Computergestützte Strukturbiologie
(Strukturelle Bioinformatik)

MD Algorithmen

Sommersemester 2009

Peter Güntert

MD Algorithmen

- Energieminimierung
- Integration der Bewegungsgleichungen
- Temperaturkontrolle
- Druckkontrolle
- Periodische Randbedingungen
- Zwangsbedingungen: SHAKE

Energieflächen

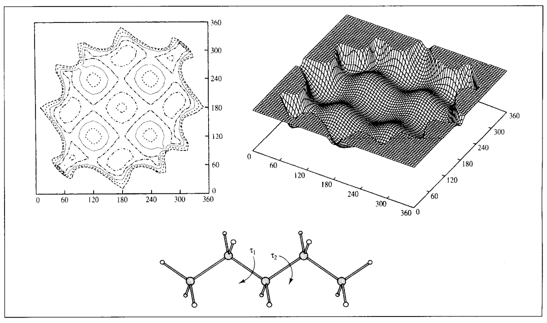


Fig. 5.1: Variation in the energy of pentane with the two torsion angles indicated and represented as a contour diagram and isometric plot. Only the lowest-energy regions are shown.

Lokale Minima, globales Minimum

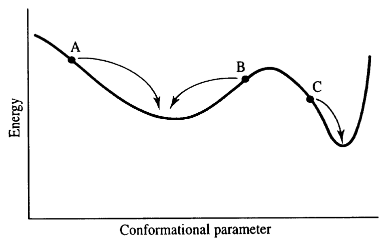


Fig. 5.3: A schematic one-dimensional energy surface. Minimisation methods move downhill to the nearest minimum. The statistical weight of the narrow, deep minimum may be less than a broad minimum which is higher in energy.

Energieminimierungsalgorithmen

- Ohne Ableitungen
- Mit Gradienten:
  - Steilster Abstieg (steepest descent)
  - konjugierte Gradienten (conjugate gradients)
- Mit zweiter Ableitung: Newton-Raphson Methode

Simplex Algorithmus

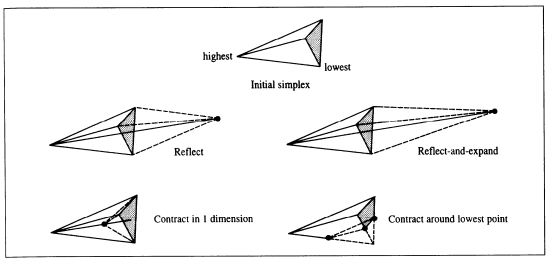


Fig. 5.4: The three basic moves permitted to the simplex algorithm (reflection, and its close relation reflect-and-expand; contract in one dimension and contract around the lowest point). (Figure adapted from Press W H, B P Flannery, S A Teukolsky and W T Vetterling 1992. Numerical Recipes in Fortran. *Cambridge, Cambridge University Press.)*

Simplex Algorithmus

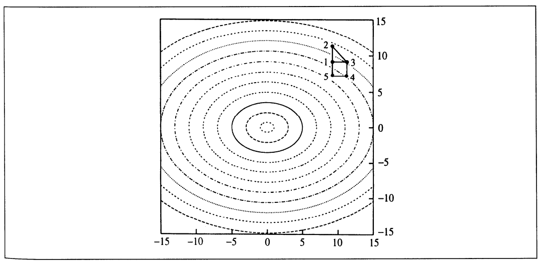


Fig. 5.5: The first few steps of the simplex algorithm with the function $x^2 + 2y^2$. The initial simplex corresponds to the triangle 123. Point 2 has the largest value of the function and the next simplex is the triangle 134. The simplex for the third step is 145.

Eindimensionale Minimierung

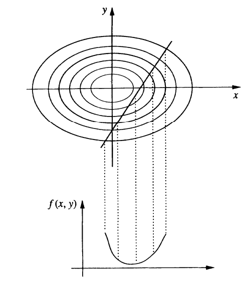


Fig. 5.7: A line search is used to locate the minimum in the function in the direction of the gradient.

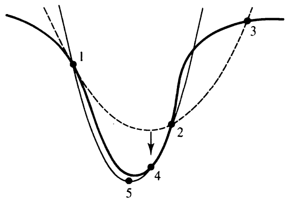


Fig. 5.8: The minimum in a line search may be found more effectively by fitting an analytical function such as a quadratic to the initial set of three points (1,2 and 3). A better estimate of the minimum can then be found by fitting a new function to the points 1, 2 and 4 and finding its minimum.

Energieminimierung: Steepest descent

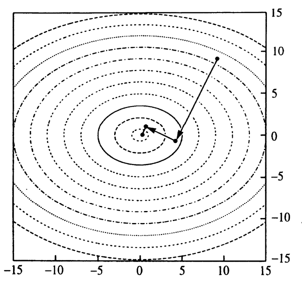


Fig. 5.9: Application of steepest descents to the function $x^2 + 2y^2$.

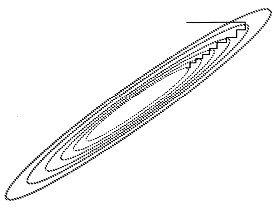


Fig. 5.10: The steepest descents method can give undesirable behaviour in a long narrow valley.

Konjugierte Gradientenmethode

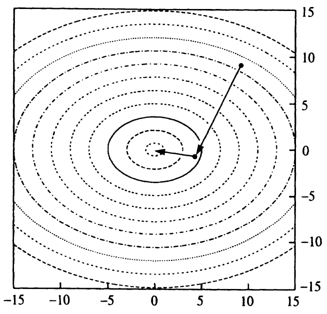


Fig. 5.11: Application of conjugate gradients method to the function $x^2 + 2y^2$.

Taylor Reihenentwicklung

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2}(x - x_0)^2$$

$$+ \frac{f'''(x_0)}{3!}(x - x_0)^3 + \ldots$$

$$\mathbf{r}(t + \delta t) = \mathbf{r}(t) + \delta t\mathbf{v}(t) + \tfrac{1}{2}\delta t^2\mathbf{a}(t) + \tfrac{1}{6}\delta t^3\mathbf{b}(t) + \tfrac{1}{24}\delta t^4\mathbf{c}(t) + \cdots$$

$$\mathbf{v}(t + \delta t) = \mathbf{v}(t) + \delta t\mathbf{a}(t) + \tfrac{1}{2}\delta t^2\mathbf{b}(t) + \tfrac{1}{6}\delta t^3\mathbf{c}(t) + \cdots$$

$$\mathbf{a}(t + \delta t) = \mathbf{a}(t) + \delta t\mathbf{b}(t) + \tfrac{1}{2}\delta t^2\mathbf{c}(t) \cdots$$

$$\mathbf{b}(t + \delta t) = \mathbf{b}(t) + \delta t\mathbf{c}(t) + \cdots$$

Verlet Algorithmus

The Verlet algorithm uses the positions and accelerations at time t, and the positions from the previous step, $\mathbf{r}(t - \delta t)$, to calculate the new positions at $t + \delta t$, $\mathbf{r}(t + \delta t)$. We can write down the following relationships between these quantities and the velocities at time t:

$$\mathbf{r}(t + \delta t) = \mathbf{r}(t) + \delta t\mathbf{v}(t) + \tfrac{1}{2}\delta t^2\mathbf{a}(t) + \cdots \tag{7.6}$$

$$\mathbf{r}(t - \delta t) = \mathbf{r}(t) - \delta t\mathbf{v}(t) + \tfrac{1}{2}\delta t^2\mathbf{a}(t) - \cdots \tag{7.7}$$

Adding these two equations gives

$$\mathbf{r}(t + \delta t) = 2\mathbf{r}(t) - \mathbf{r}(t - \delta t) + \delta t^2\mathbf{a}(t) \tag{7.8}$$

The velocities do not explicitly appear in the Verlet integration algorithm. The velocities can be calculated in a variety of ways; a simple approach is to divide the difference in positions at times $t + \delta t$ and $t - \delta t$ by $2\delta t$:

$$\mathbf{v}(t) = [\mathbf{r}(t + \delta t) - \mathbf{r}(t - \delta t)]/2\delta t \tag{7.9}$$

Alternatively, the velocities can be estimated at the half-step, $t + \tfrac{1}{2}\delta t$:

$$\mathbf{v}(t + \tfrac{1}{2}\delta t) = [\mathbf{r}(t + \delta t) - \mathbf{r}(t)]/\delta t \tag{7.10}$$

Leap-frog Algorithmus

The *leap-frog* algorithm uses the following relationships:

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \delta t\mathbf{v}(t+\tfrac{1}{2}\delta t) \tag{7.11}$$

$$\mathbf{v}(t+\tfrac{1}{2}\delta t) = \mathbf{v}(t-\tfrac{1}{2}\delta t) + \delta t\mathbf{a}(t) \tag{7.12}$$

To implement the leap-frog algorithm, the velocities $\mathbf{v}(t+\frac{1}{2}t)$ are first calculated from the velocities at time $t-\frac{1}{2}\delta t$ and the accelerations at time t. The positions $\mathbf{r}(t+\delta t)$ are then deduced from the velocities just calculated together with the positions at time $\mathbf{r}(t)$ using Equation (7.11). The velocities at time t can be calculated from

$$\mathbf{v}(t) = \tfrac{1}{2}[\mathbf{v}(t+\tfrac{1}{2}\delta t) + \mathbf{v}(t-\tfrac{1}{2}\delta t)] \tag{7.13}$$

The velocities thus 'leap-frog' over the positions to give their values at $t+\frac{1}{2}\delta t$ (hence the name). The positions then leap over the velocities to give their new values at $t+\delta t$, ready for the velocities at $t+\frac{3}{2}\delta t$, and so on. The leap-frog method has two advantages over the

Velocity Verlet Algorithmus

The *velocity Verlet* method [Swope *et al.* 1982] gives positions, velocities and accelerations at the same time and does not compromise precision:

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \delta t\mathbf{v}(t) + \tfrac{1}{2}\delta t^2\mathbf{a}(t) \tag{7.14}$$

$$\mathbf{v}(t+\delta t) = \mathbf{v}(t) + \tfrac{1}{2}\delta t[\mathbf{a}(t) + \mathbf{a}(t+\delta t)] \tag{7.15}$$

The velocity Verlet method is actually implemented as a three-stage procedure because, as can be seen from Equation (7.15), to calculate the new velocities requires the accelerations at both t and $t+\delta t$. Thus in the first step the positions at $t+\delta t$ are calculated according to Equation (7.14) using the velocities and the accelerations at time t. The velocities at time $t+\frac{1}{2}\delta t$ are then determined using:

$$\mathbf{v}(t+\tfrac{1}{2}\delta t) = \mathbf{v}(t) + \tfrac{1}{2}\delta t\mathbf{a}(t) \tag{7.16}$$

New forces are next computed from the current positions, thus giving $\mathbf{a}(t+\delta t)$. In the final step, the velocities at time $t+\delta t$ are determined using:

$$\mathbf{v}(t+\delta t) = \mathbf{v}(t+\tfrac{1}{2}\delta t) + \tfrac{1}{2}\delta t\mathbf{a}(t+\delta t) \tag{7.17}$$

Beemans Algorithmus

Beeman's algorithm [Beeman 1976] is also related to the Verlet method:

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \delta t\mathbf{v}(t) + \tfrac{2}{3}\delta t^2\mathbf{a}(t) - \tfrac{1}{6}\delta t^2\mathbf{a}(t-\delta t) \tag{7.18}$$

$$\mathbf{v}(t+\delta t) = \mathbf{v}(t) + \tfrac{1}{3}\delta t\mathbf{a}(t) + \tfrac{5}{6}\delta t\mathbf{a}(t) - \tfrac{1}{6}\delta t\mathbf{a}(t-\delta t) \tag{7.19}$$

The Beeman integration scheme uses a more accurate expression for the velocity. As a consequence it often gives better energy conservation, because the kinetic energy is calculated directly from the velocities. However, the expressions used are more complex than those of the Verlet algorithm and so it is computationally more expensive.

Gear Predictor-Corrector Algorithmen

The predictor–corrector methods [Gear 1971] form a general family of integration algorithms from which one can select a scheme that is correct to a given order. These methods have three basic steps. First, new positions, velocities, accelerations and higher-order terms are predicted according to the Taylor expansion, Equations (7.2)–(7.4). In the second stage, the forces are evaluated at the new positions to give accelerations $\mathbf{a}(t+\delta t)$. These accelerations are then compared with the accelerations that are predicted from the Taylor series expansion, $\mathbf{a}^{c}(t+\delta t)$. The difference between the predicted and calculated accelerations is then used to 'correct' the positions, velocities, etc., in the correction step:

$$\Delta\mathbf{a}(t+\delta t) = \mathbf{a}^{c}(t+\delta t) - \mathbf{a}(t+\delta t) \tag{7.22}$$

Then

$$\mathbf{r}^{c}(t+\delta t) = \mathbf{r}(t+\delta t) + c_0\Delta\mathbf{a}(t+\delta t) \tag{7.23}$$

$$\mathbf{v}^{c}(t+\delta t) = \mathbf{v}(t+\delta t) + c_1\Delta\mathbf{a}(t+\delta t) \tag{7.24}$$

$$\mathbf{a}^{c}(t+\delta t)/2 = \mathbf{a}(t+\delta t)/2 + c_2\Delta\mathbf{a}(t+\delta t) \tag{7.25}$$

$$\mathbf{b}^{c}(t+\delta t)/6 = \mathbf{b}(t+\delta t)/6 + c_3\Delta\mathbf{a}(t+\delta t) \tag{7.26}$$

Gear has suggested 'best' values of the coefficients $c_0, c_1, \ldots$. The set of coefficients to use depends upon the order of the Taylor series expansion. In Equations (7.23)–(7.26) the expansion has been truncated after the third derivative of the positions (i.e. $\mathbf{b}(t)$). The appropriate set of coefficients to use in this case is $c_0 = \frac{1}{6}$, $c_1 = \frac{5}{6}$, $c_2 = 1$ and $c_3 = \frac{1}{3}$.

Energie-fluktuation

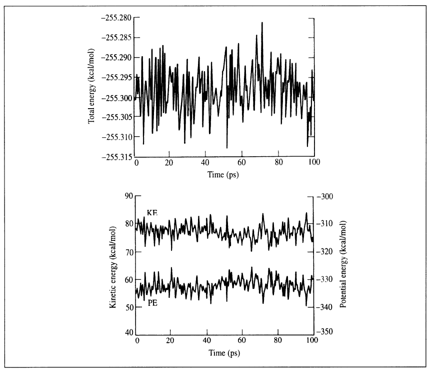


Fig. 7.3: Variation in total energy versus time for the production phase of a molecular dynamics simulation of 256 argon atoms at a temperature of 100 K and a density of 1.396 g cm^{-3} (top). The time step was 10 fs and the equations of motion were integrated using the velocity Verlet algorithm. The variations in the kinetic and potential energies are also shown (bottom). The graphs have different scales.

Zeitschrittlänge

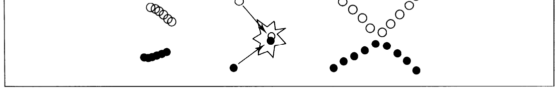


Fig. 7.4: With a very small time step (left) phase space is covered very slowly; a large time step (middle) gives instabilities. With an appropriate time step (right) phase space is covered efficiently and collisions occur smoothly.

System	Types of motion present	Suggested time step (s)
Atoms	Translation	10^{-14}
Rigid molecules	Translation and rotation	5×10^{-15}
Flexible molecules, rigid bonds	Translation, rotation, torsion	2×10^{-15}
Flexible molecules, flexible bonds	Translation, rotation, torsion, vibration	10^{-15} or 5×10^{-16}

Table 7.1 The different types of motion present in various systems together with suggested time steps.

Energieerhaltung

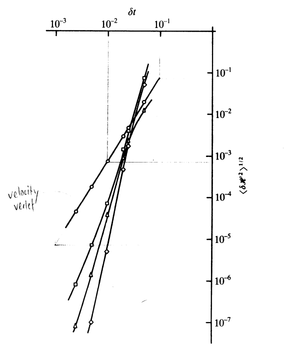


Fig. 3.3 Energy conservation of various algorithms. The system studied is as for Fig. 3.1. We calculate RMS energy fluctuations $\langle \delta \mathscr{H}^2 \rangle^{1/2}$ for various runs starting from the same initial conditions, and proceeding for the same total simulation time t_{run}, but using different time steps δt and corresponding numbers of steps $\tau_{run} = t_{run}/\delta t$. The plot uses log-log scales. The curves correspond to velocity Verlet (circles), Gear fourth-order (squares), Gear fifth-order (triangles), and Gear sixth-order (diamonds) algorithms.

Temperatur

Momentane Temperatur *T*(*t*):

$$\frac{1}{2} N k_B T(t) = E_{kin}(t) = \sum_{i=1}^{n} \frac{1}{2} m_i v_i^2$$

N = Anzahl Freiheitsgrade (*N* = 3*n*), *n* = Anzahl Atome

Methoden fur MD Simulation bei konstanter Temperatur:

- (strikt) konstante kinetische Energie und Temperatur
- erweitertes System mit zusätzlichem Freiheitsgrad
- <u>schwache Kopplung an ein Wärmebad</u>

Druck

Druck = Kraft pro Flächeneinheit auf die Wände des Systems
Aber: bei periodischen Randbedingungen keine Wand vorhanden

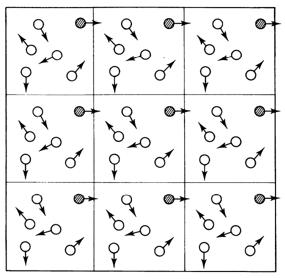
Virialsatz: $$P = \frac{2}{3V}\left[E_{kin} + \frac{1}{2}\sum_{i=1}^{n} \vec{r}_{ij} \cdot \vec{F}_{ij}\right]$$

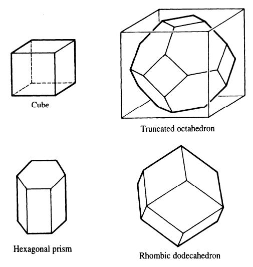
P = Druck, *V* = Volumen

Methoden fur MD Simulation bei konstantem Druck:

- (strikt) konstanter Druck
- erweitertes System mit zusätzlichem Freiheitsgrad
- <u>schwache Kopplung</u>

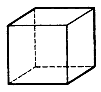
Periodische Randbedingungen





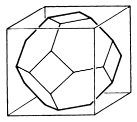
Periodische Randbedingungen

Rectangular box, side 2*a* (*x*) bv 2*b* (*y*) by 2*c* (*z*)



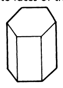
```
x = x − 2 × a × AINT(x/a)
y = y − 2 × b × AINT(y/b)
z = z − 2 × c × AINT(z/c)
A common alternative is:
x = x − a × ANINT(x/a)
y = y − b × ANINT(y/b)
z = z − c × ANINT(z/c)
```

Truncated octahedron derived from cube of side 2*a*



```
x = x − 2 × a × AINT(x/a)
y = y − 2 × b × AINT(y/a)
z = z − 2 × c × AINT(z/a)
if (ABS(x) + ABS(y) + ABS(z)) ≥ 1.5 × A
then
   x = x − SIGN(a, x)
   y = y − SIGN(a, y)
   z = z − SIGN(a, z)
endif
```

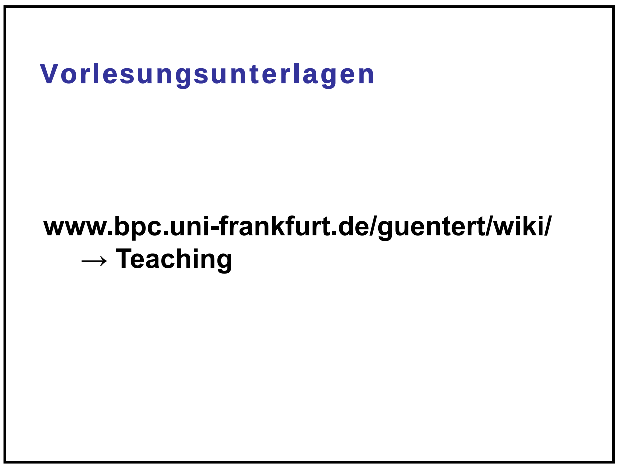
Hexagonal prism of length 2*a* (in *z* direction) and distance between opposite faces of the hexagon 2b



```
z = z − 2 × a × AINT(z/a)
   x = x − 2 × b × AINT(x/b)
if (ABS(x) + √3 × ABS(y)) ≥ 2 × B then
   x = x − SIGN(b, x)
   y = y − SIGN(√3 × b, y)
endif
```

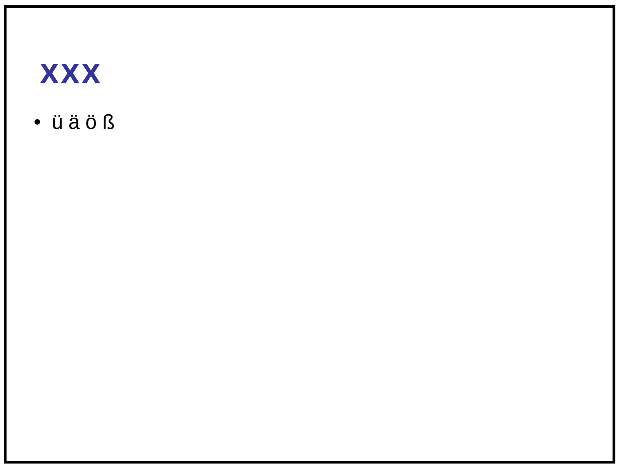
Literatur

- Andrew R. Leach: *Molecular Modellling, Principles and Applications,* Prentice Hall, 2001.
- M. P. Allen & D. J. Tildesley: *Computer Simulation of Liquids*, Clarendon Press, 1987.
- Tamar Schlick: *Molecular Modeling and Simulation*, Springer, 2006.



Vorlesungsunterlagen

www.bpc.uni-frankfurt.de/guentert/wiki/
→ Teaching



XXX

- ü ä ö ß