high dimensions. It is unheard of having effective minimizers in high dimensions without derivatives, since we must have a sense of direction where to go. That sense of direction is given by the potential gradient.

Overall, the potential derivatives are dominated by calculations of distances. It is therefore useful to consider the derivative of a distance between two particles, as a function of the Cartesian coordinates.

$$r_{ij} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_i - z_j\right)^2}$$
$$\frac{dr_{ij}}{dw_i} = \frac{\left(w_i - w_j\right)}{\sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_i - z_j\right)^2}} \qquad w = x, y, z$$

With the above formulas at hand it is straightforward to do all kind of derivatives with extensive use of the chain rule. For example the Lennard Jones term

$$\frac{d}{dw_k}\sum_{i>j}\left(\frac{A}{r_{ij}^{12}} - \frac{B}{r_{ij}^{6}}\right) = \sum_{j\neq k}\left(-\frac{12A}{r_{kj}^{13}} + \frac{6B}{r_{kj}^{7}}\right)\frac{dr_{kj}}{dw_k} = \sum_{j\neq k}\left(-\frac{12A}{r_{kj}^{13}} + \frac{6B}{r_{kj}^{7}}\right)\frac{\left(w_k - w_j\right)}{r_{kj}}$$
$$= \sum_{j\neq k}\left(-\frac{12A}{r_{kj}^{14}} + \frac{6B}{r_{kj}^{8}}\right)\left(w_k - w_j\right)$$

Note that the expression above depends only on even power of the distance (14 and 8) which is good news, meaning that no square roots are required. Square roots are the curse of simulations and are much more expensive to compute compare to add/multiply etc. Unfortunately, electrostatic interactions: energy and derivatives yield odd powers, and are more expensive to compute. The potential and its derivatives require square root calculations. And here is the expression for a set of independent atoms (for convenience we forget about molecules here)

$$\frac{d}{dw_k}\sum_{i>j}\frac{c_i c_j}{r_{i,j}} = \sum_{i\neq j} -\frac{c_i c_j}{r_{i,k}^2}\frac{\left(w_k - w_i\right)}{r_{i,k}} = -\sum_{i\neq j}\frac{c_i c_j}{r_{i,k}^3}\left(w_k - w_i\right)$$

- Here is a little tricky question. To compute the energy, (which is a single scalar), for N atoms we need to calculate $N^2/2$ terms (assuming all unique distances contribute to the energy), and then add them up. How many terms we need to compute for the gradient of the potential?
- Here is another trick question going back to our data structure. Suppose that I am adding strong electric field, E, along the Z axis at a specific direction and the corresponding (added) energy takes the form $U_{E-field} = \sum_i c_i \cdot E \cdot z_i$. Is the data structure that we proposed (keeping the Cartesian components of one particle vector together is still ok?

Calculations of potential gradients are a major source of errors in programming molecular modeling code. It is EXTREMELY useful to check the analytical derivatives against numerical derivatives computed by finite difference, when the expected accuracy should be of at least a few digits. For example

$$\frac{dU}{dq_k} \simeq \frac{U\left(q_1,...,q_k+\Delta/2,...,q_N\right)-U\left(q_1,...,q_k-\Delta/2,...,q_N\right)}{\Delta} \qquad \forall k$$

For the water system the step Δ can be 10^{-6} (using double precision) and the expected accuracy is at least of 3-4 digits. More than that suggest an error.

This concludes (in principle) our discussion of the **steepest descent minimization algorithm. From the above discussion it is quite clear that minimization in the neighborhood of a stationary point of the potential (like a minimum) is difficult**. Near the minimum the gradient of the potential is close to zero, subject to potential numerical error and support the use of only extremely small step that are harder to converge numerically. In that sense the algorithm we discuss next is complementary to the steepest descent approach. It is working well in the neighborhood of a minimum. It is not working so well if the starting structure is very far from a minimum, since the large step taken by this algorithm (**Newton Raphson) relies on the correctness of the quadratic expansion of the potential energy surface near a minimum**.

We start by considering a linear expansion of the potential derivatives at the current position R in the neighbor hood of the desired minimum R_m. At the minimum, the gradient is (of course) zero. We have

$$\left[\nabla U\left(R_m\right)\right]_j = 0 \simeq \left[\nabla U\left(R\right)\right]_j + \sum_i \frac{d^2U}{dq_j dq_i}\left(q_i - q_{im}\right)$$

The entity that we wish to determine from the above equation is R_m the position of the minimum. The square bracket with a subscript $\left[...\right]_j$ denotes the *j* vector element. The last term is a multiplication of matrix by a vector. From now onwards we shall denote the second derivative matrix by $\tilde{U}$. We are attempting to do so by expanding the force linearly in the neighbor hood of the current point. This is one step up in expansion compared to the steepest descent minimization; however, it is not sufficient in general. In the simplest version of the Newton Raphson (NR) approach very large steps are allowed. Large steps can clearly lead to problems if the linear expansion is not valid. Nevertheless, the expansion is expected to be valid if we are close to a minimum, since any function in the neighborhood of a minimum can be expanded (accurately) up to a second order term

$$U\left(R\right) \approx U\left(R_m\right) + \frac{1}{2}\sum_{i,j}\left(R-R_m\right)^t \tilde{U}\left(R-R_m\right)$$

Note that we did not write down the first order derivatives (gradient) since they are zero in at the minimum. This is one clear advantage of NR with respect to steepest descent minimizer (SDM). SDM relies on the first derivatives only, derivatives that vanish in the neighborhood of a minimum. It is therefore difficult for SDM to make progress in thee circumstances while NR can do it in one single “shot” as we see blow.
We write again the equation for the gradient in a matrix form

$$\left[\nabla U\left(R_m\right)\right] = 0 \simeq \left[\nabla U\left(R\right)\right] + \left[\tilde{U}\left(R\right)\cdot\left(R_m - R\right)\right]$$

which we can formally solve (for R_m) as

$$R_m = R - \left(\tilde{U}(R)\right)^{-1} \cdot (\nabla U)$$

The matrix $\left(\tilde{U}\right)^{-1}$ is the inverse of the matrix $\tilde{U}$, namely $\left(\tilde{U}\right)^{-1}\tilde{U} = I$ where I is the identity matrix. In principle there is nothing in the above equation that determine the norm of the step $|R_m - R|$ that we should take. If the system is close the quadratic the second derivative matrix is roughly a constant and a reasonably large step to ward the minimum can be taken without violating significantly validity of the above equation. The ability to take a large step in quadratic like system is a clear advantage of NR compare to SDM. However for systems that are not quadratic the matrix is not a constant and only small steps (artificially enforced) should be used. In our water system NR should be used with care and only sufficiently close to a minimum. There are a few technical points that specifically should concern us with the water dimer optimization problem. We will be concern with the following

1. Does the matrix $\tilde{U}$ have an inverse, and what can we do if it does not?
2. How to find the inverse (or solve the above linear equations)?

The bad news is that the matrix $\tilde{U}$ for molecular systems (as the water dimer is) does not have in general an inverse. The problem is the six degrees of freedom that we mentioned earlier and do not affect the potential energy: three overall translation and three overall rotations have zero eigenvalues making the inverse singular. This is easy to see as follows

Let the eigenvectors of the $\tilde{U}$ be $\hat{e}_i$ and the eigenvalues λ_i. It is possible to write the matrix $\tilde{U}$ as the following sum

$$\tilde{U} = \sum_i \lambda_i e_i \cdot e_i^t$$

where the outer product

$$e_i \cdot e_i^t = \begin{pmatrix} e_{i1} \\ e_{i2} \\ \cdots \\ e_{iN} \end{pmatrix} \cdot \begin{pmatrix} e_{i1} & e_{i2} & \cdots & e_{iN} \end{pmatrix} = \begin{pmatrix} e_{i1} \cdot e_{i1} & e_{i1} \cdot e_{i2} & \cdots & e_{i1} \cdot e_{iN} \\ e_{i2} \cdot e_{i1} & e_{i2} \cdot e_{i2} & \cdots & e_{i2} \cdot e_{iN} \\ \cdots & \cdots & \cdots & \cdots \\ e_{iN} \cdot e_{i1} & e_{iN} \cdot e_{i2} & \cdots & e_{iN} \cdot e_{iN} \end{pmatrix}$$

generates a symmetric NxN matrix. Since the vectors e_i are orthogonal to each other ($e_i \cdot e_j = \delta_{ij}$), it is trivial to write down the inverse in this case

$$\left(\tilde{U}\right)^{-1} = \sum_i \frac{e_i \cdot e_i^t}{\lambda_i}$$

However, this expression is true only if all λ_i are different from zero. The eigenvectors represent directions of motion and the eigenvalues are associated with the cost in energy for moving in a direction determined by the corresponding eigenvector. However moving along the direction of global translation (or global rotation) does not change the energy, therefore their corresponding eigenvalues must be equal to zero, and the inverse impossible to get.

One way of getting around this problem is by "shifting" the eigenvalues of the offending eigenvectors. If we know in advance the six offending eigenvectors we can raise their eigenvalues to very high values (instead of zero) by adding to the matrix outer products of these vectors multiplied by very high value (see below). Contribution of eigenvectors with very high value will diminish when we compute the inverse, since the inverse is obtained by dividing by the corresponding eigenvalues. We do not have to find all the eigenvectors and the eigenvalues as is written above, it is sufficient if we affect the few specific eigenvectors and define a new matrix $\tilde{U}$ to obtain a well behaved $\left(\tilde{U}\right)^{-1}$. The question remains (of course) is how to find these six eigenvectors. The most straightforward (and inefficient) way to do it is to actually compute the eigenvectors and eigenvalues by a matrix diagonalization procedure. Lucky we do not have to do that since the translation and rotation eigenvectors are known from the Eckart conditions. We have for translation

$$\sum_i m_i\left(r_i - r_i^0\right)=0 \qquad r_i = \left(x_i, y_i, z_i\right)$$

and for rotation

$$\sum_i m_i \cdot r_i \times \left(r_i - r_i^0\right) = 0$$

- Mention the possibility of using Lagrange multipliers here

The last multiplication is a vector product, and the difference $r_i - r_i^0$ is assumed to be small. The coordinate vectors r_i^0 are reference vectors used to define the coordinate system and are constants.

For the record, we write the vector product explicitly

$$r_i \times r_i^0 = \begin{vmatrix} e_x & e_y & e_z \\ x_i & y_i & z_i \\ x_i^0 & y_i^0 & z_i^0 \end{vmatrix} = \hat{e}_x \cdot \left(y_i z_i^0 - z_i y_i^0\right) - \hat{e}_y \left(x_i z_i^0 - z_i x_i^0\right) + \hat{e}_z \left(x_i y_i^0 - y_i x_i^0\right)$$

The two equations defined six constraints. This is since they are vector equations, each of the vectors has 3 components – x,y,z. To obtain the eigenvectors associated with these constraints we need to compute the gradients of the above constraints. We have three vectors for translations that we denote by e_{tx}, e_{ty}, e_{tz}, and three vectors for the rotation e_{rx}, e_{ry}, e_{rz}.

Here are the translation vectors

$$\hat{e}_{tx} = \left(m_1, 0, 0, m_2, 0, 0, ..., m_N, 0, 0\right)$$

$$\hat{e}_{ty} = \left(0, m_1, 0, 0, m_2, 0, ..., 0, m_N, 0\right)$$

$$\hat{e}_{tz} = \left(0, 0, m_1, 0, 0, m_2, ..., 0, 0, m_N\right)$$

And here are the rotations.

$$\hat{e}_{rx} = \left(0, m_1 \cdot z_1^0, -m_1 \cdot y_1^0, 0, m_2 \cdot z_2^0, -m_2 \cdot y_2^0, \ldots, 0, m_N \cdot z_N^0, -m_N \cdot y_N^0\right)$$
$$\hat{e}_{ry} = \left(m_1 \cdot z_1^0, 0, -m_1 \cdot x_1^0, m_2 \cdot z_2^0, 0, -m_2 \cdot x_2^0, \ldots, m_N \cdot z_N^0, 0, -m_N \cdot x_N^0\right)$$
$$\hat{e}_{rz} = \left(m_1 \cdot y_1^0, -m_1 \cdot x_1^0, 0, m_2 \cdot y_2^0, -m_2 \cdot x_2^0, 0, \ldots, m_N \cdot y_N^0, -m_N \cdot x_N^0, 0\right)$$

A vector that is orthogonal to the above six does not include overall rotation or translation component. So our goal is to work in the reduced space that does not include the above six. Note that the Eckart conditions are linear so the constraints are constant (independent of the current coordinates). This makes the manipulation of these vectors straightforward to do, and doing it only once at the beginning of the calculation. One approach is to modify input vectors (project out from them the offending part). This is however, quite expensive and will need further work for any incoming vector. A much simpler procedure is to modify the matrix, which is what we shall do.

*** IMPORTANT CORRECTION: For the application of the Eckart conditions to the optimization problem we must set all the masses to one. In the calculation of the potential the masses was not used and it should not be used also in the construction of the constraints. So we have (for minimization)

$$\sum_i \left(r_i - r_i^0\right) = 0 \qquad r_i = \left(x_i, y_i, z_i\right)$$

and for rotation

$$\sum_i r_i \times \left(r_i - r_i^0\right) = 0$$

*** End of correction

Note that the so produced six vectors are not orthogonal. They span the complete six-fold space of eigenvectors with eigenvalues that are equal to zero. However, attempting to use them within our procedure require them to be orthonormal. We are making it into that point by performing Gram Smith process on the space of the constraints' derivatives.

What we do is basically the following. We have a set of N linearly independent vectors We pick one of them at random (call it for convenience $\hat{e}_1$) and normalize it

$$\hat{e}_1^{'} = \frac{\hat{e}_1}{\left|\hat{e}_1\right|}$$

Let $\hat{e}_2$ be the second vector that we pick from the set. We make it orthogonal to $\hat{e}_1^{'}$ and normalize it.

$$\hat{e}_2^{'} = \frac{\hat{e}_2 - \left[\left(\hat{e}_2\right)^t \cdot \hat{e}_1^{'}\right]\hat{e}_1^{'}}{\left|\hat{e}_2 - \left[\left(\hat{e}_2\right)^t \cdot \hat{e}_1^{'}\right]\hat{e}_1^{'}\right|}$$

The next vector on the agenda we make it orthogonal to the previously constructed vector $e_1^{'}$ and $e_2^{'}$ and normalize it. The same process is used for the rest of the base vectors

With orthonormal representation of the constraint space $\left\{e_i^{'}\right\}_{i=1,\dots,6}$, we can redefine the second derivative matrix using a shifting procedure to MUCH higher values for all the overall body motions. We have

$$\tilde{U}' = \tilde{U} + \sum_{i=1,\dots,6} \lambda_{up} e_i^{'} \cdot \left(e_i^{'}\right)^t$$

where λ_{up} is a very large number in accord with the idea that we promote earlier (once the inverse is computed $1/\lambda_{up}$ will be significantly lower than anything else (say $\lambda_{up} = 10^8$ for the water dimmer).

The modified second derivative matrix is now ready to prime time NR optimization, since it has a straightforward inverse (we must be a little careful though, it is possible that a point on the energy surface will be found for which the second derivative is zero even if it is not global motion). Here we ignore this possibility, assuming that we are sufficiently close to a minimum such that all the non-zero eigenvalues are positive. In the case that the quadratic expansion is accurate (and in sharp contrast to SDM) the minimization will converge in one step.

- What will happen to our solution if the eigenvalues are negative?

So far we made an important step forward establishing that the inverse of the adjusted second derivative matrix is likely to exist. There remains the problem of how to determine the inverse efficiently.

In fact we do not need to compute an inverse explicitly since all we need is to solve a linear equation of the type $Ax = b$ where A and b are known matrix and vector, and x is the unknown vector that we seek. It is the same as our problem written as $\left[\tilde{U}(R)\cdot(R_m - R)\right] = -\nabla U(R)$.

We start with a subset of problem that is easy to understand and to solve (triangular problems) and then we work our way up to the full Gaussian elimination.

Triangular problems

Example

$$\begin{pmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

which is rather easy to solve. We can immediately write $x_1 = b_1 \cdot a_{11}$. Using the (now) known value of x_1 we can write for x_2, $x_2 = \left(b_2 - a_{21} \cdot x_1\right) / a_{22}$. Similarly we can write for x_3, $x_3 = \left(b_3 - a_{31} \cdot x_1 - a_{32} \cdot x_2\right) / a_{33}$

For the general case we can write an implicit solution (in terms of the "earlier" $x_j \quad j = 1, \ldots, i-1$)

$$x_i = \left(b_i - \sum_{j=1}^{i-1} a_{ij} \cdot x_j \right) \Big/ a_{ii}$$

Note that a similar procedure applied to the upper triangular matrix

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

To solve a general linear problem we search for a way of transforming the matrix to a triangular form (which we know already how to solve). Formally, we seek the so-called LU decomposition in which the general A matrix is decomposed into a lower triangular matrix L, and an upper triangular matrix U ($A = LU$). Note that if such a decomposition is known, we can solved the linear problem in two steps

Step 1.

$Ax = b$

$LUx = b$

$L(Ux) = b$

$Ly = b$ find y using the lower triangular matrix L

Step 2.

$Ux = y$ find x using the upper triangular matrix U

A way of implementing the above idea in practice is using Gaussian elimination. Gaussian elimination is an action that leads to a LU decomposition discussed above, even if the analogy is not obvious. In this course we will not prove the equivalence.

Gaussian elimination

Consider the following system of linear equations

$$\begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{pmatrix} \begin{pmatrix} x \\ x \\ x \\ x \\ x \end{pmatrix} = \begin{pmatrix} x \\ x \\ x \\ x \\ x \end{pmatrix}$$

where x denotes any number different from zero.
We can eliminate the unknown x_1 from rows 2 to n (the general matrix is of size $n \times n$). We multiply the first row by a_{i1}/a_{11} and subtract the result from row i. By repeating the process $n-1$ times we obtain the following (adjusted) set of linear equations that has the same solution

$$\begin{pmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \end{pmatrix} \begin{pmatrix} x \\ x \\ x \\ x \\ x \end{pmatrix} = \begin{pmatrix} x \\ x \\ x \\ x \\ x \end{pmatrix}$$

We can work on the newly obtained matrix in a similar way to eliminate x_2 from row3 3 to n. This we do by multiplying the second row by a_{i2}/a_{22} and subtract the results from rows 3 to n. The (yet another) new matrix and linear equations will be of the form

$$\begin{pmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \end{pmatrix} \begin{pmatrix} x \\ x \\ x \\ x \\ x \end{pmatrix} = \begin{pmatrix} x \\ x \\ x \\ x \\ x \end{pmatrix}$$

It should be obvious how to proceed with the elimination and to create (an upper) triangular matrix that we know by now how to solve.

- So what is the structure of the water dimer??

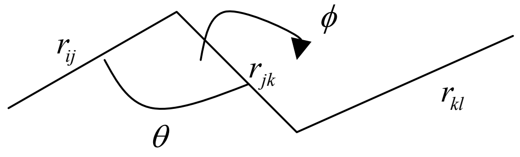
A few comment on the energy function that we have used. The non bonded term (the Lennard Jones and the electrostatic terms) are to a good approximation what is out there today. A number of highly advanced functional forms that described intra and inter biomolecular interactions are using essentially the same formula we wrote above for water. Of course different atoms will be assigned different charges or different Lennard Jones parameters (one of the most widely used procedures is to assign a specific Lennard Jones parameter to an atom and not to a pair, and to use a combination rule between atom types). For example, for interactions between carbon and oxygen atoms we use atomic parameters like $\overline{A}_c$ and $\overline{B}_c$ for carbon and $\overline{A}_o$ and $\overline{B}_o$ for oxygen, to give a pair interaction of the following form

$$U_{LJ}\left(r_{co}\right) = \frac{\overline{A}_c \overline{A}_o}{r_{co}^{12}} - \frac{\overline{B}_c \overline{B}_o}{r_{co}^{6}}$$

In practical terms this (approximate) procedure means that we have only order of N parameters that determine the potential instead of N^2 if all pairwise interactions are considered independent. Similar separation to atomic properties holds of course for the electrostatic in which we used atomic charge to describe interactions. The electrostatic interactions representation by atomic properties is easier to justify than the separation into atomic properties of the LJ terms. Nevertheless, the above separation is used even without a question.

While the potential functional form of the LJ potential and electrostatic of water – water interaction are conceptually the same for much more complex systems, the internal interaction can be significantly more complex. We have used internal interactions to force the individual water molecules to behave rigidly. Other cases need to be flexible.

For example, internal flexibility of a protein chain is clearly very important to describe the folding process. Internal interactions of polymer are usually described by two body terms (distances), three body terms (angles) and four body terms (torsions).



If r_{ij}, r_{jk}, r_{kl} are bond vectors along four chemically linked atoms, an angle term is defined by the scalar product $\cos(\theta_{ijk}) = \frac{r_{ij} \cdot r_{jk}}{|r_{ij}||r_{jk}|}$ $\theta \in [0, \pi]$ and the torsion term is defined as the angle between the two planes that are determined by the two pairs of vectors, one plane by r_{ij}, r_{jk} and a second plane r_{jk}, r_{kl}. The cosine of the angles between the planes is determined as follows. We first determine unit vectors perpendicular to each of the planes by

$e_1 = \frac{r_{ij} \times r_{jk}}{|r_{ij} \times r_{jk}|}$ $\quad e_2 = \frac{r_{jk} \times r_{kl}}{|r_{jk} \times r_{kl}|}$. These vectors are used to obtain $\cos(\phi) = e_1 \cdot e_2$ $\phi \in [0, 2\pi]$.

Because of the larger range of ϕ when compared to θ we need to determine the sign of $\sin(\phi)$ to determine the angle uniquely. We can do it by determining the sign of $r_{jk} \cdot (e_1 \times e_2)$. More details on the energy function of more complex polymers will come later.

So far we consider only minimum energy points of the complete coordinate space. This can be a sound approximation if the lowest energy minimum is very deep. As we shall see in the follow up discussion, however, when the minimum is not so deep then structures that are different from the minimum can make a significant contribution.

Consider first the obvious extreme case in which thee are no interactions at all, i.e. that the potential is a constant and there are no special (minimum energy) configurations. The molecules are simply contained within a volume V and this is it. Rather than thinking of special points we should think about the probability (density) of being at alternative position. For a constant potential and a volume V the probability density is simply $\frac{1}{V}$ (i.e. $\int_0^V \frac{1}{V} dV' = 1$)

Water entropy and free energy
It is possible (and definitely the case for liquid water) that we will find many structures of the same energy. We therefore **need to adjust our thinking of which of the energies (and structures) are most probable**. For example we can have 10^6 conformations of energy E and only one structure of $E-\varepsilon$, should we take the structure with $E-\varepsilon$ as the most probable? The fact that the higher energy is a lot more abundant must count for something. On a golf course the probability of finding the ball not in a hole is much larger than finding it in a hole, (even though the energies of the holes are lower).

With the above discussion we realize that we need to consider **the weight of a given energy in a way that goes beyond its value only**. Consider an energy E, the probability of observing this energy is determined for the purpose of this class in the so-called canonical ensemble – a collection of system states that are characterized by a constant temperature T, volume V and number of particles N. **The weight of an energy is given by** $w(E) = \Omega(E)\exp\left[-\frac{E}{kT}\right]$**.** The constant k is essentially a unit conversion factor from energy to temperature. For example converting from temperature in degrees Kelvin to kcal/mol, the constant *k* is 0.002 (kcal/mol/degrees K). The energy is still a very important factor (the weight is decreasing exponentially as the energy increases), however, it is not the only factor. At the limit of high temperatures the only factor remaining is the number density of configuration at a given energy -- $\Omega(E)$. We use a continuous description in which energy values span a continuum.

To obtain probabilities from the weights we need to normalize the weights

$$P(E) = \frac{w(E)}{\int w(E)\,dE}$$

We call the partition function the normalization factor (that we just wrote above for the probability). **The partition function is usually denoted by** Q (and sometimes also by Z).

$$Q = \int w(E)\,dE = \int \Omega(E)\exp\left[-\frac{E}{kT}\right]dE$$

The partition function can also be written as a direct integral (or a sum in discrete space) over coordinates. In contrast to energies, every coordinate defines a unique state (and therefore there is a no need for a factor of the number of states like in the energy)

$$Q = \int \exp\left[-\frac{E(X)}{kT}\right] dX$$

The integral over the energy is just one dimensional and therefore a lot easier to do that the integral over coordinates of dimensions $3N$. However, this is clearly a misleading representation of the difficulties, since we usually do not know $\Omega(E)$ that requires additional complex calculations. One possibility of computing $\Omega(E)$ (if we are interested in this quantity, and indeed it is of significant interest in statistical mechanics, making the most important function in the microcanonical ensemble – the collection of systems with constant energy, volume, and number of particles) is to compute the following $3N$ integral

$$\Omega(E_0) = \int \delta(E_0 - E(X)) dX$$

where E_0 is the energy value for which we wish to determine the number density, and $\delta(...)$ is the Dirac's delta function, defines as follows

$$\delta(X - X_0) = 0 \quad X \neq X_0$$
$$\int \delta(X - X_0) dX = 1$$

It is possible to think on the delta function as a limit of functions. For example we can consider a box function
$\Delta(X) = 0$ for $X - a/2 > X$ & $X > X_0 + a/2$
$\Delta(X) = 1/a$ for $X - a/2 < X < X + a/2$
at the limit in which $a \to 0$

After this long detour at least we understand that the calculation of the partition function is quite complex and involves a very high dimensional integral. The integral is usually very difficult to perform in realistic systems but it is of exceptional importance in computational statistical physics. One of the reasons is the estimate of what is the probability of finding the system in a thermodynamic state. A thermodynamic state is a collection of many microscopic states, but not all the microscopic states. An example of a thermodynamic state is of liquid water, another thermodynamic state is of solid water. Each one of them contains numerous microscopic states. Another example is of folded and unfolded thermodynamic states of a protein. While we will sometime approximate the folded state of a protein using a single structure, the reality is that even the folded state is a collection of microscopic coordinate sets. It is of course true that the number of folded coordinate sets is much smaller than the number of unfolded coordinate sets.

The relative probability of being in a thermodynamic state A as opposed to thermodynamic state B is given by the ration of their corresponding partition functions.

$$\frac{P(A)}{P(B)}=\frac{Q(A)}{Q(B)}$$

This by itself is a good reason to invest in computing the partition function, but there is more. The partition function makes it possible to compute many thermodynamic functions such as energies (average energies over the relevant ensemble $\bar{E}$, entropies S, free energies F, and more). These entities can be measured in experiments and provide useful tests of our model. Here is a quick list

$$F(A)=-kT\log\left[Q(A)\right]$$

$$\bar{E}(A)\equiv\langle E\rangle=-\frac{\partial\log\left[Q(A)\right]}{\partial(1/kT)}=\frac{\int E(X)\exp\left[-\frac{E(X)}{kT}\right]dX}{\int\exp\left[-\frac{E(X)}{kT}\right]dX}$$

$$S=-\left(F-\bar{E}\right)/T$$

$$C=\frac{\partial\bar{E}}{\partial T}$$

It should be obvious now why people discuss the global FREE energy minimum of a protein. This is the most probable state. The energy alone does not take into consideration the degeneracy of different coordinate states, and the free energy using the correct weight to estimate the state probability is a better measure. In the case of protein folding the difference between energy and free energy is of particular importance since the weight of unfolded state (due to their large number) is considerably larger than their energy alone. Note that at the limit of low temperature ($T\rightarrow 0$, the temperature is always positive), the free energy becomes identical to the energy, assuming only a single and unique global minimum.

This can be shown as follows. Let $E_{\min}$ be the lowest energy, and for simplicity we use a sum instead of an integral. The running index i is over alternative microscopic states.

$$F=-kT\log\left[\sum\exp\left[-E_i/kT\right]\right]=-kT\log\left[\exp\left[-\frac{E_{\min}}{kT}\right]\sum_i\exp\left[-\frac{\left(E_i-E_{\min}\right)}{kT}\right]\right]$$

For temperatures that approach zero each term in the sum in which $E_i-E_{\min}>0$ will become vanishing small. Only one term will remain in which E_i is exactly equally $E_{\min}$. Since we assumed that the ground state is not degenerate the sum is reduced to the number 1.

$$F \underset{T\to 0}{\to} -kT\log\left[\exp\left[-\frac{E_{\min}}{kT}\right]\cdot 1\right] = E_{\min}$$

In the other extreme, at the limit of high temperatures the free energy becomes

$$F \underset{T\to 0}{\to} -kT\log\left[\sum_i 1\right] = -kT\log[J]$$

where J is the number of states. In the continuous integration limit it becomes the volume of the system.

How are we going to estimate the value of the integrals? Consider first a straightforward brute force approach. Let us assume that we approximate each dimension by roughly ten quadrature points. This number of points is of course low but for slowly varying potentials should be reasonable. The problem is however of dimensionality. For most complex systems in biophysics the number of degrees of freedom will be at least in the hundreds. This means that the number of energy evaluation that we will need to compute (one per point of integration) will be 10^{100} which is beyond what we can do today with existing computational resources. (State of the art simulations today are limited by $10^8 - 10^9$ energy evaluations, so there is a long way to go before we will get closer to 10^{100}.

The task at hand seems completely hopeless; nevertheless there is some hope. First we are rarely interested in the absolute value of the free energy (or energy) and only in the difference. This makes it possible to present the calculations of free energy differences (for example) as computations of averaging. And averages can be computed efficiently by randomized algorithms, our next frontier.

Random numbers:
A random variable, X, is a real value function defined on a sample space. We can compute the probability, $p(X)$, that the random variable will take a given value from the sampled space. For a continuous variable X, $p(X)$ is a probability density $\left(\int_a^b p(X)\,dX = 1 \quad p(X) > 0\right)$. The average value of X can be computed (as we know intuitively) by selecting a large numbers of sample points from the sample space.

$$\bar{X} \cong \frac{1}{n}\sum_i X_i$$

Here the exact average is given asymptotically for $n \to \infty$ assuming that $\bar{X}$ and $\bar{X}^2$ exist (see below).

The exact average (mean) is obtained using the probability density (if known)

$$\mu = \bar{X} = \int p(X) \cdot X \cdot dX$$

The intuitive observation that the average converged to a given value is supported by the Law of large numbers:

The Law of Large Numbers
Suppose that $X_1, X_2, ..., X_n$ are random sample of numbers from a distribution for which a mean μ and a variance $\sigma^2 \equiv \langle X^2 \rangle - \langle X \rangle^2$ exist. The average of the sample is defined as

$$\bar{X}_n = \frac{1}{n} \sum_{i=1,...,n} X_i$$

This average is approaching (not surprisingly) the mean μ and the variance is approaching (more surprising) σ^2/n. Her is a quick argument what it is so. In the derivation we assumed that X_i and X_j are independent

$$\begin{aligned}
&\sigma^2(\bar{X}_n) = \langle \bar{X}_n \bar{X}_n \rangle - \langle \bar{X}_n \rangle^2 \\
&= \frac{1}{n^2} \left\langle \left(\sum_{i=1}^{n} X_i \right) \left(\sum_{j=1}^{n} X_j \right) \right\rangle - \frac{1}{n^2} \left\langle \left(\sum_{i=1}^{n} X_i \right) \right\rangle^2 \\
&= \frac{1}{n^2} \left(n \langle X^2 \rangle + (n^2 - n) \langle X \rangle^2 \right) - \frac{1}{n^2} n^2 \langle X \rangle^2 \\
&= \frac{1}{n} \left(\langle X^2 \rangle - \langle X \rangle^2 \right) = \frac{\sigma^2(X)}{n}
\end{aligned}$$

The amusing result of the Law of Large Number is the reduction in the variance as a function of the sampled points regardless of the dimensionality of the system. So in principle we can compute a high dimensional integral (provided that we can present them as an average). Of course we cannot forget about dimensionality completely since we are also required to sample in a way that will **truly reflect the underlining distribution**. The last is a non-trivial issue that depends on the specific problem at hand and is hard to prove even for the simplest cases. We will discuss different examples as we work along. Consider a simple example of using a randomized algorithm for sampling: Estimating the number π

Consider a square with an edge length 1 and a circle embedded in it with a radius of ½. Two independent uniform random numbers (the probability density is a constant) between 0 and 1 make a vector (X_1, X_2) are always found inside the square. The ratio of sample points that are contained within the circle, N_c to the number of points that are contained within the square (all of the sampled points) N_t is approximating

$$\frac{N_c}{N_t} \cong \frac{\pi}{4}$$

We have found a way of approximating the integral

$$\int_0^1 \int_0^1 dX_1 dX_2 \Theta\left(X_1^2 + X_2^2\right)$$
$$\Theta(r) = \begin{Bmatrix} 1 & r \le 1/2 \\ 0 & r > 1/2 \end{Bmatrix}$$

by sampling random numbers. Note that in the above example we sampled random number from a uniform distribution between zero and one. Lucky for us these are the type of (pseudo)-random numbers that we can obtain easily on the computer and are used as a starting point to generate other type of random numbers.

How quickly are we going to arrive to the exact number? There are two components that contribute to this estimate. First, we can use the result of the Law of Large Numbers above to estimate that the rate of convergence will be proportional to $N^{1/2}$. This rate is not great (if we run one million points the accuracy we obtain is just of the order of 0.001). However, at least within the argument given so far it is independent of the dimensionality of the problem and therefore much more appropriate to determine averages of high dimensions. For example, if we wish to determine the average energy (for example) of water.

However, this is only a part of the story, and there is also a second component that is more subtle. We need to sample points in high dimension space according to the pre-specified distribution. It was easy to do with a uniform distribution above. It is not so simple in the cases that we are interested in.

Consider for example an entity like the average energy. I.e. we wish to compute

$$\bar{E} = \frac{\int U(X) \exp\left[-\frac{U(X)}{kT}\right] dX}{\int \exp\left[-\frac{U(X)}{kT}\right] dX}$$

the above expression can be thought as the average of the potential $U(X)$ with a probability density of configurations (coordinates) given by $\frac{\exp\left[-\frac{U(X)}{kT}\right]}{\int \exp\left[-\frac{U(X)}{kT}\right]dX}$. This is the weight divided by a sum over all weights (to make sure that the probability of finding the system somewhere is one.

If we have a way of generating configurations according to the above distribution then we should get convergence rate of $N^{1/2}$ essentially independent of the dimensionality. This is a lot better (in principle) that the exponential growth in computational efforts using the grid. Unfortunately, we do not know (yet) how to generate configurations from the above distribution. , We should be able to generate configurations according to the above weight, and this is not so simple. We can generate random numbers, but how to generate random numbers so that the weight of the configurations would come out as the above exponential weighting with the energy is not so obvious.

Perhaps we should use what we know (generate random numbers with uniform distributions to evaluate the integral (**we can easily change the range of uniform random numbers generated on the computer, (they are generated between 0 and 1) how**?). In the last case the function that we will average over in the nominator will be $U(X)\exp\left[-\frac{U(X)}{kT}\right]$. Actually, we will need to do two averages with the uniform distribution. In the denominator we will need to average $\exp\left[-\frac{U(X)}{kT}\right]$.

These averages unfortunately are very difficult to converge. Most random uniform configurations that we will generate will correspond to high energies (bonds highly distorted, water molecules overlap, and high Lennard Jones potential values). High energies will make exponentially small contributions to the above integral. From the many many points that we will sample only very few will make non-zero contributions making the integral result highly unreliable. There is only a tiny fraction of configuration space that makes a significant contribution to the integral; most of the space makes a zero contribution. It is our job at present to come up with a scheme that will effectively sample the (small) relevant space. This is where the idea of Markov chains is coming into play.

The direct stochastic sampling in molecular systems attempting to average quantities of the type $\exp\left[-\frac{U(X)}{kT}\right]f(X)$ (where $f(X)$ is a slowly varying function compared to

$\exp\left[-\frac{U(X)}{kT}\right]$) is bound to fail since only an extremely small fraction of the points that we try is indeed of any relevance to the integral.

How to generate configurations that make a significant contribution to the integral? There are two main approaches to address this problem: Monte Carlo and Molecular Dynamics. Both approaches generate Markov chains (but of different flavors) that are leading to the desired statistics with conformations that contribute significantly to the integral.

Suppose that I have a sequence of L conformations: $\left(X_1, X_2, ..., X_N\right)$. Choosing yet another conformation X_{N+1} may in principle depend on all the previous set of conformations. We write a conditional probability $P\left(X_{N+1} \mid X_1 X_2, ..., X_N\right)$ to denote the possibility of long-range dependence in the sequence. Note that this sequence is clearly not the same kind of random numbers that we discuss so far that are completely independent. For example when we computed π or made an argument about the Law of Large number we assumed that the correlation does not exist. The average $\left\langle X_i X_j \right\rangle = \left\langle X_i \right\rangle \left\langle X_j \right\rangle$. This is clearly not true when correlation is taken into account
We write well know formula from probability
$P\left(X_i, X_j\right) = P(X_i) P\left(X_j \mid X_i\right) \approx P(X_i) P\left(X_j\right)$
The approximate sign holds as equal only if X_i and X_j are independent and therefore $P\left(X_j \mid X_i\right) \approx P(X_j)$. A Markov chain is a step between having absolutely no memory at all (between sequential structure) to very deep memory that extends to many structures. In Markov chain the memory extend only one step back. That is a Markov chain is defined by a sequence of samples such that for any sample i in the set we have $P\left(X_i \mid X_1 X_2, ..., X_{i-1}\right) = P\left(X_i \mid X_{i-1}\right)$. Our memory in a Markov chain is extended only one step into the past. The probability of a sequence of N specific samples using Markov chains is therefore written as
$P\left(X_1 X_2, ..., X_N\right) = P(X_1) \prod_{i=1,...,N-1} P\left(X_{i+1} \mid X_i\right)$
The right hand side can be interpreted as follows. Give me an initial structure X_1 what is the probability that I will generate with this seed a sequence of structure $X_2, ..., X_N$ when I am given also a transition rule $P\left(X_{i+1} \mid X_i\right)$.

Markov chains can make a powerful tool if they can generate for us configurations that (a) make relevant configurations to a given integral (b) are sampled from a known distribution. Perhaps the most famous Markov chain of configurations is generated by the Metropolis algorithm. The Metropolis algorithm generates configurations that asymptotically reproduce the distribution determined by the weight $\exp\left(-\frac{U(X)}{kT}\right)$, which is one item we are after. Here is the algorithm in brief

1. Start from a configuration $X = X_0$
2. Generate a displacement (step) at random δ and continue a new coordinate vector $X+\delta$. This is called a trial move and it may be accepted as a new configuration (or may not be accepted).
3. Compute the new energy $U\left(X+\delta\right)$ and compare to the previous energy $U\left(X\right)$. Note that the trial move (and of course also the true moves) depend only one step to the past.
4. If the new energy is lower than the previous energy accept the new move and set $X+\delta \to X$.
5. If the new energy is higher or equal the previous energy accept with a probability $\exp\left(-\frac{U\left(X+\delta\right)-U\left(X\right)}{kT}\right)$. If accepted set $X+\delta \to X$, is not set $X \to X$ (even if it is not accepted count the last (trivial) move! And use it as a new configuration)
6. Return to 2 if not done

One of the remarkable features of the Metropolis algorithm is that the configurations indeed converge to have the desired weight $\exp\left(-\frac{U(X)}{kT}\right)$. To show this we need the concept of equilibrium. Equilibrium is obvious when we watch macroscopic systems. We look at a glass of water and there are no changes there. However, we also know that many water molecules are changing their positions during this idea and microscopically, or when we examine a single molecule a lot is happening. It is only when we examine a large number of molecules, or a large number of configurations sampled from the same set of structures that the equilibrium concept makes sense. The equilibrium can be thought of as a sampling process of one molecule from a very large series of water glasses, one water molecule from a glass, or (and alternatively) sampling of one water molecule from the same glass. The repeated perturbation induced by the measurement is expected to be small and if the sampling process is Markovian (or close to that), i.e. rapid loss of correlation between different sampling of water molecules, the two sampling procedure should give the same result. In the language of computations it means that we can use a single long Markov/Metropolis chain to represent independent sampling from similar and identical copies of our system.

Assuming that an equilibrium exists (i.e. if we have a large number of samples the observation is not changing) what will be the equilibrium state?
In an equilibrium state the probability of being at a specific coordinate should not change from one Metropolis step to the next.

We may require
$$P\left(X_i\right)P\left(X_{i+1}^{'} \mid X_i\right)=P\left(X_i^{'}\right)P\left(X_{i+1} \mid X_i^{'}\right)$$

where i is the index of the Markov chain and X and X' denotes different configurations. For simplicity (and sake of argument) assume further that there are only two structures -- X and X', then we have for the transition "rules"

$$P(X'|X) = \min\left[1, \exp\left(-\frac{U(X')-U(X)}{kT}\right)\right]$$

$$\frac{P(X'|X)}{P(X|X')} = \exp\left(-\frac{U(X')-U(X)}{kT}\right)$$

and the highly desired property that

$$\frac{P(X')}{P(X)} = \exp\left(-\frac{U(X')-U(X)}{kT}\right)$$

Computing (canonical) averages in high dimensions

A canonical average is performed with the weight $w(X_i) = \exp\left[-\frac{U(X_i)}{kT}\right]$. Let $A(X)$ be the quantity we are interested to average. Formally, we wish to perform the integral

$$\frac{\int A(X)\exp\left[-\frac{U(X)}{kT}\right]dX}{\int \exp\left[-\frac{U(X)}{kT}\right]dX}$$

The above integral is an estimate of an average that we will also write as $\langle A(X)\rangle$. We sample configurations X_i according to the Metropolis algorithm knowing that the resulting configurations (if everything is done correctly) will be distributed according to the canonical weights.
The following procedure is followed

Initiate a Markov chain from a configuration X_0,
determine (and keep) the energy $U(X_0)$
Set $A_{total} = 0$

Start the Markov chain, and run for N steps.
The current step (can be 0) is denoted by i

1. Generate a small random displacement for the coordinate vector, say δX. For example, if we have N water molecules pick one of the water molecules at random and change the coordinates of this single water molecule (at random). The internal distances within the water molecules are not affected (suggest a

way of achieving that goal). Add the displacement to the current coordinate vector (say X_i) to determine an intermediate coordinate set $\bar{X}_i = X_i + \delta X$

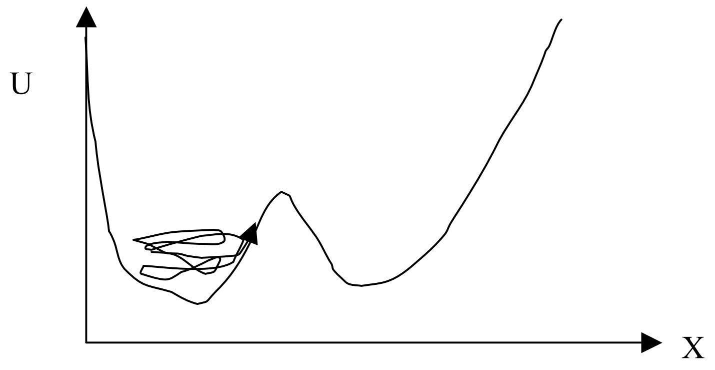
2. Determine the energy of the new configuration, $U\left(\bar{X}_i\right)$
3. Determine the energy difference $\Delta U \equiv U\left(\bar{X}_i\right) - U\left(X_i\right)$
4. If $\Delta U \le 0$ accepts the step ($\bar{X}_i$ now becomes X_{i+1})
5. If $\Delta U > 0$, sample a random number, r, from a uniform distribution between 0 and 1. If $\exp\left[-\frac{\Delta U}{kT}\right]$ is larger than r accepts this step ($\bar{X}_i$ becomes X_{i+1}), otherwise do not accept this trial move (i.e. X_{i+1} is set to be the previous step X_i)
6. Compute $A\left(X_{i+1}\right)$
7. Add to the total sum of $A-s$: $A_{total} = A_{total} + A\left(X_i\right)$
8. If number of steps less than N keep looping (return to 1), otherwise exit loop

On exiting loop computes average: $A_{average} = A_{total}/N$

Comment: In the calculations of the averages, we did not employ any weight. All the values of *A* were summed directly. Shouldn't we multiply by the Boltzmann factor ($w\left(X_i\right) = \exp\left[-\frac{U\left(X_i\right)}{kT}\right]$)?

Another comment: sometimes the first few hundreds or thousands steps in the averaging are ignored (not used in the calculation of the average), any idea why?

Yet another comment: potential problems with the sampling are correlation between the points (correlations if still finite, makes the convergence slower than $1/\sqrt{N}$). Another potential problem is the existence of barriers that restrict sampling to only part of the space (see schematic view below)



Free energies
Let us now think on free energies. How can we compute the free energies? The immediate problem is that we cannot write them (in a simple way) as averages.

$$F = -kT\log\left[\int dX \exp\left[-\frac{U(X)}{kT}\right]\right]$$

May be what we should do is to try the trick below

$$\frac{1}{F} = -kT\log\left[\frac{\int \exp\left[+\frac{U(X)}{kT}\right]\cdot\exp\left[-\frac{U(X)}{kT}\right]dX}{\int \exp\left[-\frac{U(X)}{kT}\right]dX}\right] = -kT\log\left[\left\langle \exp\left[+\frac{U(X)}{kT}\right]\right\rangle\right]$$

when we write an average like $\langle ... \rangle$ it is to be understood that the weight is canonical. Can you suggest why the trick above will NOT work?

While procedures to compute absolute free energies are hard to come by, algorithms to compute free energy DIFFERENCFES are available. We show below two common approaches (and explain the differences) to compute free energy differences. The methods are the free energy perturbation method and thermodynamic integration.

Free energy perturbation:
Consider the free energy difference:

$$F_A - F_B = -kT\log\left[\int \exp\left[-\frac{U_A(X)}{kT}\right]dX\right] + kT\log\left[\int \exp\left[-\frac{U_B(X)}{kT}\right]dX\right] =$$

$$= -kT\log\left[\frac{\int \exp\left[-\frac{U_A(X)}{kT}\right]dX}{\int \exp\left[-\frac{U_B(X)}{kT}\right]dX}\right] =$$

$$= -kT\log\left[\left[\frac{\int \exp\left[-\frac{\{U_A(X)-U_B(X)\}}{kT}\right]\exp\left[-\frac{U_B(X)}{kT}\right]dX}{\int \exp\left[-\frac{U_B(X)}{kT}\right]dX}\right]\right] =$$

$$= -kT\log\left\langle \exp\left[-\frac{\{U_A(X)-U_B(X)\}}{kT}\right]\right\rangle_B$$

The last expression looks like an ordinary average of the entity $\exp\left[-\frac{\{U_A(X)-U_B(X)\}}{kT}\right]$, which is the energy difference between the two states computed with different potentials (the potentials associated with the different states: A and B, and then their difference is put into the exponent. Note also that the weight is done with respect to state B and we added to the usual weighted averaging the subscript $\langle\ldots\rangle_B$. Of course similar expression can be derived when the averaging is done over state A. Note that the expression so far is exact and the term "perturbation" is misleading.

Since the most important issue application of free energies is to compute their difference (this way we know which state is more likely) it looks like our problem is solved. There is however a subtle difficulty that will motivate us to look for an alternative. Numerical calculations with free energy perturbation are not so popular these days because of the following problem:
We are attempting to average a difference in energy that we put in the exponent. The exponent is a very sensitive function of the differences. The worst difference that can happen to us (from the perspective of proper averaging and good statistics) is a coordinate set with large negative difference in energy. In this cased the value of the exponent can be enormous. For example a single point at room temperature ($kT = 0.6$ kcal/mol) with energy difference of –30kcal/mol will have a weight of $7.2\cdot 10^{11}$. This rare fluctuation will overwhelm all other observation and we will have in practice only one sampling point, even if we run for hundreds of million steps, each with a contribution of order 1.

It is therefore a good idea to try to make the states as similar as possible and to compute the differences of only very similar states. Consider a parameter λ that interpolates between states A and state B. For example, we define
$U_\lambda(X) = (1-\lambda)U_A + \lambda U_B$ where $\lambda \in [0,1]$. We can write

$$dF_\lambda = F_{\lambda+d\lambda} - F_\lambda = -kT\log\left[\left\langle \exp\left[-\frac{U_{\lambda+d\lambda}-U_\lambda}{kT}\right]\right\rangle_\lambda\right]$$

$$F_B - F_A = \int_0^1 dF_\lambda$$

Every individual average is much simpler to perform (considering the problem with exponential weighting that was mentioned above). This approach has much better convergence property that the direct approach that we mention earlier. Problems, however, remain.

Even if the states are almost identical, a situation we can achieve using a fine interpolation parameter (which means that most of the differences are very close to zero, and the free energy difference is close to zero too), fluctuations can cause a lot of trouble. A sample of 100 million points all with values close to zero may still be dominated by a single fluctuation and large exponent. The extreme sensitivity of the results to negative energy difference (due to the exponent) makes the above integral very difficult to converge.

Another adjustment to help overcome this problem is described below.

Thermodynamic integration:
Consider a continuous variable λ that interpolates from states A and B. Let us assume that the expansion we have in $\delta\lambda$ is indeed small in that case we can expand the exponent up to the first order. This is not completely kosher since we just argue the exponent can be large, what is done here is to get better statistics using a poorer approximation to the integral. The poorer approximation makes it possible however to get better statistics.
We have

$$dF_\lambda = F_{\lambda+d\lambda} - F_\lambda = -kT\log\left[\left\langle \exp\left[-\frac{U_{\lambda+d\lambda}-U_\lambda}{kT}\right]\right\rangle_\lambda\right] \approx$$

$$\approx -kT\log\left[1-\left[\left\langle\left[-\frac{U_{\lambda+d\lambda}-U_\lambda}{kT}\right]\right\rangle_\lambda\right]\right] \approx \left[\left\langle\left[-\frac{U_{\lambda+d\lambda}-U_\lambda}{kT}\right]\right\rangle_\lambda\right]$$

$$F_B - F_A = \int_0^1 dF_\lambda$$

which by the end of the day looks remarkably simple (just an average over the energy differences

Umbrella sampling and generalized ensembles.

One way of thinking on umbrella sampling (to be described below) is as an alternative way of computing free energy differences (alternative to free energy perturbation). This is the way we are going to present the umbrella sampling technique below. However, we should keep in mind that the way the technique was introduced and widely used has a different flavor.
The text below introduces the umbrella sampling in a perturbation-like-language, but we then show how hydrophobicity can be solved in the different language. It will also be a good opportunity to discuss hydrophobicity, one of the most important biochemical forces in biology, and to demonstrate one of the first applications of the umbrella sampling procedure. Finally, we will discuss how the umbrella sampling procedure can be extended into the most advanced techniques available today of generalized ensembles.

Consider an interpolating parameter λ between the two states, which are as usual A and B. As in the free energy perturbation case we define the potential energy $U_\lambda(R)$. The potential is a function of the coordinates of the atoms in the system -- R, and depends parametrically on the variable λ. The interpolating parameter is defined in the range

$\lambda \in [0,1]$ such that $U_0(R) \equiv U_A(R)$ and $U_1(R) = U_B(R)$. At present we shall not assume a linear dependence of $U_\lambda(R)$ on λ. This will allow more natural introduction of hydrophobicity and the concept of reaction coordinates. As before we are interested in the difference between free energies of different λ variables.

The important difference in the umbrella sampling approach as compared to the free energy perturbation method is the concept of λ as a usual variable (like a newly added coordinate). In that sense the potential $U_\lambda(R)$ becomes $U(R,\lambda)$ where λ can take any value like the Cartesian coordinate of one of the atoms. Our interest remains of course at the specific values $\lambda = 0$ and $\lambda = 1$. However, to speed up the convergence we can use different set of tricks to compute the free energy difference with the extended form of the potential and the sampling space. The free energy of state $\lambda_\#$ is now defined as

$$F_{\lambda_\#} = -kT \log\left[\int \delta(\lambda - \lambda_\#) \exp\left[-\frac{U(R,\lambda)}{kT}\right] dR d\lambda\right]$$

The difference seems at first sight trivial, since we introduced a new variable λ and immediately eliminates it with the help of Dirac's delta function -- $\delta(\lambda - \lambda_\#)$ (for example $\lambda_\# = 0$ for the A state). In the next step we remove the delta function and put instead a biasing potential in λ. We modify our potential energy to include a term that restrains the values of the interpolating parameter to the neighborhood of $\lambda_\#$.

Here comes the modified potential $U_\# = U(R,\lambda) + \kappa(\lambda - \lambda_\#)^2$ where κ is an empirically determined constant. The constant is expected to be high so that the sampled fluctuations in λ, (λ is now a coordinate like any other), will be small and restricted to the neighborhood of $\lambda_\#$.

We define an alternative free energy to the above, which we denote by $\bar{F}_{\lambda_\#}$.

$$\bar{F}_{\lambda_\#} = -kT \log\left[\int \exp\left[-\frac{U_\#}{kT}\right] dX d\lambda\right]$$

All entities (say A) associated with biased sampling will be denoted by $\bar{A}$. The newly defined free energy is clearly NOT the physical free energy that we are after. The really clever component of the umbrella sampling procedure is the conversion of a distribution sampled with a biasing potential to the true distribution. Consider the probability of observing λ_1 in a biased sampling. We have

$$\bar{P}_{\lambda_\#}(\lambda_1) = \frac{\int \delta(\lambda - \lambda_1) \exp\left[-\frac{U_\#}{kT}\right] dX d\lambda}{\int \exp\left[-\frac{U_\#}{kT}\right] dX d\lambda}$$

We remind the reader that log ratio of the probabilities is directly proportional to the free energy difference. So the discussion above of probabilities (in the biased ensemble) is equivalent to a discussion of free energies. It is important to emphasize that there are two λ -s above - $\lambda_\#$ is the value at the minimum of the biasing potential, and λ_1 is an arbitrary value that we wish to sample. Due to the biasing potential we anticipate that most observed values of λ_1 will be close to $\lambda_\#$. Note however that $U\left(R,\lambda\right)$ also influences the distribution of the observed λ -s and the combined effect is not obvious. Hence the distribution is not necessarily symmetric and Gaussian-like as expected from the biasing potential alone.

It is of interest to compare the biased distribution to the distribution without the bias. We have

$$P\left(\lambda_1\right)=\frac{\int \delta\left(\lambda-\lambda_1\right)\exp\left[-\frac{U}{kT}\right]dXd\lambda}{\int \exp\left[-\frac{U}{kT}\right]dXd\lambda}$$

A few manipulations of the exact (unbiased) distributions are carried below. A quick reminder is that $U_\# = U+\kappa\left(\lambda-\lambda_\#\right)^2$

$$P\left(\lambda_1\right)=\frac{\int \delta\left(\lambda-\lambda_1\right)\exp\left[-\frac{U_\#}{kT}\right]\exp\left[\frac{\kappa\left(\lambda-\lambda_\#\right)^2}{kT}\right]dXd\lambda}{\int \exp\left[-\frac{U_\#}{kT}\right]dXd\lambda}\cdot$$

$$\cdot\frac{\int \exp\left[-\frac{U_\#}{kT}\right]dXd\lambda}{\int \exp\left[-\frac{U_\#}{kT}\right]\exp\left[\frac{\kappa\left(\lambda-\lambda_\#\right)^2}{kT}\right]dXd\lambda}$$

It may take more than a single stare, but the above expression is indeed identical to the more direct expression for $P\left(\lambda_1\right)$ written above. The above expression is a proof that a complex expression can always be written in yet more complicated way. Nevertheless, there are some interesting aspects to the above expression.

The new expression has two ratios that were deliberately separated into two lines that look (each) like an average. Averages are our holy grail for Markov chain – Metropolis calculations. We write

$$P\left(\lambda_1\right)=\frac{\left\langle \delta\left(\lambda-\lambda_1\right)\exp\left[\frac{\kappa\left(\lambda-\lambda_\#\right)^2}{kT}\right]\right\rangle_\#}{\left\langle \exp\left[\frac{\kappa\left(\lambda-\lambda_\#\right)^2}{kT}\right]\right\rangle_\#}=\exp\left[\frac{\kappa\left(\lambda_1-\lambda_\#\right)^2}{kT}\right]\frac{\left\langle \delta\left(\lambda-\lambda_1\right)\right\rangle_\#}{\left\langle \exp\left[\frac{\kappa\left(\lambda-\lambda_\#\right)^2}{kT}\right]\right\rangle_\#}$$

An observation: the term $\left\langle \exp\left[\frac{\kappa(\lambda-\lambda_{\#})^2}{kT}\right]\right\rangle_{\#}$ is independent of λ_1 and is a multiplying "constant" if we consider alternative values for λ_1. This is important since if we change now from λ_1 to another λ (say λ_2), the above expression remains constant, and can be eliminated from the equation as we show below.

Another observation is that we were able to write the exact average in terms of the average with the biasing potential. We have $\bar{P}_{\lambda_{\#}}(\lambda_1) = \langle \delta(\lambda-\lambda_1)\rangle_{\#}$ which is the average we computed with the biasing potential, on the other hand $P(\lambda_1)$ is what we really wanted. We write

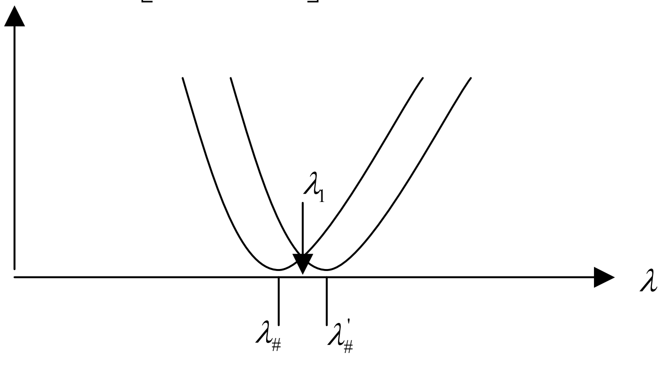
$$P(\lambda_1) = \bar{P}_{\lambda_{\#}}(\lambda_1)\cdot \exp\left[\frac{\kappa(\lambda_1-\lambda_{\#})^2}{kT}\right] C_{\#}$$

$$C_{\#} = \left\langle \exp\left[\frac{\kappa(\lambda-\lambda_{\#})^2}{kT}\right]\right\rangle$$

The advantage of having the biased potential in this case (as opposed to just compute $P(\lambda)$ in a straightforward way) is that the statistics in the neighborhood of $\lambda_{\#}$ is likely to be excellent. This is not obvious at all in simulating $P(\lambda)$ directly since most observed value of λ in the Markov chain could appear elsewhere (i.e. far from $\lambda_{\#}$) resulting in poor estimate of the probability near $\lambda_{\#}$. On the other hand the calculation with the biased potential leaves us with an unknown, the entity that we called above $C_{\#}$.

Consider next a different biasing potential in which the equilibrium position $\lambda_{\#}$ is slightly shifted to $\lambda_{\#}^{'} = \lambda_{\#} + \Delta$. We can still write (and compute) the distribution at λ_1 since by constructing the new biasing potential leads to significant sampling at λ_1 as well

$$P(\lambda_1) = \bar{P}_{\lambda_{\#}^{'}}(\lambda_1)\cdot \exp\left[\frac{\kappa(\lambda_1-\lambda_{\#}^{'})^2}{kT}\right] C_{\#}^{'}$$



If we divide now the two equations for $P(\lambda_1)$ (computed with the two biasing potentials) we have

$$1 = \frac{\bar{P}_{\lambda_\#}(\lambda_1)\cdot \exp\left[\frac{\kappa(\lambda_1-\lambda_\#)^2}{kT}\right] C_\#}{\bar{P}_{\lambda'_\#}(\lambda_1)\cdot \exp\left[\frac{\kappa(\lambda_1-\lambda'_\#)^2}{kT}\right] C'_\#}$$

$$\frac{C'_\#}{C_\#} = \frac{\bar{P}_{\lambda_\#}(\lambda_1)\cdot \exp\left[\frac{\kappa(\lambda_1-\lambda_\#)^2}{kT}\right]}{\bar{P}_{\lambda'_\#}(\lambda_1)\cdot \exp\left[\frac{\kappa(\lambda_1-\lambda'_\#)^2}{kT}\right]}$$

Note that the left side of the lower equation is independent of λ_1 which means that the right hand side equation must be independent of λ_1 as well. Therefore we can choose any value of λ_1 (repeatedly) and estimate the ratio of the constants. The ratio of the constants is all we need (no need for absolute value) since we are interested in the ratio of probabilities (which is equivalent to free energy difference). For example

$$F_A - F_B = -kT\log\left[\frac{P(0)}{P(1)}\right] = -kT\log\left[\frac{P(0)}{P(\Delta)}\frac{P(\Delta)}{P(2\Delta)}\cdots\frac{P(N\Delta)}{P(1)}\right]$$

where each of the ratios inside the log can be estimated as described above

Hydrophobic force

The hydrophobic free energy, one of the most significant factors in biochemistry and biophysics leads to aggregation of non-polar groups in aqueous solutions. The aggregation minimizes their exposed surface area with water. It is the “force” that drives protein folding, drives the formation of membranes, and helps in the establishment of protein-protein interactions.

The hydrophobic free energy is a source of considerable theoretical and computational debate. The basic mechanism of what the hydrophobic interaction is all about can be understood qualitatively in the following terms.

The costs of breaking hydrogen bonds in water are very high (a hydrogen bond is defined by a hydrogen atom bridging between two negatively charged atoms (significant partial charge)). In water the set up will be a hydrogen atom bridging between two oxygen atoms. The hydrogen is covalently bonded to one of the oxygen atoms. In reality it can also hop between the oxygen atoms and change the identity of the oxygen atom it is linked to. This mechanism however, is not included in our model of the water molecule.

Since the costs of the hydrogen bonds are high the system is trying to keep them as much as possible (the probability of conformations with smaller number of hydrogen bonds decay exponentially following Boltzmann factor). Every oxygen has the option of two hydrogen bonds maintaining overall tetrahedral geometry once the covalently linked hydrogen atoms are also taken into consideration. In this case we anticipate a lot of options for reorganization and hoping of water molecules and hydrogen bonds. In other words there are many alternative configurations that satisfy a large (maximal) number of hydrogen bonds and correspond to roughly the same energy values. We have a large number of degenerate configurations with about the same energy. The state of the system is described by an ever growing entropy that is given by the the number of available conformations.

And here comes the hydrophobic effect. If we place non-polar groups in water (groups in which the partial charges on the atoms are very small (or zero)), there are no hydrogen-bonding partners in near the non-polar group. Since the number of hydrogen bonding is large (see previous paragraph), all is not lost and the water molecules will find an alternative hydrogen bond to a nearby water molecule. However, the number of alternatives that it now has is reduced. This reduction is translated into a decrease in the entropy and an increase in the free energy (less probably state). So now we understand why water environment does not like non-polar solutes. Oil and water create separate phases. Again, it is important to emphasize that there is no direct repulsion between the non-polar and water molecules. It is an interruption to the number of choices a hydrogen bond seeker can have.

It is intuitively appealing to think on the hydrophobic effect as proportional to the surface of the non-polar entity that is embedded in the water. The larger is the non-polar surface area the more candidates to hydrogen bonding are lost. Some phenomenological models of hydrophobicity therefore write the free energy as $F = \gamma A$ where γ is a constant and A is the non polar surface area. This model makes it possible to ignore the explicit water effect. However, it is clearly less accurate (model based) compared to the direct calculations of water molecules near hydrophobic surfaces.

What will happen if we have two non-polar spheres immersed in water? We can base our argument on the above discussion. Given that the non-polar spheres are already in the water and we are unable to send them away at least we should minimize the “damage”. This means try to minimize the surface area that the water molecules “see”. The way to do this is to bring the hydrophobic group together, in a way that some of the surface is shared between the non-polar groups and is hidden from the surrounding water molecules. In an essence the entropy of the water molecules drives the hydrophobic residues together and acts as an attractive force between the hydrophobic residues (of course in the above description there is absolutely no force between the non-polar groups, it is all in the water).

How we can test if the above hand waving argument actually reflects reality or at the least agrees with a detailed molecular model that was parameterized against other properties?

We can compute the free energy difference for the case in which the two non-polar sphere as a function of their distance separation - q. The absolute free energy looks like something that we discussed already quite extensively

$$F(q_0) = -kT\log\left[\int \delta(q-q_0)\exp\left[-\frac{U(R,q)}{kT}\right]dRdq\right]$$

and the free energy differences are therefore easy to write down (for example, in the framework of free energy perturbation)

$$F(q_0+\delta) - F(q_0) = -kT\log\left[\frac{\int \exp\left[-\frac{U(R,q_0+\delta}{kT}\right]dR}{\int \exp\left[-\frac{U(R,q_0}{kT}\right]dR}\right]$$

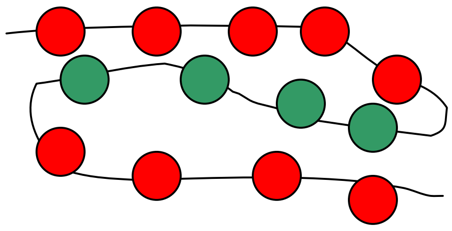
$$= -kT\log\left\langle \exp\left[-\frac{U(R,q_0+\delta)-U(R,q_0)}{kT}\right]\right\rangle_U$$

The above expression can be computed using the Metropolis algorithm since they are exactly averages and free energy differences of the type that we discussed before.

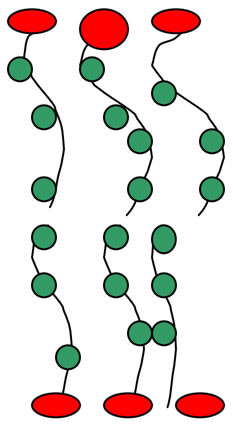
Can you suggest how the calculation with umbrella sampling will look like?

If the hydrophobic force is reflected in our model then we should have a minimum of the free energy at small distances. The free energy will increase until it will reach a flat value at large distances. The observation of such a minimum was an important step in the detailed (microscopic) understanding of hydrophobic forces.

Keeping in mind the model it is now easy to understand why protein collapse to a compact shape (but not necessarily why they fold into a unique structure). Pictorially the one-dimensional chain collapse in away that non-polar residue will aggregate together and polar and charged residues (amino acids) will point to the water. The picture below uses green to denote hydrophobic non-polar residues and red to denote polar residues (hydrogen bond providers).



Membranes are different from protein in the way the individual chains of the forming molecules (lipids) are arranged. It is mostly non polar and only the head is charged. They therefore aggregate (among numerous other possibilities) to form bilayers (water is on top and below, and the membrane should be thought as extended to the right and to the left)



Brownian dynamics

So far we were only concerned with equilibrium averages. The Markov chain and the Metropolis algorithm are not concerned with time scales but with efficient way of computing averages. Many questions in biophysics that are of interest to us are connected with time scales. How quickly a protein folds? How quickly a certain type of enzymatic reaction occurs? These questions are of high relevance to biology, since many of the processes occur at conditions that are far from equilibrium, and order of events is determined by kinetics and not necessarily by stability and equilibrium considerations. The type of questions just posed means that we have to follow explicitly the time dependence of the different processes in order to decide what are the likely events of the next time frame. For this purpose we need a model that can compute also dynamics (time dependence) in addition to equilibrium average.

The simplest attempt in that direction is to use the Metropolis Markov chains we discussed and to argue that they are also likely to present the true dynamics of the system (i.e. arguing that sequential structures along the Markov chain are also sequential structures in time). This clearly depends on the nature of the random step that we have chosen. If the random step is correctly reflecting the sequence of events then the true time formulation may be similar to the Metropolis sequence of structures. The above argument (that is sometimes founding the literature) is however a tautology, and there is no reason to assume to start with that a Markov chain chosen arbitrarily provides a good description of the system evolution in time.

The time evolution of the system is described by a differential equation. The most famous equation of them all is the Newton's equation

$$M\frac{d^2X}{dt^2} = -\frac{dU}{dX}$$

In this equation the mass matrix (M) multiplies the vector of accelerations $\left(\frac{d^2X}{dt^2}\right)$ and is equal to the force $\left(-\frac{dU}{dX}\right)$.

This equation of motions can be solved by different numerical procedures, which will be discussed later. At the moment we start with a different model of dynamics, which is closely linked to the minimization algorithms we already discussed, and therefore provide an interesting viewpoint. We wrote a minimization algorithm (SDM) as a solution of a differential equation:

$$\frac{dX}{dt} = -\nabla U$$

A somewhat related dynamical model (related to the minimization) is the model of Brownian dynamics. It is the same model that explains why smoke particles are going up near a fire despite the fact that the particles are heavier than the air, and was studied first by (of course) Brown. The differential equation for Brownian dynamics is very similar to the above minimization algorithm with one important difference, we write

$$\gamma\frac{dX}{dt} = -\nabla U + R$$

The variable γ is a friction constant that is multiplying the velocity (the derivative of the coordinate with respect to time). Hence the friction coefficient time the velocity provide the frictional force that is mass independent (good for the smoke particle), and must be equal for the rest of the forces in the system that are written on the right hand side of the equation. There are two forces on the right hand side. One of them is the usual systematic force that is derivable from a potential $(-\nabla U)$, which we sort of know and understand. The second term (R) is an interesting new item. It is a random force that mimics (for the smoke case) the "kicks" hot air molecules provide to the smoke particle, kicks that we do not follow explicitly and wish to model indirectly. In case of protein simulations its role is to mimic the "kicks" of water molecules (for example) on the protein chain. These kicks provide excess energy to the system and prevent it from collapsing to a minimum that would have been obtained in the absence of this random force (then only theminimization algorithm would have been remained).

What are the properties of the random force and how can we obtain a solution for this stochastic differential equation? Well, the random force R is usually assumed to be distributed normally, such that $\langle R \rangle = 0$. The average here is over time, i.e. if we take many time slots (like in the sequence of the Metropolis Markov chain) and use the different times to do the averaging the overall average of the random force will be zero. The normal distribution of random forces for a particular time *t* therefore looks something like

$$P\left(R(t)\right) \propto \exp\left[-\frac{R(t)^2}{2\sigma^2}\right]$$

and what remains to be determined is the variance, which can be extracted from the expression below (the expression is a realization of a general theorem called dissipation-fluctuation theorem – the fluctuations here are of the random force R and the dissipation is presented by the friction constant γ)

$$\left\langle R(t)R(t')\right\rangle = 2\gamma kT\delta(t-t')$$

Note that a few more observations on the process were added to the above equation. First the random forces are uncorrelated in time. Because of the delta function $\left(\delta(t-t')\right)$ on the right hand side equation, we conclude that the random forces at different times are uncorrelated. We also note that there is connection between the random force and the temperature (kT). In fact, without bothering with a proof (which is beyond the scope of the present lecture), Brownian dynamics is guaranteed to give us structures that in the final equilibrium state will be sampled with out favorite weight $w \propto \exp\left(-\frac{U}{kT}\right)$, which is nice. The random force that feeds energy is balanced by the minimization part of the algorithm that is taking energy way. The balance between the random force and the SDM is done in such a way that by the end (rather surprisingly) the structures are sampled with the "temperature" weight. This is the reason the temperature appears in the variance in the above formula for the force fluctuations.

How are we solving this equation numerically? We will use a discrete time step Δt and solve for the coordinate vector in small time steps. This is similar to what we have done before in minimization. On the new side we are required to sample random numbers from a Gaussian distribution with a given mean and variance. If we know how to do that (and for the moment we are going to assume that we know) we can straightforwardly write down a suggestion for an algorithm

Algorithm:

Initialize coordinates -- X_0

1. Sample a random force from a Gaussian distribution with a mean of zero $\langle R\rangle = 0$ and variance $\left\langle R^2\right\rangle = 2\gamma kT \cdot \frac{1}{\Delta t}$ (where Δt is the time step used for the integration). Can you explain how to move from the general formula for the force fluctuations to the formula above for the Gaussian variance of the finite difference formula?

2. Compute a step in coordinate $\Delta X = \frac{\Delta t}{\gamma}\left(-\frac{dU}{dX} + R\right)$
3. Compute a new configuration $X_{i+1} = X_i + \Delta X$
4. Check that the total number of steps has not been reached, if not go to 1.

Rigorously the algorithm can be more tricky since the extreme short range of the correlations for the random force are making it a little difficult to handle (it is a discontinuous function in time that is non-trivial to integrate). From practical view point we recognize that true discontinuity of this type did not occur in nature (even a water molecule that provides a kick to a protein is actually correlated on a short time scale) and we will remain with the above simple algorithm that assume loss of correlation within the time window of Δt.

Random numbers and computers

Clearly a number that is produced on the computer in a deterministic way cannot be truly random. So a valid question is what do we mean by a "random number" and how can we test it?

We consider pseudo-random numbers that share some properties with random numbers but obviously are reproducible on the computers and therefore are not truly random.

A useful definition of true random numbers is lack of correlations. If we consider the product of two random numbers r_1 and r_2, -- $r_1 \cdot r_2$ and we average over possible values of r_1 and r_2, we should have $\langle r_1 \cdot r_2 \rangle = \langle r_1 \rangle \langle r_2 \rangle$.

So a test for a random number generator would be $\left\langle \prod_{i=1}^{N} r_1 .... r_N \right\rangle \overset{?}{=} \prod_{i=1}^{N} \langle r_i \rangle$

Essentially all the existing random number generators fail eventually on this kind of test. The common generators are cyclic in nature. There is an L -- large integer such that $r_{i+L} = r_i$, hence the number of random numbers that can be generated is finite.

Widely used random number generators are based on the following simple (and fast) operations:

$$I_{k+1} = \alpha I_k + \beta \ \text{(mod m)}$$

The integers I_k are between zero and m-1. Dividing by m provides a floating point between 0 and 1. If all is well the sequence of the integers is uniformly distributed at the interval [0,1]

Example: $\alpha = 2,147,437,301$ $\beta = 453,816,981$ m=2^{32}

- Using random numbers suggests a procedure to estimate π

To improve the quality and the "randomness" of numbers generated by the above procedure it is useful to have a long vector of random numbers and to shuffle them (randomly)
In MATLAB
Rand(n,m) provide an nxm matrix of random numbers.

The above procedure provides random numbers generated from a uniform distribution. Can we generate random numbers from other probability distribution (e.g. normal)?

A general procedure for doing it is based on the probability function. Let $p(x)dx$ be the probability of finding between x and $x+dx$. Suppose that we want to generate a series of points x and then compute a function of these points $y(x)$. What will be the distribution of the y -s? It will be connected to the probability function of the x -s.

$$p(x)dx = p(y)\left|dy\right|$$

$$p(y) = p(x)\left|\frac{dx}{dy}\right|$$

Example: suppose $y(x) = -\log_e\left(x\right)$

$$p(y)dy = \left|\frac{dx}{dy}dy\right| = e^{-y}dy$$

Another example: Gaussian

We want

$$p(y)dy = \frac{1}{\sqrt{2\pi}}\exp\left[-y^2/2\right]dy$$

select

$$y_1 = \sqrt{-2\log(x_1)}\cos(2\pi x_2)$$
$$y_2 = \sqrt{-2\log(x_1)}\sin(2\pi x_2)$$

$$x_1 = \exp\left[-\left(y_1^2 + y_2^2\right)/2\right]$$
$$x_2 = \frac{1}{2\pi}\arctan\left(\frac{y_2}{y_1}\right)$$

$$\begin{vmatrix} \dfrac{\partial x_1}{\partial y_1} & \dfrac{\partial x_1}{\partial y_2} \\ \dfrac{\partial x_2}{\partial y_1} & \dfrac{\partial x_2}{\partial y_2} \end{vmatrix} = -\left[\frac{1}{\sqrt{2\pi}}\exp\left(-y_1^2/2\right)\right]\left[\frac{1}{\sqrt{2\pi}}\exp\left(-y_2^2/2\right)\right]$$

Note that there is one-to-one correspondence between x and y

Another procedure of generating Gaussian random numbers (we are specifically interested in Gaussian random number following the introduction to Brownian dynamics) is the use of Central Limit Theorem.

This is a remarkable theorem that states the following. Given a sample space, X is a random variable and $p(X)$ is the probability density of this variable. The probability density is arbitrary, but it is assumed that the first and the second moment of the distribution exists. I.e.
$\mu = \int p(X)XdX$ and $\sigma^2 = \int p(X)(X-\mu)^2 dX$ are both finite. (This is not a trivial request. A popular distribution in natural sciences is of Lorentzian $p(X) = \dfrac{1}{a+X^2}$ $X \in [-\infty, \infty]$. This distribution has no finite second moment).

The central limit theorem (CLT) is concerned with a random variable Y which is a summation over random variables X_i that are sampled independently, potentially from different distribution functions (with finite first and second moments).

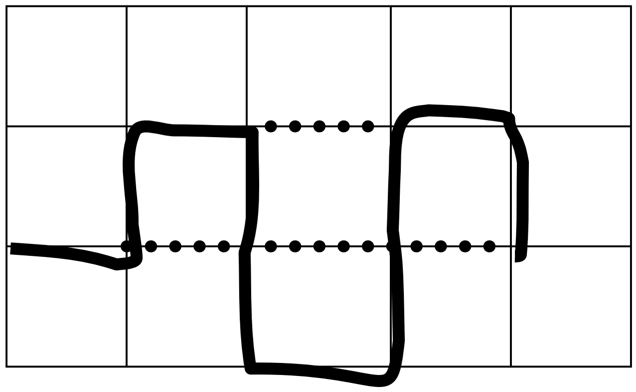
$$Y_j = \sum_{i=1,\dots,N} X_i$$

According to the CLT for $N \to \infty$, $p(Y)$ approaches a Gaussian distribution such that $\mu_Y = \sum \mu_{X_i}$ and $\sigma_Y^2 = \sum \sigma_{X_i}^2$

While this is an asymptotic result in practice the sum of a few tens of random numbers sampled from a uniform distribution are doing quite well in reproducing a Gaussian distribution.

Counting conformations on a lattice

Here we discuss a simple model for protein folding mostly associated with the name of Ken Dill, which is the so called HP model on a two dimensional lattices. The HP model considers only two types of amino acids Hydrophobic (H) and Polar (P). The energy of the systems is a sum of <u>contact</u> energies. Two amino acids are considered to be in a contact if their (Cartesian norm 2) distance is lower than a critical value *a* (which is also the lattice constant) and at the same time they are separated along the chain by more than two amino acid. Energies of contacts are assigned according to the identity of the two amino acids (H-H contact scores –1, and all other contact types score 0).



An amino acid is situated at a cross point and the contacts for the above "protein" chain are indicated by a dotted line. There are four contacts for the chain above and the lowest possible energy is therefore –4. Note also that the chain is not allowed to cross itself and the lattice is considered to be infinite (i.e. there are no wall effects in which the chain may not penetrate in).

The advantage of using the HP model and the two dimensional lattice is the enormous simplicity of the protein folding problem in that space. For the two dimensional problem above exhaustive counting of all conformations is possible for two-dimensional chains that are not too long. It is also possible to use symmetry to reduce significantly the number of conformations that require counting. The first pair of amino acids can have a fixed orientation, since alternative positioning can be related by symmetry (four rotations). Moreover conformations resulting from the second bond going by 90 degrees to the left are symmetry related to the second bond going to the right (the "straight" bond is unique). This effectively reduces the number of amino acids that we need to exhaustively consider to N-3 where N is the total number of amino acids. It is clear that an absolutely highest bound on the number of conformations is $2 \cdot 3^{N-3}$. We will consider chain of length of 14 amino acids with a number of configurations significantly less than a million. We will therefore count them directly.

How to represent a chain on the lattice and how to compute its energy? The simplest representation stores the Cartesian coordinates of the chain for example $(x_1,y_1)(x_2,y_2)...(x_N,y_N)$. This representation is however inconvenient for conformational sampling. It does not contain the covalent bonding between sequential amino acid. As a result it is difficult to ensure that a representation in the Cartesian coordinates will result in correctly bound polymer (without significant additional calculations). To ensure correct covalent bonding it is useful to consider bond representation. A single bond vector connecting the centers represents sequential amino acids (say i and $i+1$). In two dimensions it is convenient to represent these vectors by complex (purely imaginary or purely real) numbers. For example, the polymer drawn above is represented by the set of vectors $(1)(i)(1)(-i)(-i)(1)(i)(i)(1)(-i)$. (i) is the square root of -1. In the above set it is easy to generate plausible conformations, starting from an acceptable structure. Consider for example a 90 degree rotation around the fifth bonds (the allowed moves are ± 90 degree rotations around bonds). Multiplying the bonds 6 and up by i we the proposed alternative chain conformation $(1)(i)(1)(-i)(-i)(i)(-1)(-1)(i)(1)$

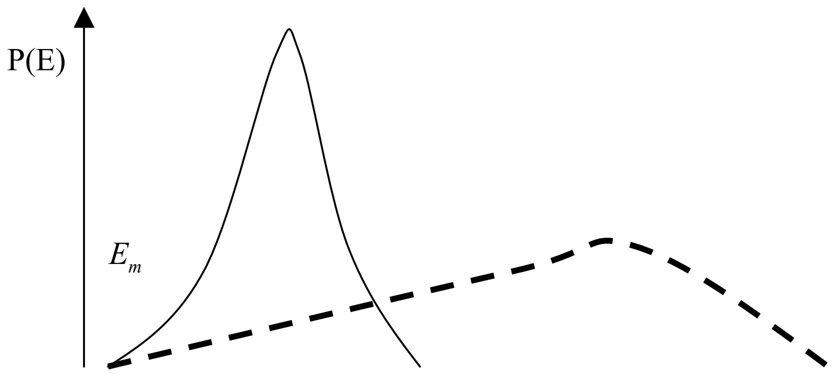
Note that after this operation we do not know if the chain so created has amino acids overlapping the same site, nor the number of contacts between different amino acids. To assess the contacts and overlaps we need to generate a Cartesian representation from the above chain representation.

To measure the stability of a conformation we will consider the Z score. It is a normalized energy measure. If E_m is the lowest energy conformation, the stability of a protein structure is estimated by

$$Z = \frac{\langle E \rangle - E_m}{\sigma} \qquad \sigma = \sqrt{\langle E^2 \rangle - \langle E \rangle^2}$$

The idea behind this measure is that it is not enough just to have the energy low but we also need to make sure that there are not many other structures close by that will make the energy difference insignificant (if considered for example within the Boltzmann weight -- $\exp\left[-\frac{E_m - \langle E \rangle}{kT}\right]$ that is very close to one.

Pictorially we can have something like



From the perspective of protein stability the dashed –line distr [illegible] better. For a given (fixed) number of total structures the minimum energy [illegible] of the dashed-line distribution feels significantly less competition wi [illegible] ative structure. Note also that the Z score measure is approximate. I [illegible] ut not very likely) that a highly asymmetric distribution of energies is obtained that makes the estimate of competing structures, which is based on the second moment only, inaccurate.

We can ask now what kind of sequence HHPP…. is giving the most stable structure. Specifically, for a fixed number of H and P residues (only permutation of the positions of the H and the P are allowed) what is the sequence that will lead to the most stable structure (regardless what this structure might be). For the simple model of H/P this is not very difficult. We start by considering the structure. The energy is related to the number of contacts and we therefore seek a structure with a maximal number of contacts for a given sequence length. For a structure with a maximal number of contacts we attempt to "fill" all the contacts with H rather than with P.

For a fixed structure, the situation is a little more tricky, since the only way we can make the given structure better than other structure (with more or the same number of contacts) is by using the order of the amino acids to make the energy of competing structures worse by making sure that contacts are populated mostly with P-s. This problem of finding a sequence that makes a structure most stable is called the protein design problem.

Obviously the protein-folding problem is the following: given a sequence can we determine the most stable structure. Usually the sequence provided is for a real protein and we can expect that the lowest energy structure is indeed the most stable (if our energy function indeed reflects experiment). However, for a general random sequence (that still has a lowest energy minimum) we expect that the lowest energy minimum is not significant as measured by the above Z score. In other words the special property of native proteins: folding to a unique 3D structures is not something we see for any sequence of amino acids but only for a small selected set that is at least partially optimized for a maximal number of contacts. The native sequence is not fully optimized for stability in true native proteins (in contrast to our two-dimensional models) since other consideration, such as the needs to design active sites and to maintain flexibility influence sequence selection.

Dynamics with constraints

So far we were after folding of a protein on a lattice. Here we are expanding to real space and consider a model of protein folding in real space. There are still going to be only two types of amino acids H and P. A protein chain is presented by a set of N three-dimensional vectors, where N is the number of amino acids and each amino acid is presented by a single Cartesian triplet (x,y,z). Instead of the contact model of the lattice we shall use LJ potential to describe interactions between pairs of amino acids.

$$U_{HH} = 4\left[\frac{1}{r^{12}} - \frac{1}{r^{6}}\right] \qquad U_{HP} = 0.001\left[\frac{1}{r^{12}} - \frac{1}{r^{6}}\right] \qquad U_{PP} = 0.001\left[\frac{1}{r^{12}} - \frac{1}{r^{6}}\right]$$

Along the protein chain only amino acids separated by at least two bonds are considered for the above LJ interactions. The way we will deal with "bonds" between amino acids is by using Lagrange's multipliers.

Lagrange multipliers

Consider an optimization problem with a constraint. For example, we wish to minimize a function $f(x,y,z) = x^2 + y^2 + z^2$ with respect to the variables x, y, z. The global minimum is clearly when $(x = 0, y = 0, z = 0)$.

However we now add a constraint $\sigma = x + y + z - 1 = 0$. Hence our solution should be a minimum of f and at the same time satisfies the condition σ. In principle there can be more than one function that we wish to minimize and more than one constraint. However, the discussion here is limited to one function, but potentially more than one constraint. Note that each of the constraints removes one variable. For example the above constraint established a relation between $x, y,$ and z, and there are no more independent. This suggests the first approach we can use to solve the constrained problem.

How can we solve this problem? One approach is to use the constraint(s) to solve for one of the variables. For example, we can use $z = 1 - x - y$ and minimize the new function $f(x,y,z) \equiv \bar{f}(x,y) = x^2 + y^2 + (1 - x - y)^2$. The minimization in this case is trivial and can be done analytically (no need for MATLAB or numerical analysis). We have

$$\frac{d\bar{f}}{dx} = \frac{d\bar{f}}{dy} = 0 \qquad \frac{d\bar{f}}{dx} = 2x - 2(1 - x - y) = 0 \qquad \frac{d\bar{f}}{dy} = 2y - 2(1 - x - y) = 0$$

$$\Rightarrow x = 1/3 \quad y = 1/3 \quad z = 1/3$$

The difficulty with the above (which is nevertheless an exact procedure) is that it is not always easy to solve explicitly for one of the variables. For example consider the constraint: $\sigma = \exp\left[x^2 + y^{\frac{1}{5}} + 1/z\right] + y^2 + z^7 - x = 0$, can you express z in terms of x and y? Moreover, the problem is treated in a very asymmetric way, who said that we should substitute z and not y or x? This may not seems important but in large systems such choices can lead to numerical instabilities.

The Lagrange multipliers approach makes it possible to do the constrained optimization in a symmetric way almost like an optimization without constraints. We consider a new function $g(x,y,z,\lambda)$ with one more variable, λ compared to three variables of $f(x,y,z)$. The new function is given by

$$g(x,y,z,\lambda) = f(x,y,z) + \lambda\sigma(x,y,z)$$

If we now wish to determine a stationary point of $g(x,y,z,\lambda)$, first derivatives are useful

$$\frac{dg}{dx} = \frac{df}{dx} + \lambda\frac{d\sigma}{dx} = 0$$
$$\frac{dg}{dy} = \frac{df}{dy} + \lambda\frac{d\sigma}{dy} = 0$$
$$\frac{dg}{dz} = \frac{df}{dz} + \lambda\frac{d\sigma}{dz} = 0$$
$$\frac{dg}{d\lambda} = \sigma = 0$$

Let us look at the concrete example we had above and solve it in the framework of Lagrange's multipliers

$$g(x,y,z,\lambda) = x^2 + y^2 + z^2 + \lambda(x+y+z-1)$$
$$\frac{dg}{dx} = 2x + \lambda = 0$$
$$\frac{dg}{dy} = 2y + \lambda = 0$$
$$\frac{dg}{dz} = 2z + \lambda = 0$$
$$\frac{dg}{d\lambda} = (x+y+z-1) = 0$$

The four equations with four unknown can be solved by eliminating λ from the first two equations –

$$\frac{dg}{dx} - \frac{dg}{dy} = 2x - 2y = 0 \Rightarrow x = y$$
$$\frac{dg}{dx} - \frac{dg}{dz} = 2x - 2z = 0 \Rightarrow x = z$$

and now substituting in the last equation to have $3x - 1 = 0 \quad x = 1/3$. We therefore also have $x = 1/3 \;\; y = 1/3 \;\; z = 1/3 \;\; \lambda = -2/3$. It is not really necessary here to determine the Lagrange multiplier since it is only a tool to solve the constrained optimization problem. However it is interesting to find out what it does. The Lagrange multiplier is multiplying a term which is the constraint derivative and balances the gradient of the original function. This is why the derivatives of the constraints are sometimes called "constraint force".

Lagrange's multipliers are extremely useful in many branches of mechanics. For example, in simulation of robotic motions, one usually assumes that the length of the robot's hand is fixed and only the orientation is changing when we optimize the structure of the robot. The constant length of the robot hand can be modeled with a constraint and the use of Lagrange's multipliers.

Another place where Lagrange's multipliers are useful is in simulation of protein folding in which we wish to keep the bond length fixed. This is our next topic.

Brownian dynamics with bond constraints

Consider the differential equation for Brownian dynamics

$$\gamma \frac{dX}{dt} = -\nabla U + R$$

where X is the coordinate vector for all the amino acids in the system U the potential energy, and R a vector of the random forces operating on all degrees of the freedom in the system. We consider a protein model in which a single amino avid is represented by a point and all sequential amino acids are connected by a bond. The bonds are constrained to an equilibrium position. The constraints are added to the equations of motion are constraints' force we have

$$\gamma \frac{dX}{dt} = -\nabla U + R + \sum_l \lambda_l \sigma_l$$
$$\sigma_l = \left(d_{i,i+1}^2 - d_0^2\right) = 0 \qquad \forall l$$

where the index l is running over the constraints (the number of bonds in our protein model. For example for a 14-mers we have 13 constraints. The λ_l in the equation of the first line are the undetermined Lagrange's multipliers (for the moment) that we need to compute with the help of the constraints' equation in the second line.

Consider a finite difference version of the above equation

$$X_{i+1} = X_i + \frac{\Delta t}{\gamma}\left[-\nabla U\left(X_i\right) + R_i + \sum_l \lambda_{li} \sigma_l\left(X_i\right)\right] = X_{i+1}^{(0)} + \frac{\Delta t}{\gamma}\left[\sum_l \lambda_{li} \sigma_l\left(X_i\right)\right]$$

where the index i here is the index of time. We also define $X_{i+1}^{(0)}$ which is the coordinate of step $i+1$ in the absence of the constraints.
The above equation is ill determined. While it suggests solving the equations of motion as a function of time in steps, we are still left with unknowns, which are the value of all the Lagrange's multipliers λ_l. These unknowns must be determined with the help of the constraints equation. In the finite difference picture we insist that the constraints of the new step will be satisfied exactly.

$$\sigma_l\left(X_{i+1}\right)=0\simeq\sigma_l\left(X_i^{(0)}\right)+\frac{\Delta t}{\gamma}\nabla\sigma_l\left(X_{i+1}^{(0)}\right)\sum_{l'}\lambda_{l'}\sigma_{l'}\left(X_i\right)$$

These are linear equations (the number of equations is equal to the number of constraints) that can be used to determine λ_l. Note that the scalar product below determined the matrix elements of a new matrix A -- $\frac{\Delta t}{\gamma}\nabla\sigma_l\left(X_i^{(0)}\right)\cdot\nabla\sigma_l\left(X_i\right)\equiv A_{ll'}^{(0)}$. We have

$$A^{(0)}\lambda^{(0)}=-\sigma\left(X_{i+1}^{(0)}\right)$$

Note that the solution of the linear equation is not the end of the story since the $\sigma_l\left(X_{i+1}\right)$ was only approximated by the linear expression. We simply redefine out current guess instead of $X_i^{(1)}=X_i^{(0)}+\sum_l\lambda_l^{(0)}\nabla\sigma_l\left(X_i\right)$ and reiterate the linearization of the constraints' equation attempting to satisfy the constraint of the next step (this is similar to Newton Raphson procedure of a sequence of linear approximations to a function). We write

$$\sigma_l\left(X_{i+1}\right)=0\simeq\sigma_l\left(X_{i+1}^{(1)}\right)+\frac{\Delta t}{\gamma}\nabla\sigma_l\left(X_{i+1}^{(1)}\right)\sum_{l'}\lambda_{l'}^{(1)}\sigma_{l'}\left(X_i\right)$$

For the n iteration we have

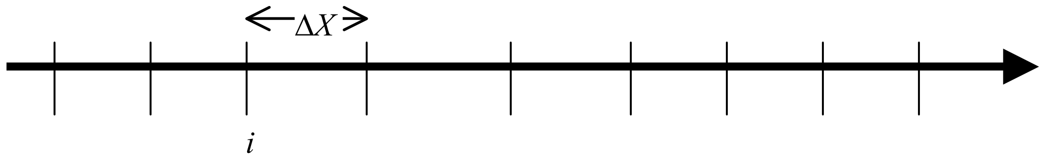
$$\sigma_l\left(X_{i+1}\right)=0\simeq\sigma_l\left(X_{i+1}^{(n)}\right)+\frac{\Delta t}{\gamma}\nabla\sigma_l\left(X_{i+1}^{(n)}\right)\sum_{l'}\lambda_{l'}^{(n)}\sigma_{l'}\left(X_i\right)$$

The iterations stop (convergence is assumed) when $\left|\sigma_l\left(X_{i+1}^{(n)}\right)\right|\le\varepsilon$ where ε is a small positive number, such small value implied that the Lagrange's multipliers will be set to zero as well.

Thinking about and solving partial differential equations

It is useful to start thinking on partial differential equation in terms of finite difference and discrete formulas. Consider a one-dimensional problem with a discrete set of states ordered along the X axis. We labeled the states by i. The probability of being at the discrete state i is p_i, the distance between states $i\pm1$ and i is $\pm\Delta X$. We consider the evolution of the probability as a function of time. We have

$$p_i\left(t+\Delta t\right)-p_i\left(t\right)=-k_{i,i\pm1}p_i+k_{i+1,i}p_{i+1}+k_{i-1,i}p_{i-1}$$



For a free diffusion (the problem that we shall consider first) we expect the rate constants to be similar for all sites and all directions. We therefore re-write the above equation as

$$p_i\left(t+\Delta t\right)-p_i\left(t\right)=-2kp_i+kp_{i+1}+kp_{i-1}$$

(the loss of probability from site i is in both directions – to $i+1$ and to $i-1$, therefore we have the factor of 2). We define now the diffusion constant as

$$D = k\left(\Delta X\right)^2$$

and have a finite difference equation quite close by now to the diffusion equation

$$p_i\left(t+\Delta t\right) - p_i\left(t\right) = \frac{D}{\left(\Delta X\right)^2}\left(-2kp_i + kp_{i+1} + kp_{i-1}\right)$$

Taking the limits of Δt and ΔX going to zero we have an equation for the probability density

$$\frac{\partial p\left(X,t\right)}{\partial t} = D\frac{\partial^2 p\left(X,t\right)}{\partial X^2}$$

The probability density is normalized to one $\int p\left(X,t\right)dX = 1$ and is always positive. We also consider the range of the free diffusion in the interval $\left[-\infty,\infty\right]$, which means that we can ignore the boundary conditions. The analytical technique of solving this type of equation is by Fourier (and Laplace) transforms. We perform a Fourier transform over space to obtain an algebraic equation in the Fourier variable. We define:

$$\tilde{p}\left(k,t\right) = \int \exp\left(-ik\cdot X\right)p\left(X,t\right)dX$$

and have

$$\frac{\partial \tilde{p}\left(k,t\right)}{\partial t} = -k^2 D\tilde{p}\left(k,t\right)$$

For which we can write immediately

$$\tilde{p}\left(k,t\right) = \exp\left[-Dk^2t\right]\tilde{p}\left(k,t=0\right)$$

For the initial condition $p\left(X,t=0\right) = \delta\left(X\right)$, we obtain the solution

$$p\left(X,t\right) = \left(4\pi Dt\right)^{1/2}\exp\left(-\frac{X^2}{Dt}\right)$$

Going back to numeric we note that we can write the finite difference equation above as a matrix equation. Consider a vector P with elements p_i, we can write a matrix equation for P

$$\frac{dP}{dt} = AP \qquad A = \begin{pmatrix} -2k & k & 0 \\ k & -2k & k \\ 0 & k & \dots \end{pmatrix}$$

Note that at least for our simple problem the matrix A is symmetric making the analysis considerable simpler and straightforward. Let U be the matrix that diagonalizes A, that is $\Gamma = U^t AU$ is a diagonal matrix. We can multiply the matrix equation from the left by U^t and insert between A and P the identity matrix $I = U^t U$ to have

$$\frac{d\left(U^t P\right)}{dt}=\left(U^t A U\right)\left(U^t P\right)$$
$$\frac{dQ}{dt}=\Gamma Q$$
$$Q(t)=Q(t=0)\exp[\Gamma t]$$

Where we define $Q(t)\equiv U^t P$ is the representation of the probability density in the space of the eigenvectors of A. The set of eigenvalues are $\{\gamma_i\}$ making it possible to write the decay rate for every element of Q -- q_i its own relaxation rate
$$q_i(t)=q_i(0)\exp[\gamma_i t]$$

Note that the most interesting eigenvalues are the largest ones (least negative). The zero eigenvalue represents a time independent non-decaying component of equilibrium. The next least negative eigenvalue is the slowest relaxation rate in the system that in many cases corresponds to experimental measurements, done on long time scales in which the faster relaxation rates already disappeared. In general initial diffusion population will include more than a single relaxation rate. To describe the evolution in time of the probability density we need the relaxation times (the eigenvalues of the matrix A --- γ_i) and the projection to different relaxation vectors (the eigenvectors of the matrix A stored in the matrix U)

In our concrete example above the discrete equation can be solved exactly with a similar Fourier transform tricks. The finite difference equation is:

$$-2kP_j + kP_{j+1} + kP_{j+1} = \frac{dP_j}{dt}$$
Fourier transforming with respect to time, we have:
$$-2k\tilde{P}_j + k\tilde{P}_{j+1} + k\tilde{P}_{j-1} = i\omega\tilde{P}_j$$
The function $\tilde{P}$ is the continuous Fourier transform (over time) of P. Guessing a solution of the type
$$\tilde{P}_j = P_0\exp(i\Delta X)$$
we have

$$k\tilde{P}_0\left[-2+\exp[i\Delta X]+\exp[-i\Delta X]\right]=i\omega\tilde{P}_0$$
$$\left[\frac{i\omega}{k}+2-2\cos(\Delta X)\right]\tilde{P}_0=0$$
The last equation offers a condition on the frequencies and the rate coefficient given that the eigenvalues of the matrix A are $2\left(1-\cos(\Delta X)\right)$. Note also that the eigenvector themselves are oscillatory.

Consider now a step up in complexity. We consider diffusion in the presence of a biasing force or potential. The diffusion equation is modified to

$$\frac{\partial P(X,t)}{\partial t} = \frac{1}{M\gamma}\frac{\partial}{\partial X}\left[\frac{dU}{dX}P(X,t)\right] + D\frac{\partial^2 P(X,t)}{\partial X^2}$$

The diffusion constant D is related to the friction constant by the formula $D = \frac{kT}{M\gamma}$. The extra term presents "drift" which is a force in a specific direction that operated on the density and is determined with the help of the potential gradient. The equation is still linear and can be written in a matrix form (slightly more complex though). To have a better feeling of what the extra term means we consider equilibrium limit. In equilibrium there are no observed changes in the distribution as a function of time, and the density is a function of the coordinates only, we have

$$0 = \frac{1}{M\gamma}\frac{\partial}{\partial X}\left[\frac{dU}{dX}P_{eq}(X)\right] + D\frac{\partial^2 P_{eq}(X)}{\partial X^2}$$

It is no so easy to solve the above partial differential equation cold but we can "guess" a solution, which (not surprisingly) is

$$P_{eq}(X) = A\exp\left(-\frac{U(X)}{kT}\right)$$

This can be verified by substituting the guess into the above equation.

$$0 = \frac{1}{M\gamma}\frac{\partial}{\partial X}\left[\frac{dU}{dX}\cdot A\exp\left[-\frac{U(X)}{kT}\right]\right] + D\frac{\partial^2 A\exp\left[-\frac{U(X)}{kT}\right]}{\partial X^2}$$

$$0 = \frac{1}{M\gamma}\frac{\partial}{\partial X}\left[\frac{dU}{dX}\cdot A\exp\left[-\frac{U(X)}{kT}\right]\right] - \frac{D}{kT}\frac{\partial}{\partial X}\left[\frac{\partial U}{\partial X}\cdot A\exp\left[-\frac{U(X)}{kT}\right]\right]$$

which is zero since (definition above) $D = \frac{kT}{M\gamma}$.

Note that in general the problem is still linear in the probability density and therefore can be discretized and solve in a matrix form exactly like in the free diffusion case. An important difference is that the resulting matrix A is no longer symmetric ($A_{ij} \neq A_{ji}$) which makes out life a little more interesting.