Molecular dynamics

Given N particles with initial positions $\underline{r}_i(0)$ & velocities $\dot{\underline{r}}_i(0)$, and the potential energy $U = U(\underline{r}_1 \cdots \underline{r}_N)$

$\rightarrow \sum_{i<j} U_2(\underline{r}_i - \underline{r}_j)$

solve

$$m_i \ddot{\underline{r}}_i = -\frac{\partial}{\partial \underline{r}_i} U(\{\underline{r}_i\}) .$$

Unlike MC, this is the actual physical evolution of the system.

Initial data $\rightarrow$ equilibrated state $\xrightarrow{\text{no force}}$ eg. properties same as MC

$\xrightarrow{\text{force}}$ non-equilibrium behaviour

New issues:

integrate system of ODEs efficiently

control temperature

generate a flow / apply forcing

NB: double precision calculation required

In MC energies are compared so 3 significant figures OK

ODE solutions more sensitive to round off errors

Small configuration changes "continuously".

Bulk of calculation is force evaluation

$\underline{F} = -\nabla U$, $U = \sum_{i<j} U_2(r_{ij})$: $\sum_{}^{N(N-1)} = O(N^2)$ terms

must be computed at every step because all atoms move
(In MC, one atom moves/step, force on it has $O(N)$ terms.)

Want to avoid $O(N^2)$:

long range forces (e.g. Coulomb for ions)

recall $E(R)$ = energy of a charge in sphere of radius R

$\sim R^{3-n}$ if $V \sim r^{-n}$: bad thing

for neutral system, this cancels but only conditional convergence - need fancy summation methods (Ewald)

→ later.

short-range forces:

(1) cut off $\underline{F}$ at r_c, use uniform ($g(r) = 1$) or other approx. for tail

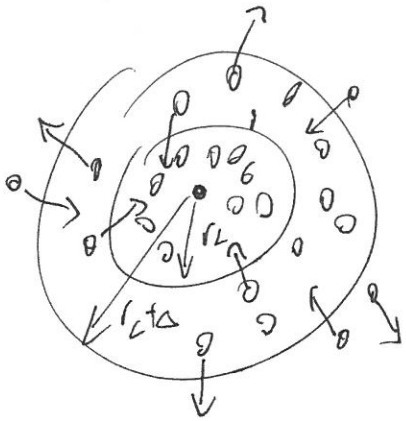
(2) each atom only interacts with "neighbors" at $r < r_c$, and since the integration is in small time-steps, any atom has the same set of neighbors for a while (OK for liquid or solid, may be iffy in a gas)

Verlet list

$$U = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} U_{ij}$$

time step 1: search atoms $\underline{r}_2 \to \underline{r}_N$ for $|\underline{r}_i - \underline{r}_1| < r_c + \Delta$,

r_c = cutoff, Δ = "skin thickness", + put these in a list $(2, 17, 111, \cdots 250, 0 \cdots 0)$.



Idea: if $\Delta r > r_c + \Delta$, the atom can't get to $r_c < \Delta$ until some time (# steps) has elapsed

atoms with $\Delta r < r_c$ do interact, atoms with $r_c < \Delta r < r_c + \Delta$ might or might not interact over some time interval.

Then: search $\underline{r}_3 \to \underline{r}_N$ for atoms with $|\underline{r}_2 - \underline{r}_i| < r_c + \Delta$

add these to the list

search $\underline{r}_4 \to \underline{r}_N$ for atom with $|\underline{r}_3 - \underline{r}_i| < \Delta + r_c$

etc.

→ long list of "candidates" for interaction

steps $1 \to k-1$: test atoms on the list for $\Delta r < r_c$, for these compute $\underline{F}_{ij}$, store in $\underline{F}_i$

step k: recompute the list

⋮

Balance - small Δ ↔ small k: fast search but frequent list recomp.

large Δ ↔ large k: long search but infrequent "

Common choice: for a hybrid { time step = 0.005 (later); $r_c = 2.5$; $\Delta = 1.0$; $k = 10$ or 20

Result: time = $\frac{1}{k} O(N^2)$ + $\frac{k-1}{k} O(N)$
list use list

Coding: 2 arrays: { list = (nbrs of 1, nbrs of 2, ...); nabors = pointer array for list

nabors(i) = where on list does nbrs of i start

r list = $r_L + \Delta$

make the list: $\ell = 0$

```
do i = 1, n-1
   nabors(i) = ℓ+1
   do j = i+1, n
      compute r_ij (minimum image rule)
      if (r_ij < r_list) then
         ℓ = ℓ+1
         list(ℓ) = j
      endif
   enddo
enddo
```

use it:

```
do i = 1, n-1                                   ← no. nbrs. of i
   if (nabors(i) = nabors(i+1)) cycle
   do k = nabors(i), nabors(i+1) - 1
      j = list(k)
      compute r_ij        (min. image)
      if (r_ij < r_c) then
         compute F_ij
         F_i = F_i + F_ij                       ← Newton's 3rd law
         F_j = F_j - F_ij
      endif
   enddo
enddo
```

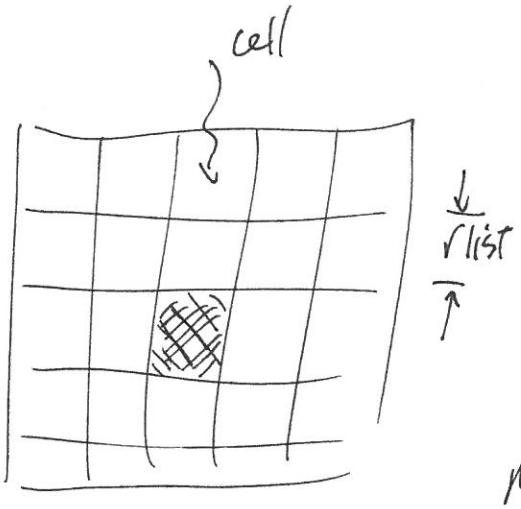
The code used here actually combines the 2 steps, also embeds $g(r)$ computation into it.

—x—

Better method for large systems:

well-separated particles have no chance of getting to $\Delta r < r_c$, so no point to putting them on the list.

→ Divide simulation volume into cells



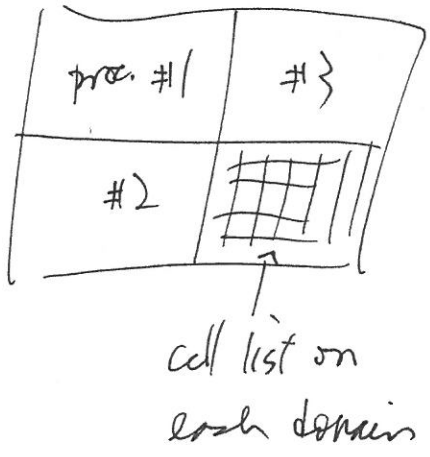
list for each cell only needs input from adjoining cells (in 3d)
box size ~ r list

More overhead in assigning atoms to cells but shorter searches.

Strictly $O(N)$

In practice, LLC beats Verlet for $n >$ few 10^3
not in code here
see A+T Chapter 5, F+S Appendix F

Very large systems ($n \sim 10^6$ or more):
domain decomposition on a parallel computer



one processor ↔ one ~~spt~~ domain
integrates for atoms in its domain

complications:
atoms move btw. processors
" interact with atoms on "next" processor

→ messy MPI code.

Numerical integration : 3N ODEs, "$\ddot{\underline{x}} = f(\underline{x})$"

want a method that's accurate + stable,

↳ $O(\Delta t^3)$ or better

and efficient in the MD context:

1 force evaluation per time step

Euler-Whn: $x(t+\Delta t) = x(t) + v(t)\Delta t + \frac{1}{2} f(t)\Delta t^2 + \frac{1}{3!}\frac{d}{dt}f(x(t))\Delta t^3 + \dots$

inaccurate

messy

: awkward, ? stability

Runge-Kutta: $x(t+\Delta t) = x(t) + \Delta t\left\{\frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4\right\} + O(\Delta t^5)$

$k_1 = \Delta t\, f(x,t)$

$k_2 = \Delta t\, f\left(x(t) + \frac{k_1}{2}, t + \frac{\Delta t}{2}\right)$

etc. : too many force evaluations

Implicit: e.g. Adams-Moulton

$x(t+\Delta t) = x(t) + \frac{\Delta t}{12}\left(5f(x(t+\Delta t)) + 8f(x(t)) - f(x(t-\Delta t))\right) + O(\Delta t^4)$

↑ needs iteration to solve → repeated eval. of f.

Predictor/corrector ("Gear") methods:

start from Taylor series $x(t+\Delta t) = x(t) + \Delta t\,\dot{x}(t) + \frac{\Delta t^2}{2}\ddot{x}(t) + \dots$

$\dot{x}(t+\Delta t) = \dot{x}(t) + \Delta t\,\ddot{x}(t) + \dots$

let $x_0 = x(t)$, $x_1 = \Delta t\,\dot{x}$, $x_2 = \frac{1}{2}\Delta t^2\ddot{x}$ etc

$$\rightarrow \quad x_0(t+\Delta t) = x_0(t) + x_1(t) + x_2(t) + x_3(t) + \dots$$

$$x_1(t+\Delta t) = x_1 + 2x_2 + 3x_3 + \dots$$

$$x_2(t+\Delta t) = x_3(t) + 3x_3 + \dots$$

$$x_3(\cdot) = x_3 + \dots$$

This is the predictor step, depends on old values only

Then use pred positions to evaluate the force, which should be $m\cdot\ddot{x}(t+\Delta t) \rightarrow x_2^{corr}$ which $\neq x_2^{pred}$

$\rightarrow \delta x_2 = x_2^{corr} - x_2^{pred}$

Use this to correct the other derivatives

$$x_n^{corr} = x_n^{pred} + f_{n2}\,\delta x_2 \qquad f_{12} = 1$$

where the f_{n2} are found by choosing the errors up to some order in Δt & retaining stability

If terms upto x_3 are kept: $f_{02} = \frac{1}{120}$, $f_{12} = \frac{3}{4}$ $f_{32} = \frac{1}{12}$

Code uses $x_0 \rightarrow x_5$, coeff in program

Simpler algorithms:

start $x(t+\Delta t) = x(t) + \Delta t\,\dot{x}(t) + \frac{\Delta t^2}{2}\ddot{x}(t) + \frac{\Delta t^3}{3!}\dddot{x}(t) + O(\Delta t^4)$

not advisable to use this directly

also $x(t-\Delta t) = x(t) - \Delta t\,\dot{x} + \frac{\Delta t^2}{2}\ddot{x} - \frac{\Delta t^3}{3!}\dddot{x} + \cdots$

add: $x(t+\Delta t) = 2x(t) - x(t-\Delta t) + \Delta t^2 a(t) + O(\Delta t^4)$,

$a(t) = \ddot{x}(t) = F(x(t))/m$

subtract: $x(t+\Delta t) - x(t-\Delta t) = 2\Delta t\,\dot{x}(t) + \frac{\Delta t^3}{3}\dddot{x}(t) + \cdots$

so $v(t) = \dot{x}(t) = \frac{x(t+\Delta t) - x(t-\Delta t)}{2\Delta t} + O(\Delta t^2)$

Verlet alg: { store $x(t)$ & $x(t-\Delta t)$
compute $a(t) = F(x(t))/m$
" $x(t+\Delta t)$.

actually, can replace $x(t-\Delta t)$ by $x(t+\Delta t)$

so $2 \times 3N$ storage reqd.

velocity not directly needed, but can be found if reqd from x's:

to compute v w/o storing both $x(t \pm \Delta t)$

note $$v\left(t + \frac{\Delta t}{2}\right) = \frac{x(t) - x(t-\Delta t)}{\Delta t} + O(\Delta t^2)$$

If more accurate velocity is needed, note

$$O(\Delta t^2) \leftrightarrow \frac{\Delta t^2}{6}\ddot{a}(t) = \frac{\Delta t}{12}\left[a(t+\Delta t) - a(t-\Delta t)\right] + O(\Delta t^3).$$

but again more storage is req'd.

One advantage of Verlet is time-reversibility, since Δt & $-\Delta t$ enter symmetrically

A (big) disadvantage is that in $x(t+\Delta t)$, $x(t)$ & $x(t-\Delta t)$ are $O(1)$, & $\Delta O(\Delta t^2)$ may be lost in round-off error

To avoid latter — "leap-frog" form of Verlet:

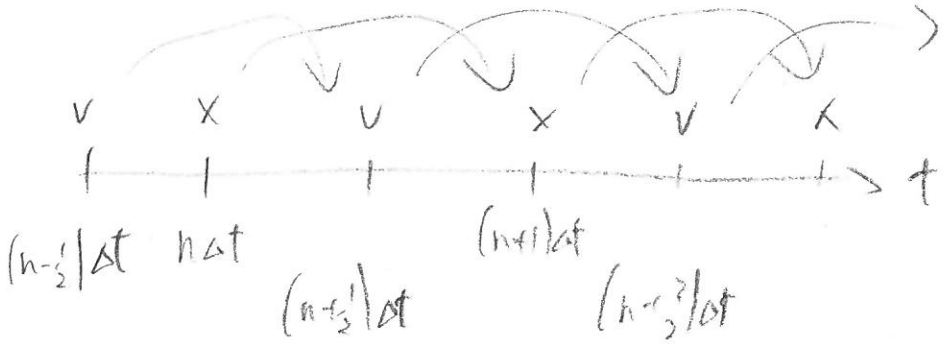
write $$x(t+\Delta t) = x(t) + \Delta t\left[\underbrace{\frac{x(t) - x(t-\Delta t)}{\Delta t}}_{v(t-\Delta t/2)} + \Delta t\, a(t)\right]$$

$\underbrace{\qquad\qquad}_{v(t+\Delta t/2)}$

? this is the "add" eqn above

so $\begin{cases} v(t+\Delta t/2) = v(t-\Delta t/2) + \Delta t\, a(t) \\ \quad \hookrightarrow f(x(t))/m \\ x(t+\Delta t) = x(t) + \Delta t\, v(t+\Delta t/2) \end{cases}$

one comes from



This is algebraically the same as Verlet so
x is still accurate to $O(\Delta t^4)$, but the
momentum is better: $O(1) + O(\Delta t)$.

Storage is the same: $x(t) \;\&\; v(t-\Delta t/2)$

$\longrightarrow x(t+\Delta t) \;\&\; v(t+\Delta t/2)$

; $2 \times 3N$

Velocity $O(\Delta t^2)$ in order of these, which
means $E = \sum \frac{1}{2} m v^2 + U$ only $O(\Delta t^2)$

to do better:

"velocity Verlet" $\begin{cases} x(t+\Delta t) = x(t) + v(t)\Delta t + \frac{1}{2} a(t) \Delta t^2 \\ v(t+\Delta t) = v(t) + \frac{1}{2}\left(a(t) + a(t+\Delta t)\right) \Delta t \end{cases}$

(15)

Can show this is equivalent to usual Verlet for x,

(a) write out v:

(a) $v(t+\Delta t) = v(t) + \Delta t\, a(t) + \frac{\Delta t^2}{2}\dddot{x}(t) + O(\Delta t^3)$

$v(t) = v(t+\Delta t) - \Delta t\, a(t+\Delta t) + \frac{\Delta t^2}{2}\dddot{x}(t+\Delta t) + O(\Delta t^3)$

$= v(t+\Delta t - \Delta t) =$

substract:

$v(t+\Delta t) - v(t) = -\left(v(t+\Delta t) - v(t)\right) + \frac{\Delta t}{2}\left(a(t) + a(t+\Delta t)\right) + O(\Delta t^3)$

$$v(t+\Delta t) = v(t) + \frac{\Delta t}{2}\left(a(t) + a(t+\Delta t)\right) + O(\Delta t^3)$$

(b) VV als sump

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

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

add: $x(t+2\Delta t) + x(t) = 2x(t+\Delta t) + \left(v(t+\Delta t) - v(t)\right)\Delta t + \frac{1}{2}\left(a(t+\Delta t) - a(t)\right)\frac{\Delta t^2}{2}$

then substit. v - explicite

$x(t+2\Delta t) = 2x(t+\Delta t) - x(t) + a(t+\Delta t)\Delta t^2$

which is usual Verlet at $t+\Delta t$

Predictor-corrector vs Velocity Verlet

"older" work
more accurate
→ high T or ρ cases

"recent" work
less storage
reversible → better E conservation

— x —

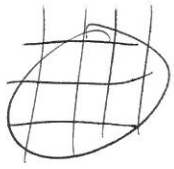
Equilibration

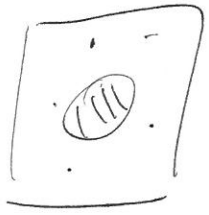
Initial configuration generally not in equilibrium

(1) E conservation fixed KE + PE not not each
→ no control over how much E goes into PE/KE

$\frac{3}{2} N k T = KE$ not fixed

(2) High KE at start → disordered system

e.g. drop in vapor start from  spherical section of lattice

want to get  , may get 

instead start at low T = solid
gradually raise T → the solid drop melts

(or apply external confining force - more programming)

→ Need a "thermostat"

(3) E drifts during simulations due to algorithm or round-off errors

smaller ΔT ⟷ better cons of E ⟷ longer simulation

(4) Fluid flow or external forces do work → heat system.

Could be negligible macroscopically but important in a "small" system like an MD simulation

Ideally T controlled by external environment -

fix T_{wall} - suck out heat

but this only works for small velocity / forcing

—— ×

Simple fix — constant kinetic energy thermostat

```
sum = 0
do i=1,n
  sum = sum + vx(i)^2
        ||    vy
        ||    vz
enddo
scale = sqrt(3NkT/m / sum)
do i=1,n
  vx(i) = vx(i) * scale
    ||      ||
    Σ       ||
enddo
```

fixed $\sum \frac{1}{2} m v^2 = \frac{3}{2} N k T$

In NVT ensemble, can show KE has a Poisson dist

→ wrong ensemble, differences $O(1/N)$

Time unit:

basic units are ε = energy, σ = length, m = mass

dimensional analysis → $\tau = \sigma\sqrt{m/\varepsilon}$ as the unique combination with dimensions of time

Argon parameters for ε, σ, m → $\tau = 2.2$ ps

Time step $\Delta = 0.005\,\tau$

In the code, everything is non-dimensionalized using $\varepsilon, \sigma, m\,(=1)$ and τ.

Physically, if $V_{ij} = 4\varepsilon\left[\left(\frac{r}{\sigma}\right)^{-12} - \left(\frac{r}{\sigma}\right)^{-6}\right]$

the potential minimum is at $r_0 = 2^{1/6}\sigma$

and if V_{ij} is expanded about r_0,

$$V_{ij} \doteq -\varepsilon + \underbrace{\frac{1}{2}V''(r_0)(r-r_0)^2} + \cdots$$

harmonic oscillator potential

the frequency of small oscillations about r_0 is

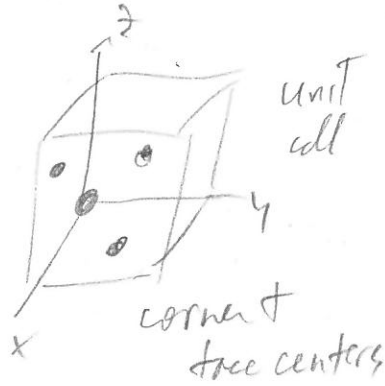
$$\nu = \frac{2\pi}{T} = 2\pi\sqrt{V''(r_0)/m} = \cdots = 1.1\,\tau$$

Schematic code lj-vv.f (pr similar)

declare arrays
state params : # particles
ρ, T, m
steps, output intervals
σ, r_c, r_{list}

positions: x∅, y∅, z∅
velocities: x1, y1, z1

initial positions : fcc lattice



initial velocities

$\underline{p}_{total} = 0$

$KE = \frac{3}{2} N k_B T$

initial force
do kb = 1, max kb
first VV step
new forces
second VV step
correct for periodic
compute pT.E
rescale velocities
enddo
[save pos + vel]

one file