Computational Physics – Lecture 9: Molecular dynamics II

Kristel Michielsen
Institute for Advanced Simulation
Jülich Supercomputing Centre
Research Centre Jülich
k.michielsen@fz-juelich.de
http://www.fz-juelich.de/ias/jsc/qip

Contents

- Molecular dynamics simulations
  - Boundary conditions
  - Forces
    - Lennard-Jones potential
    - Truncation and shift of potentials
    - Linked list
  - Initial conditions
  - Integration algorithms
    - Euler
    - Euler-Cromer
    - Leap-frog
    - Verlet
    - Velocity Verlet
    - Hamiltonian splitting methods (simplectic)

Molecular dynamics simulation
Ingredients

- Boundary conditions
- Forces
- Initial conditions
- Integration algorithm
- Time step
- Equilibration
- Measurements

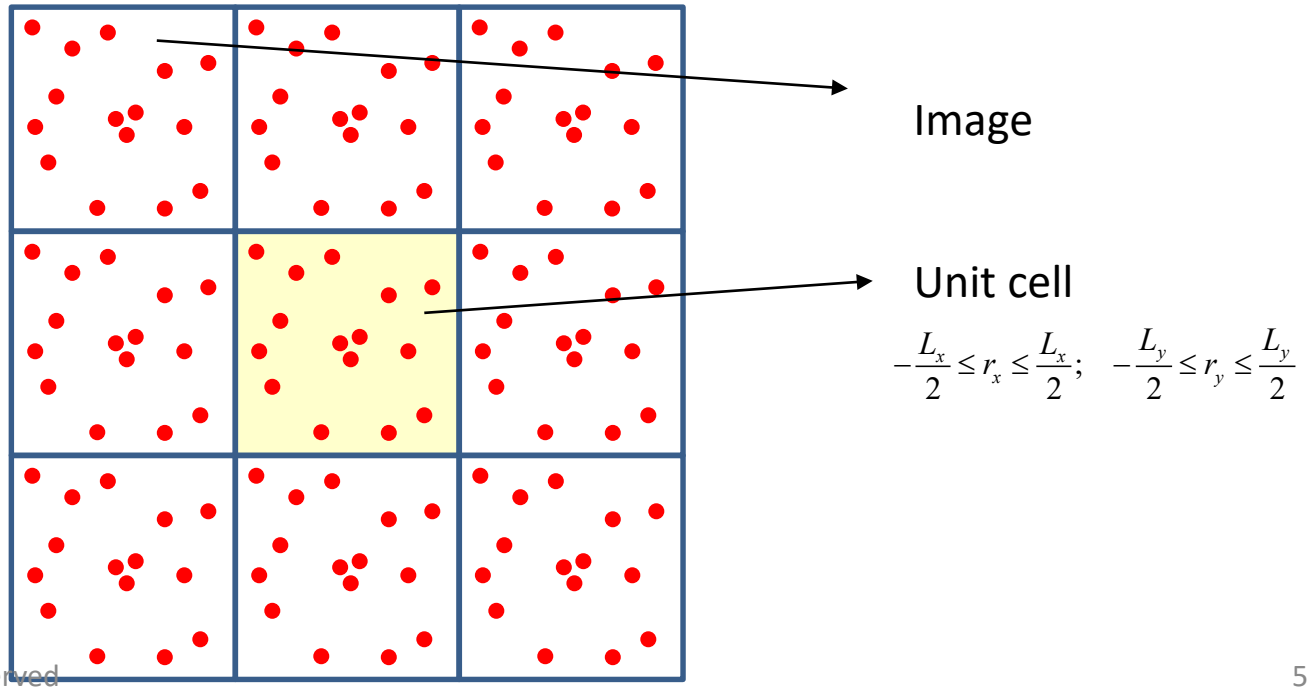
Molecular dynamics simulation
Boundary conditions

- Finite and infinite systems are very different
  - How large must a relatively small system be to yield results that resemble the behavior of the infinite system? No unique answer.
- Simulations are performed in a box of some sort
  - For a relatively small system the number of particles located near the box boundary is large compared to the number of interior particles
    → a simulation will fail to capture the typical state of an interior particle which is reflected in the measurements

Molecular dynamics simulation

Boundary conditions

- For simulations not addressing effects of "walls" it is better to remove them
- Periodic boundary conditions: Example



Molecular dynamics simulation
Boundary conditions

- Periodic boundary conditions:
  - Unit cell can have an arbitrary triclinic shape
    - Easiest and often used: rectangular box
    - Optimal: molecular-shaped box that minimizes the volume while the distances between any particle in a unit cell and any particle of the image remain larger than a specified value (can tremendously reduce the MD simulation time in e.g. the simulation of a macromolecule in a solvent)
  - Length scale: 10^4-10^8 atoms can be included → size of the computational cell is tens of nanometers

Molecular dynamics simulation

Boundary conditions

- Artifacts of periodicity are avoided if the interaction potentials are modified so that they vanish for distances larger than half the smallest length of the unit cell
- Periodic boundaries must be taken into account in the integration of the equations of motion and in the computation of the interactions, e.g. if all particles lie in a simulation box with x-coordinates between $-L_x/2$ and $L_x/2$
  - If $r_{xn} \geq L_x/2$ then $r_{xn} \rightarrow r_{xn} - L_x$
  - If $r_{xn} < -L_x/2$ then $r_{xn} \rightarrow r_{xn} + L_x$

Molecular dynamics simulation
Ingredients

- Boundary conditions
- Forces
- Initial conditions
- Integration algorithm
- Time step
- Equilibration
- Measurements

Molecular dynamics simulation

Forces

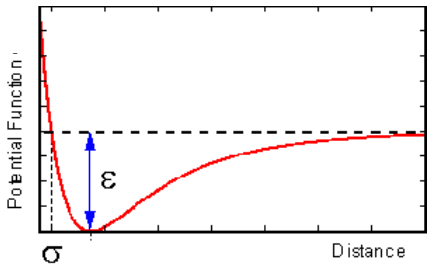
- General force fields for molecular systems are not of sufficient quality for universal use
- The interactions, at the simplest level, occur between pairs of particles and provide the two principal features of an interatomic force
  - resistance to compression: interaction repels at close range
  - binding particles together in the solid and liquid states: particles must attract each other over a range of separations

Molecular dynamics simulation

Forces

- Commonly used pair potential: Lennard-Jones (LJ)
  For a pair of particles i and j located at $\mathbf{r}_i$ and $\mathbf{r}_j$ the potential energy is

$$u(r_{ij}) = 4\varepsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right]$$



  where $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ and $r_{ij} \equiv |\mathbf{r}_{ij}|$, σ: measure of the range of the potential, ε: strength of the interaction.
  The potential
  - is positive for small r_{ij}: repulsion (short-range)
  - is negative for large r_{ij} : attraction (long-range)
  - goes to zero for large r_{ij}

Molecular dynamics simulation

Forces

- Forces acting on the particles

$$\mathbf{F}_i = \sum_{j \neq i}^{N} \mathbf{f}_{ij}, \mathbf{f}_{ij} = -\frac{du(r_{ij})}{dr_{ij}} \cdot \frac{\mathbf{r}_{ij}}{r_{ij}}$$

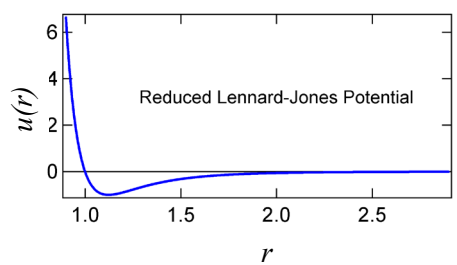
$$\mathbf{f}_{ij} = \frac{48\varepsilon}{r_{ij}^2}\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \frac{1}{2}\left(\frac{\sigma}{r_{ij}}\right)^{6}\right]\mathbf{r}_{ij}$$

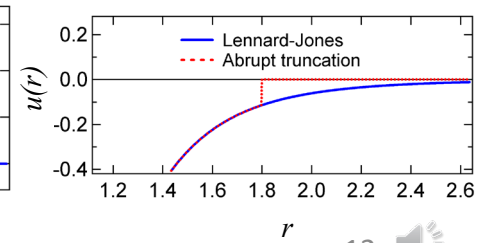
Molecular dynamics simulation

Forces: Truncation and shift of potentials

- Calculation of all particle interactions would require the calculation of $N(N-1)/2$ forces.
- Computing forces is the most demanding part of an MD simulation → introduce a cut-off radius r_c
- A simple cut-off of the potential leads to a discontinuity and therefore to a delta-function in the force → introduce a shift in the potential

$$u'(r_{ij}) = \begin{cases} u(r_{ij}) - u(r_c) & \text{if } r < r_c \\ 0 & \text{if } r \geq r_c \end{cases}$$



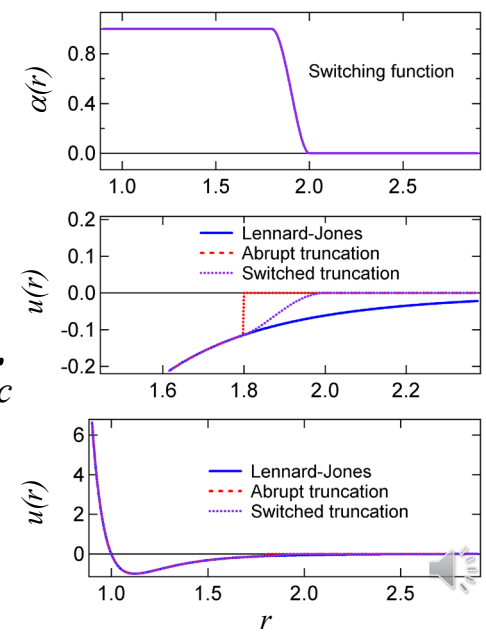


Molecular dynamics simulation

Forces: Truncation and shift of potentials

- A cut-off and shift of the potential leads to a discontinuity in the force
  - Causes errors when higher order integration algorithms are used that rely on the existence of force derivatives
  - Can be avoided by shifting the force function to zero at r_c . Several "switching functions" are used, e.g. $u''(r_{ij}) = \alpha(r_{ij})u(r_{ij})$ with

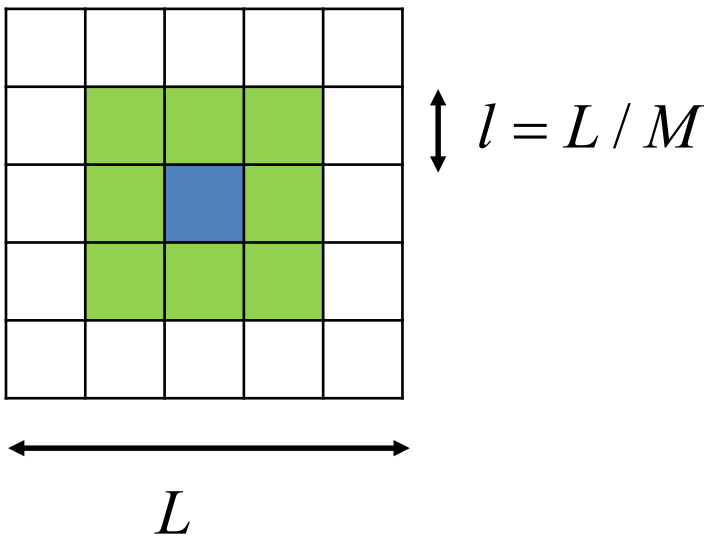
$$\alpha(r_{ij}) = \begin{cases} 1 & r_{ij} < r_c' \\ \dfrac{(r_c - r_{ij})^2 (r_c - 3r_c' + 2r_{ij})}{(r_c - r_c')^3} & r_c' \le r_{ij} \le r_c \\ 0 & r_{ij} > r_c \end{cases}$$



Molecular dynamics simulation

Forces: Linked list

- The volume L^3 of the MD simulation box is divided into M^3 sub cells with linear dimension $l = L / M > r_c$



  - Looking for neighbors for a particle in the blue sub cell is limited to searching in the green neighboring sub cells (26 in total for a cube)
  - Average number of particles in a sub cell $N_M = N / M^3$

Molecular dynamics simulation

Forces: Linked list

- Instead of calculating $N(N-1) \sim N^2$ particle interactions, $27 \times N_M \times N$ particle interactions have to be calculated: $O(N^2) \rightarrow O(N)$ for pair search algorithm
  - For simple symmetric potentials (e.g. Lennard-Jones) the calculation can be further reduced by a factor of two
- In practice: Define two arrays
  - `head [k=1, …, M³]` ("head" of the chain): its k-th element contains the largest particle number of all particles of cell k
  - `ll [i=1, …, N]` ("linked list"): number of the next particle (with lower particle number) in the presently treated cell; when `ll [i]=0` the cell is finished

Molecular dynamics simulation

Forces: Linked list

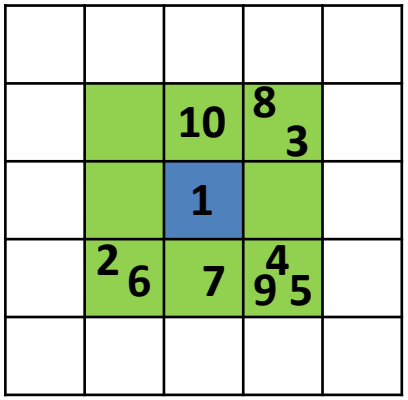
– Pseudocode:

```
ncell=M*M*M                 ! Number of sub cells
lcell=L/M                   ! Linear dimension sub cell
do k=1,ncell
    head(k)=0  ! Initialize array head with "heads" of each sub cell
enddo
do i=1,N                    ! Loop over all particles
    icell=1+int(x(i)/lcell)+int(y(i)/lcell)*M+
    int(z(i)/lcell)*M*M  ! Cell to which particle (x,y,z) belongs
    ll(i)=head(icell)  ! Build linked list
    head(icell)=i      ! Update head
enddo
```

May be performed at each MD step to update the linked-cell pointer arrays.

Molecular dynamics simulation

Forces: Linked list



- Example:
  - Outcome of the code:

```
i=1      icell=5      ll(1)=0      head(5)=1
i=2      icell=1      ll(2)=0      head(1)=2
i=3      icell=9      ll(3)=0      head(9)=3
i=4      icell=3      ll(4)=0      head(3)=4
i=5      icell=3      ll(5)=4      head(3)=5
i=6      icell=1      ll(6)=2      head(1)=6
i=7      icell=2      ll(7)=0      head(2)=7
i=8      icell=9      ll(8)=3      head(9)=8
i=9      icell=3      ll(9)=5      head(3)=9
i=10     icell=8      ll(10)=0     head(8)=10
```

  - Scanning the list in case of force calculations:
    Particles in cell 5 interact with particles from cells 1,2,3,5,8 and 9.
    The other cells are empty.

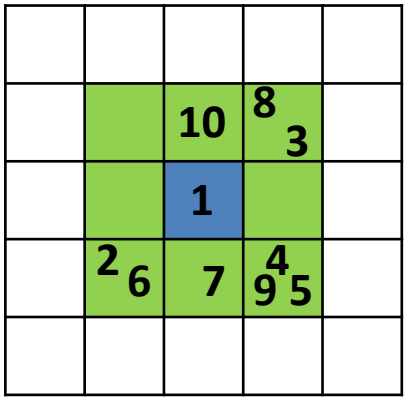
`icell=1  head(1)=6` → interaction (1-6) `ll(6)=2`
→ interaction (1-2) `ll(2)=0`

Molecular dynamics simulation

Forces: Linked list

- Outcome of the code:

```
i=1      icell=5      ll(1)=0      head(5)=1
i=2      icell=1      ll(2)=0      head(1)=2
i=3      icell=9      ll(3)=0      head(9)=3
i=4      icell=3      ll(4)=0      head(3)=4
i=5      icell=3      ll(5)=4      head(3)=5
i=6      icell=1      ll(6)=2      head(1)=6
i=7      icell=2      ll(7)=0      head(2)=7
i=8      icell=9      ll(8)=3      head(9)=8
i=9      icell=3      ll(9)=5      head(3)=9
i=10     icell=8      ll(10)=0     head(8)=10
```



- `icell=2  head(2)=7` → interaction (1-7) `ll(7)=0`
  `icell=3  head(3)=9` → interaction (1-9) `ll(9)=5`
  → interaction (1-5) `ll(5)=4`
  → interaction (1-4) `ll(4)=0`
  `icell=5  head(5)=1` `ll(1)=0`
  `icell=8  head(8)=10` → interaction (1-10) `ll(10)=0`
  `icell=9  head(9)=8` → interaction (1-8) `ll(8)=3`
  → interaction (1-3) `ll(3)=0`

Molecular dynamics simulation
Ingredients

- Boundary conditions
- Forces
- Initial conditions
- Integration algorithm
- Time step
- Equilibration
- Measurements

Molecular dynamics simulation

Initial conditions

- In principle, the initial conditions are not so important if the system naturally tends to equilibrium (ergodicity)
- Simple choice: Start with the particles on a lattice and draw initial momenta from a uniform or Gaussian distribution. The momenta are adjusted so that
  - the kinetic energy has the target value $3NT/2$ ($k_B = 1$)
  - the total momentum is zero to avoid the system moving as a whole

Molecular dynamics simulation
Ingredients

- Boundary conditions
- Forces
- Initial conditions
- Integration algorithm
- Time step
- Equilibration
- Measurements

Molecular dynamics

Integration algorithms: Euler

- Euler algorithm

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

  - Memory requirement: $9N$
  - Simple and fast
  - Not stable
  - Local [global] error $O((\Delta t^2))$ [$O((\Delta t))$]
  - Not symplectic

Molecular dynamics

Integration algorithms: Euler

– Not time-reversible: 👎

- Positions

$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \mathbf{v}(t)\Delta t$$

$$\mathbf{r}(t-\Delta t) = \mathbf{r}(t) - \mathbf{v}(t)\Delta t$$

$$\mathbf{r}(t+\Delta t) - \mathbf{r}(t-\Delta t) = 2\mathbf{v}(t)\Delta t \neq 0$$

- Velocities

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

$$\mathbf{v}(t-\Delta t) = \mathbf{v}(t) - \mathbf{a}(t)\Delta t$$

$$\mathbf{v}(t+\Delta t) - \mathbf{v}(t-\Delta t) = 2\mathbf{a}(t)\Delta t \neq 0$$

Molecular dynamics

Integration algorithms: Euler-Cromer

- Euler-Cromer algorithm

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

  – Memory requirement: $9N$
  – Simple and fast
  – Local [global] error $O((\Delta t^2))$ [$O((\Delta t))$]
  – Symplectic → stable
  – Not time-reversible

Molecular dynamics

Integration algorithms: Verlet

- Verlet algorithm

Taylor series expansions

$$\mathbf{r}(t+\Delta t)=\mathbf{r}(t)+\mathbf{v}(t)\Delta t+\frac{1}{2}\mathbf{a}(t)(\Delta t)^2+\frac{1}{6}\mathbf{a}'(t)(\Delta t)^3+O((\Delta t)^4)$$

$$\mathbf{r}(t-\Delta t)=\mathbf{r}(t)-\mathbf{v}(t)\Delta t+\frac{1}{2}\mathbf{a}(t)(\Delta t)^2-\frac{1}{6}\mathbf{a}'(t)(\Delta t)^3+O((\Delta t)^4)$$

Summation of both equations:

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

Velocities can be obtained from:

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

Molecular dynamics

Integration algorithms: Verlet

- Memory requirement: $9N$
- Simple and fast (forces calculated once per step)
- Positions: sum of two large and one small term (round-off error)
- Local [global] error in position $O((\Delta t^4))$ [$O((\Delta t^2))$]
- Local [global] error in velocity $O((\Delta t^2))$ [$O((\Delta t^2))$]
- Symplectic
- Time –reversible
- Velocities are not calculated directly. Sometimes more precise values for the velocities are required, e.g. if system properties that depend on velocity are required.
- Not self-starting: $\mathbf{r}(t)$ and $\mathbf{r}(t-\Delta t)$ are needed to get $\mathbf{r}(t+\Delta t)$ → use Euler method for the first step

Molecular dynamics

Integration algorithms: Leap-frog

- Leap-frog algorithm

  Defining the velocity at the midpoint between times t and $t-\Delta t$ gives

$$\mathbf{v}\left(t-\frac{\Delta t}{2}\right)=\frac{\mathbf{r}(t)-\mathbf{r}(t-\Delta t)}{\Delta t}$$

  Using an approximation for the derivative, the acceleration can be defined as

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

Molecular dynamics

Integration algorithms: Leap-frog

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

Defining the velocity at the midpoint between times t and $t+\Delta t$ gives

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

$$\mathbf{r}(t+\Delta t)=\mathbf{r}(t)+\mathbf{v}\left(t+\frac{\Delta t}{2}\right)\Delta t$$

Velocities (half integer time steps) and positions (integer time steps) are successively “leap-frogged” over each other

Molecular dynamics

Integration algorithms: Leap-frog

- Memory requirement: $9N$
- Not really fast, but time required for the integration is usually small compared to the time for the force calculation
- Loss of accuracy due to round-off error in Verlet algorithm is corrected
- Local [global] error $O((\Delta t^3))$ [$O((\Delta t^2))$]
- Symplectic
- Time –reversible
- Velocities are included explicitly in the method
- Velocities and positions are calculated at different times. If velocities at integer times are needed
  $\rightarrow \mathbf{v}(t) = [\mathbf{v}(t - \Delta t/2) + \mathbf{v}(t + \Delta t/2)]/2$
- Not self-starting

Molecular dynamics

Integration algorithms: Velocity Verlet

- Velocity Verlet algorithm

$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)(\Delta t)^2 + O((\Delta t)^3)$$

$$\mathbf{v}\left(t+\frac{\Delta t}{2}\right) = \mathbf{v}(t) + \frac{1}{2}\mathbf{a}(t)\Delta t + O((\Delta t)^2)$$

$$\mathbf{v}(t+\Delta t) = \mathbf{v}\left(t+\frac{\Delta t}{2}\right) + \frac{1}{2}\mathbf{a}(t+\Delta t)\Delta t + O((\Delta t)^2)$$

$$\boxed{\begin{aligned} \mathbf{r}(t+\Delta t) &= \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)(\Delta t)^2 + O((\Delta t)^3) \\ \mathbf{v}(t+\Delta t) &= \mathbf{v}(t) + \frac{1}{2}\left[\mathbf{a}(t) + \mathbf{a}(t+\Delta t)\right]\Delta t + O((\Delta t)^3) \end{aligned}}$$

Molecular dynamics

Integration algorithms: Velocity Verlet

- Memory requirement: $9N$
- Simple and fast
- Forces are calculated once per time step
- Local [global] error $O((\Delta t^3))$ [$O((\Delta t^2))$]
- Symplectic
- Time –reversible
- Velocities and positions are calculated at the same times

Most widely used molecular dynamics algorithm

Molecular dynamics

Integration algorithms: Hamiltonian splitting methods (simplectic)

- For sampling, one wants a long trajectory → the integration algorithm must be stable
- A stable algorithm is not necessarily accurate
- Product-formula (splitting) approach allows the construction of stable algorithms with specified order of accuracy in the time step
- Unfortunately, we need a little more knowledge of classical mechanics
  - More complicated than the algorithms themselves

Molecular dynamics

Integration algorithms: Hamiltonian splitting methods (simplectic)

- The Hamilton equations of motions can be written as

Poisson brackets

$$\frac{dX_{n,\alpha}}{dt} = \left\{ X_{n,\alpha}, H \right\} \equiv \mathsf{L}\, X_{n,\alpha}$$

Liouville operator

$$= \frac{\partial X_{n,\alpha}}{\partial r_{n,\alpha}} \frac{\partial H}{\partial p_{n,\alpha}} - \frac{\partial H}{\partial r_{n,\alpha}} \frac{\partial X_{n,\alpha}}{\partial p_{n,\alpha}} \quad , \quad \text{where } X_{n,\alpha} = r_{n,\alpha} \text{ or } X_{n,\alpha} = p_{n,\alpha}$$

or

$$\boxed{\frac{dX_{n,\alpha}}{dt} = \mathsf{L}\, X_{n,\alpha} \Rightarrow X_{n,\alpha}(t) = e^{t\mathsf{L}}\, X_{n,\alpha}(0)}$$

Exponential of an operator

Molecular dynamics

Integration algorithms: Hamiltonian splitting methods (simplectic)

- Product-formula approach: Find a "useful" decomposition of the Liouville operator L

$$H(\mathbf{r},\mathbf{p}) = \frac{1}{2}\sum_{n=1}^{N}\mathbf{p}_n^2 + V(\mathbf{r}), \quad H_{\mathbf{p}} = \frac{1}{2}\sum_{n=1}^{N}\mathbf{p}_n^2, \quad H_{\mathbf{r}} = V(\mathbf{r}) \qquad (m_n = 1)$$

$$L_{\mathbf{p}} X_{n,\alpha} \equiv \{X_{n,\alpha}, H_{\mathbf{p}}\} = p_{n,\alpha}\frac{\partial X_{n,\alpha}}{\partial r_{n,\alpha}}, \quad L_{\mathbf{r}} X_{n,\alpha} \equiv \{X_{n,\alpha}, H_{\mathbf{r}}\} = F_{n,\alpha}(\mathbf{r})\frac{\partial X_{n,\alpha}}{\partial p_{n,\alpha}} \quad \Rightarrow \quad L = L_{\mathbf{p}} + L_{\mathbf{r}}$$

Note : $L_{\mathbf{p}} L_{\mathbf{p}} r_{n,\alpha} = L_{\mathbf{r}} L_{\mathbf{r}} p_{n,\alpha} = 0$

- Matrix notation

$$\frac{d}{dt}\begin{pmatrix} r_{n,\alpha}(t) \\ p_{n,\alpha}(t) \end{pmatrix} = \begin{pmatrix} L_{\mathbf{p}} & 0 \\ 0 & L_{\mathbf{r}} \end{pmatrix}\begin{pmatrix} r_{n,\alpha}(t) \\ p_{n,\alpha}(t) \end{pmatrix}$$

$$\begin{pmatrix} r_{n,\alpha}(t+\Delta t) \\ p_{n,\alpha}(t+\Delta t) \end{pmatrix} = e^{L\Delta t}\begin{pmatrix} r_{n,\alpha}(t) \\ p_{n,\alpha}(t) \end{pmatrix} = \exp\left(\Delta t\begin{pmatrix} L_{\mathbf{p}} & 0 \\ 0 & L_{\mathbf{r}} \end{pmatrix}\right)\begin{pmatrix} r_{n,\alpha}(t) \\ p_{n,\alpha}(t) \end{pmatrix}$$

Molecular dynamics

Integration algorithms: Hamiltonian splitting methods (simplectic)

- First-order product formula approximation yields the Euler-Cromer algorithm (two variants)

Also for matrices:

$$e^{tL} = \sum_{j=0}^{\infty} \frac{t^j}{j!} L^j$$

In this case:

$$e^{tL_{\mathbf{p}}} = 1 + tL_{\mathbf{p}} \quad (L_{\mathbf{p}}^2 X_{n,\alpha} = 0)$$

$$e^{A+B} \approx e^A e^B \text{ with } \mathbf{A} = \begin{pmatrix} L_p & 0 \\ 0 & 0 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 0 & 0 \\ 0 & L_r \end{pmatrix}$$

$$e^{(\Delta t)L} = \exp\left(\Delta t \begin{pmatrix} L_{\mathbf{p}} & 0 \\ 0 & L_{\mathbf{r}} \end{pmatrix}\right) \approx \exp\left(\Delta t \begin{pmatrix} L_{\mathbf{p}} & 0 \\ 0 & 0 \end{pmatrix}\right) \exp\left(\Delta t \begin{pmatrix} 0 & 0 \\ 0 & L_{\mathbf{r}} \end{pmatrix}\right) = \begin{pmatrix} 1+(\Delta t)L_{\mathbf{p}} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1+(\Delta t)L_{\mathbf{r}} \end{pmatrix}$$

$$\begin{pmatrix} r_{n,\alpha}(t+\Delta t) \\ p_{n,\alpha}(t+\Delta t) \end{pmatrix} \approx \begin{pmatrix} 1+(\Delta t)L_{\mathbf{p}} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1+(\Delta t)L_{\mathbf{r}} \end{pmatrix} \begin{pmatrix} r_{n,\alpha}(t) \\ p_{n,\alpha}(t) \end{pmatrix} = \begin{pmatrix} 1+(\Delta t)L_{\mathbf{p}} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} r_{n,\alpha}(t) \\ p_{n,\alpha}(t) + (\Delta t)F_{n,\alpha}(r_{n,\alpha}(t)) \end{pmatrix} = \begin{pmatrix} r_{n,\alpha}(t) + (\Delta t)(p_{n,\alpha}(t) + (\Delta t)F_{n,\alpha}(r_{n,\alpha}(t))) \\ p_{n,\alpha}(t) + (\Delta t)F_{n,\alpha}(r_{n,\alpha}(t)) \end{pmatrix}$$

OR

$$e^{A+B} \approx e^B e^A$$

$$e^{(\Delta t)L} = \exp\left(\Delta t \begin{pmatrix} L_{\mathbf{p}} & 0 \\ 0 & L_{\mathbf{r}} \end{pmatrix}\right) \approx \exp\left(\Delta t \begin{pmatrix} 0 & 0 \\ 0 & L_{\mathbf{r}} \end{pmatrix}\right) \exp\left(\Delta t \begin{pmatrix} L_{\mathbf{p}} & 0 \\ 0 & 0 \end{pmatrix}\right) = \begin{pmatrix} 1 & 0 \\ 0 & 1+(\Delta t)L_{\mathbf{r}} \end{pmatrix} \begin{pmatrix} 1+(\Delta t)L_{\mathbf{p}} & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} r_{n,\alpha}(t+\Delta t) \\ p_{n,\alpha}(t+\Delta t) \end{pmatrix} \approx \begin{pmatrix} 1 & 0 \\ 0 & 1+(\Delta t)L_{\mathbf{r}} \end{pmatrix} \begin{pmatrix} 1+(\Delta t)L_{\mathbf{p}} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} r_{n,\alpha}(t) \\ p_{n,\alpha}(t) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1+(\Delta t)L_{\mathbf{r}} \end{pmatrix} \begin{pmatrix} r_{n,\alpha}(t) + (\Delta t)p_{n,\alpha}(t) \\ p_{n,\alpha}(t) \end{pmatrix} = \begin{pmatrix} r_{n,\alpha}(t) + (\Delta t)p_{n,\alpha}(t) \\ p_{n,\alpha}(t) + \Delta t F_{n,\alpha}(r_{n,\alpha}(t) + (\Delta t)p_{n,\alpha}(t)) \end{pmatrix}$$

Molecular dynamics

Integration algorithms: Hamiltonian splitting methods (simplectic)

- Second-order product formula approximation: velocity Verlet algorithm

$$e^{A+B} \approx e^{B/2} e^{A} e^{B/2}$$

$$e^{\Delta t L} = \exp\left(\Delta t \begin{pmatrix} L_{\mathbf{p}} & 0 \\ 0 & L_{\mathbf{r}} \end{pmatrix}\right) \approx \exp\left(\frac{\Delta t}{2}\begin{pmatrix} 0 & 0 \\ 0 & L_{\mathbf{r}} \end{pmatrix}\right)\exp\left(\Delta t \begin{pmatrix} L_{\mathbf{p}} & 0 \\ 0 & 0 \end{pmatrix}\right)\exp\left(\frac{\Delta t}{2}\begin{pmatrix} 0 & 0 \\ 0 & L_{\mathbf{r}} \end{pmatrix}\right)$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1+(\Delta t)L_{\mathbf{r}}/2 \end{pmatrix}\begin{pmatrix} 1+(\Delta t)L_{\mathbf{p}} & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 1+(\Delta t)L_{\mathbf{r}}/2 \end{pmatrix}$$

$$\begin{pmatrix} r_{n,\alpha}(t+\Delta t) \\ p_{n,\alpha}(t+\Delta t) \end{pmatrix} \approx \begin{pmatrix} 1 & 0 \\ 0 & 1+(\Delta t)L_{\mathbf{r}}/2 \end{pmatrix}\begin{pmatrix} 1+(\Delta t)L_{\mathbf{p}} & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 1+(\Delta t)L_{\mathbf{r}}/2 \end{pmatrix}\begin{pmatrix} r_{n,\alpha}(t) \\ p_{n,\alpha}(t) \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1+(\Delta t)L_{\mathbf{r}}/2 \end{pmatrix}\begin{pmatrix} 1+(\Delta t)L_{\mathbf{p}} & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} r_{n,\alpha}(t) \\ p_{n,\alpha}(t)+(\Delta t)F_{n,\alpha}(r_{n,\alpha}(t))/2 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1+(\Delta t)L_{\mathbf{r}}/2 \end{pmatrix}\begin{pmatrix} r_{n,\alpha}(t)+\Delta t(p_{n,\alpha}(t)+(\Delta t)F_{n,\alpha}(r_{n,\alpha}(t))/2) \\ p_{n,\alpha}(t)+(\Delta t)F_{n,\alpha}(r_{n,\alpha}(t))/2 \end{pmatrix}$$

$$= \begin{pmatrix} r_{n,\alpha}(t)+\Delta t(p_{n,\alpha}(t)+(\Delta t)F_{n,\alpha}(r_{n,\alpha}(t))/2) \\ p_{n,\alpha}(t)+(\Delta t)F_{n,\alpha}(r_{n,\alpha}(t))/2+(\Delta t)F_{n,\alpha}(r_{n,\alpha}(t)+\Delta t(p_{n,\alpha}(t)+(\Delta t)F_{n,\alpha}(r_{n,\alpha}(t))/2))/2 \end{pmatrix}$$

Summary

- Molecular dynamics simulations
  - Boundary conditions
  - Forces
  - Initial conditions
  - Integration algorithms
    - Euler
    - Euler-Cromer
    - Leap-frog
    - Verlet
    - Velocity Verlet
    - Hamiltonian splitting methods (simplectic)