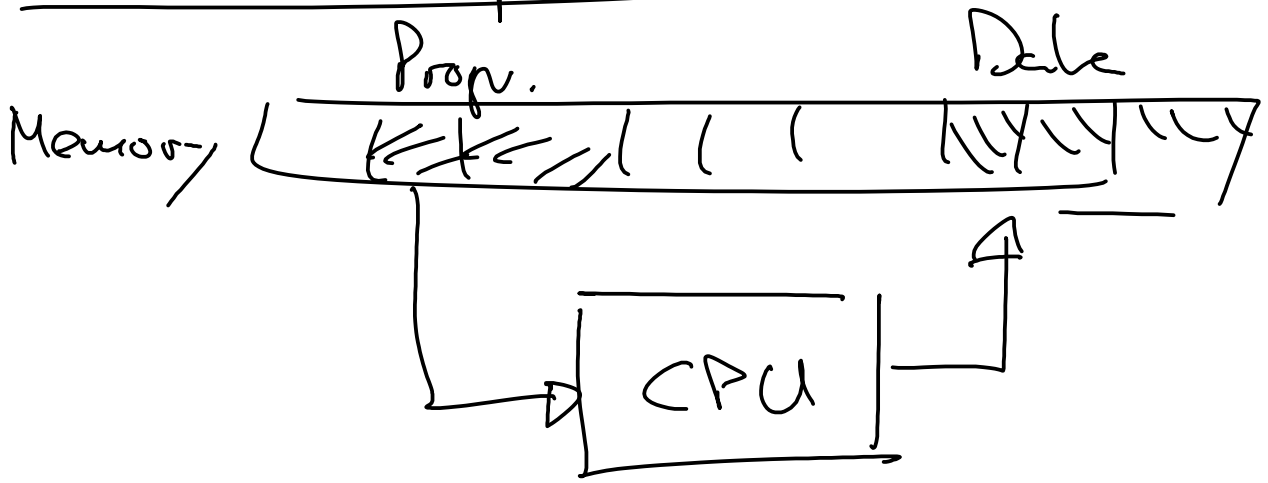
1. Basics :

1.1. Computer :



General structure

Advanced systems :

- Parallel Computers : – Many CPU, Shared or separate memory
- Vector Computers : – Powerful CPU with fast memory access
  – Large registers
- Specialized units : – graphics, floating point, ...

1.2 Programming:

- Execute Case: Machine language
- Higher level languages:

Fortran: - classic language for scientific codes
- simple, clear structure

C: - universal language
- "close" to the machine

C++: - Extension of C including
  - object oriented prog.
  - generic programming
- more complex data structures,
→ good for GUI, Graphics...

Scripting languages: - Perl, Tcl/Tk, Python
→ good for data mining

1.3 Numbers in a Computer:

- **Integer:** n-Bits: $-2^{N-1} \leq x \leq 2^N$

$42: 00101010 = 2^5 + 2^3 + 2^1$

Complement for neg. numb. } $-42: 11010110 = \sim(42) + 1$

Check: $-1 + 1 = 0$

```
  1111 1111
| 0000 0001
-------------
# 0000 0000
```

- **Floating Point Numbers:**

$X = (+/-)\ \text{Mantissa} * \text{Basis}^{\text{Exp.}}$

- Convention: remove "phantom bit" ~~1.0 0 10011 · 2~~

$\pi = 1.1001\,0010\,000\ldots \cdot 2^1$

$42 = 1.0101 \cdot 2^5$

- Standard for memory representation & numerics: IEEE

64-bit double:

1 bit: sign

11 bits: exp.

52 bits: mantissa

π:

0 | 10000000000 | 1001001000 ....

↑ 11 52

$10^{-322} \leq x \leq 10^{308}$

32-bit float : 1 bit sign
8 bits exponent
23 bits mantissa

range: $10^{-44} \leq x \leq 10^{38}$

---

overflow: • number too large:
— "good" : value = "inf"
— "bad" : value = 0, not a number "nan"

underflow : • soft underflow : break phantom-bit convention
• "bad" immediately switch to 0

1.4 Precision :

What is smallest $\varepsilon > 0$ such that $1 + \varepsilon > 1$?

double: $\varepsilon \sim 10^{-16}$

float: $\varepsilon \sim 10^{-7}$

1.5 Tips :

- avoid differences of similar numbers

$$e^{-x} = \sum_{n=0}^{\infty} \frac{1}{n!} (-x)^n \longrightarrow \text{better } \frac{1}{e^x}$$

- reformulate dangerous exp.:

$$\frac{1 - \cos x}{x^2} \text{ for small } x \longrightarrow \frac{1}{2} \left[\frac{\sin \frac{x}{2}}{\frac{x}{2}} \right]^2$$

- start sums from smallest numbe:

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$$

- compensated summation

For $S = a + b$ use

$$t = a + b$$
$$e = (t - a) - b$$
$$s = t - e$$

- use sensible units: don't measure size of an atom in km

2. Classical Many-Particle Systems:

2.1 Introduction:

$$\vec{r} = \{x_1, y_1, z_1, x_2, y_2 z_2, \dots\}$$

$$\ddot{\vec{r}} = \vec{f}(\vec{r}, \dot{\vec{r}}, t)$$

We're interested in $\vec{r}(t)$ for given $\vec{r}(0)$, $\dot{\vec{r}}(0)$

- Numerically more useful: 1st order eqn.

$$\dot{\vec{y}} = \vec{f}(\vec{y}, t)$$

$\vec{r}, \dot{\vec{r}}, t \rightarrow$ new name $\vec{v} \equiv \dot{\vec{r}}$

$$\vec{y} = \{\vec{r}, \vec{v}\} = \{r_1 \dots r_n, v_1 \dots v_n\}$$

$$\dot{\vec{v}} \equiv \ddot{\vec{r}} = f(\vec{r}, \vec{v}, t)$$
$$\dot{\vec{r}} = \vec{v}$$

$$\begin{aligned} \dot{y}_{n+1} &= f_1(\vec{y}, t) \\ &\vdots \\ \dot{y}_{2n} &= f_n(\vec{y}, t) \\ \dot{y}_1 &= y_{n+1} \\ &\vdots \\ \dot{y}_n &= y_{2n} \end{aligned}$$

- Directly: Hamilton theory

$$\dot{p}_i = -\frac{\partial H}{\partial q_i}$$
$$\dot{q}_i = +\frac{\partial H}{\partial p_i}$$

Example: $m\ddot{x} = -4\alpha x^3$

$$\dot{v} = -\frac{4\alpha}{m} x^3$$
$$\dot{x} = v$$

$$\dot{y}_1 = -\frac{4\alpha}{m} y_2 \qquad y_1 \equiv v$$
$$\dot{y}_2 = y_1 \qquad y_2 \equiv x$$

Problem: Solve the eqn.

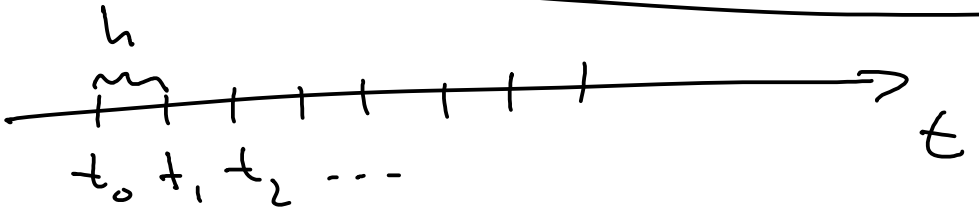
↳ 100 ... 1000 particles → precise alg.

⇒ Runge-Kutta method

↳ many particles (1 mio.) → simple alg.

⇒ Approx. ⇓ Verlet alg.

2.2 Euler Method:



$t_n = t_0 + h \cdot n$

We know $\vec{y}(t_n)$, look for $\vec{y}(t_{n+1})$

$$\frac{d\vec{y}}{dt} = \vec{f}(\vec{y}, t), \quad \frac{\vec{y}(t_{n+1}) - \vec{y}(t_n)}{h} \approx \vec{f}(\vec{y}, t_n)$$

$$\boxed{\vec{y}(t_{n+1}) \approx \vec{y}(t_n) + h\,\vec{f}(\vec{y}, t_n)}$$

Error: $\vec{y}(t_n+h) = \vec{y}(t_n) + h\dot{\vec{y}}(t_n) + \underbrace{\frac{h^2}{2}\ddot{\vec{y}}(t_n)}_{\text{Error}}$

$\longrightarrow$ Error of Euler-Method $\sim h^2$

$\longrightarrow$ Check for $\ddot{x} = -\alpha x$

Higher prec. with same time step h ?

2.3 Runge-Kutta Method:

Notation $\vec{y}(t_n) \equiv \vec{y}_n$

$\frac{y_{n+1} - y_n}{h}$ is approx of $\dot{y}_n$ with error h^2

Better: Consider $\frac{y_{n+1} - y_n}{h}$ as approx for $\dot{y}_{n+\frac{1}{2}} \equiv y(t_n + \frac{h}{2})$

$$y_{n+1} = y_{n+\frac{1}{2}} + \frac{h}{2}\dot{y}_{n+\frac{1}{2}} + \frac{h^2}{8}\ddot{y}_{n+\frac{1}{2}} + \frac{h^3}{48}\dddot{y}_{n+\frac{1}{2}}$$

$$y_n = y_{n+\frac{1}{2}} - \text{"} + \text{"} - \text{"}$$

$$\hookrightarrow \quad \frac{y_{n+1} - y_n}{h} = \dot{y}_{n+\frac{1}{2}} + \frac{h^2}{24}\dddot{y}_{n+\frac{1}{2}}$$

$$\vec{y}_{n+1} = \vec{y}_n + h\,\dot{\vec{y}}_{n+\frac{1}{2}} = \vec{y}_n + h\,\vec{f}(t_{n+\frac{1}{2}}, y_{n+\frac{1}{2}})$$

To get $f(t_{n+\frac{1}{2}}, y_{n+\frac{1}{2}})$ we use Euler:

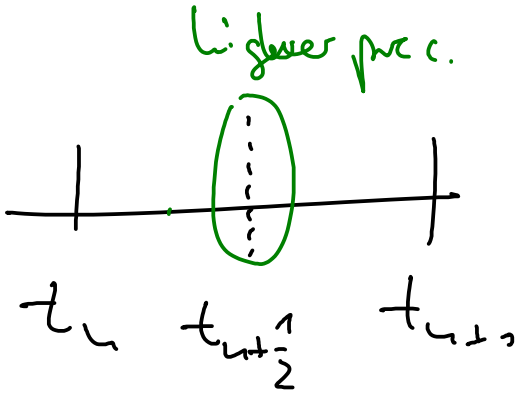
$$\boxed{\vec{y}_{n+1} = \vec{y}_n + h\,\vec{f}\left[t_n + \frac{h}{2},\ \vec{y}_n + \frac{h}{2}\vec{f}(t_n, \vec{y}_n)\right]}$$

Runge-Kutta 2nd order :

$$\vec{k}_1 = h\,\vec{f}(t_n, \vec{y}_n)$$

$$\vec{k}_2 = h\,\vec{f}\left(t_n + \frac{h}{2}, \vec{y}_n + \frac{\vec{k}_1}{2}\right)$$

$$\vec{y}_{n+1} = \vec{y}_n + \vec{k}_2 + \underbrace{O(h^3)}_{\text{Error}}$$



RK 4th order

Runge-Kutta 4th order :

$$\vec{k}_1 = h\,\vec{f}(t_n, \vec{y}_n)$$

$$\vec{k}_2 = h\,\vec{f}\left(t_n + \frac{h}{2}, \vec{y}_n + \frac{\vec{k}_1}{2}\right)$$

$$\vec{k}_3 = h\,\vec{f}\left(t_n + \frac{h}{2}, \vec{y}_n + \frac{\vec{k}_2}{2}\right)$$

$$\vec{k}_4 = h\,\vec{f}(t_n + h, \vec{y}_n + \vec{k}_3)$$

$$\vec{y}_{n+1} = \vec{y}_n + \frac{1}{6}\left(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4\right)$$

Error $\sim h^5$

2.4 Some stat. physics :

- Real systems (10^{23} particles) : averages, correlation

Pressure : $\langle p(N, V, T) \rangle$

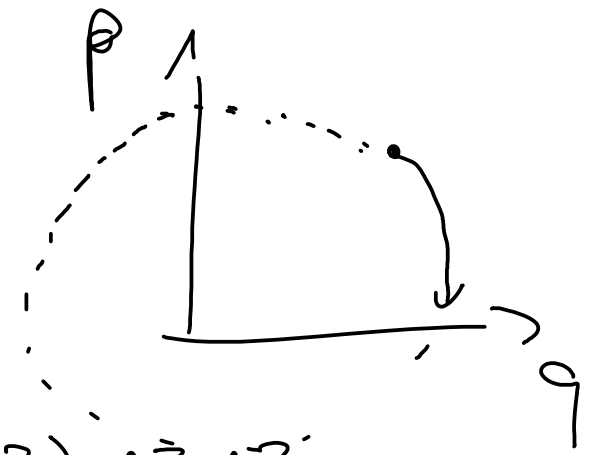
pair correlation function :

$$g(r) = \frac{V}{N^2} \left\langle \sum_{i=1}^{N} \sum_{j \neq i} \delta(r - r_{ij}) \right\rangle ; \quad r_{ij} = |\vec{r}_i - \vec{r}_j|$$

- $\langle \ldots \rangle$ average over <u>ensemble</u> ≡ many copies of physical system

- Eqn. of. motion : Hamilton Function $H(q_i, p_i)$

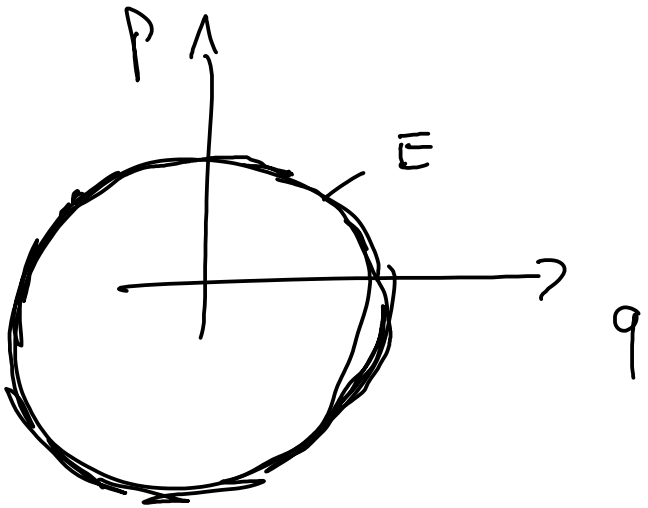
- Ensemble ⟶ Distr. of points in phase space
  → Prob. density $\rho(\vec{p}, \vec{q})$



$$\langle f(\vec{p}\vec{q}) \rangle = \frac{\iint \rho(\vec{p}\vec{q})\, f(\vec{p}\vec{q})\, d\vec{p}\, d\vec{p}}{\iint \rho(\vec{p},\vec{q})\, d\vec{p}\, d\vec{q}}$$

- <u>Micro canonical ensemble</u> :

  – systems with fixed E, N, V



→ $E \hat{=}$ surface in phase space

→ every point on surface has same prob.

$$\rho(E) = \begin{cases} \frac{1}{\Omega(E)} & E \le H(q,p) \le E+\delta E \\ 0 & \text{otherwise} \end{cases}$$

$$\langle f \rangle = \frac{\int \dots \int_{E < H(\vec{q},\vec{p}) \le E+\delta E} f(\vec{q},\vec{p})\, dq_1 \dots dq_n dp_1 \dots dp_n}{\int \dots \int_{E \le H \le E+\delta E} dq_1 \dots dq_n dp_1 \dots dp_n}$$

Ergodic hypothesis:

- a single system will reach every point on energy surface if we "wait long enough"

$$\langle f \rangle_{micro} \triangleq \langle f \rangle_t = \frac{1}{T} \int_0^T f(t)\, dt$$

- Canonical ensemble : — energy exchange with "world"

→ temperature

$$\rho(E) = \frac{e^{-\beta H(\vec{q},\vec{p})}}{Z}$$

$$Z = \int \cdots \int e^{-\beta H(\vec{p},\vec{q})}\, dq_1 \cdots dq_n\, dp_1 \cdots dp_n$$

$$\beta = \frac{1}{k_B T}$$

- Grand canonical ensemble :

  – energy & particle exchange with "world"

  → temperature & chemical potential

- Thermostates : – Nosé - Hoover thermost.

  – Nosé - Poincare thermost.

2.5 Integration methods for many particles :

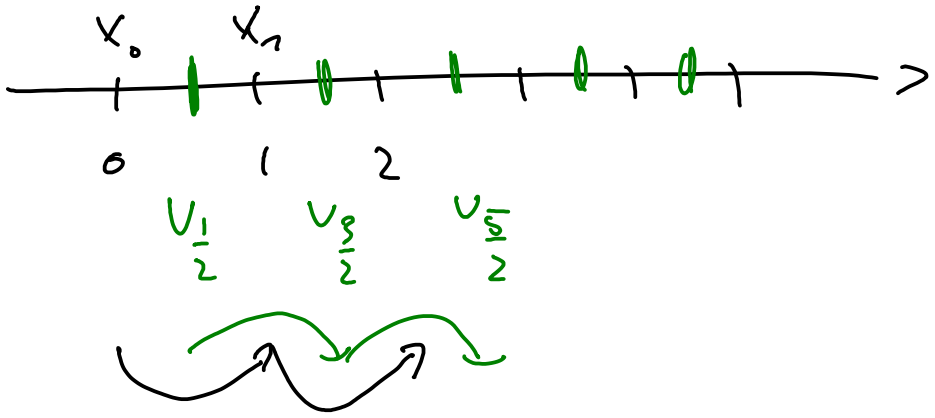
- Runge-Kutta very precise but 4 force calculations in every time step
- need something better

- **leap-frog algorithm** :

$$\dot{x} = v$$
$$\dot{v} = F(x)$$

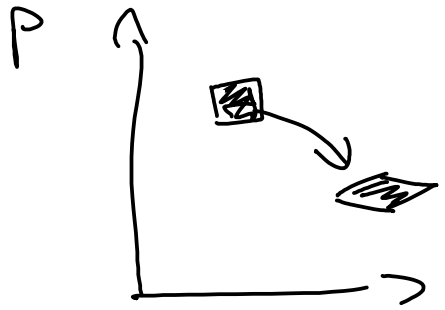
$$\begin{aligned} x_{n+1} &= x_n + h\, v_{n+1/2} \\ v_{n+\frac{3}{2}} &= v_{n+\frac{1}{2}} + h\, F(x_{n+1}) \end{aligned}$$

Feynman lectures Vol I, Ch. 9.6



- **Velocity Verlet-Algorithm** :

$$\begin{aligned} v_{n+\frac{1}{2}} &= v_n + \frac{h}{2} F(x_n) \\ x_{n+1} &= x_n + h\, v_{n+\frac{1}{2}} \\ v_{n+1} &= v_{n+\frac{1}{2}} + \frac{h}{2} F(x_{n+1}) \end{aligned}$$



Hamiltonian dynamics preserves phase space

Liouville's theorem

- Does our algorithm have this property?

$(x_n, v_n) + dx, dv$; $\begin{pmatrix} x_n \\ v_n \end{pmatrix} \to \begin{pmatrix} x_n \\ v_{n+\frac{1}{2}} \end{pmatrix} \to \begin{pmatrix} x_{n+1} \\ v_{n+\frac{1}{2}} \end{pmatrix} \to$

$\begin{pmatrix} x_{n+1} \\ v_{n+1} \end{pmatrix}$

$$\begin{pmatrix} dx' \\ dv' \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ \frac{h}{2}F'(x_{n+1}) & 1 \end{pmatrix}\begin{pmatrix} 1 & h \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ \frac{h}{2}F'(x_n) & 1 \end{pmatrix}}\begin{pmatrix} dx \\ dv \end{pmatrix}$$

$\det = 1 \Rightarrow$ Volume of Box $dx \times dv$ is conserved

$\Rightarrow$ symplectic

- Nice property: Energy is not exactly, but instead energy for slightly disturbed Hamiltonian is conserved:

$$\tilde{H} = H_0 + h H_1$$

Harmonic oscillator: $H = \frac{1}{2}(p^2 + q^2)$

$$\begin{pmatrix} q(t) \\ p(t) \end{pmatrix} = \begin{pmatrix} \cos t & \sin t \\ -\sin t & \cos t \end{pmatrix} \begin{pmatrix} q(0) \\ p(0) \end{pmatrix}$$

Euler:

$$\begin{pmatrix} q' \\ p' \end{pmatrix} = \begin{pmatrix} 1 & \tau \\ -\tau & 1 \end{pmatrix} \begin{pmatrix} q(0) \\ p(0) \end{pmatrix}$$

$$(p'^2 + q'^2) = (1 + \tau^2)(q^2 + p^2)$$

RK4:

$$(q'^2 + p'^2) = \left(1 - \frac{1}{72}\tau^6 + \dots\right)(q^2 + p^2)$$

Sympl. Euler:

$$x_{n+1} = x_n + h\, v_n$$

$$v_{n+1} = v_n + h\, F(x_{n+1})$$

$$\Rightarrow \begin{pmatrix} q' \\ p' \end{pmatrix} = \begin{pmatrix} 1 & \tau \\ -\tau & 1 - \tau^2 \end{pmatrix} \begin{pmatrix} q \\ p \end{pmatrix} \rightarrow \boxed{\frac{1}{2}(q^2 + p^2) + \frac{\tau}{2} pq = \text{const.}}$$

Summary:

$$\dot{\vec{y}} = \vec{f}(\vec{y}, t)$$

- Euler:

$$\vec{y}(t_{n+1}) = \vec{y}(t_n) + h\vec{f}(\vec{y}(t_n), t_n)$$

symplectic:

$$\vec{y} = \{x, v\} \qquad x_{n+1} = x_n + h v_n$$
$$\vec{f} = \{v, f(x)\} \qquad v_{n+1} = v_n + h f(x_{n+1})$$

- Leap frog (symplectic):

$$x_{n+1} = x_n + h v_{n+\frac{1}{2}}$$
$$v_{n+\frac{3}{2}} = v_{n+\frac{1}{2}} + h f(x_{n+1})$$

- Verlet:

$$v_{n+\frac{1}{2}} = v_n + \frac{h}{2} f(x_n)$$
$$x_{n+1} = x_n + h v_{n+\frac{1}{2}}$$
$$v_{n+1} = v_{n+\frac{1}{2}} + \frac{h}{2} f(x_{n+1})$$

Higher order methods :

Runge Kutta : 2nd order :

$$\vec{k}_1 = h\,\vec{f}(t_n, \vec{y}_n)$$
$$\vec{k}_2 = h\,\vec{f}\left(t_n + \frac{h}{2}, \vec{y}_n + \frac{\vec{k}_1}{2}\right)$$
$$\vec{y}_{n+1} = \vec{y}_n + \vec{k}_2 + O(h^3)$$

4th order

$$\vec{k}_1 = h\,\vec{f}(t_n, \vec{y}_n)$$
$$\vec{k}_2 = h\,\vec{f}\left(t_n + \frac{1}{2}, \vec{y}_n + \frac{\vec{k}_1}{2}\right)$$
$$\vec{k}_3 = h\,\vec{f}\left(t_n + \frac{1}{2}, \vec{y}_n + \frac{\vec{k}_2}{2}\right)$$
$$\vec{k}_4 = h\,\vec{f}(t_n + h, \vec{y}_n + \vec{k}_3)$$
$$\vec{y}_{n+1} = \vec{y}_n + \frac{1}{6}(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4) + O(h^5)$$

Typical potentials:

$$U(\vec{r}_1 \dots \vec{r}_N) = \sum_{i \neq j} U(|\vec{r}_i - \vec{r}_j|)$$

$$U(r) = \frac{1}{r} \qquad \text{(Coulomb, Gravitation)}$$

Effective potentials:

Lennard-Jones: $U(r) \sim \left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}$

Exponential: $U(r) \sim e^{-r/a} - \frac{b}{r^6}$

Chapman-Enskog: $U(r) \sim \frac{\alpha}{r^\gamma} \qquad \gamma > 2$

Yukawa-like: $U(r) \sim e^{-\alpha r}/r$

$$H = H_{ion} + H_{ion-el} + H_{ee}$$

$$H = \sum_i \frac{p_i^2}{2m} + \underline{\underline{V}}(q_i)$$

ab-initio md. dyn. → solve electronic many-particle problem approx., get forces, do MD for ions

→ program packages are available

e.g. abinit in linux

3. Classical Monte Carlo :

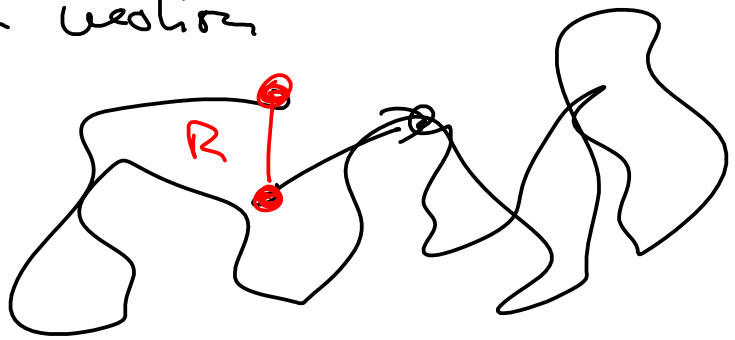
- Monte-Carlo : Methods involving some randomness & statistics

- Applications : – natural : physical systems with randomness
  eg. radioactive decay

$$\frac{\Delta N}{N} = -\lambda \Delta t$$

$\Delta t \to 0$, $N \to \infty$ $\Rightarrow$ exp form
works only for large N

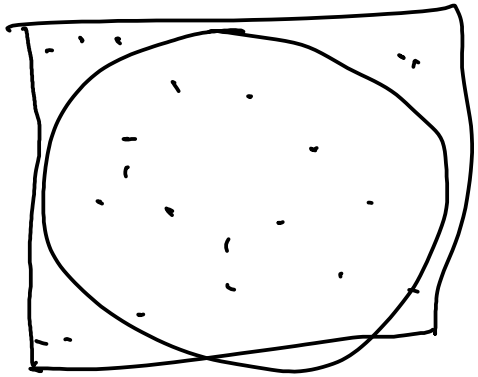
– Brownian motion



$R \sim \sqrt{N}$
$\sim \sqrt{t}$

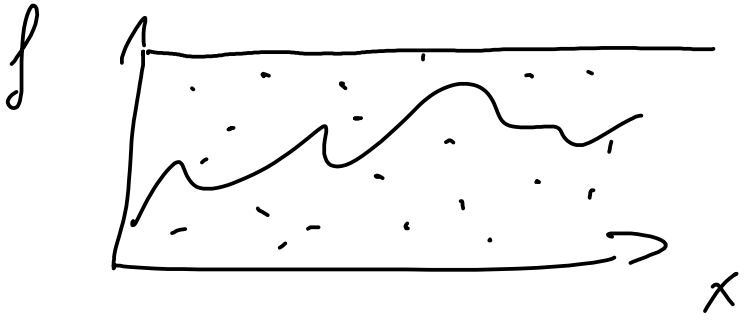
- More important: Integration & Simulation of high dimensional equs. / systems.

Exp: Area of circle:



$$S = \frac{\#\text{ points in circle}}{\text{total }\#} = \frac{\text{area of circle}}{\text{area of square}}$$

More general: $\int \cdots \int f(x_1 \ldots x_n)\, dx_1 \ldots dx_n$



- Stat. phys.:

$$Z = \sum_{i=0}^{N} e^{-\beta E_i}$$

N is very large

4.2 Random numbers:

/dev/random

- Computers usually are deterministic → need 'pseudo' random number generators

- Popular approach:

$$r_{i+1} = (a r_i + c) \bmod m$$

drand48()

$m = 2^{48}$

$a = 0x5DEECE66D$

$c = 0xB$

Toy example : good generator $a = 106$, $c = 1283$.

$m = 6075$

period : 6075

bad generator : $a = 97$, c, m as above

period 4860

$r_1, r_2, r_3, r_4 \ldots$ → look at lattice structure

(bad generator → layers missing)

- Quality measures:

Moments: $\langle x^k \rangle = \frac{1}{N} \sum_{i=1}^{N} x_i^k$ difference $\sim \sqrt{N}$

for uniform random numbers $\langle x^k \rangle = \int_0^1 x^k \, dx = \frac{1}{k+1}$

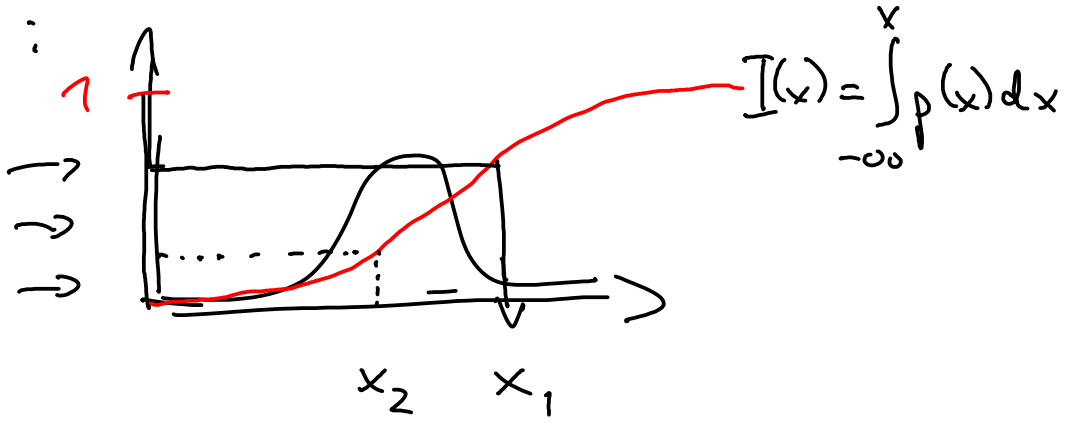
- Autocorrelation: $C(k) = \frac{1}{N}\sum_{i=1}^{N} x_i x_{i+k} \sim \frac{1}{4}$

$x \in [0, 1)$

- Compare simulation with r_i and $1 - r_i$ $\quad r \in [0, 1)$

Non-uniform random numbers:

$dp = p(x)\,dx$



If we can calculate $I(x)$ then

$x = I^{-1}(r)$ where $r \in [0, 1)$ uniform

Eg.

Exponential dist.: $p(x) = \begin{cases} \frac{1}{\lambda} e^{-x/\lambda} & x > 0 \\ 0 & \text{otherwise} \end{cases}$ $\longrightarrow$ $\boxed{x = -\lambda \ln r}$

Gauß-distr.:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-x^2/2\sigma^2} \, dx$$

Trick: go to 2D

$$p(x,y)\,dx\,dy = u(r_1, r_2)\,dr_1\,dr_2$$

with $p(x,y) = u(r_1, r_2) \underbrace{\left| \frac{\partial(r_1, r_2)}{\partial(x,y)} \right|}$

$$=: \frac{\partial r_1}{\partial x}\frac{\partial r_2}{\partial y} - \frac{\partial r_1}{\partial y}\frac{\partial r_2}{\partial x}$$

$$r_1 = e^{-(x^2+y^2)/2}$$

$$r_2 = \frac{1}{2\pi} \arctan \frac{y}{x}$$

Box-Muller:

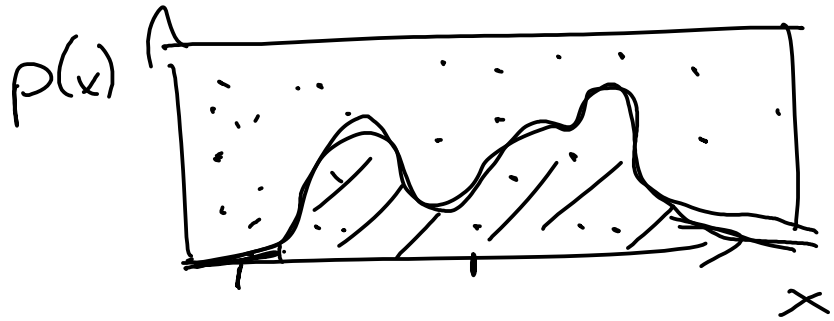
$$x = \sqrt{-2 \ln r_1} \cos 2\pi r_2$$

$$y = \sqrt{-2 \ln r_1} \sin 2\pi r_2$$

with $r_1, r_2 \in [0,1)$ uniform

Then x, y are Gaussian with mean 0 and $\sigma^2 = 1$

more general distr. : von Neumann rejection

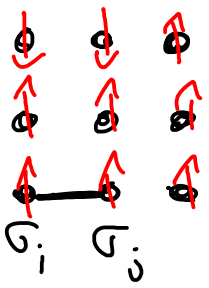


uniform points in rectangle, take x-values of points below $p(x)$ as random variables (with distr. p)

3.4 Some stat. phys.:

Typical model: Ising:

$$H = J \sum_{\langle ij \rangle} \sigma_i \sigma_j + B \sum_i \sigma_i$$



$\sigma_i = \pm 1$

$J < 0$ ferromagnet $J > 0$ antiferrom.

Eigenstates: $\vec{\sigma} = \{\sigma_1 \dots \sigma_N\}$, $E = H(\vec{\sigma})$

number of states: 2^N

eg. 2D system $20 \times 20 \rightarrow 2^{400} = 10^{120}$

$\Downarrow$

# of protons in universe 10^{80}

$$Z = \sum_{n=0}^{2^N} e^{-\beta E_n}$$

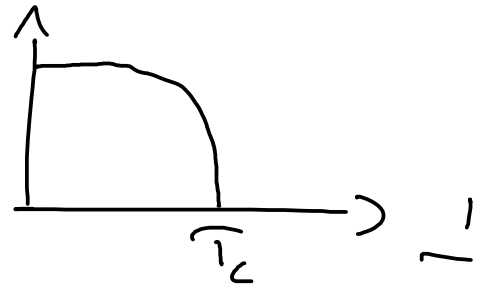
we can never sum all terms

Quantities of interest:

Specific heat: $C = \frac{1}{V}\frac{dE}{dT} = \beta^2 (\langle E^2 \rangle - \langle E \rangle^2)/V$

Magnetisation: $M = \frac{1}{\beta}\frac{d \ln Z}{dB} = \langle \sum_i \sigma_i \rangle$

Susceptibility: $\chi = \frac{\beta}{V}(\langle M^2 \rangle - \langle M \rangle^2)$



Correlations:

$$G(i,j) = G(r_i - r_j) = \langle \sigma_i \sigma_j \rangle - \langle \sigma_i \rangle \langle \sigma_j \rangle$$

($r = r_i - r_j$)

- $|r| \gg 1$ & $T \neq T_c$: $G(r) \sim |r|^{\alpha} e^{-|r|/\xi}$
- $T \to T_c$: ξ diverges : 2nd transition

$$\xi = \xi_{0+} \left| 1 - \frac{T}{T_c} \right|^{-\nu} + \dots \qquad T > T_c$$

$$\xi = \xi_{0-} \left| 1 - \frac{T}{T_c} \right|^{-\nu} + \dots \qquad T < T_c$$

Similar divergences / analytic behaviors for other quantities.

$$C = C_{reg} + C_0 \left| 1 - \frac{T}{T_c} \right|^{-\alpha}$$

$$M = \begin{cases} M_0 \left(1 - \frac{T}{T_c}\right)^{\beta} & T < T_c \\ 0 & T > T_c \end{cases}$$

3.5 Markov-chain Monte-Carlo

- Looking for: $Z = \sum_i e^{-\beta E_i}$ $\quad \langle A \rangle = \frac{1}{Z} \sum_i A_i e^{-\beta E_i}$
- Idea: "importance sampling"
- Markov chain

$$\dots \xrightarrow{W} \{\sigma_i\} \xrightarrow{W} \{\sigma_i'\} \xrightarrow{W} \{\sigma_i''\} \to \dots$$

- Condition: $$P(\{\sigma_i\}) = \frac{e^{-\beta E(\{\sigma_i\})}}{Z}$$

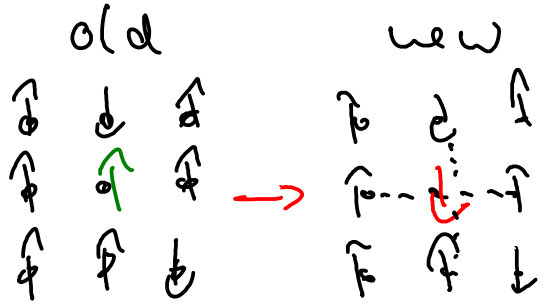
- Detailed Balance

$$P(\{\sigma_i\}) W(\{\sigma_i\} \to \{\sigma_i'\}) = P(\{\sigma_i'\}) W(\{\sigma_i'\} \to \{\sigma_i\})$$

- If we have such an algorithm W then

$$\langle A \rangle = \underbrace{\sum_{\{\sigma_i\}} A(\{\sigma_i\}) P(\{\sigma_i\})}_{\hat{=} \text{ microcan. integrals}} \approx \underbrace{\frac{1}{N} \sum_{i=1}^{N} A(\{\sigma_i\})}_{\hat{=} \text{ "time" average}}$$

- **local updates**
  - change configuration only locally when going from $\{\sigma_i\} \to \{\sigma_i'\}$
  - easy to program
  - problem: convergence is very slow close 2nd order phase transitions

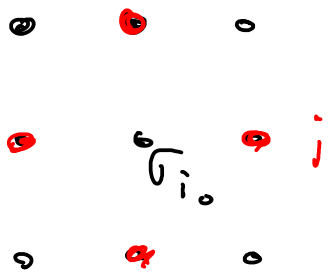


- **Metropolis-algorithm (1953)**

$$W(\{\sigma_i\}_{old} \to \{\sigma_i\}_{new}) = \begin{cases} 1 & E_{new} < E_{old} \\ e^{-\beta(E_{new} - E_{old})} & \text{otherw.} \end{cases}$$

- **Heat bath**

$$W(\{\sigma_i\}_{old} \to \{\sigma_i\}_{new}) = \frac{e^{-\beta \sigma_{i_0}' S_{i_0}}}{\sum_{\sigma_{i_0}} e^{-\beta \sigma_{i_0} S_{i_0}}}$$



$$S_{i_0} = -\sum_j \sigma_j \qquad (j \text{ neighbours } i_0)$$

- **Convergence of local update alg.**

Autocorrelation function

$$A(k) = \frac{\langle O_i O_{i+k} \rangle - \langle O_i \rangle \langle O_i \rangle}{\langle O_i^2 \rangle - \langle O_i \rangle^2}$$

- Ideal situation / algorithm

$$\Delta(k) \xrightarrow{k\to\infty} \propto e^{-k/\tau}$$

- Problem of local-update MC

close to 2nd phase transitions

$$\tau \sim \xi^z \qquad \tau \sim L^z$$

→ "critical slowing down"

local updates : $z = 2$

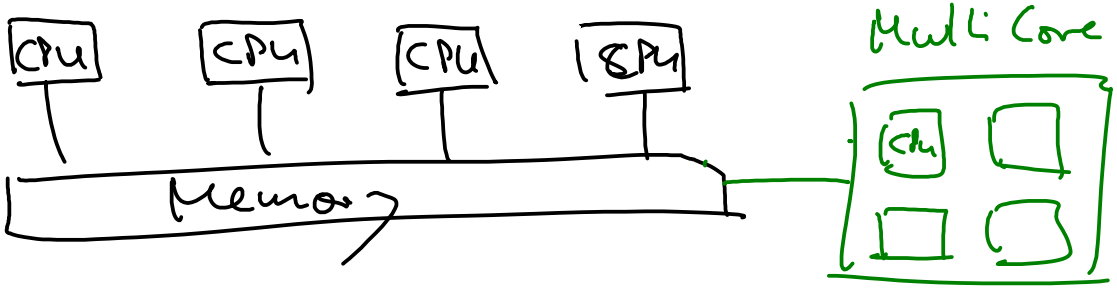
cluster algorithms : $z \geq 0.5$ or $\tau \sim \ln(\xi)$

worm

4. Programming of Parallel Computers

4.1 Typical Architectures:

Symmetric Multiprocessing (SMP)



→ many CPU's share Memory.

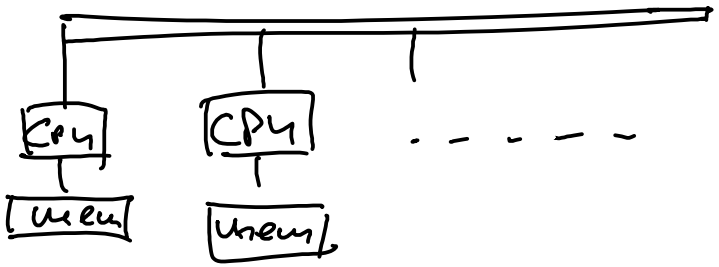
(→ Multi-Core closely related)

→ Adv.: easy to program

→ Disadv.: so far only a few CPU's

8 ... 64

Cluster Computing



→ many "normal" Computers with fast interconnect

Ethernet: 1 Gbit/s

Infiniband } > 10 Gbit/s — bandwidth

Quadrics }

→ low Latency

Ethernet: 30 - 100 μsec

Infiniband: < 5 μsec

Today: Most supercomputers are mixture of both concepts

www.top500.org

Cluster-Comp.

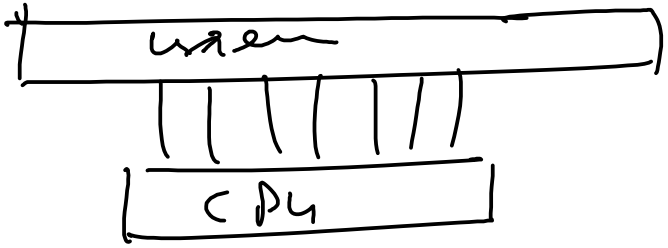
Adv: - Many CPU's

Disadv: - not so easy to program

CPU-mem 64 GByte/s.

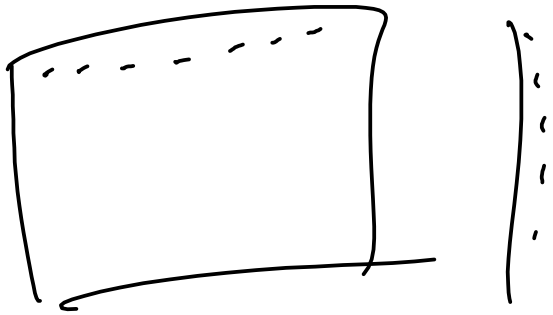
over network 1 GByte/s

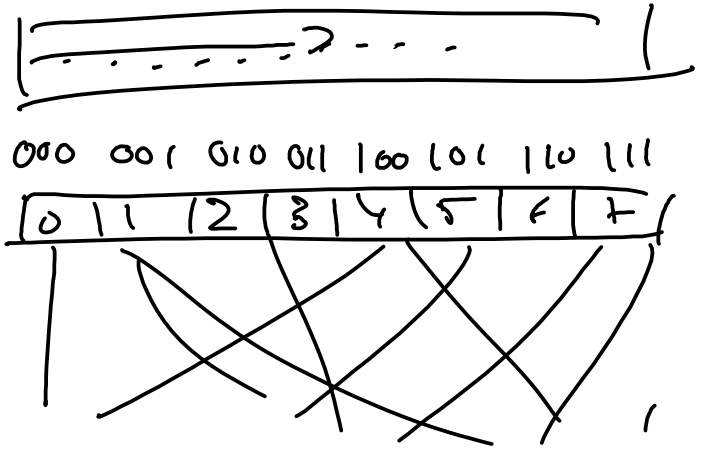
Vector-Computer:



NEC

Cray





SSE (123)

MMX

"pseudo" vectorization

4.2 Programming of SMP machines

- Relatively easy: since all CPU's have fast access to memory
- Mechanism supplied by operating system: threads
  - threads are part of a process
  - Adv.: – starting threads faster than starting processes
    – threads share resources (mem., time)
  - Standard: POSIX threads
  - Tutorial: www.llnl.gov/computing/tutorials/pthreads/
- Disadv: You need to do everything yourself

- OpenMP: technique to make thread-programming easy
  ↳ automatically create thread structures via
  #pragma omp directives

↳ easy to include in existing serial code

↳ Standard: www.openmp.org

↳ Tutorial: www.llnl.gov/computing/tutorials/openMP/

↳ Disadv: works only on SMP machines

4.3. Programming of Clusters:

- Clusters = networked computers
  - → Programmer needs to distribute data & to program communication
- Standard: Message Passing Interface (MPI)
  - ↳ there are many MPI-implementations
  - ↳ Linux: MPICH, LAM/MPI → OpenMPI

↳ Different Versions : - most compilers: MPI 1.2

- more recent MPI 2

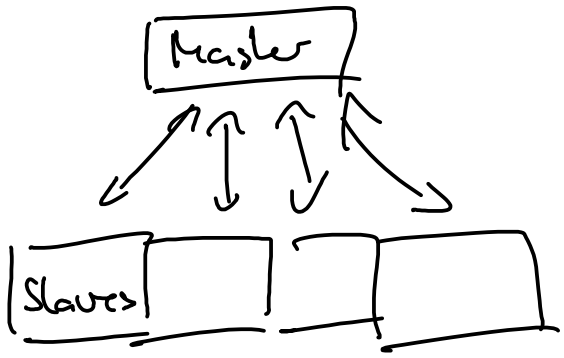
↳ dynamic creation of processes

↳ one-sided communication

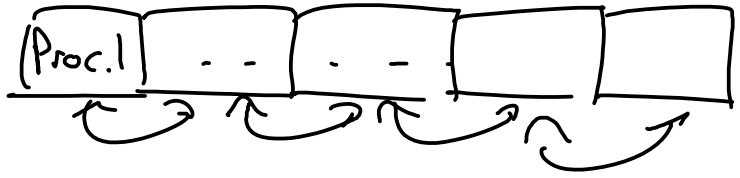
↳ parallel I/O

↳ you can program SMP machines with MPI

- Programming models :



Democracy



- Tutorial: www.llnl.gov/computing/tutorials/mpi/

17. - 20. March 2008

10 a.m., lecture hall

Course on Parallel Programming

see ad. below lecture hall