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 measure (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 is chosen empirically to be 600 kcal/molÅ^2.

The potential between n water molecules using the TIP3P model is

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

the indices i, j are for water molecules, the indices k, l are for the atoms of a single molecule. 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

$A = 629,400$ kcal Å^{12}/mol

$B = 625.5$ kcal Å^{6}/mol

$c_0 = -0.834$

$c_H = 0.417$

$k = 332.1$

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

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

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.

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 component 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 and 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 $3N$ $q_i - s$. It is useful to know that more general formulation would be $\Delta^{(n)} = \left[\sum_i (q_i - q_{i0})^n \right]^{\frac{1}{n}}$. 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 $n = 0$ 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).

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

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

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

Similarly we have for δR

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

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

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

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 have 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 $U(R_0 + \delta R)$ 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 large step could be taken since we anticipate significant change in the function (to the better). Of course the step still cannot be too large since our arguments are based on 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 $(|\nabla U \le \varepsilon|)$

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 as 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 $U(R(\tau))$ 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 assumptions) 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 is to make the step size 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 $\nabla U(R + \Delta \cdot \nabla U) = 0$ 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 $U\left(R + \Delta \cdot \nabla U\right)$ can be used and 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 lot 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 coordinate to perfection using something like 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 dimension. It is unheard of having effective minimizers in high dimensions without derivatives:

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.

$$r_{ij} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_i - z_j\right)^2}$$

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

It is therefore 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 root are the curse of simulations and are much more expensive to compute compare to add/multiply etc. Unfortunately, electrostatic interactions 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 forgot 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?

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

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

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 $\left|R_m - R\right|$ 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 $\tilde{U}$ has 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

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

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.

$$\hat{e}_{rx} = \left(0, m_1 \cdot z_1^0, -m_1 \cdot y_1^0, 0, m_2 \cdot z_2^0, -m_2 \cdot y_2^0, ..., 0, m_N \cdot z_N^0, -m_N \cdot y_N^0\right)$$

$$\hat{e}_{ry} = \left(m_1 \cdot z_1^0, 0, -m_1 \cdot x_1^0, m_2 \cdot z_2^0, 0, -m_2 \cdot x_2^0, ..., m_N \cdot z_N^0, 0, -m_N \cdot x_N^0\right)$$

$$\hat{e}_{rz} = \left(m_1 \cdot y_1^0, -m_1 \cdot x_1^0, 0, m_2 \cdot y_2^0, -m_2 \cdot x_2^0, 0, ..., m_N \cdot y_N^0, -m_N \cdot x_N^0, 0\right)$$

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.

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 peak one of them at random (call it for convenience $\hat{e}_1$) and normalize it

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

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

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

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. 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" $x_j \quad j = 1, ..., i-1$)

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

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.