Global optimization on the potential energy surface

Stefan.Goedecker@unibas.ch

- Lecture notes: http:/comphys.unibas.ch/teaching.htm

Basic computer skills

- Linux operating system
- some editor; recommended: vi
- Fortran programming
- Some visualization software; recommended: V_sim : http://www-drfmc.cea.fr/sp2m/L_Sim/V_Sim/index.en.html

The total energy and its significance

The so-called total energy E is a function that gives the energy of an atomistic system in terms of its atomic positions. Denoting the atomic positions of the system being composed of N_{at} atoms by $\mathbf{R}_i$ where $i = 1, \ldots, N_{at}$ our function has the following form:

$$E = E(\mathbf{R}_1, \ldots, \mathbf{R}_{N_{at}})$$

The position of the j-th atom $\mathbf{R}_j$ is a vector of length 3 with thethree components (X_j, Y_j, Z_j). Since the electrons are responsible for the interactions of atoms in condensed matter systems one has to solve in principle the electronic Schrödinger equation to obtain E. The solution of the many-electron Schroedinger equation gives several sets of corresponding eigenvalues E_i and eigenvectors Ψ_i.

$$H\Psi_i = E_i\Psi_i$$

The eigenvalues have the significance of energies and the eigenvector are the wavefunctions. The set with the lowest energy E_0 corresponds to the electronic ground state and the other sets to excited electronic states. The energies E_i of each set depend on the positions of the atoms since the Hamiltonian is parametrized by the atomic positions:

$$\mathcal{H}(\mathbf{R}_1,...,\mathbf{R}_{N_{at}},\mathbf{r}_1,...,\mathbf{r}_N) = \tag{1}$$
$$\sum_{i=1}^{N} -\frac{1}{2}\nabla^2_{\mathbf{r}_i} + \sum_{i=1}^{N}\sum_{j=1}^{i-1}\frac{1}{|\mathbf{r}_i-\mathbf{r}_j|} - \sum_{i=1}^{N}\sum_{j=1}^{N_{at}}\frac{Z_j}{|\mathbf{r}_i-\mathbf{R}_j|} + \sum_{i=1}^{N_{at}}\sum_{j=1}^{i-1}\frac{Z_iZ_j}{|\mathbf{R}_i-\mathbf{R}_j|}$$

where $\nabla^2_{\mathbf{r}_i} = \frac{\partial^2}{\partial x_i^2} + \frac{\partial^2}{\partial y_i^2} + \frac{\partial^2}{\partial z_i^2}$. The various functions $E_i(\mathbf{R}_1,...,\mathbf{R}_{N_{at}})i$ are called Born-Oppenheimer surfaces. The Born-Oppenheimer surfaces of excited electronic states are only rarely used, for instance in photochemical reactions and we will therefore in the following only consider the ground state Born-Oppenheimer surfaces E_0 which is also called Potential Energy Surface (PES). For convenience we will also drop the subscript and simply put $E = E_0$.

The exact solution of the many electron Schrödinger equation is numerically extremely expensive and can only be done for small systems. Even approximations to the exact Schrödinger equation such as density functional theory are numerically quite expensive. For this reason we will use in this course the simplest model for the PES, namely the Lenaard Jones (LJ) potential. The interaction between two LJ atoms is given by

$$E = V(r) = 4\varepsilon\left((\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6}\right) \tag{2}$$

ε is the depth of the well and σ the distance at which the potential is zero. The

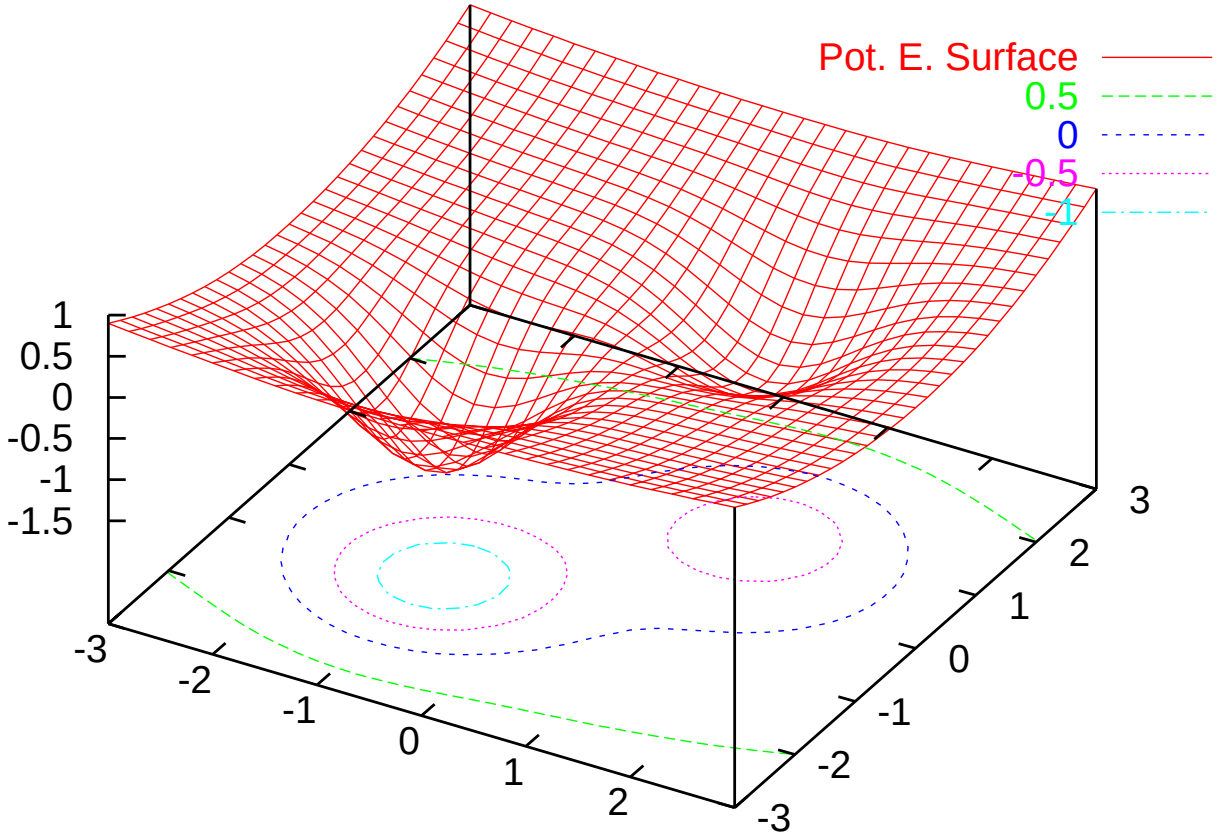
interaction between N_{at} LJ atoms is given by

$$E = \sum_{i<j\leq N_{at}} V(|\mathbf{R}_i - \mathbf{R}_j|) \tag{3}$$

The Lennard Jones potential (with appropriate values ε and σ) describes well the interaction of noble gas atoms, but also some metallic clusters can approximately be modelled.

Landmark points of the potential energy surface: Minima and saddle points

A toy PES: 2 local minima, 1 saddle point



- global minimum: ground state
- other local minima: metastable states
- one saddle point: transition state

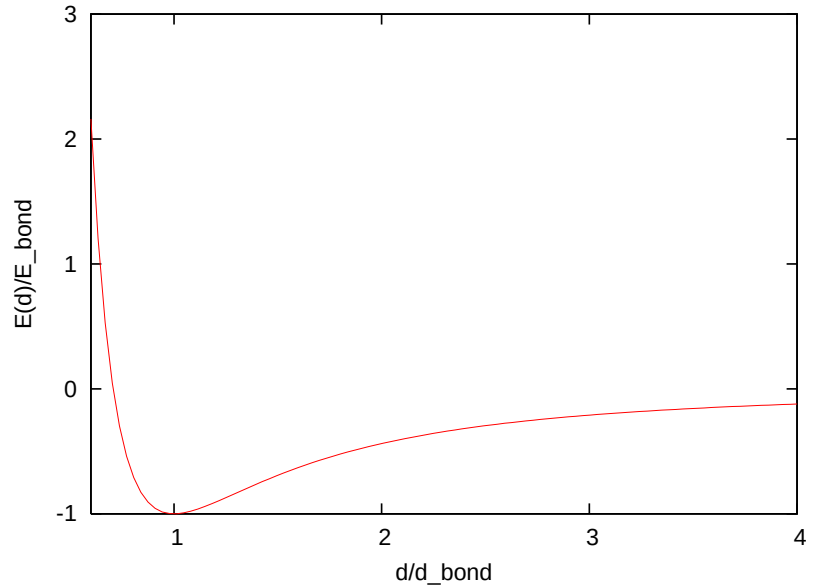
Significance of Minima

The knowledge of the total energy function allows us to determine the equilibrium atomic positions. We can calculate the partial derivatives $\frac{\partial E}{\partial \mathbf{R}_j}$. This is a vector of length 3 with the components $\left(\frac{\partial E}{\partial X_j}, \frac{\partial E}{\partial Y_j}, \frac{\partial E}{\partial Z_j}\right)$. The force acting on atom j, $\mathbf{f_j}$, is the negative of this partial derivative.

$$\mathbf{f_j} = -\frac{\partial E}{\partial \mathbf{R}_j} \tag{4}$$

A stable atomic configuration is by definition a configuration where the forces on the atoms vanishes, i.e where all the partial derivatives of the total energy with respect to the atomic positions are zero. A equilibrium configuration (or equilibrium geometry) of a molecule corresponds therefore to a stationary point of the total energy function. Since the configuration has also to conserve its stability under small displacements away form the staionary point, the stationary point has to be a local minimum of the total energy function. This means that small displacements away from the stationary point will always lead to an increase in energy. From a mathematical point of view a local minimum is characterized by the fact the the curvature along any line going through the local minimum is positive or, equivalently, that the Hessian matrix is positive definite.

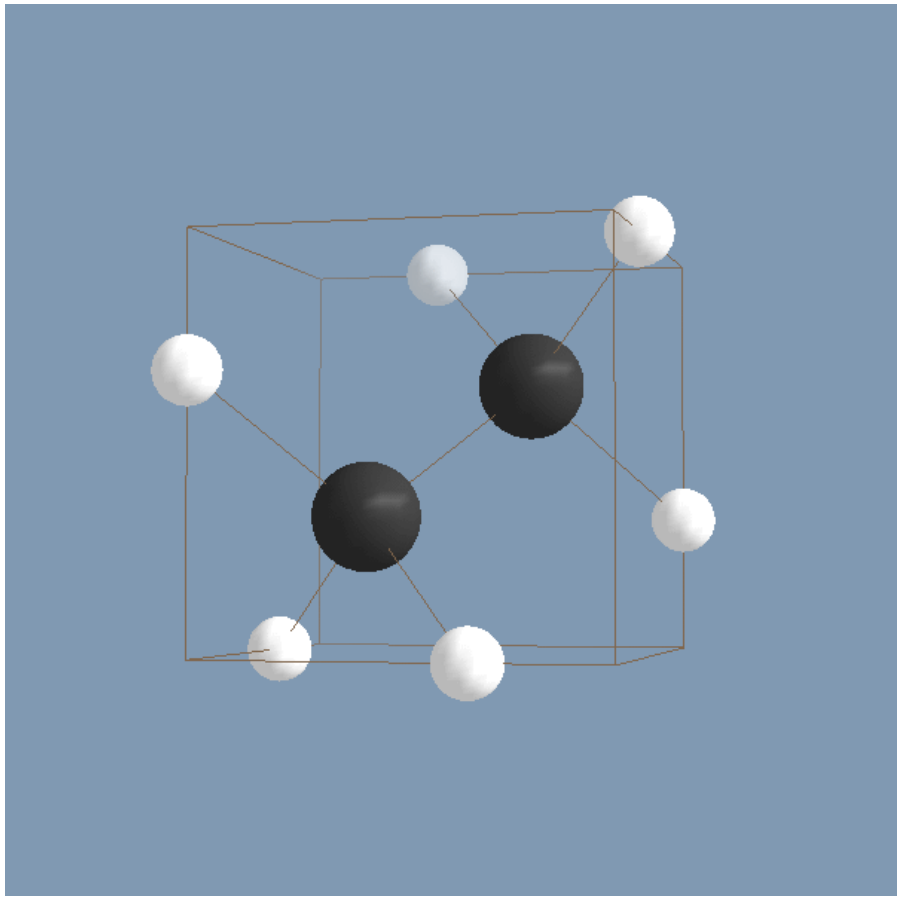
Let us illustrate the concept by the total energy function of the simplest molecule, namely the H_2 molecule. Even though the total energy function is formally a function of 6 variables, it is clear from physical considerations that the energy depends only on the distance between the 2 hydrogen atoms. Hence the total energy can be reduced from a 6 dimensional to a one dimensional function. The function is sketched below.

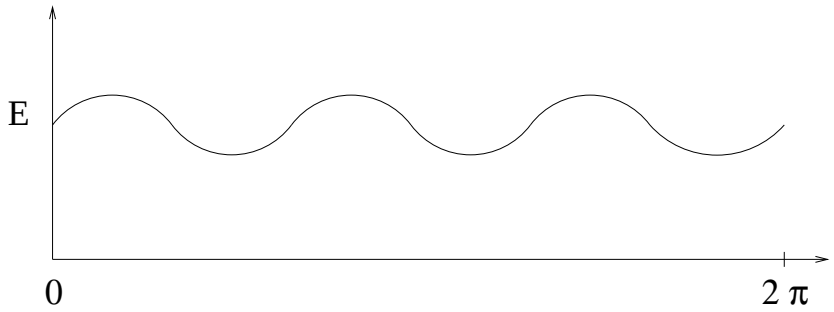


The minimum of the above curve gives obviously the bond length of the hydrogen molecule and the depth of the curve its binding energy.
Configurations of more complicated molecules can in principle be obtained in the same way, namely by finding local minima of the total energy. Numerical methods that allow us to find local minima of high dimensional functions will be discussed later.

Realistic PESs have an important difference from the simple ones discussed up to now. The number of minima is in general huge. As a matter of fact, the number of local minima increases exponentially with respect to the number of atoms om the system. This can easily be seen for some simple model system such as alkanes or bulk silicon.

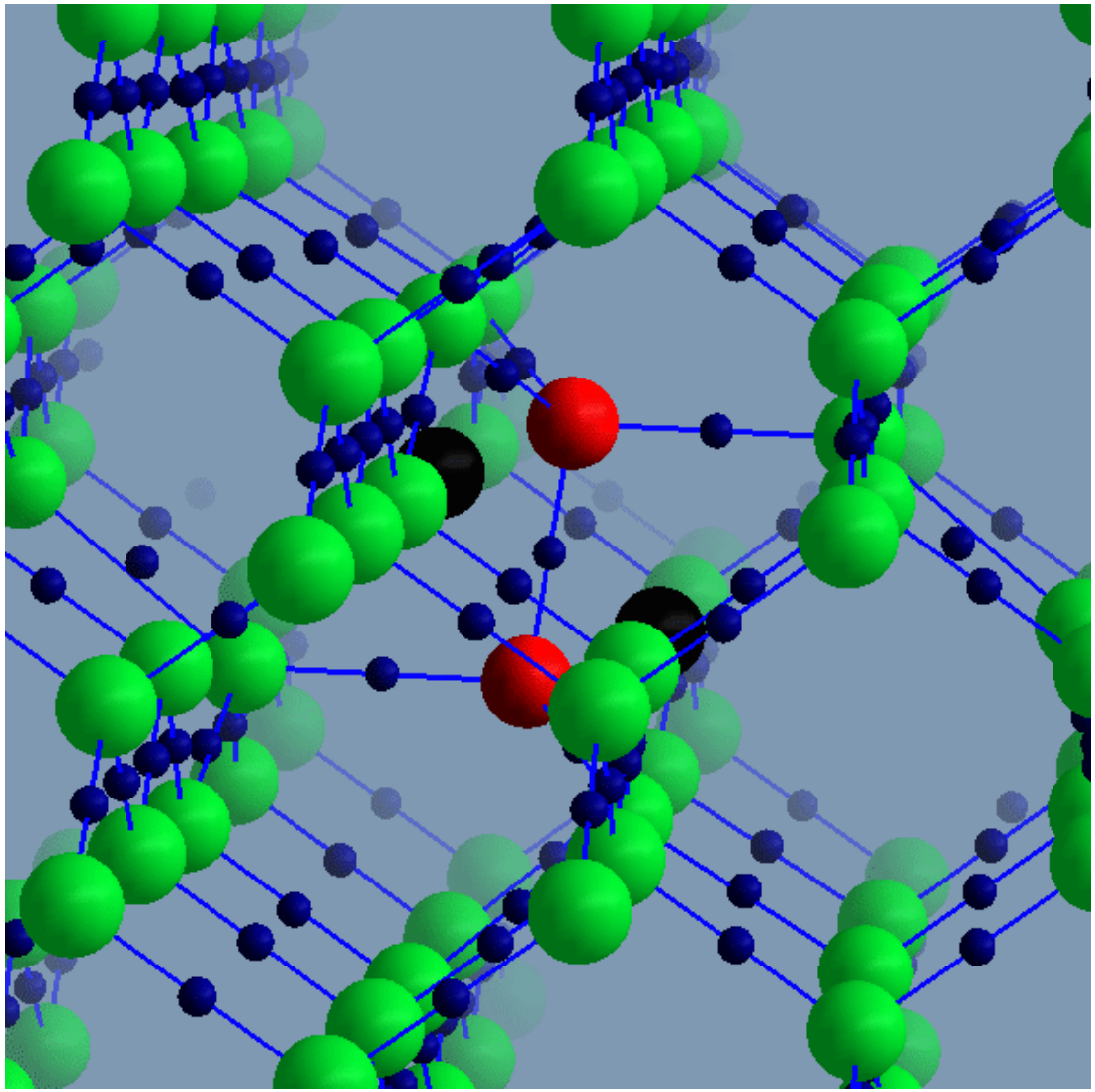
First example: alkane family, C_nH_{2n+2}: $O(3^n)$ local minima





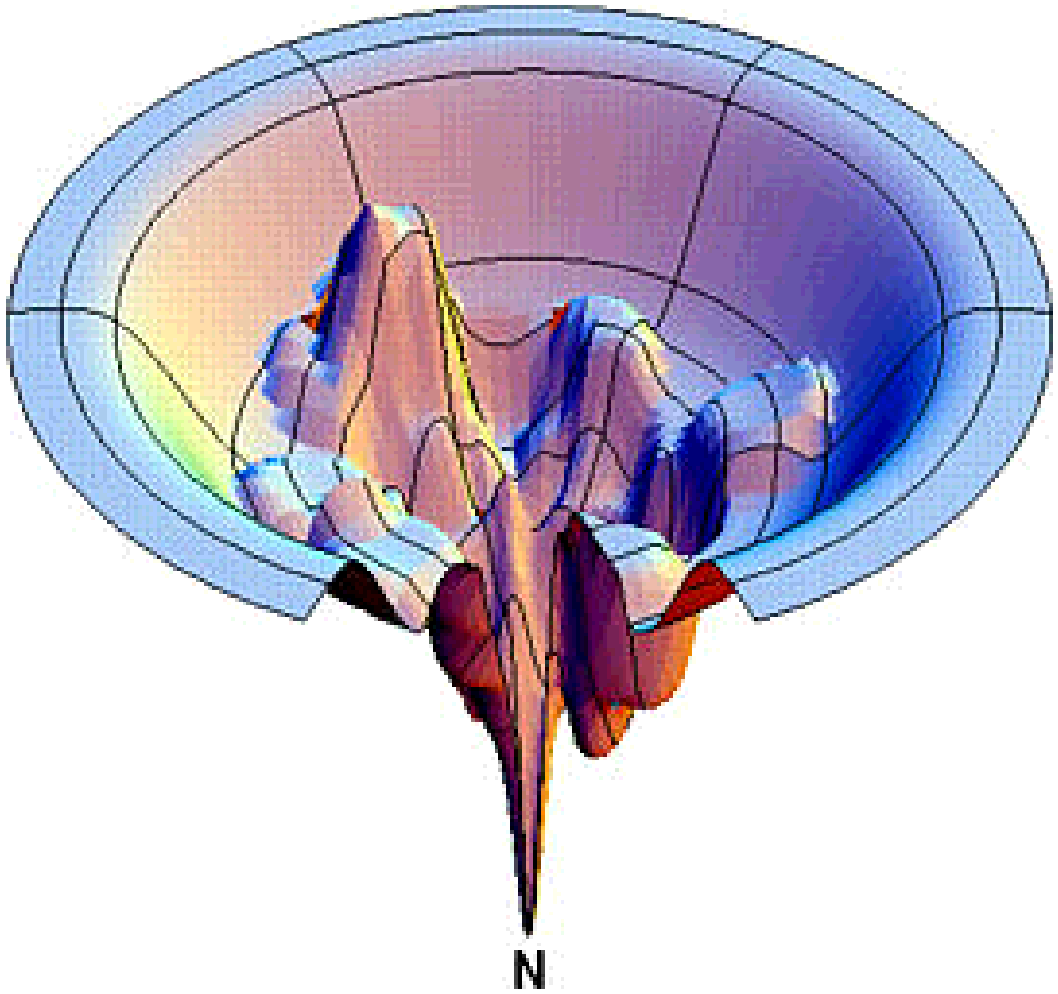
Second example: bulk silicon: $O(3^n)$ local minima

Wooten-Winer-Weaire process:



Visualizing the potential energy surface

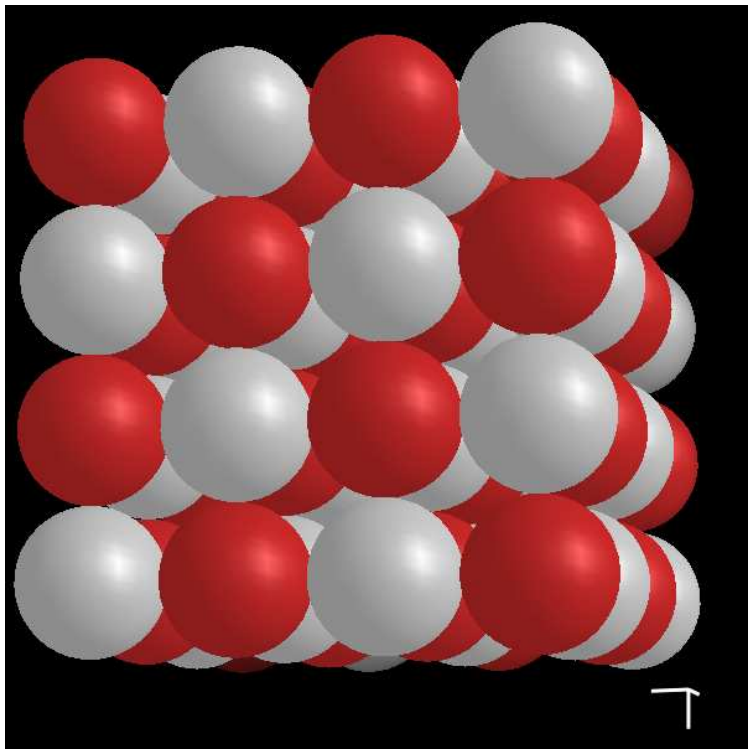
Visualizing the entire PES is in general impossible due to its high dimensionality. Below is an heroic effort for the highest possible dimension!

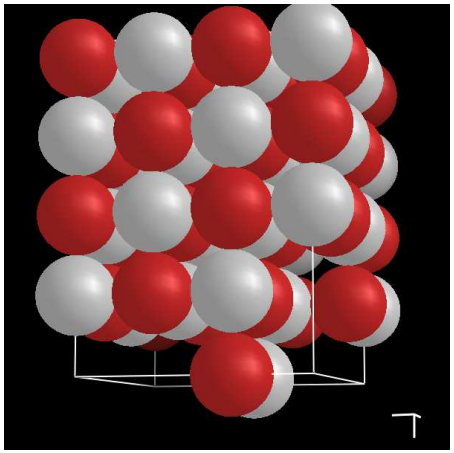


from: S. Chan and K. Dill, Proteins: Structure, Function, and Genetics, **30**, *2 (1998)*

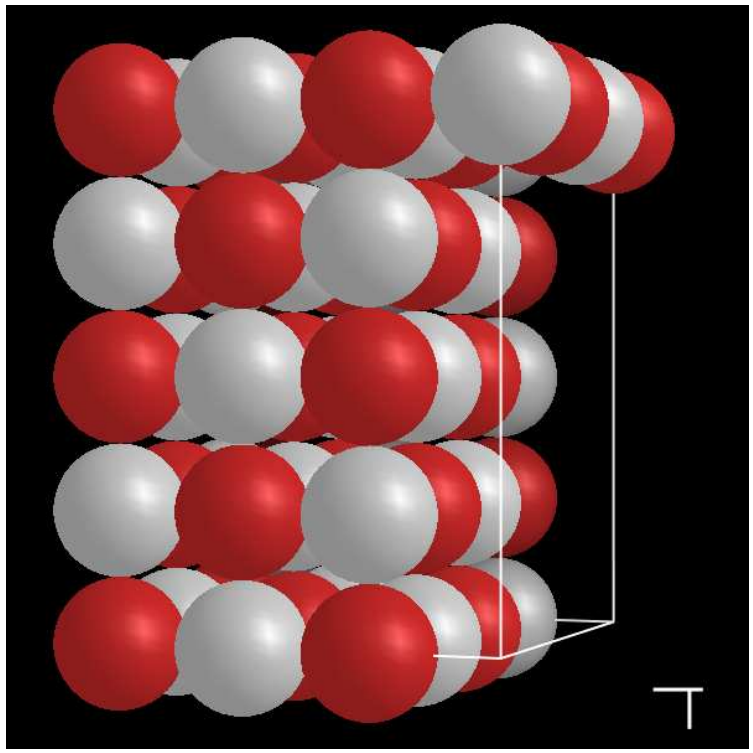
Funnels are typically associated to structural motifs

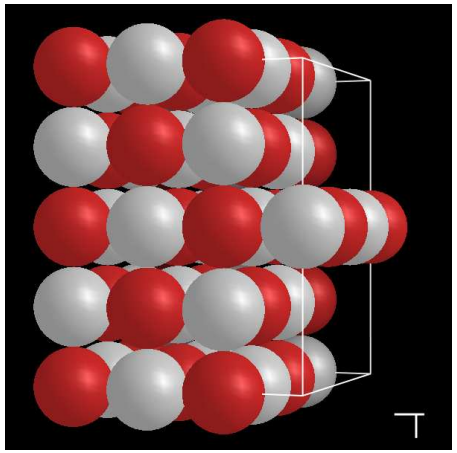
4 x 4 x 4 funnel





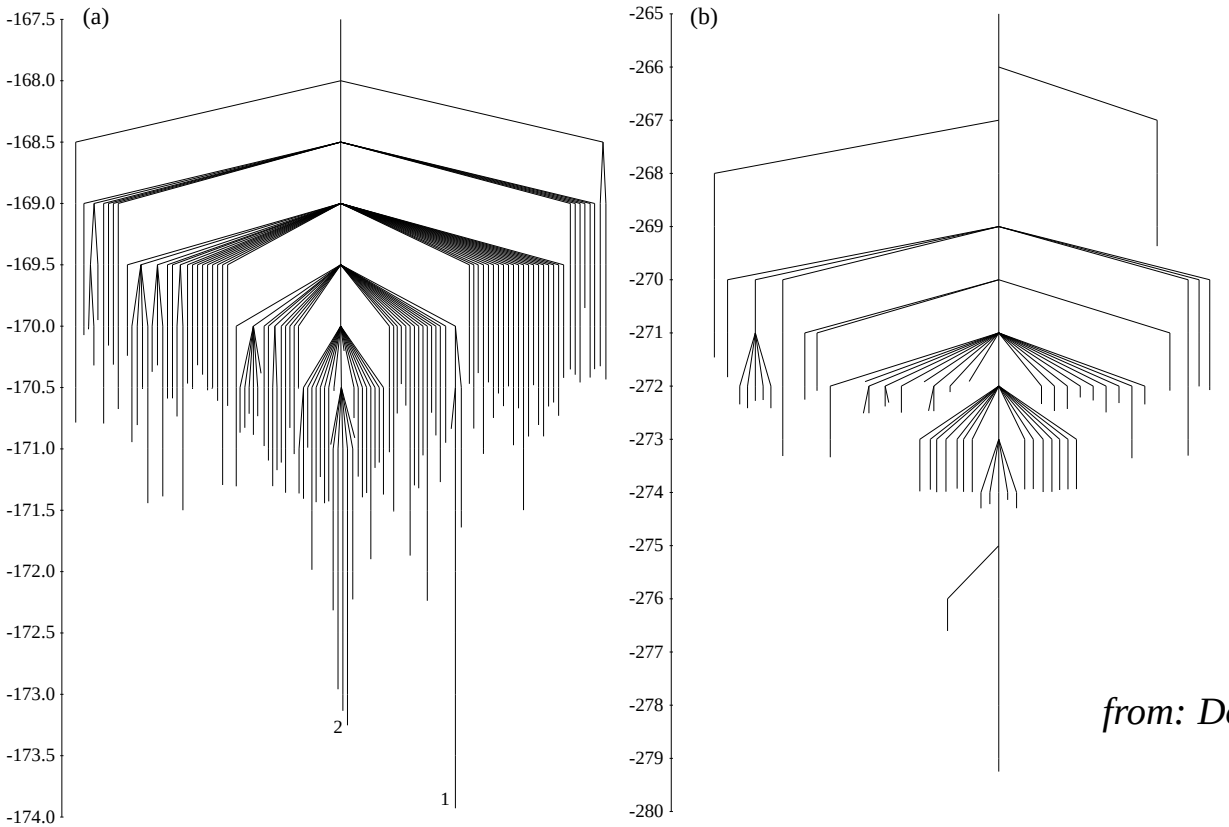
3 x 5 x 4 funnel





Disconnectivity graphs

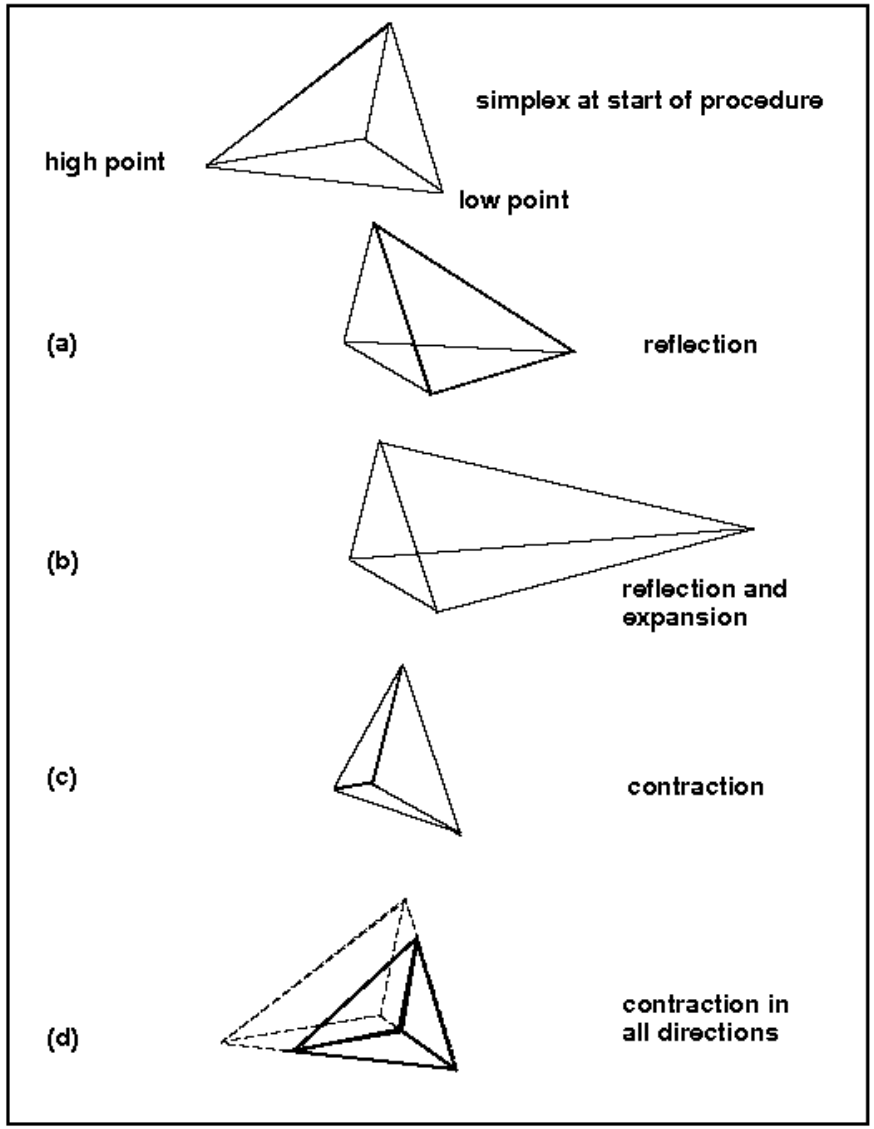
So-called disconnectivity graphs are probably the best way to visualize PESs. The height of each leave gives the energy of a local minimum. The highest point of the lines connecting different leaves gives the height of the barrier that has to be overcome on the PES when crossing from one minimum to the other one. The appearance of such a disconnectivity graph allows to detect easily the character of a PES. The disconnectivity graph on the right hand side below corresponds to a single funnel system that can easily find its ground state (global minimum), whereas the graph on the left hand side corresponds to a two funnel system.



from: Doye, Miller, Wales

Minimization without gradient: The downhill simplex method

The Downhill simplex method of Nelder and Mead is an extremely simple and stable method to minimize a function if no derivatives are available. It requires as an initial input a simplex in the N dimensional space together with the functional values on the corner of this simplex. A simplex in an N dimensional space has $N+1$ corners. A simplex in two dimensions is a triangle, in three dimensions an, in general, not regular tedrahedron. During the minimization this simplex undergoes a series of transformation shown in the Figure to the right. Im most transformations the corner with the highest functional value is moved.



The simplex is able to adapt to the topology of the function to be minimized. In a strechted valey for instance it becomes also elongated. Nevertheless its convergence rate is typically much lower than for the methods exploiting gradient information. Like all methods which do not stop based on gradient threshold norm, the downhill simplex method can get stuck even though it has not yet reached a local minimum. Therefore it is recommended to do some restarts with different initial simplexes, before claiming to have found the minimum. A movie showing the behaviour of the simplex during a minimization is available at: http://optlab-server.sce.carleton.ca/POAnimations2007/NonLinear7.html.

Minimizing a continuous 1-dim function

Minimizing a smooth function $E(x)$ is considerably easier than minimizing a non-smooth or even discontinuous function. If the first derivative exists, it points in the direction of the strongest increase of the function. The opposite direction consequently gives the strongest decrease of the function. The so-called steepest descent iteration

$$x_{l+1} = x_l - \alpha E'(x_l) \tag{5}$$

will therefore converge to the minimum $E(x_M)$ of the function E if the step size α is sufficiently small. If α is too large the iteration will diverge.

Next, we will discuss the case where the second derivative exists as well. Using in a combined way the information on the first and second derivative gives the most efficient minimization algorithms. The information about even higher derivatives is typically not used since this would be too complicated. Consequently we can assume in our discussion of minimization algorithms that we have to minimize a quadratic function. Then we can do a Taylor expansion of the function f and its derivative around an arbitrary point $\tilde{x}$

$$\begin{aligned} E(x) &= E(\tilde{x}) + (x-\tilde{x})\,E'(\tilde{x}) + \frac{1}{2}\,(x-\tilde{x})^2\,E'' \\ E'(x) &= E'(\tilde{x}) + (x-\tilde{x})\,E'' \end{aligned} \tag{6}$$

The stationary point $x = x_M$ where the derivative vanishes can easily be obtained by solving Eq. 6.

$$x_M = \tilde{x} - E'(\tilde{x})/E'' \tag{7}$$

We assume it is a minimum (i.e. $E'' > 0$) and not a maximum. Eq. 7 gives rise to the Newton iteration

$$x_{l+1} = x_l - E'(x_l)/E'' \tag{8}$$

The iteration of Eq. 8 will obviously converge in a single step for a quadratic function, but several iterations are needed for a general function. In the case of a quadratic function we did not have to worry where to evaluate the second derivative since it was a constant. This is of course not any more true for a general function. As a matter of fact we see that for the one-dimensional case we are discussing, Eq. 5 and Eq. 8 are identical if we put $\alpha = 1/E''$. Therefore one best adopts for the one-dimensional case the point of view that we just do steepest descent iterations where α is of the order of $1/E''$, but small enough to ensure convergence. In this case we do not have to answer the question where to evaluate E''.

Exercise: *Minimize the function describing the LJ potential (Eq. 2) numerically using Eq. 5. For which starting values does the iteration of Eq. 8 diverge if we evaluate E'' at x_l*

Minimizing continuous many-dimensional functions

The basic concepts of the 1-dimensional case can be carried over into the many dimensional case. The steepest descent iteration becomes

$$\vec{x}_{l+1} = \vec{x}_l - \alpha \vec{g}(\vec{x}_l) \tag{9}$$

where $\vec{g}(\vec{x}) = \nabla E(\vec{x})$ is the gradient of the function E. As in the 1-dim case this will converge to a minimum if α is sufficiently small.

For a function where the second derivatives exist we can again do a Taylor expansion

$$E(\vec{x}) = E(\tilde{\vec{x}}) + (\vec{x} - \tilde{\vec{x}})^T \vec{g}(\tilde{\vec{x}}) + \frac{1}{2} (\vec{x} - \tilde{\vec{x}})^T A (\vec{x} - \tilde{\vec{x}}) \tag{10}$$

$$\vec{g}(\vec{x}) = \vec{g}(\tilde{\vec{x}}) + A(\vec{x} - \tilde{\vec{x}}) \tag{11}$$

where A is the Hessian matrix

$$A(i,j) = \frac{\partial}{\partial x(i)} \frac{\partial}{\partial x(j)} E(\vec{x}) \tag{12}$$

For a quadratic form the Hessian matrix would not depend on the evaluation point, for a general function it of course does and the problem where to evaluate it will be postponed. Solving Eq. 11 for $\vec{x}$ leads to the Newton iteration

$$\vec{x}_{l+1} = \vec{x}_l - A^{-1} \vec{g}(\vec{x}_l) = \vec{x}_l - \vec{p}_l \tag{13}$$

Note that we have to solve in each iteration of the Newton method a linear system of equations for the preconditioned gradient vector p

$$A\vec{p} = \vec{g} \tag{14}$$

There are several basic problems with the Newton iteration:

- As mentioned before, it is not clear where to evaluate it for a non-quadratic form
- Realistic functions are not quadratic forms and so the theory is anyway only an approximation.
- The calculation of the exact Hessian matrix is numerically too expensive for complicated high-dimensional functions
- The matrix inversion of Eq. 14 is too expensive for high-dimensional functions.

Let us therefore define a slightly more general iteration that we will call preconditioned steepest descent iteration

$$\vec{x}_{l+1} = \vec{x}_l - P\,\vec{g}(\vec{x}_l) \tag{15}$$

where P is a still unspecified preconditioning matrix. Evidently we get the steepest descent iteration of Eq. 9 if we put $P = \alpha I$ and we get the Newton iteration of Eq. 13 if we put $P = A^{-1}$

Convergence analysis of the steepest descent iteration

For the convergence analysis we will again assume that we are already sufficiently close to the minimum, so that the function is a quadratic form. Because by definition the gradient vanishes at x_M the Taylor expansion of Eq. 10 becomes

$$E(\vec{x}) - E(\vec{x}_M) = \frac{1}{2}(\vec{x} - \vec{x}_M)^T A (\vec{x} - \vec{x}_M)$$

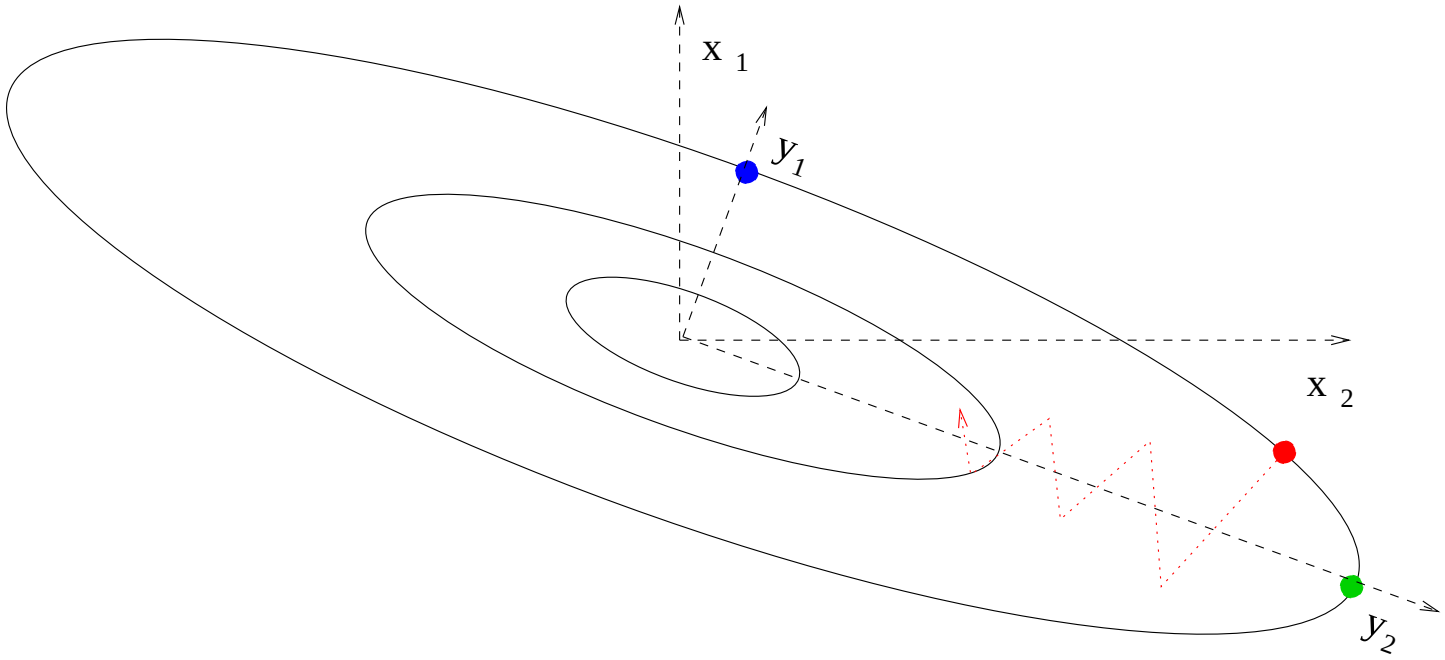
By shifting the origin (such that $\vec{x}_M = 0$) and the function (such that $E(\vec{x}_M) = 0$) we can without any restriction consider the simpler case

$$E(\vec{x}) = \frac{1}{2}\vec{x}^T A \vec{x} \tag{16}$$

Since the Hessian A is a positive definite symmetric matrix, we can go into an coordinate system $\vec{y}$, that is obtained by applying a unitary transformation U on the original coordinate system $\vec{x}$, where A becomes a positive real diagonal matrix $D = U^T A U$.

$$f(\vec{y}) = \frac{1}{2}\sum_k D(k,k)\, y(k)^2 \qquad ; \qquad g(k) = D(k,k)\, y(k) \tag{17}$$

Things are illustrated in the figure below. The ellipsoids represent the equipotential lines of the function f. The axis of the y coordinate system coincide with the principal axis of the ellipsoids.



Let us now assume that at a certain stage of a steepest descent iteration the current point $\vec{x}_l$ coincides with the blue dot. Since in this case the gradient points exactly in the direction of the minimum, we can find the minimum of this one-dimensional subproblem with a single steepest descent step if we chose $\alpha = 1/D(1,1)$. If we are at the green dot the same arguments apply except that now $\alpha = 1/D(2,2)$. In general our current iterations points are not located on any principle axis. The gradient of an arbitrary point such as the red dot has components of both principal axis. In order to guarantee convergence we have to be conservative and to choose $\alpha = 1/max[D(1,1),D(2,2)]$. Since the components of the gradient that correspond to principal axis with small eigenvalues will be damped too strongly, a steepest descent iteration in more than two di-

mensions is approaching the minimum very slowly by a large number of zigzag moves.
The generalization to more than 2 dimensions is obvious. The α of a steepest descent iteration has to be taken to be the reciprocal of the largest eigenvalue of the Hessian. Let us now examine the convergence rate for the multi-dimensional case in a more mathematical way. Since the steepest descent iteration is invariant under unitary transformations of the coordinate system we can without restriction consider a diagonal Hessian.
The steepest descent iteration then becomes

$$x_{l+1}(k) = x_l(k) - \alpha d(k) x_l(k)$$

where $d(k)$ is the vector containing the diagonal elements of the diagonal matrix A. Hence

$$x_{l+1}(k) = x_1(k)(1 - \alpha d(k))^l$$

where $\vec{x}_1$ is the starting vector for the iteration. Convergence can only be obtained if $|1 - \alpha d(k)| < 1$. Hence α can be at most twice of the reciprocal of the largest eigenvalue. So let us put

$$\alpha = t/d_{max} \tag{18}$$

where t is in between 0 and 2. For $t = 1$, the component k that will converge most slowly is the one associated to the smallest eigenvalue. Requiring this

component to be equal to a certain precision p gives

$$(1 - t\frac{d_{min}}{d_{max}})^l = p$$

The number of iterations l necessary to obtain this precision p is then given by

$$l = \ln(p)/\ln(1 - t\frac{d_{min}}{d_{max}}) \tag{19}$$

If $\frac{d_{min}}{d_{max}}$ is small, this is asymptotically equal to

$$l = -\ln(p)\frac{d_{max}}{t\, d_{min}} \propto \kappa \tag{20}$$

The ratio between the largest and the smallest eigenvalue of the Hessian matrix is called the condition number $\kappa = \frac{d_{max}}{d_{min}}$. We have thus the result that the number of iterations is proportional to the condition number κ in the steepest descent method. This is a big problem. As we will see the conditioning number is typically growing rapidly with respect to the size of the physical system represented by the matrix. Hence the number of iterations is growing substantially as well.

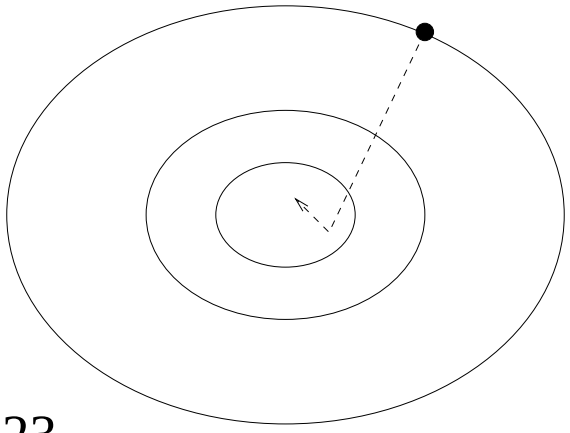
Convergence analysis of the preconditioned steepest descent iteration

The convergence analysis of the preconditioned steepest descent iteration of Eq. 15 is analogous to the one for the simple steepest descent iteration. The only difference is that we perform the analysis in a coordinate system that diagonalizes PA instead of A. The number of iterations is consequently given by the same formula

$$l = -\ln(p)\frac{d_{max}}{t\, d_{min}}$$

the only difference being that d_{max} and d_{min} are now the largest and smallest eigenvalues of PA. If the conditioning number of PA is smaller than of A, the number of iterations of the preconditioned steepest descent method will be reduced compared to the simple steepest descent method. A good preconditioning matrix is a compromise between 2 requirements. On the one hand it should give a small condition number, on the other hand it should be easy to calculate and to apply to the gradient. A frequent choice for P is a diagonal or sparse matrix.

Topology of preconditioned problem:



Steepest descent with line minimization

The prescription for a steepest descent iteration with line minimization for a function E is formally identical to an ordinary steepest descent minimization

$$\vec{x}_{l+1} = \vec{x}_l - \alpha \vec{g}$$

The difference is that α is not fixed, but optimized such that

$$\frac{\partial}{\partial \alpha} E(\vec{x} + \alpha \vec{g}) = 0$$

The line minimization ensures that the function will decrease at each iteration point. This does however not imply that one comes as close as possible to the minimum. As a matter of fact it turns out that with an optimal value of t (Eq.18) the convergence is as fast as with line minimization. In addition one iteration is much cheaper without the line minimization. The conclusion is that one should avoid line minimizations unless one can not at all estimate the largest eigenvalue of the Hessian matrix. If this estimation is possible steepest descent with some feedback is a recommendable strategy.

Steepest descent with energy feedback

A simple and powerful modification of the simple steepest descent method is the steepest descent with energy feedback. Assuming that the functional value represents the energy, we decrease the step size α if the energy rises in an iteration, otherwise we increase it. Since we know that in the case of an energy increase the parameter t (Eq.18) is roughly twice as large as would be optimal for the elimination of the stiff components, associated to large eigenvalues of the Hessian, we decrease α by a factor of $1/2$. If the energy goes down, as it should, we slightly increase α (e.g. by a factor of 1.05) to speed up the convergence.

Steepest descent with gradient feedback

In practice one finds that the following feedback gives slightly faster convergence than the energy feedback. At each iteration one calculates the angle between the current gradient vector and the gradient vector from the previous iteration. If the angle is larger than let's say 60 degrees, the step size α is decreased by a factor of $1/2$, otherwise it is increased by 1.05. In this way one avoids that consecutive gradients are pointing in opposite directions, which is obviously not desirable for a fast convergence.

Exercise: *An Lennard-Jones cluster is some artificial system where the 'atoms' interact through the Lennard-Jones potential. The potential energy E of such a cluster is*

$$E = \sum_{i=1,N_{at}} \sum_{j=1,i-1} 4\left(\frac{1}{|\mathbf{R}_i - \mathbf{R}_j|^{12}} - \frac{1}{|\mathbf{R}_i - \mathbf{R}_j|^{6}}\right)$$

Equilibrium geometries are given by minima of the potential energy. Minimize the energy to find an equilibrium geometry using the steepest descent method with energy and gradient feedback. α *will turn out to be of the order of 1.d-3. Which method is more efficient? The subroutine lenjon.f90 (available on http:/comphys.unibas.ch/teaching.htm) can be used to calculate the energy and forces (= negative gradient) of a Lennard-Jones cluster and the file posinp.xyz (on same website) contains the coordinates of the lowest energy cluster containing 38 atoms. Displace the atoms slightly from the geometry given in this file and use either one of the above mentioned minimization methods. Of course, one should in this way regain the coordinates of the cluster in the file posinp.xyz. If the atoms are strongly displaced one might however fall into another local minimum. The format of the file posinp.xyz is by most visualization programs such as v_sim software provided at http://www-drfmc.cea.fr/L_Sim/V_Sim/ The first line gives the number of atoms, the second is empty and the remaining lines give the atom type followed by its x,y and z coordinates.*

The conjugate gradient (CG) method

Eq. 11 tells us that finding the minimum of a quadratic form is equivalent to solving a linear system. The (preconditioned) steepest descent method can therefore be considered as the simplest method for solving a linear system of equations. There are however more powerful methods. One of the most popular methods is the conjugate gradient method. It is based on a bi-orthogonal sequence $\vec{g}_i, \vec{h}_i$

$$\vec{g}_i^T \vec{g}_j = \sum_k g_i(k) g_j(k) = \delta_{i,j}$$

$$\vec{h}_i^T A \vec{h}_j = \sum_{k,l} h_i(k) A(k,l) h_j(l) = \delta_{i,j}$$

Solving the system of equations

$$A\vec{x} = \vec{y}$$

is easy in the space spanned by h_i's. Writing $\vec{x} = \sum_j c_j \vec{h}_j$ one obtains

$$\sum_j c_j A \vec{h}_j = \vec{y}$$

Multiplying from the left by $\vec{h}_i$ one obtains

$$\sum_j c_j \vec{h}_i^T A \vec{h}_j = c_i = \vec{h}_i^T \vec{y}$$

In implementations of the conjugate gradient method one is simultaneously generating the bi-orthogonal sequence and then updating the approximate solution $\vec{x}$. For a m dimensional matrix there are at most m non-zero vectors $\vec{h}_i$. Therefore the exact solution has to be found after at most m iterations. This property is sometimes stressed in mathematics books. It is however not the property that makes conjugate gradient so useful in practice because

- It only holds for linear systems, whereas in practice the conjugate gradient method is usually applied for minimization problems where the function is not a quadratic form.
- Even for linear systems, it is violated in finite precision arithmetic because of rounding errors
- m iterations are far too expensive for large matrices

What makes the conjugate gradient method superior to the steepest descent method is its faster convergence rate. It can be shown that the number of iterations l is

$$l \propto \sqrt{\kappa}$$

instead of Eq. 20. For badly conditioned systems a lot can thus be gained by using the conjugate gradient instead of the steepest descent method, for well conditioned (or preconditioned) systems not much can be gained.

Generation of bi-orthogonal sequence

Here is the conjugate gradient formulation for the minimization of an arbitrary function E. Given an initial input guess x_0 we calculate $\vec{g}_0 = \nabla E(\vec{x}_0)$ and put $\vec{h}_0 = \vec{g}_0$.
Consecutive steps l:

- Determine by a line minimization the α_l that gives the lowest energy. That is usually done by finding the point where the derivative vanishes.

$$\frac{\partial}{\partial \alpha_l} E(\vec{x}_l + \alpha_l \vec{h}_l) = 0$$

- Update the solution

$$\vec{x}_{l+1} = \vec{x}_l + \alpha_l \vec{h}_l$$

- Calculate new gradient

$$\vec{g}_{l+1} = \nabla E(\vec{x}_{l+1})$$

- Calculate new $\vec{h}$

$$\vec{h}_{l+1} = \vec{g}_{l+1} + \gamma \vec{h}_l$$

where (Polak Ribiere)

$$\gamma = \frac{(\vec{g}_{l+1} - \vec{g}_l)^T \vec{g}_{l+1}}{\vec{g}_l^T \vec{g}_l}$$

For the case of a quadratic function one could simplify the Polak Ribiere formula by using the orthogonality of the vectors $\vec{g}_l$. However it turns out that for a general function where the orthogonality of the vectors $\vec{g}_l$ is not any more satisfied, the above Polak Ribiere formula is more stable.

CG assumes that we are in a quadratic region. This is frequently not the case at the start of a minimization procedure. In this case steepest descent with feedback is the method of choice. The CG will usually diverge in a strongly non-quadratic region.

The Newton method

Even though the Newton method is not widely used for finding minima we will discuss it in more detail since its practical implementation is very similar to the very useful preconditioned steepest descent iteration. So let us assume that we know the Hessian matrix $A(\mathbf{x})$ at a point $\mathbf{x}$ together with the force $\mathbf{f}(\mathbf{x})$. We can diagonalize this Hessian matrix to obtain its eigenvalues λ_i and eigenvectors $\mathbf{v}_i$ using standard routines such as the routine DSYEV from LAPACK. This routine overwrites the orginal matrix with all the eigenvectors. Each column of the matrix contains one eigenvector. The eigenvalues and the corresponding eigenvectors are in increasing order. We have now to transform the force in the new coordinate system spanned by the orthogonal set of eigenvectors. For this we have to calculate the coefficients g_i

$$g_i = \langle \mathbf{f} | \mathbf{v}_i \rangle = \sum_j f(j)\, v_i(j)$$

where $f(j)$ is the j-th component of $\mathbf{f}$ and $v_i(j)$ the j-th component of $\mathbf{v}_i$. g_i is the i-th component of the force vector in the new coordinate system. Since we are now in the principal axis coordinate system we can multiply each component by the ideal stepsize which is the inverse curvature. Since the curvature is given by the eigenvalues we have

$$\hat{g}_i = g_i / \lambda_i \tag{21}$$

Then we have to go back in our original coordinate system to get the preconditioned force $\hat{\mathbf{f}}$. The vector $\hat{\mathbf{f}}$ is what one would obtain by applying A^{-1} to $\mathbf{f}$:

$$\hat{\mathbf{f}} = \sum_i \hat{g}_i \, \mathbf{v}_i \tag{22}$$

Finally we update the atomic positions according to

$$\mathbf{R} \leftarrow \mathbf{R} + \hat{\mathbf{f}}$$

Since the energy is invariant under translations, the Hessian matrix has three eigenvectors with zero eigenvalues and is thus singular. Hence A^{-1} does not exist. Numerically the eigenvalues are not strictly zero but very small. These nearly zero eigenvalues can lead to problem in Eq. 21. Unless the system is in the field of an external potential the overall translational force has to be zero and so the three components $g(i)$ that correspond to the translations have to be zero. Analytically we have thus three cases in Eq. 21 where zero is divided by zero, in numerical work we will just divide two very small numbers. Since these numbers are essentially rounding noise the result would be completely wrong. For a molecule at equilibrium it can be shown that there are three more zero eigenvalues that correspond to rotations. If the molecule is close to a local minimum the three eigenvalues are not exactly zero but very small which will lead as well to numerical problems. To avoid such problems we have to modify

Eq. 21 to

$$\hat{g}_i = \frac{g_i}{\lambda_i + \gamma} \tag{23}$$

and this value of $\hat{g}(i)$ has then to be used in Eq. 22. γ has to be chosen such that the denominator is always positive and not too small. In the case of a molecule or cluster, a good choice for γ is to set it equal to half of the 7-th eigenvalue.

The practical implementation of the Newton method is very similar to the preconditioned steepest descent iteration. The main difference is that in the preconditioned steepest descent method one uses an approximate Hessian instead of the exact Hessian. Approximate Hessians can also have zero eigenvalues which have te be treated in a similar manner as in the Newton method.

Exercise: *Use the Newton method to find equilibrium geometries of the 38 atom LJ cluster of the previous exercise. Show that the convergence rate is much faster than with the steepest descent method. A routine hesslj.f90 that calculates the Hessian matris is available on http:/comphys.unibas.ch/teaching.htm.*

Quasi Newton (QN) methods

The basic principle of quasi Newton methods is to build up information about the Hessian matrix from the gradient evaluations during the minimization iterations. This is possible because the Hessian matrix (Eq. 12) can be obtained by finite differences from the forces or gradients

$$A(\mathbf{R}) = \begin{pmatrix} . & . & \cdots & . \\ \frac{\mathbf{g}(\mathbf{R}+h\mathbf{e}_1)-\mathbf{g}(\mathbf{R})}{h} & ; \frac{\mathbf{g}(\mathbf{R}+h\mathbf{e}_2)-\mathbf{g}(\mathbf{R})}{h} & ; \cdots ; & \frac{\mathbf{g}(\mathbf{R}+h\mathbf{e}_n)-\mathbf{g}(\mathbf{R})}{h} \\ . & . & \cdots & . \end{pmatrix}$$

Denoting by $\mathbf{G}_i$ the finite difference vector between the two gradients $\mathbf{G}_i = \mathbf{g}(\mathbf{R}+h\mathbf{e}_i) - \mathbf{g}(\mathbf{R})$ we see that the matrix element $A_{i,j}$ is given by the scalar product $\frac{1}{h}\langle \mathbf{G}_i | \mathbf{e}_j \rangle$, where $\mathbf{e}_i$ is an orthonormal set of vectors. Now very similar quantities are a by-product of any minimization. If the system is moved in a minimization step from $\mathbf{R}$ to $\mathbf{R}+\mathbf{d}$ and the forces are evaluated at both points we can calculate the curvature along the direction $\mathbf{d}_i$.

$$\frac{\partial^2}{\partial x^2} E(\mathbf{R}+x\mathbf{d})|_{x=0} = \frac{\partial}{\partial x} \langle \mathbf{g}(\mathbf{R}+x\mathbf{d}) | \mathbf{d} \rangle |_{x=0} \approx \frac{\langle \mathbf{G}(\mathbf{d}) | \mathbf{d} \rangle}{\langle \mathbf{d} | \mathbf{d} \rangle}$$

where we have again denoted by $\mathbf{G}$ the difference betwenn the gradient vectors $\mathbf{G}(\mathbf{d}) = \mathbf{g}(\mathbf{R}+\mathbf{d}) - \mathbf{g}(\mathbf{R})$. If we assume our function E to be a perfect quadratic

form then $\mathbf{G}(\mathbf{d})$ is equal to $A\mathbf{d}$. The Hessian matrix B with respect to an non-orthogonal set of vectors $\mathbf{d}_i$ is then given by

$$B_{i,j} = \langle \mathbf{d}_i | A | \mathbf{d}_j \rangle = \langle \mathbf{G}(\mathbf{d}_i) | \mathbf{d}_j \rangle$$

The eigenvalues of the Hessian A can consequently be obtained by solving the generalized eigenvalue problem for the matrices B and S

$$B\mathbf{v}_l = \lambda_l S\mathbf{v}_l \tag{24}$$

where S is the overlap matrix $S_{i,j} = \langle \mathbf{d}_i | \mathbf{d}_j \rangle$. This approach clearly fails if the vectors $\mathbf{d}_i$ are linearly dependent. Numerical problems actually already arise if the vectors $\mathbf{d}_i$ are nearly linearly dependent, i.e if the overlap matrix is nearly singular, which can be detected by very small eigenvalues of the overlap matrix. Standard Quasi Newton methods such as the popular BFGS variant, named after their inventors Broydens, Fletcher, Goldfarb and Shanno, can therefore fail in such cases. Linearly dependent vectors can be encountered if the minimization is started far away from the local minimum. If the minimiztion starts close to the local minimum where the function can be well approximated by a quadratic form such problems do generally not arise and rapid convergence is generally found. The most popular implementation of the BFGS method is the Limited memory LBFGS variant where second derivative information is exploited only

from the few last iterations. Typically a history length of about 10 is choosen. Even if the dimension of the entire Hessian matrix is in general much larger than this history length, it turns out that the convergence speed is not improved by a longer history. On the contrary, a too long history can lead to numerical instabilities.

Exercise: *Extracting curvature information from a set of gradient vectors*
Take the local minimum $\mathbf{R}_0$ *of the 38 atom LJ cluster of the previous exercise and calculate the Hessian matrix at this point using the subroutine hesslj.f90 (on web site http:/comphys.unibas.ch/teaching.htm). Find the eigenvalues of this matrix by using the LAPACK routine DSYEV. Six of these eigenvalues should be zero corresponding to the three translations and the three rotations that leave the energy invariant. These eigenvalues will serve as reference values for the second part of this exercise.*

In this second part, first perturb this minimum by a random displacement

$$\mathbf{r}_0 = \mathbf{R}_0 + a\boldsymbol{\emptyset}$$

where $\boldsymbol{\emptyset}$ *is a random vector and the amplitude a should be about 1.e-2. Generate then a sequence of configurations* $\mathbf{r}_i$, $i = 1, \ldots, n$, *by performing a steepest descent geometry optimization with a energy or gradient feedback. Consider then the sequence of displacement vectors* $\mathbf{d}_i$

$$\mathbf{d}_i = \mathbf{r}_i - \mathbf{r}_{i-1}$$

Use at each configuration $\mathbf{r}_i$ *the corresponding force* $\mathbf{f}_i$ *to obtain the gradient difference* $\mathbf{G}_i$

$$\mathbf{G}_i = -(\mathbf{f}_i - \mathbf{f}_{i-1})$$

Calculate then the overlap matrix $S_{i,j} = \langle \mathbf{d}_i | \mathbf{d}_j \rangle$ *and the Hessian matrix in this basis,* $B_{i,j} = \langle \mathbf{G}_i | \mathbf{d}_j \rangle$ *for several values of n. For a purely quadratic form the Hessian matrix B would be symmetric, i.e* $B_{i,j} = B_{j,i}$*. Since this is not the case there will be small deviations from symmetry. Check that these deviations get smaller if the initial displacement amplitude is reduced. Next calculate the eigenvalues of the generalized eigenvalue problem of Eq. 24 by using the Lapack routine DSYGV. Verify that you get for small values of n already with reasonable precision the large eigenvalues of the full Hession matrix and that all the eigenvalues lie within the spectrum of the full Hessian matrix. Verify that once n gets larger numerical instabilities arise which prevent obtaining all the* 3×38 *eigenvalues of the full Hessian matrix correctly.*

The DIIS (Direct Inversion in Iterative Subspace) minimization method

Be $\vec{c}_0$ the exact solution of a quadratic minimization problem and $\vec{c}_i$ (i=1, ..,m) a set of *m* approximate solution vectors. Their error vectors are defined by

$$\vec{e}_m = \vec{c}_m - \vec{c}_0$$

We form a new vector $\tilde{\vec{c}}_m$

$$\tilde{\vec{c}}_m = \sum_{i=1}^{m} d_i \vec{c}_i$$

If $\tilde{\vec{c}}_m$ was the exact solution, it would fulfill

$$\begin{aligned} \sum_{i=1}^{m} d_i \vec{c}_i &= \vec{c}_0 \\ \sum_{i=1}^{m} d_i (\vec{c}_0 + \vec{e}_i) &= \vec{c}_0 \\ \sum_{i=1}^{m} d_i \vec{c}_0 + \sum_{i=1}^{m} d_i \vec{e}_i &= \vec{c}_0 \end{aligned}$$

This is satisfied if

$$\sum_{i=1}^{m} d_i = 1 \qquad ; \qquad \sum_{i=1}^{m} d_i \vec{e}_i = 0$$

Last condition can only be fulfilled approximately, leading to the minimization problem

$$\min \left[\langle \sum_{i=1}^{m} d_i e_i | \sum_{i=1}^{m} d_i e_i \rangle \right]$$

under the constraint $\sum_{i=1}^{m} d_i = 1$. This leads to the system of equations

$$\begin{pmatrix} \langle e_1|e_1\rangle & \langle e_1|e_2\rangle & \dots & \langle e_1|e_m\rangle & 1 \\ \langle e_2|e_1\rangle & \langle e_2|e_2\rangle & \dots & \langle e_2|e_m\rangle & 1 \\ \dots & \dots & \dots & \dots & . \\ \langle e_m|e_1\rangle & \langle e_m|e_2\rangle & \dots & \langle e_m|e_m\rangle & 1 \\ 1 & 1 & \dots & 1 & 0 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ . \\ d_m \\ d_{m+1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \qquad (25)$$

In practice the error vectors are approximated by $\vec{e}_i = P\vec{g}_i$
The new vector is then given by

$$\tilde{\vec{g}}_m = \nabla f(\tilde{\vec{c}}_m)$$

$$\vec{c}_{m+1} = \tilde{\vec{c}}_m - P\tilde{\vec{g}}_m$$

Variable preconditioning DIIS implementation

There are possibly two preconditioning matrices $P = P_m$ and $\tilde{P} = \tilde{P}_m$ that depend on the iteration step m.
At the m-th step do

- $$\vec{g}_m = \nabla f(\vec{c}_m)$$
- $$\vec{e}_i = P_m \vec{g}_i \qquad , \qquad i = 1, \ldots, m$$
- Solve Eq. 25 to get $\tilde{\vec{c}}_m = \sum_{i=1}^{m} d_i \vec{c}_i$ Under the assumption that we are in a quadratic region, the coefficients d_i allow us then also to calculate

$$\tilde{\vec{g}}_m = \nabla f(\tilde{\vec{c}}_m) = \nabla f(\sum_{i=1}^{m} d_i \vec{c}_i) = \sum_{i=1}^{m} d_i \nabla f(\vec{c}_i) = \sum_{i=1}^{m} d_i \vec{g}_i$$

- $$\vec{c}_{m+1} = \tilde{\vec{c}}_m - \tilde{P}_m \tilde{\vec{g}}_m$$

This implementation requires to store 3 sequences of vectors: $\vec{c}_i$, $\vec{g}_i$ and $\vec{e}_i$. If the application of the preconditioning matrix P_m is cheap the $\vec{e}_i$'s can be calculated on the fly form the $\vec{g}_i$'s and one does not have to store them. The most expensive step is usually the calculation of the gradient $\vec{g}_m$, which has to be done once during each iteration.

Fixed preconditioning DIIS implementation

Only the two sequences $\vec{c}_m$ and $\vec{e}_m$ have to be stored if there is a single preconditioning matrix P that does not change during the iterations.

- $$\vec{g}_m = \nabla f(\vec{c}_m)$$
- $$\vec{e}_m = P\vec{g}_m$$
- Solve Eq. 25 to get $\tilde{\vec{c}}_m = \sum_{i=1}^{m} d_i \vec{c}_i$ Under the assumption that we are in a quadratic region, the coefficients d_i would allow us then also to calculate $\tilde{\vec{g}}_m$, even though we do not actually calculate it

$$\tilde{\vec{g}}_m = \nabla f(\tilde{\vec{c}}_m) = \nabla f(\sum_{i=1}^{m} d_i \vec{c}_i) = \sum_{i=1}^{m} d_i \nabla f(\vec{c}_i) = \sum_{i=1}^{m} d_i \vec{g}_i$$

- $$\vec{c}_{m+1} = \tilde{\vec{c}}_m - P\tilde{\vec{g}}_m = \sum_{i=1}^{m} d_i \vec{c}_i - P\sum_{i=1}^{m} d_i \vec{g}_i = \sum_{i=1}^{m} d_i(\vec{c}_i - P\vec{g}_i) = \sum_{i=1}^{m} d_i(\vec{c}_i - \vec{e}_i)$$

Steepest descent versus CG, QN and CG

Both CG, QN and DIIS assume that we are in a quadratic region. This is frequently not the case at the start of a minimization procedure. In this case steepest descent with feedback is the method of choice. Both CG QN and DIIS will usually diverge in a strongly non-quadratic region. The line minimization makes the CG however somewhat more stable than DIIS.

QN and DIIS versus CG

The DIIS method has the advantage that it is the most flexible. Even though there is also a preconditioned version of the CG method there is no preconditioned CG method that would allow for variable preconditioning. An initial Hessian which can be considered as a preconditioner can also be provided in the QN methods. However if this Hessian turns out to be inadequate the iterations done so far have to be discarded for building up information about the Hessian. Since the set of approximate solution vectors $\vec{c}_m$ in the DIIS method is arbitrary, the the DIIS method can be applied to a constrained minimization problem. Imposing constraints after each iteration modifies the sequence of approximate solution vectors generated during the iterations and would be illegal in the standard CG and QN method. In the DIIS method imposing the constraints does not bother. The disadvantage of the DIIS method compared to

the CG method is that it needs more memory to hold the set of vectors $\vec{c}_i$ and $\vec{e}_i$. If memory is limited, the sequence of vectors can be restricted to a certain maximum value. Like in the QN methods such a limited history length does not negatively affect the performance. Another advantage of both the QN and the DIIS method is that it requires only a single force evaluation per step because no line minimization is required.

Least square problems: The Levenberg Marquart method

In the case of a least square fitting problem, the Hessian near to the solution can be obtained easily from the gradient vectors as will be shown in the below. The only reason not to use information about the Hessian would be that one is still far away from a quadratic region, where some steepest descent type method is more stable.

The penalty function χ to be minimized is by definition given by

$$\chi^2(\mathbf{a}) = \sum_{i=1}^{N} \left(\frac{y_i - y(x_i;\mathbf{a})}{\sigma_i} \right)^2$$

for a nonlinear fiting problem. $\mathbf{a}$ is a vector of length M containing the fitting parameters of the function y to be optimized and the N data of the fitting data base are denoted by (x_i, y_i). The gradient is given by

$$\mathbf{g}_k = \frac{\partial \chi^2}{\partial a_k} = -2 \sum_{i=1}^{N} \frac{y_i - y(x_i;\mathbf{a})}{\sigma_i^2} \frac{\partial y(x_i;\mathbf{a})}{\partial a_k}$$

and the Hessian is given by

$$\frac{\partial^2 \chi^2}{\partial a_k \partial a_l} = -2 \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left(\frac{\partial y(x_i;\mathbf{a})}{\partial a_k} \frac{\partial y(x_i;\mathbf{a})}{\partial a_l} - (y_i - y(x_i;\mathbf{a})) \frac{\partial^2 y(x_i;\mathbf{a})}{\partial a_k \partial a_l} \right)$$

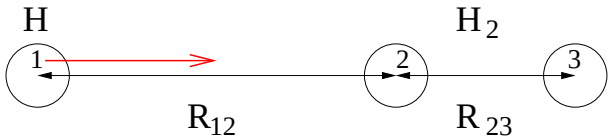
If the fit succeeds to reproduce the data perfectly, $y_i - y(x_i;\mathbf{a})$ tends to zero and the second term in the formula for the Hessian vanishes. Hence the Hessian can be obtained purely from gradient information.
In the Levenberg-Marquart method one uses now a preconditioning matrix which is given by

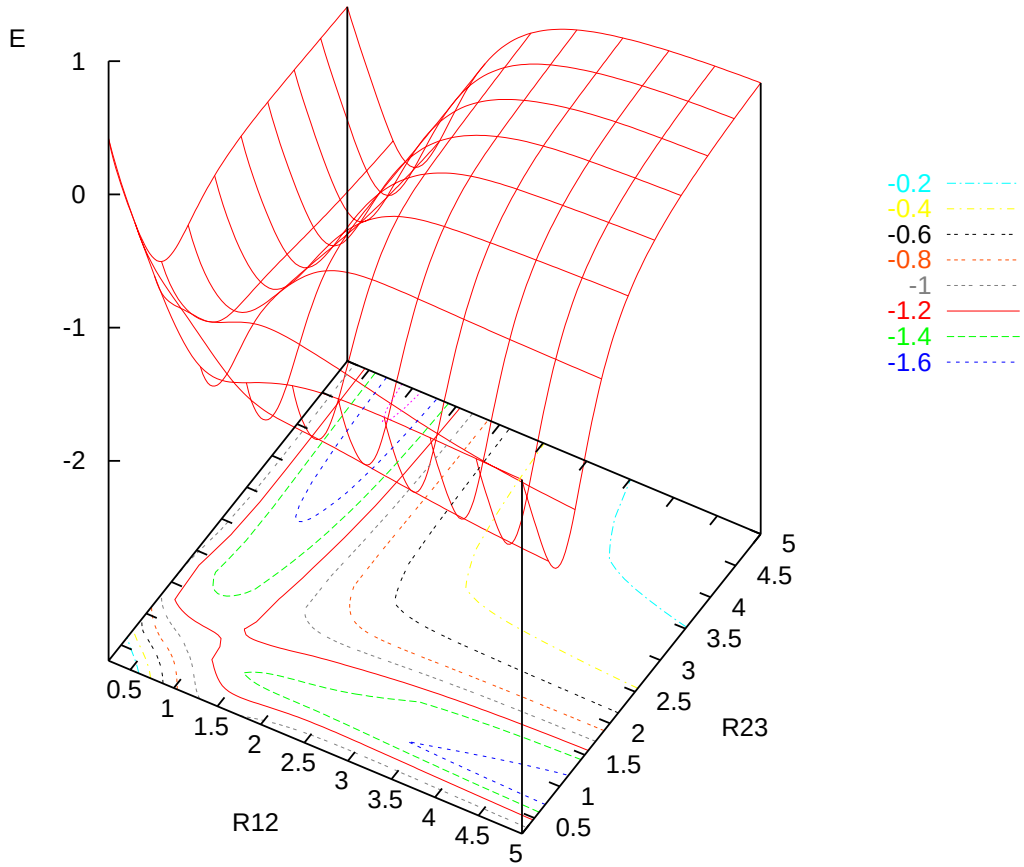
$$P_{k,l} = -2\sum_{i=1}^{N} \frac{1}{\sigma_i^2} \frac{\partial y(x_i;\mathbf{a})}{\partial a_k} \frac{\partial y(x_i;\mathbf{a})}{\partial a_l} + \lambda\delta_{k,l}$$

If λ is zero the resulting iteration $\mathbf{a} \leftarrow \mathbf{a} - P\mathbf{g}$ will be identical to a Newton iteration close to a perfect fitting solution. For very large values of λ the the iteration will coincide with a steepest descent iteration with a very small step size of approximately $1/\lambda$. In the Levenberg-Marquart the value of λ is adjusted by a feedback loop to be close to optimal. One starts with a rather large value of λ such that the first steps are steepest descent like. Then the value of λ is reduced to come closer to the more efficient Newton iteration. If the λ is however too small large steps can be taken and these steps might not be correct because the preconditioning matrix P is not the exact Hessian. In this case the value χ^2 will increase and then a feedback mechanism will again increase the values of λ. This is very similar to the steepest descent iteration with energy feedback that was discussed previously.

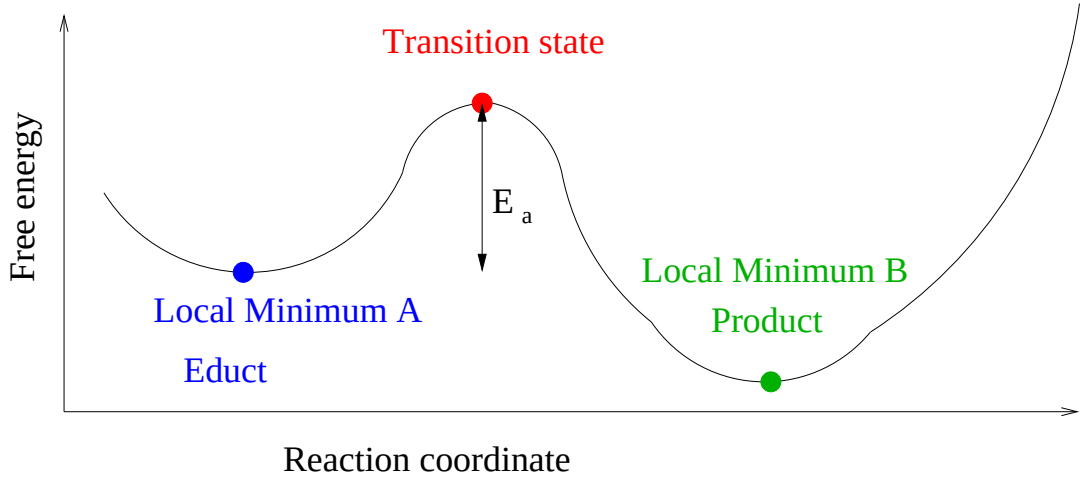
Saddle points

In addition to the local minima of the total energy function the saddle points are also of interest. Their height and shape determines the dynamics of a molecular system. Within harmonic transition state theory one can obtain the rate of a chemical reaction from the properties of the saddle point. Let us look at the simplest chemical reaction, $H + H_2 \rightarrow H_2 + H$, whose energy function is sketched below as a function of the two distances between the hydrogen atoms. We assume that the 3 atoms all move along a line.





In realistic cases the total energy function is a very high dimensional function that can not be visualized. For this reason one introduces the so-called reaction coordinate and plots the energy as a function of this reaction coordinate.



The reaction coordinate consists of the two path that one obtains if one starts a steepest descent minimization from the saddle point going slightly in the direction of the two minima connected by the saddle point. Even though the transition is a saddle point on the high dimensional total energy surface it becomes a maximum along the reaction coordinate. The height of the barrier measured relative to the educt is called the activation energy. We have plotted the energy along the reaction coordinate and so the barrier height is an energy difference. In reality one should consider the free energy and in transitions state theory the activation energy is actually a free energy. In most cases the pure energy contributions are however more important than the entropy contributions.

Rare events

Since the time step in MD is very small it is not possible to follow events that take place on time scales of a micro second or more. Events that take place very rarely on the time scale of molecular vibrations are called rare events. Doing MD to observe such a rare event would be boring. Essentially one would observe endless oscillations around a certain local minimum of the total energy surface until all of a sudden the event takes place. When the event 'takes place' the MD trajectory goes over a barrier into the bassin associated to another local minimum. In the language of chemistry such an event is usually a chemical reaction. The initial minimum corresponds to the educt of the chemical reaction and the final local minimum to the product of the chemical reaction. In physics such an event could for instance be the annihilation of a defect in a crystal. The initial local minimum corresponds to a crystal containing a stable defect and the final local minimum to the perfect crystal. What one would like to know for rare events is how frequently they happen on average. Let us denote this average time by τ. In this way one knows for instance the rate konstant k of a chemical reaction which is simply $1/\tau$. Harmonic transition state theory gives an approximate formula for k

$$k = \frac{k_B T}{\hbar} \exp(\frac{E_a}{k_B T}) \tag{26}$$

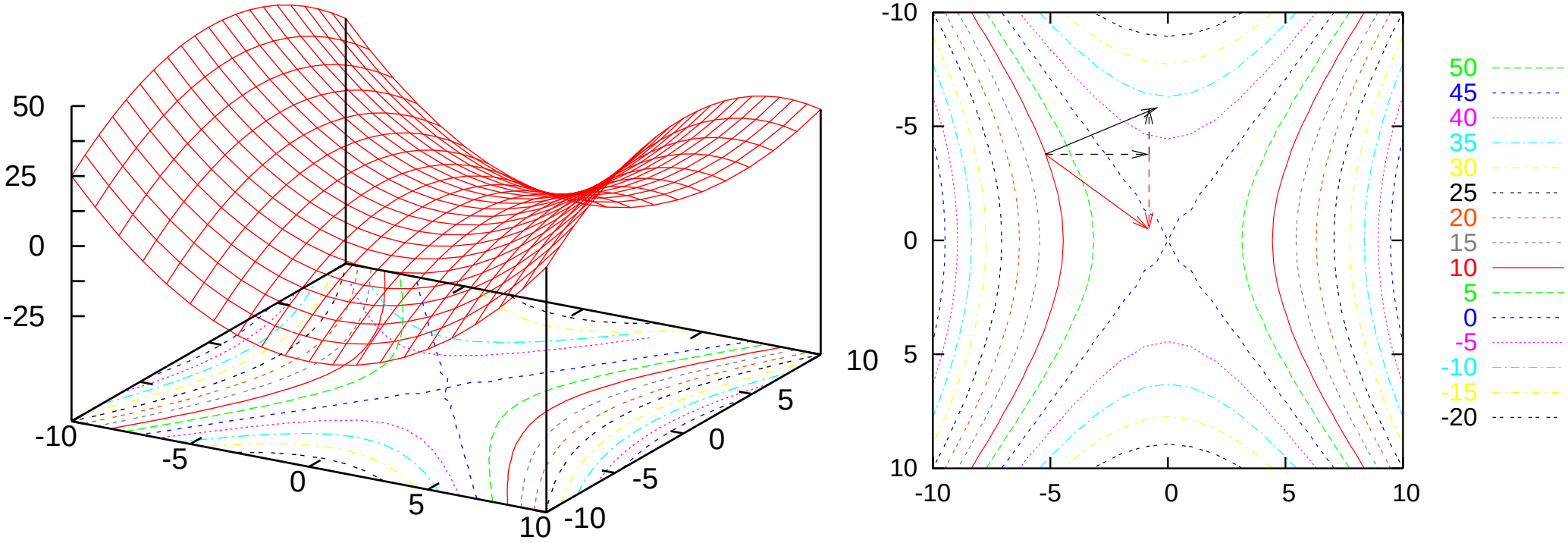
where k_B is Boltzmann's constant, $\hbar$ Plank's constant, E_a the activation energy.

Finding saddle points

Finding saddle points is more difficult than finding minima. In the latter case one just has to go downhill whereas in the former case one has to go downhill along certain directions but uphill along other directions. The direction along which one has to go uphill is the direction with negative curvature. Determining this direction requires finding the eigenvector of the Hessian matrix that has a negative eigenvalue. Calculating the Hessian matrix is typically rather difficult. Except for simple model potentials such as the LJ potential it can not be calculated analytically but only numerically at rather high cost of computer time. For the moment let us assume that the Hessian matrix is available.

The topology of a saddle point of a 2-dim function is shown on the next side. The surface is already aligned such that the prinipal axis coincide with the x and y axis. Along the x axis the curvature is 1, along the y axis it is -1/2. The left hand side of the figure shows both the surface and the contour lines in the x y plane. The right hand side shows only the contour lines together with the force (black arrow) at a certain iteration of our saddle point search. As expected the force does not point in the direction of the saddle point. Like in the case of Newton's method we have to decompose the force into its components along the principal axis (dotted arrays). The component along the component with eigenvector 1 is not modified since the optimal step size for curvature 1 is 1. The component along the eigenvector with eigenvalue -1/2 is multiplied

by -2. The negative sign comes from fact that we want to go uphill along this component and the value of two from the fact that the absolute value of the curvature is 1/2. Adding together the scaled components gives an vector (red arrow) that points towards the saddle point.



The obvious generalization to higher dimensional problems is the following. To find a first order saddle point, i.e a saddle point where one eigenvalue of the Hessian is negative and all other are positive we have to first decompose the force **f** into the components c_i of eigenvectors of the hessian matrix. Denoting the eigenvectors by $\mathbf{v_i}$ we have

$$c_i = \langle \mathbf{f} | \mathbf{v}_i \rangle = \sum_j f(j)\, v_i(j)$$

Then we have to scale each component by the curvature taking into account that we want to go downhill except along the component with the smallest curvature. This is achieved by scaling the first component as

$$d_1 = -c_1/|\lambda_1| \tag{27}$$

and all other component as

$$d_i = c_i/|\lambda_i| \tag{28}$$

Close to a first order saddle point there will always be one negative eigenvalue. Far away all eigenvalues may however be positive. Taking the absolute value of λ_1 in Eq. 27 ensures that one also moves uphill along one mode if all eigenvalues are positive.

In the case of finding saddle points we encounter the same problem as in the case of finding minima with the Newton method. Translation and rotational

modes have eigenvalues that are zero or close to zero. In addition some modes can be close to zero 'by accident' if an eigenvalue changes sign during the saddle point search. Analogeous to the case of the Newton minimization we therefore have to modify again Eq. 27 and Eq. 28:

$$d_1 = -\frac{c_1}{|\lambda_1| + \gamma} \tag{29}$$

and all other component as

$$d_i = \frac{c_i}{|\lambda_i| + \gamma} \tag{30}$$

After having scaled the different components we have to assemble the preconditioned force vector $\hat{\mathbf{f}}$ (shown by the red arrow in the above figure)

$$\hat{\mathbf{f}} = \sum_i d_i \mathbf{v_i}$$

Finally the atoms are moved according to

$$\mathbf{R}_i \leftarrow \mathbf{R}_i + \hat{\mathbf{f}}$$

Iterating the above described steps leads into the saddle point.

Exercise: *Saddle points in the LJ 38 cluster*

Implement a subroutine that finds local minima by using the Newton method. A subroutine hesslj.f90 that returns the Hessian matrix for LJ systems can be downloaded from http:/comphys.unibas.ch/teaching.htm. To diagonalize the Hessian matrix use the DSYEV routine from the Lapack library. To test your implementation of the Newton method take the global minimum found previously and displace the atoms randomly by .1 units. The convergence rate should be considerably faster than with the steepest descent method.

Using as a starting point the Newton geometry optimization routine, write next a routine that can find saddle points. Use this routine then to find several saddle points that lead away from the global minimum. Save in addition to the coordinates of the saddle point $\mathbf{R}_{saddle}$ *the direction along which the curvature is negative* $\mathbf{D}_{neg}$ *which is the eigenvector belonging to the negative eigenvalue. Plot the energy along this direction, i.e. plot the function*

$$f(s) = E(\mathbf{R}_{saddle} + s\mathbf{D}_{neg})$$

For each of the saddle points found, determine next which 2 minima it connects. To do this use the two points $\mathbf{R}_{saddle} \pm s_0\mathbf{D}_{neg}$ *as a starting point for a steepest descent geometry optimisation. Finally collect all the results to plot the energy along the reaction coordinate.*

Finding negative curvature modes without diagonalizing the Hessian matrix

The method to locate saddle points presented in the previous section is conceptually simple and efficient in the sense that only a relatively small number of number of iterations steps is needed to find the saddle point. One single iteration step is however in practice typically too expensive since the calculation of the Hessian matrix is required. If finite differences are used at least $6N_{at}$ force evaluations are required to calculate the Hessian matrix. Other approaches that are applicable on the density functionla level require a similar numerical effort. Fortunatley the eigenmode with the lowest (in our case negative) curvature at a point $\mathbf{R}$ can be calculated by an alternative method, the dimer method, that requires in most cases a much small number of force evaluations. The dimer consists of two physical configurations $\mathbf{R}+\frac{1}{2}\mathbf{d}$ and $\mathbf{R}-\frac{1}{2}\mathbf{d}$ that are separated by a given small distance d. If one minimizes the sum of the energies $E(\mathbf{R}+\frac{1}{2}\mathbf{d})+E(\mathbf{R}-\frac{1}{2}\mathbf{d})$ with respect to $\mathbf{d}$ under the constraint that the distance d be constant, the vector $\mathbf{d}$ will align along the mode with the lowest curvature. Since the point $\mathbf{R}$ is fixed we can alternatively also consider the expression

$$\frac{1}{2}\left(E(\mathbf{R}+\frac{1}{2}\mathbf{d})-2E(\mathbf{R})+E(\mathbf{R}-\frac{1}{2}\mathbf{d})\right) \tag{31}$$

for the minimization which represents a finite difference for the second deriva-

tive. The Taylor expansion of the first term in these energy sum gives

$$E(\mathbf{R}+\frac{1}{2}\mathbf{d}) \approx E(\mathbf{R}) + \langle \mathbf{d}|\mathbf{g}(E(\mathbf{R})\rangle + \frac{1}{2}\langle \mathbf{d}|A|\mathbf{d}\rangle$$

Doing the analogeous Taylor expension for $E(\mathbf{R}-\frac{1}{2}\mathbf{d})$ and inserting both of them into Eq. 31 gives

$$E(\mathbf{R}+\frac{1}{2}\mathbf{d}) - 2E(\mathbf{R}) + E(\mathbf{R}-\frac{1}{2}\mathbf{d}) \approx \langle \mathbf{d}|A|\mathbf{d}\rangle$$

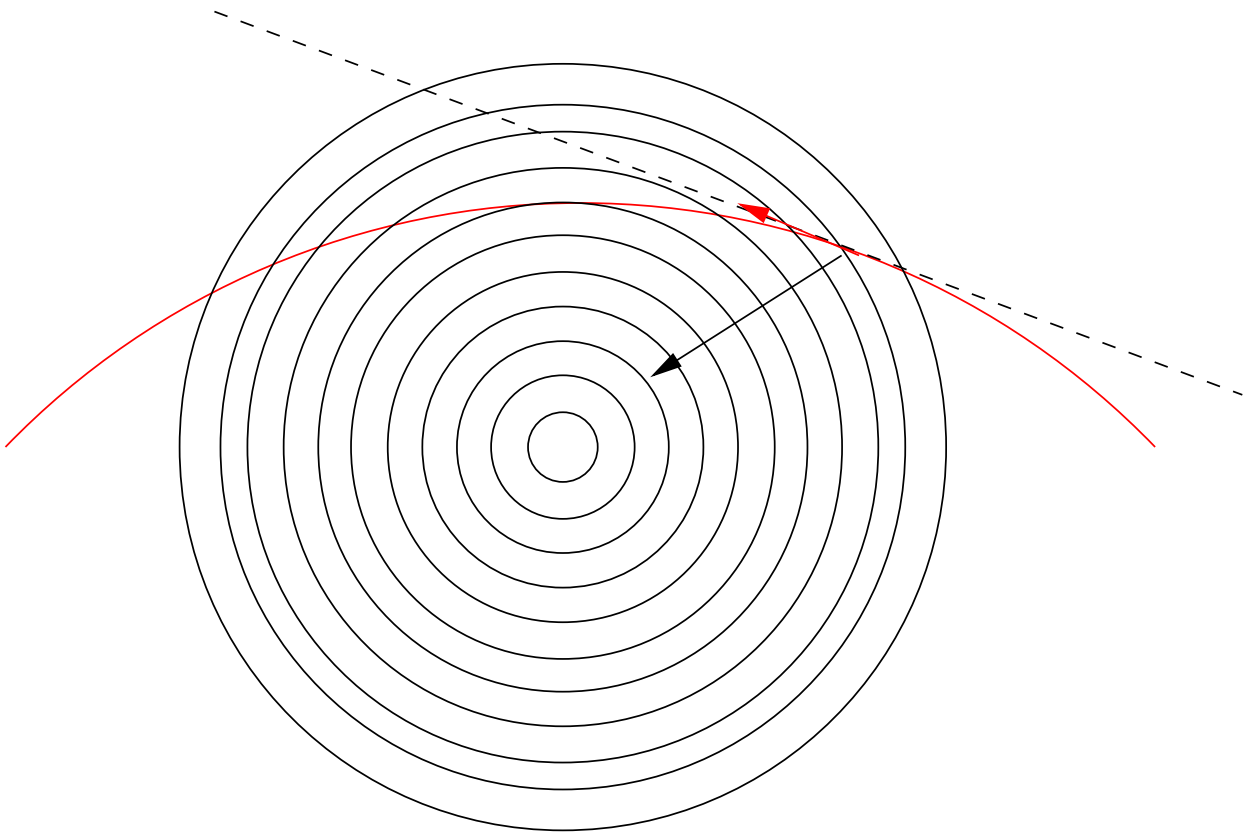
Minimizing $\frac{1}{2}\langle \mathbf{d}|A|\mathbf{d}\rangle$ under the constraint of a fixed length $\frac{1}{2}\langle \mathbf{d}|\mathbf{d}\rangle = const$ is by definition identical to finding the eigenvector associated to the lowest eigenvalue as can for instance be seen by adding the constraint with a Lagange parameter λ to the condition that the constrained gradient be zero:

$$A\mathbf{d} - \lambda\mathbf{d} = 0$$

The Lagrange equation is also the basis for a numerical minimization of the energy in Eq. 31. The gradient $\mathbf{g}$ is given by

$$\mathbf{g} = A\mathbf{d} - \frac{\langle \mathbf{d}|A|\mathbf{d}\rangle}{\langle \mathbf{d}|\mathbf{d}\rangle}\mathbf{d}$$

As usual, following the Langrange gradient conserves the constraint only to first order and therefore an explicit renormalization of the distance is required after each update in any minimization method.



Quasi Newton methods to find saddle points

We have seen how an approximative Hessian matrix can be build up from finite differences of the gradient vectors in a Quasi Newton minimization. We have also seen how the lowest curvature directions can be found by a dimer rotation without the need to calculate and diagonalize the Hessian matrix. These two ingredients can be combined to obtain a very efficient method to locate saddle points. In this method dimer rotations are alternatingly performed with translations of the dimer center. The gradient vector is decomposed in each iteration into the component parallel to the dimer and its orthogonal complement. The step size along the parallel direction is, in the usual way, given by the inverse curvature. Near a saddle point this curvature will be negative and in this way the parallel component will be inverted. If one is far away from the saddle point where the lowest curvature mode might be positive the parallel part is explicitly inverted to avoid ending up in a minimum. The orthogonal component is preconditioned by the approximate Hessian obtained during the previous iterations from the differences of the gradient vectors. If the dimer rotation was tight enough to find the exact lowest curvature direction and if the Quasi Newton Hessian was identical to the exact Hessian, this prescription would be identical to the operations given by Eq. 27,28 for the Newtom method. In practice both quantities are only approximate which is however sufficient in practice to obtain fast convergence at greatly reduced cost compared to a Newton method.

Noise in Quasi Newton methods

The standard formulas from calculus tell us that the derivative $f'(x)$ can be obtained as a finite difference of the neigbhouring functional values $f(x+h)$ and $f(x)$ and that the error is proportional to h, i.e.

$$f'(x) = (f(x+h) - f(x))/h + O(h)$$

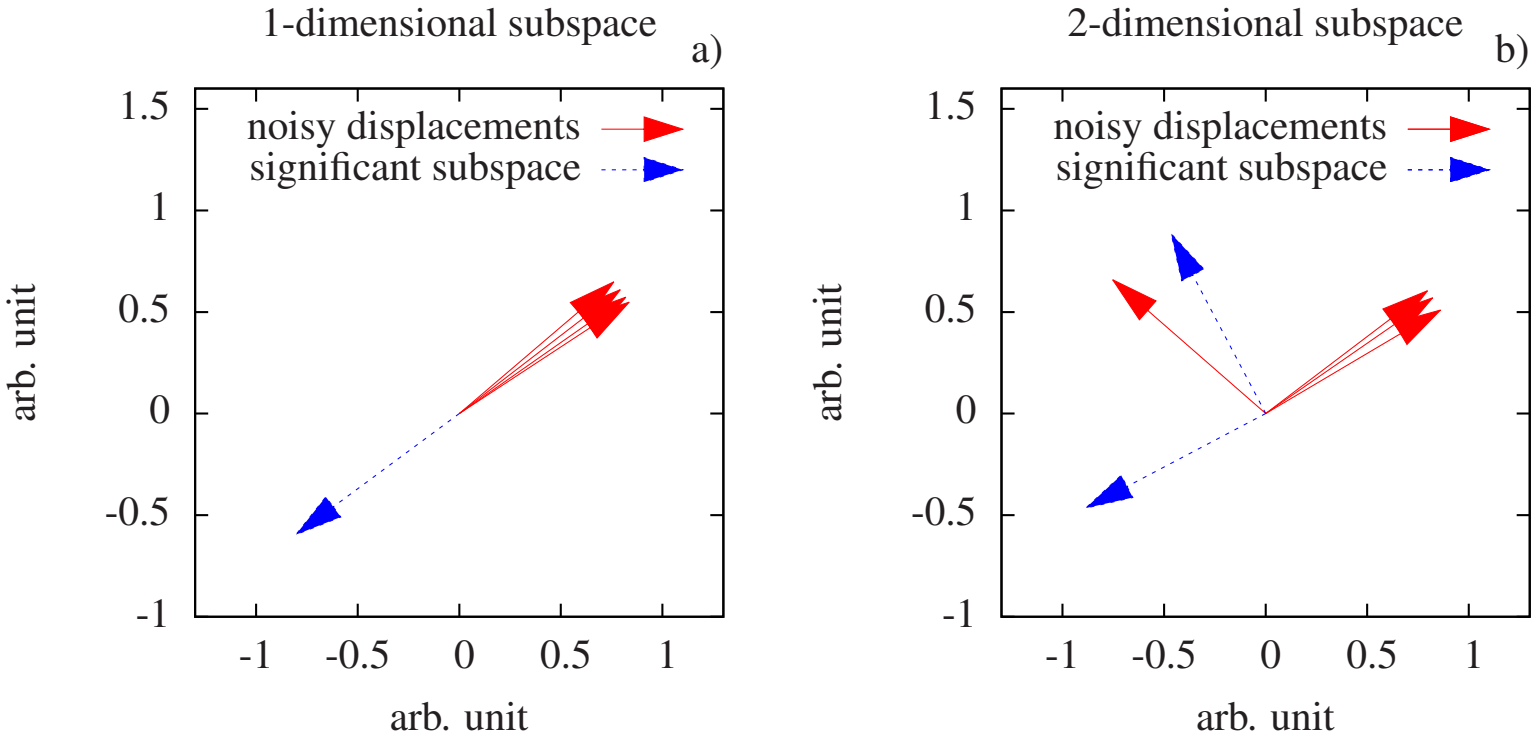
Whereas in the analytical formula the error from the finite difference decreases with decreasing h this is typically not the case in numerical work. If the function $f(x)$ is evaluated numerically it is always contaminated by noise, i.e the numerical evaluation does not return $f(x)$ but $f(x)+\sigma(x)$ where σ is the noise. Usually $\sigma(x)$ is much smaller than $f(x)$ and mo problems arise. If one take however finite differences of two very similar functional values (very small h in our example) problems can arise. I the difference in the functional values is much less than the noise level then

$$f(x+h)+\sigma(x+h) - f(x) - \sigma(x)) = \approx \sigma(x+h) - \sigma(x)$$

and the finite difference can give completely wrong values for the derivative. Exactly this scenario applies if Hessian curvature information is obtained in a Quasi Newton method from finite differences of the gradients close to convergence. In this case the gradient is approaching zero and the noise level can

be comparable or even larger than the norm of the gradients. Numerical optimization clearly becomes impossible and meaningless in the extreme case, where gradients are completely dominated by noise. The optimization proceudre should therefore contain some diagnostic tools that recognize when the noise level becomes so important that the optimization procedure should be stopped. The negative effect of a moderate noise level can however be attenuated by a noise elimination process as described in the following.

The figure below shows a set of vectors $\mathbf{v}_i$'s. The similar vectors represent noisy forces. So without noise they would be identical. Let us now consider



linear combinations of these vectors $\mathbf{v}_i$'s

$$\mathbf{w} = \sum_l c_l \mathbf{v}_l$$

under the constraint that

$$\sum_l c_l^2 = 1$$

If we take nearly identical vectors $\mathbf{v}_l$ that differ only by some noise, then we can obviously form a vector $\mathbf{w}$ that is very short. If on the other hand the vectors are different and in particular if they are orthogonal, we can not obtain a short vector by a linear combination of them. The longest and shortest vectors as well as vectors of intermediate length can be obtained in a mathematically rigourous way by diagonalizing the overlap matrix S given by

$$S_{i,j} = \langle v_i | v_j \rangle = \sum_k \mathbf{v}_i(k)\mathbf{v}_j(k)$$

The eigenvalues λ_k of the this matrix are the squared lengths of the linear combinations of the original set of vectors $\mathbf{v}_l$ and the corresponding eigenvectors $\mathbf{c}^k$ the expansion coefficients

$$\mathbf{w}_k = \sum_l c_l^k \mathbf{v}_l$$

with

$$\langle w_k | w_k \rangle = \lambda_k$$

So vectors $\mathbf{w}_k$ with short length are essentially linear combinations of noisy forces. By excluding these vectors on can generate a subspace of lower dimension. This subspace, shown by the blue vectors in the figure above, represents then the significant subspace within which a subspace Hessian can be build up with confidence using the Quasi Newton method.

Two sided saddle point mehods

The modified Newton method presented in the previous section assumes that one has an initial structure which, ideally, is close to a saddle point. It will then converge to this close by saddle point. Methods of this type are called one sided methods. Frequently one has however only two local minima and one has no good guess for an approximate saddle point. So-called two-sided methods can in this case find the saddle point connecting the two minima. A widely used method is the nudged eleastic band method. It is a refinement of the simple idea to span an elastic rubber band from one minimum to the other over the potential energy surface. The potential is evaluated along certain points $\mathbf{R}_i$ on the elastic band. These P points are called images. Mathematically this gives rise to an object function defined as

$$f(\mathbf{R}_0, \mathbf{R}_1, ..., \mathbf{R}_P) = \sum_{i=0}^{P} V(\mathbf{R}_i) + \sum_{i=1}^{P} \frac{k}{2} (\mathbf{R}_i - \mathbf{R}_{i-1})^2$$

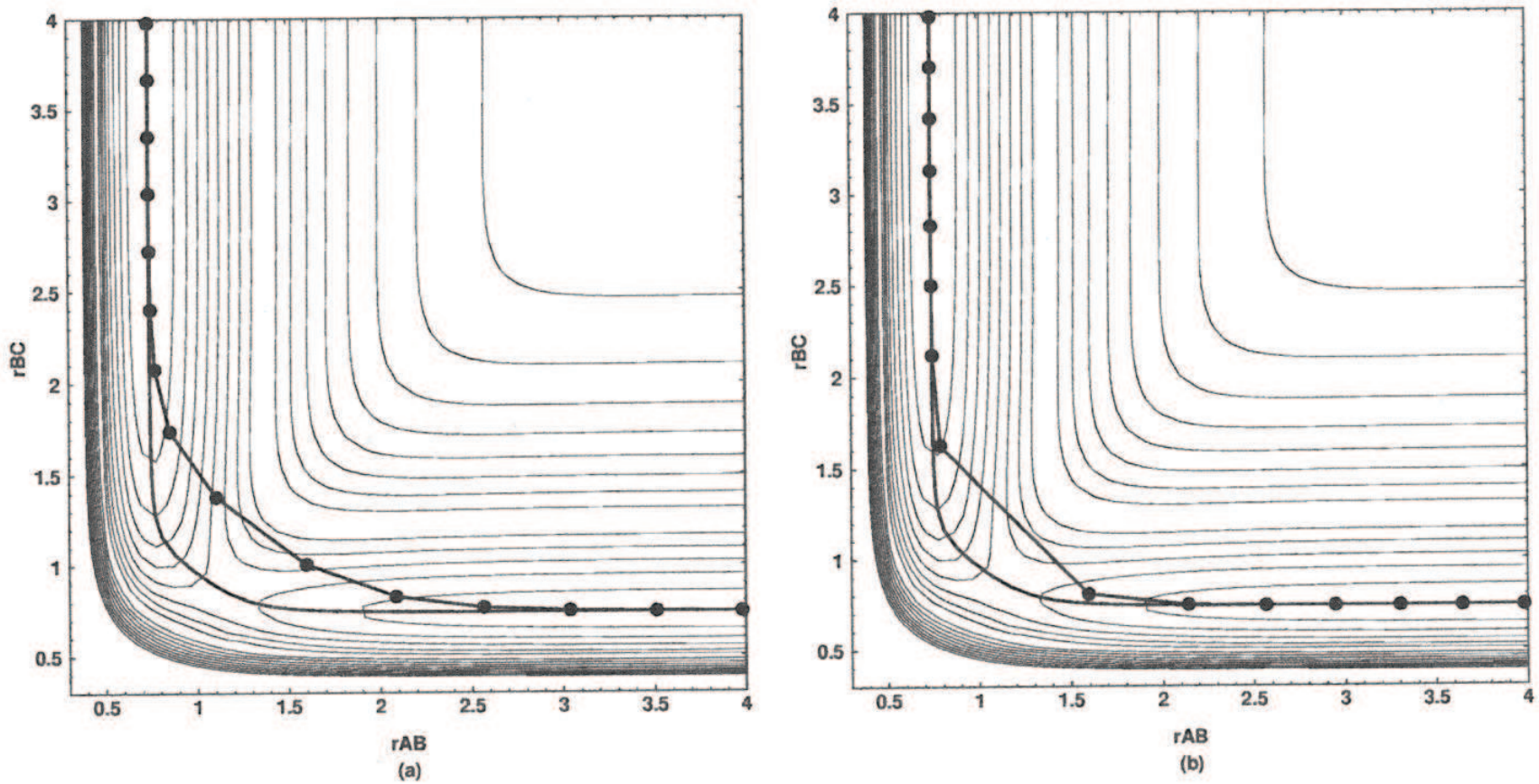
The force acting on image i is then given by

$$F_i = -\nabla V(\mathbf{R}_i) + \mathbf{F}_i^S$$

where

$$F_i^S = -k(\mathbf{R}_{i+1} - \mathbf{R}_i) + k(\mathbf{R}_i - \mathbf{R}_{i-1})$$

As shown in the figure below this method has a serious shortcoming. In the region close to the saddle point there is a corner cutting and so the saddle point is not on the band and significantly lower in energy than the highest point along the path. This corner cutting can be reduced if the spring constant *k* is reduced. But in this case the two images bracketing the saddle point will be pulled down too much by the action of physical force.



These problems can be cured in a simple way. From the physical force one uses only the component perpendicular to the band and from the elastic spring force F^s only the parallel component

$$\tilde{F}_i = -\nabla V(\mathbf{R}_i)_\perp + \mathbf{F}^s_{i\|}$$

where

$$\mathbf{F}^s_{i\|} = \mathbf{F}^s_i \cdot \hat{\mathbf{t}}\, \mathbf{t}$$

and

$$\nabla V(\mathbf{R}_i)_\perp = \nabla V(\mathbf{R}_i) - \nabla V(\mathbf{R}_i) \cdot \hat{\mathbf{t}}\, \mathbf{t}$$

$\hat{\mathbf{t}}$ is the unit tangent vector to the band. Obviuously this tangent vector is numerically only well defined if there are many images along the path, i.e if their distance is short in which case on can either put

$$\hat{\mathbf{t}} = \frac{\mathbf{R}_i - \mathbf{R}_{i-1}}{|\mathbf{R}_i - \mathbf{R}_{i-1}|}$$

or

$$\hat{\mathbf{t}} = \frac{\mathbf{R}_{i+1} - \mathbf{R}_i}{|\mathbf{R}_{i+1} - \mathbf{R}_i|}$$

The nudged elestic band method requires as an input guess all the images which specify the path along the band. This can actually be the most difficult

part in practice. One possibility is to do a simple linear interpolation between minimum A and B. However it is in principle unknown which atom of configuratioon A maps onto which atom of configuration B. If this mapping is not optimal very high energy path can result which miss the true saddle point. If if the mapping is correct a linear interpoation can also lead to some very high energy configuration where for instance some atoms come very close. This can therefore lead to convergence problems if the poetntail energy surface is calculated within density functional theory. As a matter of fact finding saddle points wth two sided methods is nowadays still a problem for which no methods exist which could do the job in a fully automatic way.

Global geometry optimization

The minimization methods discussed previously such as the steepest descent or conjugate gradient method can be used for local geometry optimizations. In a local geometry optimization you fall into the local minimum that is closest to your starting point. Because the condition number of the Hessian is frequently bad, the convergence of these methods can be slow, but nevertheless it is guarantueed that they will finally find a local minimum. Finding the global minimum of the total energy function is a much more complicated problem. There exists no algorithm that will find the global minimum with certainty within a computing time that grows less than exponentially with respect to the system size. Systematically exploring the high dimensional space is impossible in practice. Covering it with a grid of m points in each direction would require $m^{3N_{at}}$ grid points because the dimensionality of the Born-Oppenheimer surface of a system of N_{at} atoms is $3N_{at}$. In spite of the mentioned theoretical obstacles there are however algorithms that can find the global minimum for moderately complex systems within acceptable computing time. This is due to the fact that many local minima are frequently grouped together in funnel like structures (as discussed previously) where many algorithms rather easily 'fall' down in the minimum at the bottom of the funnel. If this minimum at the bottom is the global minimum one has succeded, if not the algorithm has to be able to climb out of this wrong funnel. This is the difficult part where many algorithms fail.

Genetic algorithms

Genetic or evolutionary algorithms try to mimic the Darwinistic evolution to solve global optimization problems. The principle is the survival of the 'fittest' and genetic algorithms are for instance using steps that are called mutations and crossovers. The basic quantity is a population of individuals that are represented by their genes. Numerically these genes are binary strings. A mutation consists of a random change of a gene, i.e. of a flip of one or several of the bits in a gene. It is thus similar to a trial step move in a Monte Carlo method. What is really new compared to Monte Carlo methods is the concept of gene crossovers. Given two genes of two individuals a crossover point is first determined at random and then the genes are combined as shown below to obtain a child.

1 0 0 1 1 1 1 0 0 1 'mother gene'
1 0 1 1 0 0 0 1 1 1 'father gene'

1 0 0 1 0 0 0 1 1 1 'child gene'

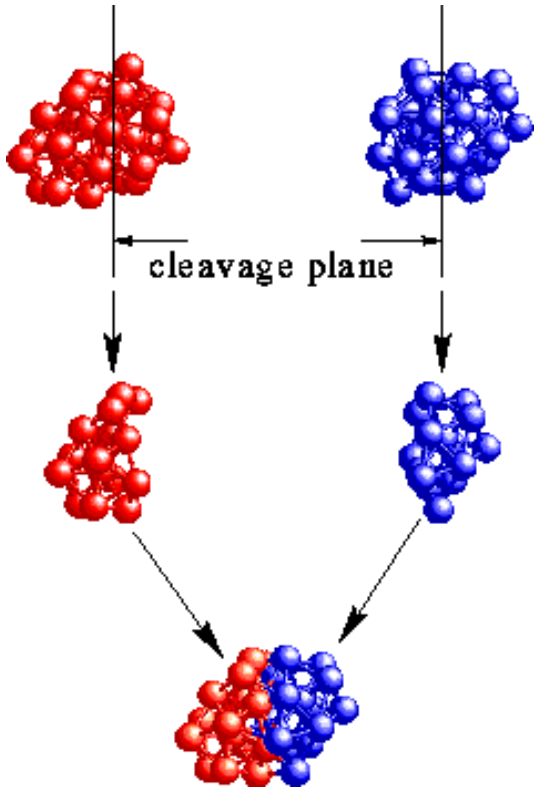
Gene crossing makes only sense if neighboring genes determine common functionalities. This can be easily seen by going back to biology. If for instance in the example above, the first 4 genes encode the functionality of ear and the last 6 the functionality of the eye, then the child has a certain chance having both good ears and a good eyes assuming that the mother had good ears and

the father good eyes. If however the first 5 genes determine the ear and the last five the eye then the above crossover after the fourth bit will very likely result in both ears and eyes that do not work very well.
After performing the operations of mutation and crossovers on a population comes the final survival step. The fitness of each individual i in a population that may consist of parents and children generated by both mutations and crossovers is measured by its fitness f_i which would be in a physical problem for instance the negative of the energy of a configuration. The average fitness of our population $< f >$ of N individuals is given by

$$< f >= \frac{1}{N} \sum_{i=1}^{N} f_i$$

The survival rate of an individual is then proportional to $f_i / < f >$.
Repeating the processes of mutation/crossover and survival gives fitter and fitter populations and the hope is that finally a population might contain a 'perfect' individual, which in the mathematical language would be the global maximum/minimum.
Applying standard genetic algorithms to structural optimization is problematic for several reasons. First, it is unnatural to represent atomic positions by short binary strings. A continuous problem is in this way mapped onto a discrete problem. Second it is not quite clear how to do the crossover in an efficient

way. In order to optimize the structure of clusters, people devised a crossover process that is not based on binary strings but simply combines the parts of two clusters as shown below.



Genetic algorithms based on this geometric crossover mechanism work well for clusters that have icosahedral ground states but are not applicable for finding more complicated ground state structures. For periodic structures it seems to be much easier to come up with efficient crossover and mutation moves and evolutionary algorithms work very well and have already allowed to discover many interesting new materials.

Simulated annealing

Simulated annealing is based on thermodynamics. At a sufficiently low temperature the system will be in the ground state, i.e. in the global minimum ε_0 of the potential energy surface since all other minima ε_i have a Boltzmann weight $\exp(-\beta(\varepsilon_i - \varepsilon_0))$ that is vanishingly small. The Eliminating noise in Quasi Newton methodimplex simplest implementation of simulated annealing is based on molecular dynamics. The system is propagated using Newton's equations of motion. Ergodicity ensures that the thermodynamic Boltzmann distribution is finally reached. Molecular dynamics based simulated annealing is thus imitating what is happening in nature during a crystallization process. While the system is slowly cooling down the atoms move according to Newton's law and find finally the global minimum, which is the perfect crystal structure. One is thus only left with setting up a prescription for the cooling rate. The simplest cooling recipe is just to impose an exponential decrease of the temperature. A template program implementing this simplest simulated annealing method is shown below. Some values (4.d0, .9999d0 etc) are just examples and other values may be more appropriate in other contexts. It has to be stressed that there is no guarantee that the global minimum will be obtained at the end of the run. Frequently the trajectory gets trapped in other local minima, i.e. it has not enough energy to go from the present minimum over a barrier into another local minimum. One should therefore do many runs with differ-

ent initial atomic positions and different parameters before one can state with a certain confidence that one has found the global minimum.

```
        read initial atomic positions and calculate initial forces

        ref_kin=4.d0
1000    continue
        ref_kin=ref_kin*.99999
        if (ref_kin.le.1.d-3) goto 2000

        DO A VELOCITY VERLET MD STEP AND CALCULATE THE KINETIC ENERGY act_kin

        if (act_kin.gt.ref_kin) then
             reduce velocities by a factor of .99d0
        else
             increase velocities by a factor of 1.01d0
        endif
        goto 1000
2000 continue

        write final atomic positions
```

The thermodynamic approach to the global minimum search

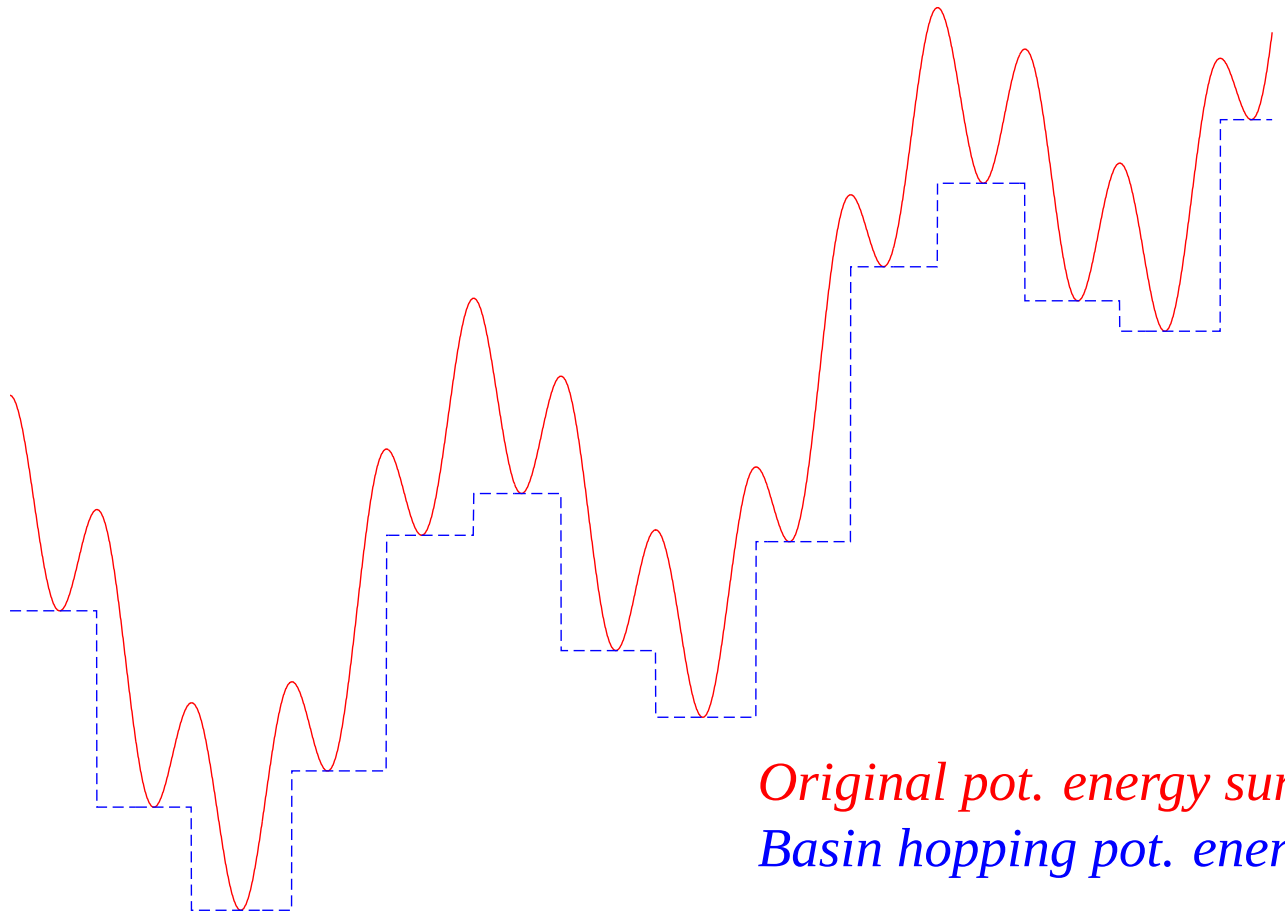
- Basic idea:
  A Monte Carlo algorithm is used to generate a thermodynamic Boltzmann distribution. At sufficiently low temperature the ground state configuration will be the dominant configuration.

- Problem:
  Thermodynamics does not tell us how fast the thermodynamic equilibrium distribution is obtained

  – Monte Carlo works fine for funnel like landscapes
  – Monte Carlo is not efficient if the landscape consists of several funnels: system remains trapped in a 'wrong' funnel

Trapping in Monte Carlo

Because of the Boltzmann factor $\exp(-\beta(E_{new} - E_{current}))$ all moves that try to escape from a basin or (wrong) funnel are rejected at low temperatures



Original pot. energy surface

Basin hopping pot. energy surface

Minima hopping algorithm

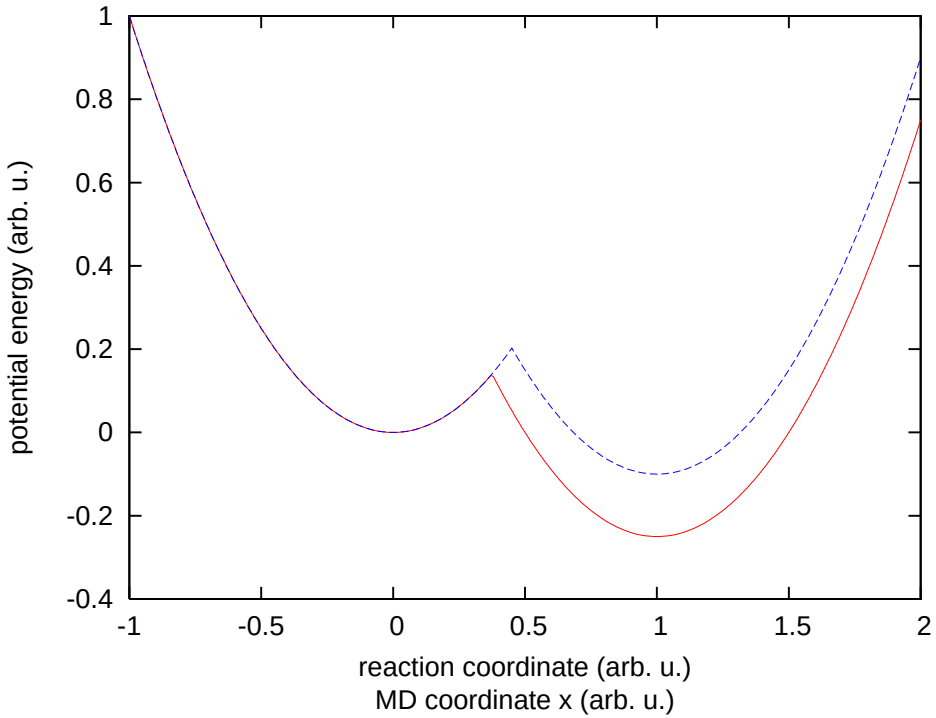
The minima hopping algorithm combines the local geometry optimizations applied in several other methods such as in genetic algorithms or basin hopping with the molecular dynamics based moves found in simulated annealing. Because both local geometry optimizations and molecular dynamics can be performed for nearly any system is is applicable to virtually any solid state system such as molecules, cluster, proteins and crystalline solids. Two basic features are mainly responsible for its high efficiency:

- It exploits the Bell-Evans-Polyani principle, i.e. it moves from one minimum to the next by crossing low barriers.
- By a build in feed back mechanism trapping in some region of the configurational space is excluded. Hence the algorithm does not explore again and again known configurational regions but explores new regions instead.

```
        initialize a current minimum 'Mcurrent'

        ESCAPE  TRIAL PART
MDstart: start a MD trajectory with kinetic energy Ekinetic from current minimum
        'Mcurrent'. Once potential energy reaches another minimum along the trajectory
         stop MD and optimize geometry to find the closest local minimum 'M'

         if ('M' equals 'Mcurrent') then
            Ekinetic = Ekinetic*beta1  (beta1 > 1)
            goto MDstart
         else if ('M' equals  a minimum visited previously) then
            Ekinetic = Ekinetic*beta2  (beta2 > 1)
            goto MDstart
         else if ('M' equals  a new minimum ) then
            Ekinetic = Ekinetic*beta3  (beta3 < 1)
         endif

       DOWNWARD PREFERENCE PART
         if ( energy('M') - energy('Mcurrent') < Ediff ) then
            accept new minimum: 'Mcurrent' = 'M'
            add 'Mcurrent' to history list
            Ediff = Ediff*alpha1 (alpha1 < 1)
         else if rejected
            Ediff = Ediff*alpha2 (alpha2 > 1)
         endif

         goto MDstart
```

Significance of the Bell-Evans-Polanyi (BEP) principle for global optimization

The BEP principle comes originally from chemistry, where it states that highly exothermic chemical reactions have a low activation energy. In the context of global optimization it implies that low barrier escape paths lead on the average into lower local minima than high barrier escape paths. This basic tendency can be seen from the figure below where the PES along the reaction pathway is represented by two parabolas. Pulling down the blue parabola will also lower the saddle point represented by the intersection ofthe two parabolas.

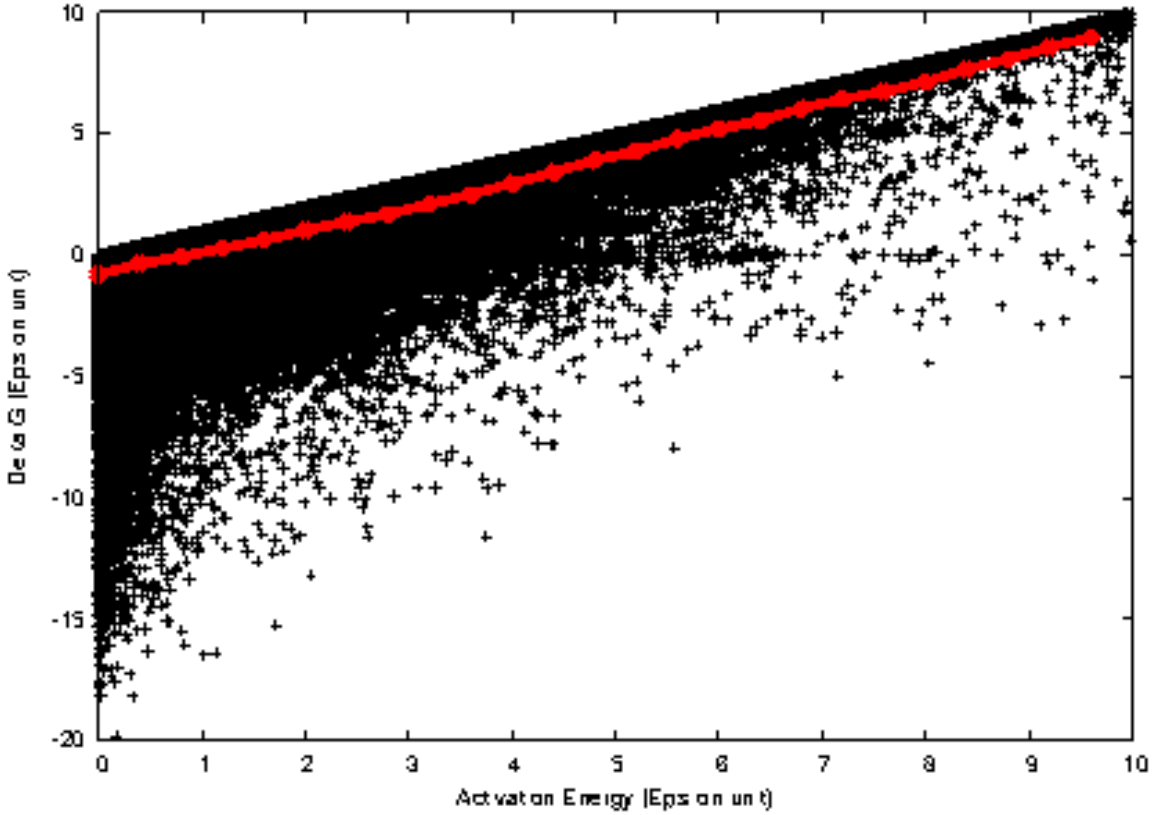


$E(x) = min\left(x^2, (x-1)^2 - .1\right)$

$E(x) = min\left(x^2, (x-1)^2 - .25\right)$

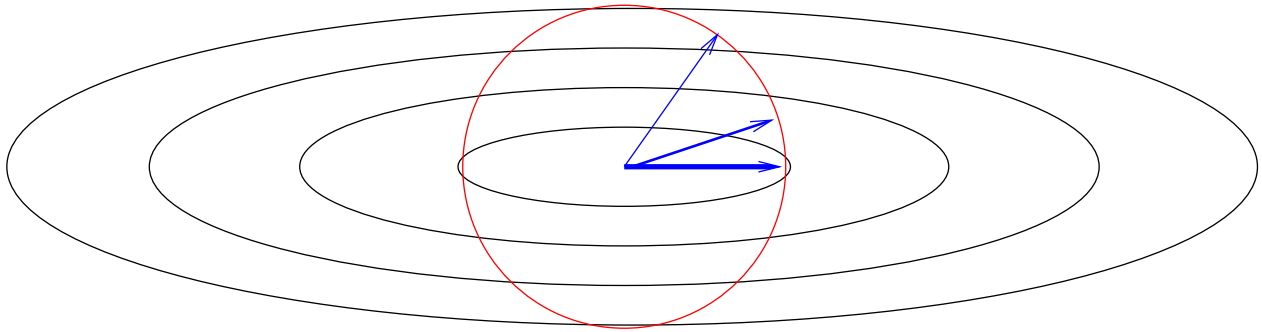
Numerical verification of the BEP principle for a LJ cluster

The pictorial arguments put forward to establish the validity of the BEP principle are of course approximate and it is to be expected that there are exceptions. The numerical test of the validity of the BEP principle in the figure below shows indeed that it can be strongly violated in certain cases as can be seen from the strong scattering of the black crosses in the plot of activiation energy (barrier height) vs the energy difference between the two connected minima. If one takes however averages in this huge data set of 30 000 sadlle points one obtaind the red line which indicates a virtually perfect validity of the BEP principle on average. This validity on average is sufficient in the context of global minimization.



Optimal MD trajectories

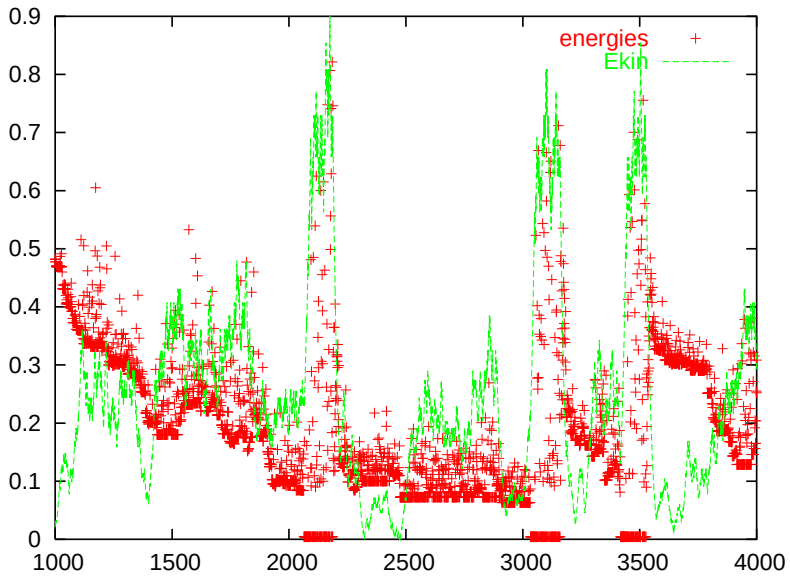
MD trajectories that start out in ”soft” directions lead fastest over a low energy saddle point into another minimum
Soft directions are found by minimizing the energy of a second image system under the constraint that it has a fixed distance from the local minimum.

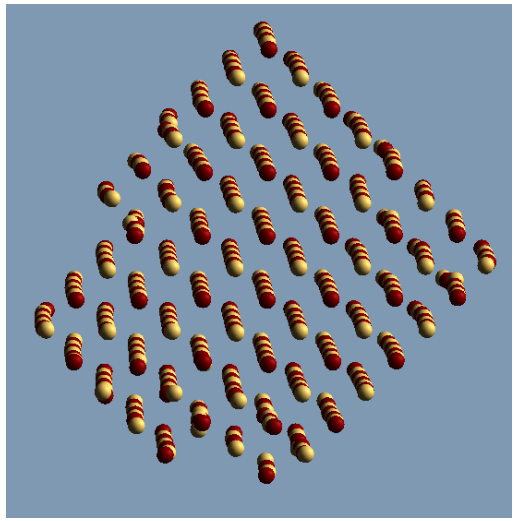


Empirical correlations between the curvature along the initial direction of the MD trajectory and the bqarrierheight overcome by the trajectory can be found:
Low curvature leads on average to low barriers

Example: minima hopping for a 512 atom NaCl cluster

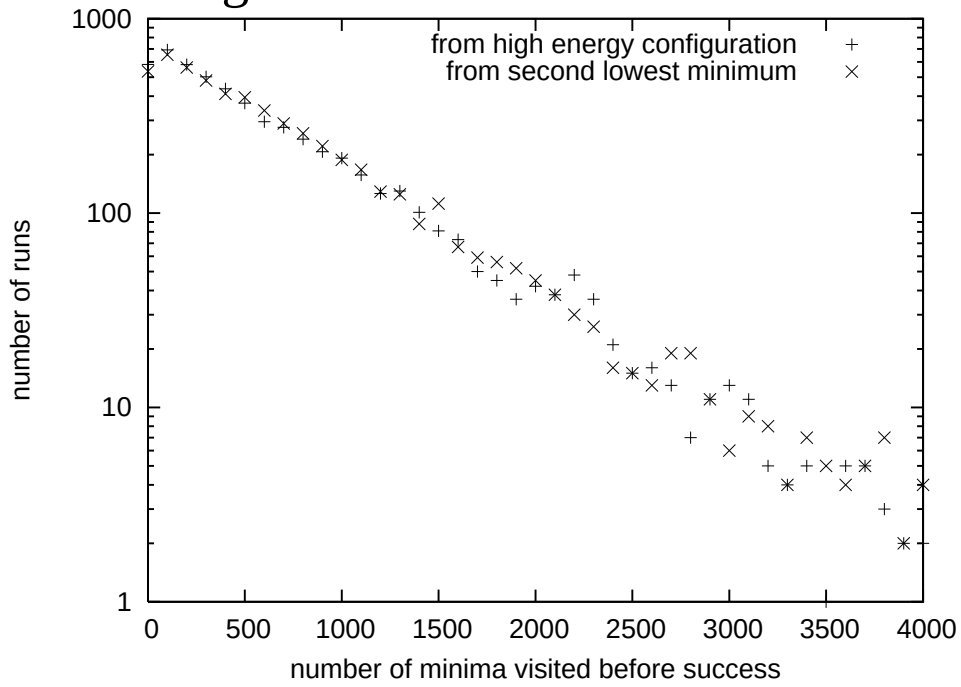
This cluster is well suited to illustrate how minima hopping works in practice. In addition to the ground state funnel associated to the global minimum which is a 8 by 8 by 8 cube the system has other funnels which correspond to other orthorombic shapes such as the 7 by 8 by 9 structure shwon below on the right. These other orthorombic shapes can of course not be build perfectly since the number of atoms does not match. The figure below on the left shows the energies of the various minima visited during a minima hopping run together with the kinetic energy of the molecular dynamics trajectory. It can be seen that the kinetic energy always increases strongly if the system gets trapped in a funnel which then allows to escape from the funnel. The escape from a funnel can thus be considered as some kind of melting event, whereas the system freezes when it goes down within a funnel.





Distribution of search lengths

All the methods for global optimization discussed in this course are stochastic. As a consequence the time needed to find the global minimum varies from one run to the next. Such a distribution of search lengths of a minima hopping run is shown in the figure below. In the case of minima hopping, random numbers are used to generate the initial velocity vector in the molecular dynamics escape step. For a single funnel system the distribution is exponential. For an exponential distribution one serial long search is equally effective as several short parallel searches whose summed length is equal to the length of the serial search. For multifunnel systems the distribution is a superposition of several exponential functions. In such a case several shorter parallel searches are more efficient than a long serial search.



The BigDFT software package

Solves the many-electron Schrödinger equation in the Kohn-Sham density functional approximation using a wavelet basis
This basis set combines the advantages of

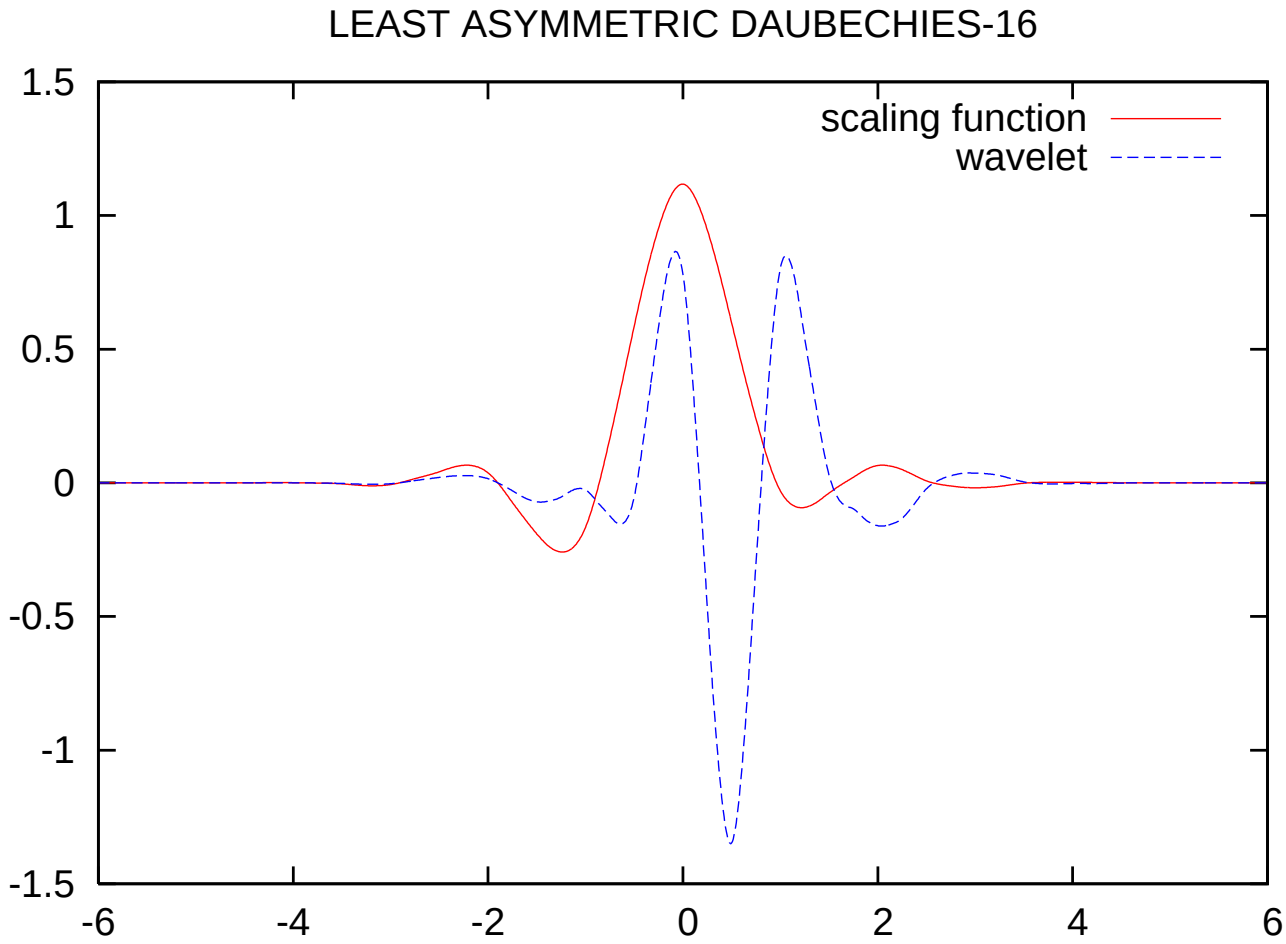
Plane waves:

- Systematic, orthogonal basis set
- Localization in Fourier space allows for efficient preconditioning techniques.

Gaussians:

- Localized in real space: well suited for molecules and other open structures.
- Adaptivity

High order Daubechies wavelets are used to represent wavefunctions



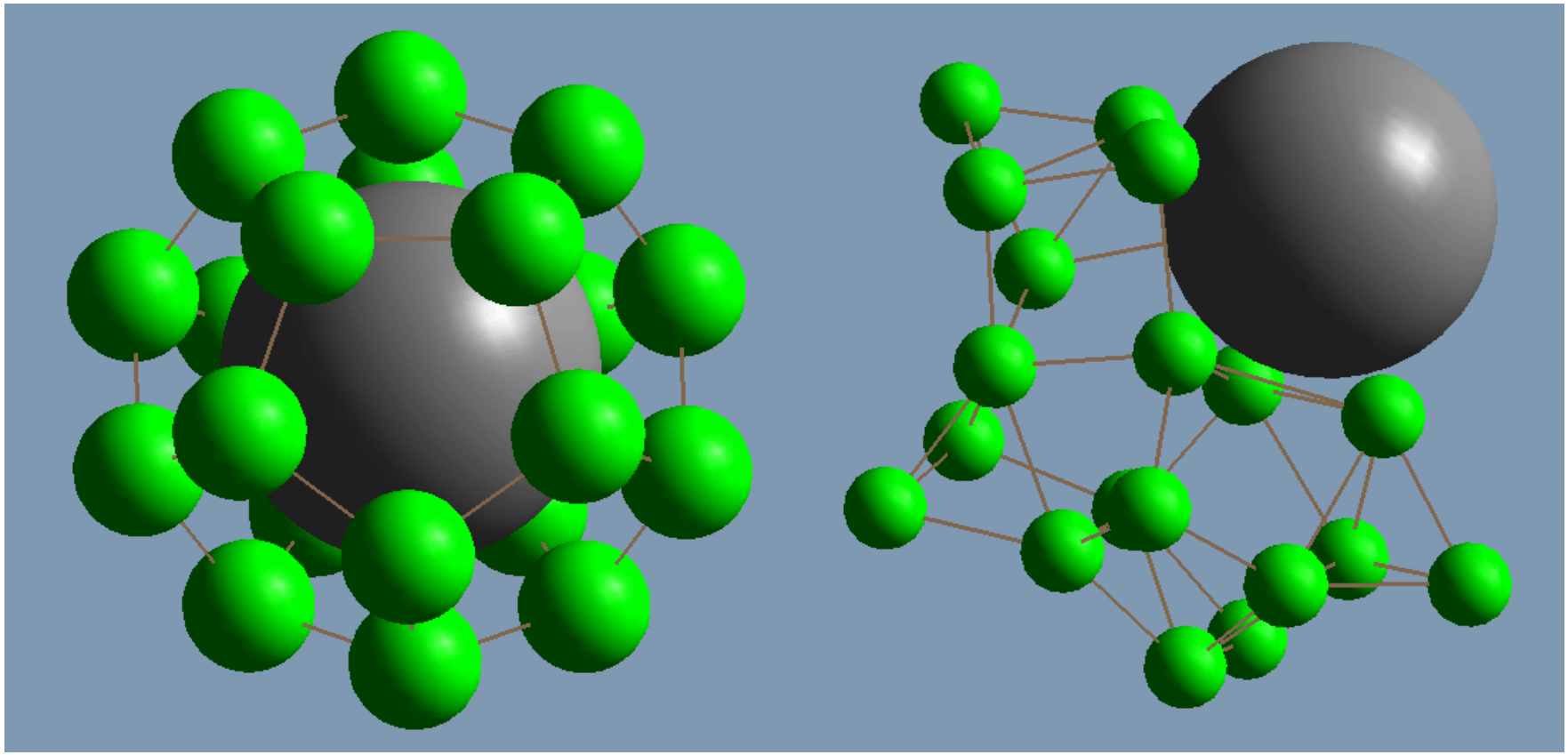
Application of minima hopping to the formation of C_{60}

Visualization at: http://www.unibas.ch/comphys/comphys/a.gif

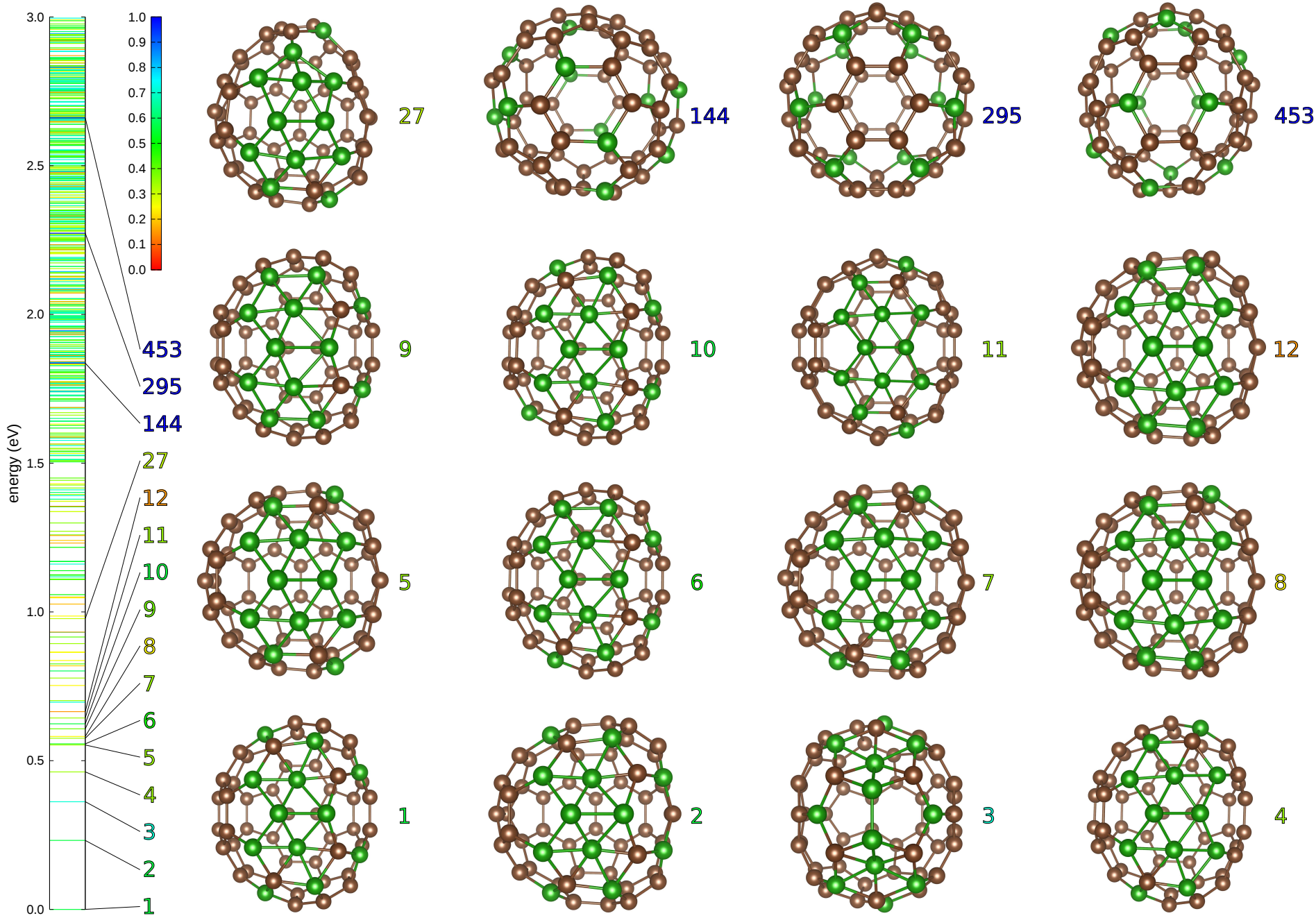
Application of dual minima hopping to metal doped silicon clusters

Fullerene like endohedrally doped Si_{20} cages are only metastable configurations but not the ground state

Fullerene ground state

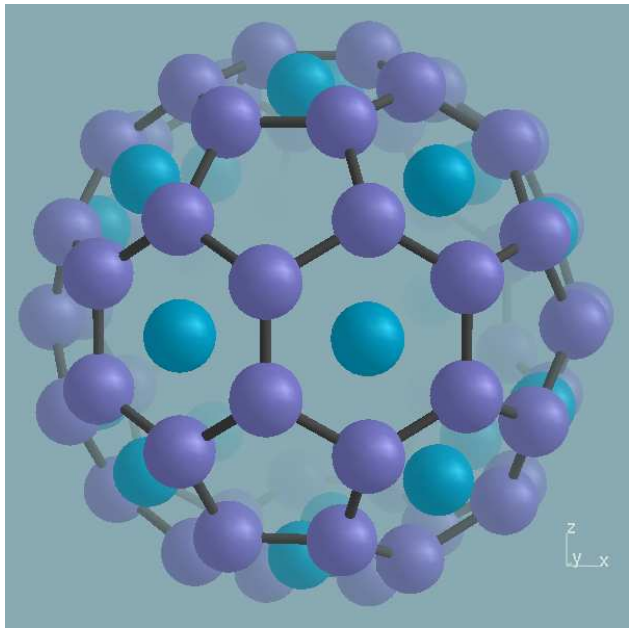


Application of minima hopping to heterofullerenes

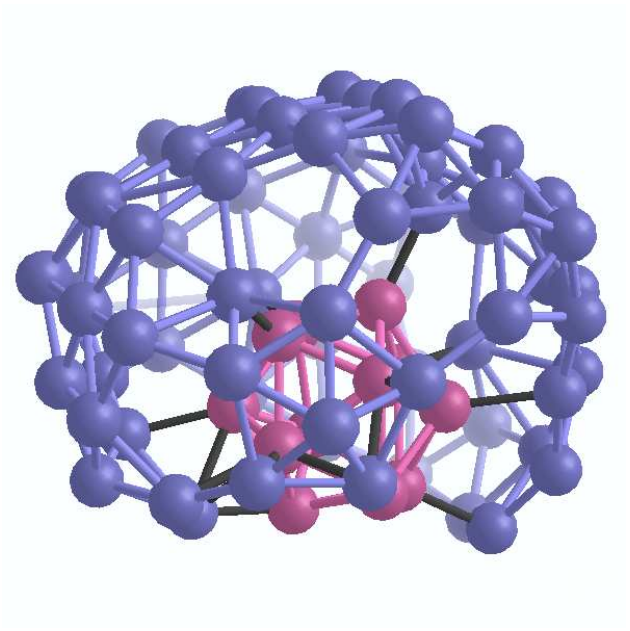


The ground state of B_{80}

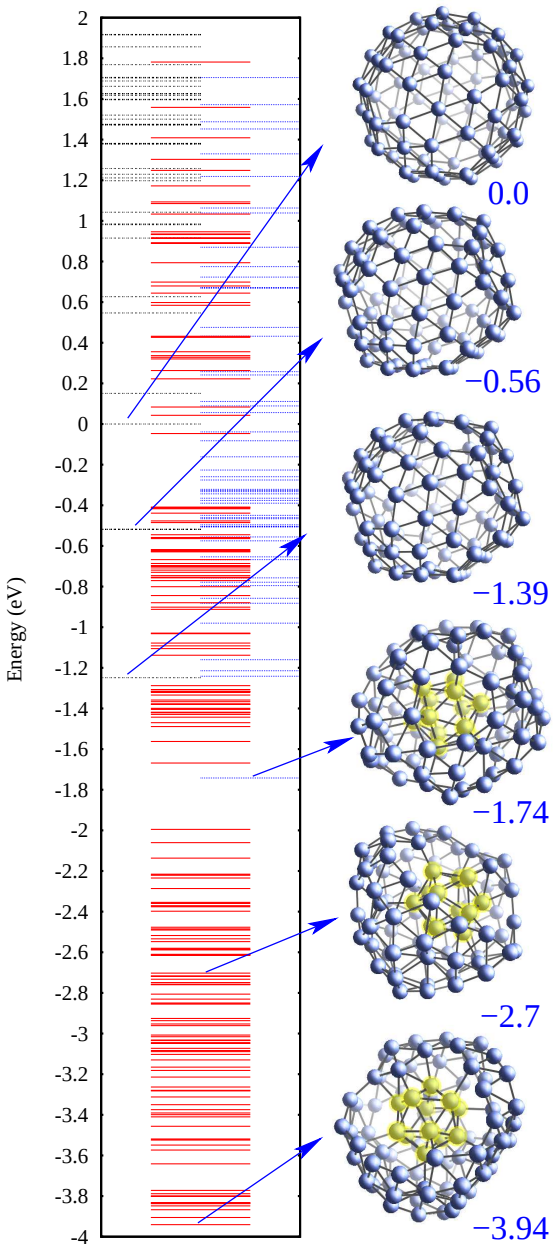
Szwacki B_{80} fullerene

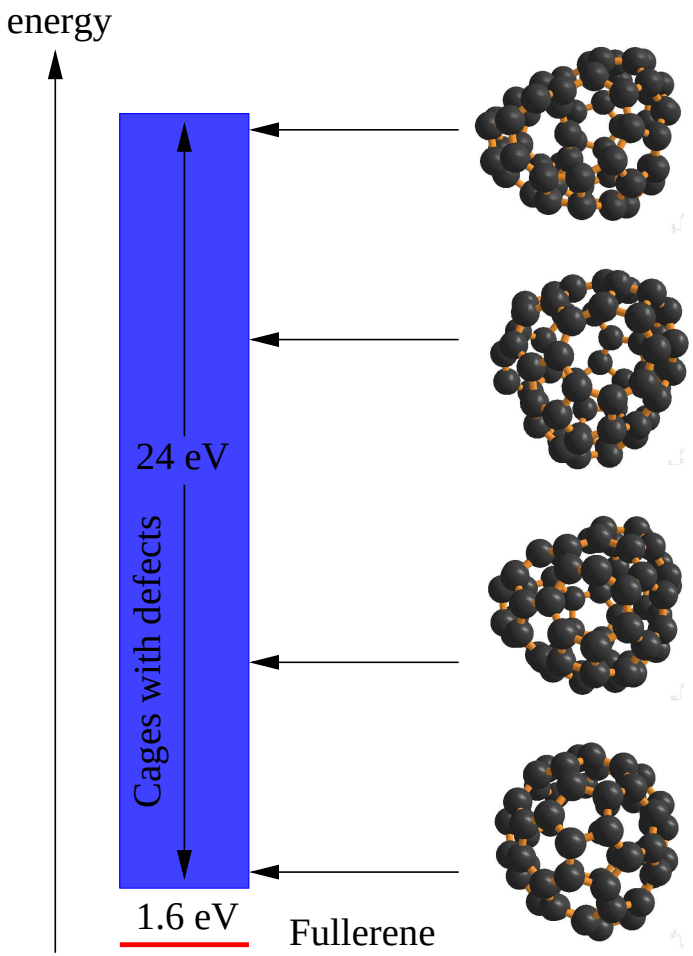


B_{80} ground state

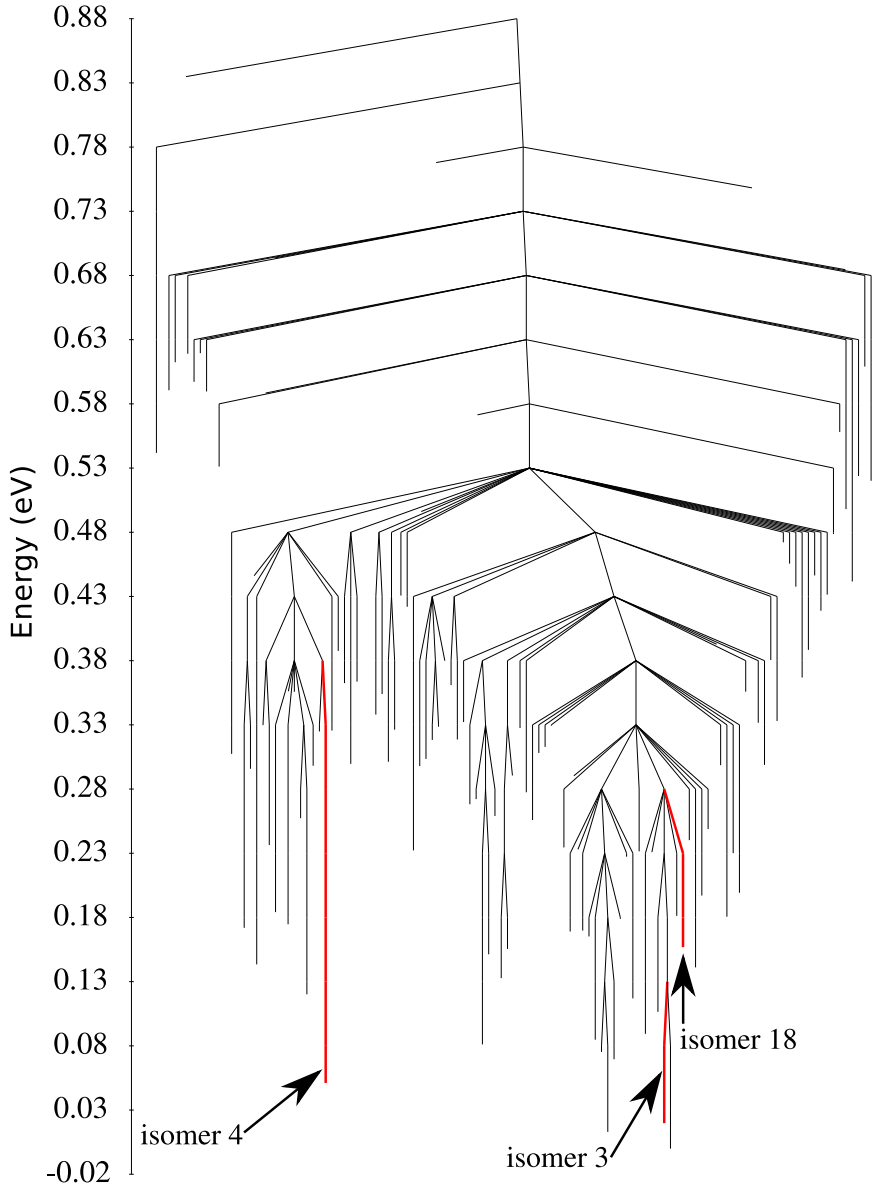


The configurational energy spectrum of C_{60} and B_{80}

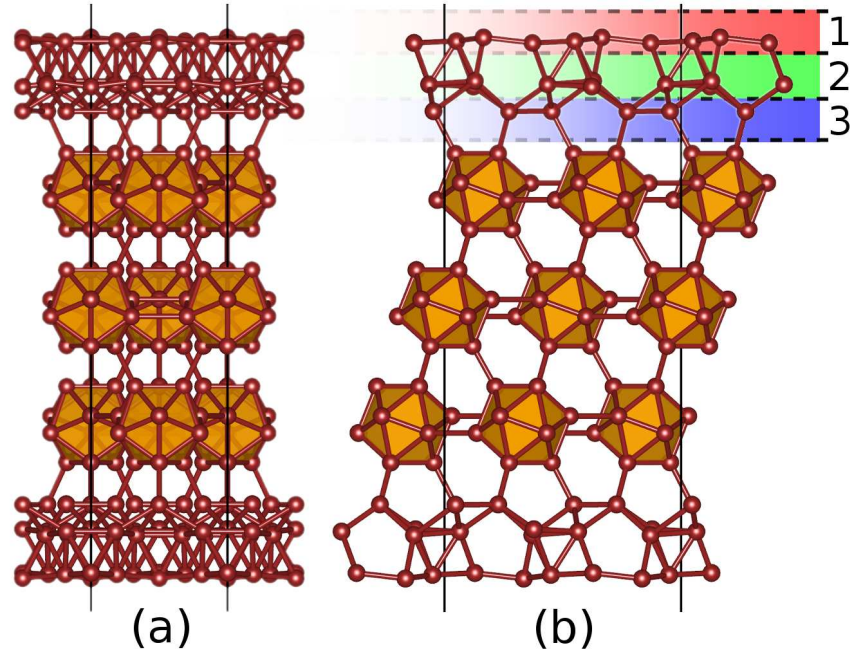


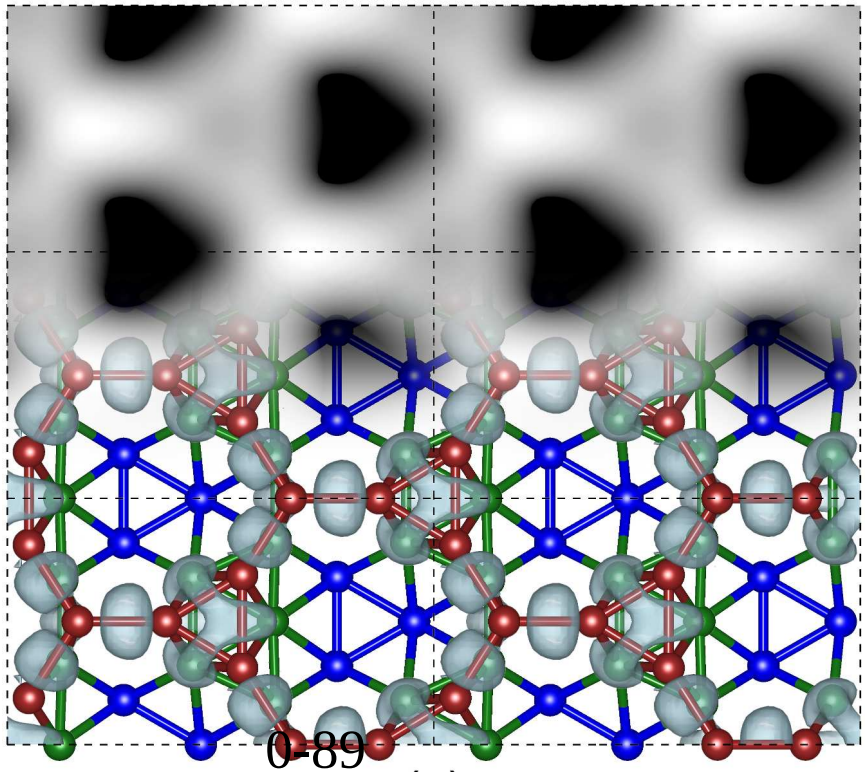


Disconnectivity graphs for the Au_{26} cluster



(111) surface reconstruction of boron leads to sheets

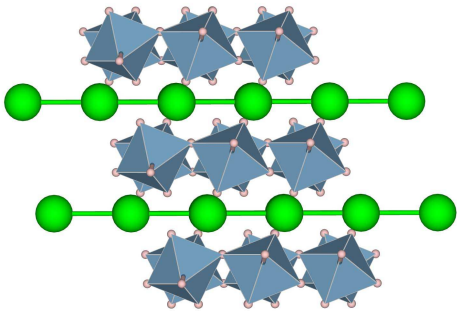




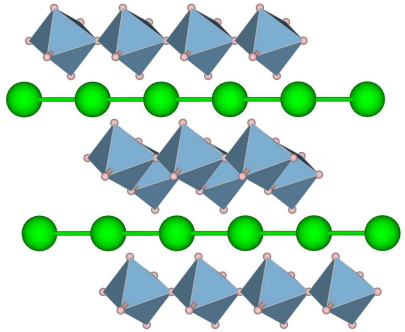
0-89

Application of minima hopping to alanates

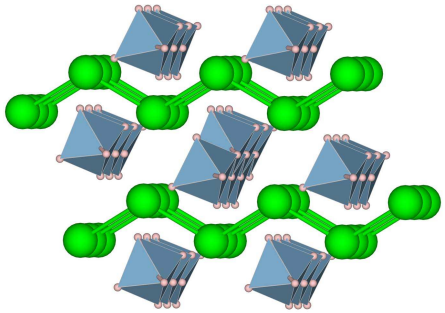
Ground state is polymer like



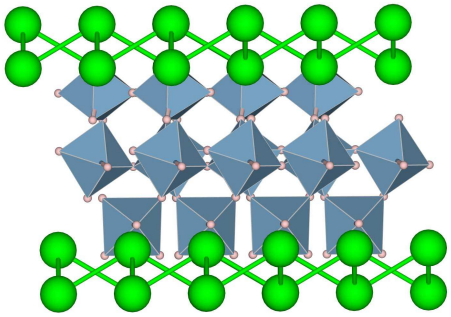
$LiAlH_4$, p-$P2_1/c$ phase



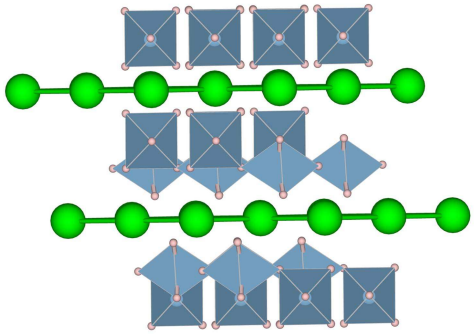
$NaAlH_4$, C2/m phase



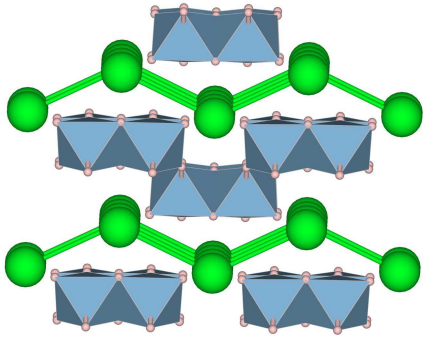
$KAlH_4$, P-1 phase



$Mg(AlH_4)_2$, $P2_1$ phase

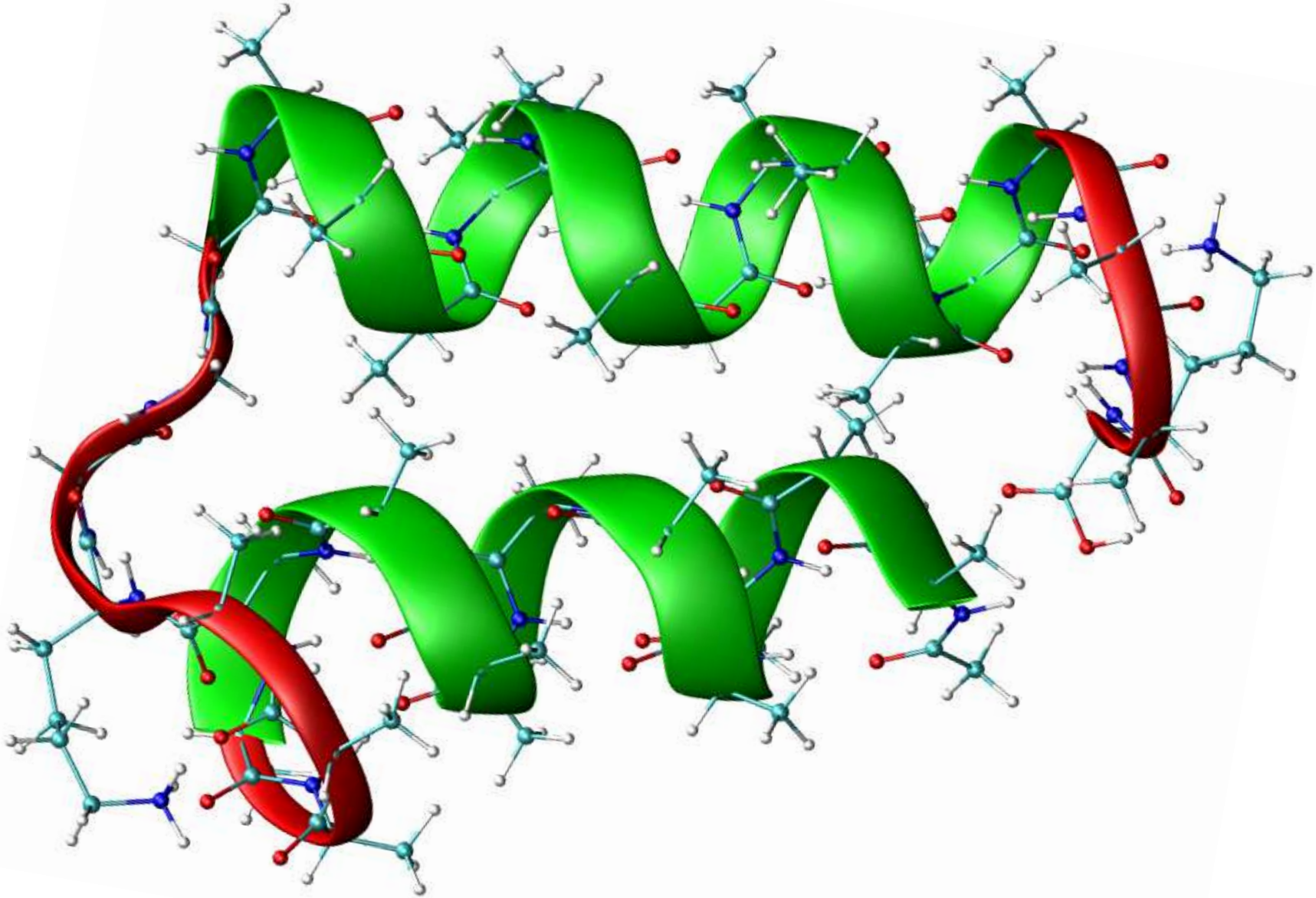


$Ca(AlH_4)_2$, $P2_1/c$ phase



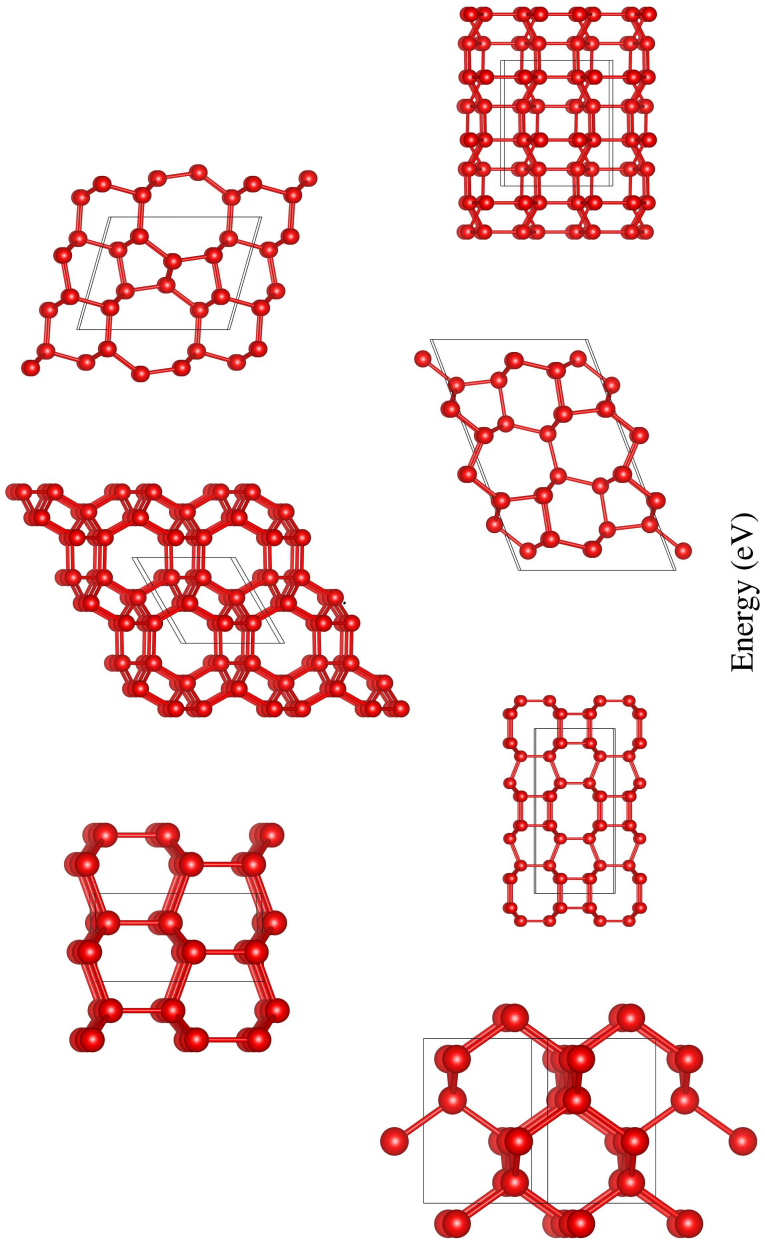
$Sr(AlH_4)_2$, Pm phase

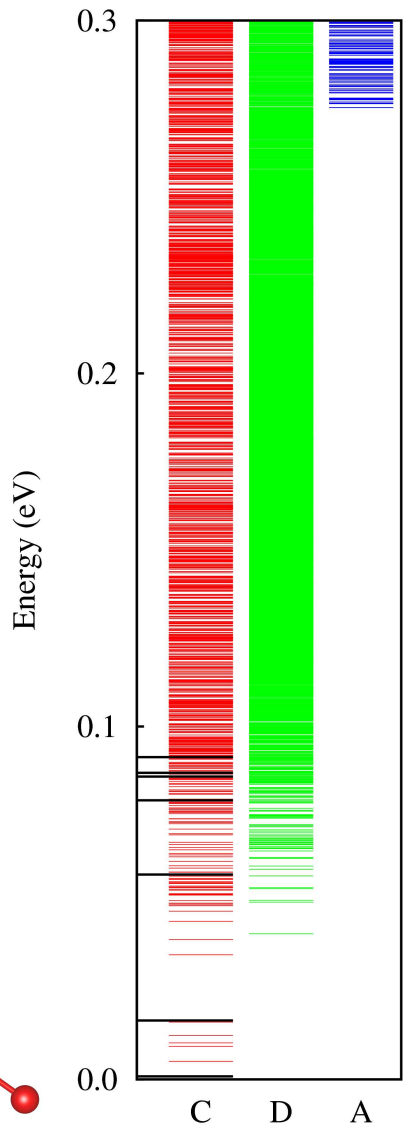
Application of minima hopping to protein folding

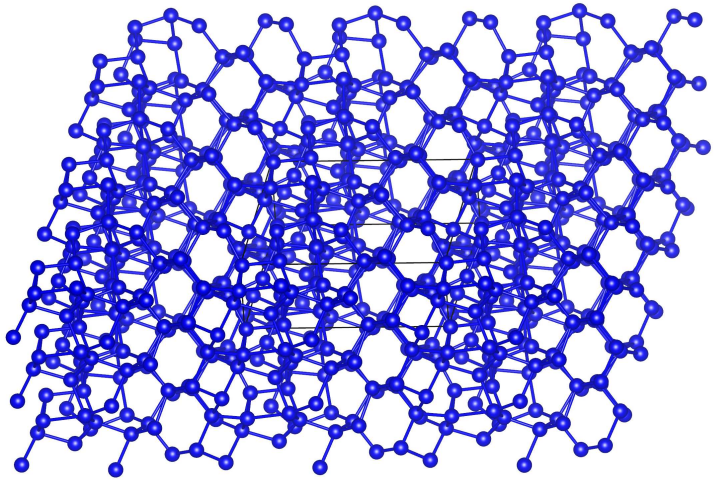


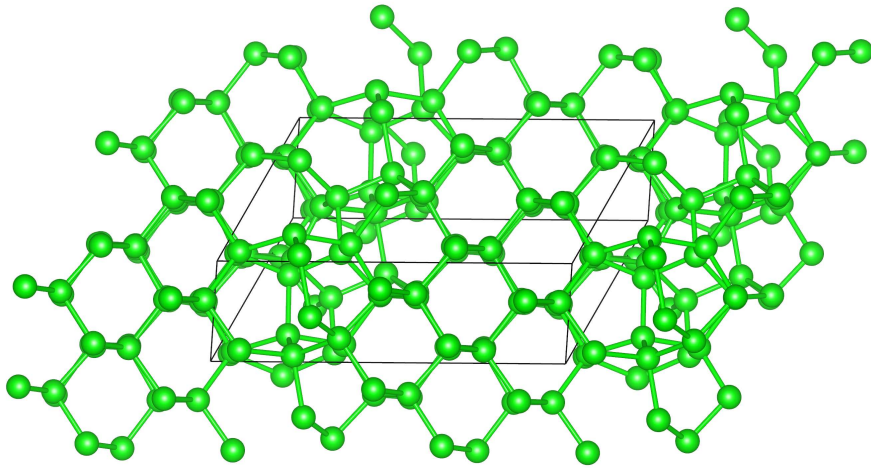
Low energy crystaline structures of silicon

Many crystalline structures exist within a small energy interval









Quantifying similarities between structures

Quantifying similarities between structures is required in many contexts in atomistic simulations. The most elementary question is whether two structures are identical or not. This question is far more complicated than one might think. Even if two structures are absolutely identical one structure will in most cases be rotated and translated with respect to the other one. For larger system it is in such a case in general impossible to detect by eye whether the structures are identical or not. In addition the two structures are in general not absolutely identical but only nearly identical because of the presence of noise coming either from an experiment or a simulation. In the case of experiment, two identical structures can for instance be measured by two different experimental setups which will lead to slightly different results. In the case of a computer simulation we have learned that a geometry optimization is stopped if the force norm is below a certain non-zero threshold. Therefore the structures are never perfectly relaxed and two structures that would become identical if the relaxation was continued until the forces are really zero, will in practice be slightly different. Quantifying similarities in terms of a distance is also very important for various classification methods such as clustering. In an clustering process on puts together similar structure that have a small configurational distance amomg each other into a so-called cluster.

Properties of a metric

It is natural to characterize the dissimilarity between two structures p and q by a real number $d(p,q) \geq 0$. In order to give meaningful results $d(p,q)$ should satisfy the properties of a metric, namely

- coincidence axiom: $d(p,q) = 0$ if and only if $p \equiv q$,
- symmetry: $d(p,q) = d(q,p)$,
- triangle inequality: $d(p,q) + d(q,r) \geq d(p,r)$.

The coincidence axiom ensures that two configurations p and q are identical if their distance is zero. The triangle inequality is essential for clustering algorithms. If it is not satisfied, then it could happen that a configuration that is within one cluster is also part of another cluster even though the distance between the two clusters is very large in configuration space.

The RMSD

A configuration of n atoms is represented by $R \equiv (r_1, r_2, \ldots, r_n) \in \mathbb{R}^{3\times n}$, where the column vector r_i represents the Cartesian coordinates of atom i. The root mean square deviation (RMSD), a distance based on the simple Frobenius norm,

$$\|R^p - R^q\| = \Big(\sum_{i=1}^{n} \|r_i^p - r_i^q\|^2\Big)^{1/2} \tag{32}$$

can not be used to compare two configurations p and q, because it is not invariant with respect to translations or rotations of one configuration relative to the other. For this reason the commonly used root-mean-square distance (RMSD) is defined as the minimum Frobenius distance over all translations and rotations. By minimizing $\sum_i^n \|r_i^p + d - r_i^q\|^2$ with respect to the translation vector d one obtains $\sum_i^n (r_i^p + d - r_i^q) = 0$. Therefore we will assume in the following that all r_i are measured with respect to the centroids of the corresponding configuration which allows us to drop the minimization with respect to the translation d. Then, finding the rotation U around the common centroid which minimizes

$$RMSD_l(p,q) = \frac{1}{\sqrt{n}} \min_U \|R^p - U R^q\| \tag{33}$$

is a local minimization problem and hence we denote this version of the RMSD by RMSD_l. The use of quaternions allows to find the optimal rotation in single step extremely rapidly.

Quaternions

A quaternion $\mathcal{Q} = (Q_0, Q_1, Q_2, Q_3)$ is an extension of the idea of complex numbers to one real (Q_0) and three imaginary parts. According to the Euler's rotation theorem, a rotation in space which keeps one point on the rigid body (centroid in our case) fixed, can be represented by four real numbers: one for the rotation angle and three for the rotation axis (we assume that the center of rotation is on the origin). A unit quaternion, i.e. $\|\mathcal{Q}\|^2 = Q_0^2 + Q_1^2 + Q_3^2 + Q_4^2 = 1$, can conveniently represent this axis-angle pair as

$$\mathcal{Q} = \left(\cos\left(\frac{\theta}{2}\right), \hat{\mathbf{u}}\sin\left(\frac{\theta}{2}\right)\right)$$

where θ is the rotation angle around the unit axis $\hat{\mathbf{u}} = a\hat{\mathbf{i}} + b\hat{\mathbf{j}} + c\hat{\mathbf{k}}$. The corresponding orthogonal rotation matrix is

$$\mathcal{U} = \begin{bmatrix} Q_0^2 + Q_1^2 - Q_2^2 - Q_3^2 & 2Q_1Q_2 - 2Q_0Q_3 & 2Q_1Q_3 + 2Q_0Q_2 \\ 2Q_1Q_2 + 2Q_0Q_3 & Q_0^2 - Q_1^2 + Q_2^2 - Q_3^2 & 2Q_2Q_3 - 2Q_0Q_1 \\ 2Q_1Q_3 - 2Q_0Q_2 & 2Q_2Q_3 + 2Q_0Q_1 & Q_0^2 - Q_1^2 - Q_2^2 + Q_3^2 \end{bmatrix}.$$

Assume that $\mathcal{U}$ is the 3×3 equivalence of the rotation operator U in Eq. 33. The optimum rotation $\mathcal{U}$ which minimizes RMSD, indeed maximizes the correlation between R^p and R^q, i.e. the atomic Cartesian coordinates with respect to the common center of mass. Based on quaternions, the optimum $\mathcal{U}$ is given by

Q which is identical to the principal eigenvector of the 4×4 symmetric, traceless matrix

$$\mathcal{F} = \begin{bmatrix} \mathcal{R}_{xx}+\mathcal{R}_{yy}+\mathcal{R}_{zz} & \mathcal{R}_{yz}-\mathcal{R}_{zy} & \mathcal{R}_{zx}-\mathcal{R}_{xz} & \mathcal{R}_{xy}-\mathcal{R}_{yx} \\ \mathcal{R}_{yz}-\mathcal{R}_{zy} & \mathcal{R}_{xx}-\mathcal{R}_{yy}-\mathcal{R}_{zz} & \mathcal{R}_{xy}+\mathcal{R}_{yx} & \mathcal{R}_{xz}+\mathcal{R}_{zx} \\ \mathcal{R}_{zx}-\mathcal{R}_{xz} & \mathcal{R}_{xy}+\mathcal{R}_{yx} & -\mathcal{R}_{xx}+\mathcal{R}_{yy}-\mathcal{R}_{zz} & \mathcal{R}_{yz}+\mathcal{R}_{zy} \\ \mathcal{R}_{xy}-\mathcal{R}_{yx} & \mathcal{R}_{xz}+\mathcal{R}_{zx} & \mathcal{R}_{yz}+\mathcal{R}_{zy} & -\mathcal{R}_{xx}-\mathcal{R}_{yy}+\mathcal{R}_{zz} \end{bmatrix}$$

where $\mathcal{R}$ is the correlation matrix whose elements are given by $\mathcal{R}_{xy} = \sum_i^n x_i^p y_i^q$. Note that, without applying U on R^q, Eq. (33) is given by

$$RMSD(p,q) = \sqrt{\frac{1}{n}\left(\|R^p\|^2 + \|R^q\|^2 - 2\lambda^*\right)}$$

where λ^* is the largest eigenvalue of $\mathcal{F}$.

Globally minimized RMSD

The RMSD_l is however not invariant under index permutations of chemically identical atoms. If the configuration p and q are identical, Eq. (33) will be different from zero if we permute for instance in R^q the positions r_i^q and r_j^q of atoms i and j. This is important since for instance during molecular dynamics or a geometry optimization, atoms may be flipped. The minimum Frobenius distance obtained by considering all possible index permutations for an arbitrary rotation U is

$$RMSD_P(p,q) = \frac{1}{\sqrt{n}} \min_P \|R^p - U R^q P\|, \tag{34}$$

P being an $n \times n$ permutation matrix. This assignment problem is solved in polynomial time using the so-called Hungarian algorithm. However, what is really needed is a solution of the combined problem of the global minimization over all rotations and permutations, namely

$$RMSD(p,q) = \frac{1}{\sqrt{n}} \min_{P,U} \|R^p - U R^q P\|. \tag{35}$$

The global minimum RMSD fulfills all the properties of a metric. The coincidence and symmetry properties are easy to see. Using the standard triangle

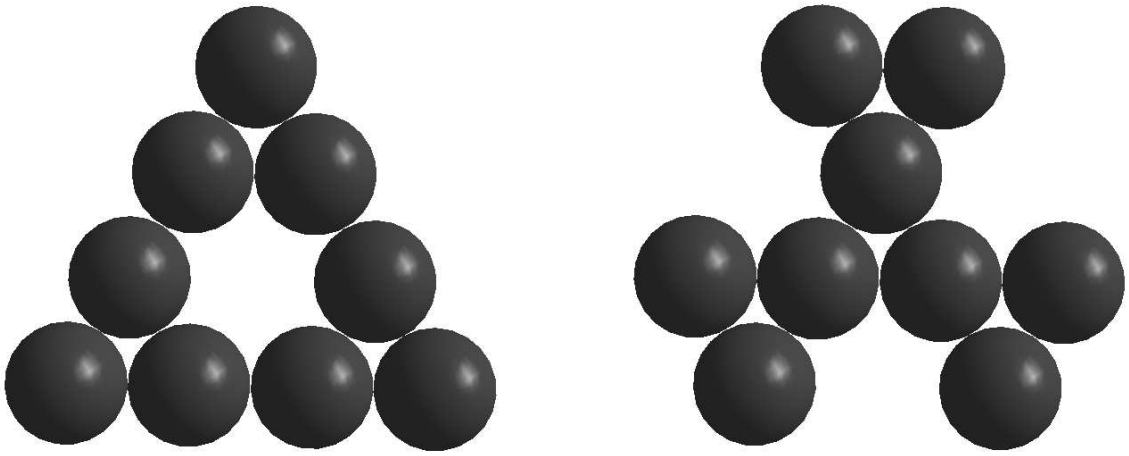
inequality, the proof of the triangle property is as follows:

$$
\begin{aligned}
& RMSD(p,q) + RMSD(q,r) \\
&= \frac{1}{\sqrt{n}} \min_{P,U} \|U R^p P - R^q\| + \frac{1}{\sqrt{n}} \min_{P,U} \|R^q - U R^r P\| \\
&= \frac{1}{\sqrt{n}} \|U_{pq} R^p P_{pq} - R^q\| + \frac{1}{\sqrt{n}} \|R^q - U_{rq} R^r P_{rq}\| \\
&\geq \frac{1}{\sqrt{n}} \|U_{pq} R^p P_{pq} - R^q + R^q - U_{rq} R^r P_{rq}\| \\
&\geq \frac{1}{\sqrt{n}} \|R^p - U_{rp} R^r P_{rp}\| \\
&= RMSD(p,r)
\end{aligned}
$$

where $\min_{P,U} \|U R^p P - R^q\|$ is denoted by $\|U_{pq} R^p P_{pq} - R^q\|$. In some applications, one might apply restrictions to the permutations in order to reduce the size of the permutation space. For instance, in an application to organic molecules only equivalent atoms have to be permuted. Equivalent atoms in an organic molecule are considered for example those that have identical connectives. Finding the globally minimized RMSD is in principle again a global optimization problem. In practice it can be solved for not too large system by Monte Carlo methods.

Fingerprints

The fact that finding the globally minimized RMSD is a numerically costly global optimization problem calls for metrics that are cheaper to calculate. Quantities that allow to characterize a structure are frequently called fingerprints. If distances can be calculated betwenn two such fingerprints, they can be an alternative to the RMSD. As consequence a very large number of fingerprints describing a structure way have been proposed. Many of these fingerprint are based on interatomic distances. It turns out that interatomic distances alone do not give optimal fingerprints. In some cases interatomic distances can not even uniquely characterize a structure. Two distinct structures with identical interatomic distances are called homo-metric and an example is shown below.



Fingerprints based on eigenvalues of symmetric matrices

We consider symmetric $M \times M$ matrices whose elements depend only on the interatomic distances $r_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\|$ of an N-atom configuration. Vectors V containing eigenvalues of such a matrix form a configurational fingerprint that allows to identify a structure. The normalized Euclidean distance

$$\Delta_V(p,q) = \frac{1}{\sqrt{N}} \|V^p - V^q\| \tag{36}$$

measures the dissimilarly between p and q with no need to superimpose them. A simple example of such a matric is the contact matrix C. Its matrix element $C_{i,j}$ is 1 if the distance between atom ii and j is less than the sum of the covalents radii plus some margin and zero otherwise. Diagonal elements can be set to zero. The matrix elements of this matrix change discontinuously if bonds are stretched by more than the margin. Therefore some smooth cutoff function is more appropriate like for instance $C_{i,j} = exp(-\frac{-(\mathbf{r}_i - \mathbf{r}_j)^2}{(\sigma_i + \sigma_j)^2}$. As a consequence the eigenvalues of this function will then also vary smoothly. The N eigenvalues of such a smooth $N \times N$ contact matrix will however not be sufficient to characterize the structure in a unique way, since there are $3N-6$ degress of freedom. It can be shown that for any matrix of dimension M less than $3N-6$ there exists a subspace of at least dimension $M-3N+6$ where all the structures have the same fingerprint. For a unique characterisation we have therefore to

find a matrix whose dimension is larger than $3N - 6$. One possible matrix of this type is a overlap matrix, familiar form quantum mechanical calculations, where one has one s and 3 p-type atomic orbitals per atom. In this way one obtains fingerprint vectors of length $4N$ which are long enough to characterize the structure in a unique way. Other possibilities are the eigenvalues of the Kohn-Sham Hamiltonian or the eigenvalues of the Hessian matrix.