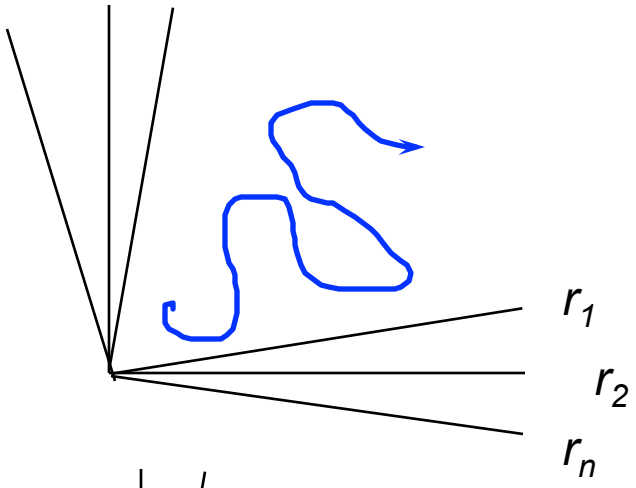
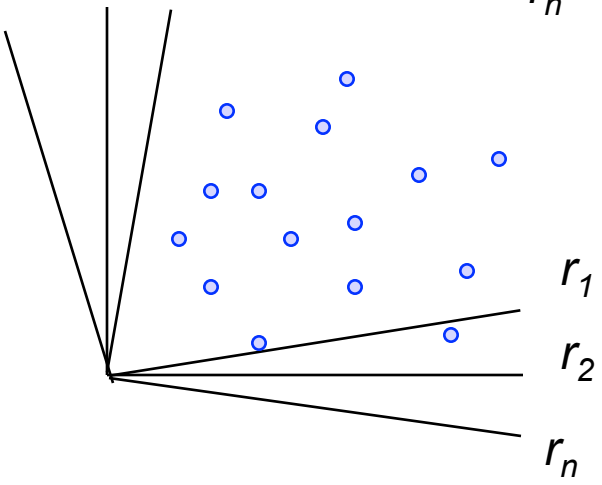
Molecular simulation of complex systems

- Molecular dynamics: solve equations of motion



MD

- Monte Carlo: importance sampling



MC

Molecular Dynamics

Basics (4.1, 4.2, 4.3)
Liouville formulation (4.3.3)
Multiple timesteps (15.3)

Molecular dynamics

Is based on Newton's equations.

$$F_i = m_i a_i = m_i \frac{d^2 x_i(t)}{dt^2}$$

for i=1 .. N particles

the force F is given by the gradient of the potential

$$F_i = -\frac{\partial V(r^N)}{\partial r_i}$$

given the potential, one can integrate the trajectory ***x(t)*** of the whole system as a function of time.

Numerical integration

This is an N-body problem, which can only be solved numerically (except in very special cases)

$$x(t+\Delta t) = x(t) + \dot{x}(t)\Delta t + \frac{1}{2}\ddot{x}(t)\Delta t^2 + \frac{1}{6}\dddot{x}(t)\Delta t^3 + \ldots$$

at least, in principle..

Naïve implementation: truncation of Taylor expansion

$$x(t+\Delta t) = x(t) + \dot{x}(t)\Delta t + \frac{1}{2}\ddot{x}(t)\Delta t^2$$

Wrong!

The naive "forward Euler" algorithm

- is not time reversible
- does not conserve volume in phase space
- suffers from energy drift

Better approach: "Verlet" algorithm

Verlet algorithm

compute position in next and previous time steps

$$x(t+\Delta t) = x(t) + \dot{x}(t)\Delta t + \frac{1}{2}\ddot{x}(t)\Delta t^2 + \frac{1}{6}\dddot{x}(t)\Delta t^3 + \frac{1}{24}\ddddot{x}(t)\Delta t^4 ...$$

$$x(t-\Delta t) = x(t) - \dot{x}(t)\Delta t + \frac{1}{2}\ddot{x}(t)\Delta t^2 - \frac{1}{6}\dddot{x}(t)\Delta t^3 + \frac{1}{24}\ddddot{x}(t)\Delta t^4 ...$$

+

$$x(t+\Delta t) + x(t-\Delta t) = 2x(t) + \ddot{x}(t)\Delta t^2 + \mathcal{O}(\Delta t^4) ...$$

or

$$x(t+\Delta t) = 2x(t) - x(t-\Delta t) + \ddot{x}(t)\Delta t^2$$

Verlet

Verlet algorithm

- is time reversible
- does conserve volume in phase space
- (is “symplectic”)
- does not suffer from energy drift

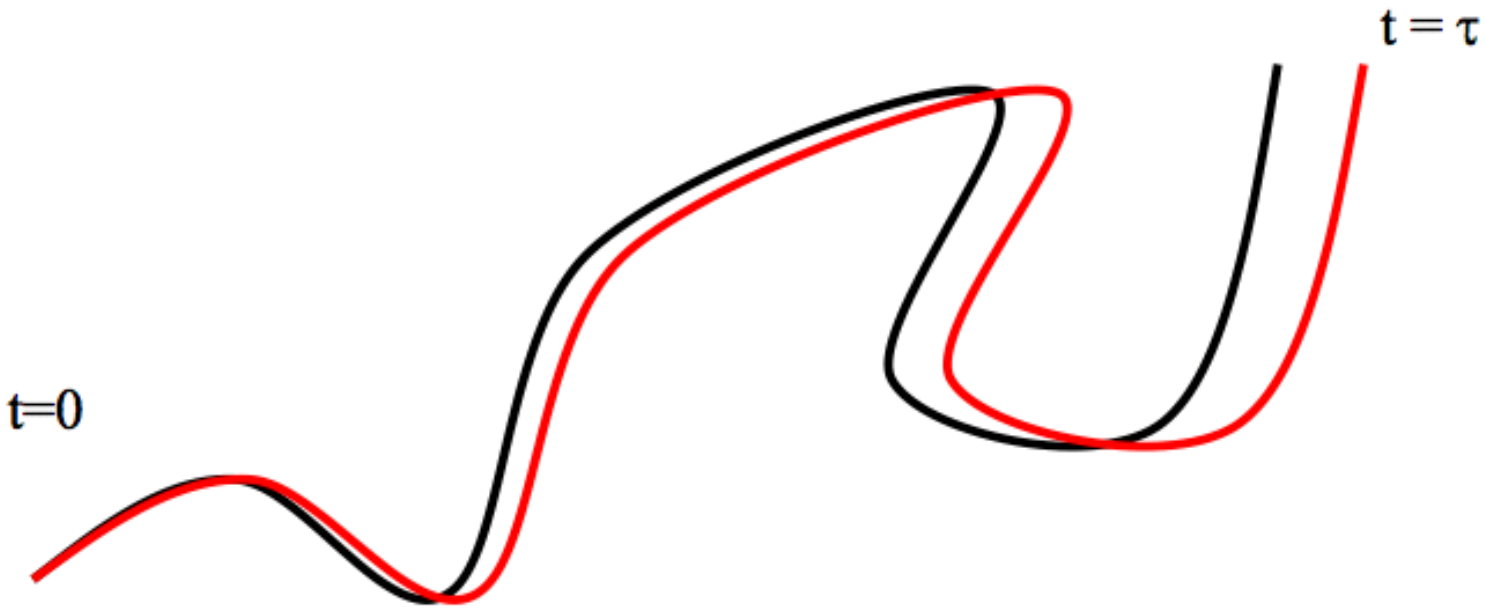
...but is it a good algorithm?

i.e. does it predict the time evolution of the system correctly???

Molecular chaos

Dynamics of “well-behaved” classical many-body system is chaotic.

Consequence: Trajectories that differ very slightly in their initial conditions diverge exponentially (“Lyapunov instability”)



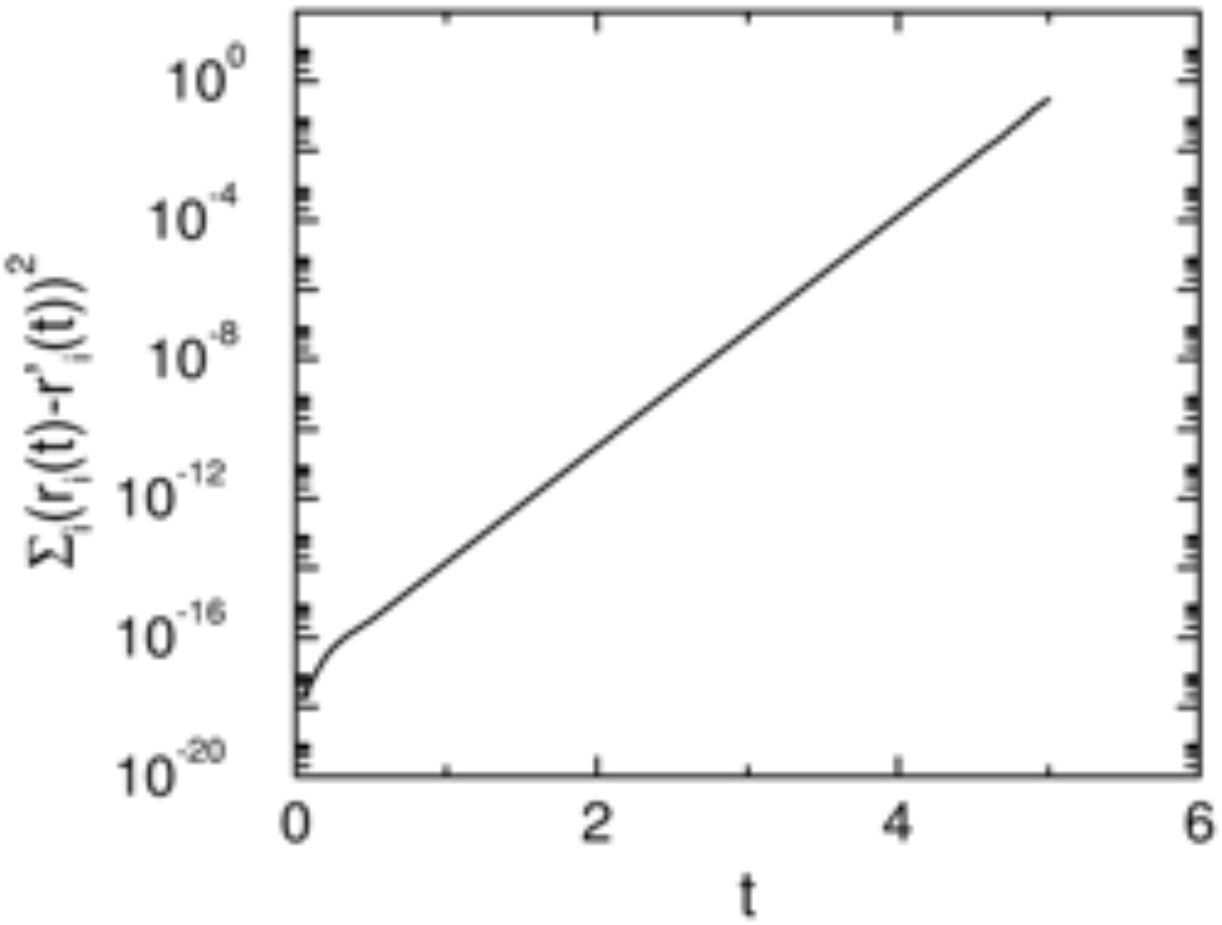
Lyapunov instability

The Lyapunov disaster in action...

$$\left(\mathbf{r}^N(0), \mathbf{p}^N(0)\right)$$

$$\left(\mathbf{r}^N(0), \mathbf{p}_1(0), \cdots, \mathbf{p}_j(0) + \varepsilon, \mathbf{p}_i(0) - \varepsilon \cdots, \mathbf{p}_N(0)\right)$$

$$\varepsilon = 10^{-10}$$



Any small error in the numerical integration of the equations of motion, will blow up exponentially....

always...

...and for any algorithm!!

SO:

Why should anyone believe Molecular Dynamics simulations ???

Answers:

1. In fact, one should not…

2. Good MD algorithms (e.g. Verlet) can also be considered as good Monte Carlo algorithms –they therefore yield reliable STATIC properties (“Hybrid Monte Carlo”)

3. What is the point of simulating dynamics, if we cannot trust the resulting time-evolution???

4. All is well (probably), because of...

   The Shadow Theorem.

Shadow theorem

- For any realistic many-body system, the shadow theorem is merely a hypothesis.
- It basically states that Good algorithms generate numerical trajectories that are “close to” a REAL trajectory of the many-body system.
- Question: Does the Verlet algorithm indeed generate “shadow” trajectories?
- Take a different look at the problem.
  - Do not discretize NEWTON’s equation of motion…
  - ...but discretize the ACTION

Lagrangian Classical mechanics

- Newton: $$F_i = m_i \frac{d^2 x_i(t)}{dt^2}$$

- Lagrange:
  - Consider a system that is at a point r_0 at time t=0 and at point r_t at time t=t, then the system follows a trajectory r(t) such that:

$$S \equiv \int_0^t dt' \mathcal{L}(r(t'))$$

is an extremum. The Lagrangianis L defined as:

$$\mathcal{L}(r(t)) = K - U(r)$$

kinetic energy

Langrangian

For example, if we use cartesian coordinates:

$$\mathcal{L}(r(t)) = \sum_{i=1}^{N} \frac{1}{2} m_i \dot{r}_i^2 - U(r_1, r_2, \ldots r_N)$$

What does this mean?

Consider the "true" path R(t), with R(0)=r_0 and R(t)=r_t.
Now, consider a path close to the true path:

$$r(t') = R(t') + \delta r(t')$$

Then the action S is an extremum if

$$\frac{\partial S}{\partial r(t')} = 0 \ \text{ for all } t$$

what does this mean?

Discretized action

$$S_{cont} = \int_{t_0}^{t_1} dt \mathcal{L}(t)$$

$$S_{disc} = \Delta t \sum_{i=0}^{i_{max}} \mathcal{L}(t_i) \qquad \mathcal{L}(t_i) = K(t_i) - U(t_i)$$

For a one dimensional system this becomes

$$\mathcal{L}(t_i)\Delta t = \frac{1}{2} m \Delta t \frac{(x_{i+1} - x_i)^2}{\Delta t^2} - U(x_i)\Delta t$$

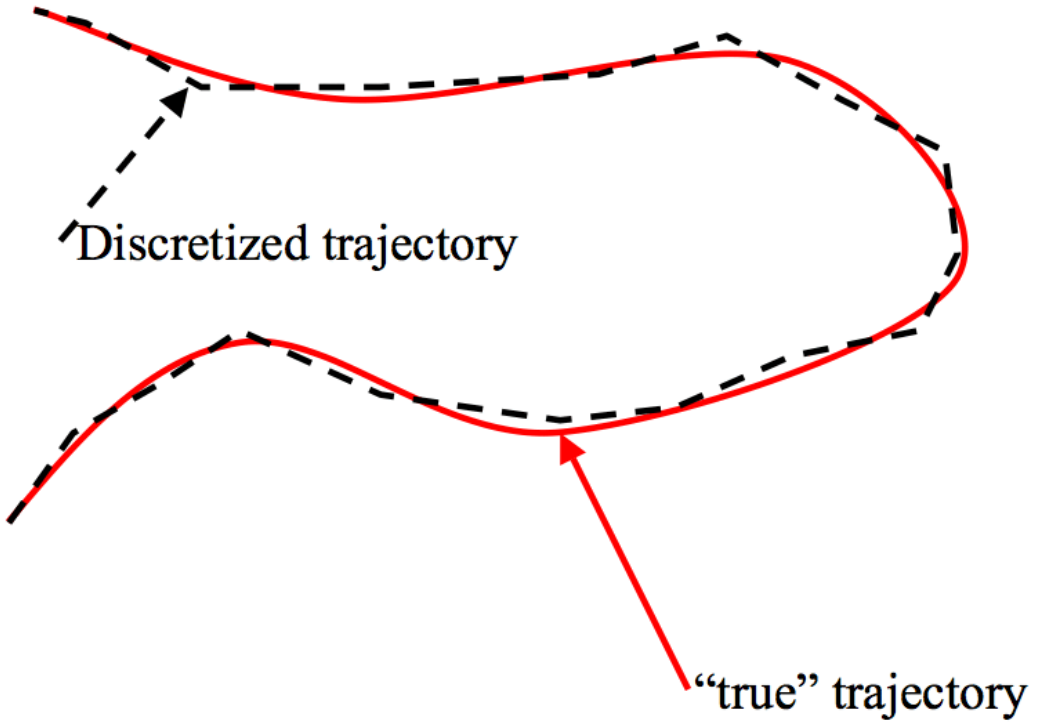
$$S_{disc} = \sum_{i=1}^{i_{max}} \left[\frac{m(x_{i+1} - x_i)^2}{2\Delta t} - U(x_i)\Delta t \right]$$

Minimize the action

Now do the standard thing: Find the extremum for small variations in the path, i.e. for small variations in all x_i.

$$\frac{\partial S_{disc}}{\partial x_i} = 0 \ \text{ for all } i$$

This will generate a discretized trajectory that starts at time t_0 at X, and ends at time t at X_t.



Minimizing the action

$$\frac{\partial S_{disc}}{\partial x_i} = \frac{\partial}{\partial x_i} \sum_{i=1}^{i_{max}} \left[\frac{m(x_{i+1} - x_i)^2}{2\Delta t} - U(x_i)\Delta t \right]$$

$$\frac{\partial S_{disc}}{\partial x_i} = \frac{-m(x_{i+1} - x_i) + m(x_i - x_{i-1})}{\Delta t} - \Delta t \frac{\partial U(x_i)}{\partial x_i}$$

$$0 = \frac{m}{\Delta t} \left(2x_i - x_{i+1} - x_{i-1} - \frac{\Delta t^2}{m} \frac{\partial U(x_i)}{\partial x_i} \right)$$

$$0 = 2x_i - x_{i+1} - x_{i-1} - \frac{\Delta t^2}{m}\frac{\partial U(x_i)}{\partial x_i}$$

$$x_{i+1} = 2x_i - x_{i-1} + \frac{\Delta t^2}{m}F(x_i)$$

- which is the Verlet algorithm!
- The Verlet algorithm generates a trajectory that satisfies the boundary conditions of a REAL trajectory –both at the beginning and at the endpoint.
- Hence, if we are interested in statistical information about the dynamics (e.g. time-correlation functions, transport coefficients, power spectra...) ...then a “good” MD algorithm (e.g. Verlet) is fine.

Practical MD

Algorithm 3 (A Simple Molecular Dynamics Program)

```
program md                          simple MD program

call init                           initialization
t=0
do while (t.lt.tmax)                MD loop
   call force(f,en)                 determine the forces
   call integrate(f,en)             integrate equations of motion
   t=t+delt
   call sample                      sample averages
enddo
stop
end
```

Comment to this algorithm:

1. *Subroutines* `init`, `force`, `integrate`, *and* `sample` *will be described in Algorithms 4, 5, and 6, respectively. Subroutine* `sample` *is used to calculate averages like pressure or temperature.*

Algorithm 4 (Initialization of a Molecular Dynamics Program)

```
subroutine init              initialization of MD program
sumv=0
sumv2=0
do i=1,npart
  x(i)=lattice_pos(i)        place the particles on a lattice
  v(i)=(ranf()-0.5)          give random velocities
  sumv=sumv+v(i)             velocity center of mass
  sumv2=sumv2+v(i)**2        kinetic energy
enddo
sumv=sumv/npart              velocity center of mass
sumv2=sumv2/npart            mean-squared velocity
fs=sqrt(3*temp/sumv2)        scale factor of the velocities
do i=1,npart                 set desired kinetic energy and set
  v(i)=(v(i)-sumv)*fs        velocity center of mass to zero
  xm(i)=x(i)-v(i)*dt         position previous time step
enddo
return
end
```

Algorithm 5 (Calculation of the Forces)

```
subroutine force(f,en)                    determine the force
en=0                                      and energy
do i=1,npart
   f(i)=0                                 set forces to zero
enddo
do i=1,npart-1
  do j=i+1,npart                          loop over all pairs
     xx=x(i)-x(j)
     xx=xx-box*nint(xx/box)               periodic boundary conditions
     r2=xx**2
     if (r2.lt.rc2) then                  test cutoff
       r2i=1/r2
       r6i=r2i**3
       ff=48*r2i*r6i*(r6i-0.5)            Lennard-Jones potential
       f(i)=f(i)+ff*xx                    update force
       f(j)=f(j)-ff*xx
       en=en+4*r6i*(r6i-1)-ecut           update energy
     endif
  enddo
enddo
return
end
```

Algorithm 6 (Integrating the Equations of Motion)

```
subroutine integrate(f,en)          integrate equations of motion
sumv=0
sumv2=0
do i=1,npart                        MD loop
   xx=2*x(i)-xn(i)+delt**2*f(i)     Verlet algorithm (4.2.3)
   vi=(xx-xn(i))/(2*delt)           velocity (4.2.4)
   sumv=sumv+vi                     velocity center of mass
   sumv2=sumv2+vi**2                total kinetic energy
   xn(i)=x(i)                       update positions previous time
   x(i)=xx                          update positions current time
enddo
temp=sumv2/(3*npart)                instantaneous temperature
etot=(en+0.5*sumv2)/npart           total energy per particle
return
end
```

Lennard Jones potentials

- The Lennard-Jones potential

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

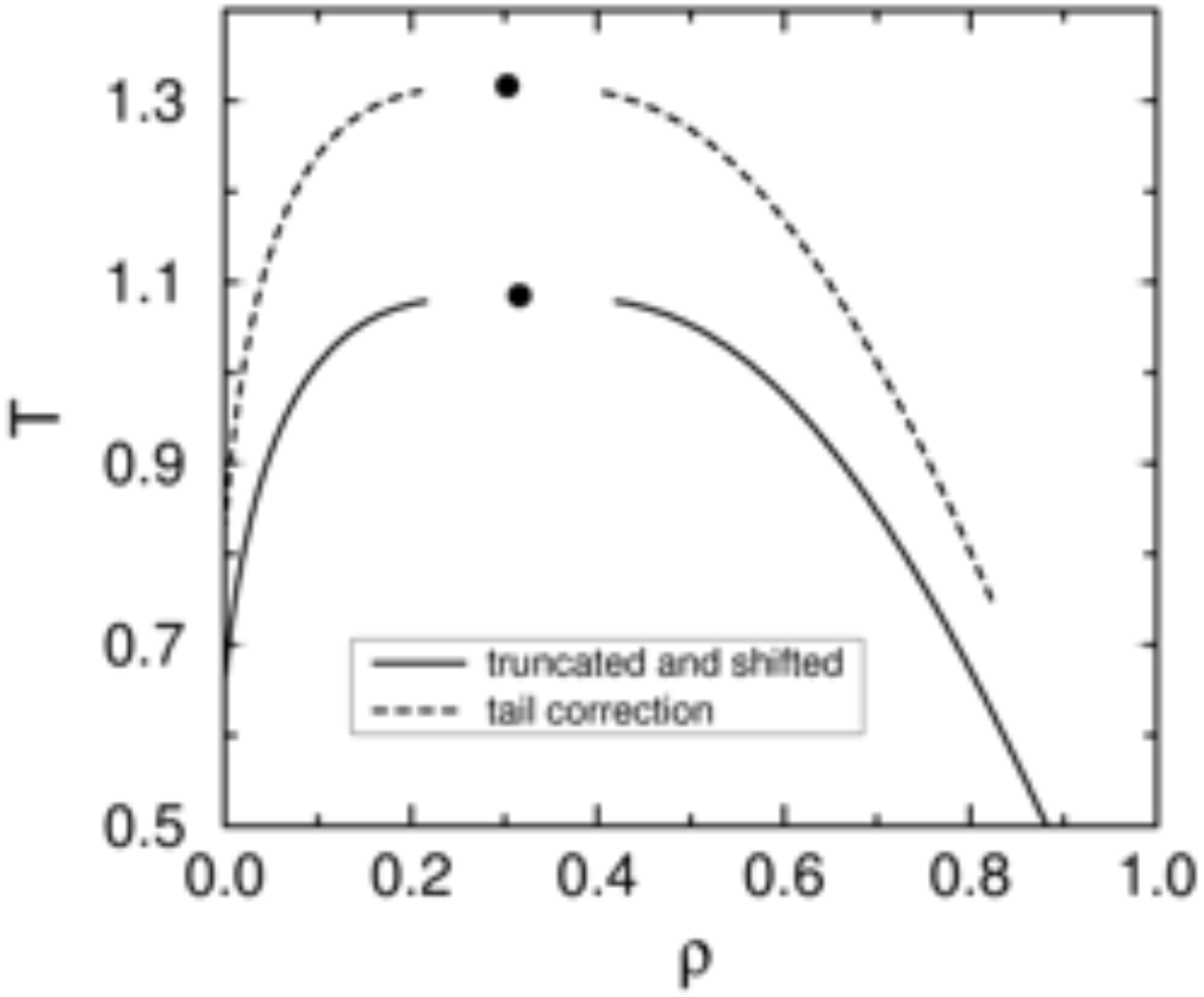
- The truncated Lennard-Jones potential

$$u(r) = \begin{cases} u^{LJ}(r) & r \le r_c \\ 0 & r > r_c \end{cases}$$

- The truncated and shifted Lennard-Jones potential

$$u(r) = \begin{cases} u^{LJ}(r) - u^{LJ}(r_c) & r \le r_c \\ 0 & r > r_c \end{cases}$$

Phase diagrams of Lennard Jones fluids



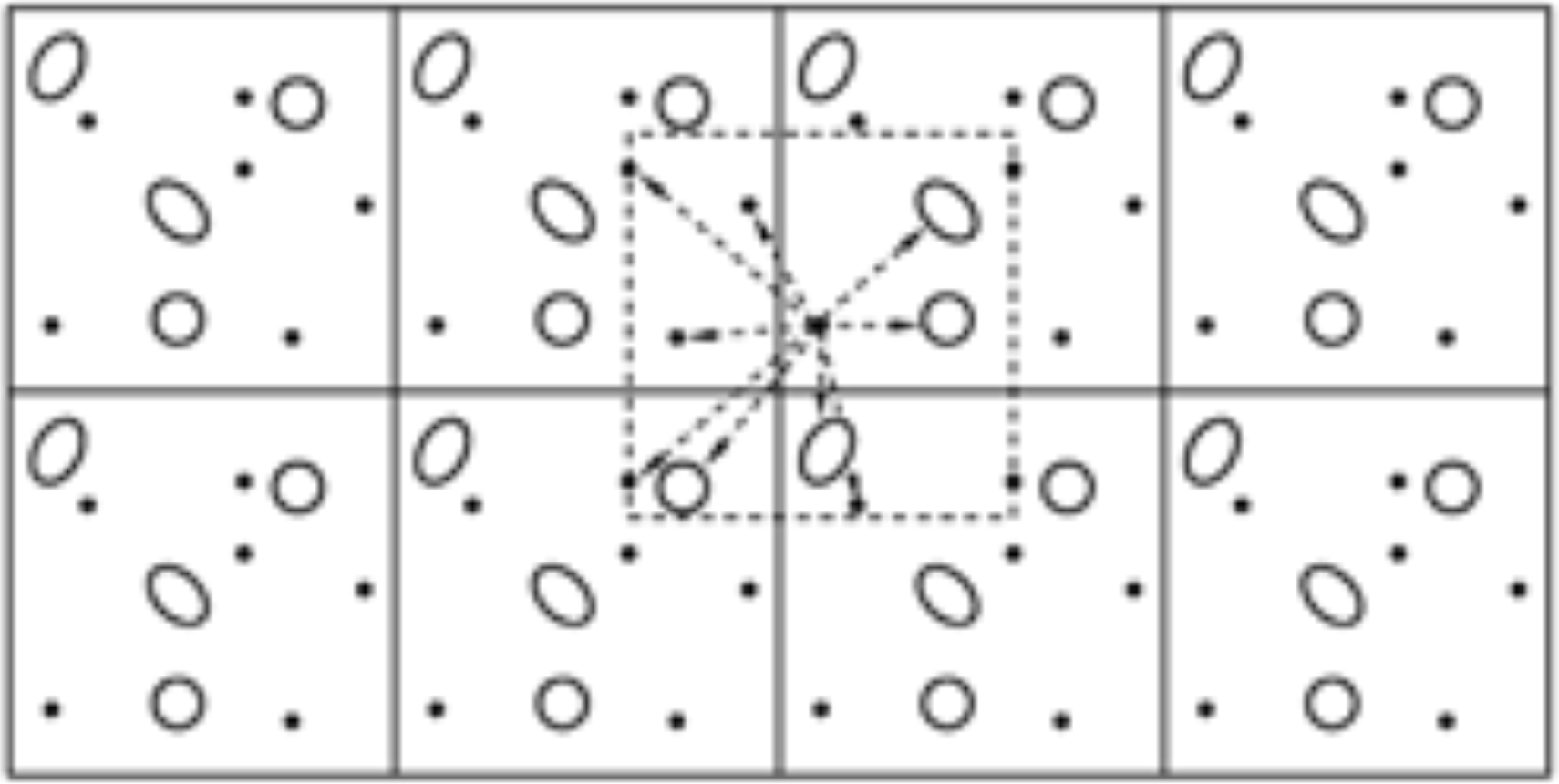
Issues related to MD

- Initialization
  - Total momentum should be zero (no external forces)
  - Temperature rescaling to desired temperature
  - Particles/atoms/molecules start on a lattice/ or random positions

- Force calculations
  - Periodic boundary conditions
  - Straightforward force: Order N^2 algorithm:
  - neighbor lists, linked cell: Order N
  - Truncation and shift of the potential
  - Electrostatics: Ewald summation $O(N^{1.5})$ or PME: O(NlogN)

- Integrating the equations of motion
  - Controlling the temperature by a Thermostat
  - Ergodicity theorem
  - Verlet or velocity Verlet
  - Multiple time steps

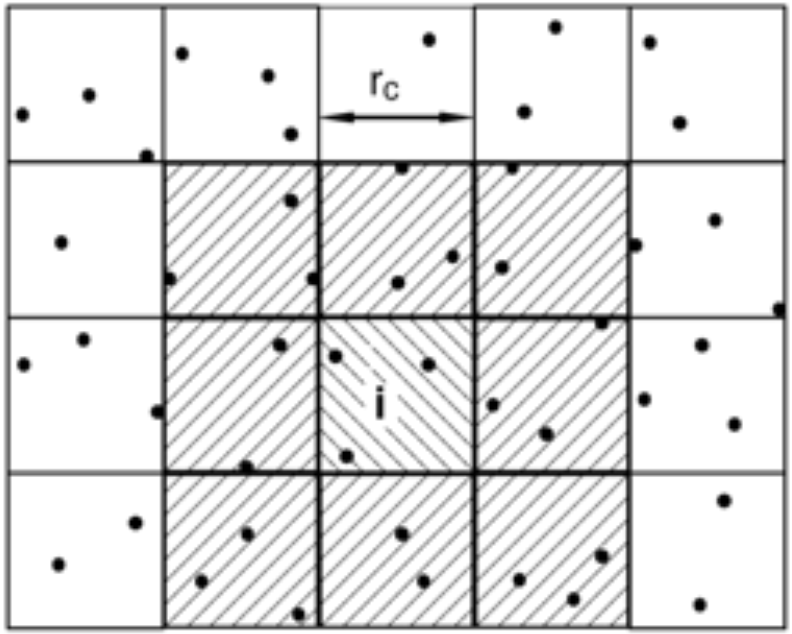
Periodic boundary conditions

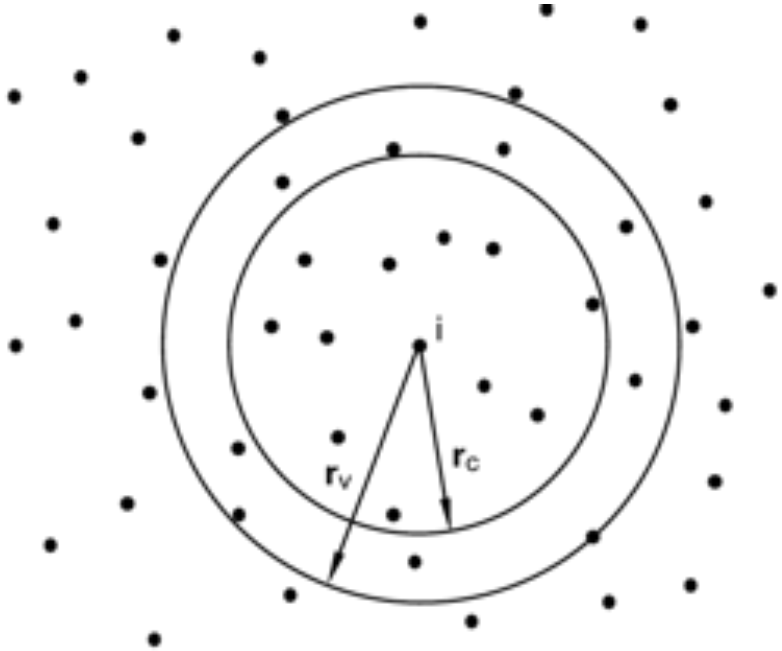


Saving CPU

- Cell list

Verlet List





NVE and NVT

- MD generates the NVE ensemble
- NVT by thermostats
  - Andersen
  - Nose Hoover
- Ergodicity

Lagrangian approach

Lagrangian is sum of two terms

$$\mathcal{L}(\dot{r}, r) = K(\dot{r}) - U(r) = \frac{m\dot{r}^2}{2} + U(r)$$

$$\frac{\partial \mathcal{L}}{\partial \dot{r}} = \frac{\partial K}{\partial \dot{r}} = p \qquad \frac{\partial \mathcal{L}}{\partial r} = -\frac{\partial U}{\partial r} = F$$

$$p = \frac{\partial \mathcal{L}(\dot{r}, r)}{\partial \dot{r}} \qquad \dot{p} = \frac{\partial \mathcal{L}(\dot{r}, r)}{\partial r}$$

Newton : F=ma

Hamiltonian approach

The Hamiltonian is defined as

$$H(p,r) = p\dot{r} - \mathcal{L}(\dot{r},r)$$

$$H(p^N, r^N) = U(r^N) + \sum_i \frac{p_i^2}{2m_i}$$

Hamilton's equations are then

$$\dot{r} = \frac{\partial \mathcal{H}(r,p)}{\partial p} = \frac{p}{m}$$

$$\dot{p} = -\frac{\partial \mathcal{H}(r,p)}{\partial r} = -\frac{\partial U(r^N)}{\partial r}$$

Integrating equations of motion (by Verlet) conserves the Hamiltonian

Conservation of Hamiltonian

$$dH(p,r) = \frac{\partial H}{\partial p} dp + \frac{\partial H}{\partial r} dr$$

$$\frac{\partial H}{\partial p} = \dot{r} \qquad \frac{\partial H}{\partial r} = -\dot{p}$$

$$\frac{dH(p,r)}{dt} = \frac{\partial H}{\partial p} \dot{p} + \frac{\partial H}{\partial r} \dot{r} = \dot{r}\dot{p} - \dot{p}\dot{r} = 0$$

So a solution to the Hamiltonians equation conserves the TOTAL energy

$$E = K + U$$

MD generates NVE ensemble

In general the MC phase space density is

$$\rho(x) = e^{-\beta \mathcal{H}(x)}/Z \qquad Z = \int e^{-\beta \mathcal{H}(x)} dx$$

with $x = \{p^N, r^N\}$

Integrating over momenta gives

$$Z = \frac{1}{N!\Lambda^{3N}} \int e^{-\beta \mathcal{U}(r)} dr$$

N! comes from indistinguishability of particles.

But MD conserves Hamiltonian H= E = constant (and constant total P).

$$\rho(x) = \delta[E - \mathcal{H}(x)]/g(E) \qquad g(E) = \int dx \delta[E - \mathcal{H}(x)]$$

with instantaneous temperature $k_B T = \sum_{i=1}^{N} \frac{m v_i^2}{N_f}$

Thermostat: From NVE to NVT

Introduce thermostat in MD trajectory:

- stochastic thermostats
  - **Andersen**
  - Langevin
  - Bussi (2007)

- deterministic thermostat
  - **Nose-Hoover**

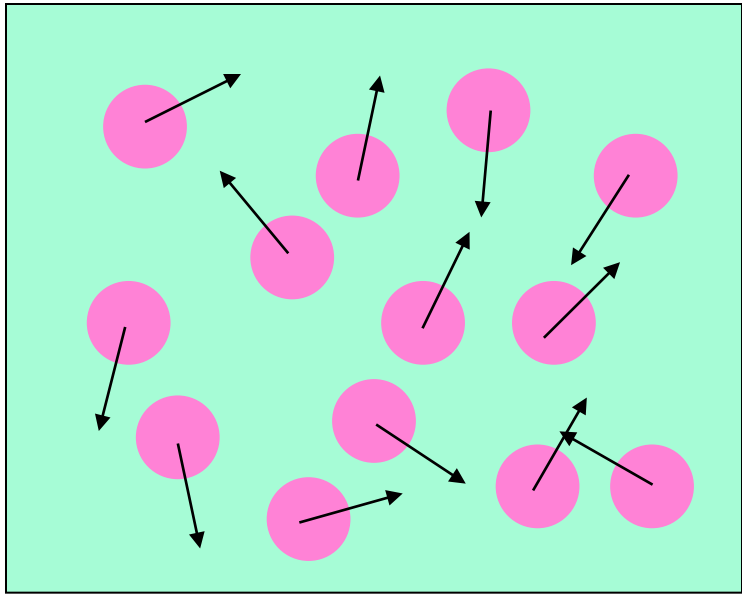
All of these alter the velocities such that the trajectory samples the canonical NVT ensemble, and the partition function becomes

$$Z = \frac{1}{N!\Lambda^{3N}} \int e^{-\beta \mathcal{U}(r)} dr$$

These thermostats differ in how they achieve this

Andersen Thermostat

- Every particle has a fixed probability to collide with the Andersen demon



- After collision the particle is give a new velocity

$$P(v) = \left(\frac{\beta}{2\pi m}\right)^{3/2} \exp\left[-\beta m v^2/2\right]$$

- The probabilities to collide are uncorrelated (Poisson distribution)

$$P(t;\nu) = \nu \exp\left[-\nu t\right]$$

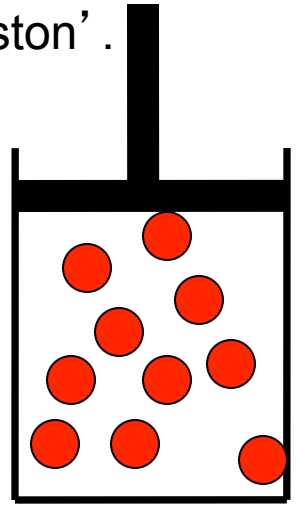
Nose Hoover thermostat

goal: compute MD trajectory sampling NVT ensemble.

Take kinetic energy out of the system and put it back in via a 'piston'.

piston can be seen as additional variable *s* storing kinetic energy

Approach: extended Lagrangian



extended variable

$$\mathcal{L}_{\text{Nose}} = \sum_{i=1}^{N} \frac{1}{2} m s^2 \dot{r}_i^2 - U\left(r^N\right) + \frac{1}{2} Q \dot{s}^2 - \frac{g}{\beta} \ln s$$

effective mass

constant to be set

$$p_i = \frac{\partial \mathcal{L}}{\partial \dot{r}_i} = m s^2 \dot{r}_i \qquad p_s = \frac{\partial \mathcal{L}}{\partial \dot{s}} = Q \dot{s}$$

$$\mathcal{H}_{\text{Nose}} = \sum_{i=1}^{N} \frac{p_i^2}{2 m s^2} + \frac{p_s}{2Q} + U\left(r^N\right) + \frac{g}{\beta} \ln s$$

Nose-Hoover Thermostat

$$\mathcal{H}_{\text{Nose}} = \sum_{i=1}^{N} \frac{p_i^2}{2ms^2} + \frac{p_s}{2Q} + U\left(r^N\right) + \frac{g}{\beta} \ln s$$

now define

$$\mathcal{H}(p', r) = \sum_{i=1}^{N} \frac{p_i'^2}{2m_i} + U(r^N) \qquad p' = p/s$$

then it is possible to show that the partition function Z_{nose} is

$$Z_{nose} \propto \frac{1}{N!} \int dp'^N dr^N \exp\left[-\beta \frac{3N+1}{g} \mathcal{H}(p', r)\right]$$

for g=3N+1 the system samples the canonical distribution if p' is interpreted as the real momentum

Nose-Hoover Thermostat

$$\mathcal{H}_{\text{Nose}} = \sum_{i=1}^{N} \frac{p_i^2}{2ms^2} + \frac{p_s}{2Q} + U\left(r^N\right) + \frac{g}{\beta} \ln s$$

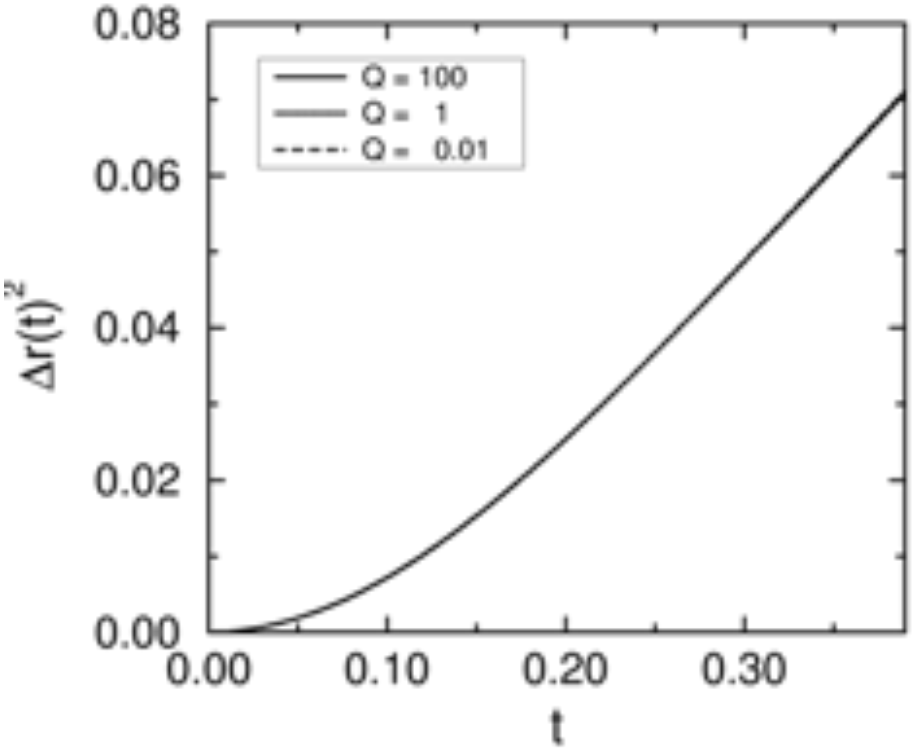
equations of motion follow from Hamilton's equations.

$$\frac{\mathrm{d}r_i}{\mathrm{d}t} = \frac{\partial \mathcal{H}_{\text{Nose}}}{\partial p_i} = \frac{p_i}{ms^2} \qquad \frac{\mathrm{d}p_i}{\mathrm{d}t} = -\frac{\partial \mathcal{H}_{\text{Nose}}}{\partial r_i} = -\frac{\partial U\left(r^N\right)}{\partial r_i}$$

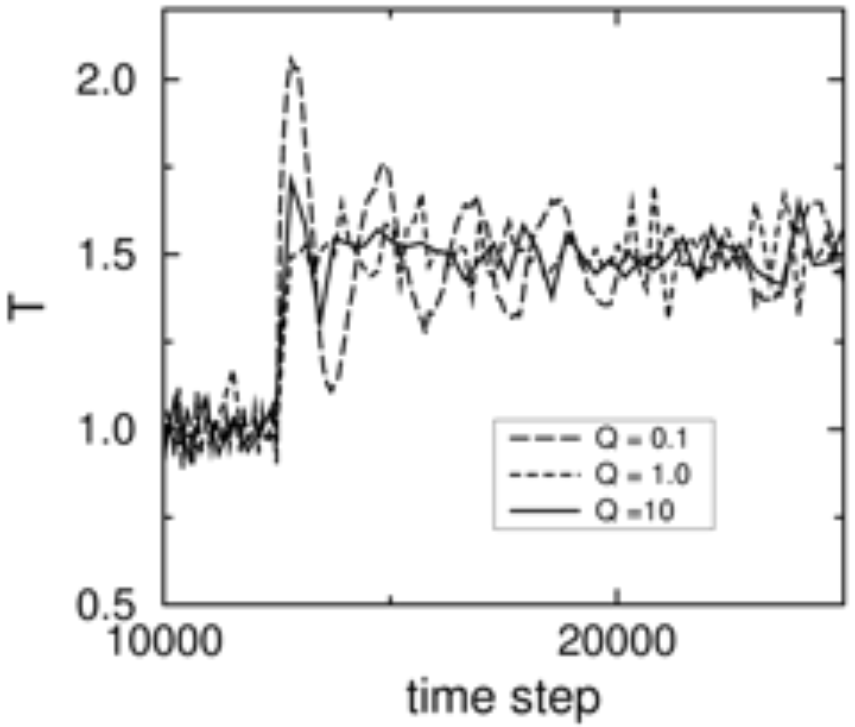
$$\frac{\mathrm{d}s}{\mathrm{d}t} = \frac{\partial \mathcal{H}_{\text{Nose}}}{\partial p_s} = \frac{p_s}{Q} \qquad \frac{\mathrm{d}p_s}{\mathrm{d}t} = -\frac{\partial \mathcal{H}_{\text{Nose}}}{\partial s} = \frac{1}{s}\left(\sum \frac{p_i^2}{ms^2} - \frac{g}{\beta}\right)$$

Effect of mass Q

Lennard-Jones fluid



mean square displacement



temperature relaxation

Ergodicity theorem

time averages over a NVT MD trajectory

$$\bar{A} = \frac{1}{\mathcal{T}} \int_0^{\mathcal{T}} A(t) dt$$

ensemble average

$$\langle A \rangle = \frac{\int dr^N A(r^N) \exp(-\beta \mathcal{U}(r^N))}{\int dr^N \exp(-\beta \mathcal{U}(r^N))}$$

Ergodicity theorem states that for an ‘ergodic system’

$$\bar{A} = \langle A \rangle$$

Verlet vs Velocity Verlet

Verlet algorithm

$$\mathbf{r}(t+\Delta t) \approx 2\mathbf{r}(t) - \mathbf{r}(t-\Delta t) + \frac{\Delta t^2}{m}\mathbf{f}(t)$$

Downside regular verlet algorithm: velocity is not known.

Velocity verlet (Andersen 1983):

$$\mathbf{r}(t+\Delta t) \approx \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{\Delta t^2}{2m}\mathbf{f}(t)$$

$$\mathbf{v}(t+\Delta t) \approx \mathbf{v}(t) + \frac{\Delta t}{2m}\left[\mathbf{f}(t+\Delta t) + \mathbf{f}(t)\right]$$

(Is based on Trotter decomposition of Liouville operator formulation, also basis of Multiple time steps).

Liouville formulation

$$f\left(\mathbf{p}^N, \mathbf{r}^N\right)$$

Depends implicitly on t

$$\dot{f} = \dot{\mathbf{r}}\frac{\partial f}{\partial \mathbf{r}} + \dot{\mathbf{p}}\frac{\partial f}{\partial \mathbf{p}}$$

$$\mathrm{i}L \equiv \dot{\mathbf{r}}\frac{\partial}{\partial \mathbf{r}} + \dot{\mathbf{p}}\frac{\partial}{\partial \mathbf{p}}$$

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \mathrm{i}Lf$$

Beware: this solution is equally useless as the differential equation!

Solution

$$f\left(t\right) = \exp\left(\mathrm{i}Lt\right) f\left(0\right)$$

$$\mathrm{i}L \equiv \mathrm{i}L_r + \mathrm{i}L_p = \dot{\mathbf{r}}\frac{\partial}{\partial \mathbf{r}} + \dot{\mathbf{p}}\frac{\partial}{\partial \mathbf{p}}$$

Let us look at them separately

$$f(t) = \exp\left(\mathrm{i}L_r t\right) f(0)$$

$$= \exp\left(\dot{\mathbf{r}}(0)t\frac{\partial}{\partial \mathbf{r}}\right) f(0)$$

$$= \exp\left(\dot{\mathbf{p}}(0)t\frac{\partial}{\partial \mathbf{p}}\right) f(0)$$

Expand exponential

$$= \sum_{n=0}^{\infty} \frac{\left(\dot{\mathbf{r}}(0)t\right)^n}{n!}\frac{\partial^n}{\partial \mathbf{r}^n} f(0)$$

$$= \sum_{n=0}^{\infty} \frac{\left(\dot{\mathbf{p}}(0)t\right)^n}{n!}\frac{\partial^n}{\partial \mathbf{p}^n} f(0)$$

Taylor expansion of f

$$= f\left(\mathbf{p}^N(0), \left(\mathbf{r}(0) + \dot{\mathbf{r}}(0)t\right)^N\right)$$

$$= f\left(\left(\mathbf{p}(0) + \dot{\mathbf{p}}(0)t\right)^N, \mathbf{r}^N(0)\right)$$

Shift of coordinates

$$\mathbf{r}(0) \rightarrow \mathbf{r}(0) + \dot{\mathbf{r}}(0)t$$

Shift of momenta

$$\mathbf{p}(0) \rightarrow \mathbf{p}(0) + \dot{\mathbf{p}}(0)t$$

$$\mathrm{i}L_r \Rightarrow \mathbf{r}(0) \rightarrow \mathbf{r}(0) + \dot{\mathbf{r}}(0)t$$

$$\mathrm{i}L_p \Rightarrow \mathbf{p}(0) \rightarrow \mathbf{p}(0) + \dot{\mathbf{p}}(0)t$$

We have *noncommuting* operators!

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

Trotter identity

$$e^{A+B} = \lim_{P\to\infty}\left(e^{A/2P}e^{B/P}e^{A/2P}\right)^P$$

$$e^{A+B} \approx \left(e^{A/2P}e^{B/P}e^{A/2P}\right)^P$$

$$\frac{A}{P} = \frac{\mathrm{i}L_p t}{P} \qquad \frac{B}{P} = \frac{\mathrm{i}L_r t}{P} \qquad \Delta t = \frac{t}{P}$$

$$i L_r \Delta t \Rightarrow \mathbf{r} \rightarrow \mathbf{r} + \dot{\mathbf{r}} \Delta t$$

$$i L_p \Delta t \Rightarrow \mathbf{p} \rightarrow \mathbf{p} + \dot{\mathbf{p}} \Delta t$$

$$\left(e^{\left(iL_p \Delta t/2\right)} e^{\left(iL_r \Delta t\right)} e^{\left(iL_p \Delta t/2\right)} \right)^P$$

$$e^{\left(iL_p \Delta t/2\right)} f\left(\mathbf{p}^N(0), \mathbf{r}^N(0)\right) = f\left(\left[\mathbf{p}(0) + \frac{\Delta t}{2} \dot{\mathbf{p}}(0) \right]^N, \mathbf{r}^N(0) \right)$$

Velocity Verlet!

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

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \frac{\Delta t}{2m} \left[\mathbf{F}(t) + \mathbf{F}(t + \Delta t) \right]$$

$$\mathbf{p}(0) \rightarrow \mathbf{p}(0) + \frac{\Delta t}{2} \left[\dot{\mathbf{p}}(0) + \dot{\mathbf{p}}(\Delta t) \right]$$

$$\mathbf{r}(0) \rightarrow \mathbf{r}(0) + \Delta t \dot{\mathbf{r}}(\Delta t/2) = \mathbf{r}(0) + \Delta t \dot{\mathbf{r}}(0) + \frac{\Delta t^2}{2m} \mathbf{F}(0)$$

Velocity Verlet:

$$e^{(iL_p\Delta t/2)}e^{(iL_r\Delta t)}e^{(iL_p\Delta t/2)}$$

```
Call force(fx)
Do while (t<tmax)
```

$$e^{(iL_p\Delta t/2)}:\mathbf{v}\left(t+\frac{\Delta t}{2}\right)\rightarrow\mathbf{v}(t)+\frac{\Delta t}{2m}\mathbf{f}(t)$$

```
vx=vx+delt*fx/2
```

$$e^{(iL_r\Delta t)}:\mathbf{r}(t+\Delta t)\rightarrow\mathbf{r}(t)+\Delta t\mathbf{v}(t+\Delta t/2)$$

```
x=x+delt*vx
Call force(fx)
```

$$e^{(iL_p\Delta t/2)}:\mathbf{v}(t+\Delta t)\rightarrow\mathbf{v}(t+\Delta t/2)+\frac{\Delta t}{2m}\mathbf{f}(t+\Delta t)$$

```
vx=vx+delt*fx/2
enddo
```

Liouville Formulation

Velocity Verlet algorithm:

$$\mathbf{p}(t+\Delta t)=\mathbf{p}(t)+\frac{\Delta t}{2}\left[\dot{\mathbf{p}}(t)+\dot{\mathbf{p}}(t+\Delta t)\right]$$

$$\mathbf{r}(t+\Delta t)=\mathbf{r}(t)+\Delta t\dot{\mathbf{r}}(t)+\frac{\Delta t^2}{2m}\mathbf{F}(t)$$

Three subsequent coordinate transformations in either **r** or **p** of which the *Jacobian* is one: *Area preserving*

$$\mathbf{p}(t+\Delta t/2)=\mathbf{p}(t)+\frac{\Delta t}{2}\mathbf{F}(\mathbf{r})$$
$$\mathbf{r}(t)=\mathbf{r}(t)$$
$$J_1=\det\begin{vmatrix}1 & \frac{\Delta t}{2}\frac{\partial\mathbf{F}(\mathbf{r})}{\partial\mathbf{r}}\\ 0 & 1\end{vmatrix}=1$$

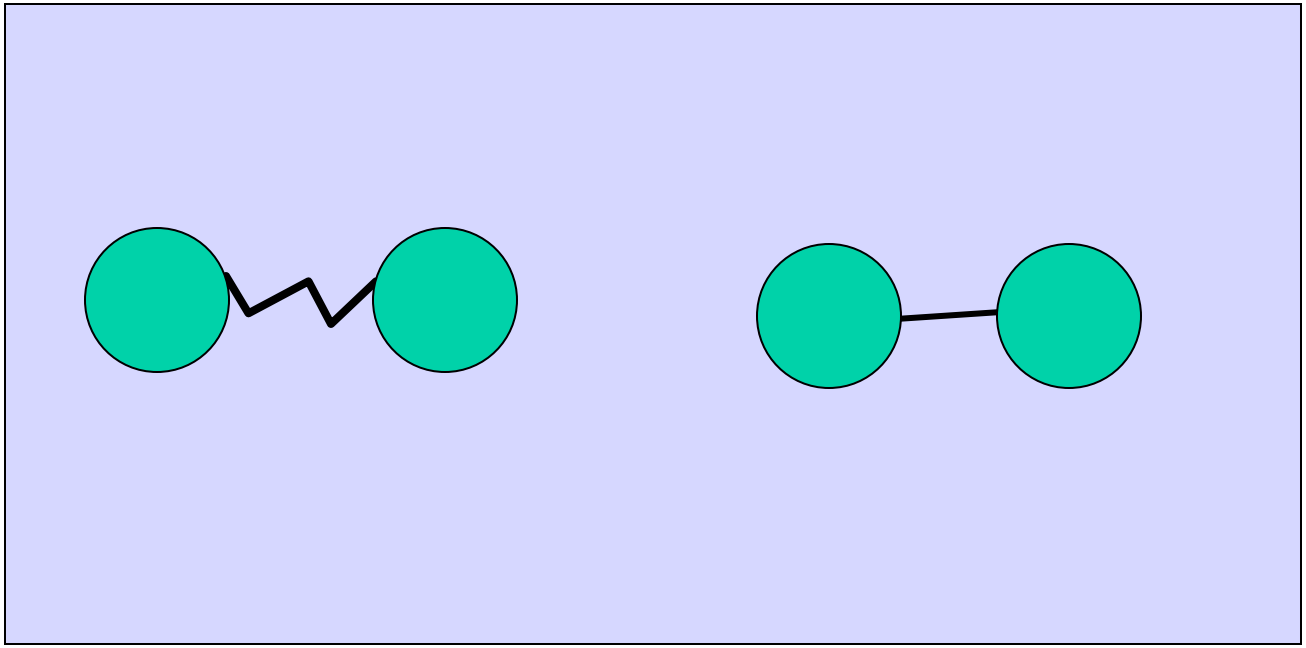
$$\mathbf{p}(t+\Delta t/2)=\mathbf{p}(t+\Delta t/2)$$
$$\mathbf{r}(t+\Delta t)=\mathbf{r}(t)+\frac{\Delta t}{m}\mathbf{p}(t+\Delta t/2)$$
$$J_2=\det\begin{vmatrix}1 & 0\\ \Delta t/m & 1\end{vmatrix}=1$$

$$\mathbf{p}(t+\Delta t)=\mathbf{p}(t+\Delta t/2)+\frac{\Delta t}{2}\mathbf{F}(\mathbf{r}(t))$$
$$\mathbf{r}(t)=\mathbf{r}(t)$$
$$J_3=\det\begin{vmatrix}1 & \frac{\Delta t}{2}\frac{\partial\mathbf{F}(\mathbf{r})}{\partial\mathbf{r}}\\ 0 & 1\end{vmatrix}=1$$

Other Trotter decompositions are possible!

Multiple time steps

- What to use for stiff potentials:



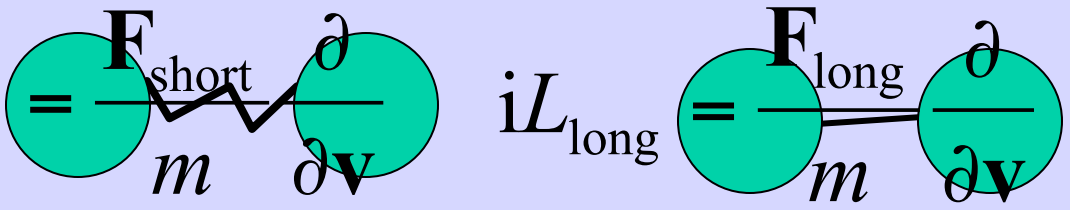
Multiple Time steps

$$\mathbf{F} = \mathbf{F}_{\text{short}} + \mathbf{F}_{\text{long}}$$

$$iL \equiv iL_r + iL_p = \mathbf{v}\frac{\partial}{\partial \mathbf{r}} + \frac{\mathbf{F}}{m}\frac{\partial}{\partial \mathbf{v}}$$

$$iL_p = iL_{\text{short}} + iL_{\text{long}}$$

$$iL_{\text{short}} = \frac{\mathbf{F}_{\text{short}}}{m}\frac{\partial}{\partial \mathbf{v}} \qquad iL_{\text{long}} = \frac{\mathbf{F}_{\text{long}}}{m}\frac{\partial}{\partial \mathbf{v}}$$



Trotter expansion:

$$e^{i(L_{\text{long}} + L_{\text{short}} + L_r)\Delta t} \approx e^{iL_{\text{long}}\Delta t/2} e^{i(L_{\text{short}} + L_r)\Delta t} e^{iL_{\text{long}}\Delta t/2}$$

Introduce: $\delta t = \Delta t / n$

$$e^{i\left(L_{\text{long}} + L_{\text{short}} + L_r\right)\Delta t} \approx e^{iL_{\text{long}}\Delta t/2}\left[e^{iL_{\text{short}}\delta t/2} e^{iL_r \delta t} e^{iL_{\text{short}}\delta t/2}\right]^n e^{iL_{\text{long}}\Delta t/2}$$

$$e^{i\left(L_{\text{long}}+L_{\text{short}}+L_r\right)\Delta t} \approx e^{iL_{\text{long}}\,\Delta t/2}\left[e^{iL_{\text{short}}\,\delta t/2}e^{iL_r\delta t}e^{iL_{\text{short}}\,\delta t/2}\right]^n e^{iL_{\text{long}}\,\Delta t/2}$$

$$iL_{\text{long}}\,\Delta t/2 \Rightarrow v \rightarrow v + F_{\text{long}}\,\Delta t/2m$$

$$iL_{\text{short}}\,\delta t/2 \Rightarrow v \rightarrow v + F_{\text{short}}\,\delta t/2m$$

$$iL_r\delta t \Rightarrow r \rightarrow r + v\delta t$$

First

$$e^{iL_{\text{long}}\,\Delta t/2} f\left[r(0),v(0)\right] = f\left[r(0),v(0)+F_{\text{long}}(0)\Delta t/2m\right]$$

Now n times:

$$\left[e^{iL_{\text{short}}\,\delta t/2}e^{iL_r\delta t}e^{iL_{\text{short}}\,\delta t/2}\right]^n f\left[r(0),v(0)+F_{\text{long}}(0)\Delta t/2m\right]$$

$$e^{\left(iL_{p\mathrm{Long}}\Delta t/2\right)}:\mathbf{v}\left(t+\frac{\Delta t}{2}\right)\rightarrow\mathbf{v}\left(t\right)+\frac{\Delta t}{2m}\mathbf{f}_{\mathrm{long}}\left(t\right)$$

```
Call force(fx_l
vx=vx+delt*fx_l
Do ddt=1,n
```

$$e^{\left(iL_{p\mathrm{Short}}\delta t/2\right)}:\mathbf{v}\left(t+\right.$$

```
   vx=vx+ddelt*fx_short/2
```

$$e^{\left(iL_{r}\delta t\right)}:\mathbf{r}\left(t+\delta t\right)\rightarrow\mathbf{r}\left(t\right)+\delta t\mathbf{v}\left(t+\Delta t/2+\delta t/2\right)$$

```
   x=x+ddelt*
   Call force
```

$$e^{\left(iL_{p\mathrm{Short}}\delta t/2\right)}:\mathbf{v}\left(t+\right.$$

```
   vx=vx+ddel
enddo
```

$$e^{iL_{\mathrm{long}}\Delta t/2}\left[e^{iL_{\mathrm{short}}\delta t/2}e^{iL_{r}\delta t}e^{iL_{\mathrm{short}}\delta t/2}\right]^{n}e^{iL_{\mathrm{long}}\Delta t/2}$$

$$iL_{\mathrm{long}}\Delta t/2\Rightarrow v\rightarrow v+F_{\mathrm{long}}\Delta t/2m$$

$$iL_{\mathrm{short}}\delta t/2\Rightarrow v\rightarrow v+F_{\mathrm{short}}\delta t/2m$$

$$iL_{r}\delta t\Rightarrow r\rightarrow r+v\delta t$$

$$e^{iL_{\mathrm{long}}\Delta t/2}\left[e^{iL_{\mathrm{short}}\delta t/2}e^{iL_{r}\delta t}e^{iL_{\mathrm{short}}\delta t/2}\right]^{n}e^{iL_{\mathrm{long}}\Delta t/2}$$

$$iL_{\mathrm{long}}\Delta t/2\Rightarrow v\rightarrow v+F_{\mathrm{long}}\Delta t/2m$$

$$iL_{\mathrm{short}}\delta t/2\Rightarrow v\rightarrow v+F_{\mathrm{short}}\delta t/2m$$

$$iL_{r}\delta t\Rightarrow r\rightarrow r+v\delta t$$

Algorithm 29 (Multiple Time Step)

```
 subroutine
*     multi(f_long,f_short)

 vx=vx+0.5*delt*f_long
 do it=1,n
    vx=vx+0.5*(delt/n)*f_short
    x=x+(delt/n)2*vx
    call force_short(f_short)
    vx=vx+0.5*(delt/n)*f_short
 enddo
 call force_all(f_long,f_short)
 vx=vx+0.5*delt*f_long
 return
 end
```

Multiple time step, `f_long` is the long-range part and `f_short` the short-range part of the force

velocity Verlet with time step Δt

loop for the small time step

velocity Verlet with timestep $\Delta t/n$

short-range forces

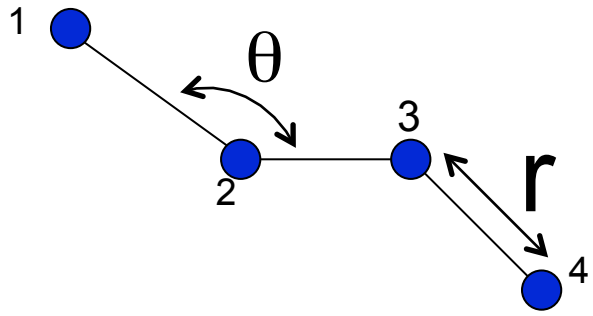
all forces

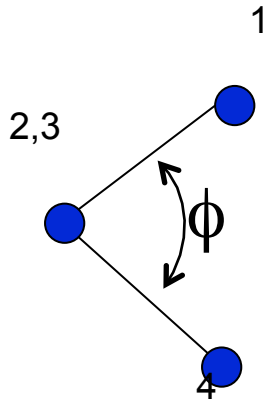
All-atom force fields for biomolecules

- Potential energy for protein

$$V(\mathbf{r}) = \sum_{bonds} k_r (r - r_{eq})^2 + \sum_{angles} k_\theta (\theta - \theta_{eq})^2 + \sum_{dihedrals} \frac{1}{2} v_n (1 + \cos(n\phi - \phi_0)$$

$$+ \sum_{i<j} \left(\frac{a_{ij}}{r_{ij}^{12}} - \frac{b_{ij}}{r_{ij}^{6}} + \frac{q_i q_j}{\varepsilon r_{ij}} \right)$$





vdW interactions only between non-bonded |i-j|>4

Currently available empirical force fields

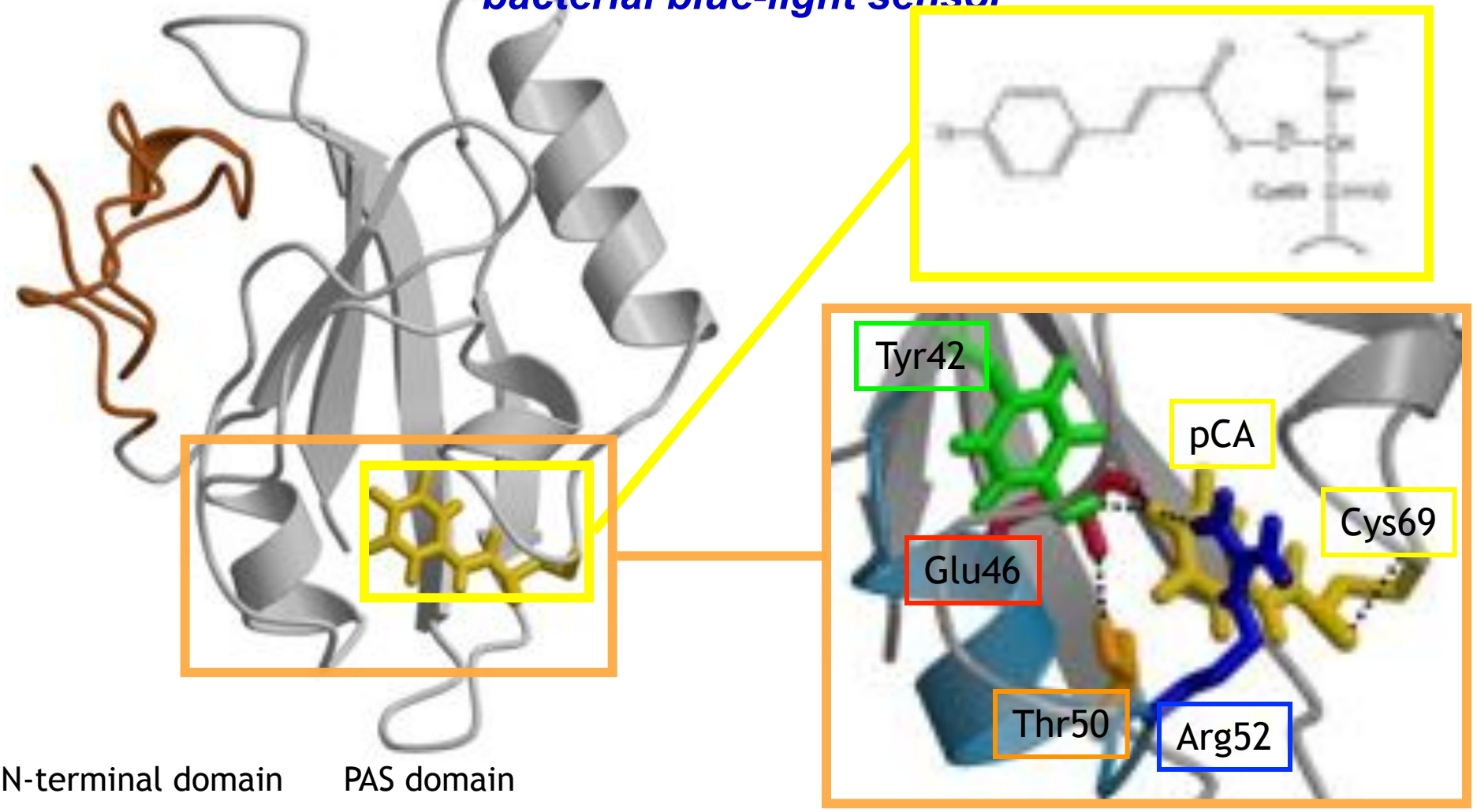
- CHARMm (MacKerrel et 96)
- AMBER (Cornell et al. 95)
- GROMOS (Berendsen et al 87)
- OPLS-AA (Jorgensen et al 95)
- ENCAD (Levitt et al 83)

- Subtle differences in improper torsions, scale factors 1-4 bonds, united atom rep.
- Partial charges based on empirical fits to small molecular systems
- Amber & Charmm also include ab-initio calculations
- Not clear which FF is best : top 4 mostly used

- Water models also included in description
  - TIP3P, TIP4P
  - SPC/E
- Current limit: 10^5 atoms, 1000 ns

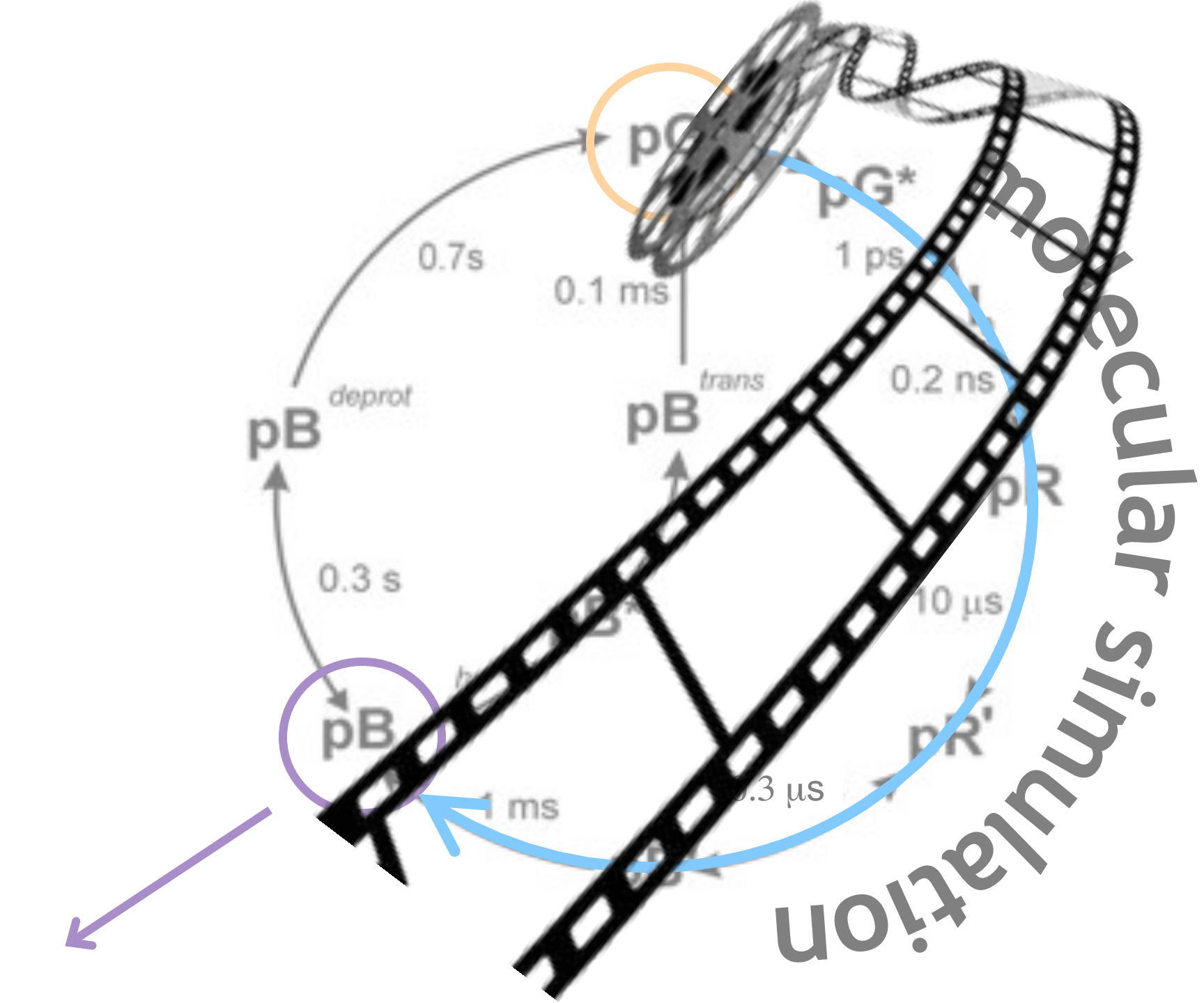
Photoactive Yellow Protein

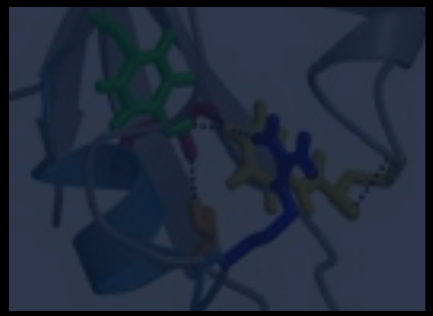
bacterial blue-light sensor

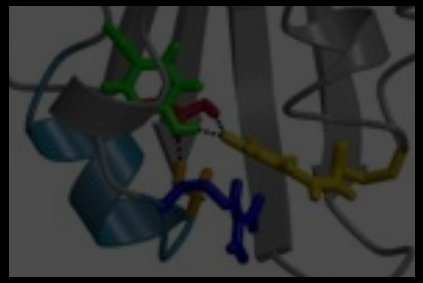


Absorption of a blue-light photon triggers the photo cycle

J. Vreede et al. Biophys. J. 2005

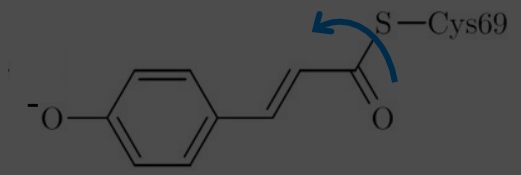

pG
pG*
0.7s
1 ps
0.1 ms
0.2 ns
pB deprot
pB trans
pR
0.3 s
10 µs
pB
pR'
1 ms
.3 µs
molecular simulation

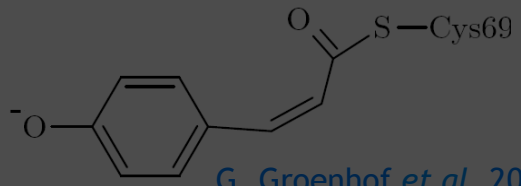


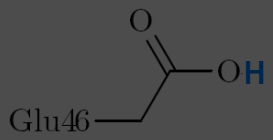


n
s
μ
s
m
s

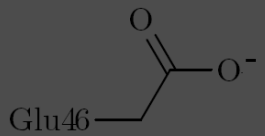
Formation of signaling state

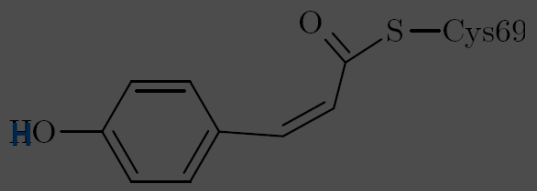


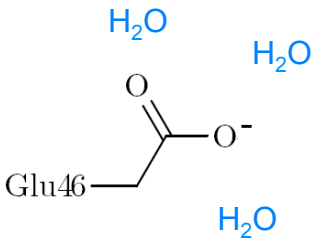


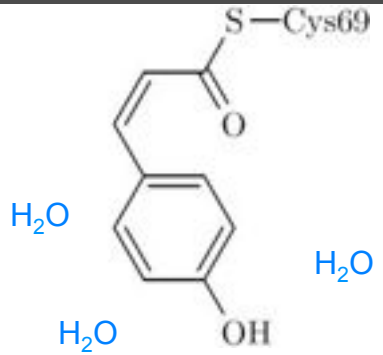


G. Groenhof *et al.* 2005, 2008



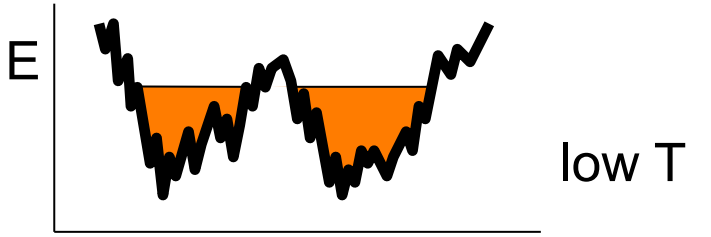




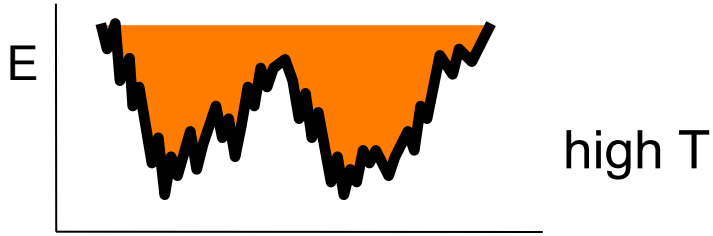


Replica Exchange works for MD

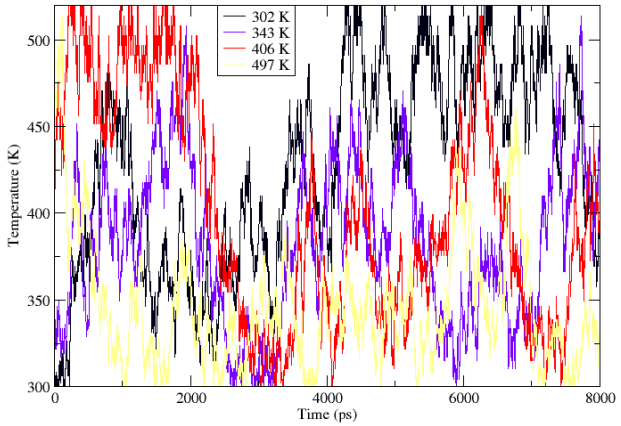
High barriers in energy landscape: difficult to sample



Barriers effectively low: easy to sample



Consider M replica's in the NVT ensemble at a different temperature.



$$\frac{i \to j}{j \to i} = \exp -\left[(\beta_i - \beta_j)(U(j) - U(i))\right]$$

A swap between two systems of different temperatures (T_i,T_j) is accepted if their potential energies overlap.

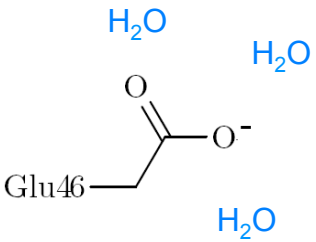
Advantage: better sampling phase space

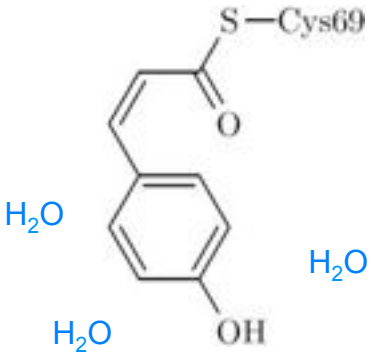
Partial unfolding

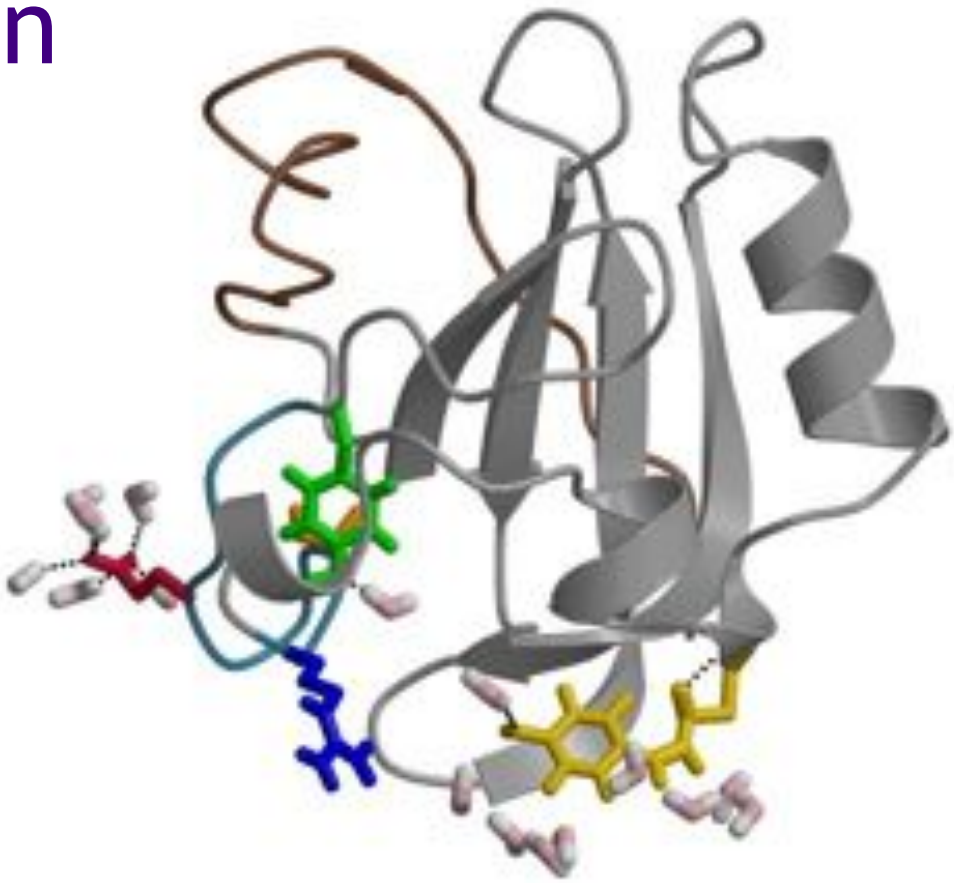
- Loss of α-helical structure
- Exposure of hydrophobic groups
- Increased flexibility in parts of the protein backbone

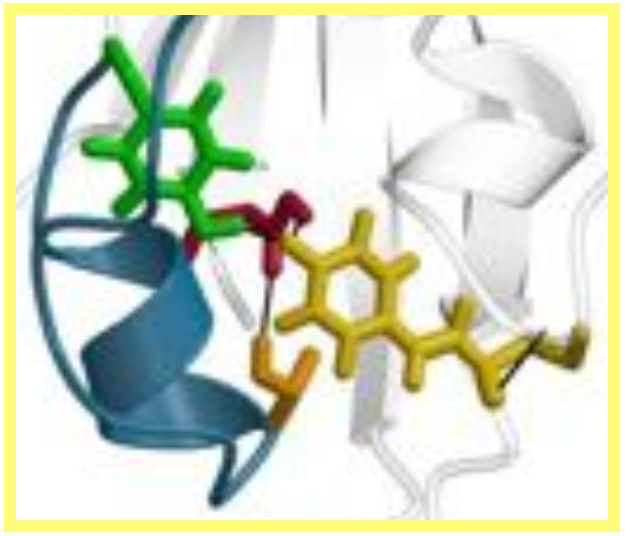
Molecular simulation

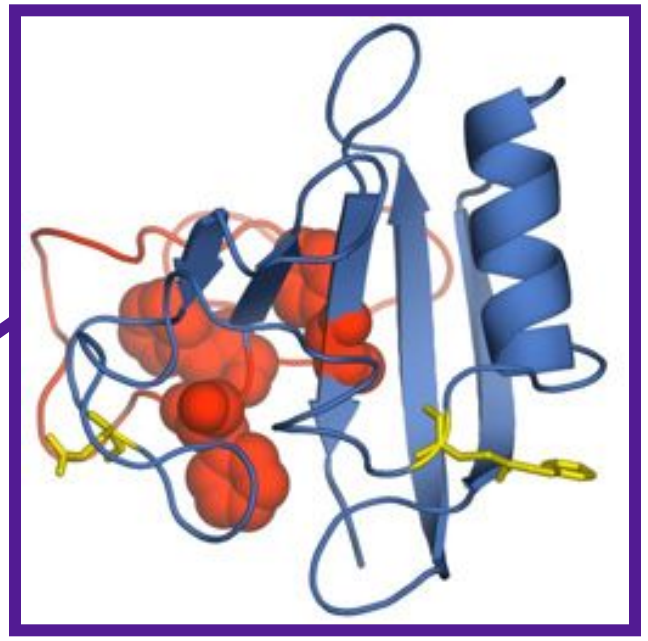
cis-chroH + Glu46-
Force field MD (gromacs)
Gromos96 - SPC water - PME
Replica Exchange

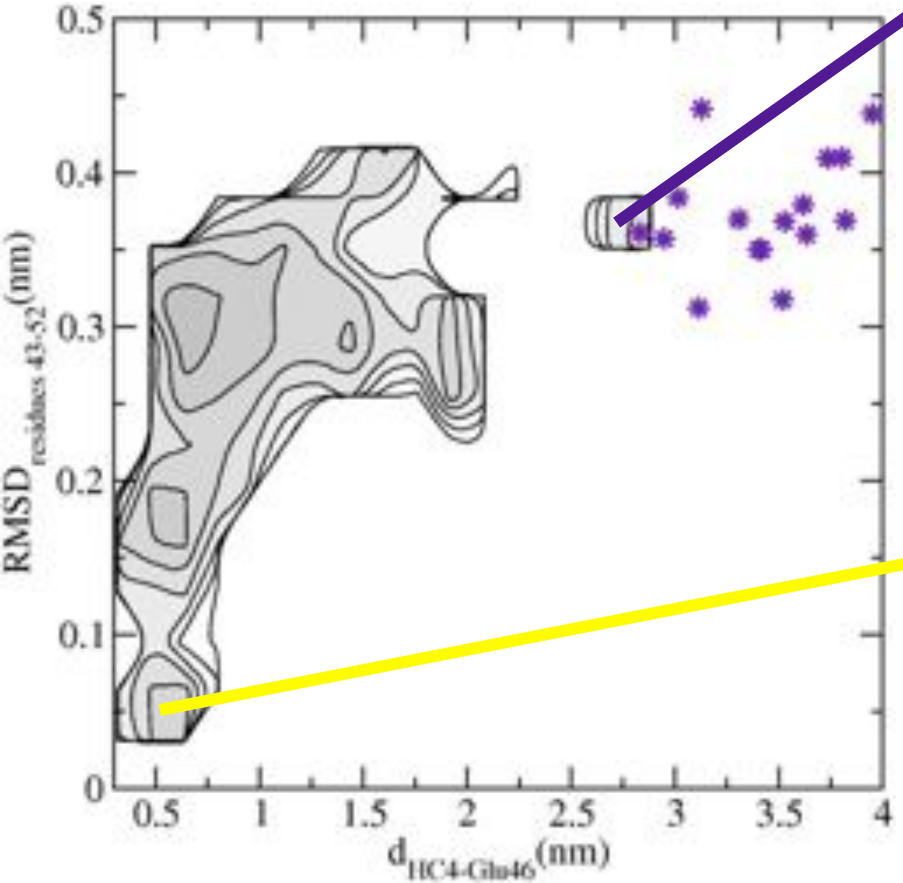


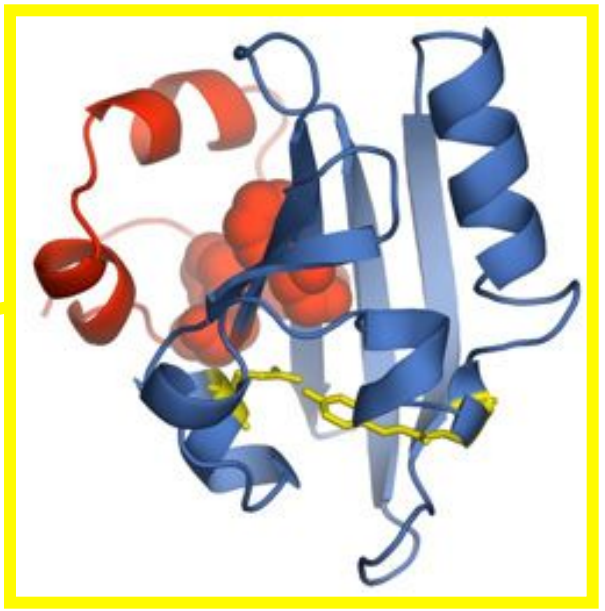












Exp: Bernard et al. Structure 2005

A REMD trajectory