* Updated Oct 7, 2009

Development of simple classical molecular dynamics codes

Summer, 2009

Byoungseon Jeon
Department of Applied Science
University of California, Davis

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Contents

- Day 1
  - Basic of Unix/Linux
  - Primitive C/C++ programming
  - Implementation of Makefile
  - Basic configuration of MD
- Day 2
  - Building primitive MD codes
  - Verlet integration/Forcefield/PBC/IC
- Day 3
  - Thermostat (NVT ensemble)
  - Long-range potential

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Pre-requisite

- Unix/Linux command and editors
- Basic programming of C/C++/Fortran
- (Under)graduate mathematics
- Graduate level statistical thermodynamics
  - Ensemble theory: micro-canonical, canonical
  - Natural variables: NVE, NVT, NPT, …
  - Partition theorem
- We will SKIP any theoretical background

古傳 classical MD

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Day 1

- Unix/Linux
  - Very OLD operating system but powerful
- C/C++
  - Multi-purpose programming language
- Makefile
  - Management of MANY source files
- Basic molecular dynamics (MD) code
  - Data structure
  - Unit-conversion
  - Verlet integration

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Beginning

Unix/Linux, C/C++ programming,
compiler, Makefile

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Language and compilers

- ANSI C/C++
  - C++ is a super-set of C
  - We will use C mostly, and some features of C++
  - gcc/g++
- Procedural programming
  - Why not OOP?
  - Modular structure

Sample code 1

- Hello world
  - What is header files?
  - How to compile/link?
  - How to make executables?

```
[sif:WEB/lecture/md_1] bjeon% more hello.c
#include <stdio.h>

int main()
{
  printf("hello world\n");
  return 0;
}
[sif:WEB/lecture/md_1] bjeon% gcc -o a.out hello.c
[sif:WEB/lecture/md_1] bjeon% ./a.out
hello world
```

```
# g++ hello.cc
Or
# g++ -c hello.o
# g++ -o a.out hello.o
```

```
[sif:WEB/lecture/md_1] bjeon% more hello.cc
#include <iostream>

int main()
{
  std::cout << "hello world" << std::endl;
  return 0;
}
[sif:WEB/lecture/md_1] bjeon% g++ hello.cc
[sif:WEB/lecture/md_1] bjeon% ./a.out
hello world
```

Interface for human

Interface for machine

Add more object files or library

Compiling and linking

- You have to KNOW how UNIX is working
- g++ hello.c -lm -L/usr/lib -I/usr/include
- Library
  - "-lxxx" means linking to "libxxx.a" at the current directory by default
  - "-L/aaa" can assign the directory /aaa to look for library files
  - The order of library files might be important (-laaa -lbbb ≠ -lbbb -laaa)
- Header file
  - "-I/mmm" means that the compiler will look into /mmm directory to find header files

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Compiling option

- Common options
  - For naming ”-o”, for debugging, “-g
- Optimization
  - Machine/OS/compilers dependent
  - Examples: -O0, -O1,-O2, -O3, -fast, -fast-sse
  - Higher order optimization employs more sophisticated algorithm but doesn’t guarantee better speed always
  - Aggressive optimization may hurt accuracy
  - Find the optimal option for maximum speed per code/machine/compiler, without hurting accuracy
  - 3-5 speed gain is common compared to non-optimization

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Compiling option (cont.)

af90 -m64 -Ofast -speed_math=10 -march=core -xINTEGER
g95 -march=nocona -ffast-math -funroll-loops -O3
gfortran -march=native -ffast-math -funroll-loops -O3
ifort -O3 -fast -ipo -no-prec-div
lf95 --fast -static -x -
f95 -O4 -mismatch_all -ieee=full -Bstatic
pathf90 -mcpu=em64t -Ofast -WOPT:if_conv=0 -OPT:ro=1 -LNO:fu=9:full_unroll_size=7000
pgf95 -Bstatic -V -fastsse -Munroll=n:4 -Mipa=fast,inline -tp p7-64
sunf95 -fast -xtarget=pentium4

* Fortran compiler options: from polyhedron.com

Pointer

- VERY VERY IMPORTANT in C/C++
- Dynamic memory allocation
  - malloc/free in C
  - new/delete in C++
  - Strongly recommend in most of programming
- Can be used for data structure like TREE
- Should be careful when use - mostly the origin of bugs you make

Dynamic memory allocation

- C style

```
[sif:WEB/lecture/md_1] bjeon% more dyn.c
#include <stdlib.h>

int main()
{
  double *xx;
  xx = (double *) malloc(sizeof(double)*3);
  xx[0] = xx[1] = xx[2] = 0.0;
  free(xx);
  return 0;
}
[sif:WEB/lecture/md_1] bjeon% gcc dyn.c
```

- C++ style

```
[sif:WEB/lecture/md_1] bjeon% more dyn.cc
int main()
{
  double *xx;
  xx = new double [3];
  xx[0] = xx[1] = xx[2] = 0.0;
  delete(xx);
  return 0;
}
[sif:WEB/lecture/md_1] bjeon% g++ dyn.cc
```

Sample code 2

- Practice multiple source files
- How to pass data between routines?
  - Call by value or call by reference
- Split the source file into 3 pieces

```
# g++ -c math_1.cc
# g++ -c math_2.cc
# g++ -c math_3.cc
# g++ -o a.out math_1.o math_2.o math_3.o
```

```
#include <iostream>
#include <cmath>

void calc_sin_1(double x, double y)
{
   y = sin(x);
}

void calc_sin_2(double x, double &y)
{
   y = sin(x);
}

int main()
{
  double x = 3.1415, y = 10.;
  calc_sin_1(x,y);
  std::cout << x << " " << y << std::endl;
  calc_sin_2(x,y);
  std::cout << x << " " << y << std::endl;
  return 0;
}

[sif:WEB/lecture/md_1] bjeon% g++ math.cc
[sif:WEB/lecture/md_1] bjeon% ./a.out
3.1415 10
3.1415 9.26536e-05
```

古傳 classical MD

UCDAVIS DEPARTMENT OF APPLIED SCIENCE

Makefile

- Make “what”?
- Grammar of Makefile

```
# example for foo
.SUFFIXES: .o .cc
CXX = g++
FLAGS = -O3 -ansi
LIB = -lm
INCLUDE =
OBJT = math_1.o math_2.o math_3.o
TARG = b.out
${TARG}: ${OBJT}
        ${CXX} ${FLAGS} -o ${TARG} ${OBJT} ${LIB}
.cc.o:
        ${CXX} ${FLAGS} -c $<
clean:
        rm ${OBJT} ${TARG}
[sif:WEB/lecture/md_1] bjeon% make
g++ -O3 -ansi -c math_1.cc
g++ -O3 -ansi -c math_2.cc
g++ -O3 -ansi -c math_3.cc
g++ -O3 -ansi -o b.out math_1.o math_2.o math_3.o -lm
[sif:WEB/lecture/md_1] bjeon% make
make: `b.out' is up to date.
[sif:WEB/lecture/md_1] bjeon%
```

古傳 classical MD

UCDAVIS DEPARTMENT OF APPLIED SCIENCE

Other UNIX/Linux utilities

- grep
  - Find and locate a “word” or “expression” from the files
- gprof
  - Profiler
  - Check the resource usage
- gdb
  - Debugger

Basic MD code

Data structure, unit, Verlet integration

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Data structure

- The basic of code design
- What do we need?
  - Some number of particles will interact (*N*)
  - The potential is usually a function of particle position (*x*)
  - Force is (*f*) a gradient of potential
  - Particles are moving with some velocity (*v*)

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Data structure (cont.)

- Required data
  - Position, velocity, force (or acceleration) of each particle
- How do we handle them?
  - Brute-force

```
void force (double *xx, double *xy, double *xz,
            double *vx, double *vy, double *vz,
            double *fx, double *fy, double *fz, …)
```

    - Not graceful
  - Structured data

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Structured data

- Structure/Class
  - Contain the related data set into a single piece
  - Easy to handle
  - Increase readability

```
typedef struct
{
Double x[3], v[3], f[3];
} particle
particle *q;
```

```
void force (particle *q)
```

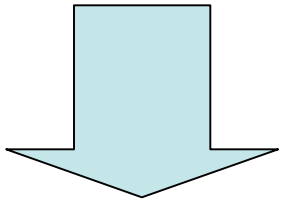
q[14].x[0]:= 1st component of position at 15th particle

q[99].v[2]:= 3rd component of velocity at 100th particle

Structured Data (cont.)

- In FORTRAN95, derived data-set can be used in the same manner

```
call Acceleration(N20, NumberOfTotalNode, NumberOfCell, DisplOfNode, &
     AccelOfNode, &
     MassOfNode, dxPhi4Node, dyPhi4Node, dxPhi4Gauss, dyPhi4Gauss, &
     NumberofDOI, NodeOfDOI, NumberOfGDOI, GaussOfDOI, Jacobian, &
     NumberOfExtForce, NodeExtF, ExtForce, CoordOfNode, &
     NumberOfMasterNode, NumberOfSlaveNode, NodeOfMaster, NodeOfSlave, &
     TimeIncrement, dxPhi, dyPhi, NodeOfDOI2, NumberOfDOI2 )
```



```
CALL ACCEL_FEM(NS, n, e, bulk, dt)
```

古傳 classical MD

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Unit normalization

- Conversion from physical system to numerical system
- You have to be consistent when you make an input and analyze the result
  - Ex: amu for mass, angstrom for length, eV for energy, and 10.1805053 fs for time
  - Units are coupled each other and cannot be given arbitrarily
- Refer Allen & Tildesley's book

古傳 classical MD

UCDAVIS DEPARTMENT OF APPLIED SCIENCE

Verlet integration

- Time integration of kinetics
- BASIC frame of molecular dynamics
- Round-off error is not accumulated extensively
- Position, Velocity Verlet, leap-frog
  - Equivalent each other
  - Velocity Verlet is commonly used in classical MD

古傳 classical MD

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Velocity Verlet

$$x(t+dt) = x(t) + dt \cdot v(t) + \frac{1}{2} dt^2 a(t)$$

$$v(t+dt) = v(t) + \frac{1}{2} dt\left(a(t) + a(t+dt)\right)$$

we don't know yet !

- Velocity update is done by two-fold

x = x + dt*v+0.5*dt*dt*f

v = v + 0.5*dt*f

new f is calculated by new x

v = v + 0.5*dt*f

古傳 classical MD

UCDAVIS DEPARTMENT OF APPLIED SCIENCE

Sample code 3

- Work on sample_3 directory
- Edit verlet.cc
  - Fill the line for velocity Verlet formulation
  - Position and velocity terms
- Do not modify any other files
- Try “make” and make sure there is no error message

```
main()
{
  int i=0;
  double dt = 0.1;
  particle *q;
  q = new particle [N];

  data_read(q);

  do {
    verlet_1(q,dt);
    force(q);
    verlet_2(q,dt);
    i++;
      } while (i<10);
  delete(q);
}
```

古傳 classical MD

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

MD code skeleton

```
main()
{
  int i=0;
  double dt = 0.1;
  particle *q;
  q = new particle [N];

  data_read(q);

  do {
    verlet_1(q,dt);
    force(q);
    verlet_2(q,dt);
    i++;
      } while (i<10);
  delete(q);
}
```

```
void verlet_1(particle *q, double dt)
{
  int i, j;
  for (i=0;i<N;i++) {
    for (j=0;j<3;j++) {
      q[i].x[j] += dt * q[i].v[j] + 0.5*dt*dt*q[i].f[j];
      q[i].v[j] += 0.5*dt*q[i].f[j];
    }
  }
}

void verlet_2(particle *q, double dt)
{
  int i, j;
  for (i=0;i<N;i++) {
    for (j=0;j<3;j++) {
      q[i].v[j] += 0.5*dt*q[i].f[j];
    }
  }
}
```

* You can write a full MD code with less than 100 lines !!!

古傳 classical MD

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Time step

- The most difficult decision which should be made
- Stability issue
  - Numerical integration should be fast enough to catch-up the gradient of the potential
- Mostly, you will decide it by trial and error
  - Reference: 1fs for H(=1amu) at 300K but this is quite rigid
  - Vibrational analysis will require much smaller one

古傳 classical MD

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Conclusion of today

- Master UNIX system and programming language
- Understanding how O/S (UNIX/Linux), H/W are working is very important
- When stuck, GOOGLE
- Experience is the most important
  - Take your time
  - Getting stuck, means you will evolve
- Actual MD coding
  - Not difficult, not long

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Homework assignment

- Potential?
  - What does it describe?
  - Relation to forces?
- Ergodicity?
  - What do you want from MD calculations?
  - Is your MD system representing the actual physical system?

Day 2

- Primitive molecular dynamics (MD) code (cont.)
  - Force-field
    - Making a loop
    - Short-range potential
  - Periodic boundary condition
  - File I/O
  - Nitty gritty
- Post-processing
  - File format: xyz, pdb
  - Visualizer: j-mol, gnuplot, vmd

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Forcefield

- Potential = energy surface describing the system particles
  - Gradient of the potential = bearing force of the corresponding particle
  - Force => acceleration => velocity => position
  - New position updates potential and force
- Paired potential: potential between two particles
- Group potential: many-body interaction like EAM
- External potential: electric/magnetic field

古傳 classical MD

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Forcefield (cont.)

- Lennard-Jones potential
  - One of the short-range pair potential
  - 12/6 potential
  - Power of 6 comes from the interaction of dipole/dipole
  - Power of 12 is a kind of dummy (numerically cheap)

12-6 potential implementation

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

$$-\frac{dv(r)}{dx} = -\frac{dr}{dx}\frac{dv(r)}{dr} = \frac{dr}{dx}4\varepsilon\left\{\left(\frac{\sigma}{r}\right)^{12}\frac{12}{r} - \left(\frac{\sigma}{r}\right)^{6}\frac{6}{r}\right\} = 24\varepsilon\left\{2\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right\}\frac{\Delta x}{r^2}$$

- Practice !
- r = distance b/w two particles
- Δx = distance vector b/w two particles
- If same calculation is repeated, then save it and re-use

古傳 classical MD

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Short-range potential

- Potential is decaying fast
- Only if the particle distance is smaller than a criterion (cut-off radius)
- Complete decaying is not done and smoothing might be required
- Pair interaction loops are required

How to make pair interaction?

- Brute-force

```
for (i=0;i<N;i++){
    for (j=0;j<N;j++) {
        if (i != j) {
            dx_ij = q[i].x - q[j].x;
            q[i].f += …;
            }
    }
}
```

- For *N* particles, $N \times N = N^2$ interactions

- Coupled Loop

```
for (i=0;i<N-1;i++){
    for (j=i+1;j<N;j++) {
        dx_ij = q[i].x - q[j].x;
        q[i].f += …;
        q[j].f -= …;
    }
}
```

- For *N* particles, *N(N-1)/2* interactions, even without conditional statement

古傳 classical MD

UCDAVIS DEPARTMENT OF APPLIED SCIENCE

Periodic Boundary Condition (PBC)

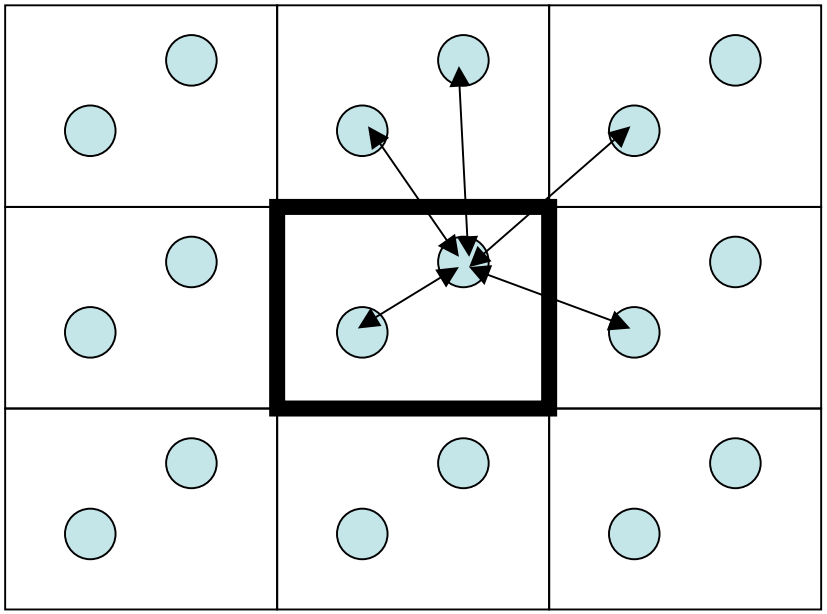
- Facilitate the continuum scale problems with atomistic scale simulations
- Infinitely large system but periodic
  - It means that the particle interaction will be mingled with neighboring IMAGEs
  - Finding the closest particle becomes an issue
  - Simulation box size is coupled with cut-off radii of the potential

古傳 classical MD

UCDAVIS DEPARTMENT OF APPLIED SCIENCE

PBC (cont.)

- Infinite size means infinite number of interactions including self-interaction
- Short-range potential makes it easy
  - Under certain condition, the closest pairs are found in the neighboring images or in itself



古傳 classical MD

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

PBC (cont.)

- Assume that the box size > 2×cut-off radius
  - will exclude self-interaction from neighboring images
- Only neighboring images are tested
- The closest pair is the answer
  - Do we have to test 3×3×3=27 (26 neighbors, 1 simulation cell itself) boxes?
  - Answer is NO: nearest integer operation will save you
- What if the box size < 2×cut-off radius ?

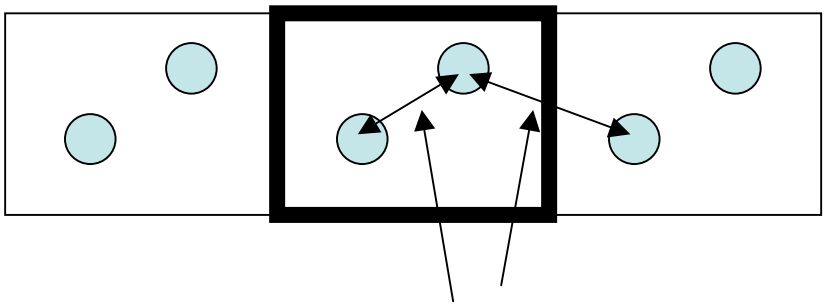
UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Finding the closest pair

- All-pair examination

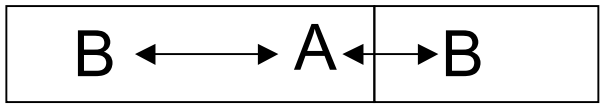
```
for (i=0;i<N-1;i++){
    for (j=i+1;j<N;j++) {
        dx = q[i].x - q[j].x;
        dx = dx - L*nint(x/L);
    }
}
```

nint(0.1) = 0
nint(0.9) = 1
nint(-0.9) = -1
nint(-1.1) = -1



Which one is the closest pair ?

  - nint() operation remaps the pair distance as the closest one



x = -0.7L x = 0.3L
nint(x/L) = -1 nint(x/L)=0
x-L*nint(x/L)=0.3L x-L*nint(x/L)=0.3L

- Why the second closest one is not included in the calculation?

古傳 classical MD

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

nint() function

- FORTRAN provides internal functions such as NINT/ANINT/DNINT
- In C/C++ programming, developers should provide

```
int nint(double x)
{
  int y;
  x > 0.0 ? y=int(floor(x+0.5)): y=int(ceil(x-0.5));
  return y;
}
```

```
double dnint(double x)
{
  double y;
  x > 0.0 ? y=floor(x+0.5): y=ceil(x-0.5);
  return y;
}
```

When do we need PBC truncation?

- Force calculation
  - The closest pair should be found
- Position update
  - If the particle passes the interface of the simulation box, PBC should be applied

Sample code 4

- Implement and fill the routine
  - Verlet and force routine
- Run and check the results
  - Run 10~100 time steps
  - Check the total energy

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Sample code 4 (answer)

```
for(i=0;i<N;i++) for(k=0;k<3;k++) q[i].f[k] = 0.0;

for (i=0;i<N-1;i++) {
  for (j=i+1;j<N;j++) {
    r2 = 0.0;
    for (k=0;k<3;k++) {
      x = q[i].x[k] - q[j].x[k];
      dx[k] = x - l_box * dnint(x/l_box);
      r2 += dx[k]*dx[k];
    }
    sigma_6 = pow(sigma,6.)/pow(r2,3.);
    sigma_12 = sigma_6*sigma_6;
    e_potential += 4.*epsln*(sigma_12 - sigma_6);
    force_term = 24.*epsln*(2.*sigma_12 - sigma_6)/r2;

    for (k=0;k<3;k++) {
      q[i].f[k] += force_term*dx[k];
      q[j].f[k] -= force_term*dx[k];
    }
  }
}
```

```
int i, j;
double x;
for (i=0;i<N;i++) {
  for (j=0;j<3;j++) {
    q[i].x[j] += dt * q[i].v[j] + 0.5*dt*dt*q[i].f[j]/xm;
    q[i].x[j] = q[i].x[j] - l_box*dnint(q[i].x[j]/l_box);
    q[i].v[j] += 0.5*dt*q[i].f[j]/xm;
  }
}
```

古傳 classical MD

UCDAVIS DEPARTMENT OF APPLIED SCIENCE

Initial Condition

- We randomized the initial particle positions but any better configuration?
  - Then what about initial velocity?
- Configuring initial condition might be pain
- Ergodicity
  - Therefore larger systems are favored (?)
- My suggestion
  - From perfect lattice structure or random setup with minimum particle-particle distance
  - Relax them at low temperature for 0.1~1 ps
  - This is REALLY arbitrary
  - Finally it is up to **YOU**

古傳 classical MD

UCDAVIS DEPARTMENT OF APPLIED SCIENCE

File I/O

- Input file
  - Particle data: position, velocity, connectivity, …
  - Particle property: mass, charge, …
  - Potential data
- Output
  - Snapshot/animation of particle motion
  - Temporal trajectories of energy, temperature, pressure, force, …

古傳 classical MD

UCDAVIS DEPARTMENT OF APPLIED SCIENCE

Visualizing atomic data

- Free atomic visualizer
  - jmol/VMD/Mercury …
  - We practice jmol
- File format
  - .xyz: primitive one but easy to use
  - .pdb: standard for biophysics application
  - Tons of file formats to be used

古傳 classical MD

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Practice

- Download jmol
  - uncompress
  - double click “jmol.jar”
- Modify the sample code to produce .xyz format output
- Load *.xyz or *.pdb
- Practice animation with multiple frames

- .xyz format

```
N
Frame = n energy = X
H x y z
O x y z
.
.
.
N
Frame = n+1 energy = Y
H x y z
O x y z
.
.
.
```

How to analyze MD results ?

- Mostly, overall MD view (snapshot) is not useful
- Check how the energy (total, kinetic, potential) and temperature are changing
- Investigate the thermodynamic data statistically
  - Histogram is a good way to study the statistics
- Experience is really important

Nitty gritties

- Programming bugs ?
  - Check if there is any difference w/ -O0, -O1, -O2
  - Low level optimization results should be same to non-optimization run
  - Try different compilers if available
- Logical bug?
  - Who knows? It is you who are responsible
- Correct potential ?
  - Run NVE test up to 10,000 time steps and check the total energy
  - Micro-canonical ensemble should yield a constant total energy

古傳 classical MD

UCDAVIS DEPARTMENT OF APPLIED SCIENCE

Nitty gritties (cont.)

- When crashed
  - Manual book-keeping or debugger
  - Check error messages and google it
  - NaN: Not a Number
  - Check the initialization of variables
  - Experience
- Optimize your code
  - Math. functions are very expensive
  - Reduce any redundant calculations
  - Reduce conditional statement but it is not so expensive
- Make code clear and provide document
  - No one knows (or cares) what you did last summer

Conclusion of today

- Short range potential implementation using all pair-wise interactions
  - Cell-sorting/Verlet list will be required for larger systems
- PBC implementation using nearest integer operations
- You should configure correct IC
- Visualization by Jmol

古傳 classical MD

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Homework Assignment

- Why do we have to do constant temperature simulations?
  - Describe the condition of the actual physical systems
  - What is the TEMPERATURE? If the temperature of MD system is 300K, then do all the particles have 300K of KE?
- If the potential is not short-range, then how will you solve the problem with periodic boundary condition? Or open boundary condition?
  - When do we have such conditions?
  - If you enlarge the simulation box, then will it be enough to handle the long-range interactions?

古傳 classical MD

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Day 3

- Thermostat
  - Fluctuation dissipation theorem
  - How to use random numbers
  - Modify Verlet loop
  - Good temperature?
  - Statistics
- Long-range interactions
  - Ewald sum
  - Particle Mesh Ewald

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Constant temperature simulation

- How the physical system keeps the temperature?
  - Complex physical phenomenon of radiation, conduction, phonon interaction
  - BUT we don't care what they are exactly
  - External effect can be modeled as thermal noise
  - The concept of heat-bath
- Canonical ensemble
  - NVT simulation
  - Choice of thermostat: isokinetic, Berendsen, statistical, Nosé-Hoover, …

古傳 classical MD

UCDAVIS DEPARTMENT OF APPLIED SCIENCE

Thermostat

- Definition of temperature $T = \frac{mv2}{N_{DOF}} = \frac{2KE}{3N_{pt}}$
  - From partition theorem
  - Not unique
- Kinds of thermostat
  - Isokinetic: rescale linearly
  - Statistical: fluctuation dissipation theorem
  - Berendsen: remove noise from the statistical thermostat, leaving global term only
  - Nosé-Hoover: …
  - The choice is up to you

古傳 classical MD

UCDAVIS DEPARTMENT OF APPLIED SCIENCE

Fluctuation dissipation theorem

- Langevin dynamics

$$mx'' + \alpha x' - F = \beta(t) \qquad <\beta(t)>= 0$$

$$<\beta(t)\beta(t')>= 2\alpha k_B T\delta(t-t')$$

- Conclusion
  - Using certain (like Gaussian) random noise, the temporal average reaches the given temperature
- Implementation of thermal noise
  - Modification of Verlet routine
  - Use random number generator

古傳 classical MD

UCDAVIS DEPARTMENT OF APPLIED SCIENCE

Random number generator

- Not REAL random number
  - Pseudo-random number
  - Certain numbers are generated in order
  - You will have same answer unless initialized
  - srand()
- Formulate distribution curve
  - Flat: [0,1)
  - Gaussian
  - Once you design a certain distribution, you have to double-check if the mean/standard distribution match

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Random # generator (cont.)

- Flat distribution

```
double rand_double() // random number generator
{
  double x = ((double) rand())/((double) RAND_MAX);
  return x;
}
```

- Gaussian

```
double normal_dist()
{
  double v1, v2;   double r = 1.0;
  do {
    v1 = 2.*rand_double() - 1.0;
    v2 = 2.*rand_double() - 1.0;
    r = v1*v1 + v2*v2;
  } while (r >= 1.);
  double x = v1*sqrt(-2.*log(r)/r);
  return x;
}
```

古傳 classical MD

UCDAVIS DEPARTMENT OF APPLIED SCIENCE

Modifying Verlet routine

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \mathbf{v}(t)\delta t\Big(1 - \frac{\alpha\delta t}{2m}\Big) + \frac{\delta t^2}{2m}\mathbf{F}(t),$$

$$\mathbf{v}(t+\frac{\delta t}{2}) = \mathbf{v}(t)\Big(1 - \frac{\alpha\delta t}{2m}\Big) + \frac{\delta t}{2m}\mathbf{F}(t),$$

$$\mathbf{F}(t+\delta t) = \mathbf{f}(t+\delta t) + \eta(t+\delta t)\sqrt{2\alpha T_0/\delta t}.$$

from potential

random number w/ Gaussian distribution

$$\mathbf{v}(t+\delta t) = \frac{\mathbf{v}(t+\frac{\delta t}{2}) + \frac{\delta t}{2m}\mathbf{F}(t+\delta t)}{1+\frac{\alpha\delta t}{2m}}.$$

古傳 classical MD

UCDAVIS DEPARTMENT OF APPLIED SCIENCE

Sample code 5

- Work at sample_5
- Fill the empty lines of verlet.cc
- Print the temporal data of temperature
  - Make sure the unit conversion
  - How does it change? Noise?
  - Increase the number of time steps
  - Increase the number of particles

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Sample code 5 (answer)

```
for (i=0;i<N;i++) {
  for (j=0;j<3;j++) {
    q[i].x[j] += dt * q[i].v[j] * (1. - 0.5*alpha*dt/xm)
                 + 0.5*dt*dt*q[i].f[j]/xm;
    q[i].x[j] = q[i].x[j] - l_box*dnint(q[i].x[j]/l_box);
    q[i].v[j] = q[i].v[j]*(1. - 0.5*alpha*dt/xm) + 0.5*dt*q[i].f[j]/xm;
  }
}
```

```
beta = sqrt(2.0*alpha*temperature/dt);
for (i=0;i<N;i++) {
  for (j=0;j<3;j++) {
    eta = normal_dist();
    q[i].f[j] += eta*beta;
    q[i].v[j] = (q[i].v[j] + 0.5*dt*q[i].f[j]/xm)/(1.+0.5*alpha*dt/xm);
    e_kinetic += 0.5*xm*q[i].v[j]*q[i].v[j];
  }
}
```

- What is the (average) temperature?
  - In terms of statistics
  - What is the good thermostat?
  - How can we evaluate the results?

古傳 classical MD

UCDAVIS DEPARTMENT OF APPLIED SCIENCE

Long-range potential

- Coulomb potential between charged particles (mono-pole)
  - Common in biophysics application (water)
  - Cannot be truncated like short-range potential
  - New method/implementation required
  - Very expensive calculation
- In terms of boundary condition
  - Open space: brute-force or TREE
  - PBC: Ewald/Lekner summation

古傳 classical MD

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Numerical implementation of Ewald sum

- Ewald sum: refer Kittel's book
  - Reciprocal term costs NG^3
- Particle-Particle Particle-Mesh (PPPM)
  - LAMMPS
- Particle Mesh Ewald (PME)
  - NAMD

古傳 classical MD

UCDAVIS DEPARTMENT OF APPLIED SCIENCE

Ewald summation

- Effective summation rule of 1/r with PBC
- You may not reach the convergence with brute-force summation
- Split the summation into two different summations
  - Direct/reciprocal sum
  - Different (higher than a single) convergence speed in each space
- Kittel's book
- Ewald sum = Direct + Reciprocal + constant.

古傳 classical MD

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Ewald sum (cont.)

- Direct sum
  - Same strategy of short-range potential (pair interaction)
  - Error function required
  - Cut-off radius is required

$$U_r = \sum_i \sum_{i<j} \sum_{\mathbf{n}} q_i q_j \frac{\operatorname{erfc}(a r_{ij,\mathbf{n}})}{r_{ij,\mathbf{n}}}.$$

- Reciprocal sum
  - In the reciprocal space, structure factor is calculated
  - Linear but cubic of reciprocal vectors

$$U_m = \frac{1}{2\pi V} \sum_{\mathbf{n}} \frac{\exp(-\pi^2 |m|^2 / a^2)}{|m|^2} \left\{ \left[\sum_i q_i \cos(2\pi \mathbf{m} \cdot \mathbf{r}_i) \right]^2 + \left[\sum_i q_i \sin(2\pi \mathbf{m} \cdot \mathbf{r}_i) \right]^2 \right\}.$$

古傳 classical MD

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Why Particle Mesh Ewald?

- Sometimes, reciprocal sum is very expensive
  - Biophysics applications
  - PME does not touch direct sum
- Calculation of structural factors
  - Use of FFT
  - Mapping of the charge on FFT grids

古傳 classical MD

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Conclusion

- You just got started
  - Constraint dynamics, NPT ensemble, radial distribution, velocity correlation, …
  - Try more examples, trial problems, and compare with other codes
- Experience is important

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE

Next Step

- Writing a code for water molecule analysis
  - Everything is there
  - Short/long-range interaction, constraint dynamics (chain molecule), NPT ensemble, …
- Parallel programming
  - Distributed environment: MPI
  - Shared memory: OpenMP, Posix-thread
- Other High Performance Computing
  - GPGPU

古傳 classical MD

UCDAVIS
DEPARTMENT OF APPLIED SCIENCE