13 Estimating errors

Computer simulation can be thought as a virtual experiment. There are

- systematic errors
- statistical errors.

As systematical errors can be very complicated to find, statistical significance of results **can and should** always be estimated.

13.1 Systematic errors

Systematic errors result in a constant bias from the 'proper' behaviour, most frequently displace the average property from its proper value. Among others, they can be due to

- Fault in algorithm
- fault in model
- approximations in the algorithm
- round off errors

A simple way to check for systematic errors is to study the distribution of simple thermodynamic properties, which should usually by follow normal distribution. Another good test is to run the simulation with different

- computer or compiler
- model
- algorithms
- parameters
- code

If systematic errors can be ruled out, we are still left with statistical errors.

13.2 Statistical errors

The simulations are run finite time. Even though we argue ergodicity to hold, for any quantity $A(\mathbf{\Gamma})$

$$\frac{1}{t_{sim}} \int_0^{t_{sim}} A(\mathbf{\Gamma}(t'))dt' = \langle A \rangle_{sim} \neq \langle A \rangle$$

unless something very uncommon takes place. Standard deviation σ is a standard quantity to study statistical errors. Assume that one has obtained M values for quantity A. Then the standard deviation for **the average value** $\langle A \rangle_{sim}$,

$$\sigma_{\langle A \rangle_{sim}} = \frac{\sigma_A}{\sqrt{M}} = \frac{\sqrt{\sum_{i=1}^{M} (A_i - \langle A \rangle_{sim})^2}}{M}$$

The equation assumes our snapshots being independent from each other, which two subsequent snapshots in MD never are.

- For each property there is a correlation/relaxation time
- This is usually unknown or even the subject of the study

13.2.1 Block average method

Block average method is a simple method to find error estimated for averages in the simulation. In the method

- time steps are divided to n_b blocks of length τ_b (successive snapshots)
- $\tau_{sim} = n_b \tau_b$
- Average for each block is calculated

$$\langle A \rangle_b = \frac{1}{\tau_b} \sum_{i=1}^{\tau_b} A_i$$

As the block length τ_b is encreased, block averages are expected to be uncorrelated.

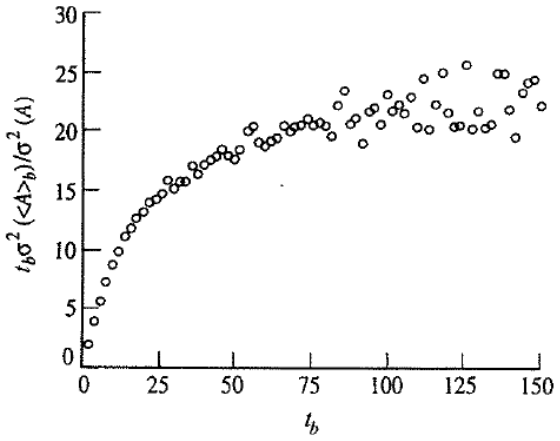
- In the limit

$$\sigma^2(\langle A \rangle_b) = \frac{1}{n_b} \sum_{b=1}^{N_b} (\langle A \rangle_b - \langle A \rangle_{sim})^2 \propto \tau_b^{-1}$$

The limiting value for obtaining uncorrelated snapshots (statistical inefficiency) can be calculated as

$$s = \lim_{\tau_b \to \infty} \frac{\tau_b \sigma^2(\langle A \rangle_b)}{\sigma^2(A)}.$$

For real simulations, the limit can not be evaluated, but the value can be read from a plot. The graph should show a steep rise for low τ_b and end in an plateu – which is interpreted as the limit.



Block average plot for obtaining $\sigma_{\langle A \rangle_{sim}}$. The limit value $s = 23$ is obtained. (Source: Leach)

When the limiting value s is found (well, approximately), the standard deviation of the **obtained average** $\langle A \rangle_{sim}$ can be calculated from

$$\sigma(\langle A \rangle_{sim}) \approx \sigma(A) \sqrt{\frac{s}{M}}.$$

If relaxation time s is known average can be calculated

- single values of A separated by s
- single values randomly picked from blocks of legth s
- coarse-grained, as average of block averages obtained using block length s

The errors can be made smaller by more extensive simulations.

13.2.2 Bootstrap method

Bootstrap method is an intuitive method to estimate statistical significance of the sampling. Let us assume that we have

- M snapshots $\mathbf{\Gamma}_i$, $i = 1, ..., M$
- Quantity $A = A(\mathbf{\Gamma})$ and therefore values $\{A_i\}_{i=1}^{M}$

and we would like to know the statistical error for our $A_{estimate} = 1/M \sum_i A_i$.

In bootstrap method we generate K trial combinations of $A_{estimate}$ by

$$A^K_{estimate} = 1/M \sum_i c_i A_i,$$

where $\mathbf{c}$ is a vector of length M

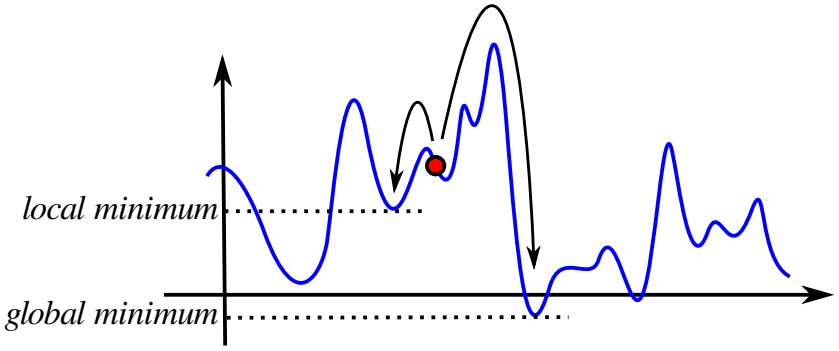
$$\mathbf{c} = (c_1, c_2, ...c_M), \quad c_i \in \mathcal{N},$$

so that $\sum_i c_i = M$. Now we have K "averages"for A derived from our set, for which we can evaluate variance σ, or calculate confidence interval with chosen confidence level. As our sampling improves, the error limit obtained from the bootstrapping $\to 0$. The number K should be as large as possible. Bootstrap method can be overly optimistic, but it is easy to implement and apply to any quantity of interest.

14 Structural optimization methods

Optimization problems are in the heart of many physical applications

- In short, optimization is about finding the set of parameters $x, y, \ldots$ for a given *objective function* $f(x, y, \ldots)$ which will yield either the minimum or maximum value
- Saddle points are important for chemical reactions.
- The extreme values searched for can be either local or global
- In one-dimensional case:



An unvisionary optimization method would be to map the complete configuration space (all possible parameters) and thereby find all possible values of the function

- However, this is obviously not a very effective approach to a multi-dimensional problem
- The computational approach is, as so often, dictated by the computational cost
- This leads to a simple rule: evaluate $f(x, y, \ldots)$ as few times as possible
- In general, it is a very difficult problem to effectively find the global extreme value

14.1 Energy Minimization

For molecular simulations, the configurations of minimal $\mathcal{U}(\mathbf{r})$ are of special interest we turn to 3N (N_f) dimensional minimization of $\mathcal{U}(\mathbf{r})$. Our aim for use of minimization can be among others

- removing hot spots from $\mathbf{r}_0$ of MD
- defects, local minima
- normal mode analysis, local and global minima

Naturally, different purposes need different methods.

- Local energy minimization is carried out using traditional optimization techniques
- Global optimization, on the other hand, has exploded during the recent years with the introduction of evolutionary structure optimization algorithms (allowed by fast enough computers)

Let us define the energy surface as the ground state energy function of N_f coordinates $\mathcal{U}(\mathbf{r})$. Therefore methods described can be appliend to quantum chemical and molecular mechanics approaches alike. Now we are looking for points $\{\mathbf{r}_m\}$ for which

$$\left.\frac{\partial \mathcal{U}}{\partial r_i}\right|_{\mathbf{r_m}} = 0, \quad \left.\frac{\partial^2 \mathcal{U}}{\partial r_i^2}\right|_{\mathbf{r}} > 0, \quad \forall i = 1, ..., N_f. \tag{14.1}$$

For the functions $\mathcal{U}(\mathbf{r})$ we have in mind

- Standard calculus methods are inapplicable
- We refer to numerical methods
  - starting from $\mathbf{r}_0$ we algorithmically obtain a sequence of points $\mathbf{r}_i$ that
  - hopefully converge to our minimum point (within some tolerance)

The ideal minimization algorithm should

- Converge to desired point

- Converge quickly
- Use little memory and be efficient
- In general, an algorithm that is good for QM may not be optimal for molecular mechanics , though.

Naturally there is no superior-proven algorithm around. We will first deal with numerical methods for close-lying minima, and then study methods for searching global minimum.

14.2 Some possible approaches

(1) Monte Carlo simulation
This can be carried out by setting the temperature $T \to 0$ K in a MC simulation (system will naturally reach for the lowest energy)

+ Can work in finding the equilibrium coordination in a liquid
− Not very efficient in finding the closest minimum

(2) MD simulation
Similar to MC, the system can be driven to $T = 0$ K slowly enough to find a local minimum structure = simulated annealing

+ Can work efficiently for the local minimum
+ Sometimes may even work in finding the global minimum if the system is first heated to a high enough temperature and then quenched slowly

(3) Iterative minimization algorithms
Dedicated algorithms for finding the local minimum for a multi-dimensional function. May involve derivatives or not.

(4) Evolutionary algorithms
Based on creating sample population of the parameter set which is let to "*evolve*" in the spirit of natural selection

- In the following, the last two of these methods are introduced in more detail

14.3 Non-derivative methods

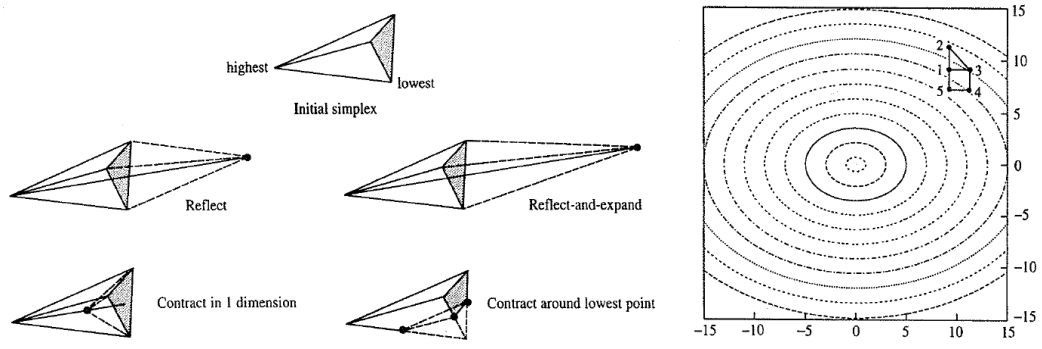
14.3.1 Simplex algorithm

For your task, simplex is a geometrical figure with N_f+1 verteces ("corners").

- For $N_f = 2$, simplex is a triangle
- For $N_f = 3$, simplex is a tetrahedron
- and so on...
- Each vertex (corner point) corresponds a vector $\mathbf{r}$ for which energy $\mathcal{U}(\mathbf{r})$ can be calculated.

The simplex moves by moving one or several of its verteces. The allowed moves are

- Reflect the highest-value point to other side of the hyperplane defined by other verteces
- Reflect-and-expand if reflect gives minimum of all verteces
- If valley floor is obtained for reflection, contract along one dimension from highest-value point
- If contraction fails, contract in all directions around lowest point.



A 4-point simplex for $N_f = 3$ and possible moves. A 3-point simplex attacing function $f(x, y) = x^2 + 2y^2$: Starting as 1-2-3 $\rightarrow$ 1-3-4 $\rightarrow$ 1-4-5. The minimum will be found eventually (it will take 30 steps to reach values < 0.1). (Source: Leach)

If one $\mathbf{r}_0$ is given, other verteces need to be generated. One possible way is to

- deviate the coordinates in each direction in turn.
- use some imagination and figure a way to get N_f extra points.

The simplex algorithm is simple and it

- is most useful for high-energy initial configurations
- rarely fails in finding a *better* solution
- can be expensive in terms of computer time
- uses many evaluations of $\mathcal{U}(\mathbf{r})$

Simplex is therefore used in combination of other minimization algorithms. A few steps with simplex in the beginning will set things going in a good direction. Note that in simplex it is crucial to have one additional point to the number of degrees of freedom. This extra vertex allows the simplex to explore the parameter space. Spend a minute thinking that.

14.4 Derivative minimization methods

Derivative methods utilize the knowledge of slope $\mathbf{g} = \nabla\mathcal{U}(\mathbf{r})$ and higher order derivatives in the formulation of the algorithms. This is of course more sophisticated and efficient for functions smooth enough. The derivative methods are the most popular choice for minimization methods.

- The derivatives contain information about curvature of the energy landscape $\mathcal{U}(\mathbf{r})$ and
- therefore can help in estimating the position of the minimum.

In general, when located at point $\mathbf{r}$ we will use the Taylor expansion to estimate $\mathcal{U}(\mathbf{r}+\mathbf{x})$

$$\mathcal{U}(\mathbf{r}+\mathbf{x}) = \mathcal{U}(\mathbf{r}) + \sum_i \left.\frac{\partial\mathcal{U}}{\partial r_i}\right|_{\mathbf{r}} x_i + \frac{1}{2}\sum_{i,j}\left.\frac{\partial^2\mathcal{U}}{\partial r_i \partial r_j}\right|_{\mathbf{r}} x_i x_j + \ldots \quad (14.2)$$

$$\approx c + \mathbf{g}\cdot\mathbf{x} + \frac{1}{2}\mathbf{x}\cdot\mathbf{H}\cdot\mathbf{x} \quad (14.3)$$

where

$$c \equiv \mathcal{U}(\mathbf{r}) \qquad \mathbf{g} \equiv \nabla\mathcal{U}|_{\mathbf{r}} \qquad [\mathbf{H}]_{ij} \equiv \left.\frac{\partial^2\mathcal{U}}{\partial r_i \partial r_j}\right|_{\mathbf{r}} \tag{14.4}$$

Matrix $\mathbf{H}$ is the *Hessian matrix* of the function $\mathcal{U}$ at $\mathbf{r}$. In derivative methods, analytical evaluation of the derivative and higher derivatives are very desired.

14.4.1 Steepest descent method

The steepest descent method is the most intuitive minimization method. Probably due to that, it usually is not the best choice.

- Easy way of understanding the steepest descent method is to picture oneself on top of a hill and looking for a way down
- In this case, one would always take the steepest slope and walk downwards until the local "valley" is reached; after this the same operation would be repeated until the bottom is reached

We will use only the first derivative (gradient) to deside

- the direction of the next step

In the method we choose

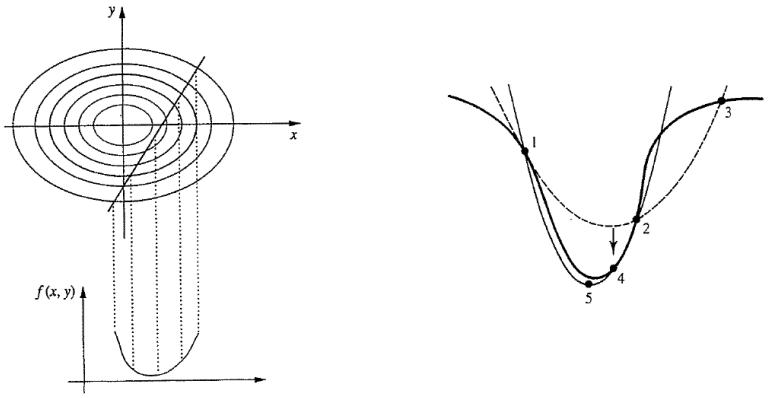
$$\mathbf{s}_k = -\frac{\mathbf{g}_k}{|\mathbf{g}_k|}, \quad \mathbf{g}_k = \nabla\mathcal{U}|_{\mathbf{r_k}}$$

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \lambda_k \mathbf{s}_k$$

and therefore we are going downhill the steepest direction. Coefficient $\lambda_k > 0$ can be obtained by numerous different ways.

Line-search The line-search method can be used in steepest descent. The method searches for the minimum $\mathbf{r}_m^{(LS)}$ in a given direction (line) from $\mathbf{r}_k$. Here the direction is given by $\mathbf{s}_k$

$$\mathcal{U}(\mathbf{r}_m^{(LS)}) = \min\{\mathcal{U}(\mathbf{r}_k + \lambda_k \mathbf{s}_k) | \lambda_k \in \mathcal{R}\}$$



Line search in action for steepest descent method. A good fit to the line search curve in 1D will help obtaining good step. (Source: Leach)
If line-search is done properly, the gradients of the two preceeding steps will be orhogonal

$$\mathbf{g}_i \cdot \mathbf{g}_{i+1} = 0,$$

which may not be optimal choice.

Arbitrary step method Line search method can be computationally demanding. Therefore step size λ_k can be set to have a predifined value that is

- increased if the previous step lead to decreasing energy
- decreased if the previous step lead to increasing energy

Problems Steepest descent method with line search makes orhogonal steps that can cause problems.

- Steepest descent method fails in long narrows valleys.
- There the algorithm may oscillate mainly between sides of the valley, not in the direction of it.

However,

- Direction of the gradient is dictated by largest interatomic forces
- Steepest descent method is good for relieving highest-energy structures in $\mathbf{r}_0$

Steepest descent is inefficient in finding the minimum, but a good method for beginning of the search.

14.4.2 Conjugate gradients method

The conjugate gradients method is a first method. The method does not show the oscillatory behaviour of the SD method in narrow valleys. In conjugate gradients method the step is calculated from

$$\mathbf{s}_k = -\mathbf{g}_k + \gamma_k \mathbf{s}_{k-1}, \quad \gamma_k = \frac{\mathbf{g}_k \cdot \mathbf{g}_k}{\mathbf{g}_{k-1} \cdot \mathbf{g}_{k-1}}$$

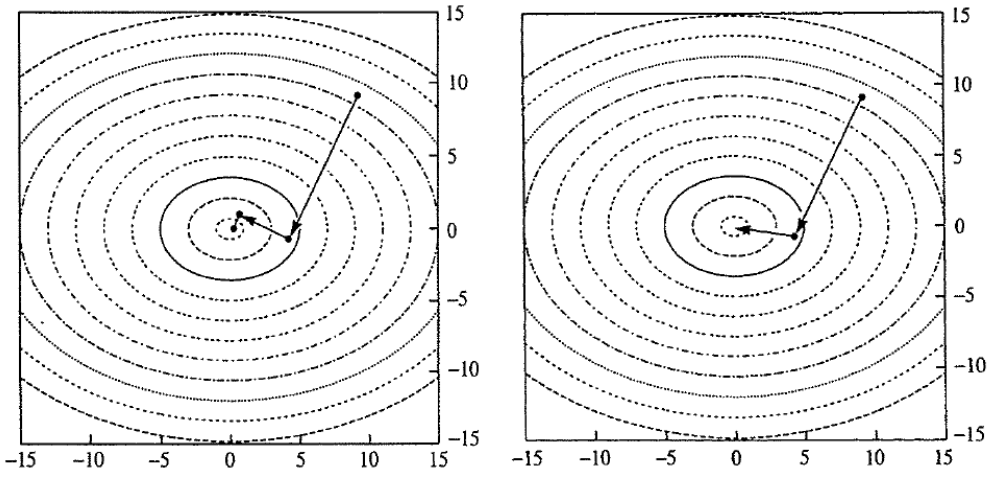
and of course

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \lambda_k \mathbf{s}_k$$

In the CG method, following relationships are fulfilled

$$\begin{aligned} \mathbf{g}_i \cdot \mathbf{g}_j &= 0, \quad i \neq j \\ \mathbf{g}_i \cdot \mathbf{H} \cdot \mathbf{g}_j &= 0, \quad i \neq j \qquad (14.5) \\ \mathbf{g}_i \cdot \mathbf{s}_j &= 0, \quad i \neq j \qquad (14.6) \\ & \qquad (14.7) \end{aligned}$$

For CG method, line-search in 1D or arbitrary step methods can be used. For a quadratic function of M variables, the CG method yields the minimum in M steps. The first step of the algorithm is a standard SD step.



Steepest descent and conjugate gradient method tackling minimum search of function $f(x, y) = x^2 + 2y^2$ starting from point (9,9). The third point for SD has function value f=0.099 whereas the second step for CG gives the exact solution.

What conjugate means Conjugate directions are "*non-interfering*" in the sense that when the new direction is conjugate to the old one, the new direction will not lead to increase in the function value with respect to the previous direction

- Say that the last line-search took place in direction $\mathbf{u}$, resulting at point $\mathbf{r}_k$.
  - The gradient $\mathbf{g}_k = \nabla\mathcal{U}|_{\mathbf{r}_k}$ must be perpendicular to $\mathbf{u}$ at the line minimum
$$\mathbf{g}_k \cdot \mathbf{u} = 0$$
- For the next step in direction $\mathbf{v}$, the function can be approximated as
$$\begin{aligned} \mathcal{U}(\mathbf{r}_k + \mathbf{v}) &= \mathcal{U}(\mathbf{r}_k) + \sum_i \left.\frac{\partial \mathcal{U}}{\partial r_i}\right|_{\mathbf{r_k}} v_i + \frac{1}{2}\sum_{i,j} \left.\frac{\partial^2 \mathcal{U}}{\partial r_i \partial r_j}\right|_{\mathbf{r_k}} v_i v_j + \ldots \quad (14.8) \\ &\approx c + \mathbf{g}_k \cdot \mathbf{v} + \frac{1}{2}\mathbf{v} \cdot \mathbf{H}^{(k)} \cdot \mathbf{v} \quad (14.9) \end{aligned}$$
  where $\mathbf{g}_k$ and $\mathbf{H}^{(k)}$ are naturally evaluated at $\mathbf{r}_k$.
- The gradient with of $\mathcal{U}$ at $\mathbf{r}_k + \mathbf{v}$ is easily calculated as
$$\nabla\mathcal{U}|_{\mathbf{r}_k+\mathbf{v}} = \mathbf{g}_k + \mathbf{H}^{(k)} \cdot \mathbf{v} \quad (14.10)$$
- The sufficient condition for us not to *spoil* the minimization along $\mathbf{u}$ is
  - to have the gradient after step $\mathbf{v}$ perpendicular to $\mathbf{u}$,
$$0 = \mathbf{u} \cdot \nabla\mathcal{U}|_{\mathbf{r}_k+\mathbf{v}} = \mathbf{u} \cdot (\mathbf{g}_k + \mathbf{H}^{(k)} \cdot \mathbf{v}) = \mathbf{u} \cdot \mathbf{H}^{(k)} \cdot \mathbf{v} \quad (14.11)$$
- If this is valid, the directions $\mathbf{u}$ and $\mathbf{v}$ are considered to be conjugated
- If minimization of function is carried out using such a set of conjugate directions, none of the directions needs to be considered twice

14.4.3 Newton-Rhapson method

We can use the quadratic Taylor expansion in calculation of the iteration steps. Thus in addition to slope, we also add information about curvature of $\mathcal{U}$ in our algorithm. This is the idea in Newton-Rhapson method, which is of second order.

- As we noted already for Taylor series expanded at point $\mathbf{r}_k$ cut to quadratic terms
$$\nabla\mathcal{U}|_{\mathbf{r}} = \mathbf{H}^{(k)} \cdot (\mathbf{r} - \mathbf{r}_k) + \mathbf{g}_k \tag{14.12}$$
Setting this to $\nabla\mathcal{U}_{|\mathbf{r}} = 0$ one has
$$\mathbf{r} = \mathbf{r}_k - \mathbf{g}_k(\mathbf{H}^{(k)})^{-1} \tag{14.13}$$

- In NR method, we simply iterate
$$\mathbf{r}_{k+1} = \mathbf{r}_k - \mathbf{g}_k(\mathbf{H}^{(k)})^{-1}, \tag{14.14}$$
where $\mathbf{g}_k$ and $\mathbf{H}^{(k)}$ are naturally evaluated at $\mathbf{r}_k$.

For a purely quadratic function, the NR converges to the exact solution with only one step.

- inverting Hessian $\mathbf{H}^{(k)}$ is laborous computationally and memory-wise
- NR is best suited for problems with $N_f < 1000$
- For minimum point, Hessian needs to be positive definite (remember, $\nabla\mathcal{U} = 0$ can be maximum or a saddle point). This sets requirement of being reasonably close to minimimum.
- In principle close to minimum $\mathcal{U}$ is close to quadratic. FAST CONVERGENCE THERE!

Therefore one should start with someother algorithm and use NR only close to minimum, where it is fast converging.

Quasi-Newton methods The NR method has some drawbacks that one would like to circumvent it by ingenious tricks. The resulting approximations of NR method are called quasi-Newton methods.

- Analytic second derivatives not always available
- Inversion of Hessian is a laborous task

Quasi-Newton methods gradually build up the inverse Hessian matrix $(\mathbf{H}^{(k)})^{-1}$ in successive iterations so that

$$\lim_{k\to\infty} \mathbf{H}_k = \left.\frac{\partial^2 \mathcal{U}}{\partial r_i \partial r_j}\right|_{\mathbf{r}_k}.$$

Of course, as $\mathbf{r}_k$ changes every step, the algorithms need to be able to

- estimate the additional information obtained each step
- transform the existing information to the new point $\mathbf{r}_k$ to provide a useful $(\mathbf{H}^{(k)})^{-1}$.

The Hessian is often initialized $\mathbf{H}_1 = \mathbf{I}$, although more detailed knowledge about bonding in the system can also be used. Naturally the better the initial Hessian, the better convergence.

14.5 Using different minimization algorithms

The choice of the algorithm is dictated by a number of factors

- memory & computational resources may couse problems for any Hessian method of over 10000 atoms
- of the two first order methods we discussed
  - Steepest descent outperforms in the beginning
  - Conjugate gradients method outperforms closer to minimum

Quantum mechanical calculations are limited to small number of atoms and therefore storing Hessian is not a problem. The more time energy evaluation takes, the fewer steps we would like to take.

- For QM optimization, quasi-Newton because no second derivatives
- For classical force fields, steepest descent and conjugate gradient methods are a good choice

Convergence criteria In real systems one always obtains estimates for the minimum. At some point of the iteration scheme the algorithm must be made stop.

- This point is much before the numerical precision
- Estimation when the result is accurate enough
- The most used criteria are
  - Change in energy
  - Step length
  - RMS gradient (or the norm)

$$RMS = \sqrt{\frac{\mathbf{g}^T\mathbf{g}}{3N}}$$

  - maximal component value in $\mathbf{g}$

$$MAX = \max\{|g_i|, \quad i = 1, ..., N_f\}$$

The approaches we presented above are rather classic in their form. We will however have a brief introduction to other kind of way of optimization. The Darwinist way.

14.6 Evolutionary algorithms

Evolutionary algorithms are based on ideas from biology. They have been implemented for physical problems only lately, mainly owing to increases in computing power. There are three basic classes of evolutionary algorithms

- Genetic algorithms
- Evolutionary programming
- Evolution strategies

14.6.1 Genetic algorithms

Evolutionary algorithms are based on the simple idea of natural selection, i.e., survival of the fittest. This is however an optimization method. It does not need to be feasible in biological sense. The canonical genetic algorithm consists of phases

(0) Generate a population of μ initial candidates (sets of coordinates). For genetic algorithm these are coded in bits 0 and 1.

(1) Evaluate the fitness for the task in question for each individual (evaluating energy $\mathcal{U}$)

(2) Generate $\mu/2$ pairs of parents. Selection is random but biased to favor the best individuals.

(3) Generate μ offsprings by recombining genes of parents

(4) Make all offspings undergo a mutation (for example every bit has 1% probability to be flipped)

(5) The new generation becomes the population and new cycle starts from (1).

- Iterate until the fitness does not (substantially) improve any longer

Many variants on the canonical genetic algorithm have been suggested. One might for example

- Leave the highest ranking unchainged and not replace them (elitist strategy, the fittest value can only improve or stay the same)
- Perform mutation or crossover only for a few individuals at a time.
- Use different mechanisms of crossover

Even though real numbers can be succesfully coded in the genetic algorithm, one usually uses binary representation.

In genetic algorithms the diversity of the population needs to be maintained with feasible choices of probalities. Possible methods include

- Selection pressure = probability of fittest to parent / propability of average to parent
  - Can be very important in preventing premature convergence
  - Can be very important in preventing too slow convergence
  - Selection pressure can be controlled by adjusting bias for reproduction of the fittest individuals
- Island model = diverse populations on different islands, some exchange of individuals
- Niching = forcing individuals to find new areas of genomes. Very similar parents have lower probability for reporduction

Evolutionary programming does not use crossovers but only mutation.

- Individuals are generated by mutations - one for each parent
- Real values for genes - mutations by adding a dandom number from Gaussian distribution
- The μ survivors obtained from tournaments with random opponents.
- Ranking as a function of wins – μ most victorious continue

Evolution strategies are very similar to evolutionary programming. They differ in two aspects

- Crossover operations are permitted in ES
- Tournaments are replaced with a straightforward ranking – μ fittest survive.
  - λ offsprings generated by crossover and mutation, the $\mu + \lambda$ individuals are ranked.

In general, evolutionary algorithms are primarily intended for global optimization. They involve significant random elements and therefore cannot be guaranteed to always converge to same solution.

- They are excellent in producing solutions close to global optimum in a reasonable time.
- As there is a population, one obtains a set of good guesses
- Obviously, the evolutionary algorithm processes have themselves no physical meaning
- Therefore, the method can only be used to search for the minimum energy structures

A simple genetic algorithm is also a great fun to program. However remember that this is a method of optimization, and not model of life to be taken too literally.

Summary

- Typical optimization problem in the context of atomistic simulations is energy minimization
- Local minima are important for structure optimization, e.g., around a defect
- Global minima, on the other hand, are required for searching for minimum energy structures
- Minimization of energy can be used for removing hot spots from $\mathbf{r}_0$.
- Conjugate gradients is an effective way to find a local minimum for a given set of atoms
- For global minimization, evolutionary algorithms have turned out to be an effective way to optimize atomic structures. They can find reasonable solutions with wery bad initial guesses.