Ionic optimisation

Georg KRESSE

Institut für Materialphysik and Center for Computational Material Science

Universität Wien, Sensengasse 8, A-1090 Wien, Austria

UNIVERSITÄT  WIEN

Overview

- the mathematical problem
  - minimisation of functions
  - rule of the Hessian matrix
  - how to overcome slow convergence
- the three implemented algorithms
  - Quasi-Newton (DIIS)
  - conjugate gradient (CG)
  - damped MD

  strength, weaknesses
- a little bit on molecular dynamics

The mathematical problem

- search for the local minimum of a function $f(\vec{x})$

  for simplicity we will consider a simple quadratic function

$$f(\vec{x}) = a + \vec{b}\vec{x} + \frac{1}{2}\vec{x}\mathbf{B}\vec{x} = \bar{a} + \frac{1}{2}(\vec{x} - \vec{x}^0)\mathbf{B}(\vec{x} - \vec{x}^0),$$

  where **B** is the Hessian matrix

$$\mathbf{B}_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}.$$

- for a stationary point, one requires

$$\vec{g}(\vec{x}) = \frac{\partial f}{\partial \vec{x}} = \mathbf{B}(\vec{x} - \vec{x}^0)$$

$$g_i(\vec{x}) = \frac{\partial f}{\partial x_i} = \sum_j \mathbf{B}_{ij}(x_j - x_j^0)$$

  at the minimum the Hessian matrix must be additionally positive definite

The Newton algorithm

educational example

- start with an arbitrary start point $\vec{x}^1$
- calculate the gradient $\vec{g}(\vec{x}^1)$
- multiply with the inverse of the Hessian matrix and perform a step

$$\vec{x}^2 = \vec{x}^1 - \mathbf{B}^{-1}\vec{g}(\vec{x}^1)$$

  by inserting $\vec{g}(\vec{x}^1) = \frac{\partial f}{\partial \vec{x}} = \mathbf{B}(\vec{x}^1 - \vec{x}^0)$, one immediately recognises that $\vec{x}^2 = \vec{x}^0$ hence one can find the minimum in one step

- in practice, the calculation of **B** is not possible in a reasonable time-span, and one needs to approximate **B** by some reasonable approximation

Steepest descent

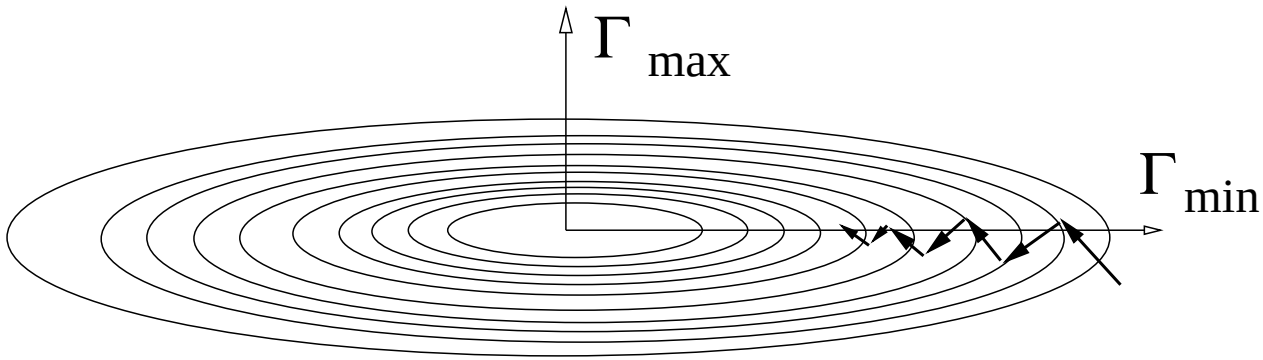
approximate **B** by the largest eigenvalue of the Hessian matrix $\rightarrow$ steepest descent algorithm (Jacobi algorithm for linear equations)

1. initial guess $\vec{x}^1$
2. calculate the gradient $\vec{g}(\vec{x}^1)$
3. make a step into the direction of the steepest descent

$$\vec{x}^2 = \vec{x}^1 - 1/\Gamma_{max}(B)\vec{g}(\vec{x}^1)$$

4. repeat step 2 and 3 until convergence is reached

for functions with long steep valleys convergence can be very slow



Speed of convergence

how many steps are required to converge to a predefined accuracy

- assume that **B** is diagonal, and start from $\vec{x}^1 = \vec{x}^0 + \mathbf{1}$

$$\mathbf{B} = \begin{pmatrix} \Gamma_1 & & 0 \\ & \ldots & \\ 0 & & \Gamma_n \end{pmatrix} \qquad \vec{x}^1 = \vec{x}^0 + \begin{pmatrix} 1 \\ \ldots \\ 1 \end{pmatrix} \qquad \text{with } \Gamma_1 < \Gamma_2 < \Gamma_3 \ldots$$

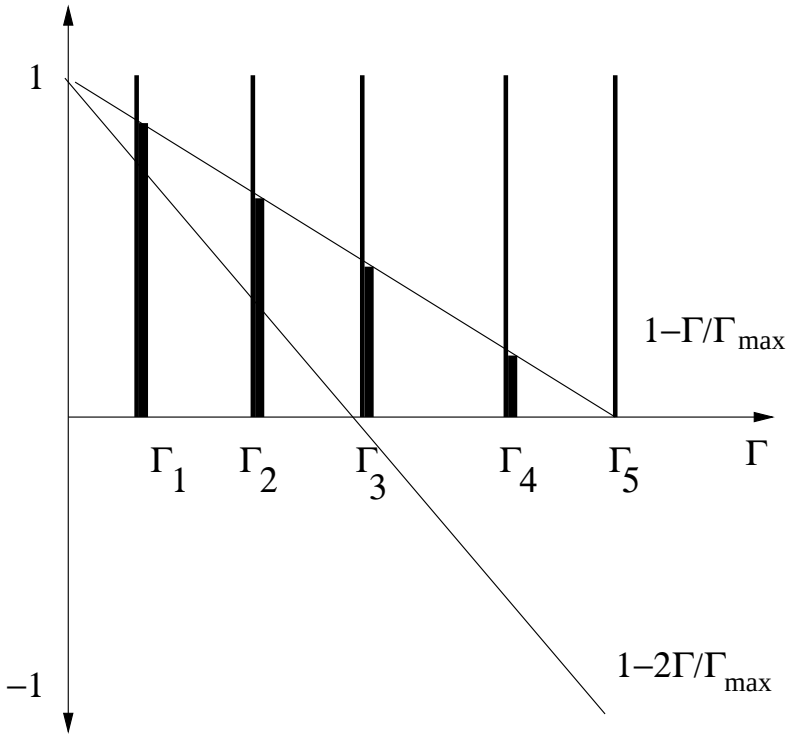
- gradient $\vec{g}(\vec{x}^1)$ and $\vec{x}^2$ after steepest descent step are:

$$\vec{g}(\vec{x}^1) = \mathbf{B}(\vec{x}^1 - \vec{x}^0) = \begin{pmatrix} \Gamma_1 \\ \ldots \\ \Gamma_n \end{pmatrix} \qquad \vec{x}^2 = \vec{x}^1 - \frac{1}{\Gamma_n}\vec{g}(\vec{x}^1) = \vec{x}^0 + \begin{pmatrix} 1 - \Gamma_1/\Gamma_n \\ \ldots \\ 1 - \Gamma_n/\Gamma_n \end{pmatrix}$$

Convergence

- the error reduction is given by

$$\vec{x}^2 = \vec{x}^0 + \begin{pmatrix} 1-\Gamma_1/\Gamma_n \\ \dots \\ 1-\Gamma_n/\Gamma_n \end{pmatrix}$$



- – the error is reduced for each component
  - – in the high frequency component the error vanishes after on step
  - – for the low frequency component the reduction is smallest

- the derivation is also true for non-diagonal matrices

  in this case, the eigenvalues of the Hessian matrix are relevant

- for ionic relaxation, the eigenvalues of the Hessian matrix correspond to the vibrational frequencies of the system

  the highest frequency mode determines the maximum stable step-width (“hard modes limit the step-size”)
  but the soft modes converge slowest

- to reduce the error in all components to a predefined fraction ε,
  k iterations are required

$$\left(1-\frac{\Gamma_{\min}}{\Gamma_{\max}}\right)^{k} \approx \varepsilon$$

$$k \ln\left(1-\frac{\Gamma_{\min}}{\Gamma_{\max}}\right) \approx \ln\varepsilon$$

$$-k\frac{\Gamma_{\min}}{\Gamma_{\max}} \approx \ln\varepsilon \quad \Rightarrow \quad k \approx -(\ln\varepsilon)\frac{\Gamma_{\max}}{\Gamma_{\min}} \qquad k \propto \frac{\Gamma_{\max}}{\Gamma_{\min}}$$

Pre-conditioning

- if an approximation of the inverse Hessian matrix is know $\mathbf{P} \approx \mathbf{B}^{-1}$, the convergence speed can be much improved

$$\vec{x}^{N+1} = \vec{x}^{N} - \lambda \mathbf{P} \vec{g}(\vec{x}^{N}).$$

- in this case the convergence speed depends on the eigenvalue spectrum of

$$\mathbf{PB}$$

- for $\mathbf{P} = \mathbf{B}^{-1}$, the Newton algorithm is obtained

Variable-metric schemes, Quasi-Newton scheme

variable-metric schemes maintain an iteration history

they construct an implicit or explicit approximation of the inverse Hessian matrix

$$\mathbf{B}^{-1}_{\text{approx}}.$$

search directions are given by

$$\mathbf{B}^{-1}_{\text{approx}}\vec{g}(\vec{x}).$$

the asymptotic convergence rate is give by

$$\text{number of iterations} \propto \sqrt{\frac{\Gamma_{\text{max}}}{\Gamma_{\text{min}}}}$$

Simple Quasi-Newton scheme, DIIS

direct inversion in the iterative subspace (DIIS)

- set of points

$$\{\vec{x}^i | i = 1, ..., N\} \quad \text{and} \quad \{\vec{g}^i | i = 1, ..., N\}$$

- search for a linear combination of x^i which minimises the gradient, under the constraint

$$\sum_i \alpha_i = 1,$$

-

$$\begin{aligned} \vec{g}(\sum_i \alpha^i \vec{x}^i) &= \mathbf{B}\left(\sum_i \alpha^i \vec{x}^i - \vec{x}^0\right) = \mathbf{B}\left(\sum_i \alpha^i \vec{x}^i - \sum_i \alpha^i \vec{x}^0\right) \\ &= \sum_i \alpha_i \mathbf{B}(\vec{x}^i - \vec{x}^0) = \sum_i \alpha_i \vec{g}^i. \end{aligned}$$

  gradient is linear in it's arguments for a quadratic function

Full DIIS algorithm

1. single initial point $\vec{x}^1$
2. gradient $\vec{g}^1 = \vec{g}(\vec{x}^1)$, move along gradient (steepest descent)

$$\vec{x}^2 = \vec{x}^1 - \lambda \vec{g}^1$$

3. calculate new gradient $\vec{g}^2 = \vec{g}(\vec{x}^2)$
4. search in the space spanned by $\{\vec{g}^i | i = 1, \ldots, N\}$ for the minimal gradient

$$\vec{g}_{\text{opt}} = \sum \alpha^i \vec{g}^i .$$

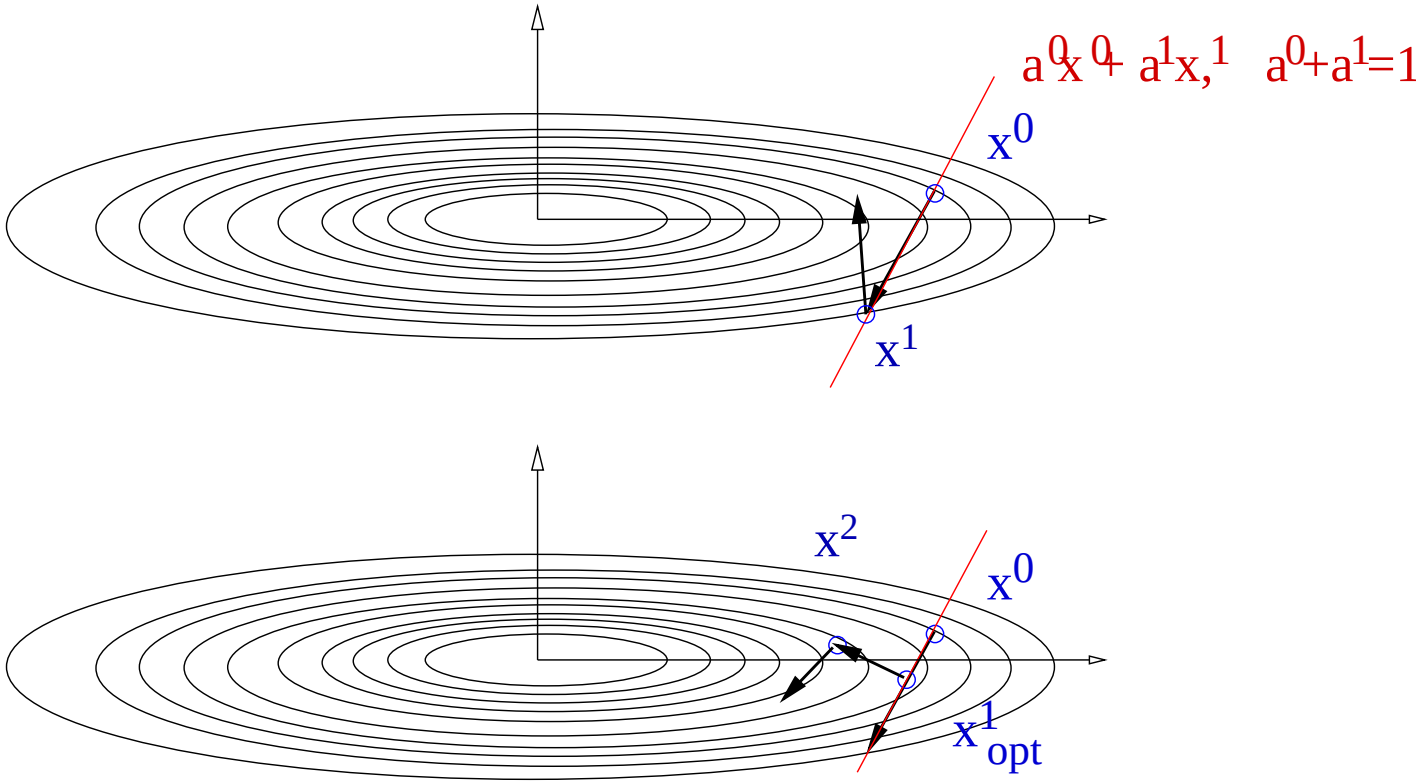
   and calculate the corresponding position

$$\vec{x}_{\text{opt}} = \sum \alpha^i \vec{x}^i$$

5. Construct a new point $\vec{x}^3$ by moving from $\vec{x}_{\text{opt}}$ along $\vec{g}_{\text{opt}}$

$$\vec{x}^3 = \vec{x}_{\text{opt}} - \lambda \vec{g}_{\text{opt}}$$

1. steepest descent step from $\vec{x}^0$ to $\vec{x}^1$ (arrows correspond to gradients $\vec{g}_0$ and $\vec{g}_1$)
2. gradient along indicated red line is now know, determine optimal position $\vec{x}^1_{\text{opt}}$
3. another steepest descent step form $\vec{x}^1_{\text{opt}}$ along $\vec{g}_{\text{opt}} = \vec{g}(\vec{x}^1_{\text{opt}})$
4. calculate gradient $x^2 \Rightarrow$ now the gradient is known in the entire 2 dimensional space (linearity condition) and the function can be minimised exactly



Conjugate gradient

first step is a steepest descent step with line minimisation

search directions are "conjugated" to the previous search directions

1. gradient at the current position $\vec{g}(\vec{x}^N)$
2. conjugate this gradient to the previous search direction using:

$$\vec{s}^N = \vec{g}(\vec{x}^N) + \gamma \vec{s}^{N-1} \qquad \gamma = \frac{(\vec{g}(\vec{x}^N) - \vec{g}(\vec{x}^{N-1})) \cdot \vec{g}(\vec{x}^N)}{(\vec{g}(\vec{x}^{N-1})) \cdot \vec{g}(\vec{x}^{N-1})}$$

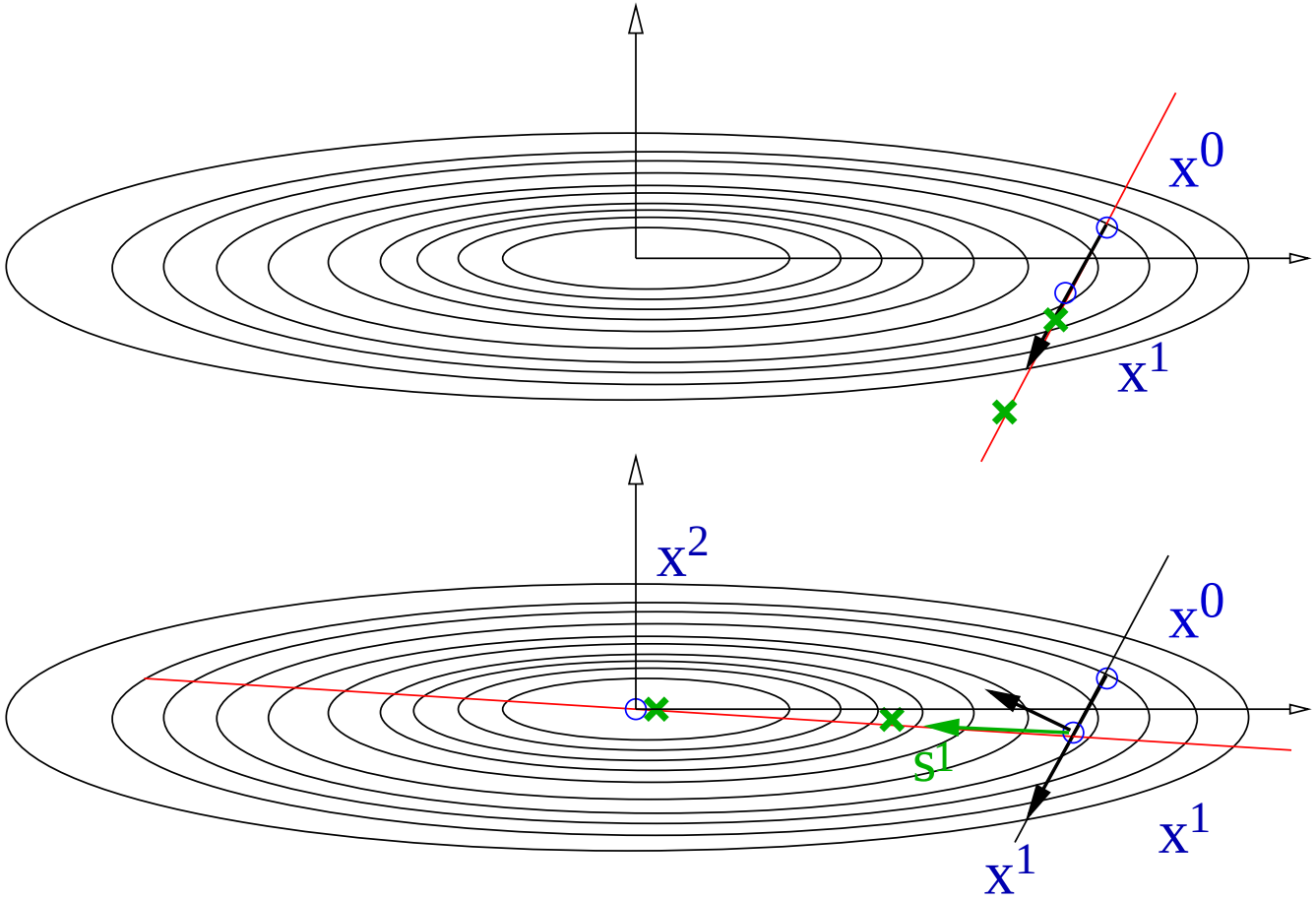
3. line minimisation along this search direction $\vec{s}^N$
4. continue with step 1), if the gradient is not sufficiently small.

the search directions satisfy:

$$\vec{s}^N \mathbf{B} \vec{s}^M = \delta_{NM} \qquad \forall N, M$$

the conjugate gradient algorithm finds the minimum of a quadratic function with k degrees of freedom in $k+1$ steps exactly

1. steepest descent step from $\vec{x}^0$, search for minimum along $\vec{g}_0$ by performing several trial steps (crosses, at least one triastep is required) $\rightarrow \vec{x}^1$
2. determine new gradient $\vec{g}_1 = \vec{g}(\vec{x}_1)$ and conjugate it to get $\vec{s}_1$ (green arrow) for 2d-functions the gradient points now directly to the minimum
3. minimisation along search direction $\vec{s}_1$



Asymptotic convergence rate

- asymptotic convergence rate is the convergence behaviour for the case that the degrees of freedom are much large than the number of steps

  e.g. 100 degrees of freedom but you perform only 10-20 steps

- how quickly, do the forces decrease?

- this depends entirely on the eigenvalue spectrum of the Hessian matrix:
  - steepest descent: $\Gamma_{max}/\Gamma_{min}$ steps are required to reduce the forces to a fraction ε
  - DIIS, CG, damped MD: $\sqrt{\Gamma_{max}/\Gamma_{min}}$ steps are required to reduce the forces to a fraction ε

$\Gamma_{max}, \Gamma_{min}$ are the maximum and minimal eigenvalue of the Hessian matrix

Damped molecular dynamics

instead of using a fancy minimisation algorithms it is possible to treat the minimisation problem using a simple "simulated annealing algorithm"

- regard the positions as dynamic degrees of freedom
- the forces serve as accelerations and an additional friction term is introduced
- equation of motion ($\vec{x}$ are the positions)

$$\ddot{\vec{x}} = -2 * \alpha \vec{g}(\vec{x}) - \mu \dot{\vec{x}},$$

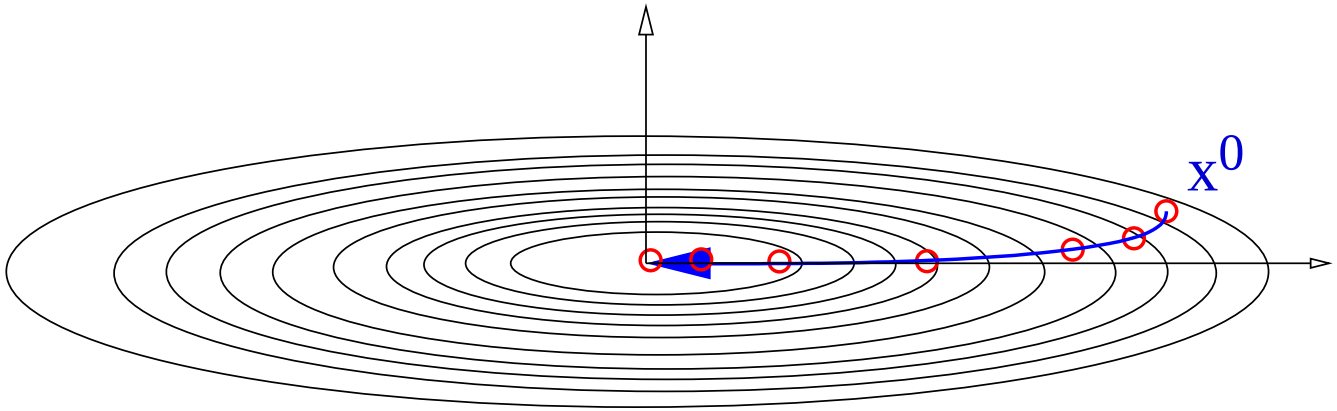
using a velocity Verlet algorithm this becomes

$$\vec{v}_{N+1/2} = \left((1 - \mu/2) \vec{v}_{N-1/2} - 2 * \alpha \vec{F}_N \right) / (1 + \mu/2)$$

$$\vec{x}_{N+1} = \vec{x}_{N+1} + \vec{v}_{N+1/2}$$

for $\mu = 2$, this is equivalent to a simple steepest descent step

- behaves like a rolling ball with a friction
  it will accelerate initially, and then deaccelerate when close to the minimum
- if the optimal friction is chosen the ball will glide right away into the minimum
- for a too small friction it will overshoot the minimum and accelerate back
- for a tool large friction relaxation will also slow down (behaves like a steepest descent)



Algorithms implemented in VASP

		additional flags	termination
DISS	`IBRION =1`	`POTIM`, `NFREE`	`EDIFFG`
CG	`IBRION =2`	`POTIM`	`EDIFFG`
damped MD	`IBRION =3`	`POTIM`, `SMASS`	`EDIFFG`

`POTIM` determines generally the step size

for the CG gradient algorithm, where line minisations are performed, this is the size of the very first trial step

`EDIFFG` determines when to terminate relaxation

positive values: energy change between steps must be less than `EDIFFG`

negative values: $|\vec{F}_i| < |\texttt{EDIFFG}| \quad \forall i = 1, N_{ions}$

DIIS

- POTIM determines the step size in the steepest descent steps
  no line minisations are performed !!
- NFREE determines how many ionic steps are stored in the iteration history

  set of points $\{\vec{x}^i | i = 1,...,N\}$ and $\{\vec{g}^i | i = 1,...,N\}$ searches for a linear combination of x^i, that minimises the gradient

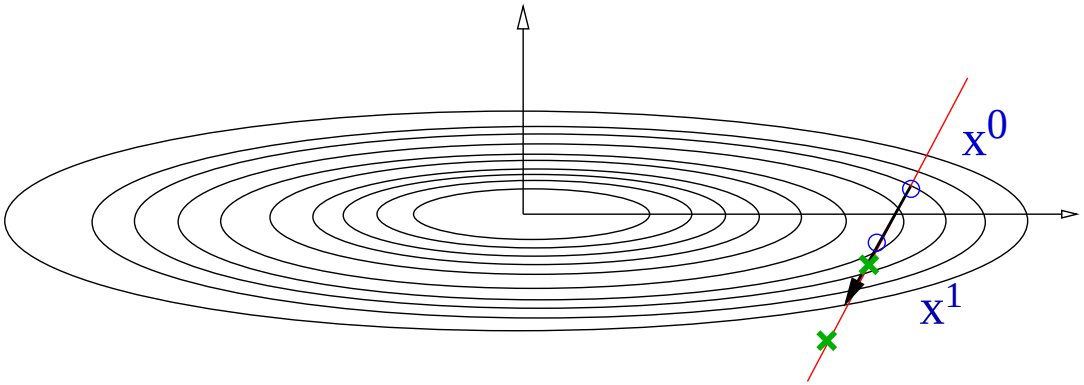
  NFREE is the maximum N
- for complex problems NFREE can be large (i.e. 10-20)
- for small problems, it is advisable to count the degrees of freedom carefully (symmetry inequivalent degrees of freedom)
- if NFREE is not specified, VASP will try to determine a reasonable value, but usually the convergence is then slower

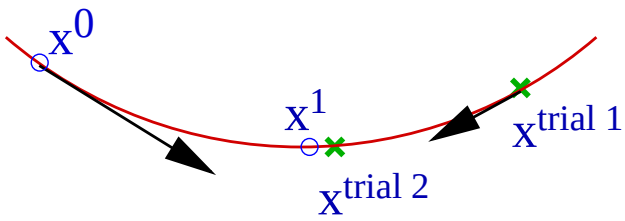
CG

- the only required parameter is `POTIM`

  this parameter is used to parameterise, how large the trial steps are

- CG requires a line minisations along the search direction





  this is done using a variant of Brent's algorithm

  – trial step along search direction (conjg. gradient scaled by `POTIM`)

  – quadratic or cubic interpolation using energies and forces at $\vec{x}_0$ and $\vec{x}_1$ allows to determine the approximate minimum

  – continue minimisation as long as approximate minimum is not accurate enough

Damped MD

- two parameters `POTIM` and `SMASS`

$$\vec{v}_{N+1/2} = \left((1-\mu/2)\vec{v}_{N-1/2} - 2*\alpha\vec{F}_N\right)/(1+\mu/2) \qquad \vec{x}_{N+1} = \vec{x}_{N+1} + \vec{v}_{N+1/2}$$

  $\alpha \propto$ `POTIM` and $\mu \propto$ `SMASS`

- `POTIM` must be as large as possible, but without leading to divergence and `SMASS` must be set to $\mu \approx 2\sqrt{\Gamma_{\min}/\Gamma_{\max}}$, where $\Gamma_{\min}$ and $\Gamma_{\max}$ are the minimal und maximal eigenvalues of the Hessian matrix

- a practicle optimisation procedure:
  - set `SMASS=0.5-1` and use a small `POTIM` of 0.05-0.1
  - increase `POTIM` by 20 % until the relaxation runs diverge
  - fix `POTIM` to the largest value for which convergence was achieved
  - try a set of different `SMASS` until convergence is fastest (or stick to SMASS=0.5-1.0)

Damped MD — QUICKMIN

- alternatively do not specify SMASS (or set SMASS<0)

  this select an algorithm sometimes called QUICKMIN

- QUICKMIN

$$\vec{v}^{new} = \begin{cases} \alpha\vec{F} + \vec{F}(\vec{v}^{old} \cdot \vec{F})/||\vec{F}|| & \text{for } \vec{v}^{old} \cdot \vec{F} > 0 \\ \alpha\vec{F} & \text{else} \end{cases}$$

  - if the forces are antiparallel to the velocities, quench the velocities to zero and restart
  - otherwise increase the “speed” and make the velocities parallel to the present forces

- I have not often used this algorithm, but it is supposed to be very efficient

Damped MD — QUICKMIN

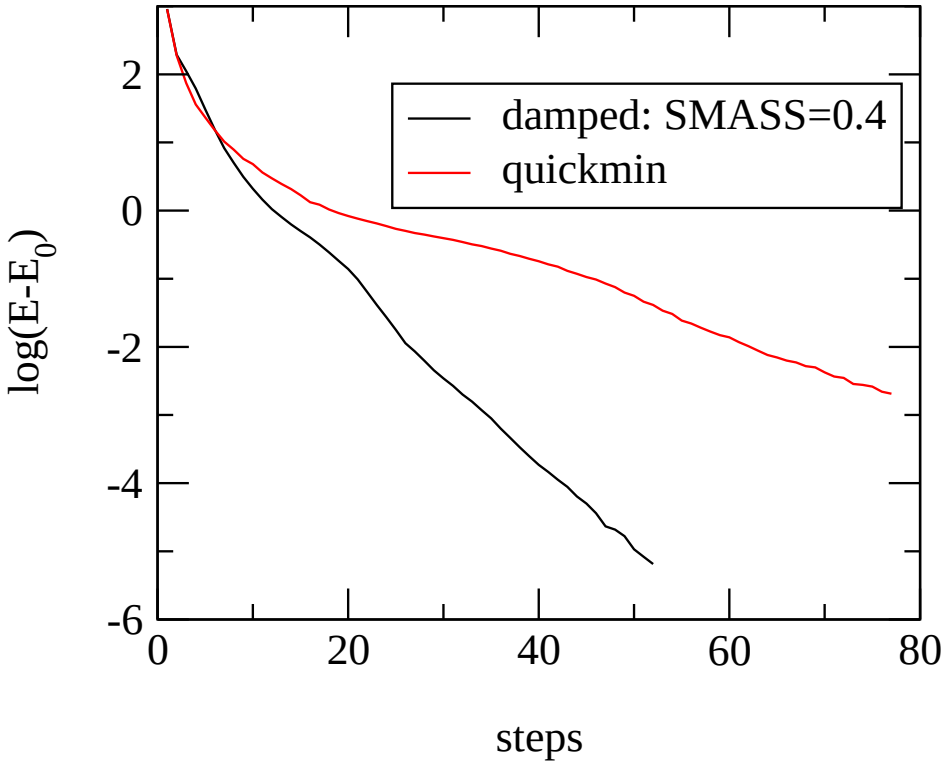
my experience is that damped MD (as implemented in VASP) is faster than QUICKMIN

but it requires less playing around

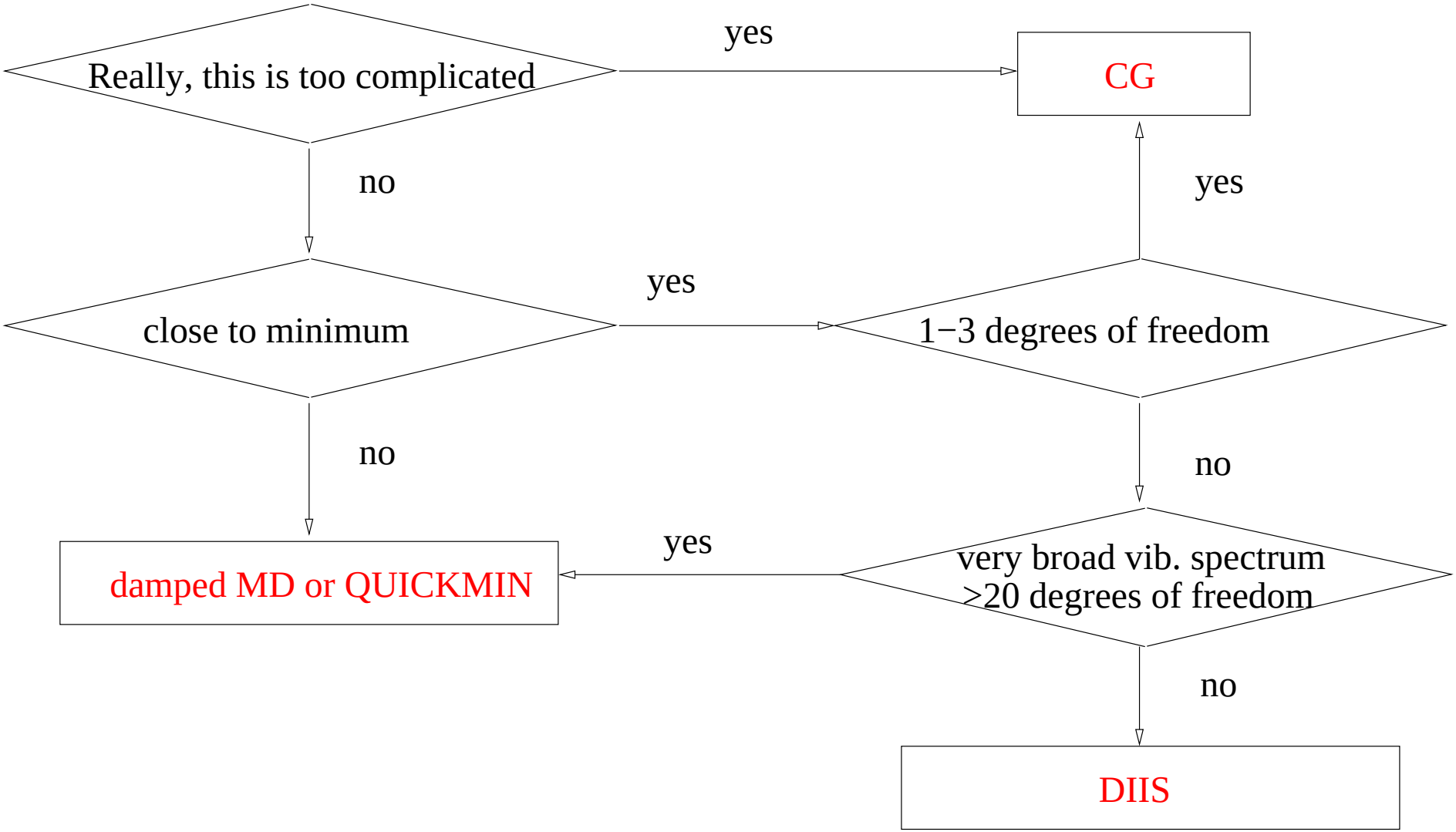
defective ZnO surface:

96 atoms are allowed to move!

relaxation after a finite temperature MD at 1000 K

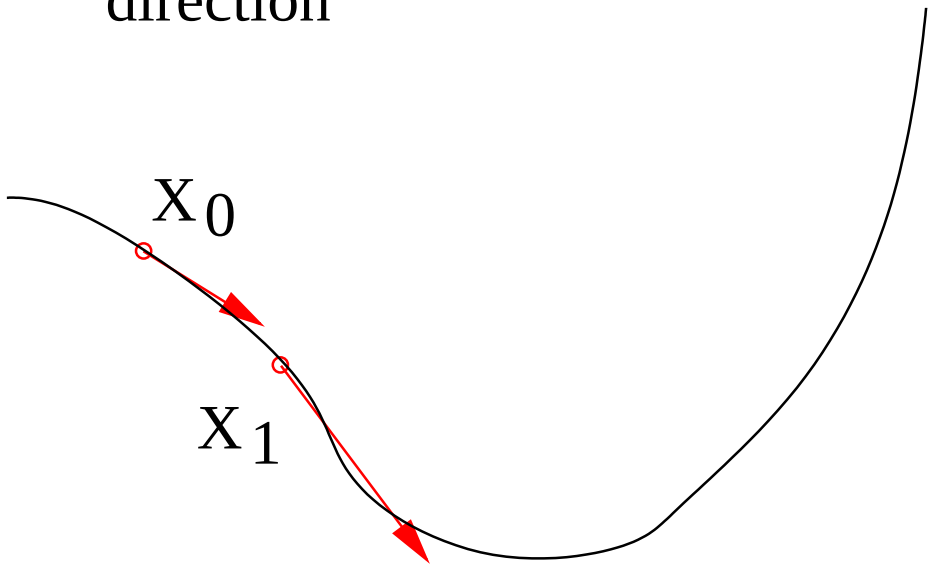


Why so many algorithms :-(... decision chart



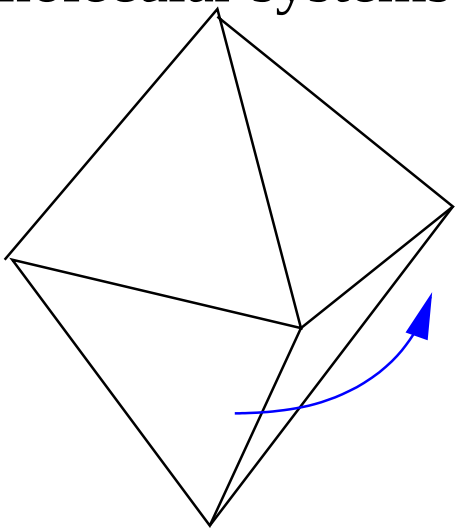
Two cases where the DIIS has huge troubles

force increases along the search direction



DIIS is dead, since it consideres the forces only
it will move uphill instead of down

rigid unit modes i.e. in perovskites (rotation)
molecular systems (rotation)



in cartesian coordinates the Hessian matrix changes when the octahedron rotates!

How bad can it get

- the convergence speed depends on the eigenvalue spectrum of the Hessian matrix
  - larger systems (thicker slabs) are more problematic (acoustic modes are very soft)
  - molecular system are terrible (week intermolecular and strong intramolecular forces)
  - rigid unit modes and rotational modes can be exceedingly soft

  the spectrum can vary over three orders of magnitudes $\Rightarrow$ 100 or even more steps might be required ionic relaxation can be painful
- to model the behaviour of the soft modes, you need very accurate forces since otherwise the soft modes are hidden by the noise in the forces

  `EDIFF` must be set to very small values (10^{-6}) if soft modes exist