

Simulação Computacional de Líquidos

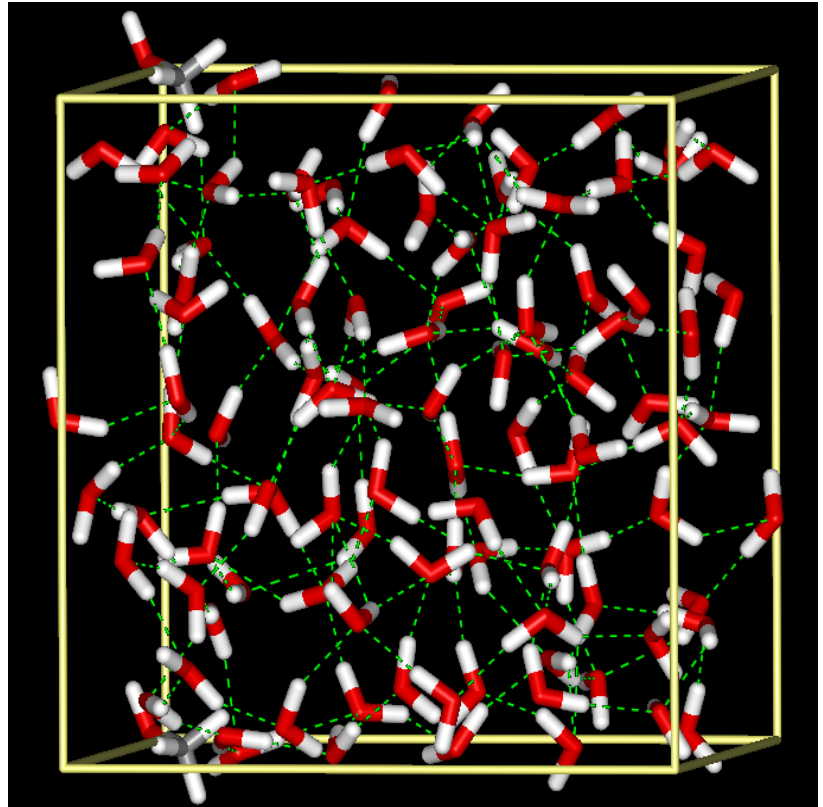
Profa. Kaline Coutinho

kaline@if.usp.br

Instituto de Física da USP

Aula 15: Introdução a Dinâmica Molecular para sistemas atômicos ou moléculas flexíveis:

- Formalismo,
- Métodos de integradores, e
- Condições iniciais.



Formalismo:

Equações de movimento na forma Lagrangeana, $L(q, \dot{q})$

$$\sum_k \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_k} \right) - \left(\frac{\partial L}{\partial q_k} \right) = 0 \quad \text{onde} \quad L(q, \dot{q}) = K(\dot{q}) - U(q)$$

k se refere às diferentes coordenadas e $q = r(t) \Rightarrow \dot{q} = v(t)$

$$L(q, \dot{q}) = \sum_i \frac{1}{2} m \dot{q}_i^2 - \sum_{i,j>i} U(q_{ij}) \quad \text{Apenas translação}$$

$$\frac{d}{dt} (m\dot{q}) + \left(\frac{\partial U}{\partial q} \right) = 0 \Rightarrow m\ddot{q} = - \frac{\partial U}{\partial q} \Rightarrow ma = F$$

3N equações
de 2a. ordem

$$\boxed{m\vec{a} = \vec{F}} \quad \text{onde} \quad \vec{F} = -\vec{\nabla} U$$

Formalismo:

Equações de movimento na forma Hamiltoniana, $H(p, q)$

Transformada de Legendre

$$H(p, q) = \dot{q}p - L(q, \dot{q}) \quad \text{onde} \quad p = \frac{\partial L}{\partial \dot{q}} = m\dot{q}$$

$$H(p, q) = \dot{q}m\dot{q} - \left(\frac{1}{2} m\dot{q}^2 - U(q) \right) = \frac{1}{2} m\dot{q}^2 + U(q)$$

$$H(p, q) = \frac{p^2}{2m} + U(q) = K(p) + U(q)$$

$$\dot{q} = \frac{\partial H}{\partial p} \Rightarrow \dot{r} = \frac{\partial H}{\partial p} \Rightarrow \dot{r} = \frac{p}{m}$$

$$\dot{p} = -\frac{\partial H}{\partial q} \Rightarrow \dot{p} = -\frac{\partial U}{\partial q} \Rightarrow \dot{p} = F$$

6N equações
de 1a. ordem

Considerações:

- As equações são reversíveis no tempo,
- Existe a conservação do momento linear e angular,

$$\vec{P} = \sum_i \vec{p}_i \quad \text{e} \quad \vec{l} = \sum_i \vec{r}_i \times \vec{p}_i$$

- Dependendo das condições de contorno pode, ou não, existir a conservação dos momentos.
- Se não existir $F_{\text{ext}}(t)$ ou $F(v)$ a energia se conserva. Considerando que K e U não dependem explicitamente de t .
- Como o gradiente de U aparece nas equações, então $U(r)$ não deve ter descontinuidades.

Idéia geral:

Dinâmica Molecular

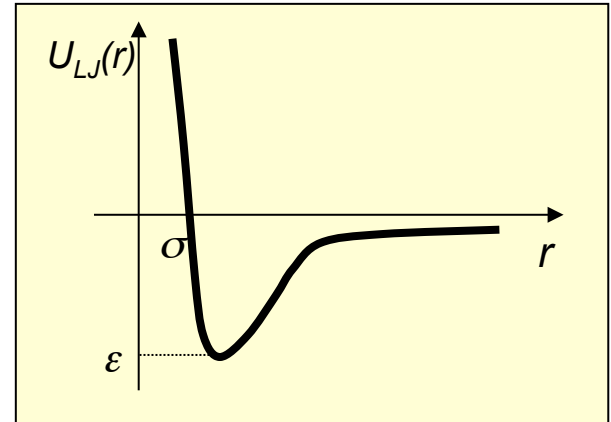
Se baseia na solução das equações de movimento para N partículas interagentes através do potencial $U(r)$.

$$\vec{\nabla} = \frac{\partial}{\partial r} \hat{r} + \frac{1}{r} \frac{\partial}{\partial \phi} \hat{\phi} + \frac{1}{r \sin \phi} \frac{\partial}{\partial \theta} \hat{\theta}$$

$$\vec{F} = -\vec{\nabla}U \quad \text{onde} \quad U(r) = 4\varepsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 \right] + \frac{q_i q_j}{r}$$

$$\vec{F} = \left\{ \frac{24\varepsilon}{r^2} \left[2 \left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 \right] + \frac{q_i q_j}{r^3} \right\} \vec{r}$$

$$\vec{a} = \frac{1}{m} \left\{ \frac{24\varepsilon}{r^2} \left[2 \left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 \right] + \frac{q_i q_j}{r^3} \right\} \vec{r}$$



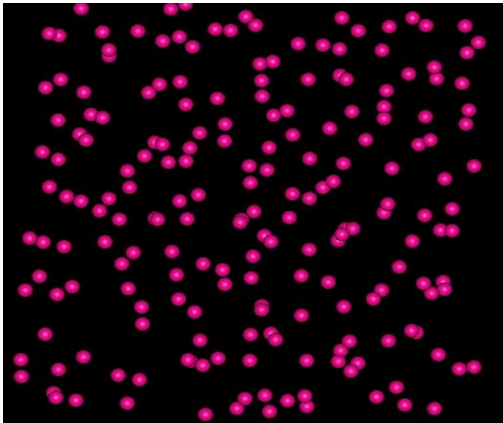
Dinâmica Molecular

Técnica que calcula as equações de Newton para cada molécula.

$$U(s_t) \Rightarrow F_t = -\nabla U_t \Rightarrow F_t = m a_t \Rightarrow s(t+\delta t) = f(s(t), v(t), a(t), \dots)$$

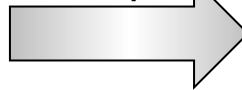
$$v(t+\delta t) = f(s(t), v(t), a(t), \dots)$$

Informações iniciais s_t e v_t

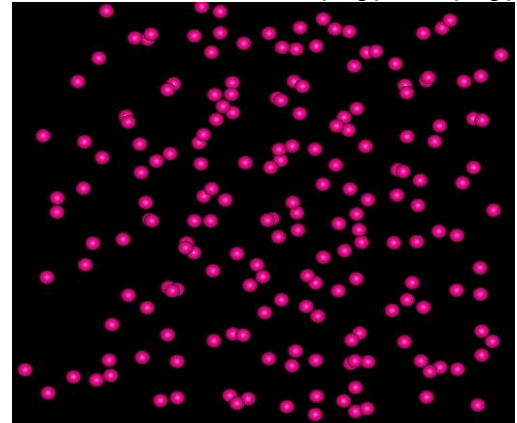


$s(t), v(t), a(t) \dots$

δt depois



calculamos $s_{t+\delta t}$ e $v_{t+\delta t}$



$s(t+\delta t), v(t+\delta t), a(t+\delta t) \dots$

Método de diferenças finitas

Um procedimento padrão para resolver equações diferenciais ordinárias, tipo as que temos, é usando métodos de diferenças finitas: dado $s(t)$, $v(t)$, $a(t)$, ..., achamos $s(t+\delta t)$, $v(t+\delta t)$, $a(t+\delta t)$, ..., onde δt é significativamente pequeno comparativamente ao tempo que uma molécula leva para viajar seu próprio comprimento ($\delta t \approx 10^{-16}$ a $10^{-14} \approx 0.1$ a $10 \times 10^{-15} \text{s} = f_s$).

Com esse tipo de método $U(r)$ deve ser suave. Caso não o seja, como é no exemplo de esferas rígidas, um tratamento explícito para tratar colisões deve ser incluído.

Muitos algoritmos foram sugeridos e aqui vamos discutir alguns:

Algoritmo Preditor-corretor [Gear, 1966]

Como as trajetórias são contínuas, podemos usar expansão de Taylor:

$$\left\{ \begin{array}{l} r_p(t + \partial t) = r(t) + \partial t v(t) + (1/2)\partial t^2 a(t) + (1/6)\partial t^3 b(t) + \dots \\ v_p(t + \partial t) = v(t) + \partial t a(t) + (1/2)\partial t^2 b(t) + \dots \\ a_p(t + \partial t) = a(t) + \partial t b(t) + \dots \\ b_p(t + \partial t) = b(t) + \dots \end{array} \right.$$

Essas equações não levam em conta as equações de movimento

onde $v(t) = \dot{r}(t)$, $a(t) = \ddot{r}(t)$ e $b(t) = \dddot{r}(t)$

$$\left\{ \begin{array}{l} r_c(t + \partial t) = r_p(t + \partial t) + C_0 \Delta a(t + \partial t) \\ v_c(t + \partial t) = v_p(t + \partial t) + C_1 \Delta a(t + \partial t) \\ a_c(t + \partial t) = a_p(t + \partial t) + C_2 \Delta a(t + \partial t) \\ b_c(t + \partial t) = b_p(t + \partial t) + C_3 \Delta a(t + \partial t) \end{array} \right.$$

$$\begin{array}{l} \Delta a(t + \partial t) = a_c(t + \partial t) - a_p(t + \partial t) \\ a_c(t + \partial t) = F(t + \partial t) / m \end{array}$$

Valores dos coeficientes corretores de Gear para eq. de 2a. ordem.

$$m\vec{a} = \vec{F} \quad \text{onde} \quad \vec{F} = -\vec{\nabla}U$$

Coeficientes	C0	C1	C2	C3	C4	C5
3	0	1	1			
4	1/6	5/6	1	1/3		
5	19/20	3/4	1	1/2	1/12	
6	3/20	251/360	1	11/18	1/6	1/60

Guarda9

N

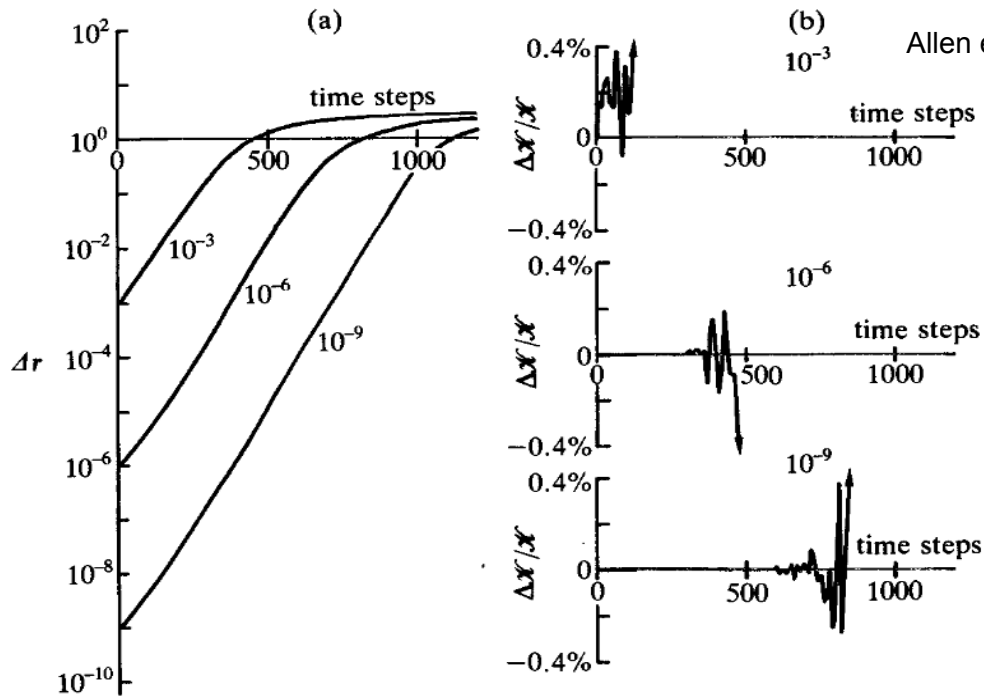
12N

15N

18N

Esses valores foram definidos para:

- i) maior precisão comparando com um sistema com solução exata como osciladores harmônicos;
- ii) duplicação de trajetórias próximas;
- iii) reversibilidade temporal.



É claro que **nenhum algoritmo de integração irá gerar trajetórias exatas por tempos longos**, mas isso não é necessário. O importante é **garantir a conservação de energia**. Assim, as trajetórias das partículas devem estar na hipersuperfície de E constante no espaço de fase, o que gera **médias corretas no ensemble microcanônico**.

Esse é o melhor algoritmo?

Crítérios para escolha de um algoritmo:

- Satisfazer os critérios de conservação de energia e **momentos** e a reversibilidade temporal;
- Permitir um δt grande;
- Duplicar trajetórias próximas;

Condições desejadas:

- Ser rápido;
- Usar pouca memória;
- Ser simples de implementar (programar).

Algoritmo de Verlet [Verlet, 1967]

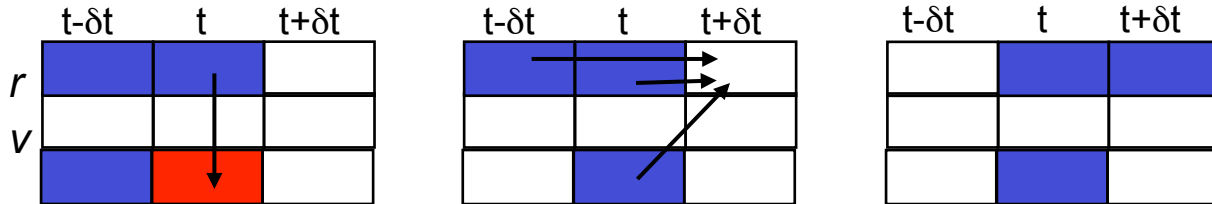
Usando expansão de Taylor também:

$$(1) \quad r(t + \partial t) = r(t) + \partial t v(t) + (1/2)\partial t^2 a(t) + (1/6)\partial t^3 b(t) + \dots$$

$$(2) \quad r(t - \partial t) = r(t) - \partial t v(t) + (1/2)\partial t^2 a(t) - (1/6)\partial t^3 b(t) + \dots$$

Somando (1) e (2):

$$r(t + \partial t) = 2r(t) - r(t - \partial t) + \partial t^2 a(t) \quad \text{Erro}(\partial t^4)$$



A velocidade não é necessária para as trajetórias, mas sim para calcular K e E. Então subtraindo (1) e (2):

$$v(t) = (r(t + \partial t) - r(t - \partial t)) / 2\partial t \quad \text{Erro}(\partial t^3)$$

Vantagens:

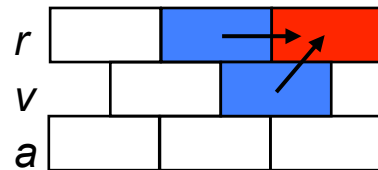
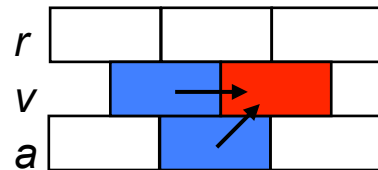
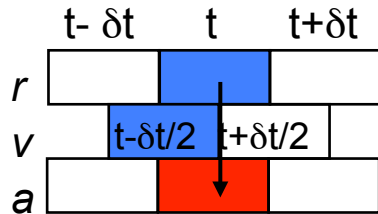
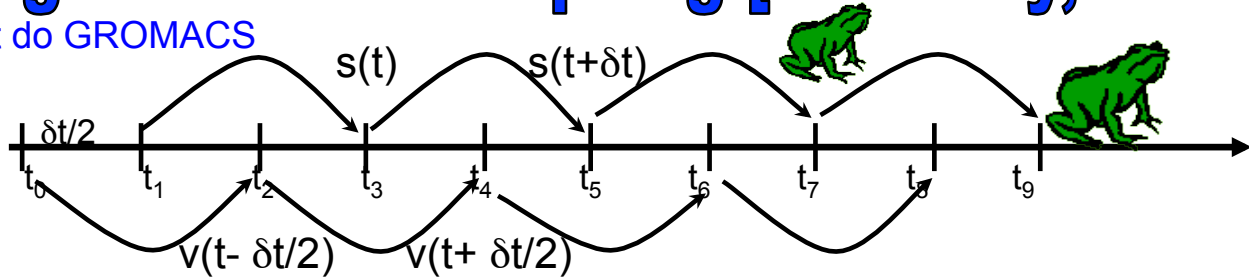
- É reversível no tempo;
- Calcula $r(t+\delta t)$ em um único passo;
- Guarda poucas variáveis 9N: $r(t-\delta t)$, $r(t)$, $a(t-\delta t)$;
- Tem excelente conservação de energia.

Desvantagens:

- Usa δt^2 que provoca imprecisão numérica.

Algoritmo de Leapfrog [Hockney, 1970]

Default do GROMACS



$$r(t + \partial t) = r(t) + \partial t v(t + \partial t / 2)$$

$$v(t + \partial t / 2) = v(t - \partial t / 2) + \partial t a(t)$$

A partir das definições:

$$v(t + \partial t / 2) = (r(t + \partial t) - r(t)) / \partial t$$

$$a(t) = (v(t + \partial t / 2) - v(t - \partial t / 2)) / \partial t$$

Para calcular K e E em t:

$$v(t) = (v(t + \partial t / 2) + v(t - \partial t / 2)) / 2$$

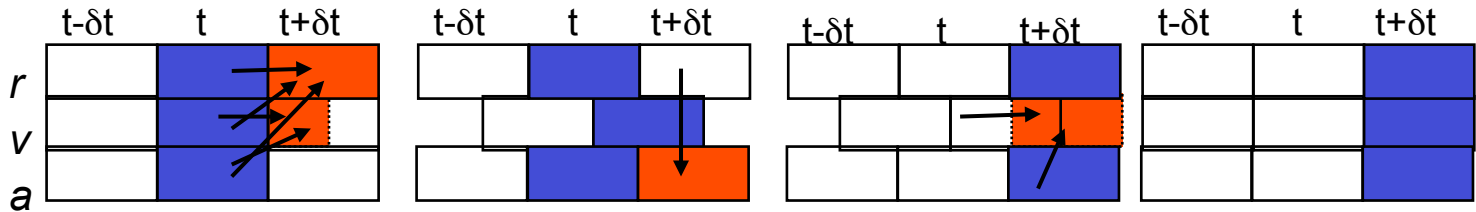
Só é exato para a(t) constante.

Algoritmo de Velocidade de Verlet [Swope, 1982]

$$r(t + \partial t) = r(t) + \partial t v(t) + (1/2)\partial t^2 a(t) \quad \text{Erro}(\partial t^3)$$

$$v(t + \partial t / 2) = v(t) + (1/2)\partial t a(t)$$

$$v(t + \partial t) = v(t + \partial t / 2) + (1/2)\partial t a(t + \partial t)$$

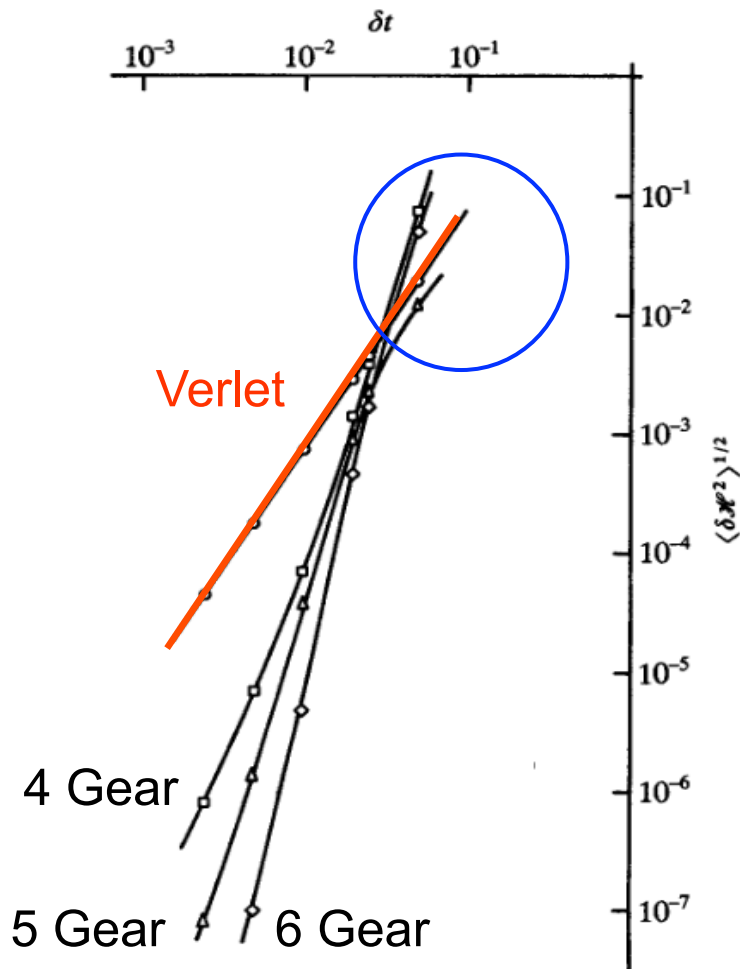


ou

$$r(t + \partial t) = r(t) + \partial t v(t) + (1/2)\partial t^2 a(t) \quad \text{Erro}(\partial t^3)$$

$$v(t + \partial t) = v(t) + (1/2)\partial t (a(t) + a(t + \partial t))$$

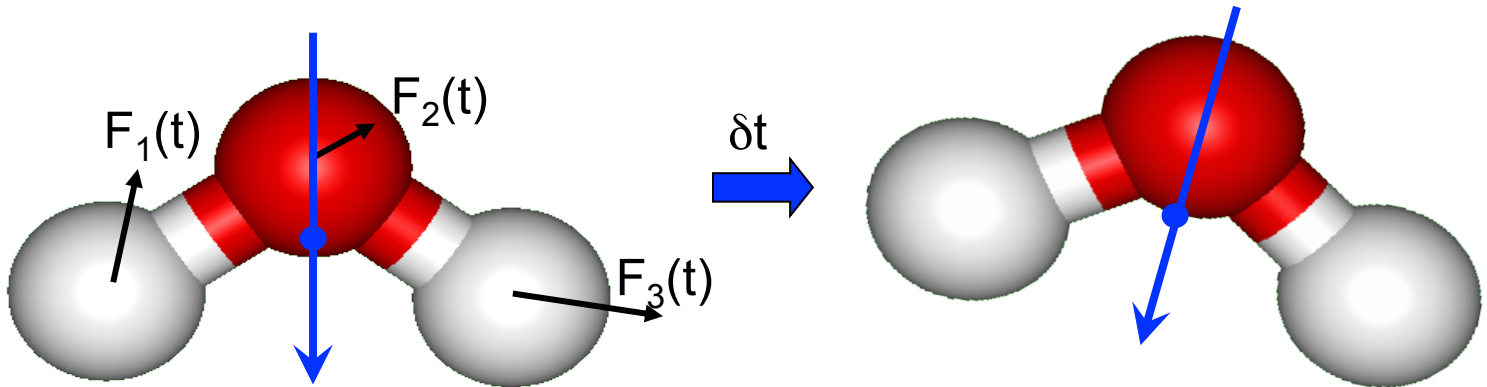
Velocidade de Verlet e Leap-frog reproduzem as mesmas trajetórias, mas em Leap-frog $v_0 = v(t - \delta t / 2)$ e em Verlet $v_0 = v(t)$.



Velocidade de Verlet melhora sua performance para δt maiores, $\delta t \approx 2\text{fs}$, e Leap-frog para δt menores, $\delta t \approx 0.1\text{fs}$.

Moléculas flexíveis

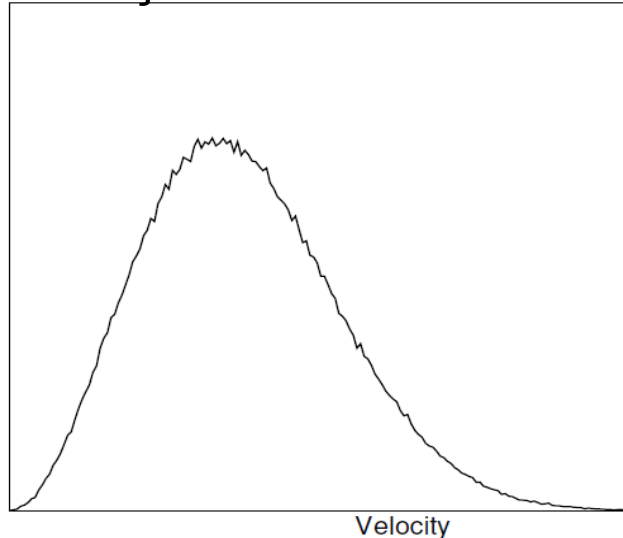
Aplicando as equações de movimento para cada átomo de uma molécula, num único δt a molécula translada, rotaciona e se deforma.



Para moléculas com átomos de H explícitos o recomendado é o algoritmo Leap-frog, $\delta t \approx 0.1\text{fs}$ e simulações mínimas de 10ns para sistemas soluto-solvente (10^8 passos), ou 50ns para proteínas e bicamadas hidratadas.

Condições iniciais:

- **Posições:** minimizadas ou próximas do equilíbrio, pois não podem inicialmente ter forças muito grandes.
- **Velocidades:** nulas ou geradas de forma aleatória satisfazendo a distribuição de Maxwell.



Distribuição de velocidades de Maxwell

Configuração Inicial (distribuição de v)

Para inicializar uma simulação MD, adicionalmente às coordenadas cartesianas dos átomos, também é necessário ter as velocidades tangenciais e angulares.

As velocidades tangenciais podem ser geradas:

- satisfazendo uma distribuição Gaussiana

$$\rho(x) = \sqrt{\frac{1}{2\pi\alpha^2}} e^{-x^2/2\alpha^2} \Rightarrow \rho(v_{ix}) = \sqrt{\frac{m_i}{2\pi kT}} e^{-(1/2)m_i v_i^2 / kT}$$

- satisfazendo uma distribuição uniforme $(-v_{\max}; +v_{\max})$ e deixando evoluir alguns Δt s (tipicamente 100 intervalos Δt)

No final, o momento linear total deve ser zerado.

$$\vec{P} = \sum_{i=1}^N m_i \vec{v}_i = 0$$

THE GLOBAL MD ALGORITHM

1. Input initial conditions

Potential interaction V as a function of atom positions

Positions \mathbf{r} of all atoms in the system

Velocities \mathbf{v} of all atoms in the system

↓

repeat 2,3,4 for the required number of steps:

2. Compute forces

The force on any atom

$$\mathbf{F}_i = -\frac{\partial V}{\partial \mathbf{r}_i}$$

is computed by calculating the force between non-bonded atom pairs:

$$\mathbf{F}_i = \sum_j \mathbf{F}_{ij}$$

plus the forces due to bonded interactions (which may depend on 1, 2, 3, or 4 atoms), plus restraining and/or external forces.

The potential and kinetic energies and the pressure tensor are computed.

↓

3. Update configuration

The movement of the atoms is simulated by numerically solving

Newton's equations of motion

$$\frac{d^2 \mathbf{r}_i}{dt^2} = \frac{\mathbf{F}_i}{m_i}$$

$$\frac{d\mathbf{r}_i}{dt} = \mathbf{v}_i; \quad \text{or} \quad \frac{d\mathbf{v}_i}{dt} = \frac{\mathbf{F}_i}{m_i}$$

↓

4. if required: Output step

write positions, velocities, energies, temperature, pressure, etc.

Fluxo do
algoritmo do
GROMACS