Energy minimization techniques

- The task of minimizing the energy of a set of atoms is a very common, yet surprisingly complex problem to solve efficiently.
  - N atoms, set of atomic coordinates $\mathbf{x} = (\mathbf{r}_{1x}, \mathbf{r}_{1y}, \mathbf{r}_{1z}, \mathbf{r}_{2x}, \ldots)$, system potential energy $V(\mathbf{x})$
  - Find $\mathbf{x}$ that minimizes $V(\mathbf{x})$
  - Examples: the equilibrium shape of a protein, the ground state configuration of an atom cluster, a minimum-energy configuration of a defect, ...
- A large variety of energy minimization techniques in numerical mathematics.
  - For large sets of atoms, one has to require that the memory requirement of the method scales as $O(N)$, which rules out many efficient techniques which require $O(N^2)$ memory.
    - In these $O(N^2)$ methods the Hessian matrix $\mathbf{A}$, $A_{ij} = \frac{\partial^2 V}{\partial x_i \partial x_j}$ is usually needed.

Energy minimization techniques

- At least the following approaches can be used to atomistic energy minimization:

  1. Monte Carlo simulation: Do an MC-simulation letting $T \to 0$.

  - Can be good e.g. in finding the equilibrium coordination in a liquid.
  - Not very efficient in finding the closest local minimum.
  - Good when non-physical moves needed to reach the equilibrium.

  2. MD simulation: Do an MD-simulation letting $T \to 0$.

  - Can be made more efficient by setting all $v = 0$ if the energy grows, or by setting $v_i = 0$ if the force $\mathbf{f}_i$ is in the opposite direction to $\mathbf{v}_i$
  - Sometimes quite efficient in finding a local minimum
  - Sometimes also good tool to find a global minimum: simulate at high T first, cooling down in cycles.

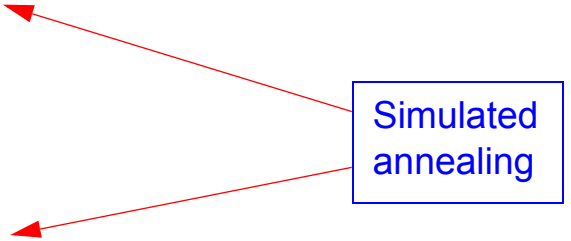
  3. Conjugate gradient

  - Very efficient method to find a local minimum.

  4. Genetic algorithm

  - Probably best method to find a global minimum from a random initial configuration.

- In this lecture package conjugate gradient and genetic algorithms are presented.



Energy minimization techniques

- A sidenote: optimization at ‘constant pressure’ : Usually the potential energy V is written as a function of the coordinates $\{x_i\}, \{y_i\}, \{z_i\}$ $(i = 1, \ldots, N)$ of the atoms in the system. When the cell edges are taken as variables it is easier to write the energy as a function of reduced coordinates $\{s_i\}, \{t_i\}, \{u_i\}$ and sizes of the simulation box in x, y, z directions: α, β, γ: $V = V(\{s_i\}, \{t_i\}, \{u_i\}, \alpha, \beta, \gamma)$ where $s_i = x_i/\alpha$, $t_i = y_i/\beta$, $u_i = z_i/\gamma$

- Now the gradient of the potential energy is $\nabla U = \begin{pmatrix} \partial V/\partial s_1 \\ \partial V/\partial s_2 \\ \cdots \\ \partial V/\partial t_1 \\ \partial V/\partial t_2 \\ \cdots \\ \partial V/\partial u_1 \\ \partial V/\partial u_2 \\ \cdots \\ \partial V/\partial \alpha \\ \partial V/\partial \beta \\ \partial V/\partial \gamma \end{pmatrix} = \begin{pmatrix} -F_{x1}\alpha \\ -F_{x2}\alpha \\ \cdots \\ -F_{y1}\beta \\ -F_{y2}\beta \\ \cdots \\ -F_{z1}\gamma \\ -F_{z2}\gamma \\ \cdots \\ -1/\alpha \sum_i F_{xi} x_i \\ -1/\beta \sum_i F_{yi} y_i \\ -1/\gamma \sum_i F_{zi} z_i \end{pmatrix} = \begin{pmatrix} -F_{x1}\alpha \\ -F_{x2}\alpha \\ \cdots \\ -F_{y1}\beta \\ -F_{y2}\beta \\ \cdots \\ -F_{z1}\gamma \\ -F_{z2}\gamma \\ \cdots \\ -W_{xx}/\alpha \\ -W_{yy}/\beta \\ -W_{zz}/\gamma \end{pmatrix}$

Conjugate gradient

- The conjugate gradient (CG) method is a general method to minimize function $f(\mathbf{x})$, where f can be any function of points $\mathbf{x}$ in N-dimensional space [Numerical Recipes, 2nd ed. ch. 10]
  - For N atoms we can write their coordinates $\mathbf{r}$ as a $3N$-dimensional vector $\mathbf{x}$ of the form

    $$\mathbf{x} = (\mathbf{r}_{1x}, \mathbf{r}_{1y}, \mathbf{r}_{1z}, \mathbf{r}_{2x}, \ldots)$$

  - The function $f(\mathbf{x})$ corresponds now to the potential energy function $V(\mathbf{r})$.
- In the CG method the gradient (force) of the function is used as a help in finding the minimum.
  - The gradient tells in which direction the function changes the most rapidly.

Conjugate gradient

- An obvious, but *not very efficient* way to minimize the energy is to always move in the direction of the negative gradient.

  - This is the so called **steepest descent method**, which for atoms can be described as follows:

    0. Start from point $\mathbf{r}_0$, set $i = 0$.

    1. Calculate $V_i(\mathbf{r}_i)$, $\mathbf{F}_i = -\nabla V_i(\mathbf{r}_i)$.

    2. If $V_{i-1} - V_i < \varepsilon$ end.

    3. Minimize $V(\mathbf{r}_i + \alpha \mathbf{F}_i)$ with respect to the scalar quantity α.

    4. Set $\mathbf{r}_{i+1} = \mathbf{r}_i + \alpha \mathbf{F}_i$ and $i = i + 1$.

    5. Return to stage 1.

  - The algorithm resembles MD, but: no time, velocity or acceleration.

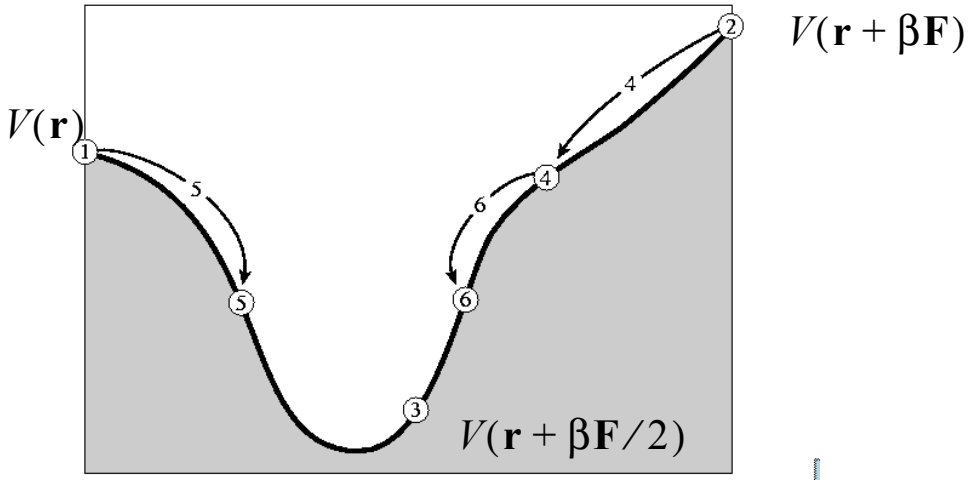
  - The line minimization in stage **3** a 1-dimensional operation in which the minimum of a function is sought by moving in a predetermined direction $\alpha \mathbf{F}_i$.

  - The line minimization is a relatively straightforward operation which is carried out in two steps.

    1. Make sure that there is a minimum and bracket it.
    2. Search it with a given accuracy.

Conjugate gradient

- Stage **1** is in principle easy to carry out. Starting from a point $\mathbf{r}$ and known direction $\mathbf{F}$, move forward some direction $\beta\mathbf{F}$. If $V(\mathbf{r}+\beta\mathbf{F}) > V(\mathbf{r})$ and in addition $V(\mathbf{r}+\beta\mathbf{F}/2) < V(\mathbf{r})$ and $V(\mathbf{r}+\beta\mathbf{F}/2) < V(\mathbf{r}+\beta\mathbf{F})$ the minimum is bracketed with the three points (1) $V(\mathbf{r})$ (3) $V(\mathbf{r}+\beta\mathbf{F}/2)$ and (2) $V(\mathbf{r}+\beta\mathbf{F})$. If these criteria are not fulfilled, increase β and try again.

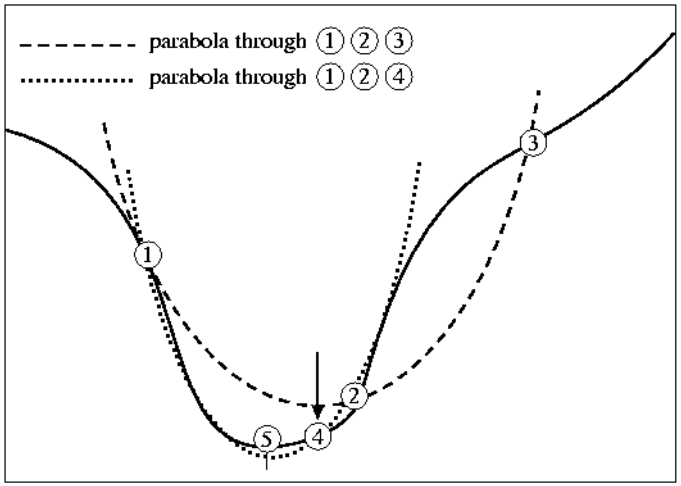


- After the minimum has been bracketed, one could of course use ordinary binary search to find it. A slightly better method turns out to be to use a golden section, i.e. let the new minimum be 0.38197 from either end.
- Often much better is to use so called inverse parabolic interpolation. In this method, a parabola is fit to the points a, b and c (corresponding to (1), (2) and (3) above), and the estimate of the minimum is the minimum of the parabola x:

$$x = b - \frac{1}{2}\frac{(b-a)^2[V(b)-V(c)] - (b-c)^2[V(b)-V(a)]}{(b-a)[V(b)-V(c)] - (b-c)[V(b)-V(a)]}$$

Conjugate gradient

- When the minimization is done once, either point a or c is replaced by point x (depending on which side of b x is*),* and the minimization step is repeated.



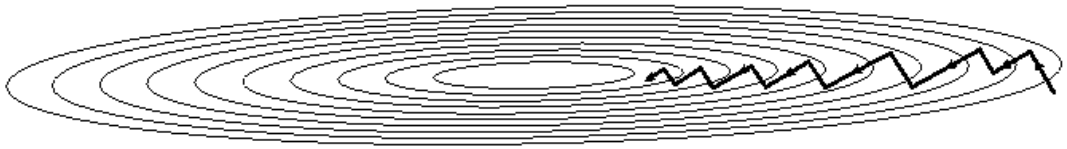
- The iteration is continued until the minimum has been found with the desired accuracy.

- A combined method: try the inverse parabolic search, but switch to the golden section if this fails.
  - One such method is the so called Brents method, which is presented in Numerical Recipes[1] (program `brent()`).

1. http://www.nr.com/

Conjugate gradient

- By combining the Steepest descent (SD)-algorithm and the Brent line minimization the energy of an atom system can be minimized. But this is still not very efficient in many dimensions. The reason is that the SD method easily winds up in a zig-zag pattern which does not move towards the minimum efficiently as in the figure below:



(a)

(b)

Conjugate gradient

- In the Conjugate gradient (CG) method the problem is solved by choosing a new "conjugate" direction of movement so that it depends on the previous direction, and does not lead to the zig-zag-pattern above.
  - What is really meant by two directions being conjugate to each other? Consider an arbitrary function $f(\mathbf{x})$ of N dimensional argument, and construct its Taylor-series around a point $\mathbf{P}$:

$$f(\mathbf{x}) = f(\mathbf{P}) + \sum_i \frac{\partial f}{\partial x_i} x_i + \frac{1}{2} \sum_{i,j} \frac{\partial^2 f}{\partial x_i \partial x_j} x_i x_j + \ldots \approx c - \mathbf{b} \cdot \mathbf{x} + \frac{1}{2} \mathbf{x} \cdot \mathbf{A} \cdot \mathbf{x}$$

$$\text{where} \quad c \equiv f(\mathbf{P}) \qquad \mathbf{b} = -\nabla f\big|_{\mathbf{P}} \qquad \mathbf{A} = \left.\frac{\partial^2 f}{\partial x_i \partial x_j}\right|_{\mathbf{P}}$$

  - The matrix $\mathbf{A}$ is the so called Hessian matrix. In this approximation the gradient of f is $\nabla f = \mathbf{A} \cdot \mathbf{x} - \mathbf{b}$, and a change in the gradient ∇f over some distance $\delta\mathbf{x}$ is again $\delta(\nabla f) = \mathbf{A} \cdot (\delta\mathbf{x})$

  - The previous direction in which we have moved is $\mathbf{u}$, gradient is $\mathbf{g}$. How to construct the next direction $\mathbf{v}$?
  - In the current point: $\mathbf{g} \perp \mathbf{u}$
  - After the next step we still want $\mathbf{g}' \perp \mathbf{u}$ $\rightarrow$ the change in the gradient $\delta(\nabla f)$ should be perpendicular to $\mathbf{u}$: $\mathbf{u} \cdot \delta(\nabla f) = 0 \Rightarrow \mathbf{u} \cdot \mathbf{A} \cdot \mathbf{v} = 0$

  - If this is valid, the directions $\mathbf{u}$ and $\mathbf{v}$ are considered to be *conjugated*.

Conjugate gradient

- In the conjugate gradient method two vectors $\mathbf{g}$ and $\mathbf{h}$ are used to calculate the new direction into which to move. $\mathbf{h}$ is the actual direction into which the line minimization is carried out.

  - In solving linear equations, these are iterated as follows:

    $\mathbf{g}_{i+1} = \mathbf{g}_i - \lambda_i(\mathbf{A} \cdot \mathbf{h}_i)$ and $\mathbf{h}_{i+1} = \mathbf{g}_{i+1} + \gamma_i \mathbf{h}_i$

    where

    $$\lambda_i = \frac{\mathbf{g}_i \cdot \mathbf{g}_i}{\mathbf{h}_i \cdot \mathbf{A} \cdot \mathbf{h}_i} = \frac{\mathbf{g}_i \cdot \mathbf{h}_i}{\mathbf{h}_i \cdot \mathbf{A} \cdot \mathbf{h}_i} \quad \text{ja} \quad \gamma_i = \frac{\mathbf{g}_{i+1} \cdot \mathbf{g}_{i+1}}{\mathbf{g}_i \cdot \mathbf{g}_i}$$

  - The vectors $\mathbf{g}$ and $\mathbf{h}$ fulfil the orthogonality and conjugation requirements:

    $$\mathbf{g}_i \cdot \mathbf{g}_j = 0 \qquad \mathbf{h}_i \cdot \mathbf{A} \cdot \mathbf{h}_j = 0 \qquad \mathbf{g}_i \cdot \mathbf{h}_j = 0$$

  - Not suitable for atomistic systems: the $N \times N$ matrix $\mathbf{A}$!

  - The crucial, saving statement is the following: if we have just minimized f in the direction $\mathbf{h}$ to some point $\mathbf{x}_{i+1}$, the new $\mathbf{g}$ can be obtained simply with

    $$\mathbf{g}_{i+1} = -\nabla f(\mathbf{x}_{i+1})$$

    and the end result corresponds to the above equations!

Conjugate gradient

- In principle this iteration algorithm gets one to an energy minimum in a system of N atoms with a memory requirement $O(N)$ and a number of iteration steps $O(N)$.

- This sounds like a problem for large numbers of atoms: if say $N = 100000$ we definitely do not want to iterate 100000 times.

- In practice the atom motion in large systems is almost always strongly correlated, and much fewer iteration steps are enough to get to a minimum.

- Typically ~ 200 steps in periodic systems and ~ 1000 steps in systems with a surface is enough to find an energy minimum with 15 digits of accuracy regardless of system size.

Conjugate gradient

- Using these equations we obtain the following algorithm for conjugate gradient energy minimization:

0. Start at point $\mathbf{r}_0$, set $i = 0$, $V_0 = V(\mathbf{r}_0)$, $\mathbf{x}_0 = -\nabla V(\mathbf{r}_0)$, $\mathbf{g}_0 = \mathbf{x}_0$, $\mathbf{h}_0 = \mathbf{x}_0$.

1. Minimize $V(\mathbf{r}_i + \alpha \mathbf{x}_i)$ with respect to the scalar α, then set $\mathbf{r}_{i+1} = \mathbf{r}_i + \alpha \mathbf{x}_i$ and evaluate $V_{i+1} = V(\mathbf{r}_{i+1})$.

2. If $V_{i+1} - V_i < \varepsilon$, quit.

3. Calculate $\mathbf{x}_i = -\nabla V(\mathbf{r}_{i+1})$ and $V_i = V(\mathbf{r}_{i+1})$.

4. Calculate $\gamma = (\mathbf{x}_i \cdot \mathbf{x}_i)/(\mathbf{g}_i \cdot \mathbf{g}_i)$

5. Set $\mathbf{g}_{i+1} = \mathbf{x}_i$.

6. Set $\mathbf{h}_{i+1} = \mathbf{g}_{i+1} + \gamma \mathbf{h}_i$ and $\mathbf{x}_{i+1} = \mathbf{h}_{i+1}$.

7. Set $i = i + 1$ and return to phase 1.

Conjugate gradient

- The above is the original, so called Fletcher-Reeves - algorithm. In some cases it is more efficient to use the so called Polak-Ribiere- version, which is identical to the above except that step 4 is:

4. Calculate $\gamma = \dfrac{(\mathbf{x}_i + \mathbf{g}_i) \cdot \mathbf{x}_i}{\mathbf{g}_i \cdot \mathbf{g}_i}$

Conjugate gradient

- The above algorithm is already a very efficient way to look for a local minimum. It also does not have anything specific to atomistic simulations; the function $V(\mathbf{r})$ can be any N-dimensional function $f(\mathbf{x})$ which has a well-defined gradient.

Material on the ACG variant of CG is from Kai Nordlund.

- In typical atomistic simulations there are special features (especially the knowledge that the atoms do have a smooth minimum) which can be utilized to optimize the algorithm, at the possible expense of generality.
- In atomistic simulations the calculation of the potential energy $V(\mathbf{r})$ is very slow, and the calculation of forces even slower.
  - In the above algorithm the line minimization-step **1** is the only step where forces are actually calculated. This step had two parts (see above):
    1. Make sure there is a minimum, and bracket it.
    2. Search it with the desired accuracy.
  - The bracketing requires at least 3 evaluations of the potential, and the Brent method line minimization typically 5-10 evaluations.
  - In atomistic systems we know, however, that the length scale is rather limited.
    - Unless the initial atom positions are really unphysical, the atoms are almost certain to be ~ 0.2 Å from the ground state position, or even closer. If we simply assume that the minimum is never farther than say 0.5 Å, we can simply get rid of step **1.** But this is clearly a bit dangerous, and still does not gain us more than 20 % or so of the efficiency.
  - It would be even better if we could get rid of the 5-10 potential evaluations needed in the Brent method. This can be achieved rather simply.

Conjugate gradient

- In this speed-up method (called ACG for reasons apparent below) we start by assuming that the 'minimum is out there'.

- The main point in the ACG method is the observation that when the original CG method line minimization of $V(\mathbf{r}_i + \alpha\mathbf{F}_i)$ with respect to the scalar α, for most steps the optimal value of the scalar α is about the same, ~ 0.05.

  - This is of course no natural constant, but seems to be valid for common Si and metal potentials. If the scalar α is almost the same in any case, it does not seem sensible to optimize it separately every time.

- So the method is as follows:

  - Set initially $\alpha = 0.05$.
  - For every step move forwards by $\alpha\mathbf{F}_i$.
  - If the potential energy goes down, increase optimistically α a bit.
  - If the potential energy goes up, disregard the previous step, decrease α and repeat the same iteration.

Conjugate gradient

- Because of the optimization of α the method might be called adaptive conjugate gradient, ACG:

0. Start from $\mathbf{r}_0$, set $i = 0$, $\mathbf{F}_0 = -\nabla V(\mathbf{r}_0)$, $\mathbf{g}_0 = \mathbf{F}_0$, $\mathbf{h}_0 = \mathbf{F}_0$, $\alpha = 0.05$
1. Store old $\mathbf{r}_i \rightarrow \mathbf{r}_i^{\text{prev}}$
2. Set $\mathbf{r}_{i+1} = \mathbf{r}_i + \alpha \mathbf{F}_i$
3. Calculate $V_{i+1} = V(\mathbf{r}_{i+1}), \quad \mathbf{F}_i = -\nabla V(\mathbf{r}_{i+1})$.
4. If $V_{i+1} > V_i$ return $\mathbf{r}_i^{\text{prev}} \rightarrow \mathbf{r}_i$, set $\alpha = \alpha/2$, return to step 2.
5. If $V_{i+1} - V_i < \varepsilon$, quit.
6. Calculate $\gamma = \dfrac{(\mathbf{x}_i + \mathbf{g}_i) \cdot \mathbf{x}_i}{\mathbf{g}_i \cdot \mathbf{g}_i}$ (Polak-Ribierre)
7. Set $\mathbf{g}_{i+1} = -\mathbf{F}_i$
8. Set $\mathbf{h}_{i+1} = \mathbf{g}_{i+1} + \gamma \mathbf{h}_i$ and $\mathbf{F}_{i+1} = \mathbf{h}_{i+1}$.
9. Increase $\alpha = 1.05\alpha$, set $i = i + 1$ and return to step 1.

- Here the constants 0.5 and 1.05 were optimized for Stillinger-Weber Si.

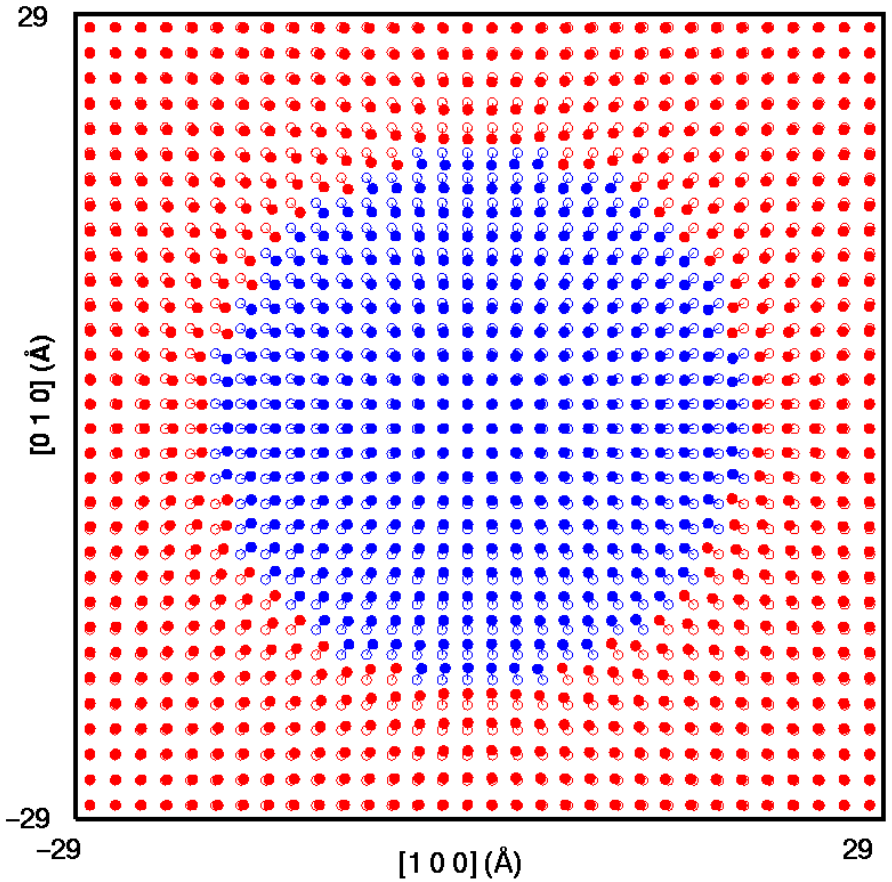
Conjugate gradient

- Written in this way the method usually needs only one potential evaluation per iteration step, except when the energy increases. In practice the energy decreases almost always, so on average the number of potential evaluations still is only about 1.1 / iteration. In the ordinary CG method this value is about 10, so in the ACG each iteration step is about 10 times faster than in CG!

- On the other hand, the ACG loses the perfect match of conjugate directions, so it needs more iterations. Still, the overall speedup of ACG vs. CG is almost always a factor of ~ 3-5.

Conjugate gradient

- As an example a 40 Å diameter Co-nanocluster in a 16×16×16 unit cell periodic Cu cell was created, and relaxed this system with EAM potentials with different methods. These calculations (in larger cells) are useful in understanding the energetics of Co nanoclusters.



- The figure above shows the atom displacements due to the minimization, but so that the displacements have been exaggerated by a factor of 3. The open circles are the original atom positions, the closed circles the final positions after minimization.
- The blue atoms are Co, red Cu. Because Co has a smaller equilibrium nearest-neighbour distance than Cu, the atoms move inwards.

Conjugate gradient

- The simulation results were as follows (computer ~ 400 Mhz Pentium[1] Linux):

Method	Et (eV)	Niter	Final E (eV)	Simulation time (s)
SD Plain	0.001	227	-59927.160	2684.20
SD Adaptive	0.001	172	-59927.052	323.56
CG Plain	0.001	27	-59927.193	363.03
CG No bracketing	0.001	27	-59927.193	251.98
ACG	0.001	70	-59927.194	128.34
MD btctau=70 fs	-	250	-59927.169	390.25

SD= Steepest Descent
CG= Conjugate gradient
MD= Molecular dynamics.

`Et` is the energy tolerance
`Niter` the number of iterations

  - We see that all methods give essentially the same result, as they should. The 0.1 eV differences may be shifts in the position of a single atoms, and hence not likely to be a significant problem.
  - The SD method with line minimization is very slow, as expected. The number of iterations is clearly the smallest in the CG methods, but they are still ~ 3 times slower than ACG.
  - A bit surprising is that the adaptive SD method is in fact faster than straight CG, and that ordinary MD is almost as fast as straight CG or adaptive SD.
  - But the ACG method clearly beats all the others by a factor of 3 or more.
- However, in a new minimization problem it is best to first implement the full CG method. After that, one can check whether it can be optimized for the particular range of problems, e.g. by a scheme similar to the one above.

1. I know, this should be updated :-)

Genetic algorithms

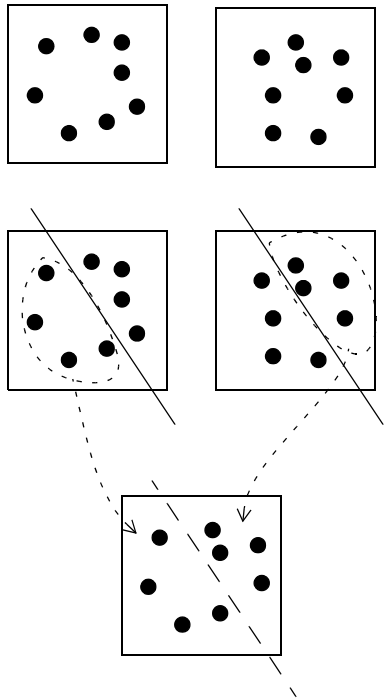
- Genetic algorithms (GA) are a popular method for looking for a global minimum, which have not been used too much in the physical sciences. They are, however, well suited at least for looking for the minimum of a fairly large set of atoms.
- Groups of atoms typically have a fairly large set of energy minima, so the ordinary methods are not well applicable for looking for a global minimum: CG only looks for the closest local minimum, and MD and Monte Carlo (simulated annealing) are fairly easily stuck to one minimum or a local region.
  - But the GA method is well suited for looking for global minima, at least for dilute atom systems.

- Genetic algorithms have obtained their inspiration from Darwin's theory of evolution.
  - The idea is to perform natural selection for some group of parameters G which describes well the real system.
  - The group is allowed to breed by mating, after which natural selection is carried out (i.e. the poorest adapted species are killed).
  - The parameters G can be considered to correspond to a gene sequence, DNA.

- Here we present the Deaven and Ho approach to genetic algorithms for atoms [Deaven and Ho, *Phys. Rev. Lett.* **75** (1995) 288].
  - Let us state the problem as follows. We have N atoms in free space, and want to find their minimum-energy configuration. The parameter set is now simply the set of atom coordinates $G = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$.
  - We illustrate there the algorithm with 2D figures; in reality it of course usually is in 3D. The difference between 2D and 3D is trivial.

Genetic algorithms

- **Deaven and Ho genetic algorithm:**



0. Start. Create random initial positions for structures, each with exactly N atoms.
1. Mating and breeding. Select two well-adjusted parents for breeding. This is done by selecting a given parent i with state G_i with the probability

$$P(G_i) \propto e^{-E(G_i)/T_\mathrm{m}}$$

where the mating 'temperature' T_m is selected as the range of energies among the whole population $\{G_i\}$. Split the two parent structures along the same line. Take one half of one parent, and another half of another parent, and join them together.
Here the added complication that a child may have a different number of atoms than the parents comes in. In this case, the lines creating the two parents are moved in opposite directions until a state where the child has equal numbers of parents is found.
2. Mutation. With a probability μ perform a mutation on the child. There are two possible kinds of mutations:

a) Move atoms in a random direction by a random distance a random number of times. The distance is of the order of the bond length, and the number of times ~ 5 - 50.
b) Move an atom *up* along the potential energy function. (Try to move over potential barriers.)

3. Minimize the energy of the child to the closest local minimum. This is done by CG or MD.
4. Natural selection. If the child has lower energy than any of the parents, allow it to stay alive. Then check that its energy does not match the energy of any parent within an energy range δE. If this is true, include it in the population, and kill the least-well adapted parent (the one with the highest E).
5. Convergence test. If convergence has not been reached, return to stage 1.

Genetic algorithms

- The energy range δE is included to prevent the population from having several identical or very similar structures.
- The mutation operation can sometimes be completely left out.
- The mating temperature reduces the probability that poorly adjusted parents get to breed. Hence they are more likely to die without giving rise to any offspring.
  - If $T_{\mathrm{m}} \gg E_{\mathrm{max}}$ all parents get to breed by about the same probability. If on the other hand $T_{\mathrm{m}} \ll E_{\mathrm{max}}$ only the best adjusted parents get to breed. (Even this can be seen to have a biological interpretation, although not a very good one: in warm climates it is easier to survive, whereas in harsher, colder climates only the best adjusted individuals can survive and breed...)

- The size of the population does not have to be very large. With Deaven and Ho, who used TB, had it usually at 4. Jura Tarus found that somewhat larger numbers work better for the Tersoff C potential.

- Deaven and Ho used their code to find the equilibrium structure for a fullerene C_{60} and other small carbon clusters, starting from random atom coordinates.
  - No other simulation method had at that time been able to produce a fullerene 'from scratch' .
  - Chelikowsky got close with MD [*Phys. Rev. Lett.* **67** (1991) 2970.], but using a to-say-the least suspicious bond-bending part in his potential.
  - Simulated annealing (a Monte Carlo method) can find the structure of molecules of the order of C_{20} , but not larger than that.

- Deaven and Ho used a Tight Binding force model, which was known to describe fullerenes well.

- Parameters: mating temperature $T_{\mathrm{m}} = 0.2\ \mathrm{eV/atom}$, energy resolution $\delta E = 0.01\ \mathrm{eV}$, population $p = 4$.

Genetic algorithms

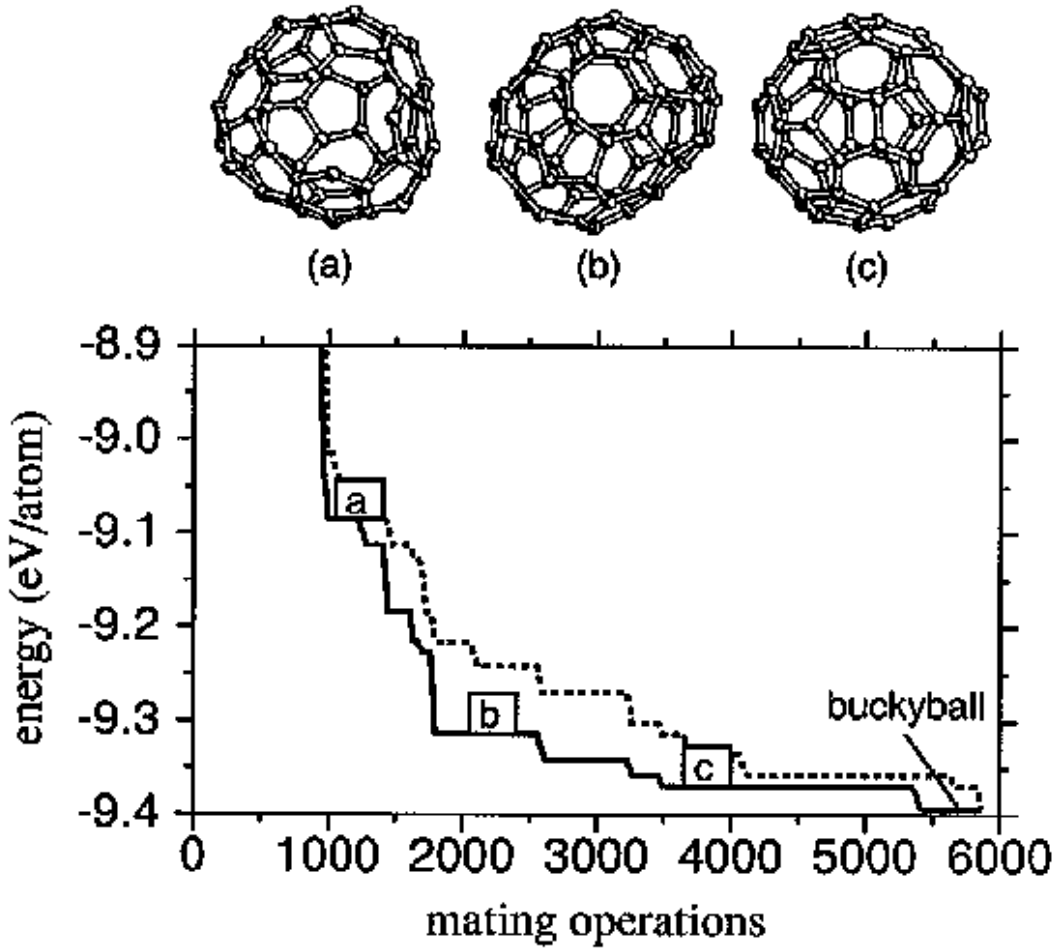
- **Fullerene C_{60} .** The algorithm finds a perfect fullerene after about 6000 mating operations starting from random coordinates, without mutations ($\mu = 0$) :

- The upper curve is the maximum energy of the population, the lower the minimun.

- We see that fairly fast (1000 mating operations) a fairly well-adjusted state (a) which still has defects (a 12-membered atom ring and two 7-membered atom rings).

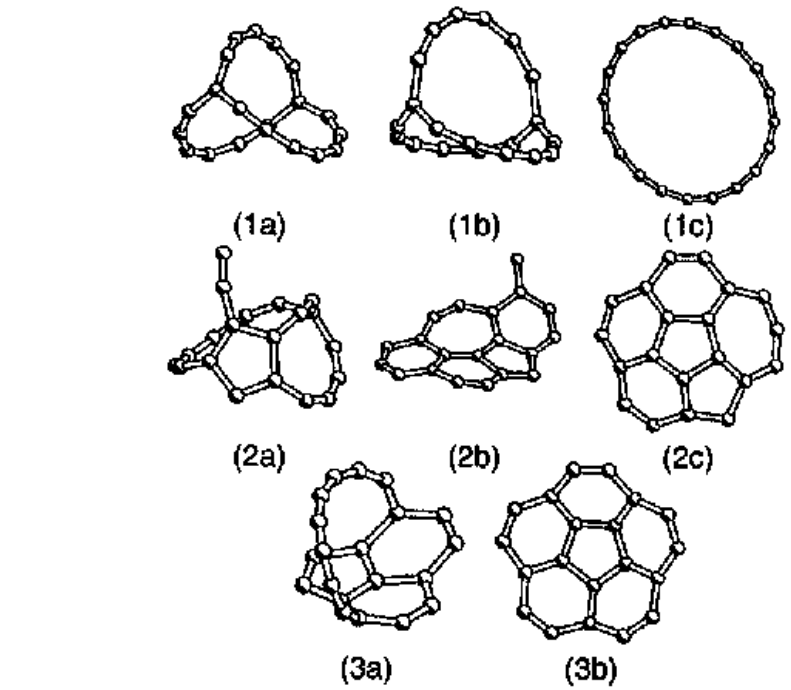
- A large fraction of the time, about 5000 mating operations, goes to removing the last defects.

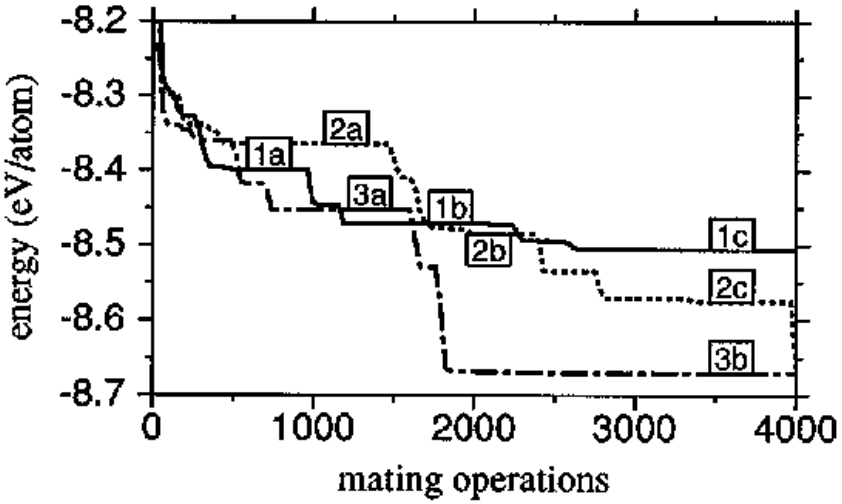
- In stage (b) there is still left a 7-atom ring, and in state (c) there already is the correct amount of pentagons and hexagons, but two pentagons adjacent to each other.



Genetic algorithms

- **Carbon cluster C_{20}**. When the genetic algorithm is run for 20 carbon atoms, the effect of mutations becomes apparent:
  - States 1 a-c and the solid line describe the results when the code is ran without mutations.
    - The structure is stuck in a round carbon circle.
    - But some $\mu = 0$ -states do find the correct structure.
  - In states 2 a -c and 3 a-b $\mu = 0.05$.
    - Now the structures find fairly fast the lowest-energy bowl form.
    - State 2 c is already close to the ground state, but the rings on the side have 5 or 7 atoms.
    - State 3 b is the correct ground state, with only 6-membered atom rings.





Genetic algorithms

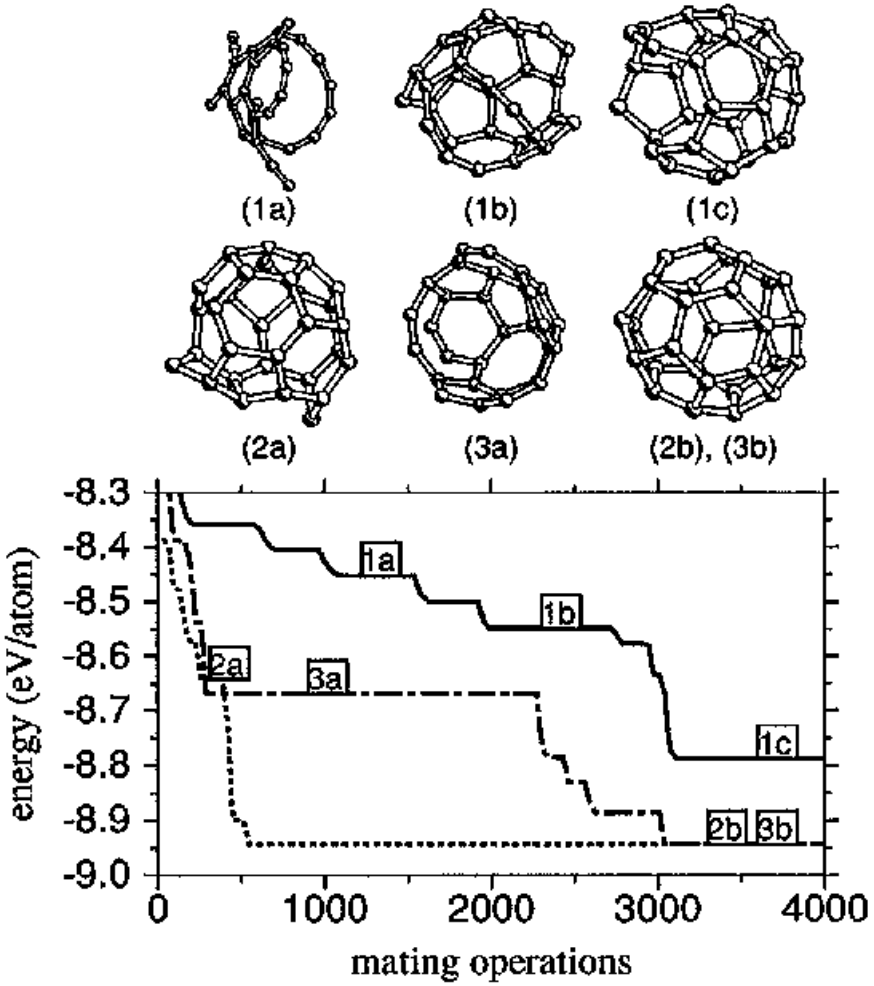
- **Carbon cluster C_{30}**

  - Most runs end up in the correct state, but some of the $\mu = 0$ states do not in 4000 mating operations found the ground state, but get stuck in state (1c).

  - With $\mu = 0.05$ almost all states end up in the correct cage structure (2b) and (3b).

  - The intermediate configurations (2a) and (3a) show that the correct final state can be reached in several different ways.

  - Only mutation, with no mating, does not lead to the correct state.



Genetic algorithms

- Here it is important to realize that the development of the GA minimization *process* does not necessarily contain any physically meaningful information.
  - The real path to the ground state probably has no relation to the GA path.
  - So only the ground state found by GA may correspond to real life (in case the experimental situation has had time to reach the ground state).
  - You probably remember: This same note applies to equilibrium MC simulations.

Genetic algorithms

- In the original way of realizing GA the information on the state is coded in a binary “gene sequence” which corresponds to DNA.

  - Let us consider the interaction between two molecules A and B [Xiao and Williams, *Chem. Phys. Lett.* **215** (1993) 17]. Both molecules can be described with a position and rotation angle, so the information needed $(x, y, z, \alpha, \theta, \phi)$.

  - If we now discretize the possible positions and angles, using e.g. 16 possibilities for each dimension, the state of the molecule can be described with 24 bits of information, for instance

    (4.5 Å,5.0 Å,9.0 Å,120^{o}, 100^{o},60^{o})=(1001:1010:1110:0110:0101:0011).

  - The breeding operation is defined such that the binary string is exchanged from some point forward (“crossover”). So if we have two parents

    P1 = (1001:1010:1110:0110:0101:0011)
    P2 = (1001:1010:1110:0100:1011:1110)

    and the exchange position is chosen to be 17, we get the children

    C1 = (1001:1010:1110:0110:0101: 1110)
    C2 = (1001:1010:1110:0100:1011: 0011)

Genetic algorithms

- In this case stage **1.** in the above algorithm simply becomes.

  1. Mating and breeding. Exchange the gene sequence of a parents with another starting from a random position.

- The mutation operation now becomes simply

  2. Mutation. With a given probability μ exchange the state of a bit ($0 \rightarrow 1$ or $1 \rightarrow 0$) for all bits in all individuals.

- Because a bit corresponds to a position or rotation angle, this directly changes the state of the individual.

- Otherwise the algorithm is essentially as that of Deaven and Ho.

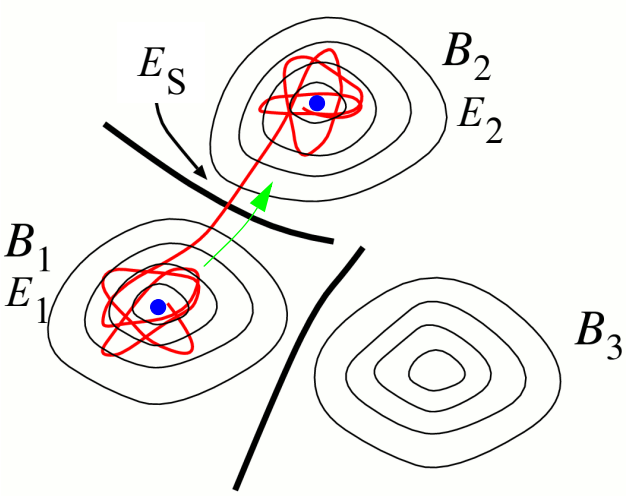
- Xiao used the algorithm to search for the ground state configurations for simple hydrocarbon molecules such as the benzene dimer. He used a population of 100 and 8 bits to code each position or angle.

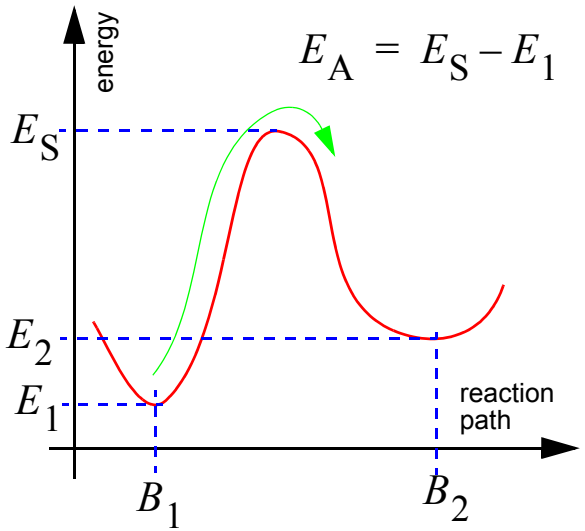
- But this approach has the problem that during the mating and mutation the state of the molecule can change quite radically, and the properties of the parents are not transferred to the children. Hence Deaven and Ho say that their method is better for optimizing atomic structure.

- GA has been applied in physics particularly in studying equilibrium structure of small clusters. [See e.g. K. M. Ho *et al.*, *Nature* **392** (1998) 582; D. M. Deaven *et al.*, *Chem. Phys. Lett.* **256** (1996)**;** J. Zhuang, *et al.*, *Phys. Rev. B* **69** (2004).]

Reaction (or minimum energy) path determination

- Thermally activated atomistic processes
  - Need to know the transition rate (events/unit time) for $B_1 \rightarrow B_2$.
  - If the probability for the event is not too low direct MD simulation is possible.
  - For really **rare events** transition state theory (TST) can be used.
    - Rate can be written in form $\nu = \nu_0 e^{-E_A/k_B T}$ $\rightarrow$ need to know the activation energy $E_A = E_S - E_1$, where E_S is so called saddle point energy.
    - From TST one can also get an estimate for the prefactor ν_0 based on vibrational properties: $\nu_0 = \prod_{i=1}^{N} \nu_i / \prod_{i=1}^{N-1} \nu'_i$, where ν_i, and ν'_i are the vibration frequencies at B_1 and the saddle point, respectively. Note that at the saddle point there is one less frequency compared to the local minimum B_1.
      - Quite often the prefactor is simply set to a typical vibration frequency in the system: $\nu_0 \approx 10^{12} s^{-1}$.





- Exactly E_A is defined as the maximum energy along the path with lowest energy (minimum energy path; **MEP**) going from B_1 to B_2 (local minima; blue dots).
  - Path here means a line in the $3N$ dimensional configuration space.

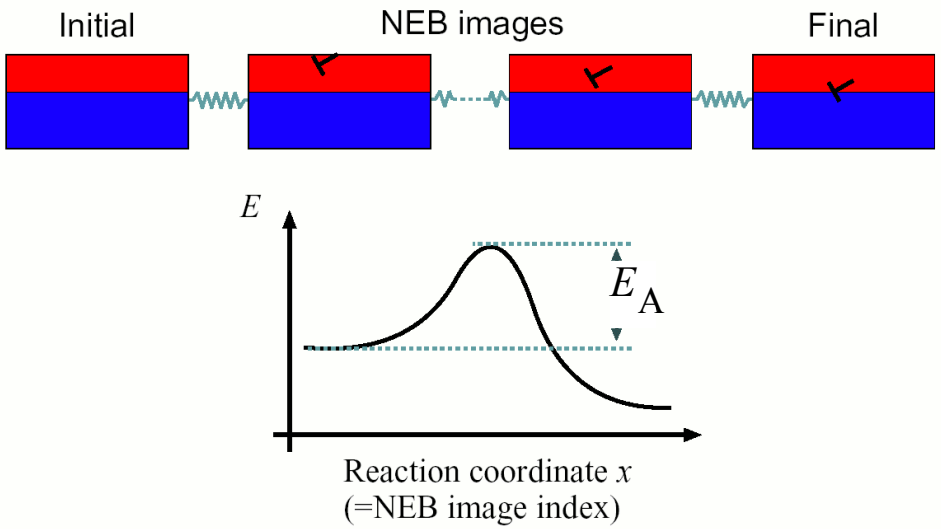
Reaction (or minimum energy) path determination

- There are many methods to do this; one of the most often used in atomistic systems is the Nudged Elastic Band method or NEB. [G. Henkelman, H. Jónsson, *J. Chem. Phys.* 113 (2000) 9901.; G. Henkelman, H. Jónsson, *J. Chem. Phys.* 113 (2000) 9978.]

- In NEB images of the system are created by interpolating the atomic coordinates between the initial and final configurations (that are usually local minima).

- Every image is connected by a spring force to its neighboring images. (End points are fixed.)

- The spring force prevents all images to fall to the nearest local potential energy minimum.



Reaction (or minimum energy) path determination

- The total force on the atoms in image i is calculated as

$$\mathbf{F}_i = \mathbf{F}^{\mathrm{S}}_{i,\parallel} - \nabla V(\mathbf{R}_i)_\perp ,$$

$\mathbf{R}_i$ is the $3N$ dimensional vector of atom coordinates in image i.

- The first term is the spring force which acts only in the tangential direction of the image chain:

$$\mathbf{F}^{\mathrm{S}}_{i,\parallel} = k[\left|\mathbf{R}_{i+1} - \mathbf{R}_i\right| - \left|\mathbf{R}_i - \mathbf{R}_{i-1}\right|]\hat{\tau}_i ,$$

where k is the spring constant and τ_i is the tangent vector of the image chain:

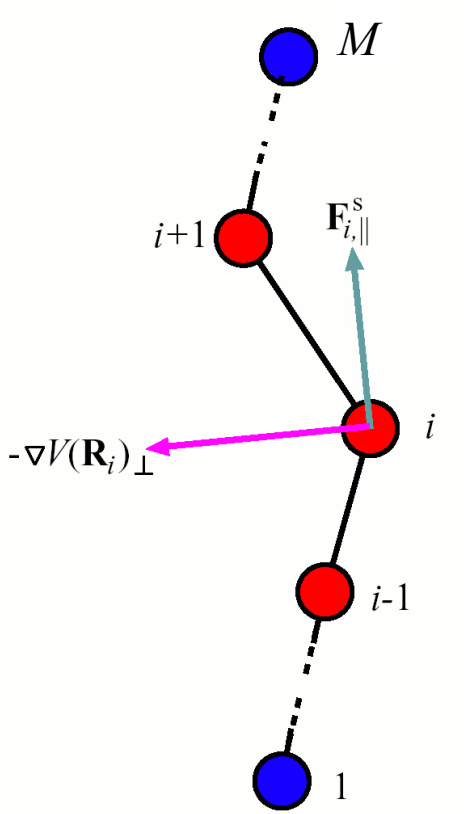
$$\tau_i = \begin{cases} \tau_i^+ & \text{if } V_{i+1} > V_i > V_{i-1} \\ \tau_i^- & \text{if } V_{i+1} < V_i < V_{i-1} \end{cases}, \quad \tau_i^+ = \mathbf{R}_{i+1} - \mathbf{R}_i, \quad \tau_i^- = \mathbf{R}_i - \mathbf{R}_{i-1}$$

- When the middle image is the minimum or maximum of the three the tangent is calculated as

$$\tau_i = \begin{cases} \tau_i^+ \Delta V_i^{\max} + \tau_i^- \Delta V_i^{\min}, & \text{if } V_{i+1} > V_{i-1} \\ \tau_i^+ \Delta V_i^{\min} + \tau_i^- \Delta V_i^{\max}, & \text{if } V_{i+1} < V_{i-1} \end{cases},$$

$$\Delta V_i^{\max} = \max(\left|V_{i+1} - V_i\right|, \left|V_{i-1} - V_i\right|)$$

$$V_i^{\min} = \min(\left|V_{i+1} - V_i\right|, \left|V_{i-1} - V_i\right|$$



Every image has N atoms. Number of images M (including the end points).

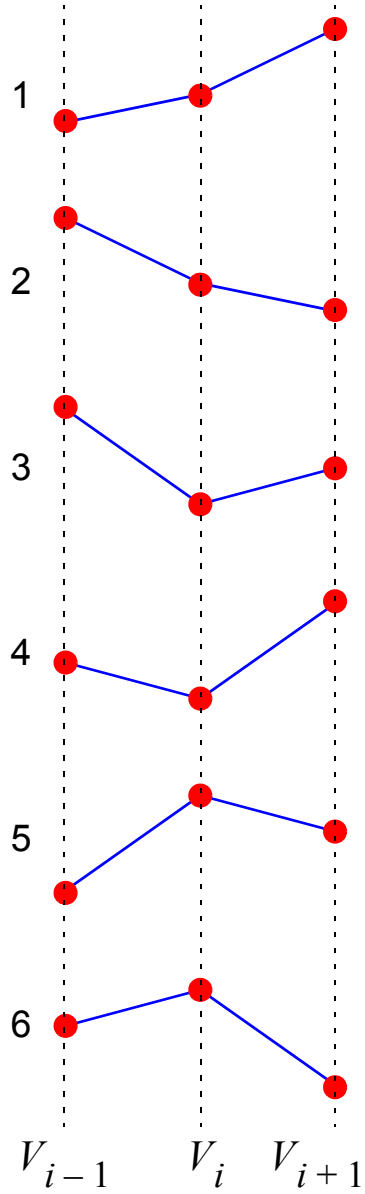
Reaction (or minimum energy) path determination

- The second term is calculated from the potential energy model of the system:

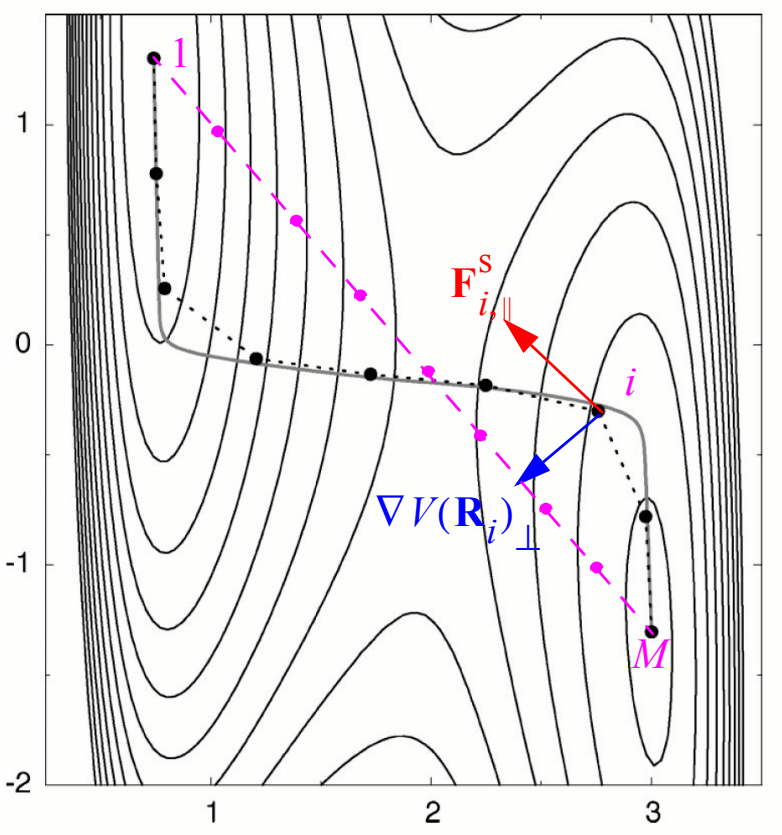
$$\nabla V(\mathbf{R}_i)_{\perp} = \nabla V(\mathbf{R}_i) - \nabla V(\mathbf{R}_i)\hat{\tau}_i \cdot \hat{\tau}_i$$

- When calculating the tangent on has to take into account all the six energy configurations of the three neighbor images shown on the right.

- The spring force tries to keep the images in the chain evenly spaced.
- The potential force is there to find the minimum energy of all images in the direction perpendicular to the image chain (=reaction path).



Reaction (or minimum energy) path determination



Let's illustrate NEB by a simple 2D potential energy surface shown on the left [G. Henkelman, H. Jónsson, *J. Chem. Phys.* **113** (2000) 9978.]

- Solid line: the real MEP
- Dashed line (magenta): initial configuration for NEB (interpolated)
- Dotted line with circles: the path obtained by NEB

Reaction (or minimum energy) path determination

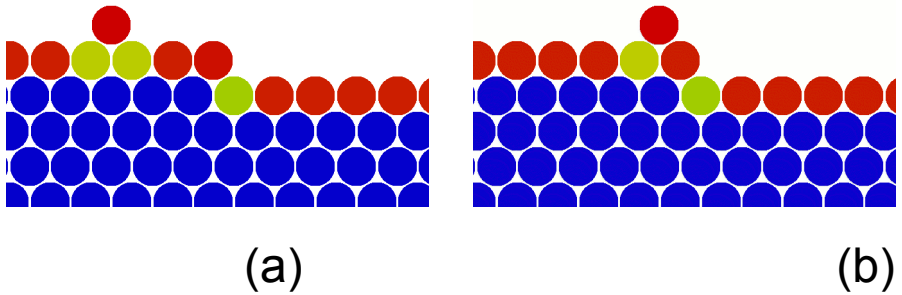
- Running a NEB simulation:
  - Create the end points by optimizing the two configurations by e.g. CG of cooling-MD.
  - Interpolate the images and remove atom overlaps.
  - Find the minimum energy path by optimizing the image system by applying the forces described above.
- Modifying an existing MD code for NEB is not difficult:
  - Input the coordinates of the image chain.
  - When calculating neighbor list skip atom pairs that belong to different images.
  - Add the calculation of tangent τ_i,
    - The inter-image distance is calculated simply as

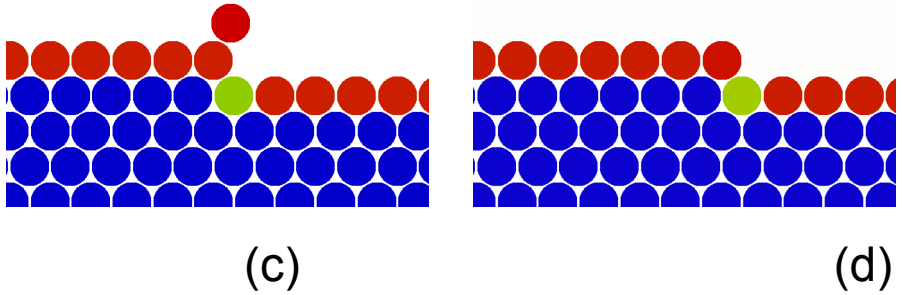
$$\left|\mathbf{R}_i - \mathbf{R}_{i-1}\right|^2 = \sum_{j=1}^{N} [(x_{i,j} - x_{i,j-1})^2 + (y_{i,j} - y_{i,j-1})^2 + (z_{i,j} - z_{i,j-1})^2],$$

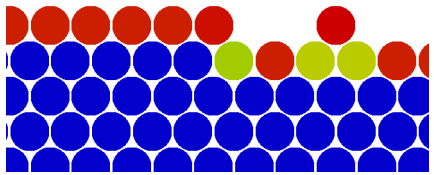
    where $\mathbf{r}_{i,j}$ is the position of the jth atom in the ith image.
  - Add the calculation of the spring force.
  - Modify the force routine to calculate only the perpendicular component of the force.
  - The only parameter is the spring force constant k. Fortunately, calculation is rather insensitive to its value.

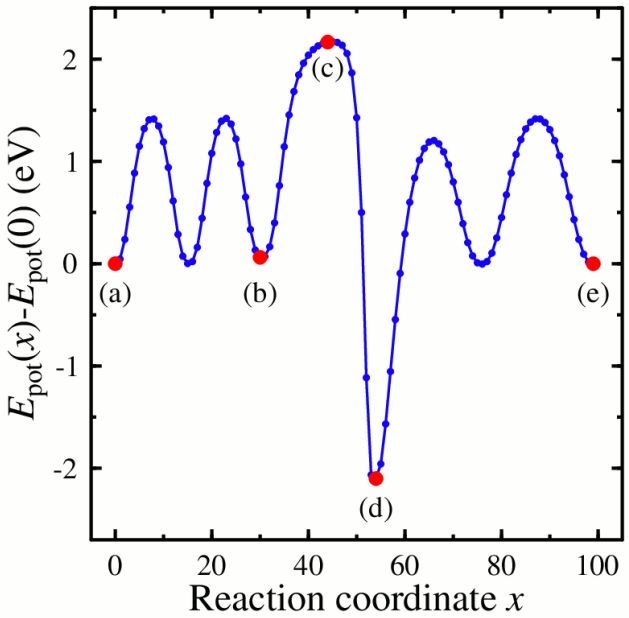
Reaction (or minimum energy) path determination

- A simple example: surface diffusion in a 2D Lennard-Jones system









Difficult to jump down from the step: Erlich-Schwoebel barrier → surface growth instabilities.