Lecture 9
CH-4114
Molecular Simulation

“Everything that living things do can be understood in terms of the jigglings and wigglings of atoms.”

- Richard P. Feynman

By
Dr. Susmita Roy
IISER-Kolkata

IISER KOLKATA

The core idea behind Leapfrog Algorithm:

A naive method (like **forward Euler**) computes:

The leapfrog algorithm improves numerical stability by staggering position and velocity updates in time (half-step apart), avoiding error-prone subtraction of small terms from large ones, and conserving energy over long simulations.

A native method (like forward Euler) computes:

$$x(t+\delta t) = x(t) + v(t)\delta t$$

$$v(t+\delta t) = v(t) + \frac{F(t)}{m}\delta t$$

- **Positions** are evaluated at integer timesteps: $x(t), x(t+\delta t), x(t+2\delta t), \ldots$
- **Velocities** are evaluated at half-integer timesteps: $v(t+\frac{1}{2}\delta t), v(t+\frac{3}{2}\delta t), \ldots$

So they "leapfrog" over each other in time.

1. **Half-step velocity update:**

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

2. **Position update:**

$$x(t+\delta t) = x(t) + v\left(t+\tfrac{1}{2}\delta t\right)\delta t$$

Leapfrog Algorithm

- Eliminates addition of small numbers O(δt^2) to differences in large ones O(δt^0)
- Algorithm

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

$$\mathbf{v}(t+\tfrac{1}{2}\delta t) = \boxed{\mathbf{v}(t-\tfrac{1}{2}\delta t) + \tfrac{1}{m}\mathbf{F}(t)\delta t}$$

- Mathematically equivalent to Verlet algorithm

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \left[\mathbf{v}(t-\tfrac{1}{2}\delta t) + \tfrac{1}{m}\mathbf{F}(t)\delta t\right]\delta t$$

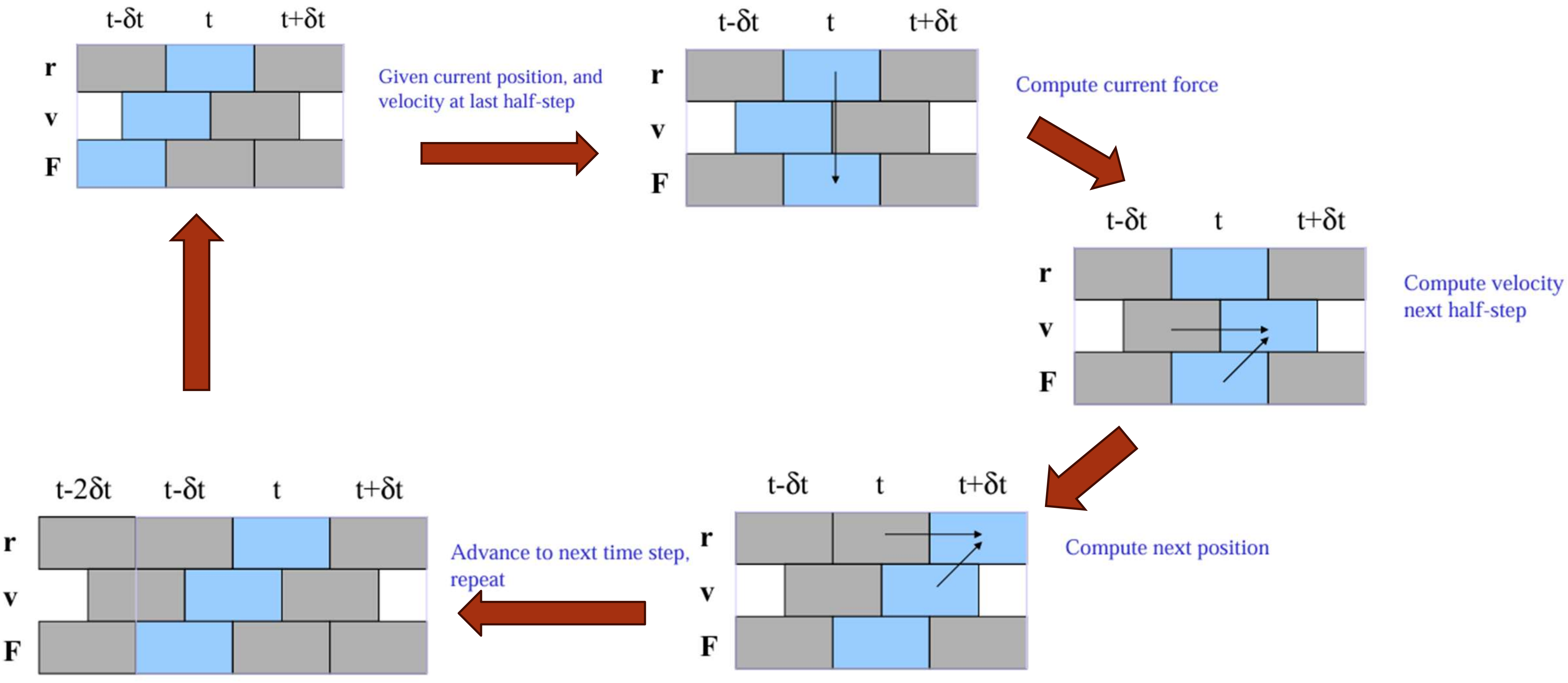
r(t) as evaluated from previous time step

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

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \left[\left(\mathbf{r}(t) - \mathbf{r}(t-\delta t)\right) + \tfrac{1}{m}\mathbf{F}(t)\delta t^2\right]$$

$$\mathbf{r}(t+\delta t) = 2\mathbf{r}(t) - \mathbf{r}(t-\delta t) + \tfrac{1}{m}\mathbf{F}(t)\delta t^2$$ original Verlet algorithm

Leapfrog Algorithm: Flow Diagram


t-δt
t
t+δt
r
v
F
Given current position, and velocity at last half-step
Compute current force
Compute velocity next half-step
Compute next position
Advance to next time step, repeat
t-2δt

Schematic from Allen & Tildesley, Computer Simulation of Liquids

Leapfrog Algorithm Loose Ends

- Initialization
  - how to get velocity at “previous time step” when starting out?
  - simple approximation

$$\mathbf{v}(t_0 - \delta t) = \mathbf{v}(t_0) - \tfrac{1}{m}\mathbf{F}(t_0)\tfrac{1}{2}\delta t$$

- Obtaining the velocities
  - interpolate

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

Velocity Verlet Algorithm

1) Forward Taylor expansions (about $t = 0$)

Position:

$$r(\Delta t) = r(0) + v(0)\Delta t + \tfrac{1}{2}a(0)\Delta t^2 + \tfrac{1}{6}j(0)\Delta t^3 + O(\Delta t^4), \qquad \text{Eqn.1}$$

where $j = \dot{a}$

Velocity:

$$v(\Delta t) = v(0) + a(0)\Delta t + \tfrac{1}{2}j(0)\Delta t^2 + O(\Delta t^3). \qquad \text{Eqn.2}$$

The position expansion accurate up to the Δt^2-term gives a position update that is correct to second order.

2) Backward Taylor expansion (expand $r(0)$ about $t = \Delta t$)

Expand $r(0)$ using a Taylor series backward from $r(\Delta t)$:

$$r(0) = r(\Delta t) - v(\Delta t)\Delta t + \tfrac{1}{2}a(\Delta t)\Delta t^2 - \tfrac{1}{6}j(\Delta t)\Delta t^3 + O(\Delta t^4). \qquad \text{Eqn.3}$$

Rearrange this to solve for $v(\Delta t)$:

$$v(\Delta t) = \frac{r(\Delta t) - r(0)}{\Delta t} + \tfrac{1}{2}a(\Delta t)\Delta t + O(\Delta t^2). \qquad \text{Eqn.4}$$

This is exact up to the retained order.

3) Substitute the forward position increment into **Eqn.4**

From the forward expansion we have

$$r(\Delta t) - r(0) = v(0)\Delta t + \tfrac{1}{2}a(0)\Delta t^2 + \tfrac{1}{6}j(0)\Delta t^3 + O(\Delta t^4).$$

Divide by Δt:

$$\frac{r(\Delta t) - r(0)}{\Delta t} = v(0) + \tfrac{1}{2}a(0)\Delta t + \tfrac{1}{6}j(0)\Delta t^2 + O(\Delta t^3).$$

Now plug into **Eqn.4**

$$\begin{aligned} v(\Delta t) &= \left[v(0) + \tfrac{1}{2}a(0)\Delta t + \tfrac{1}{6}j(0)\Delta t^2 + O(\Delta t^3)\right] + \tfrac{1}{2}a(\Delta t)\Delta t + O(\Delta t^2) \\ &= v(0) + \tfrac{\Delta t}{2}\left[a(0) + a(\Delta t)\right] + \underbrace{\left(\tfrac{1}{6}j(0)\Delta t^2 + O(\Delta t^2)\right)}_{=O(\Delta t^2)}. \end{aligned}$$

The leftover terms are $O(\Delta t^2)$ in the local truncation, so to the order we care about we get the compact and very useful form:

$$\boxed{v(\Delta t) = v(0) + \tfrac{\Delta t}{2}\left[a(0) + a(\Delta t)\right] + O(\Delta t^2)} \Rightarrow$$

(so the velocity update is accurate to second order in Δt).

4) The position update

From the forward expansion truncated to second order we get the standard position update:

$$\boxed{r(\Delta t) = r(0) + v(0)\Delta t + \tfrac{1}{2}a(0)\Delta t^2} \qquad (+\,O(\Delta t^3)\text{ terms neglected}$$

Putting Together:

Velocity Verlet Algorithm:

1. **Position update:**

$$r(t+\Delta t) \;=\; r(t) + v(t)\,\Delta t + \tfrac{1}{2}\,a(t)\,\Delta t^2$$

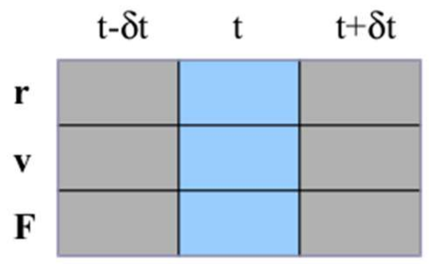
2. **Force/acceleration evaluation at new position:**

$$a(t+\Delta t) \;=\; \frac{F\big(r(t+\Delta t)\big)}{m}$$

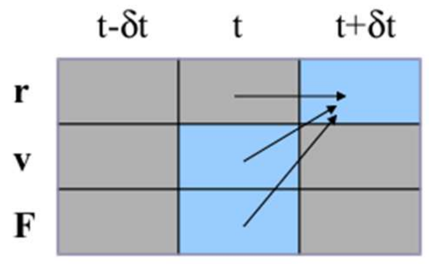
3. **Velocity update:**

$$v(t+\Delta t) \;=\; v(t) + \tfrac{\Delta t}{2}\,\big(a(t) + a(t+\Delta t)\big)$$

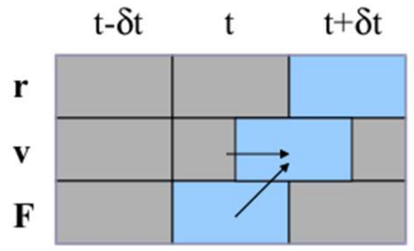
Velocity Verlet Algorithm: Flow Diagram



Given current position, velocity and force.

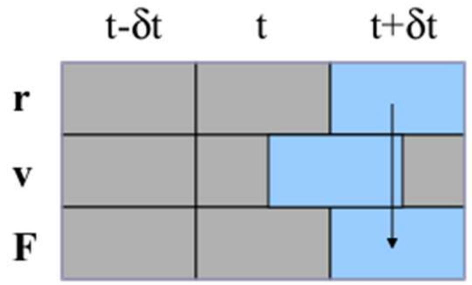


Compute new position

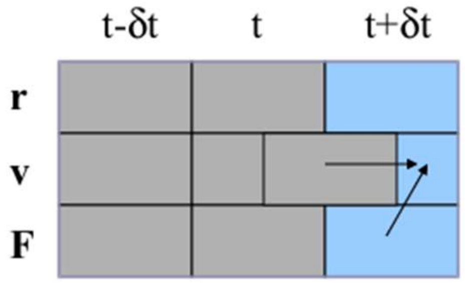


Compute velocity at half step

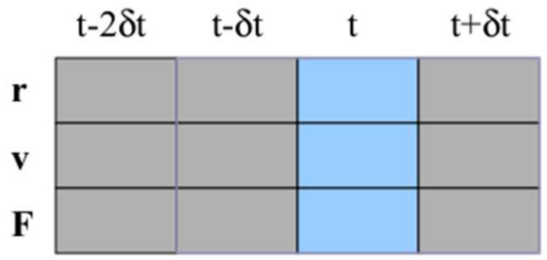
Let’s think about this scheme



Compute force at new position



Compute velocity at full step



Advance to next time step and repeat

Schematic from Allen & Tildesley, Computer Simulation of Liquids

Velocity Verlet Algorithm: Error/Properties

- **Second-order accurate** in time (global error $O(\Delta t^2)$).
- **Time-reversible** and **symplectic** (good long-term energy behaviour).
- Only **one force evaluation** per step (efficient).
- Produces velocities and positions at the *same* times (unlike leapfrog, which staggers them by half a step).
- Equivalent to leapfrog if you choose to store half-step velocities:

$$v\left(t+\tfrac{\Delta t}{2}\right)=v(t)+\tfrac{\Delta t}{2}a(t),\quad r(t+\Delta t)=r(t)+v\left(t+\tfrac{\Delta t}{2}\right)\Delta t,$$

$$v(t+\Delta t)=v\left(t+\tfrac{\Delta t}{2}\right)+\tfrac{\Delta t}{2}a(t+\Delta t).$$