Modeling & Simulation of Glass Structure

VCG Lecture 20

John Kieffer

Department of Materials Science and Engineering

University of Michigan

Overview …

- Historical perspective
- Simulation methodologies
  - Theoretical background
  - Monte Carlo simulation
  - Molecular dynamics simulation
  - Reverse Monte Carlo
- Force fields
- Information retrieval
  - Statistical mechanical formalisms
  - Structural analyses
  - Dynamics
- Application examples

Implementing Metropolis MC

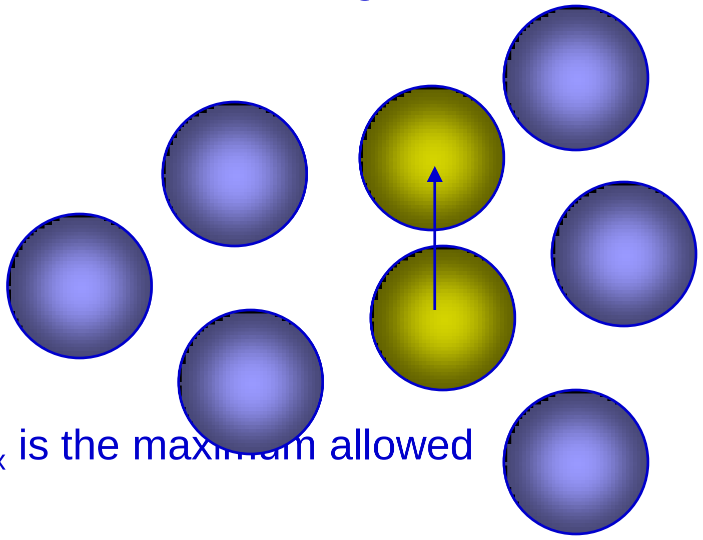
A MC simulation of a material is easy to perform.

- At each iteration of the simulation, a new configuration is generated. This is usually done by making a random change to the coordinates of a single randomly chosen particle using a random number generator (RNG).

  E.g. $x_{new} = x_{old} + \eta_1 \delta r_{max}$

  $y_{new} = y_{old} + \eta_2 \delta r_{max}$

  $z_{new} = z_{old} + \eta_3 \delta r_{max}$



  Here $\eta = 2\xi - 1$ and ξ is in the range (0,1). Δr_{max} is the maximum allowed displacement in any direction.

- A unique random number is generated for each direction.

Implementing Metropolis MC

- The potential energy of the new configuration is then calculated.
  - Since the change from the previous configuration involves the motion of a single particle, only those contributions to the energy resulting from the particle's new position need to be recalculated.
  - If periodic boundaries are used, the minimum image convention must be invoked in calculating the energy (more on this later).

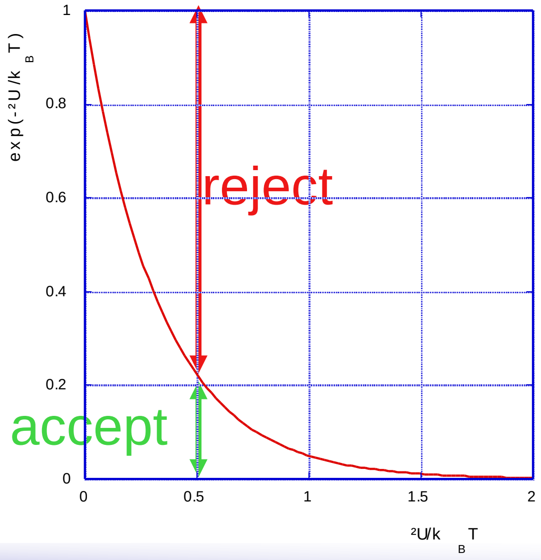
Implementing Metropolis MC

- If the new configuration is lower in energy than the previous configuration, then the new configuration is retained as the starting point for the next iteration.

- If not, then the Boltzmann factor $\exp(-[U_{new}-U_{old}]/k_BT)$ is calculated and compared to a random number between 0 and 1.
  - If the random number is less than or equal to the Boltzmann factor, the new configuration is accepted.
  - If the random number is greater than the Boltzmann factor, the new configuration is rejected and the previous configuration is retained for the next iteration.

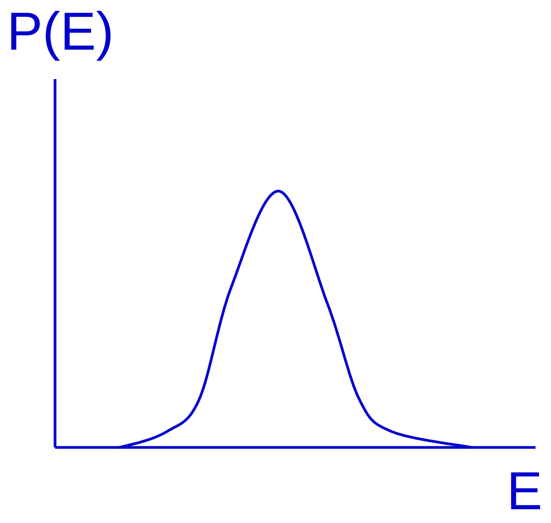
The acceptance criterion can be written as:

$$\text{rand}(0,1) \leq \exp(-\Delta U(\mathbf{r}^N)/k_BT)$$



Detailed Balance

- The acceptance criterion in Metropolis Monte Carlo is derived by imposing the condition of detailed balance
- It assures unique limiting probability distribution



N possible states

Probability that state *i* is occupied:

$\rho_i = Z^{-1} \cdot \exp[-U_i/k_B T]$

Probability distribution of states

$\rho = (\rho_1, \rho_2, \rho_3, \ldots, \rho_n, \ldots, \rho_m, \ldots, \rho_N)$

Transition matrix π:

π_{mn} = probability that system transitions from state *m* to *n*

Stochastic matrix α:

α_{mn} = probability that algorithm seeks transition from state *m* to *n*

Detailed Balance

In the Metropolis algorithm:

$$\pi_{mn} = \alpha_{mn} \quad \text{if } \rho_n \geq \rho_m \text{ and } n \neq m$$

$$\pi_{mn} = \alpha_{mn} \frac{\rho_n}{\rho_m} \quad \text{if } \rho_n < \rho_m \text{ and } n \neq m$$

$$\frac{\rho_n}{\rho_m} = \exp\left[-\left(U_n - U_m\right)/k_B T\right]$$

$$\pi_{mm} = 1 - \sum_{n \neq m} \pi_{mn}$$

Detailed Balance

Example: 2-level system; state 1 twice as likely as state 2

$$\frac{\rho_2}{\rho_1} = \exp\left[-\left(U_2 - U_1\right)/k_B T\right] = 0.5$$

$$\alpha_{12} = \alpha_{21} = 1$$

$$\pi_{12} = \alpha_{12} \exp\left[-\left(U_2 - U_1\right)/k_B T\right] = 0.5$$

$$\pi_{21} = \alpha_{21} = 1$$

$$\pi_{11} = 1 - \pi_{12} = 0.5$$

$$\pi_{22} = 1 - \pi_{21} = 0$$

Detailed Balance

Assume $\rho(1) = (\rho_1, \rho_2) = (1,0)$

$$\pi = \begin{pmatrix} 0.5 & 0.5 \\ 1 & 0 \end{pmatrix}$$

$$\rho(2) = \rho(1)\pi = [0.5, 0.5]$$
$$\rho(3) = \rho(2)\pi = [0.75, 0.25]$$
$$\rho(4) = \rho(3)\pi = [0.625, 0.375]$$
$$\rho(5) = \rho(4)\pi = [0.6875, 0.3125]$$
$$\ldots$$
$$\rho(n+1) = \rho(n)\pi = [0.666, 0.333]$$

Metropolis MC: Markov Chains

- The Metropolis algorithm generates a Markov chain of states of the system.

- A Markov chain satisfies the following two conditions:
  - The outcome of each trial depends only upon the preceding trial and not on any other previous trials.
  - Each trial belongs to a finite set of possible outcomes.

- The first condition distinguishes the MC method from the MD method: in an MD simulation, all states are connected deterministically in time.

Detailed Balance

- Why accept any moves that raise the energy?
  - Consider the alternative: any move that lowers the energy is accepted, any move that raises the energy is rejected. The consequence of this is that *thermal fluctuations would not be allowed, and phase space would not be properly explored* at the given temperature.
    - This would be equivalent to simulating at zero temperature.
  - In any system in equilibrium at temperature T, thermal fluctuations occur that occasionally raise the energy. These fluctuations allow the system to explore phase space and allow all atoms to experience all environments.

Implementing Metropolis MC

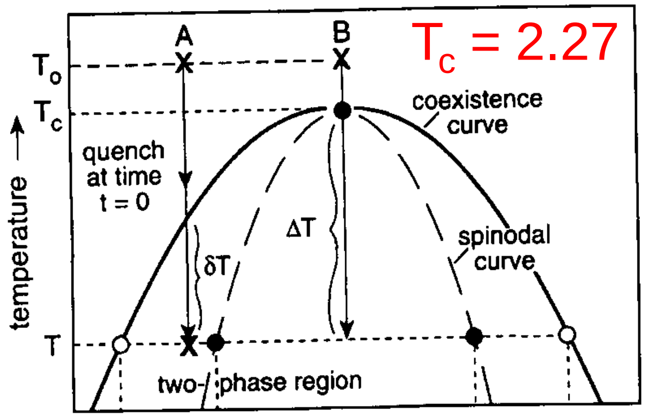
Choosing the size of the displacement:

- At each iteration, the size of the move is governed by the *maximum displacement* δr_{max}.
- The max displacement δr_{max} is an adjustable parameter whose value is usually chosen so that 50% of the trial moves are accepted.
  - ❖ If δr_{max} is too small, then many moves will be accepted but the states generated will be very similar to the previous ones and the system will explore phase space very slowly.
  - ❖ If δr_{max} is too large, then very few moves will be accepted because they lead to unfavorable overlaps.
  - ❖ δr_{max} can be adjusted automatically while the program is running to achieve the desired acceptance ratio by keeping a running score of the proportion of moves that are accepted.

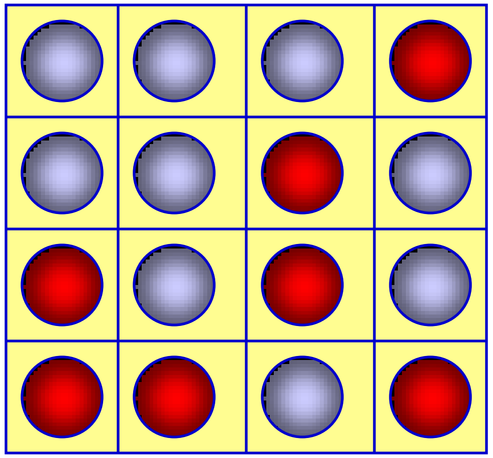
Example:
MC Simulation of a Binary Mixture

- Example: A binary mixture of two species with interaction energies ε_{AA}, ε_{BB}, and ε_{AB} and identical molecular size σ. The values of ε characterize the well depths of the van der Waals interactions between any two molecules.

- To simplify, discretize space and allow the molecules to move only by discrete amounts. That is, allow the molecules to move on the sites of a lattice.



ϕ_A



Example:

$\varepsilon_{AA} = -0.5$

$\varepsilon_{BB} = -0.5$

$\varepsilon_{AB} = -1$

$\varepsilon = \varepsilon_{AB} - (\varepsilon_{AA} + \varepsilon_{BB})/2$

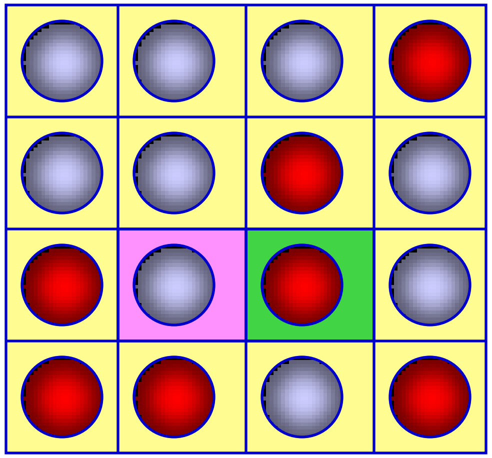
Energy of mixing

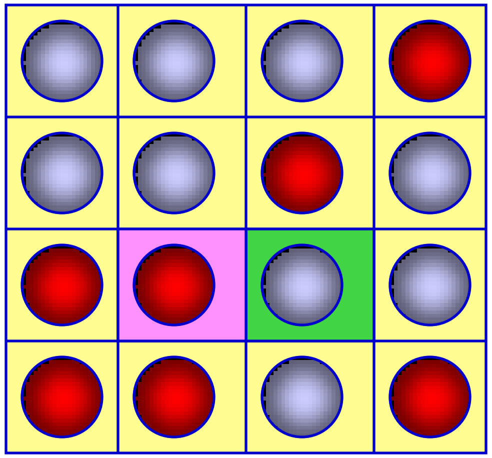
MC Simulation of a Binary Mixture

1. Choose a particle at random (pink square).
2. Choose a neighboring particle at random (green square).
   1. This defines the MC "move" - we will attempt to swap these two particles.
3. Calculate the current and new potential energy of the two particles.

$$E_{\text{pair}} = \left(\sum_{N_{AA}} \varepsilon_{AA} + \sum_{N_{BB}} \varepsilon_{BB} + \sum_{N_{AB}} \varepsilon_{AB} \right) \qquad \Delta E = E^{new}_{pair} - E^{old}_{pair}$$

Old

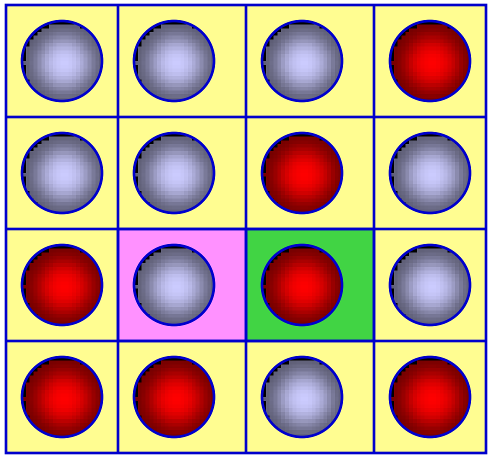


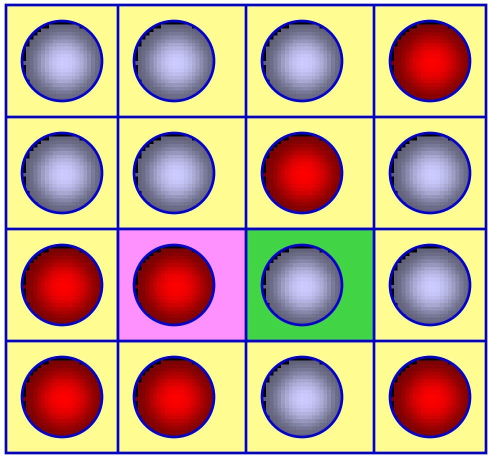


New

MC Simulation of a Binary Mixture

4. If $\Delta E < 0$, accept the move. If not, calculate Boltzmann factor and compare with a random number. Accept if $\mathrm{rand}(0,1) \leq \exp(-\Delta E/k_B T)$.

Old 

 New

5. Note bounds: If T very large, then BF close to one, and move nearly always accepted. If T very small, then BF is very small, and move is rarely accepted.

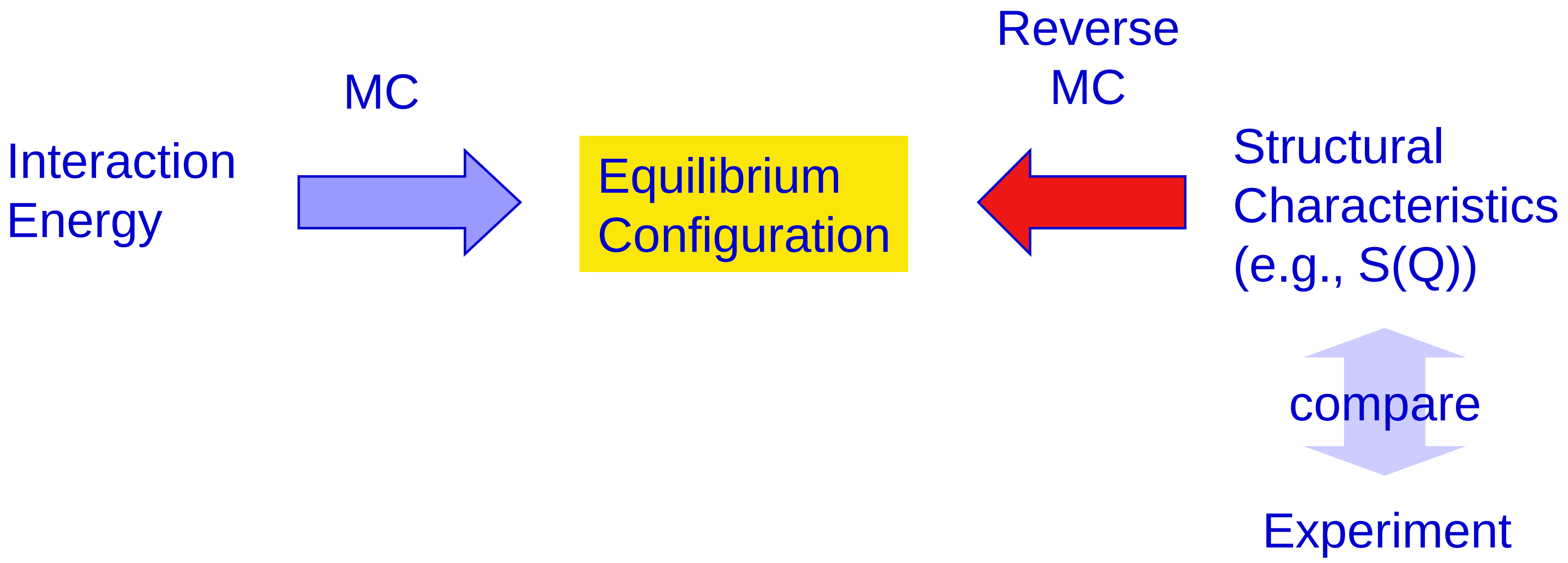
Questions?

Overview …

- Historical perspective
- Simulation methodologies
  - Theoretical background
  - Monte Carlo simulation
  - Molecular dynamics simulation
  - Reverse Monte Carlo
- Force fields
- Information retrieval
  - Statistical mechanical formalisms
  - Structural analyses
  - Dynamics
- Application examples

Reverse Monte Carlo

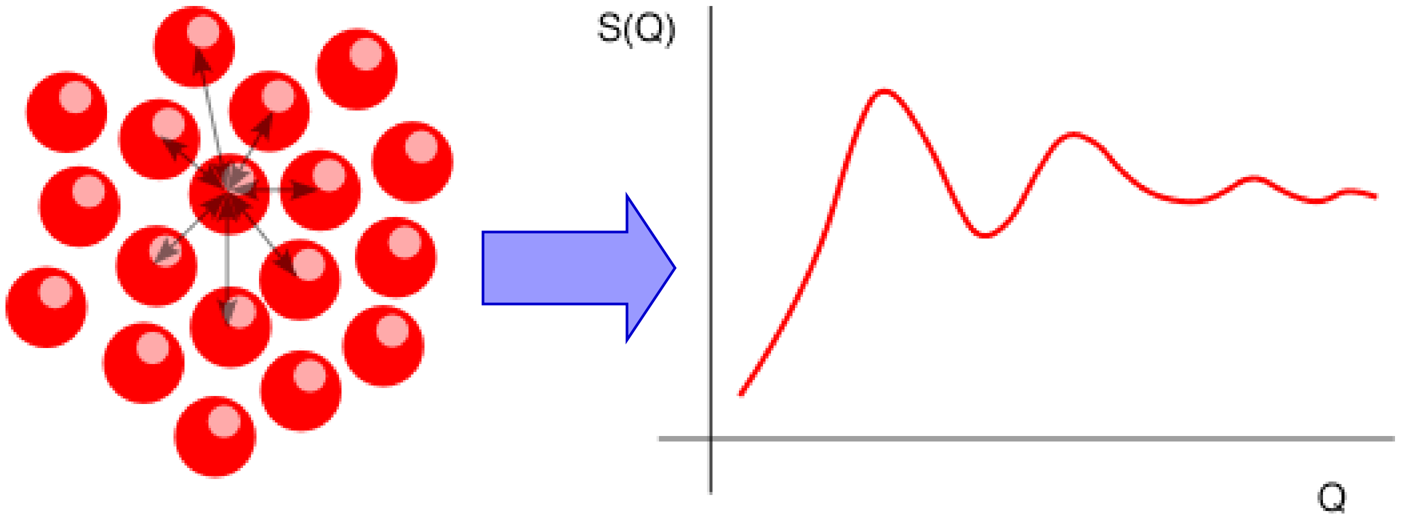
Method introduced by Robert McGreevy[†]


Interaction Energy
MC
Equilibrium Configuration
Reverse MC
Structural Characteristics (e.g., S(Q))
compare
Experiment

[†] R.L. McGreevy and L. Pusztai, Mol. Sim. **1**, 359 (1988); R.L. McGreevy, J. Phys. Cond. Matter **13**, R887 (2001)

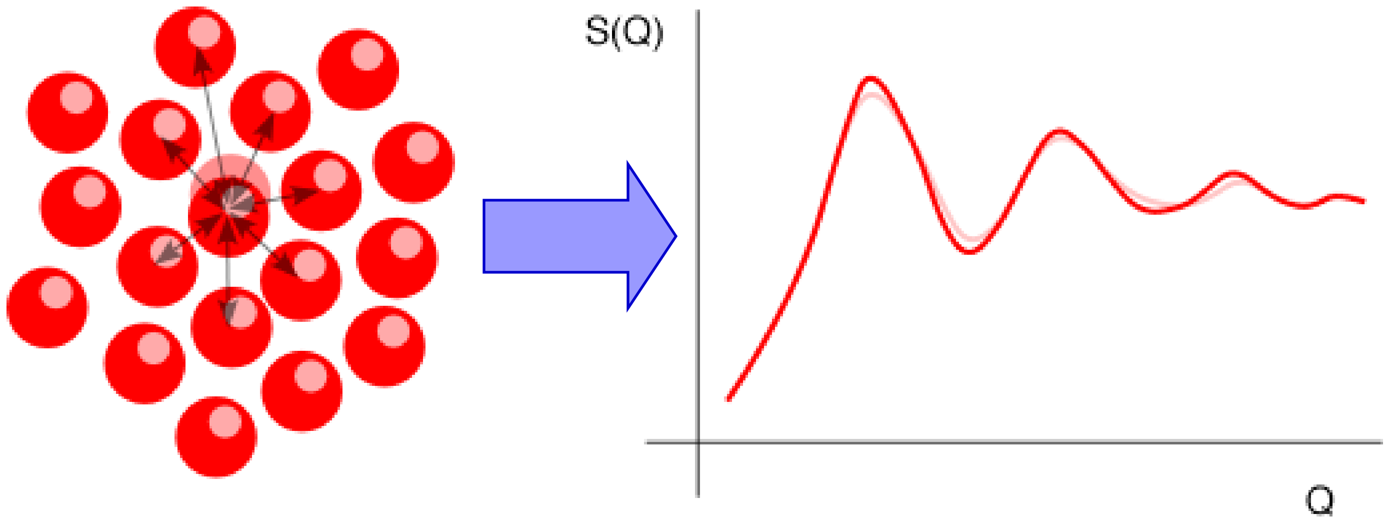
Reverse Monte Carlo

- ☐ Create “reasonable” starting configuration
- ☐ evaluate S(Q)



$$S(Q) = \frac{1}{N}\sum_{i,j} \bar{b}_i \bar{b}_j \frac{\sin\left(Q\left(\mathbf{r}_j - \mathbf{r}_i\right)\right)}{Q\left(\mathbf{r}_j - \mathbf{r}_i\right)}$$

Reverse Monte Carlo

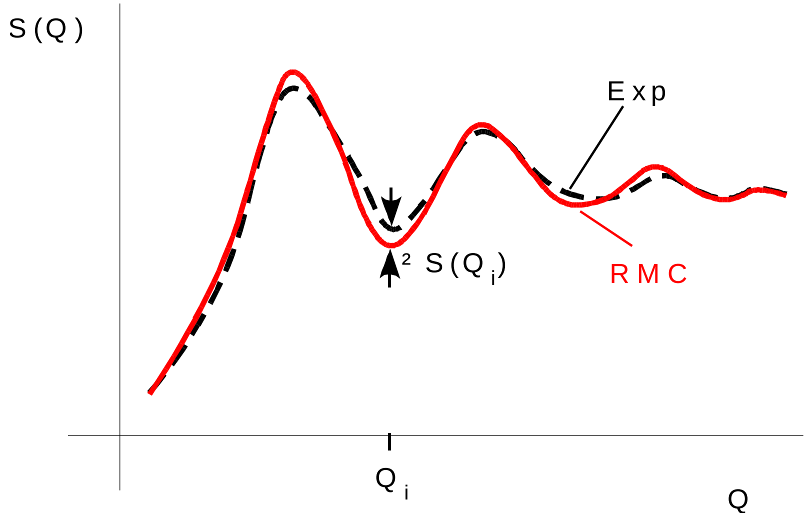


- ☐ Move one particle and re-evaluate S(Q)
- ☐ Only contribution of the one particle that moved needs to be considered

Reverse Monte Carlo

- ☐ Compare simulated S(Q) with experimental one
- ☐ Evaluate agreement factor χ^2

$$\chi^2_{S(Q)} = \sum_i \left[S_{\exp}(Q_i) - S_{\mathrm{sim}}(Q_i) \right]^2 / \sigma^2(Q_i)$$



Reverse Monte Carlo

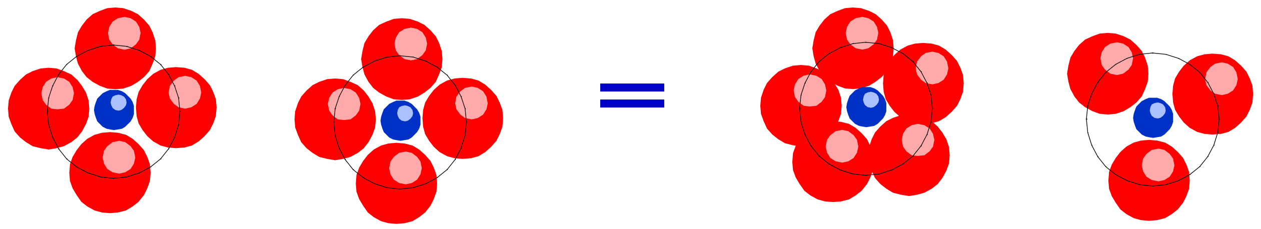
- Calculate $\Delta\chi^2 = \chi^2$ (after move) – χ^2 (before move)
- If $\Delta\chi^2 \leq 0$, accept move
- If $\Delta\chi^2 > 0$, compare to random number and move accept if

$$\mathrm{rand}\left[0,1\right] \leq \exp\left[- {}_2\ \chi^2_{S(Q)}/2\right]$$

- Otherwise reject move

Reverse Monte Carlo

- Method maximizes entropy, i.e. strives towards most disordered structure that produces given S(Q)



- Must use maximum range of experimental S(Q)
- Impose additional constraints (e.g., coordination number, bond angles, etc.)
- Combine with MD

Questions?

Overview …

- Historical perspective
- Simulation methodologies
  - Theoretical background
  - Monte Carlo simulation
  - Molecular dynamics simulation
  - Reverse Monte Carlo
- Force fields
- Information retrieval
  - Statistical mechanical formalisms
  - Structural analyses
  - Dynamics
- Application examples

Equations of motion …

Recall the Lagrangian $L = \frac{1}{2}\sum_{i=1}^{N} m_i \dot{\mathbf{r}}_i^T \dot{\mathbf{r}}_i - \sum_{i=1}^{N}\sum_{j>i}^{N} \phi(r_{ij})$

$$\frac{\partial}{\partial t}\left(\frac{\partial L}{\partial \dot{\mathbf{r}}_i}\right) = \frac{\partial L}{\partial \mathbf{r}_i}$$

$$\frac{\partial L}{\partial \dot{\mathbf{r}}_i} = m_i \dot{\mathbf{r}}_i = \mathbf{p}$$

$$\frac{\partial L}{\partial \mathbf{r}_i} = -\sum_{j \neq i} \frac{\partial \phi}{\partial r_{ij}} \frac{\mathbf{r}_{ij}}{\left|\mathbf{r}_{ij}\right|} = \mathbf{f}_i$$

$$\frac{\partial}{\partial t}\left(\frac{\partial L}{\partial \dot{\mathbf{r}}_i}\right) = \frac{\partial \mathbf{p}}{\partial t} = m_i \ddot{\mathbf{r}}_i$$

$$m_i \ddot{\mathbf{r}}_i = \mathbf{f}_i$$

Solving the equations of motion …

$$\begin{pmatrix} m_i \dfrac{\partial \mathbf{r}_i}{\partial t} \\ \dfrac{\partial \mathbf{p}_i}{\partial t} \end{pmatrix} = \begin{pmatrix} \mathbf{p}_i \\ -\nabla \phi(\mathbf{r}) \end{pmatrix}$$

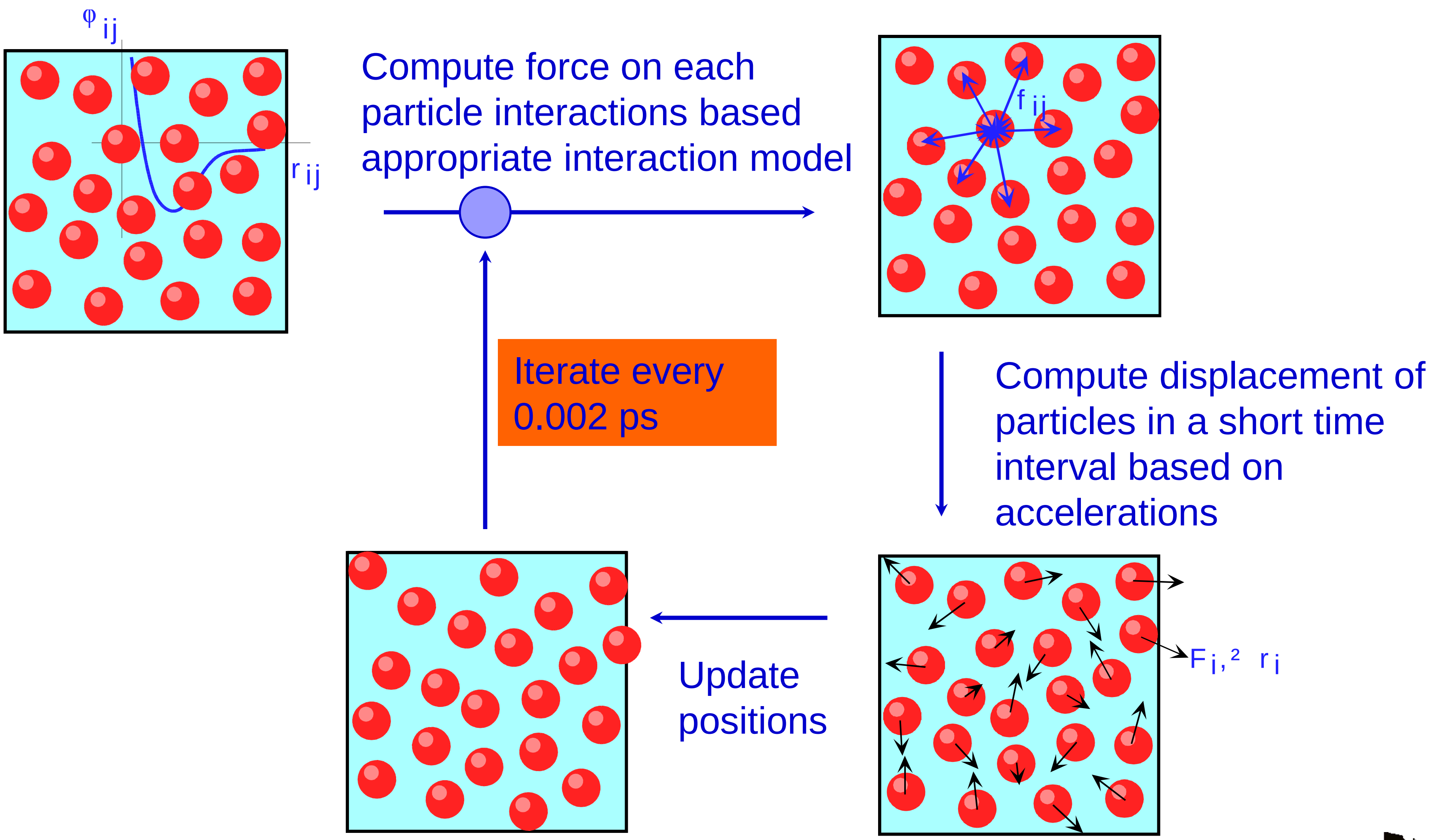
6N 1st-order equations

$$m_i \ddot{\mathbf{r}}_i \approx m_i \frac{1}{\Delta t}\left\{ \frac{\mathbf{r}(t+\Delta t)-\mathbf{r}(t)}{\Delta t} - \frac{\mathbf{r}(t)-\mathbf{r}(t-\Delta t)}{\Delta t} \right\}$$

$$= m_i \frac{\mathbf{r}(t+\Delta t)+\mathbf{r}(t-\Delta t)-2\mathbf{r}(t)}{\Delta^2 t} = -\nabla \phi(\mathbf{r})$$

3N 2nd-order equations

Molecular Dynamics Simulations



Compute force on each particle interactions based appropriate interaction model

Compute displacement of particles in a short time interval based on accelerations

Update positions

Iterate every 0.002 ps

Molecular Dynamics Simulation

- Basic MD simulation program:
  - Read in run parameters (e.g. initial energy or temperature; N)
  - Initialize positions and velocities
  - Compute forces on all particles
  - Integrate Newton's equations (F = ma)
  - Update particle positions and velocities
  - Calculate instantaneous properties
  - Stop after iterating t_{max} steps

$\left.\begin{array}{l} \\ \\ \\ \\ \\ \end{array}\right\}$ repeat

MD: Four basic parts

Initialization

- Setting up initial conditions.
- Boundary conditions.

Force Calculation

- Dealing with different kinds of forces.
- Tricks tracking particles.

Integration
Updating of x_i,v_i

- Different integration schemes.
- Different thermodynamic ensembles.

Recording of data
Calculation of properties

- What to calculate?
- Averaging data

MD Simulation: Integration Schemes

- There are many finite difference algorithms, and several are commonly used in MD simulations (many are related).
- Each has trade-offs:
  - accuracy
  - stability
  - time reversibility
  - area preserving
  - memory requirements
  - complexity
- Many research papers just on algorithm design for solving F = ma.

MD Simulation: Integration Schemes

- All algorithms start from assumption that the positions, velocities, and accelerations can be approximated by a Taylor series expansion:

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \mathbf{v}(t)\delta t + \frac{1}{2}\mathbf{a}(t)(\delta t)^2 + \frac{1}{6}\mathbf{b}(t)(\delta t)^3 + \frac{1}{24}\mathbf{c}(t)(\delta t)^4 + \ldots$$

$$\mathbf{v}(t+\delta t) = \mathbf{v}(t) + \mathbf{a}(t)\delta t + \frac{1}{2}\mathbf{b}(t)(\delta t)^2 + \frac{1}{6}\mathbf{c}(t)(\delta t)^3 + \ldots$$

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

$$\mathbf{b}(t+\delta t) = \mathbf{b}(t) + \mathbf{c}(t)\delta t + \ldots$$

where

$$\mathbf{v}_i = \dot{\mathbf{r}}_i$$

$$\mathbf{a}_i = \dot{\mathbf{v}}_i = \ddot{\mathbf{r}}_i$$

$$\mathbf{b}_i = \dot{\mathbf{a}}_i = \ddot{\mathbf{v}}_i = \dddot{\mathbf{r}}_i$$

etc.

MD Simulation: Euler

- The Euler algorithm uses the positions, velocities and accelerations at time t to calculate the new positions r(t+δt):

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

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

$$\mathbf{a}_i(t) = \frac{F_i(t)}{m_i}$$

- Simplest integration scheme.
- Bad: Not very accurate, so suffers from catastrophic energy drift. Not area preserving or time reversible.
- NOT RECOMMENDED; never used.
- Problem: uses only values at one time to estimate new values. All algorithms in use today "interpolate".

MD Simulation: Verlet

- The Verlet algorithm uses the positions and accelerations at time t, and the positions from the previous step r(t-δt), to calculate the new positions r(t+δt):

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

$$+$$

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

$$\mathbf{a}_i(t) = \frac{F_i(t)}{m_i}$$

$$\Rightarrow \quad \mathbf{r}_i(t+\delta t) = 2\mathbf{r}_i(t) - \mathbf{r}_i(t-\delta t) + \mathbf{a}_i(t)(\delta t)^2$$

- The velocities do not appear in the Verlet integration scheme: *velocities are not necessary for generating particle trajectories*.

MD Simulation: Verlet

- To obtain the new velocities, we can calculate them from the difference between the positions at two different times:

$$\mathbf{v}_i(t) = \left[\mathbf{r}_i(t+\delta t) - \mathbf{r}_i(t-\delta t)\right] / 2\delta t$$

or

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

MD Simulation: Verlet

- Advantages:
  - Implementation is straightforward.
  - Storage requirements are modest:
    - two sets of positions and one set of accelerations.
    - 9N stored numbers
- Disadvantages:
  - Positions are calculated by adding a small term of order δt^2 to the difference of two much larger terms, which may lead to a loss in precision:

$$\mathbf{r}_i(t+\delta t) = 2\mathbf{r}_i(t) - \mathbf{r}_i(t-\delta t) + \mathbf{a}_i(t)(\delta t)^2$$

  - Velocities always lag behind positions.
  - Poor stability for large δt.

MD Simulation: Leap-Frog

Hockney 1970

- Several variations on the Verlet algorithm have been developed, including the *leap-frog algorithm*:

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

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

$$\mathbf{a}_i(t) = \frac{F_i(t)}{m_i}$$

- The velocities at time t can be calculated from:

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

- The velocities leap-frog over the positions to give their values at t+δt/2. The positions then leapfrog over the velocities to give their new values at t+δt, then the velocities at t +3δt/2, and so on.

MD Simulation: Leap-Frog

- Advantages over standard Verlet:
  - Explicitly includes velocity (needed for kinetic energy).
  - Does not require calculation of differences of large numbers, so more accurate.
  - Same memory requirements (9N) as Verlet.

- Disadvantages over standard Verlet:
  - Positions and velocities are not synchronized.
    - Cannot calculate kinetic energy (from velocities) and potential energy (from positions) at the same time.
    - Not a problem - can always estimate $v(t)$ from $v(t+\delta t/2)$ and $v(t-\delta t/2)$ if you need it.

Almost always used over Verlet.

Swope, et al
1982

MD Simulation: Velocity Verlet

- The *velocity Verlet method* gives positions, velocities, and accelerations at the same time without compromising precision:

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

Looks like Euler from position eqn, but isn't!

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

- Implemented in three stages since calculating velocities requires accelerations at both t and t+δt.

Swope, et al 1982

MD Simulation: Velocity Verlet

- Actual implementation trick:
  1. Velocities at t+1/2δt are calculated using information at t.

$$\mathbf{v}_i(t+\frac{1}{2}\delta t) = \mathbf{v}_i(t) + \mathbf{a}_i(t)\frac{1}{2}\delta t$$

  2. With these terms in memory, calculate:

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

  3. Calculate $\mathbf{a}_i$(t+δt) from forces using positions at t+δt, and from this and $\mathbf{v}_i$(t+δt/2), calculate $\mathbf{v}_i$(t+δt).

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

MD Simulation: Other Integration Schemes

- The "order" of an integration method is the *degree to which the Taylor series expansion is truncated* -- it is the lowest term *not* present in the expansion. Sometimes this is not obvious.
  - Verlet is a 4th order method since 3rd order terms (not shown) cancel when expressions are added, which means positions are accurate to order δt^4. But, velocities only accurate to δt^2.
  - Velocity verlet also accurate to 4th order in positions, 2nd order in velocities.
  - Velocity-corrected Verlet is 4th order in positions and velocities.

- Predictor-corrector methods [Gear 1971] form a general family of integration algorithms from which one can select a scheme that is correct to any given order.

  - *But, not area preserving and not time-reversible*.

MD Simulation: Predictor-Corrector Methods

Three basic steps:

1. Values of r(t+δt), v(t+δt), a(t+δt) and higher order terms predicted from Taylor expansion using values at time t only:

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \mathbf{v}(t)\delta t + \frac{1}{2}\mathbf{a}(t)(\delta t)^2 + \frac{1}{6}\mathbf{b}(t)(\delta t)^3 + \frac{1}{24}\mathbf{c}(t)(\delta t)^4 + \ldots$$

$$\mathbf{v}(t+\delta t) = \mathbf{v}(t) + \mathbf{a}(t)\delta t + \frac{1}{2}\mathbf{b}(t)(\delta t)^2 + \frac{1}{6}\mathbf{c}(t)(\delta t)^3 + \ldots$$

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

$$\mathbf{b}(t+\delta t) = \mathbf{b}(t) + \mathbf{c}(t)\delta t + \ldots$$

MD Simulation: Predictor-Corrector Methods

Three basic steps:

2. Forces are calculated at r(t+δt) to give a second estimate for $a^F(t+\delta t)$. These values of acceleration are compared with those predicted from Taylor series expansion, a(t+δt).

3. The difference between the predicted and calculated accelerations is then used to correct the positions, velocities, etc. in the correction step:

$$\Delta \mathbf{a}(t+\delta t) = \mathbf{a}^F(t+\delta t) - \mathbf{a}(t+\delta t)$$

MD Simulation: Predictor-Corrector Methods

Corrected values:

$$\mathbf{r}^c(t+\delta t) = \mathbf{r}^T(t+\delta t) + c_0 \Delta\mathbf{a}(t+\delta t)$$

$$\mathbf{v}^c(t+\delta t) = \mathbf{v}^T(t+\delta t) + c_1 \Delta\mathbf{a}(t+\delta t)$$

$$\mathbf{a}^c(t+\delta t)/2 = \mathbf{a}^T(t+\delta t)/2 + c_2 \Delta\mathbf{a}(t+\delta t)$$

$$\mathbf{b}^c(t+\delta t)/6 = \mathbf{b}^T(t+\delta t)/6 + c_3 \Delta\mathbf{a}(t+\delta t)$$

- Gear has suggested “best” values of the coefficients c_0, c_1, … . Which set of “magic numbers” to use depends on the desired order of the algorithm. Here the expansion has been truncated after the third derivative of the positions (4th order), and one uses $c_0 = 1/6$, $c_1 = 5/6$, $c_2 = 1$, and $c_3 = 1/3$.

MD Simulation: Comparing Integration Schemes

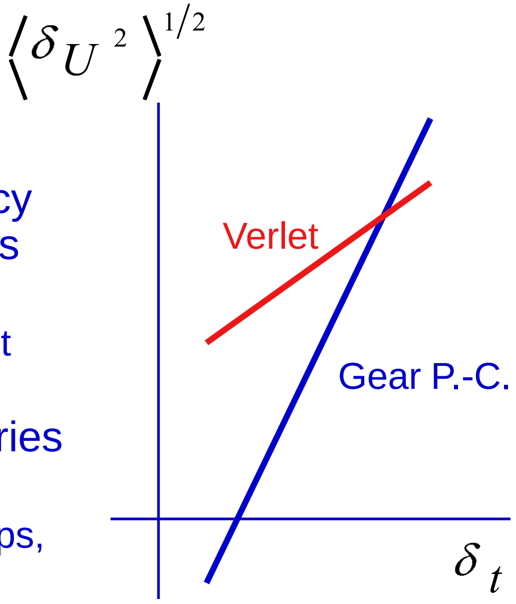
- Compare Predictor-Corrector vs. Verlet:
  - Memory req'd for Gear is (O+1)x3N, where O = order of highest derivative in Taylor series expansion, 1 is for extra acceleration, 3N = # particle coordinates. Present example: 15N.
  - Memory req'd for Verlet is 3 terms x 3N coords = 9N.
  - Gear requires *two* force evaluations per time step, but may permit a δt that is more than twice as long.

MD Simulation: Comparing Integration Schemes

- Which algorithm is "best"?
  - Fast
  - Require minimal memory
  - Easy to program

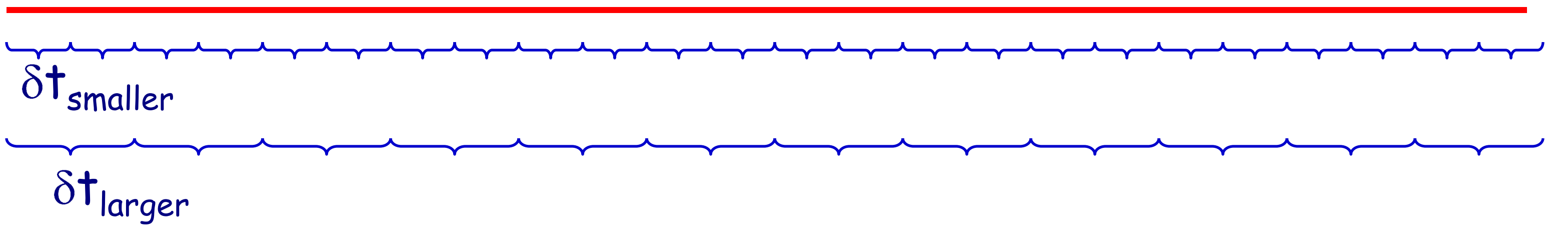
  Most meet this requirement on today's computers.

  - Time reversible - not always needed.
  - Area preserving - not always needed.
  - Accurate: *conserve energy and momentum*
    - NVE - total energy is constant, except for accuracy of algorithm (and machine precision - only matters for forces).
      - A variation of one part in 10^4 is acceptable for most applications.
    - Different algorithms vary in rate at which error varies with time step
      - E.g. Predictor-corrector accurate for short time steps, Verlet more accurate for larger time steps.
  - Stable for large δt



MD Simulation: Choosing the Time Step

- Want as large a δt as needed to generate a trajectory over time scales sufficient for problem.



- Time step δt can't be too large or integration algorithm will be inaccurate and could be unstable.
  - ❖ If step too large, force will change too much: this causes inaccuracy.
  - ❖ If forces become too large (i.e. particles move too close together in a single time step, this causes instability.)

MD Simulation: Choosing the Time Step

- Time step δt must be *smaller than the smallest time* relevant to oscillatory motion.
  - ❖ Atomic systems: $\delta t \ll t_{\text{oscillation}}$ (typically $\delta t = 2$ fs)
  - ❖ Molecular systems: $\delta t < 0.1 t_{\text{shortest osc. period}}$
    - Usually the *highest frequency vibrations are due to bond stretches*, especially due to bonds to H atoms, and $t_{\text{shortest}} = t_{\text{bond stretch}}$. ($t_{\text{C-H}} \approx 10$ fs so $\delta t = 1$ fs)

Questions?