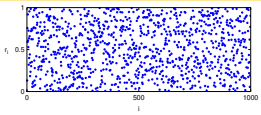
Random numbers in algorithms

- A **deterministic** algorithm is a sequence of operations giving the correct answer (or failing to do so in such a way that we know about the failure).
  Example: matrix inversion by the Gauss–Jordan elimination with full pivoting.
- A **Monte Carlo** algorithm as a procedure using (pseudo)random number to obtain a result, which is correct with certain probability; typically, a numerical result subject to a stochastic error.
  Example: Calculating the internal energy, $\langle E_{\mathrm{kin}}+E_{\mathrm{pot}}\rangle$, in a MD simulation in the NVT ensemble
- A **Las Vegas** algorithm uses random numbers to obtain a deterministic result.
  Example: matrix inversion by the Gauss–Jordan elimination with the pivot element selected at random from several (large enough) pivot candidates.

Example of pseudo random number generator +

$$n_i = 7^5 n_{i-1} \bmod (2^{31}-1), \quad r_i = n_i/2^{31}$$



Monte Carlo integration (naive Monte Carlo)

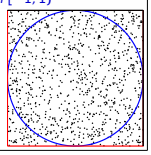
Example: Calculate π by MC integration

```
INTEGER n  total # of points
INTEGER i
INTEGER nu  # of points in a circle
REAL x,y  coordinates of a point in a sphere
REAL rnd(-1,1)  function returning a random number in interval [-1,1)

nu := 0
FOR i := 1 TO n DO
   x := rnd(-1,1)
   y := rnd(-1,1)
   IF x*x+y*y < 1 THEN nu := nu + 1

PRINT "pi=", 4*nu/n  area of square = 4

PRINT "std. error=", 4*sqrt((1-nu/n)*(nu/n)/(n-1))
```



Also “random shooting”. Generally

$$\int_\Omega f(x_1,\dots,x_D)\,\mathrm{d}x_1\dots\mathrm{d}x_D \approx \frac{|\Omega|}{K}\sum_{k=1}^{K} f(x_1^{(k)},\dots,x_D^{(k)})$$

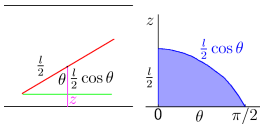
where $(x_1^{(k)},\dots,x_D^{(k)})$ is a random vector from region Ω
($|\Omega|$ = area, volume, ...; calculation of π: $\Omega=(-1,1)^2$, $|\Omega|=4$)

Exercise – Buffon's needle

Let a needle of length l be tossed randomly on a plane with parallel lines d units apart, $l \le d$. The probability that the needle crosses a line is $p = 2l/\pi d$.

[Georges-Louis Leclerc, Comte de Buffon, 1707–1788]

Proof:



expression $(a < b)$ gives 1 if the inequality holds true, 0 otherwise (Iverson bracket)

$$p = \frac{1}{d/2}\int_0^{d/2}\frac{\mathrm{d}z}{\pi/2}\int_0^{\pi/2}\mathrm{d}\theta\left(z<\frac{l}{2}\cos\theta\right) = \frac{1}{d/2}\frac{1}{\pi/2}\int_0^{\pi/2}\frac{l}{2}\cos\theta\,\mathrm{d}\theta = \frac{2l}{\pi d}$$

Usage (δp is the standard error of p)

$$\pi \approx \frac{2l}{pd}, \quad \text{where } p=\frac{n_{\mathrm{crosses}}}{n_{\mathrm{total}}}, \quad \delta p \approx \sqrt{\frac{p(1-p)}{n-1}}, \quad \delta\pi = \frac{2l}{pd}\frac{\delta p}{p}$$

(rel. error)

Credit: animation by Bc. Martin Mašek (2024)

for me: grid: pic/buffon-grid.pdf and buffon.sh

Exercises +

Easy. Calculate by Monte Carlo integration:

$$\int_{x>0,y>0,z>0,x+y+z<1} \frac{1}{|\vec{r}-\vec{r}_0|}\,\mathrm{d}\vec{r}$$

where $\vec{r}_0 = (1,1,1)$

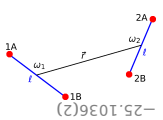
0.125227 28...

Hard. Calculate by Monte Carlo integration the second virial coefficient of the Lennard-Jones diatomics ($\epsilon/k_{\mathrm{B}}T = 1$, $\sigma = 1$) for bond length $L = \sigma$.

$$B_2 = -\frac{1}{2}\int\left[\exp\left(-\frac{u}{k_{\mathrm{B}}T}\right)-1\right]\mathrm{d}\vec{r}\,\frac{\mathrm{d}\omega_1}{4\pi}\frac{\mathrm{d}\omega_2}{4\pi}$$

$$u = u_{\mathrm{LJ}}(|\vec{r}_{1\mathrm{A}}-\vec{r}_{2\mathrm{A}}|)+u_{\mathrm{LJ}}(|\vec{r}_{1\mathrm{A}}-\vec{r}_{2\mathrm{B}}|)+u_{\mathrm{LJ}}(|\vec{r}_{1\mathrm{B}}-\vec{r}_{2\mathrm{A}}|)+u_{\mathrm{LJ}}(|\vec{r}_{1\mathrm{B}}-\vec{r}_{2\mathrm{B}}|)$$

Hints:
– $\mathrm{d}\vec{r} \to 4\pi r^2\mathrm{d}r$
– substitute $r = 1/w - 1$ (MC $\int$ is over $w \in (0,1)$)
– $\mathrm{d}\omega_i = \mathrm{d}\cos\theta_i\,\mathrm{d}\phi_i$ ($\cos\theta_i \in (-1,1)$, $\phi_i \in (0,2\pi)$)



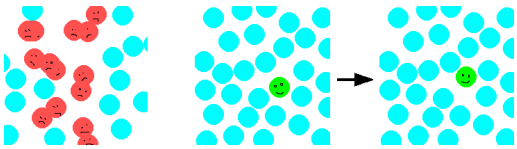
−25.1036(2)

Importance sampling

$$\langle f\rangle \approx \frac{\sum_{k=1}^{K} \mathrm{e}^{-\beta U(\vec{r}_k^N)} f(\vec{r}^N)}{\sum_{k=1}^{K} \mathrm{e}^{-\beta U(\vec{r}_k^N)}} \qquad \vec{r}_k^N = \text{random vector uniformly in the space (naive MC)}$$

$$\langle f\rangle \approx \frac{1}{K}\sum_{k=1}^{K} f(\vec{r}^{N,(k)}) \qquad \vec{r}^{N,(k)} = \text{random vector with a probability} \propto \mathrm{e}^{-\beta U(\vec{r}_k^N)}$$

Metropolis algorithm: $\vec{r}^{N,(k+1)}$ generated sequentially from $\vec{r}^{N,(k)}$



naive MC importance sampling

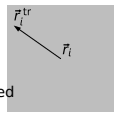
Metropolis method (intuitively)

- Choose a particle, i (e.g., randomly)
- Try to move it, e.g.:

$$\begin{aligned} x_i^{\mathrm{tr}} &= x_i + u_{(-d,d)}, \\ y_i^{\mathrm{tr}} &= y_i + u_{(-d,d)}, \\ z_i^{\mathrm{tr}} &= z_i + u_{(-d,d)} \end{aligned}$$

or in/on sphere, Gaussian,...

  so that the **probability of the reversed move is the same**

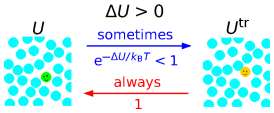


- Calculate the change in the potential energy, $\Delta U = U^{\mathrm{tr}} - U$
- **If** $\Delta U \le 0$, the change is accepted
  If $\Delta U \ge 0$, the change is accepted with probability $\exp(-\beta\Delta U)$, otherwise rejected

Because then it holds for the probability ratio:

$$\text{new : old} = p^{\mathrm{tr}} : p = \exp(-\beta\Delta U)$$

(Moves → and ← are compared: Always the probability of the energy-decreasing move = 1, and of the reversed move = Boltzmann)



A bit of theory: random variables

Random variable $\mathcal{S}$ gives values in $\{A_i\}$, $i = 1,\dots M$, with probabilities $\pi(A_i) = \pi_i$.
Normalization: $\sum_i \pi_i = 1$

Markov chain is a sequence $\mathcal{S}^{(k)}$, $k = 1,\dots,\infty$ such that $\mathcal{S}^{(k+1)}$ depends only on $\mathcal{S}^{(k)}$, or mathematically

$$\pi_j^{(k+1)} = \sum_{i=1}^{M} \pi_i^{(k)} W_{i\to j} \qquad \text{vector notation: } \boldsymbol{\pi}^{(k+1)} = \boldsymbol{\pi}^{(k)}\cdot\mathbf{W}$$

Normalization:

$$\sum_{j=1}^{M} W_{i\to j} = 1 \quad \text{for all } i$$

Example

Computer network: $\begin{cases} 1. & \text{in order} \\ 2. & \text{out of order} \end{cases}$

If in order: will crash with 10% probability
(the following day is out of order)
If out of order: gets fixed with 30% probability
(the following day is in order)

$$W = \begin{pmatrix} 0.9 & 0.1 \\ 0.3 & 0.7 \end{pmatrix}$$

$$\lim_{k\to\infty}\boldsymbol{\pi}^{(k)} = (0.75,\ 0.25)$$

Profit: $\begin{cases} 2000 & \text{in order} \\ 500 & \text{out of order} \end{cases}$

$$X = \begin{pmatrix} 2000 \\ 500 \end{pmatrix}$$

Averaged profit $= \sum \pi_i X_i = \boldsymbol{\pi}\cdot\mathbf{X} = 1625$

for me: xoctave waits 3 s to switch desktop

Detailed balance and microreversibility

We are looking for W, so that $\pi_i = \dfrac{\exp[-\beta U(A_i)]}{\sum_j \exp[-\beta U(A_j)]}$

W = stochastic matrix, transition matrix, probability matrix, Markov matrix...

Conditions:

$$W_{i\to j} \ge 0 \quad \text{for all } i,j = 1,\dots,M$$

$$\sum_{j=1}^{M} W_{i\to j} = 1 \quad \text{for all } i = 1,\dots,M$$

$$\boldsymbol{\pi}\cdot\mathbf{W} = \boldsymbol{\pi} \quad \text{sometimes “detailed balance”}$$

$$\Uparrow$$

$$\pi_i W_{i\to j} = \pi_j W_{j\to i} \quad \text{microscopic reversibility (detailed balance)}$$

If

- all states are accessible from an arbitrary state in a finite number of steps with a nonzero probability and
- no state is periodic

then the set of states is called **ergodic** and for any initial state probability distribution $\boldsymbol{\pi}^{(1)}$ there exists a limit $\boldsymbol{\pi} = \lim_{k\to\infty}\boldsymbol{\pi}^{(k)}$

Metropolis method (more scientifically precise)

One of solutions (Metropolis):

$$W_{i\to j} = \begin{cases} \alpha_{i\to j} & \text{for } i\ne j \text{ a } \pi_j \ge \pi_i \\ \alpha_{i\to j}\dfrac{\pi_j}{\pi_i} & \text{for } i\ne j \text{ a } \pi_j < \pi_i \\ 1 - \sum\limits_{k,k\ne i} W_{i\to k} & \text{for } i = j \end{cases}$$

Equivalent form:

$$W_{i\to j} = \alpha_{i\to j}\min\left\{1, \frac{\pi_j}{\pi_i}\right\} \quad \text{for } i\ne j$$

where matrix $\alpha_{i\to j} = \alpha_{j\to i}$ describes a trial change of a configuration
... equivalent to the algorithm given above

Algorithm – details

- Choose a particle (lattice site, ...) to move
- $A^{\rm tr} := A^{(k)}$ + random move (spin) of the chosen particle
- $\Delta U := U(A^{\rm tr}) - U(A^{(k)}) \equiv U^{\rm tr} - U^{(k)}$
- The configuration is accepted ($A^{(k+1)} := A^{\rm tr}$) with probability $\min\{1, {\rm e}^{-\beta\Delta U}\}$ otherwise rejected:

Version 1	Version 2	Version 3
$u := u_{(0,1)}$ IF $u < \min\{1, {\rm e}^{-\beta\Delta U}\}$ THEN $A^{(k+1)} := A^{\rm tr}$ ELSE $A^{(k+1)} := A^{(k)}$	$u := u_{(0,1)}$ IF $u < {\rm e}^{-\beta\Delta U}$ THEN $A^{(k+1)} := A^{\rm tr}$ ELSE $A^{(k+1)} := A^{(k)}$	IF $\Delta U < 0$ THEN $A^{(k+1)} := A^{\rm tr}$ ELSE $u := u_{(0,1)}$ IF $u < {\rm e}^{-\beta\Delta U}$ THEN $A^{(k+1)} := A^{\rm tr}$ ELSE $A^{(k+1)} := A^{(k)}$

- $k := k + 1$ and again and again

How to choose a particle to move

- In a cycle – check the reversibility!
  Deterring examples of microreversibility violation:
  Three species A, B, C in a ternary mixture moved sequentially in the order of A–B–C–A–B–C– ⋯
  Sequence: move molecule A – move molecule B – change volume – ⋯
- Randomly

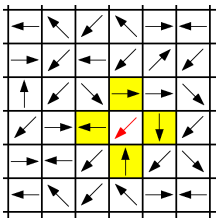
Chaos is better than bad control

Heat-bath method

good for lattice models:

$$W_{i\to j} = \frac{\exp(-\beta U_j)}{\sum_{A_k \in \mathcal{C}_{\rm part}} \exp(-\beta U_k)} \quad \text{pro } A_i, A_j \in \mathcal{C}_{\rm part}$$

- $W_{i\to j}$ does not depend on i
- interpretation: i accept a new value after thermalisation in the actual environment
- (usually one) spin chosen,
  the set of states = $\mathcal{C}_{\rm part}$
- new spin chosen ∝ Boltzmann probability
  which depends on the environment
- all the values of $W_{i\to j}$ needed
  (better, the cumulative distribution function)
  are precalculated in tables for all neighbourhoods

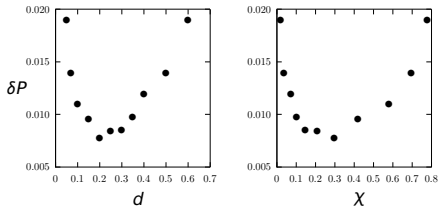


Acceptance ratio

$$\chi = \frac{\text{number of accepted configurations}}{\text{number of all configurations}}$$

χ depends on the displament d. Optimal χ depends on the system, quantity, algorithm. Often **0.3 is a good choice**. Exception: diluted systems...

$\chi = 0.3$



LJ (reduced units): $T = 1.2$, $\rho = 0.8$

Exercise

- Write a computer code for one molecule of nitrogen in a gravitational field. Determine the pressure at the elevation of 8850 m and the acceptance ratio. Pressure at sea level is 1 bar. Assume constant temperature $T = 300$ K.
  – The potential of a molecule is $u(z) = \begin{cases} \infty & \text{for } z < 0 \\ mgz & \text{for } z \geq 0 \end{cases}$, where z is its altitude
  – use the trial displacement of form $z^{\rm tr} = z + \Delta z u_{[-1,1]}$
  – optimum Δz is around 30 km (see below)
  – start from the height $z = 0$ followed by at least 20 steps of "equilibration"
  – perform at least 10000 steps
  – determine the number of cases of a molecule at heights in intervals [0,100) and [8850,8950)
  – pressure is $p_{\rm sea}\frac{\#([8850,8950))}{\#([0,100))}$

  0.377 bar
- Determine the optimum size of the trial displacement Δz and corresponding optimum acceptance ratio χ with respect to quantity "averaged height of a molecule" $\langle z\rangle$. To do this, choose several values of Δz (e.g., 5, 10, 20, 30, 50, 100 km) and calculate $\langle z\rangle$ incluuding the error estimate $\sigma(z)$; e.g., by the block method (e.g., using 100 blocks by 100 MC steps). Plot $\sigma(z)$ as a function of Δz or χ.

  ~ 30 km, $\chi = 0.3$

(Pseudo)random numbers

$$r_i = F(r_{i-1}, r_{i-2}, \ldots, r_{i-m})$$

Requirements:

- the period (smallest number p such that $r_{i+p} = r_i$) is as long as possible;
- distribution r_i is (in an interval given) uniform,
  particularly: also the lowest bits are random;
- (r_i, r_{i+1}), triplets (r_i, r_{i+1}, r_{i+2}), etc., are uncorrelated;
- the same holds for "all" functions f_i: pairs $(f_0(r_i), f_1(r_{i-1}))$, triplets $(f_0(r_i), f_1(r_{i+1}), f_2(r_{i+2}))$, etc., are uncorrelated;
- the code is fast.

History: example of a bad generator by IBM: $K(2^{16}+3, 2^{31})$

Feedback shift-register generators

$$R(A, B, C, \ldots): \quad r_i = r_{i-A} \oplus r_{i-B} \oplus r_{i-C} \oplus \ldots,$$

$\oplus$ = addition modulo 2 = XOR: $0 \oplus 0 = 1 \oplus 1 = 0$, $1 \oplus 0 = 0 \oplus 1 = 1$

Max. period is $2^{\max(A,B,\ldots)} - 1$

A word (32 or 64 bits) at once
E.g., $R(108, 250)$, $R(471, 1586, 6988, 9689)$

Example. $R(5, 2)$:

1 step:
5 4 3 2 1
1 1 0 1 1 0 $1 \oplus 1 = 0$

more steps:
1101100011111001101001000010101 11011...

here period = $2^5 - 1 = 31$ (maximum possible)

Feedback shift-register generators

Algorithm:

```
CONST A=103
CONST B=205
CONST M=255  where M is the smallest number of form 2^k − 1 so that B ≤ M
INTEGER n  unsigned integer
INTEGER r[0..M]  array, filled in advance by random numbers of any origin

one step generating a random number (all bits):
n := n+1
r[n and M] := r[(n-A) and M] xor r[(n-B) and M]
        where and and xor work bitwise
RETURN r[n and M]
```

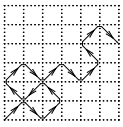
The code is especially simple as a C/C++ macro:

```
#define rnd (++n, r[n&M] = r[(n-A)&M] ^ r[(n-B)&M])
```

Pros: fast, mathematical theory for the perion and correlations

Cons: fails with some tests; e.g., the random walk → → →

Remedy:
– combine two of them (still fast)
– Mersenne twister (high quality, popular)



Congruence generators

$$K(C, M): \quad r_i = C r_{i-1} \bmod M$$

where $A \bmod B$ is a reminder after division A/B

$K(5^7, 2^{32})$: period $2^{32}/8$

$K(7^5, 2^{31}-1)$: period $2^{31} - 2$

Example. $K(5, 31)$:
1 7 18 2 14 5 4 28 10 8 25 20 16 19 9 1 7 18 2 14 5 4 28 10 8 25 20 16 19 9 1 7 18 ...

Suppression of correlations – combine 2 generators

Declare a table and fill it by random numbers using generator #1

- take randomly selected (index = random number using generator #2) item of the table
- replace the "used" number by a new random number using generator #1

Other distributions

Library functions usually give random number $u_{(0,1)}$, uniformly distributed in $(0, 1)$ (or $[0, 1)$ or $[0, 1]$ – be careful!), i.e.,

$$\phi(x) = \begin{cases} 1, & x \in (0,1) \\ 0, & x \notin (0,1) \end{cases}$$

A number uniformly distributed in interval (a, b) is

$$u_{(a,b)} = a + (b-a)u_{(0,1)}.$$

Generally: function $f(u)$ applied to $u_{(0,1)}$ →

$$\phi(y) = \sum_{x, f(x)=y} \frac{1}{|f'(x)|}$$

Inverse problem: known distribution $\phi(x)$, $\int \phi(x){\rm d}x = 1$, needed:
the distribution function $\int_{-\infty}^{y} \phi(x){\rm d}x$ must be inverted.

Example: $x = -\ln u$ gives $\phi(x) = \exp(-x)$ (check for $u = 0$!)

Gauss normal distribution

$$u_{\text{Gauss}} = \sqrt{-2 \ln u_{(0,1)}} \cos(2\pi u_{(0,1)})$$

where both random numbers $u_{(0,1)}$ are independent, function $u_{(0,1)}$ thus has to be called twice. Second indepent number: replace cos → sin.

Approximately:

$$u_{\text{Gauss}} \approx \sqrt{2}(u_{(0,1)} - u_{(0,1)} + u_{(0,1)} - u_{(0,1)} + u_{(0,1)} - u_{(0,1)})$$

General distribution $\phi(x)$ in interval (a, b)

If the method of the previous slide is too complicated:

1. generate $x = u_{(a,b)}$,
2. generate $u = u_{(0,m)}$, where m is the maximum of $\phi(x)$ in interval (a, b),
3. if $u < \phi(x)$, accept x as the number, otherwise repeat by step 1.

Multidimensional distribution

In a unit ball (ball = inside of a sphere)

1. generate $x = u_{(-1,1)}$, $y = u_{(-1,1)}$, $z = u_{(-1,1)}$,
2. calculate $r^2 = x^2 + y^2 + z^2$,
3. if $r^2 < 1$, accept vector (x, y, z), otherwise repeat by step 1.

On a unit sphere: divide $\frac{\vec{r}_{\text{in ball}}}{r}$ (check for $r \approx 0$), or:

1. $z = u_{(-1,1)}$, $\phi = u_{(0,1)}$
2. $x = \sqrt{1-z^2}\sin(2\pi\phi)$, $y = \sqrt{1-z^2}\cos(2\pi\phi)$

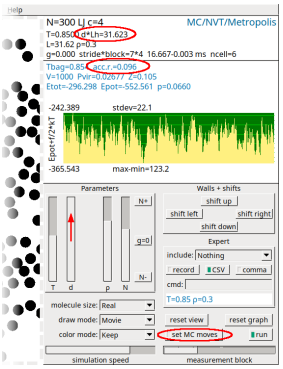
Uniform discrete distribution

$$u_N = \text{int}(N u_{(0,1)})$$

Better not like this (r is a random integer):

$$u_N = r \bmod N$$

(very bad for congruence generators – lower significant bits are not random)

SIMOLANT: Try MC yourself

- Install SIMOLANT (see previous lecture).
- Menu: Method → Monte Carlo NVT (Metropolis)
- Turn the automatic displacement size setup off (set MC move). Slider "d" will appear.
- Using the slider, change the trial displacement d and observe how the acceptance ratio (acc.r.) decreases and increases and how the configurations changes.
- Decrease temperature and/or increase density and repeat. Compare with a MD with a thermostat.
- Menu: Boundary conditions → Periodic, and set the critical temperature and density ($\sim T = 0.85$ and $\rho = 0.3$) and at least $N = 300$ particles. Which displacement size "d" leads to the fastest sampling of density fluctuations?