Introduction to RefCode

March 25, 2016

Instructor: Prof. Yeng-Long Chen

TA: 林子翔 (Tzyy-Shyang Lin)

Email: ts.lin.92@gmail.com

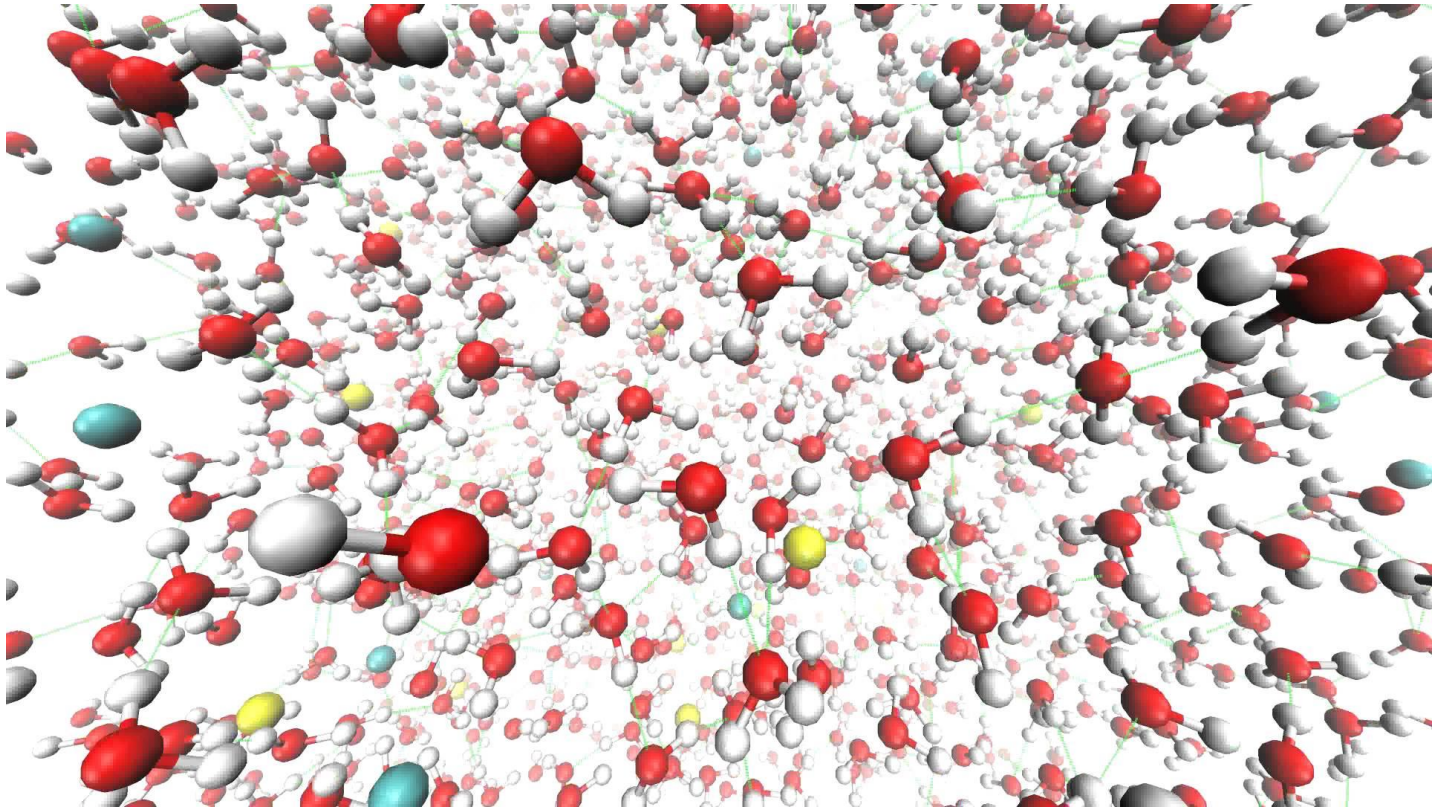
Get your account

- Server IP: 140.114.129.170 (ssh)

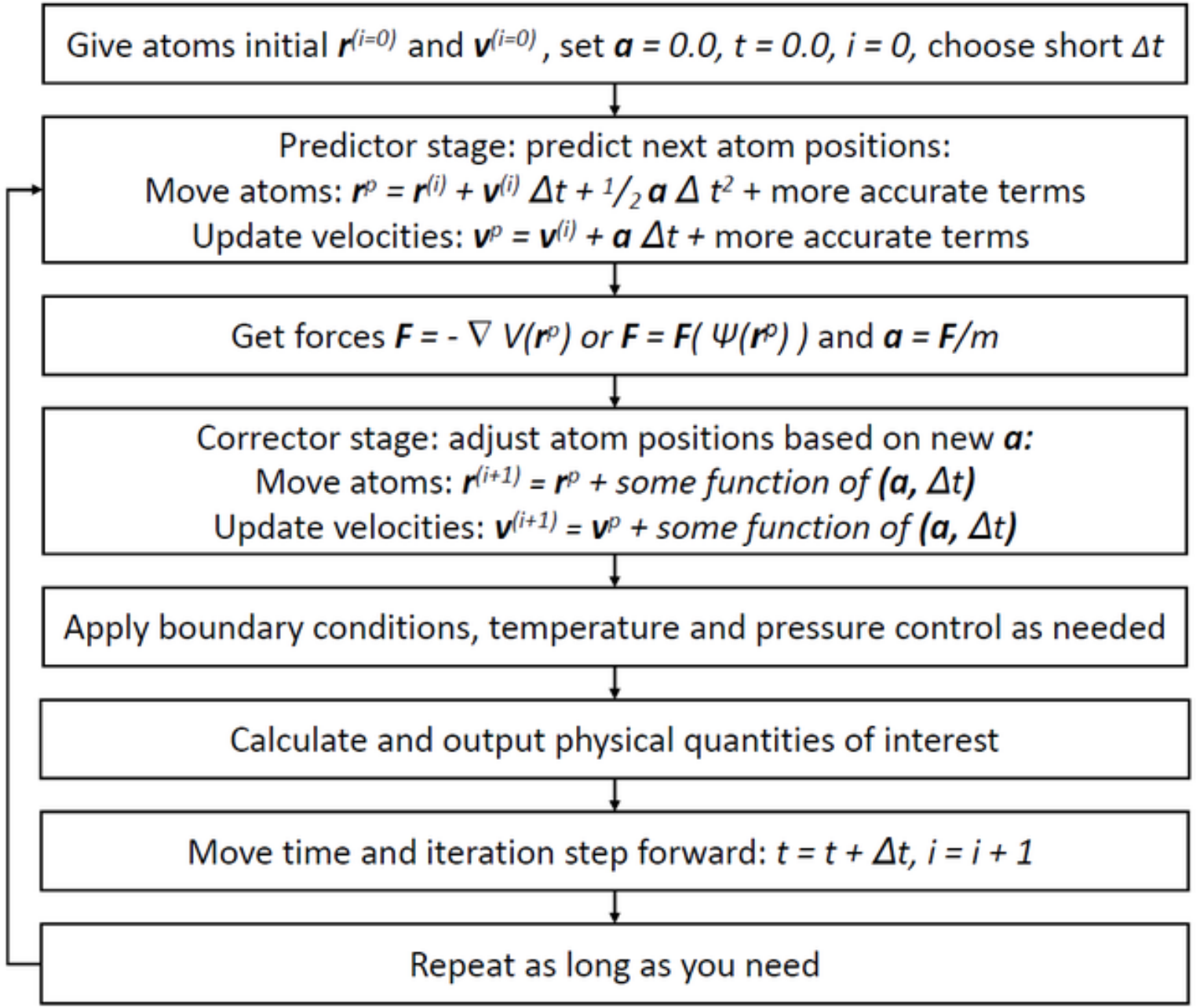
Username	Password	Student ID	Student Name	Email
2015sm1	phys7039a			
2015sm2	phys7039b			
2015sm3	phys7039c			
2015sm4	phys7039d			
2015sm5	phys7039e			
2015sm6	phys7039f			
2015sm7	phys7039g			

Last Week

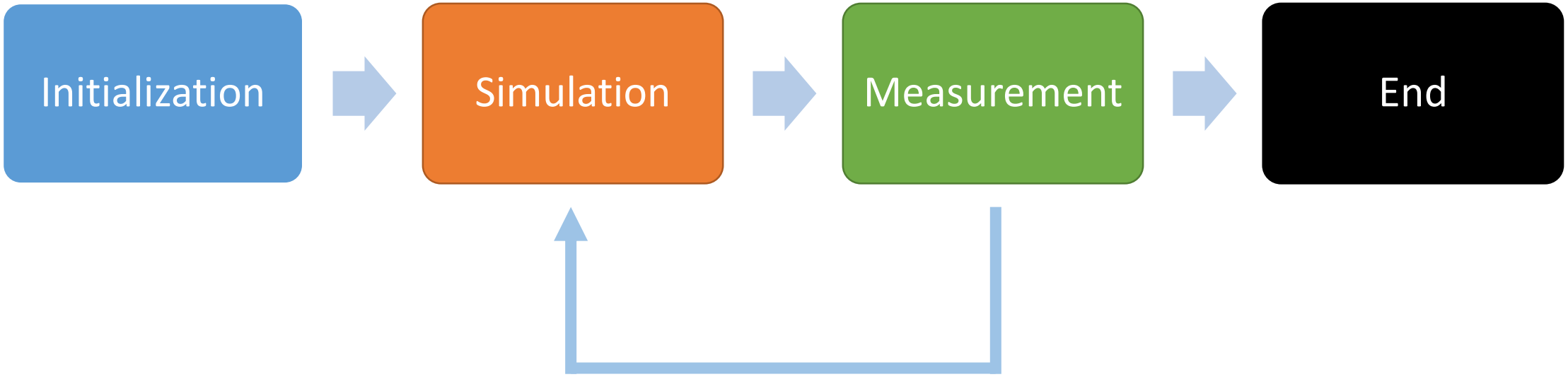
- We talked about the general aspects of MD simulation



Recipe of MD

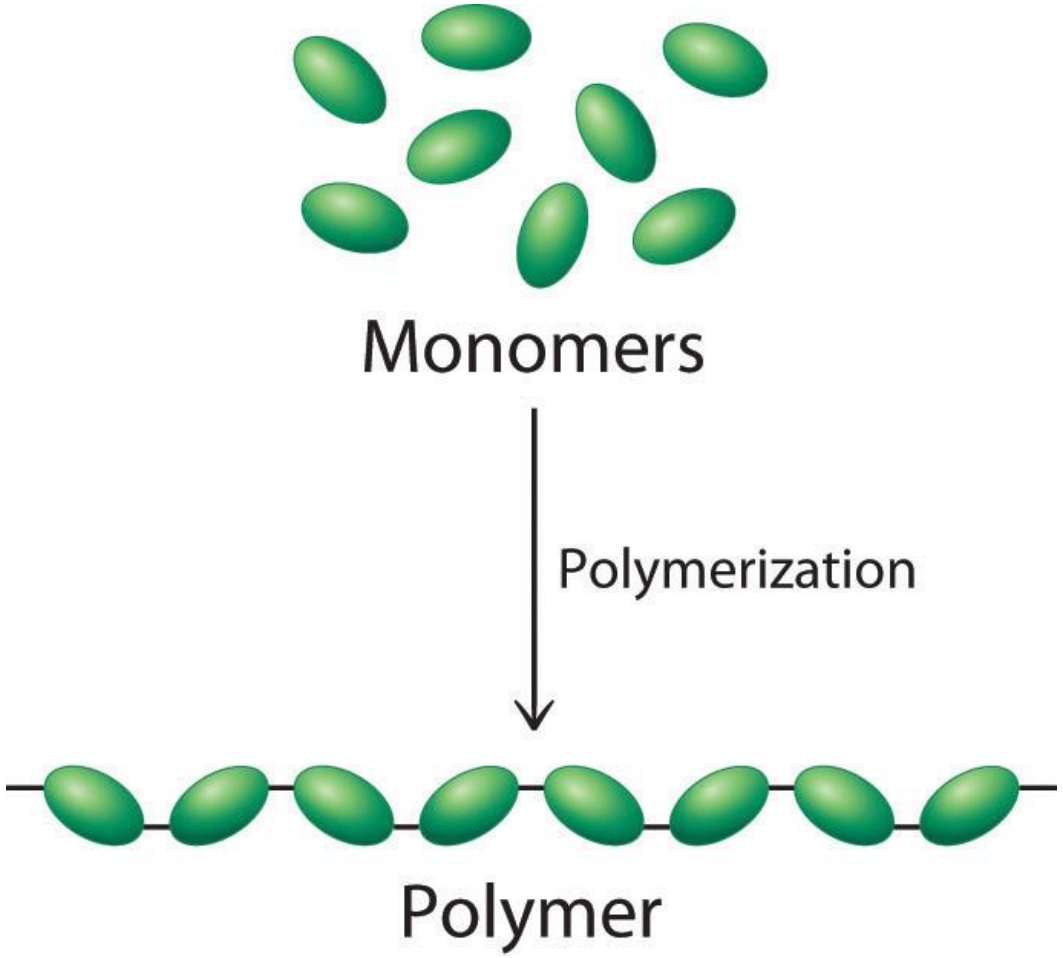

Give atoms initial $\mathbf{r}^{(i=0)}$ and $\mathbf{v}^{(i=0)}$, set $\mathbf{a} = 0.0$, $t = 0.0$, $i = 0$, choose short Δt
Predictor stage: predict next atom positions:
Move atoms: $\mathbf{r}^p = \mathbf{r}^{(i)} + \mathbf{v}^{(i)}\,\Delta t + {}^1/_2\,\mathbf{a}\,\Delta\,t^2$ + more accurate terms
Update velocities: $\mathbf{v}^p = \mathbf{v}^{(i)} + \mathbf{a}\,\Delta t$ + more accurate terms
Get forces $\mathbf{F} = -\nabla V(\mathbf{r}^p)$ or $\mathbf{F} = \mathbf{F}(\Psi(\mathbf{r}^p))$ and $\mathbf{a} = \mathbf{F}/m$
Corrector stage: adjust atom positions based on new $\mathbf{a}$:
Move atoms: $\mathbf{r}^{(i+1)} = \mathbf{r}^p$ + some function of $(\mathbf{a}, \Delta t)$
Update velocities: $\mathbf{v}^{(i+1)} = \mathbf{v}^p$ + some function of $(\mathbf{a}, \Delta t)$
Apply boundary conditions, temperature and pressure control as needed
Calculate and output physical quantities of interest
Move time and iteration step forward: $t = t + \Delta t$, $i = i + 1$
Repeat as long as you need

Simplified Recipe for MD Simulation

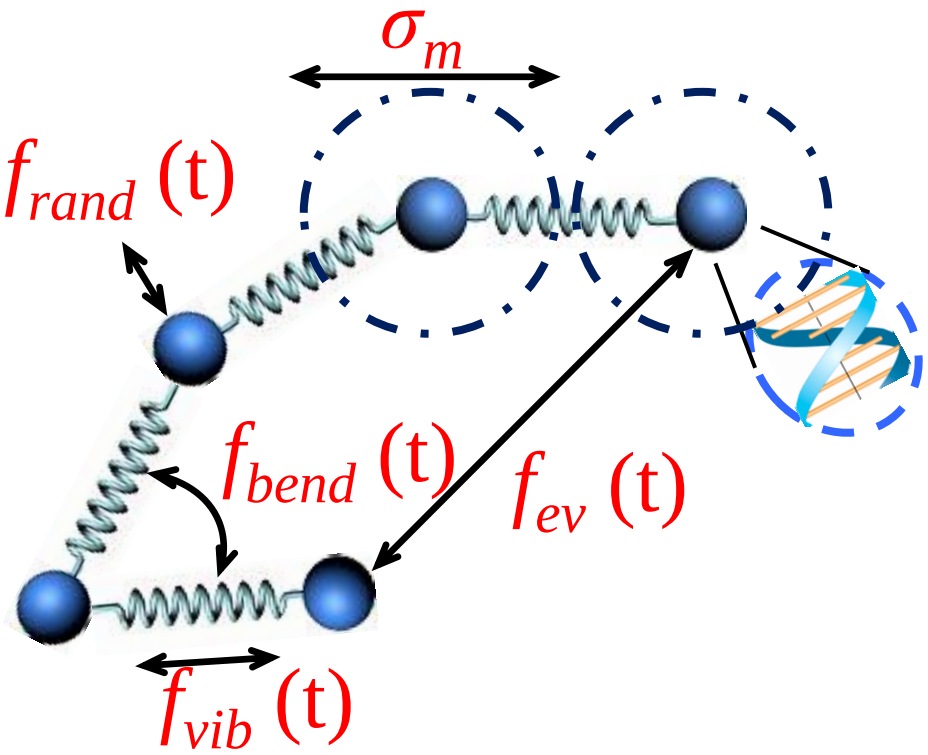


Now about the Reference Code

- The reference code simulates a model polymer system with semi-flexible chain



DNA Model with Langevin Dynamics



- Bonded Interaction
  - *Uvib*
  - *Ubend*
- Non-Bonded Int.
  - *Uev*

Numerical Method: Velocity Verlet

$$\vec{f}(t+\tau) = -\nabla U - \zeta \vec{v}(t) + \vec{f}_{rand}(t)$$

$$U = U_{vib} + U_{bend} + U_{ev} + U_{wall}$$

$$U_{vib} = \frac{k_{vib} k_B T}{2\sigma_m^2} \sum \left(\left| \vec{r}_i - \vec{r}_{i+1} \right| - \sigma_m \right)^2$$

$$U_{bend} = k_{bend} k_B T \sum \left(1 - \frac{(\vec{r}_{i-1} - \vec{r}_i)(\vec{r}_i - \vec{r}_{i+1})}{\left| \vec{r}_{i-1} - \vec{r}_i \right| \left| \vec{r}_i - \vec{r}_{i+1} \right|} \right)$$

$$U_{ev,mn} = \varepsilon_{mn} k_B T \sum \exp\left[-\alpha_{mn} \left(r_{i,j} - \sigma_{mn} \right) \right]$$

m-n: monomer-monomer, monomer-nanoparticle

$$U_{wall,n} = k_{wall,n} k_B T \left(\sigma_n - \left| \vec{r}_n \cdot \hat{i} - \vec{r}_{wall} \right| \right)^3$$

$$, if \ | \vec{r}_n \cdot \hat{i} - \vec{r}_{wall} | < \sigma_n$$

m: monomer; n: nanoparticle

Velocity Verlet Method

- $\vec{x}^{i+1} = \vec{x}^{i} + \vec{v}^{i}\Delta t + \frac{1}{2}\vec{a}^{i}\Delta t^{2}$
- $\vec{a}^{i+1} = \vec{F}^{i+1}/m = -\nabla U(\vec{x}^{i+1})/m$
- $\vec{v}^{i+1} = \vec{v}^{i} + \frac{\vec{a}^{i+1}+\vec{a}^{i}}{2}dt$

The error for velocity Verlet method is of the same order as the basic Verlet method, with position error of $O(\Delta t^{4})$, and velocity error of $O(\Delta t^{2})$

Langevin Dynamics

- $m\vec{a} = -\nabla U - \zeta\vec{v} + \sqrt{2\zeta k_B T}\vec{R}(t)$

Integration Scheme:

- $\vec{x}^{i+1} = \vec{x}^{i} + \vec{v}^{i}\Delta t + \frac{1}{2}\vec{a}^{i}\Delta t^2$
- $\vec{a}^{i+1} = [-\nabla U(\vec{x}^{i+1}) - \zeta\vec{v}^{i} + \sqrt{2\zeta k_B T}\vec{R}(t)]/m$
- $\vec{v}^{i+1} = \vec{v}^{i} + \frac{\vec{a}^{i+1}+\vec{a}^{i}}{2}dt$

- ζ: damping coefficient (energy dissipation due to viscous fluid)

- $\vec{R}(t)$: noise term (thermal fluctuation), a delta-correlated stationary Gaussian process with zero-mean,
  - $\langle\vec{R}(t)\rangle = 0$
  - $\langle\vec{R}(t)\vec{R}(t')\rangle = \delta(t - t')$

Nondimensionalization

- The following set of quantities are used to nondimensionalize the equations
  - Monomer diameter σ_M
  - Damping coefficient ζ
  - Energy scale $k_B T$

- Other physical quantities are nondimensionalized
  - $t = \tau t^* = \left(\frac{\zeta \sigma_M^2}{k_B T}\right) t^*$
  - $m = \Lambda m^* = \left(\frac{\zeta^2 \sigma_M^2}{k_B T}\right) m^*$
  - $f = \Phi f^* = \left(\frac{k_B T}{\sigma_M}\right) f^*$

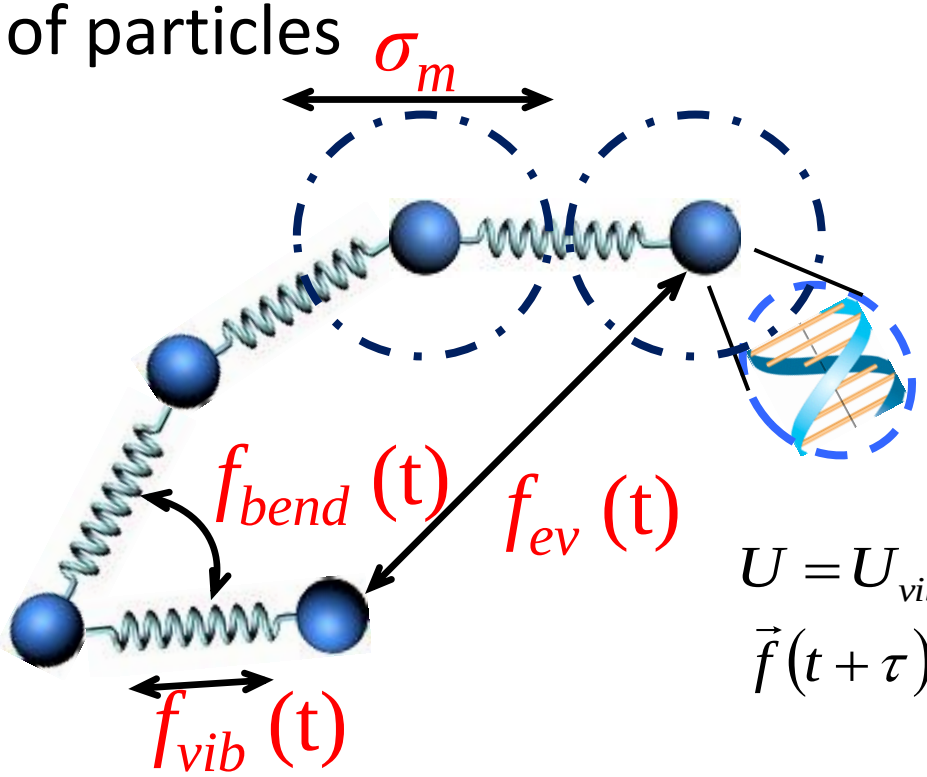
Input Parameters (input.dat)

- nstep = # of steps / cycle
- lattice = initial configuration. 0=read from config.xyz, 1=stretch config, 2=random config, 3=screw config
- Ncycl = # of cycles
- Dt = integration time step
- ELL = # of monomers
- BOX_X, BOX_Y, BOX_Z = dimensions of the simulation box

Tot step = nstep*Ncycl*microstep, microstep is defined in *int main(…)*

Simulation & DNA Energy Calculation

- Calculating energies and forces
- Updating velocity and position of particles



- euler.c
  - $\vec{v}_i(t), \vec{r}_i(t) \rightarrow \vec{v}_i(t+\Delta t), \vec{r}_i(t+\Delta t)$
- toterg.c (Utot)
  - enervib.c (Uvib)
  - enerbend.c (Ubend)
  - ener_ev.c (Uev)
  - wall_force.c (Uwall)

$$U = U_{vib} + U_{bend} + U_{ev} + U_{wall}$$

$$\vec{f}(t+\tau) = -\nabla U$$

Input Parameters (input.dat)

- WALL_FLAG = type of confinement. 0= bulk, 1=y walls, 2=yz walls
- TEMP = temperature
- KV = bond vibrational energy parameter
- KB = bending energy parameter

$$U_{vib} = \frac{k_{vib} k_B T}{2\sigma_m^2} \sum \left(\left| \vec{r}_i - \vec{r}_{i+1} \right| - \sigma_m \right)^2$$

$$U_{bend} = k_{bend} k_B T \sum \left(1 - \frac{\left(\vec{r}_{i-1} - \vec{r}_i \right)\left(\vec{r}_i - \vec{r}_{i+1} \right)}{\left| \vec{r}_{i-1} - \vec{r}_i \right| \left| \vec{r}_i - \vec{r}_{i+1} \right|} \right)$$

Input Parameters (input.dat)

- EPS_M/NP

  excluded volume parameter (magnitude) between monomers/nanoparticles

- ALPHA_M/NP

  excluded volume parameter (range) between monomers/nanoparticles

$$U_{ev,mn} = \varepsilon_{mn} k_B T \sum \exp\left[-\alpha_{mn}\left(r_{i,j} - \sigma_{mn}\right)\right]$$

m-n: monomer-monomer, monomer-nanoparticle

$$\varepsilon_{MN} = \sqrt{\varepsilon_{MM}\varepsilon_{NN}}, \alpha_{MN} = \left(\alpha_{MM} + \alpha_{NN}\right)/2$$

Input Parameters (input.dat)

- WALL_EPS_M/NP

  wall repulsion parameter (magnitude) between monomer/nanoparticles and wall

- WALL_ALPHA_M/NP

  wall repulsion parameter (range) between monomer/nanoparticles and wall

$$U_{wall,n} = \varepsilon_{wall,n} k_B T \left(\alpha_n - \left| \vec{r}_n \cdot \hat{i} - \vec{r}_{wall} \right| \right)^3$$

$$, if \ | \vec{r}_n \cdot \hat{i} - \vec{r}_{wall} | < \alpha_n$$

n: *monomer, nanoparticle*

Input Parameters (input.dat)

- SIGMA_M/NP = diameter of monomer/nanoparticle
- MASS_M/NP = monomer/nanoparticle mass
- SEED = random seed
- INTG = integration type, 0 = euler, 1 = implicit BD, 2=velocity verlet

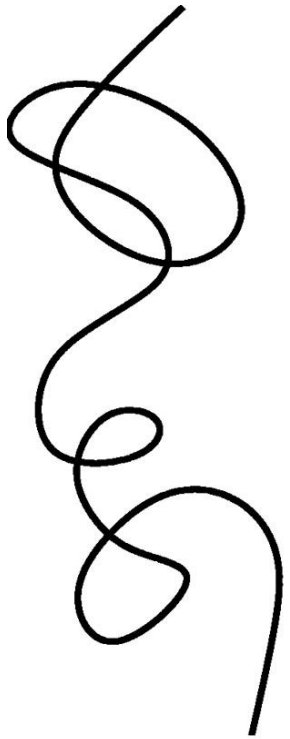
Correlating Simulation to Real System

- In previous slides, the actual value of the quantities are not assigned
- They need to be related to real systems
- The relation of $k_B T$ is straightforward, where T is temperature

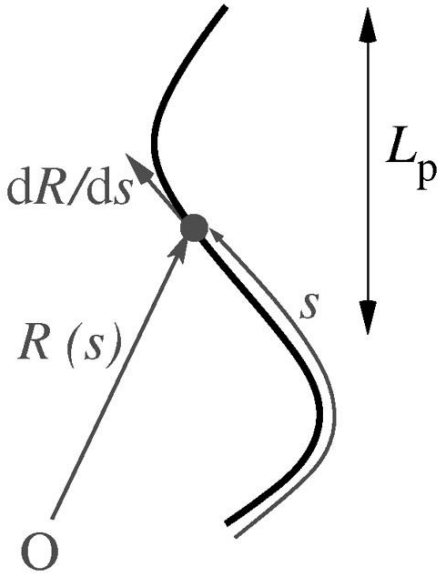
Relating σ_M

- To determine σ_M, simulation results are required
- For a semiflexible chain, with the worm-like-chain (WLC) model, the decay rate of the correlation of tangent vector ($\vec{t}(s) = d\vec{r}(s)/ds$) can be described with the persistence length l_p

$$\langle \vec{t}(s) \cdot \vec{t}(0) \rangle = e^{-s/l_p}$$



$L \gg L_p$



$L \sim L_p$

$L \ll L_p$

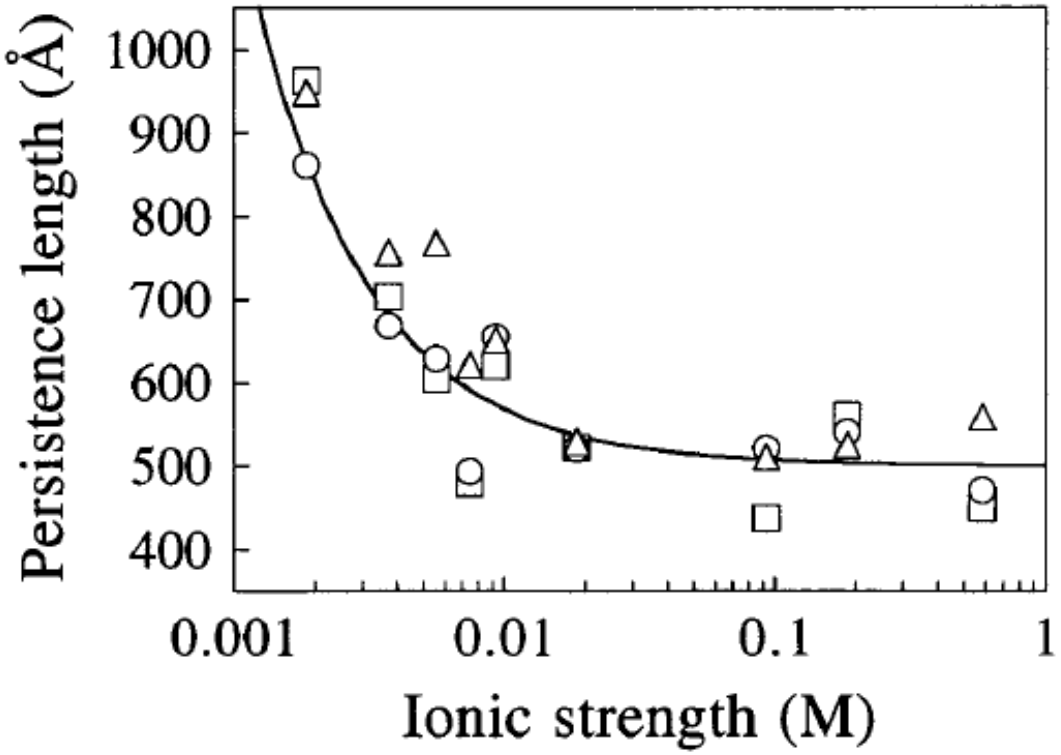
Relating σ_M

- The persistence length can be calculated in the simulation
- For example, with $k_{bend} = 5$, we find $l_p = 5\ (\sigma_M)$
- The persistence length in the simulation can be related to the persistence length of the actual system of interest
- For example, in concentrated monovalent salt solution, dsDNA has persistence length $l_p = 50$ nm
- So in this case, $5\sigma_M = 50$ nm $\Rightarrow \sigma_M = 10$ nm

Relating σ_M

- Different k_{bend} gives different l_p
- Different experimental conditions and polymer of interest correspond to different l_p
- With nondimensional simulation, a single simulation can represent many different experimental conditions

- This is the power of nondimensionalization



Persistence length dependence of dsDNA on ionic strength in monovalent salt.
Baumann et al. (1997). *Proc Nat Aca Sci*, ***94***(12), 6185-6190.

Relating ζ

- From Stokes law, $\vec{F}_d = \zeta\vec{v} = 6\pi\eta R\vec{v}$
  - $\vec{F}_d$: drag force
  - η: dynamical viscosity of the fluid
  - R: particle radius
- $\zeta = 3\pi\eta\sigma_M$ in our case, since $\eta = 1$ cP for water, if we take the result $\sigma_M = 10$ nm from previous analysis, we have $\zeta = 8.39 * 10^{-11}$ kg/s
- Similarly, we can easily relate the simulation to other conditions by using different η for different solvents or temperatures

Relating to Real System

- Other quantities can also be interpreted by similar arguments

- For more detailed explanation, please refer to the additional note on the nondimensionalization of the simulation

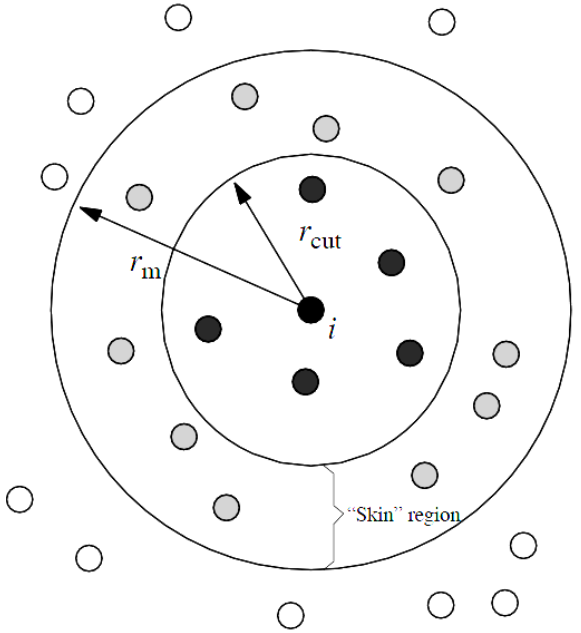
Creating Neighbor List

- Energy/Force Calculation is necessary in Molecular Simulations. The most time consuming part is non-bonded interaction, which is $O(N^2)$.
- For short range forces, there are many pairs who don't interact with each other. Thus, algorithms were developed to reduce the computing cost.
  - Verlet List
  - Cell List
  - Combined List

```
for ( int i = 0 ; i < N ; i++ )
  for ( int j = i+1 ; j < N ; j++ ) {
    F[i] = non_bonded_int(r[i], r[j]);
    F[j] = -F[i];
  }
```

Verlet list

- A list of all particles interacting with a particle is recorded and periodically updated for every particle
- Complexity of tasks
  - Creating neighbor list $\sim O(N^2)$
  - Lists are updated every n_u steps
  - The CPU time to calculate energy using Verlet list is $t_v = cn_vN + c_vN^2/n_u$
- After optimization, the overall complexity is $\sim O(N^{1.5})$



n_v: number of particles in a list
n_u: period of list updates
N: total number of particles
c: coefficient for calculating energy
c_u: coefficient for creating list

File Description

File Description

- header.h: defines the global parameters and global functions.
- main.c: the main file that handles initialization, force calculation, trajectory integration, and data output
- init.c: reads in the parameters from input.dat
- initlat.c: initializes the particle positions
- fileoutput.c: analyzes particle configurations to determine polymer size, displacement, and write output to files.
- randistrs.c/h, mtwist.c/h: generating random numbers

File Description

- euler.c: integrates the particle trajectories using Euler or Verlet algorithms
- toterg.c: calls to evaluate all the energies and forces
- enerbend.c: evaluates the bonded chain bending forces
- ener_ev.c: evaluates the non-bonded interparticle forces (WCA Potential, Repulsive Morse Potential)
- enervib.c: evaluates the bonded particleforces
- ev_nlist.c: Use the neighbor list method to evaluate non-bonded interparticle forces
- wall_force.c: evaluates forces from the wall boundaries

Measurement (fileoutput.c)

- Recording position of particles, which can later be visualized by VMD.
- Measuring and outputting some properties of particles (e.g. diffusion, molecule's structure, …)

- void fileoutput(…);
  - Outputting files:

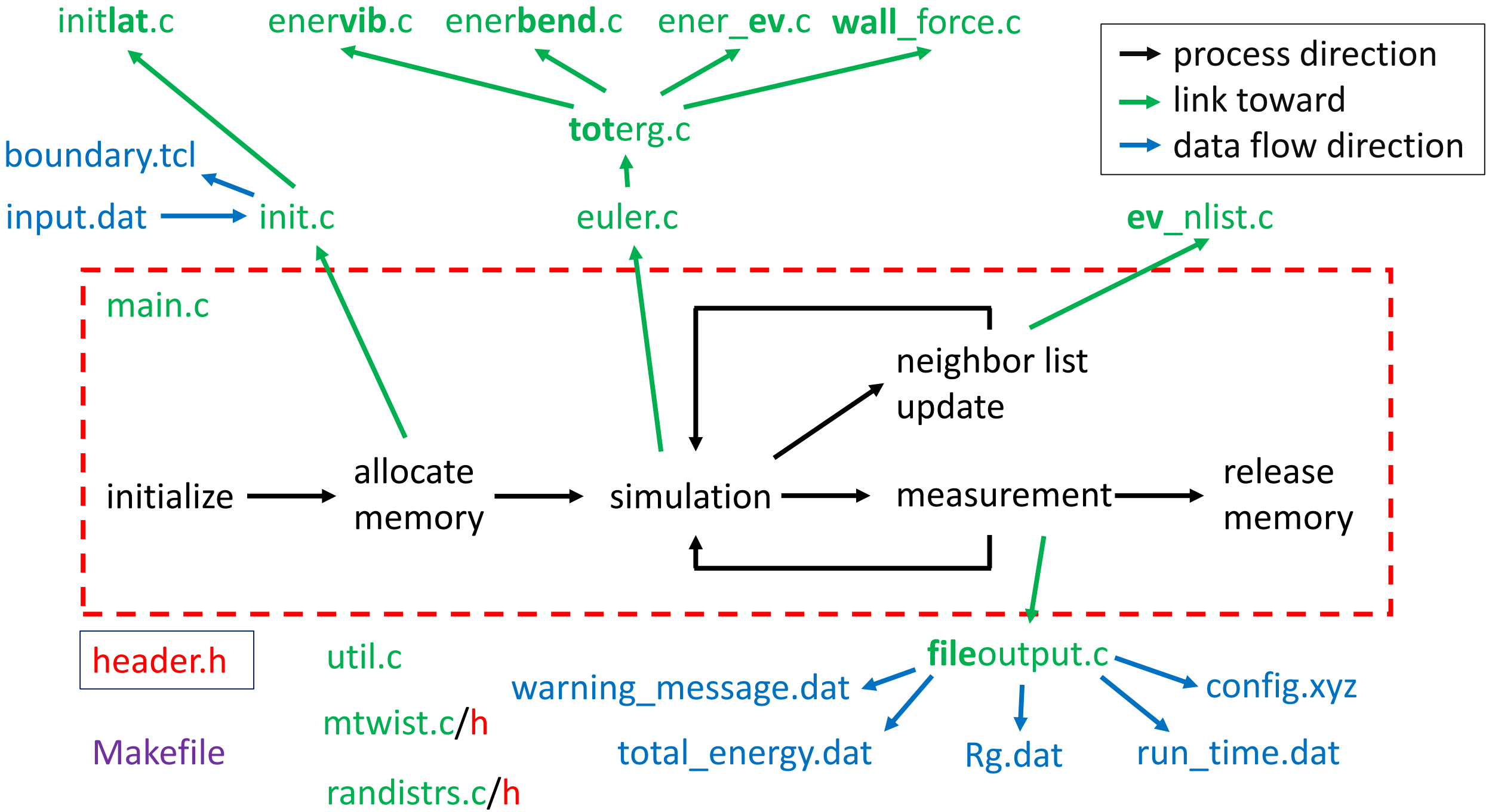
→config.xyz (particle configuration)

→run_time.dat (program efficiency)

→total_energy.dat (Etot, Ek, U)

→warning_message.dat

→Rg.dat (properties of polymer)

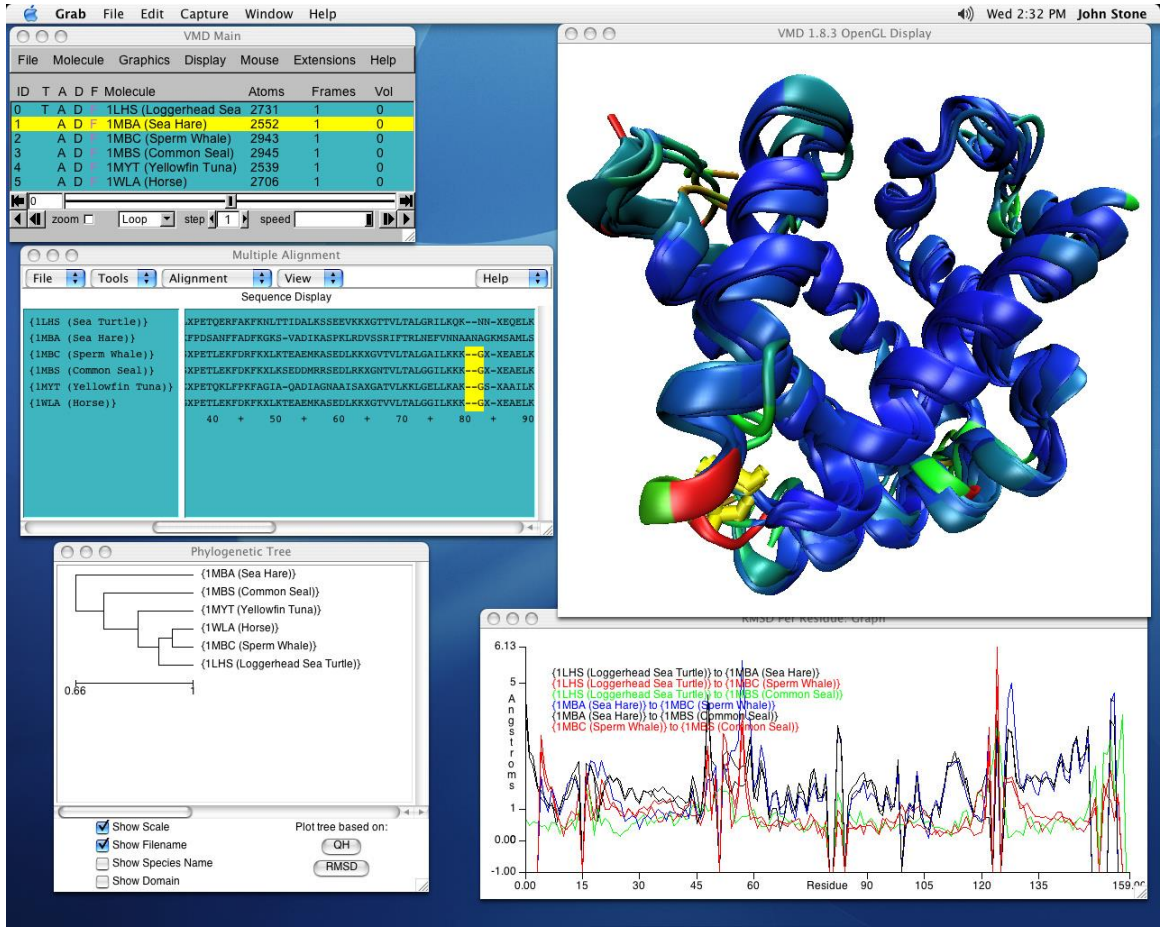

initlat.c
enervib.c
enerbend.c
ener_ev.c
wall_force.c
totenerg.c
process direction
link toward
data flow direction
boundary.tcl
input.dat
init.c
euler.c
ev_nlist.c
main.c
neighbor list update
initialize
allocate memory
simulation
measurement
release memory
header.h
util.c
fileoutput.c
warning_message.dat
config.xyz
mtwist.c/h
Makefile
total_energy.dat
Rg.dat
run_time.dat
randistrs.c/h

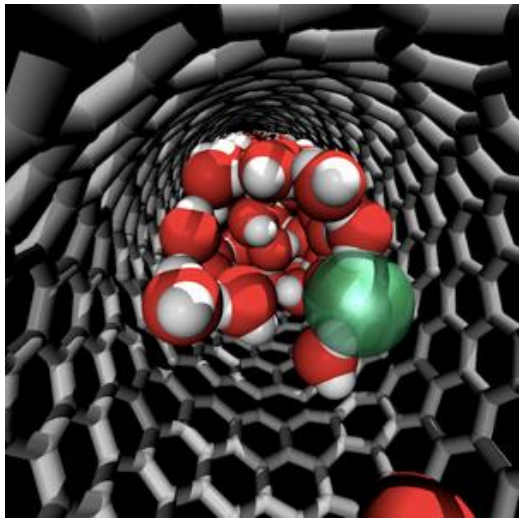
Visualizing Results

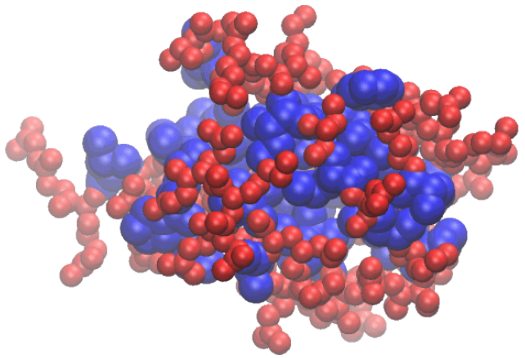
- vmd config.xyz –e boundary.tcl

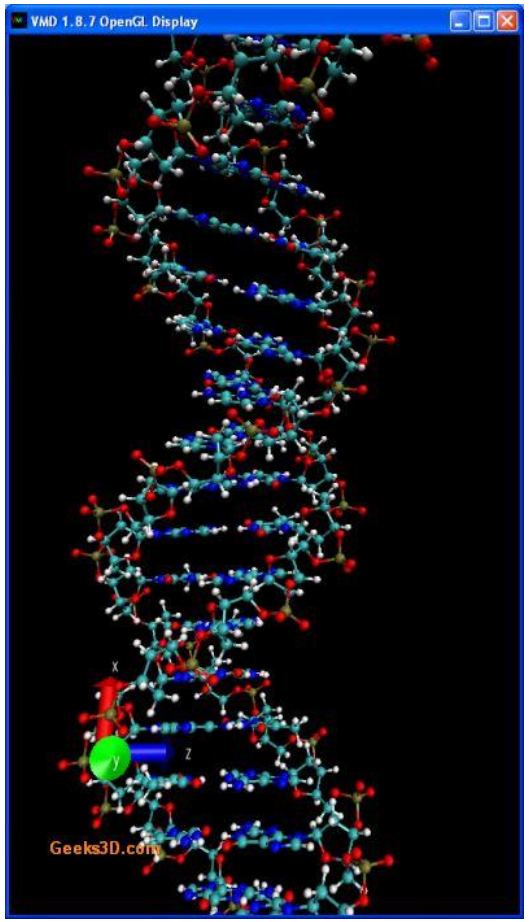
- VMD: Visual Molecular Dynamics
  - Viewing and analyzing the results of molecular dynamics simulations.
  - VMD includes embedded Tcl and Python interpreters.
  - VMD is free

VMD


Grab File Edit Capture Window Help
Wed 2:32 PM John Stone
VMD Main
File Molecule Graphics Display Mouse Extensions Help
ID T A D F Molecule Atoms Frames Vol
0 A D 1LHS (Loggerhead Sea 2731 1 0
1 A D 1MBA (Sea Hare) 2552 1 0
2 A D 1MBC (Sperm Whale) 2943 1 0
3 A D 1MBS (Common Seal) 2945 1 0
4 A D 1MYT (Yellowfin Tuna) 2539 1 0
5 A D 1WLA (Horse) 2706 1 0
zoom Loop step speed
VMD 1.8.3 OpenGL Display
Multiple Alignment
File Tools Alignment View Help
Sequence Display
{1LHS (Sea Turtle)}
{1MBA (Sea Hare)}
{1MBC (Sperm Whale)}
{1MBS (Common Seal)}
{1MYT (Yellowfin Tuna)}
{1WLA (Horse)}
40 + 50 + 60 + 70 + 80 + 90
Phylogenetic Tree
{1MBA (Sea Hare)}
{1MBS (Common Seal)}
{1MYT (Yellowfin Tuna)}
{1WLA (Horse)}
{1MBC (Sperm Whale)}
{1LHS (Loggerhead Sea Turtle)}
0.66
Show Scale
Show Filename
Show Species Name
Show Domain
Plot tree based on:
QH
RMSD
{1LHS (Loggerhead Sea Turtle)} to {1MBA (Sea Hare)}
{1LHS (Loggerhead Sea Turtle)} to {1MBC (Sperm Whale)}
{1LHS (Loggerhead Sea Turtle)} to {1MBS (Common Seal)}
{1MBA (Sea Hare)} to {1MBC (Sperm Whale)}
{1MBA (Sea Hare)} to {1MBS (Common Seal)}
{1MBC (Sperm Whale)} to {1MBS (Common Seal)}
Angstroms
6.13
5
1
0.00
-1.00
0.00 15 30 45 60 Residue 90 105 120 135





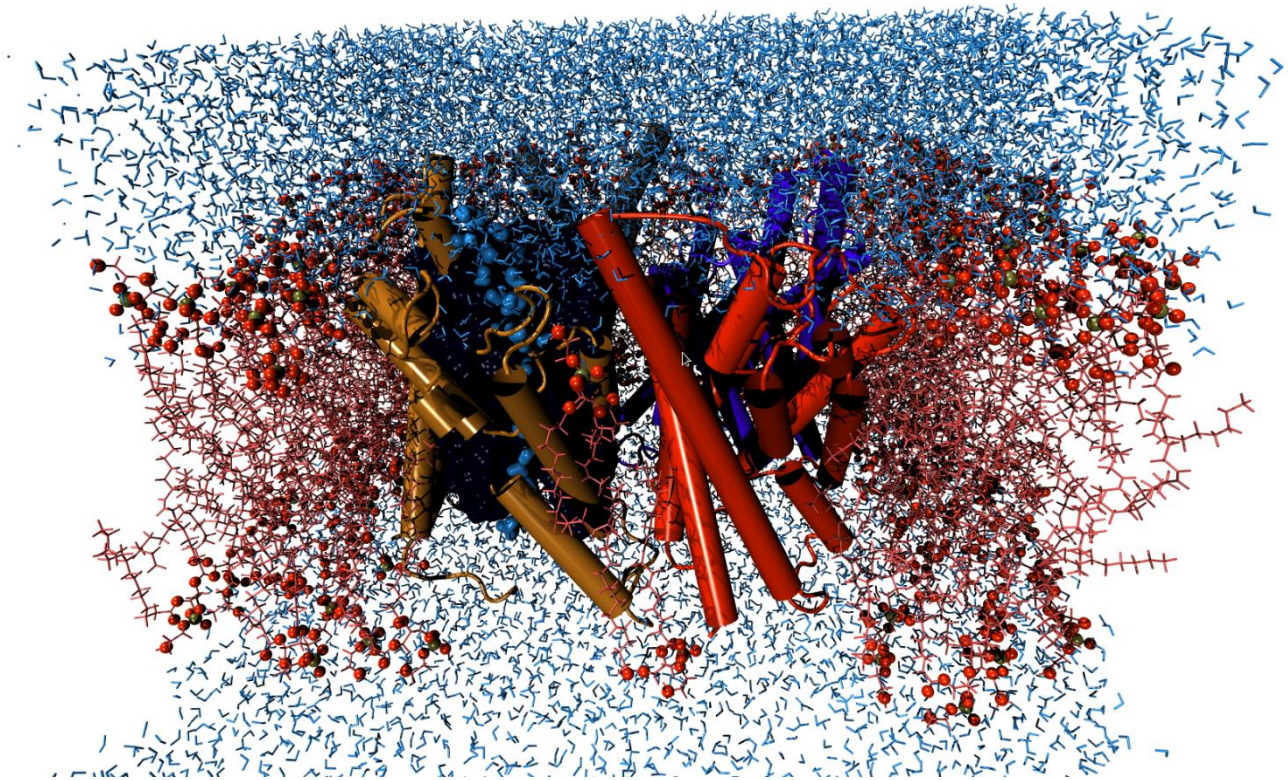

VMD 1.8.7 OpenGL Display
Geeks3D.com

VMD Download and Registration

- For downloading, go to
  - http://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=VMD , or
  - http://www.phys.tw/~b97202017/vmd-1.9.1.bin.LINUXAMD64.opengl.tar.gz
- For installing, follow the steps below
  - cd ~/Download/
  - tar –xzf vmd-1.9.1.bin.LINUXAMD64.opengl.tar.gz
  - cd vmd-1.9.1/
  - ./configure
  - cd src
  - sudo make install
  - vmd

More about VMD

- More tutorial on VMD will be given in future sessions



Any Question?

Additional Note on Nondimensionalization

- The slides for today's session and the additional note can be found on
- https://softphys.wordpress.com/

In future sessions

- VMD (visual molecular dynamics)