Molecular Dynamics

Molecular Dynamics (MD):

- Technique for computing *equilibrium* and *transport* properties of classical many-body systems
- This is, of course, is an approximation but good for a wide range of materials
- Follows dynamical trajectories dictated by equations of motion

Similar to experiments

Experiments:	MD:
1. Prepare the sample	1. Select a system with N particles
2. Hook up to measuring equipment	2. Equilibrate (eq. of motion)
3. Measure for a period of time	3. Production run (measure)

MD – how does it look like?

MD pseudo code

```
void md(void)

// INITIALIZATION
init()
t = 0

// LOOP OVER TIMESTEP
while (t < tmax)
{
        // CALCULATE FORCES
        force(f, en)
        // INTEGRATE EQS. OF MOTION
        integrate(f, en)
        t = t + delta_t
        // UPDATE AVERAGES
        sample()
}
exit()
```

Main sections:

- *init*
- *force*
- *integrate*
- *sample*

MD – how does it look like?

MD pseudo code

```
void md(void)

// INITIALIZATION
init()
t = 0

// LOOP OVER TIMESTEP
while (t < tmax)
{
        // CALCULATE FORCES
        force(f, en)
        // INTEGRATE EQS. OF MOTION
        integrate(f, en)
        t = t + delta_t
        // UPDATE AVERAGES
        sample()
}
exit()
```

init:

- *Read parameters of the run such as N, T, ρ, dt*
- *Setup initial positions and velocities*

MD – how does it look like?

MD pseudo code

```
void md(void)

// INITIALIZATION
init()
t = 0

// LOOP OVER TIMESTEP
while (t < tmax)
{
        // CALCULATE FORCES
        force(f, en)
        // INTEGRATE EQS. OF MOTION
        integrate(f, en)
        t = t + delta_t
        // UPDATE AVERAGES
        sample()
}
exit()
```

force:

- *"core" step*
- *Modeling comes into play*
- *Non-bonded (short-range)*
- *Coulombic (long-range)*

MD – how does it look like?

MD pseudo code

```
void md(void)

// INITIALIZATION
init()
t = 0

// LOOP OVER TIMESTEP
while (t < tmax)
{
        // CALCULATE FORCES
        force(f, en)
        // INTEGRATE EQS. OF MOTION
        integrate(f, en)
        t = t + delta_t
        // UPDATE AVERAGES
        sample()
}
exit()
```

integrate:

- *"core" step*
- *Numerical integration of equations of motion*

MD – how does it look like?

MD pseudo code

```
void md(void)

// INITIALIZATION
init()
t = 0

// LOOP OVER TIMESTEP
while (t < tmax)
{
        // CALCULATE FORCES
        force(f, en)
        // INTEGRATE EQS. OF MOTION
        integrate(f, en)
        t = t + delta_t
        // UPDATE AVERAGES
        sample()
}
exit()
```

sample:

- *Properties like pressure and temperature are periodically calculated here*

Example

Equations of Motion

Lets start from Hamiltonian Dynamics.

- Because Newton's 2nd law is invariant under time translations

(1) $$\mathbf{F}_i = m\ddot{\mathbf{r}}_i$$

- we expect some function of **r** and **v** whose value is constant in time, the Hamiltonian

(2) $$H(\mathbf{r}^N, \mathbf{p}^N) = const$$

- where **p** is the momentum

For an isolated system, energy (E) is conserved.

(3) $$H(\mathbf{r}^N, \mathbf{p}^N) = \frac{1}{2m}\Sigma_i \mathbf{p}_i^2 + U(\mathbf{r}^N) = E$$

Where U is the interparticle potential.

For equations of motion, first consider the total time derivative

(4) $$\frac{dH}{dt} = \Sigma_i \frac{\partial H}{\partial \mathbf{p}_i} \cdot \dot{\mathbf{p}}_i + \Sigma_i \frac{\partial H}{\partial \mathbf{r}_i} \cdot \dot{\mathbf{r}}_i + \frac{\partial H}{\partial t}$$

That for H independent of time we get

(5) $$\Sigma_i \frac{\partial H}{\partial \mathbf{p}_i} \cdot \dot{\mathbf{p}}_i + \Sigma_i \frac{\partial H}{\partial \mathbf{r}_i} \cdot \dot{\mathbf{r}}_i = 0$$

This is the general result

Put (3) definition of H and get

(6) $$\frac{1}{m}\Sigma_i \mathbf{p}_i \cdot \dot{\mathbf{p}}_i + \Sigma_i \frac{\partial U}{\partial \mathbf{r}_i} \cdot \dot{\mathbf{r}}_i = 0$$

Comparing (5) and (6) get

(7) $$\frac{\partial H}{\partial \mathbf{p}_i} = \frac{\mathbf{p}}{m} = \dot{\mathbf{r}}_i \quad \text{and} \quad \frac{\partial H}{\partial \mathbf{r}_i} = \frac{\partial U}{\partial \mathbf{r}_i}$$

That putting back into (5) get

(8) $$\Sigma_i \dot{\mathbf{r}}_i \cdot \dot{\mathbf{p}}_i + \Sigma_i \frac{\partial H}{\partial \mathbf{r}_i} \cdot \dot{\mathbf{r}}_i = 0$$

$$\Sigma_i (\dot{\mathbf{p}}_i + \frac{\partial H}{\partial \mathbf{r}_i}) \cdot \dot{\mathbf{r}}_i = 0$$

Since the velocities are all independent of one another

(9) $$-\dot{\mathbf{p}}_i = \frac{\partial H}{\partial \mathbf{r}_i}$$

that along with (7)

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

form Hamilton's equation of motion.

Note that these form a 6*N* 1st order differential equations system equivalent to the 3*N* 2nd order differential eq. system from Newton's laws.

Note that from the definition of momentum

$$\mathbf{p}_i = m\dot{\mathbf{r}}_i$$
$$\dot{\mathbf{p}}_i = m\ddot{\mathbf{r}}_i$$

Substitute into (9)

$$\frac{\partial H}{\partial \mathbf{r}_i} = -m\ddot{\mathbf{r}}_i$$

That along with (7) and Newton's 2nd get

(10) $$\mathbf{F}_i = -\frac{\partial H}{\partial \mathbf{r}_i} = -\frac{\partial U}{\partial \mathbf{r}_i}$$

Thus **F** is conservative

Newton vs Hamilton

Note:

- Under Newtonian view, motion is the result of an applied force.
- Under Hamiltonian view, motion results from the equations of motion using *H(**r**,**p**)* (forces are not there explicitly).

Recap of assumptions

Isolated system: if it were not, then there could be energy exchange with surroundings, thus:

- *H* has to contain additional terms to account for those interactions
- *H* is still conserved but would not correspond to the total energy *E*
- *E* is not conserved

Recap of assumptions

- Momentum-velocity relation is taken as

$$\mathbf{p}_i = m\dot{\mathbf{r}}_i$$

- *H* does not have explicit time dependent – a requirement for a constant *H*

Recap of Eq. of motion

- Newton:

$$\mathbf{F}_i = m\ddot{\mathbf{r}}_i$$

- Hamilton:

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

The job of the simulation is to solve them

Remark 1

- A consequence of $$-\dot{\mathbf{p}}_i = \frac{\partial H}{\partial \mathbf{r}_i}$$ is that in some circumstances a particular generalized momentum $\mathbf{p}_i$ might be conserved provided that *H* is independent of the corresponding generalized coordinate $\mathbf{r}_i$

Remark 1

- Define total momentum

$$\boldsymbol{P} = \Sigma_i \, \boldsymbol{p}_i$$

and total angular momentum

$$\boldsymbol{L} = \Sigma_i \, \boldsymbol{r}_i \times \boldsymbol{p}_i = \Sigma_i \, m_i \, \boldsymbol{r}_i \times \dot{\boldsymbol{r}}_i$$

- if *H* is invariant under translation in a particular direction, **p** is conserved in that direction
- If the system is invariant under rotations around an axis, then **L** along that axis is conserved

Remark 1

Examples:

- Infinite system – both are conserved

But because we are dealing with computers...

- Spherical box – L, but not P, is conserved
- Cubic box – P nor L are conserved
- PBC – P, but not L, is conserved (no spherical PBC)

Remark 2

The equations of motion are reversible in time.

- Particles retrace their trajectories if change sign of **p** and **v**
- **IF** we solve the equations of motion correctly, the computer-generated trajectories will (should) also have this property

Remark 3

The explicit spatial derivative of the potential in the equations of motion

$$\mathbf{F}_i = -\frac{\partial U}{\partial \mathbf{r}_i}$$

makes a difference when numerically solving the eq. of motion for a *continuous* vs a *discontinuous* potential.

Remark 3

For continuous function of positions:

- Time-step method based on Taylor expansions of **r**(t)

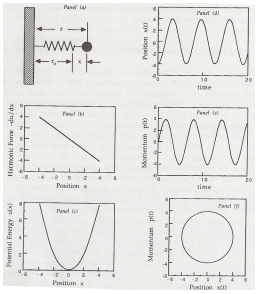
For discontinuous functions (hard spheres, square wells):

- Particle dynamics have to be treated explicitly and separately from the inter-collisional motion

Phase Space

- Goal of simulations is to calculate **r** and **p**, thus to determine the *phase space* trajectory

Example – harmonic spring



- Mass m and potential
  U = $kx^2/2$
- Energy

$$E = \frac{1}{2m}p^2 + \frac{1}{2}kx^2$$

that defines an ellipse in phase space with semiaxes

$(2mE)^{1/2}$ (mom)

and $(2E/k)^{1/2}$ (pos)

Example – harmonic spring

Note that the initial conditions x(0) and p(0) determine the constant value of the energy *E*, and thus the "volume" of the phase space that for an isolated system remains the same.

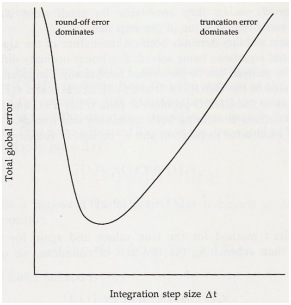
- The volume in phase space determines values for equillibrium macroscopic properties
- which means: <u>externally imposed constraints</u> are connected to macroscopic properties

Sources of error

- On solving the equations of motion we will incur in *truncation* and *round-off* sources of error
- truncation = accuracy with which a finite diff. method approximates the true soln. of the eq. of motion (e.g. in a Taylor expansion)
- round-off = all errors resulting from the implementation of the finite diff. algorithm (e.g. num of significant digits, sq. roots, exp., etc)

Errors

- Errors are related to time step:
  - Truncation error decreases with step size
  - Round-off error increases with # of calculations
- But, usually, dt is determined by trial and error



Finite difference methods

- Consider a potential that is a continuous function of spatial coordinates.
- Can use a finite difference approach where **r**, **v**, and other quantities at time *t* are calculated at *t+dt*
- Accuracy of the calculated properties depends on the size of *dt*
- A good *dt* depends on the method, but in general (t_d time to travel diameter) $dt \ll t_d$

I. Gear predictor-corrector method

The basic algorithm is the following:

1. Predict **r**, **v**, **a**, etc at time *t+dt* using values at *t*
2. Evaluate **a** = **f** / m at new positions
3. Correct predicted coordinates
4. Calculate energy, virial, order parameters, etc before returning to 2.

I. Gear predictor-corrector method

1. Predict coordinates:

   Use Taylor's expansion to predict **r**(*t*+*dt*)

- For an infinitely differentiable function f, Taylor's series expansion around *t*

$$f(t+\delta t) = \sum_{i=0} \frac{f^{(i)}(t)\delta t^i}{i!}$$

I. Gear predictor-corrector method

1. Predict coordinates:

   Expand r(t+dt):

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \frac{d\mathbf{r}(t)}{dt}\delta t + \frac{1}{2}\frac{d^2\mathbf{r}(t)}{dt^2}\delta t^2 + \frac{1}{6}\frac{d^3\mathbf{r}(t)}{dt^3}\delta t^3 + ...$$

   substituting definitions

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

I. Gear predictor-corrector method

1. Predict coordinates:

   Do the same for v(t+dt):

$$\mathbf{v}(t+\delta t) = \mathbf{v}(t) + \frac{d\mathbf{v}(t)}{dt}\delta t + \frac{1}{2}\frac{d^2\mathbf{v}(t)}{dt^2}\delta t^2 + ...$$

   substituting

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

I. Gear predictor-corrector method

1. Predict coordinates:

   Do the same for a(t+dt) and b(t+dt):

$$\mathbf{a}(t+\delta t) = \mathbf{a}(t) + \mathbf{b}(t)\delta t + ...$$

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

I. Gear predictor-corrector method

1. Predict coordinates:

   Summary

$$\begin{aligned}
\mathbf{r}^p(t+\delta t) &= \mathbf{r}(t) + \delta t\ \mathbf{v}(t) + \tfrac{1}{2}\delta t^2\mathbf{a}(t) + \tfrac{1}{6}\delta t^3\mathbf{b}(t) + ... \\
\mathbf{v}^p(t+\delta t) &= \mathbf{v}(t) + \delta t\ \mathbf{a}(t) + \tfrac{1}{2}\delta t^2\mathbf{b}(t) + ... \\
\mathbf{a}^p(t+\delta t) &= \mathbf{a}(t) + \delta t\ \mathbf{b}(t) + ... \\
\mathbf{b}^p(t+\delta t) &= \mathbf{b}(t) + ...
\end{aligned}$$

I. Gear predictor-corrector method

2. Calculate ***a*** at new position $r^p(t+dt)$, using the equations of motion, to obtain:

$$\mathbf{a}^c(t+\delta t)$$

3. Correct predicted coordinates by calculating an error

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

I. Gear predictor-corrector method

3. Correct predicted coordinates by calculating an error – use the error for the correction

$$\begin{aligned}
\mathbf{r}^c(t+\delta t) &= \mathbf{r}^p(t+\delta t) + c_0\Delta\mathbf{a}(t+\delta t)\\
\mathbf{v}^c(t+\delta t) &= \mathbf{v}^p(t+\delta t) + c_1\Delta\mathbf{a}(t+\delta t)\\
\mathbf{a}^c(t+\delta t) &= \mathbf{a}^p(t+\delta t) + c_2\Delta\mathbf{a}(t+\delta t)\\
\mathbf{b}^c(t+\delta t) &= \mathbf{b}^p(t+\delta t) + c_3\Delta\mathbf{a}(t+\delta t)
\end{aligned}$$

- values for c_i are tabulated for best convergence and depend on order of expansion (Gear's suggestion for this level: c_0 = 1/6; c_1 = 5/6; c_2 = 1; c_3 = 1/3)
- corrector step can be iterated (expensive!) for better accuracy.

II. Verlet Method

One of the most popular finite difference methods for MD.

- Start by again expanding using Taylor's expansion:

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

and expanding on the other direction

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

II. Verlet Method

Combining those two equations:

$$\mathbf{r}(t+\delta t) = 2\mathbf{r}(t) - \mathbf{r}(t-\delta t) + \delta t^2\mathbf{a}(t) + O(\delta t^4)$$

Does not need v!

For **v** we have to use

$$\mathbf{v}(t) = \frac{\mathbf{r}(t+\delta t) - \mathbf{r}(t-\delta t)}{2\delta t} + O(\delta t^2)$$

Notes:

- **r** accurate to $O(dt^4)$ while **v** to $O(dt^2)$, not too accurate!
- This algorithm is properly "centered", thus time-reversible
- **r** is updated in one single step, as opposed to previous in two

show in code

III. Velocity Verlet

- Meant to fix awkward velocity estimate from Verlet, and minimize round off error (2 big + 1 small in ***r***)
- Uses both, positions and velocities
- r(t+dt) looks like a Taylor expansion

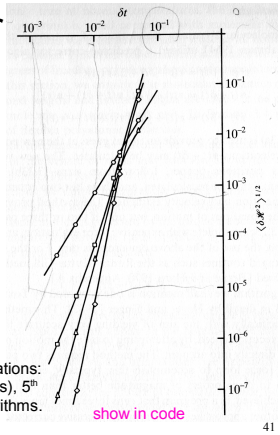
$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \delta t\,\mathbf{v}(t) + \tfrac{1}{2}\delta t^2\mathbf{a}(\mathbf{t})$$

- v(t+dt) only after getting new positions

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

Comparison between Verlet and Gear

- Apparent higher accuracy of Gear (higher order terms) is not reflected for bigger time steps – thus, Gear is not much better than the low order Verlet
- Also, high density of liquids make the 'predictor' step unreliable



Root mean square energy fluctuations: Circles – Verlet, Gear 4th (squares), 5th (trigs), 6th (diamonds) order algorithms.

show in code

Which one to choose?

- If the higher accuracy of Gear is not reflected in high-dense systems, then why not Verlet with a bigger *dt*?
  - Some applications benefit more from Gear that is more amenable to handle modified 1st and 2nd order equations of motion