CE 530 Molecular Simulation

Lecture 11
Molecular Dynamics Simulation

David A. Kofke
Department of Chemical Engineering
SUNY Buffalo
kofke@eng.buffalo.edu

Review and Preview

- MD of hard disks
  - *intuitive*
  - *collision detection and impulsive dynamics*
- Monte Carlo
  - *convenient sampling of ensembles*
  - *no dynamics*
  - *biasing possible to improve performance*
- Molecular dynamics
  - *equations of motion*
  - *integration schemes*
  - *evaluation of dynamical properties*
  - *extensions to other ensembles*
  - *focus on atomic systems for now*

Classical Equations of Motion

- Several formulations are in use
  - *Newtonian*
  - *Lagrangian*
  - *Hamiltonian*
- Advantages of non-Newtonian formulations
  - *more general, no need for "fictitious" forces*
  - *better suited for multiparticle systems*
  - *better handling of constraints*
  - *can be formulated from more basic postulates*
- Assume conservative forces

$$\vec{\mathbf{F}} = -\vec{\nabla} U$$ *Gradient of a scalar potential energy*

Newtonian Formulation

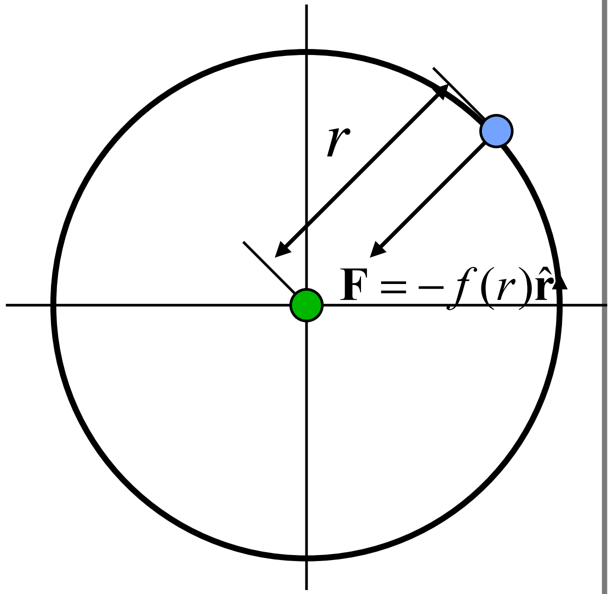
- Cartesian spatial coordinates $\mathbf{r}_i = (x_i, y_i, z_i)$ are primary variables
  - *for N atoms, system of N 2nd-order differential equations*

$$m\frac{d^2\mathbf{r}_i}{dt^2} \equiv m\ddot{\mathbf{r}}_i = \mathbf{F}_i$$

- Sample application: 2D motion in central force field

$$m\ddot{x} = \mathbf{F}\cdot\hat{\mathbf{e}}_x = -f(r)\hat{\mathbf{r}}\cdot\hat{\mathbf{e}}_x = -xf\left(\sqrt{x^2+y^2}\right)$$

$$m\ddot{y} = \mathbf{F}\cdot\hat{\mathbf{e}}_y = -f(r)\hat{\mathbf{r}}\cdot\hat{\mathbf{e}}_y = -yf\left(\sqrt{x^2+y^2}\right)$$



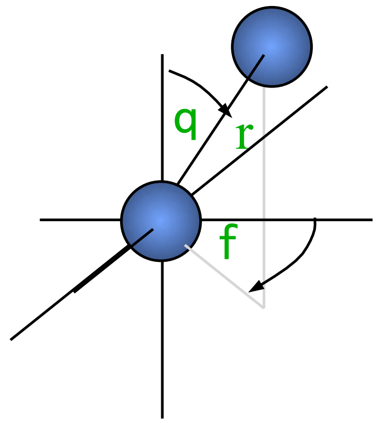
  - *Polar coordinates are more natural and convenient*

$$\boxed{mr^2\dot{\theta} = \ell}$$ *constant angular momentum*

$$\boxed{m\ddot{r} = -f(r) + \boxed{\frac{\ell^2}{mr^3}}}$$ *fictitious (centrifugal) force*

Generalized Coordinates

- Any convenient coordinates for description of particular system
  - *use q_i as symbol for general coordinate*
  - *examples*
    - diatomic $\{q_1,\ldots,q_6\} = \{x_{com}, y_{com}, z_{com}, r_{12}, q, f\}$
    - 2-D motion in central field $\{q_1, q_2\} = \{r, q\}$



- Kinetic energy
  - *general quadratic form*

$$K = \underbrace{M_0(\mathbf{q}) + \sum M_j(\mathbf{q})\dot{q}_j}_{\textit{usually vanish}} + \tfrac{1}{2}\sum\sum M_{jk}(\mathbf{q})\dot{q}_j\dot{q}_k$$

  - *examples*
    - rotating diatomic $K = \tfrac{1}{2}m\left(\dot{q}_1^2 + \dot{q}_2^2 + \dot{q}_3^2\right) + \tfrac{1}{8}m\left[\dot{r}^2 + r^2\dot{\theta}^2 + (r\sin\theta)^2\dot{\phi}^2\right]$
    - 2-D central motion $K = \tfrac{1}{2}m\left(\dot{r}^2 + r^2\dot{\theta}^2\right)$

Lagrangian Formulation

- Independent of coordinate system
- Define the Lagrangian
  - $L(\mathbf{q},\dot{\mathbf{q}}) \equiv K(\mathbf{q},\dot{\mathbf{q}}) - U(\mathbf{q})$
- Equations of motion

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_j}\right) - \frac{\partial L}{\partial q_j} = 0 \quad j = 1\ldots N$$

  - *N second-order differential equations*
- Central-force example

$$L = \tfrac{1}{2}m\left(\dot{r}^2 + r^2\dot{\theta}^2\right) - U(r)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{r}}\right) = \frac{\partial L}{\partial r} \Rightarrow \boxed{m\ddot{r} = mr\dot{\theta}^2 - f(r)} \qquad \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = \frac{\partial L}{\partial \theta} \Rightarrow \boxed{\frac{d}{dt}\left(mr^2\dot{\theta}\right) = 0}$$

$\vec{\mathbf{F}}_r = -\vec{\nabla}_r U = -f(r)$

Hamiltonian Formulation 1. Motivation

- Appropriate for application to statistical mechanics and quantum mechanics
- Newtonian and Lagrangian viewpoints take the q_i as the fundamental variables
  - *N-variable configuration space*
  - $\dot{q}_i$ *appears only as a convenient shorthand for dq/dt*
  - *working formulas are 2nd-order differential equations*
- Hamiltonian formulation seeks to work with 1st-order differential equations
  - *2N variables*
  - *treat the coordinate and its time derivative as independent variables*
  - *appropriate quantum-mechanically*

Hamiltonian Formulation 2. Preparation

- Mathematically, Lagrangian treats q and $\dot{q}$ as distinct
  - $L(q_j, \dot{q}_j, t)$
  - *identify the generalized momentum as* $\boxed{p_j = \dfrac{\partial L}{\partial \dot{q}_j}}$
  - *e.g.* if $L = K - U = \frac{1}{2} m\dot{q}^2 - U(q)$; $p = \partial L / \partial \dot{q} = m\dot{q}$
  - *Lagrangian equations of motion* $\dfrac{dp_j}{dt} = \dfrac{\partial L}{\partial q_j}$
- We would like a formulation in which p is an independent variable
  - p_i *is the derivative of the Lagrangian with respect to* $\dot{q}_i$, *and we're looking to replace* $\dot{q}_i$ *with* p_i
  - *we need …?*

Hamiltonian Formulation 3. Defintion

- …a Legendre transform!
- Define the *Hamiltonian, H*

$$
\begin{aligned}
H(\mathbf{q},\mathbf{p}) &= -\left[L(\mathbf{q},\dot{\mathbf{q}}) - \sum p_j \dot{q}_j\right] \\
&= -K(\mathbf{q},\dot{\mathbf{q}}) + U(\mathbf{q}) + \sum \frac{\partial K}{\partial \dot{q}_j}\dot{q}_j \\
&= -\sum a_j \dot{q}_j^2 + U(\mathbf{q}) + \sum (2a_j\dot{q}_j)\dot{q}_j \\
&= +\sum a_j \dot{q}_j^2 + U(\mathbf{q}) \\
&= K + U
\end{aligned}
$$

- H equals the total energy (kinetic plus potential)

Hamiltonian Formulation 4. Dynamics

- Hamilton's equations of motion
  - *From Lagrangian equations, written in terms of momentum*

Differential change in L

$$dL = \frac{\partial L}{\partial q}dq + \frac{\partial L}{\partial \dot{q}}d\dot{q}$$
$$= \dot{p}\,dq + p\,d\dot{q}$$

Legendre transform

$$H = -(L - p\dot{q})$$
$$dH = -(\dot{p}\,dq - \dot{q}\,dp)$$
$$dH = -\dot{p}\,dq + \dot{q}\,dp$$

$$\frac{dp}{dt} = \dot{p} = \frac{\partial L}{\partial q}$$ Lagrange's equation of motion

$$p = \frac{\partial L}{\partial \dot{q}}$$ Definition of momentum

$$\left\{\begin{array}{l} \dot{q} = +\dfrac{\partial H}{\partial p} \\ \dot{p} = -\dfrac{\partial H}{\partial q} \end{array}\right.$$ Hamilton's equations of motion

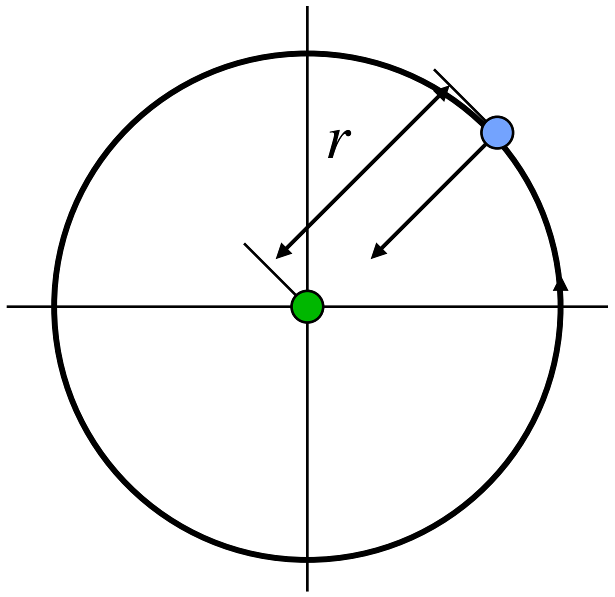
Conservation of energy $$\frac{dH}{dt} = -\dot{p}\frac{dq}{dt} + \dot{q}\frac{dp}{dt} = -\dot{p}\dot{q} + \dot{q}\dot{p} = 0$$

Hamiltonian Formulation 5. Example

- Particle motion in central force field

$$H = K + U$$

$$= \frac{p_r^2}{2m} + \frac{p_\theta^2}{2mr^2} + U(r)$$



$$\dot{q} = +\frac{\partial H}{\partial p}$$

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

$$(1)\ \frac{dr}{dt} = \frac{p_r}{m} \qquad (2)\ \frac{d\theta}{dt} = \frac{p_\theta}{mr^2}$$

$$(3)\ \frac{dp_r}{dt} = \frac{p_\theta^2}{mr^3} - f(r) \qquad (4)\ \frac{dp_\theta}{dt} = 0$$

$$\vec{\mathbf{F}}_r = -\vec{\nabla}_r U = -f(r)$$

Lagrange's equations

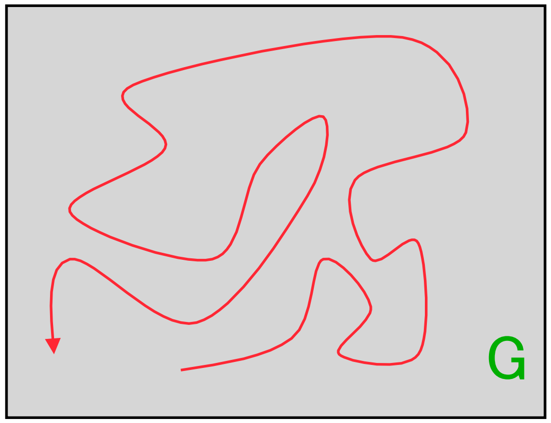
$$\boxed{m\ddot{r} = mr\dot{\theta}^2 - f(r)}$$

$$\boxed{\frac{d}{dt}\left(mr^2\dot{\theta}\right) = 0}$$

- Equations no simpler, but theoretical basis is better

Phase Space (again)

- Return to the complete picture of phase space
  - *full specification of microstate of the system is given by the values of all positions and all momenta of all atoms*
    - ➜ $G = (p^N, r^N)$
  - *view positions and momenta as completely independent coordinates*
    - ➜ connection between them comes only through equation of motion
- Motion through phase space
  - *helpful to think of dynamics as "simple" movement through the high -dimensional phase space*
    - ➜ facilitate connection to quantum mechanics
    - ➜ basis for theoretical treatments of dynamics
    - ➜ understanding of integrators



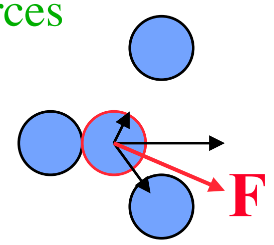
Integration Algorithms

○ Equations of motion in cartesian coordinates

$$\boxed{\begin{aligned}\frac{d\mathbf{r}_j}{dt} &= \frac{\mathbf{p}_j}{m} \\ \frac{d\mathbf{p}_j}{dt} &= \mathbf{F}_j\end{aligned}}$$

$$\left.\begin{aligned}\mathbf{r} &= (r_x, r_y) \\ \mathbf{p} &= (p_x, p_y)\end{aligned}\right\}$$ 2-dimensional space (for example)

$$\mathbf{F}_j = \sum_{\substack{i=1 \\ i\neq j}}^{N} \mathbf{F}_{ij}$$ pairwise additive forces



○ Desirable features of an integrator

- *minimal need to compute forces (a very expensive calculation)*
- *good stability for large time steps*
- *good accuracy*
- *conserves energy and momentum*
- *time-reversible*
- *area-preserving (symplectic)*

$\left.\begin{array}{l}\text{time-reversible}\\ \text{area-preserving (symplectic)}\end{array}\right\}$ More on these later

Verlet Algorithm
1. Equations

- Very simple, very good, very popular algorithm
- Consider expansion of coordinate forward and backward in time

$$\mathbf{r}(t+\delta t)=\mathbf{r}(t)+\tfrac{1}{m}\mathbf{p}(t)\delta t+\tfrac{1}{2m}\mathbf{F}(t)\delta t^2+\tfrac{1}{3!}\dddot{\mathbf{r}}(t)\delta t^3+O(\delta t^4)$$

$$\mathbf{r}(t-\delta t)=\mathbf{r}(t)-\tfrac{1}{m}\mathbf{p}(t)\delta t+\tfrac{1}{2m}\mathbf{F}(t)\delta t^2-\tfrac{1}{3!}\dddot{\mathbf{r}}(t)\delta t^3+O(\delta t^4)$$

- Add these together

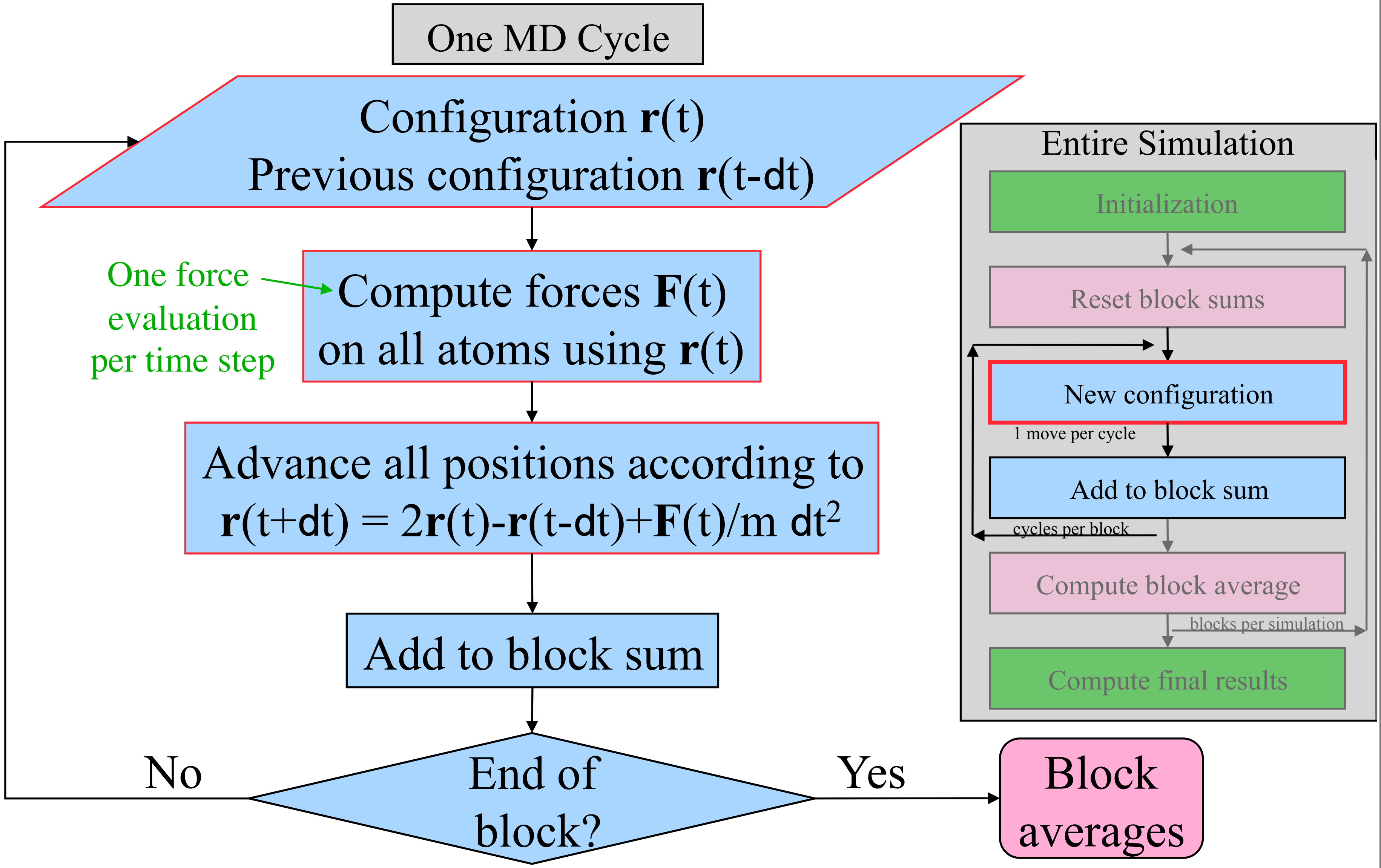
$$\mathbf{r}(t+\delta t)+\mathbf{r}(t-\delta t)=2\mathbf{r}(t)+\tfrac{1}{m}\mathbf{F}(t)\delta t^2+O(\delta t^4)$$

- Rearrange

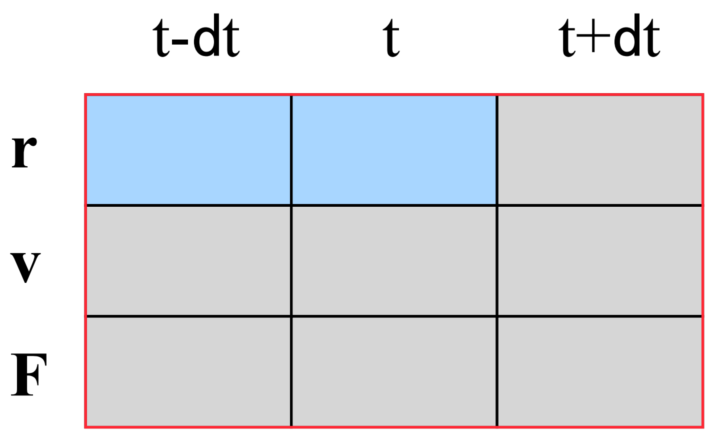
$$\boxed{\mathbf{r}(t+\delta t)=2\mathbf{r}(t)-\mathbf{r}(t-\delta t)+\tfrac{1}{m}\mathbf{F}(t)\delta t^2}+O(\delta t^4)$$

  - *update without ever consulting velocities!*

Verlet Algorithm 2. Flow diagram

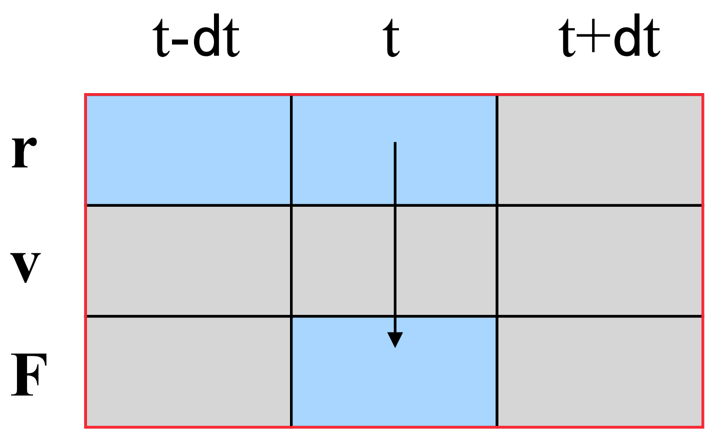


Verlet Algorithm 2. Flow Diagram



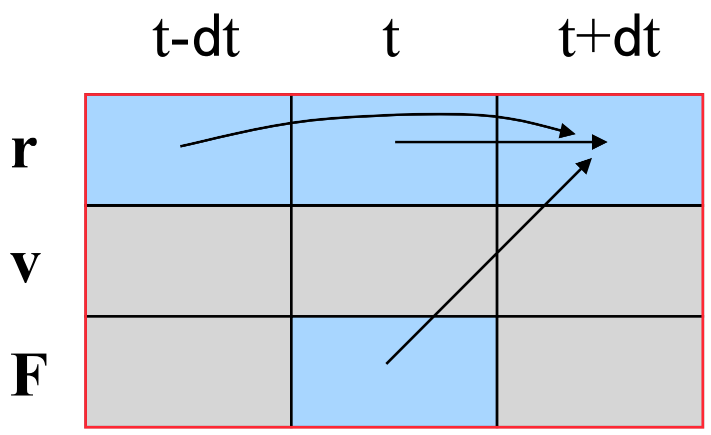
Given current position and position at end of previous time step

Verlet Algorithm 2. Flow Diagram



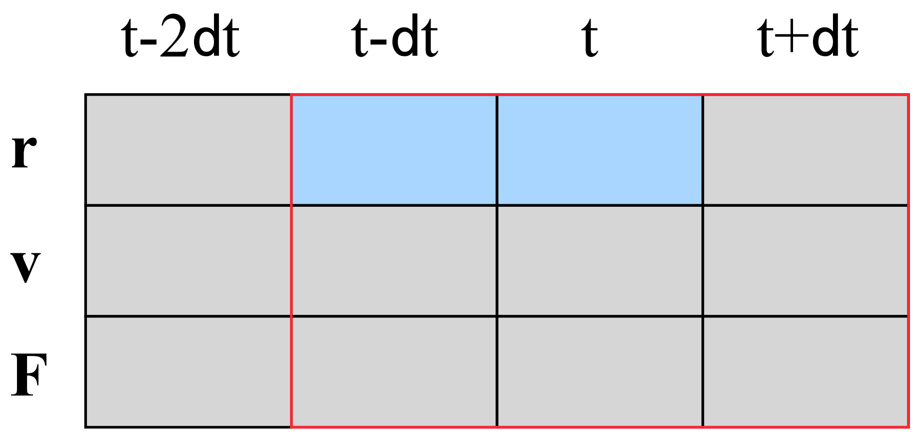
Compute the force at the current position

Verlet Algorithm 2. Flow Diagram



Compute new position from present and previous positions, and present force

Verlet Algorithm 2. Flow Diagram

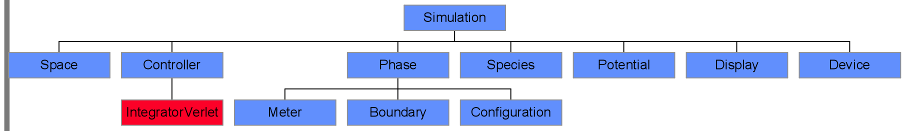


Advance to next time step, repeat

Schematic from Allen & Tildesley, Computer Simulation of Liquids

Verlet Algorithm 3. Java Code

User's Perspective on the Molecular Simulation API



Verlet Algorithm
3. Relevant Methods in Java Code

public class IntegratorVerlet extends Integrator

```
//Performs one timestep increment in the Verlet algorithm
public void doStep(double tStep) {

  atomIterator.reset();
  while(atomIterator.hasNext()) {   //zero forces on all atoms
   ((Agent)atomIterator.next().ia).force.E(0.0);  //integratorVerlet.Agent keeps a force Vector
  }
  pairIterator.allPairs(forceSum); //sum forces on all pairs

  double t2 = tStep*tStep;
  atomIterator.reset();
  while(atomIterator.hasNext()) {  //loop over all atoms, moving according to Verlet
      Atom a = atomIterator.next();
      Agent agent = (Agent)a.ia;
      Space.Vector r = a.position();  //current position of the atom
      temp.E(r);                      //save it
      r.TE(2.0);                      //2*r
      r.ME(agent.rLast);              //2*r-rLast
      agent.force.TE(a.rm()*t2);      // f/m dt^2
      r.PE(agent.force);              //2*r - rLast + f/m dt^2
      agent.rLast.E(temp);            //rLast gets present r
  }
  return;
}
```

Verlet Algorithm
3. Relevant Methods in Java Code

public class IntegratorVerlet extends Integrator

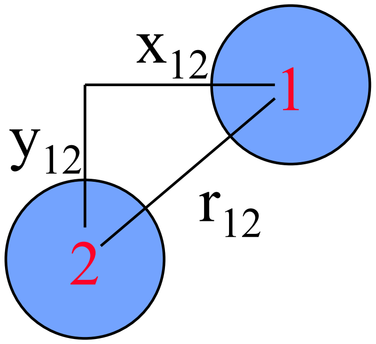
```
//(anonymous) class for incrementing the sum of the forces on each atom
forceSum = new AtomPair.Action() {
  private Space.Vector f = simulation().space.makeVector();
  public void action(AtomPair pair) {
  PotentialSoft potential = (PotentialSoft)simulation().getPotential(pair) //identify pot'l
  f.E(potential.force(pair));            //compute force of atom1 on atom2
  ((Agent)pair.atom1().ia).force.PE(f); //increment atom1 force
  ((Agent)pair.atom2().ia).force.ME(f); //increment atom2 force
  }
};

//Agent class for IntegratorVerlet; stores useful quantities in each Atom
public final static class Agent implements Integrator.Agent {
        public Atom atom;
        public Space.Vector force; //used to accumulate the force on the atom
        public Space.Vector rLast; //holds the position of the atom at the last step

        public Agent(Atom a) {     //constructor
            atom = a;
            force = atom.parentMolecule().parentPhase().parentSimulation.space.makeVector();
            rLast = atom.parentMolecule().parentPhase().parentSimulation.space.makeVector();
        }
    }
```

Forces 1. Formalism

- Force is the gradient of the potential

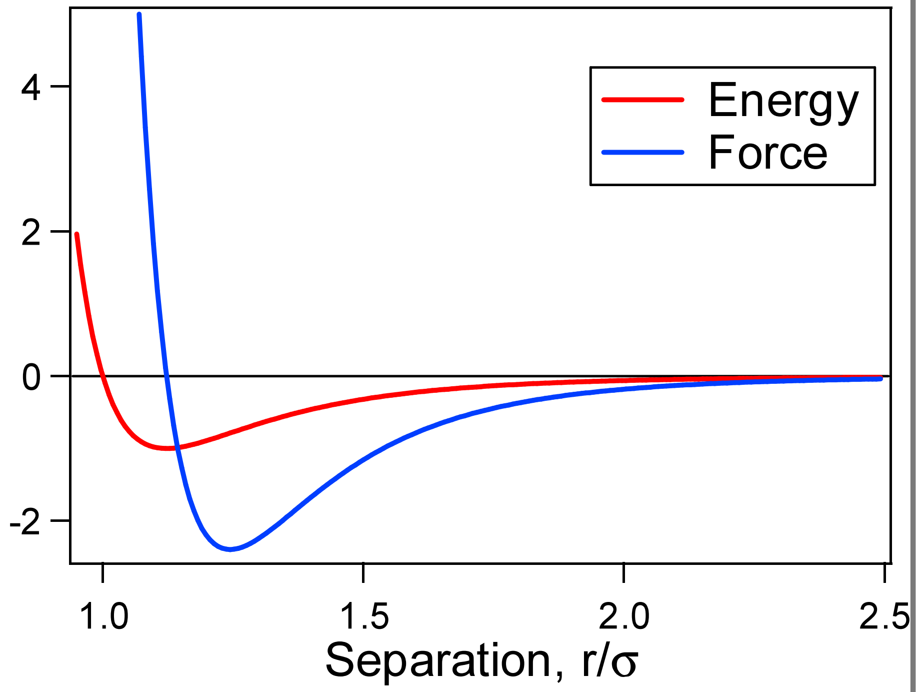
$$
\begin{aligned}
\mathbf{F}_{2\to1} &= -\nabla u(r_{12}) \\
&= -\frac{\partial u(r_{12})}{\partial x_1}\mathbf{e}_x - \frac{\partial u(r_{12})}{\partial y_1}\mathbf{e}_y \\
&= -\frac{du(r_{12})}{dr_{12}}\left[\frac{\partial r_{12}}{\partial x_1}\mathbf{e}_x + \frac{\partial r_{12}}{\partial y_1}\mathbf{e}_y\right] \\
&= -\frac{f(r_{12})}{r_{12}}\left[x_{12}\mathbf{e}_x + y_{12}\mathbf{e}_y\right]
\end{aligned}
$$

Force on 1, due to 2



$$r_{12} = \left[(x_2 - x_1)^2 + (y_2 - y_1)^2\right]^{1/2}$$

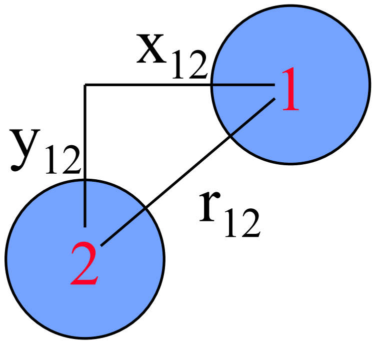
$$\mathbf{F}_{2\to1} = -\mathbf{F}_{1\to2}$$



Forces 2. LJ Model

- Force is the gradient of the potential

$$\mathbf{F}_{2\to 1} = -\frac{f(r_{12})}{r_{12}}\left[x_{12}\mathbf{e}_x + y_{12}\mathbf{e}_y\right]$$



$$r_{12} = \left[(x_2 - x_1)^2 + (y_2 - y_1)^2\right]^{1/2}$$

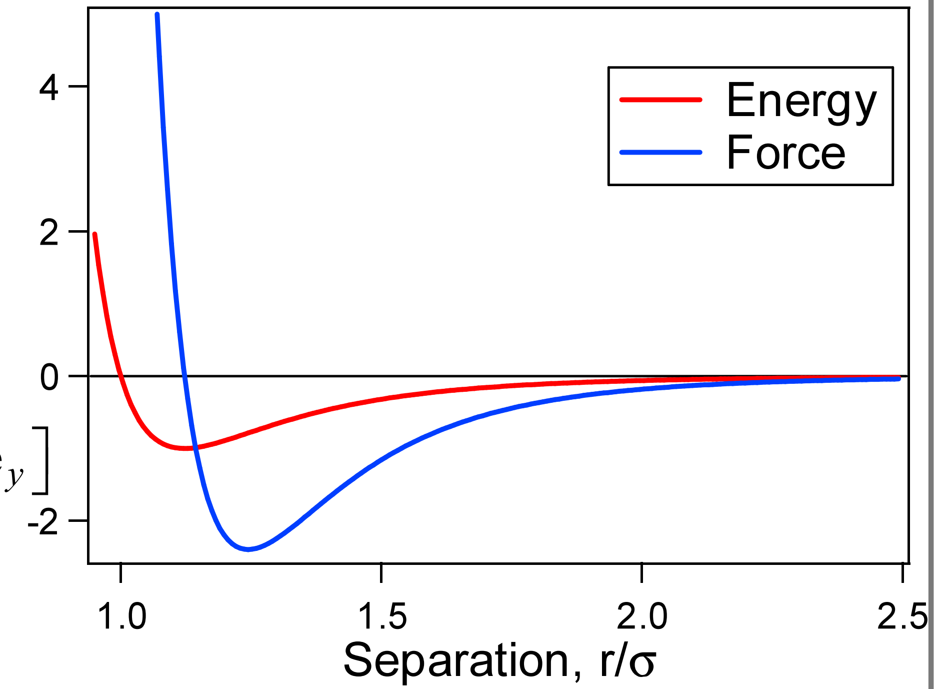
e.g., Lennard-Jones model

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

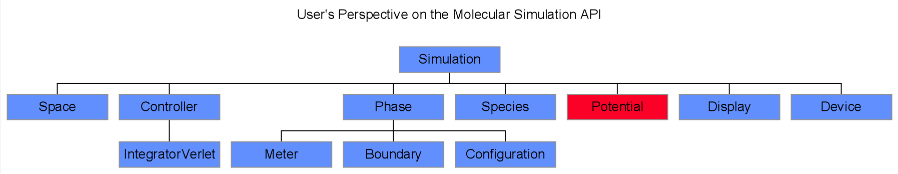
$$f(r) = -\frac{du}{dr}$$

$$= +\frac{48\varepsilon}{\sigma}\left[\left(\frac{\sigma}{r}\right)^{13} - \frac{1}{2}\left(\frac{\sigma}{r}\right)^{7}\right]$$

$$\mathbf{F}_{2\to 1} = -\frac{48\varepsilon}{\sigma^2}\left[\left(\frac{\sigma}{r_{12}}\right)^{14} - \frac{1}{2}\left(\frac{\sigma}{r_{12}}\right)^{8}\right]\left[x_{12}\mathbf{e}_x + y_{12}\mathbf{e}_y\right]$$



Forces 3. Java Code



Forces
3. Relevant Methods from Java Code

```
public class PotentialLJ implements PotentialSoft
```

```
//Space.Vector used to compute and return a force
private Space.Vector force = Simulation.space.makeVector();

public Space.Vector force(AtomPair pair) {
  double r2 = pair.r2();      //squared distance between pair of atoms
  if(r2 > cutoffDiameterSquared) {force.E(0.0);} //outside cutoff; no interaction
  else {
    double s2 = sigmaSquared/r2;  // (sigma/r)^2
    double s6 = s2*s2*s2;         // (sigma/r)^6
    force.E(pair.dr());           // f = (x12 ex + y12 ey)  (vector)
    force.TE(-48*s2*s6*(s6-0.5)/sigmaSquared);
                       // f *= -48*(sigma/r)^8 * [(sigma/r)^6 - 1/2] / sigma^2
  }
  return force;
}
```

Verlet Algorithm. 4. Loose Ends

- Initialization
  - *how to get position at "previous time step" when starting out?*
  - *simple approximation*

$$\mathbf{r}(t_0 - \delta t) = \mathbf{r}(t_0) - \mathbf{v}(t_0)\delta t$$

- Obtaining the velocities
  - *not evaluated during normal course of algorithm*
  - *needed to compute some properties, e.g.*
    - temperature
    - diffusion constant
  - *finite difference*

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

Verlet Algorithm 5. Performance Issues

- Time reversible
  - *forward time step*

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

  - *replace dt with −dt*

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

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

  - *same algorithm, with same positions and forces, moves system backward in time*
- Numerical imprecision of adding large/small numbers

$$\overbrace{\boxed{\mathbf{r}(t+\delta t)-\mathbf{r}(t)}}^{O(dt^1)} = \overbrace{\boxed{\underbrace{\boxed{\mathbf{r}(t)}}_{O(dt^0)} - \underbrace{\boxed{\mathbf{r}(t-\delta t)}}_{O(dt^0)}}}^{O(dt^1)} + \underbrace{\boxed{\tfrac{1}{m}\mathbf{F}(t)\delta t^2}}_{O(dt^2)}$$

Initial Velocities

(from Lecture 3)

- Random direction
  - *randomize each component independently*
  - *randomize direction by choosing point on spherical surface*
- Magnitude consistent with desired temperature. Choices:
  - *Maxwell-Boltzmann:* $prob(v_x) \propto \exp\left(-\frac{1}{2} m v_x^2 / kT\right)$
  - *Uniform over (-1/2,+1/2), then scale so that* $\frac{1}{N}\sum v_{i,x}^2 = kT/m$
  - *Constant at* $v_x = \pm\sqrt{kT/m}$
  - *Same for y, z components*
- Be sure to shift so center-of-mass momentum is zero

$$P_x \equiv \frac{1}{N}\sum p_{i,x}$$

$$p_{i,x} \rightarrow p_{i,x} - P_x$$

Leapfrog Algorithm

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

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

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

Leapfrog Algorithm

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

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

Leapfrog Algorithm

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

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

Leapfrog Algorithm

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

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

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

Leapfrog Algorithm

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

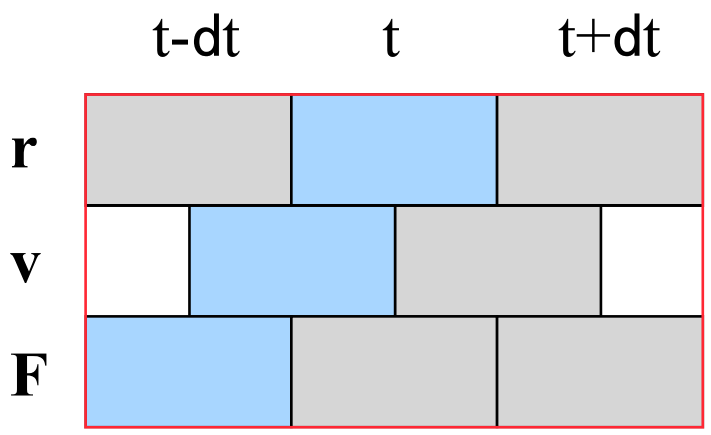
$$\overbrace{\mathbf{r}(t) = \mathbf{r}(t-\delta t)}\, + \mathbf{v}(t-\tfrac{1}{2}\delta t)\delta t$$

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

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

original algorithm

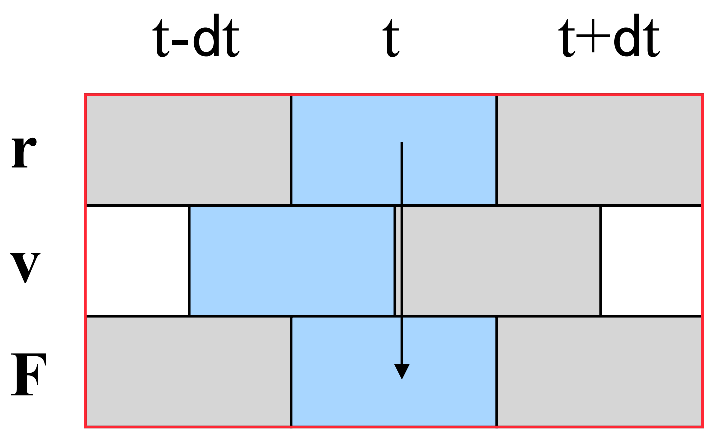
Leapfrog Algorithm 2. Flow Diagram



Given current position, and
velocity at last half-step

Schematic from Allen & Tildesley, Computer Simulation of Liquids

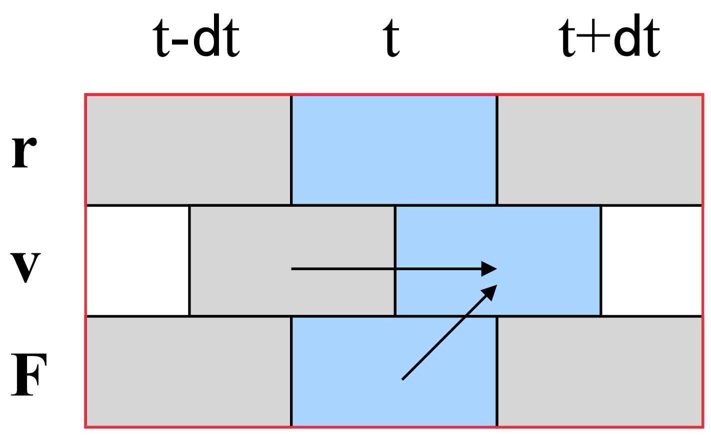
Leapfrog Algorithm 2. Flow Diagram



Compute current force

Schematic from Allen & Tildesley, <u>Computer Simulation of Liquids</u>

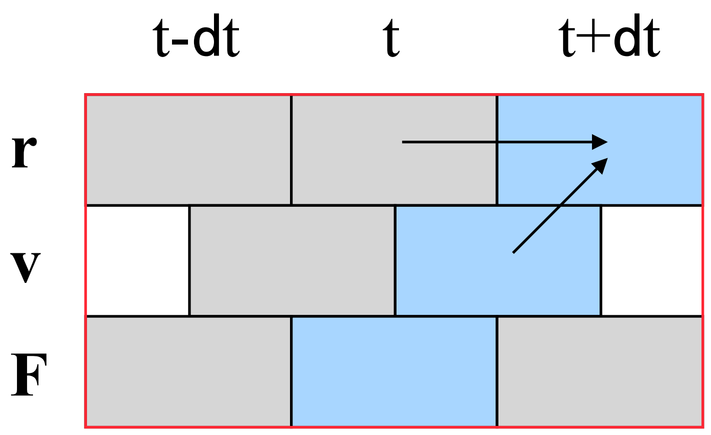
Leapfrog Algorithm 2. Flow Diagram



Compute velocity at
next half-step

Schematic from Allen & Tildesley, Computer Simulation of Liquids

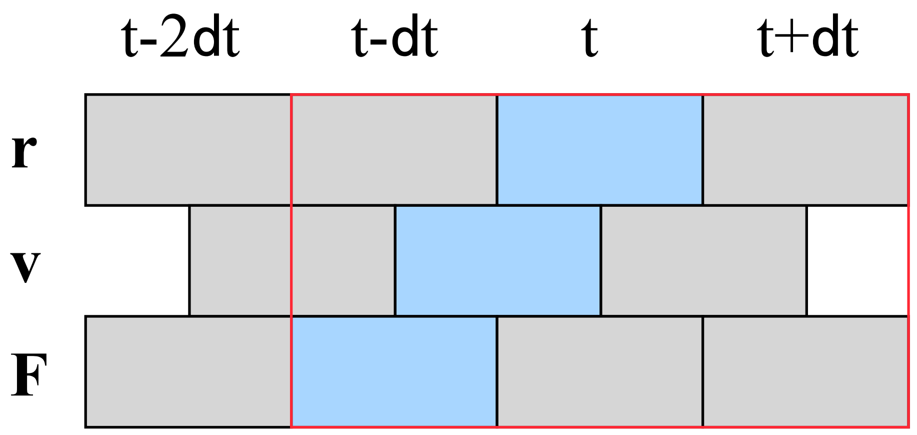
Leapfrog Algorithm 2. Flow Diagram



Compute next position

Schematic from Allen & Tildesley, Computer Simulation of Liquids

Leapfrog Algorithm 2. Flow Diagram



Advance to next time step, repeat

Schematic from Allen & Tildesley, Computer Simulation of Liquids

Leapfrog Algorithm. 3. 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

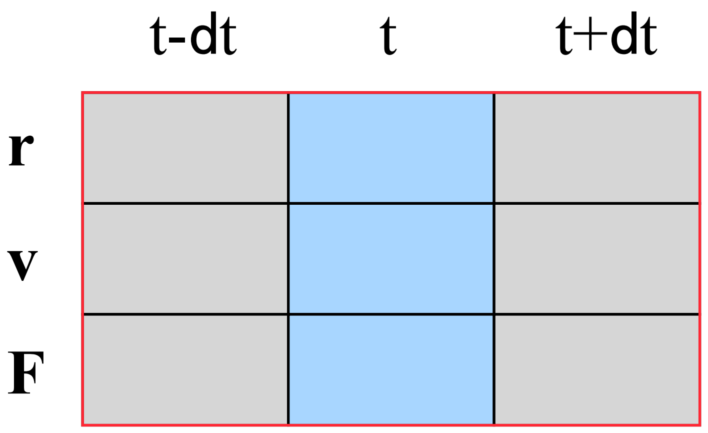
- Roundoff advantage of leapfrog, but better treatment of velocities
- Algorithm

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

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

- Implemented in stages
  - *evaluate current force*
  - *compute* **r** *at new time*
  - *add current-force term to velocity (gives* **v** *at half-time step)*
  - *compute new force*
  - *add new-force term to velocity*
- Also mathematically equivalent to Verlet algorithm (in giving values of **r**)

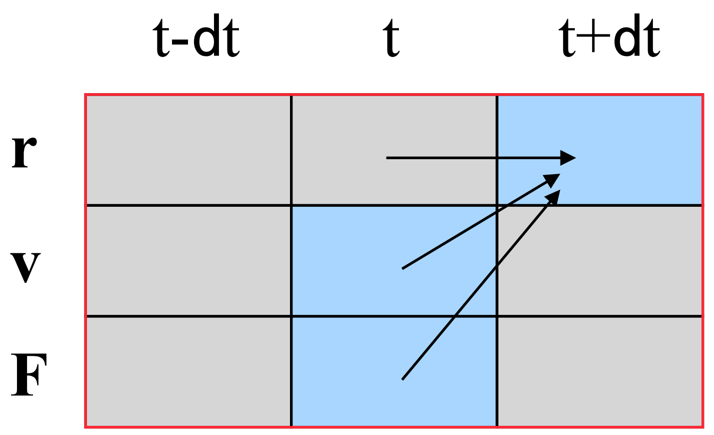
Velocity Verlet Algorithm
2. Flow Diagram



Given current position, velocity, and force

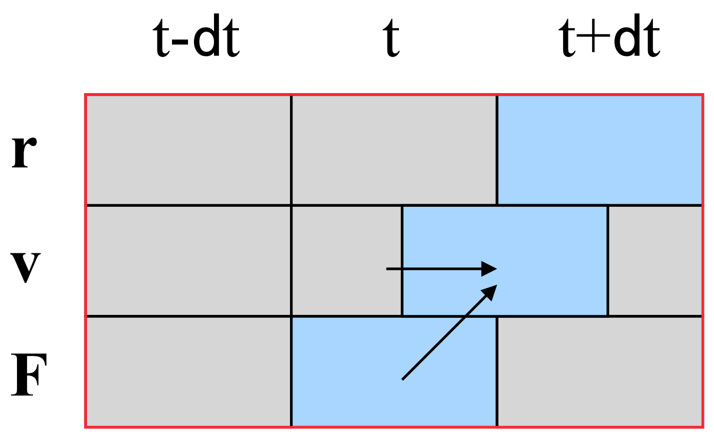
Schematic from Allen & Tildesley, Computer Simulation of Liquids

Velocity Verlet Algorithm
2. Flow Diagram



Compute new position

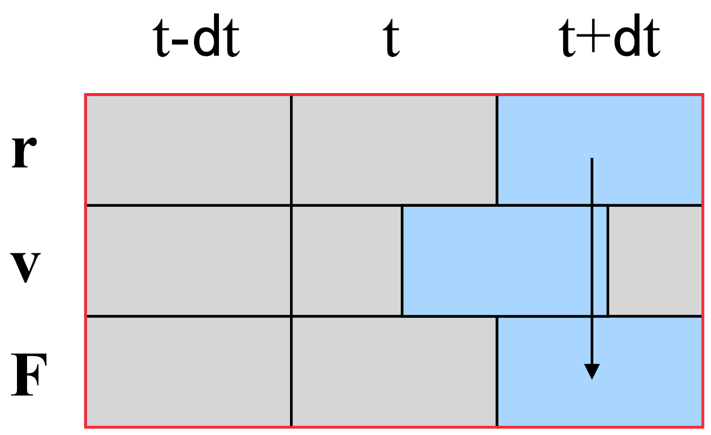
Velocity Verlet Algorithm
2. Flow Diagram



Compute velocity at half step

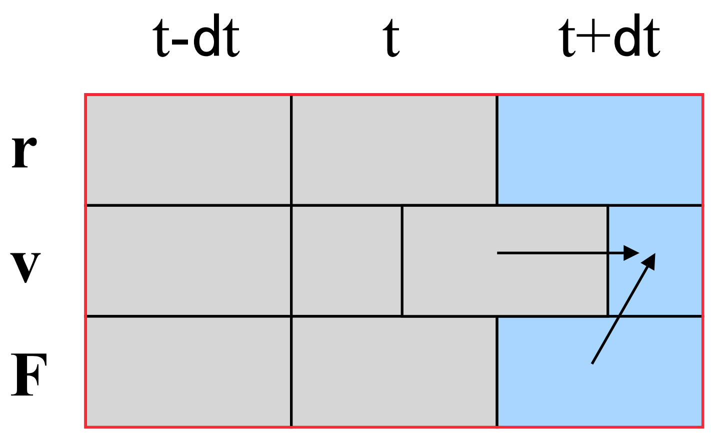
Schematic from Allen & Tildesley, Computer Simulation of Liquids

Velocity Verlet Algorithm 2. Flow Diagram



Compute force at new position

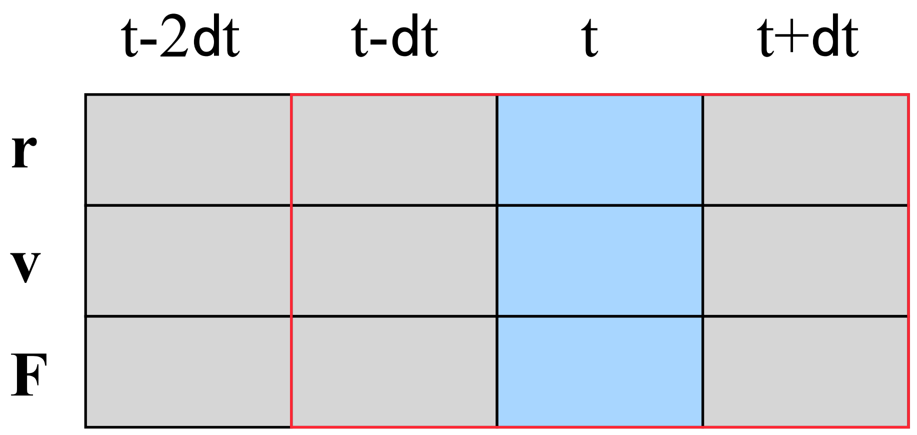
Velocity Verlet Algorithm
2. Flow Diagram



Compute velocity at full step

Schematic from Allen & Tildesley, Computer Simulation of Liquids

Velocity Verlet Algorithm 2. Flow Diagram



Advance to next time step, repeat

Schematic from Allen & Tildesley, Computer Simulation of Liquids

Other Algorithms

- Predictor-Corrector
  - *not time reversible*
  - *easier to apply in some instances*
    - constraints
    - rigid rotations
- Beeman
  - *better treatment of velocities*
- Velocity-corrected Verlet

Summary

- Several formulations of mechancs
  - *Hamiltonian preferred*
    - independence of choice of coordinates
    - emphasis on phase space
- Integration algorithms
  - *Calculation of forces*
  - *Simple Verlet algorithsm*
    - Verlet
    - Leapfrog
    - Velocity Verlet
- Next up: Calculation of dynamical properties