Idézet egy cikkből:

The Verlet algorithm

In molecular dynamics, the most commonly used time integration algorithm is probably the so-called Verlet algorithm [13]. The basic idea is to write two third-order Taylor expansions for the positions $\mathbf{r}(t)$, one forward and one backward in time. Calling $\mathbf{v}$ the velocities, $\mathbf{a}$ the accelerations, and $\mathbf{b}$ the third derivatives of $\mathbf{r}$ with respect to t, one has:

$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \mathbf{v}(t)\Delta t + (1/2)\mathbf{a}(t)\Delta t^2 + (1/6)\mathbf{b}(t)\Delta t^3 + O(\Delta t^4)$$

$$\mathbf{r}(t-\Delta t) = \mathbf{r}(t) - \mathbf{v}(t)\Delta t + (1/2)\mathbf{a}(t)\Delta t^2 - (1/6)\mathbf{b}(t)\Delta t^3 + O(\Delta t^4)$$

Adding the two expressions gives

$$\mathbf{r}(t+\Delta t) = 2\mathbf{r}(t) - \mathbf{r}(t-\Delta t) + \mathbf{a}(t)\Delta t^2 + O(\Delta t^4) \qquad (7)$$

This is the basic form of the Verlet algorithm. Since we are integrating Newton's equations, $\mathbf{a}(t)$ is just the force divided by the mass, and the force is in turn a function of the positions $\mathbf{r}(t)$:

$$\mathbf{a}(t) = -(1/m)\nabla V(\mathbf{r}(t)) \qquad (8)$$

Itt V jelöli az erőtér potenciálját, ∇V pedig a potenciál gradiensét, az erőt

As one can immediately see, the truncation error of the algorithm when evolving the system by Δt is of the order of Δt^4, even if third derivatives do not appear explicitly. This algorithm is at the same time simple to implement, accurate and stable, explaining its large popularity among molecular dynamics simulators.

A problem with this version of the Verlet algorithm is that velocities are not directly generated. While they are not needed for the time evolution, their knowledge is sometimes necessary. Moreover, they are required to compute the kinetic energy *K*, whose evaluation is necessary to test the conservation of the total energy *E=K+V*. This is one of the most important tests to verify that a MD simulation is proceeding correctly. One could compute the velocities from the positions by using

$$\mathbf{v}(t) = \frac{\mathbf{r}(t+\Delta t) - \mathbf{r}(t-\Delta t)}{2\Delta t}.$$

However, the error associated to this expression is of order Δt^2 rather than Δt^4.

To overcome this difficulty, some variants of the Verlet algorithm have been developed. They give rise to exactly the same trajectory, and differ in what variables are stored in memory and at what times. The *leap-frog* algorithm, not reported here, is one of such variants [20] where velocities are handled somewhat better.

An even better implementation of the same basic algorithm is the so-called *velocity Verlet* scheme, where positions, velocities and accelerations at time $t + \Delta t$ are obtained from the same quantities at time *t* in the following way:

$$\mathbf{r}(t + \Delta t) = \mathbf{r}(t) + \mathbf{v}(t)\Delta t + (1/2)\mathbf{a}(t)\Delta t^2$$

$$\mathbf{v}(t + \Delta t/2) = \mathbf{v}(t) + (1/2)\mathbf{a}(t)\Delta t$$

$$\mathbf{a}(t + \Delta t) = -(1/m)\nabla V\,(\mathbf{r}(t + \Delta t))$$

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t + \Delta t/2) + (1/2)\mathbf{a}(t + \Delta t)\Delta t$$

Note how we need 9*N* memory locations to save the 3*N* positions, velocities and accelerations, but we never need to have simultaneously stored the values at two different times for any one of these quantities.

Idézet egy másik cikkből:

Integration Algorithms

All the integration algorithms assume the positions, velocities and accelerations can be approximated by a Taylor series expansion:

$$r(t + \delta t) = r(t) + v(t)\delta t + \frac{1}{2}a(t)\delta t^2 + \ldots$$

$$v(t + \delta t) = v(t) + a(t)\delta t + \frac{1}{2}b(t)\delta t^2 + \ldots$$

$$a(t + \delta t) = a(t) + b(t)\delta t + \ldots$$

Where *r* is the position, *v* is the velocity (the first derivative with respect to time), *a* is the acceleration (the second derivative with respect to time), etc.

To derive the **Verlet** algorithm one can write

$$r(t+\delta t) = r(t) + v(t)\delta t + \frac{1}{2}a(t)\delta t^2$$

$$r(t-\delta t) = r(t) - v(t)\delta t + \frac{1}{2}a(t)\delta t^2$$

Summing these two equations, one obtains

$$r(t+\delta t) = 2r(t) - r(t-\delta t) + a(t)\delta t^2$$

The Verlet algorithm uses positions and accelerations at time *t* and the positions from time *t-δt* to calculate new positions at time *t+δt*. The Verlet algorithm uses no explicit velocities. The advantages of the Verlet algorithm are, *i)* it is straightforward, and *ii)* the storage requirements are modest . The disadvantage is that the algorithm is of moderate precision.

The Leap-frog algorithm

$$r(t+\delta t) = r(t) + v\left(t+\frac{1}{2}\delta t\right)\delta t$$

$$v\left(t+\frac{1}{2}\delta t\right) = v\left(t-\frac{1}{2}\delta t\right) + a(t)\delta t$$

In this algorithm, the velocities are first calculated at time *t+1/2δt*; these are used to calculate the positions, *r*, at time *t+δt*. In this way, the velocities *leap* over the positions, then the positions *leap* over the velocities. The advantage of this algorithm is that the velocities are explicitly calculated, however, the disadvantage is that they are not calculated at the same time as the positions. The velocities at time *t* can be approximated by the relationship:

$$v(t) = \frac{1}{2}\left[v\left(t-\frac{1}{2}\delta t\right) + v\left(t+\frac{1}{2}\delta t\right)\right]$$

The Velocity Verlet algorithm

This algorithm yields positions, velocities and accelerations at time *t*. There is no compromise on precision.

$$r(t+\delta t) = r(t) + v(t)\delta t + \frac{1}{2}a(t)\delta t^2$$

$$v(t+\delta t) = v(t) + \frac{1}{2}\left[a(t) + a(t+\delta t)\right]\delta t$$

Beeman's algorithm

This algorithm is closely related to the Verlet algorithm

$$r(t+\delta t) = r(t) + v(t)\delta t + \frac{2}{3}a(t)\delta t^2 - \frac{1}{6}a(t-\delta t)\delta t^2$$

$$v(t+\delta t) = v(t) + v(t)\delta t + \frac{1}{3}a(t)\delta t + \frac{5}{6}a(t)\delta t - \frac{1}{6}a(t-\delta t)\delta t$$

The advantage of this algorithm is that it provides a more accurate expression for the velocities and better energy conservation. The disadvantage is that the more complex expressions make the calculation more expensive.