Random numbers in algorithms

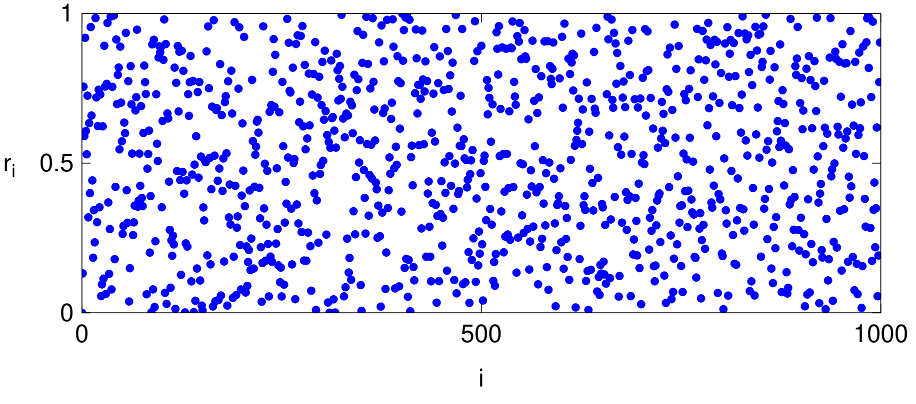
- A **deterministic** algorithm is a sequence of operations giving the correct answer (or failing to do so in such a way that we know about the failure).
  Example: matrix inversion by the Gauss–Jordan elimination with full pivoting.

- 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: Solving the traveling salesman problem by simulated annealing.

- 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)

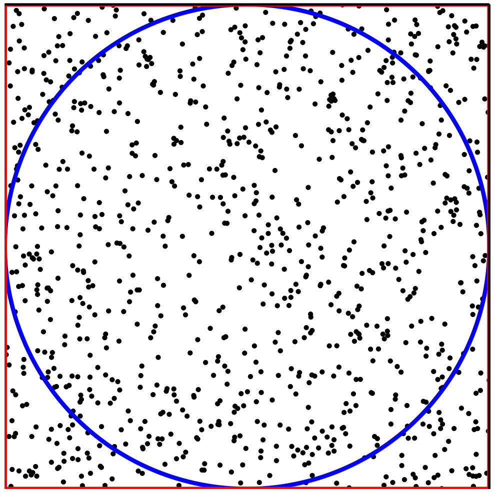
Example: Calculate π by MC integration

```
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))
```



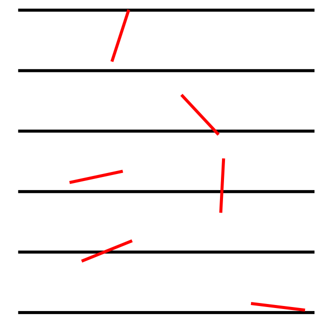
Also "random shooting". Generally

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

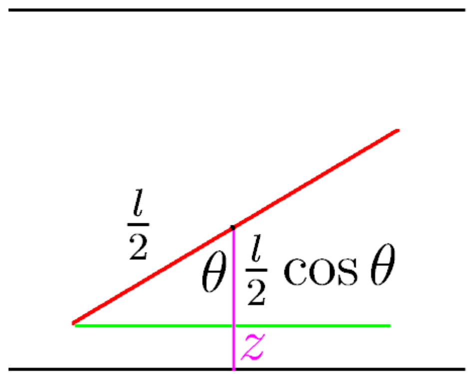
where $(x_1^{(k)},\ldots,x_D^{(k)})$ is a random vector from region Ω
($|\Omega|$ = area, volume, ...; calculation of π: $\Omega = (-1,1)^2$, $|\Omega| = 4$)

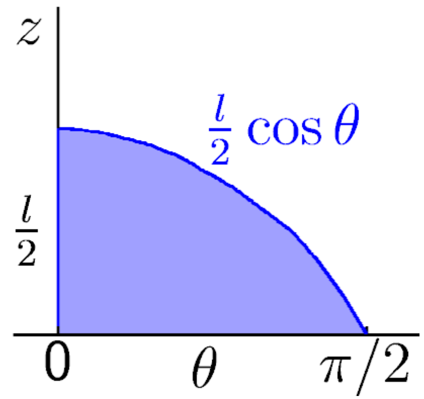
Exercise – Buffon's needle

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



Proof:





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

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

Usage (δp is the standard error of p)

rel. error

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

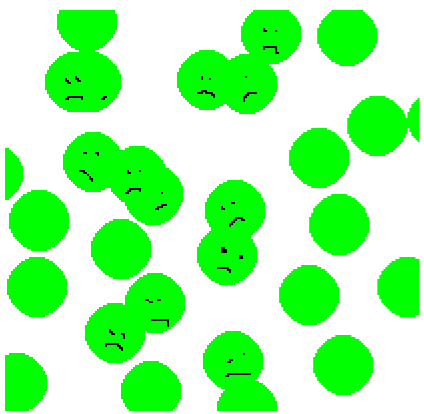
for me: grid: pic/buffon-grid.pdf and buffon.sh

Importance sampling

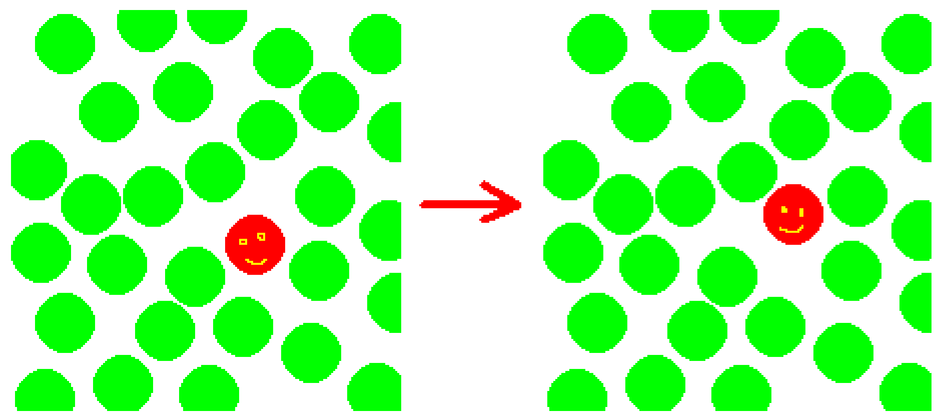
$$\sum \mathrm{e}^{-\beta U(\vec{r}^N)} f(\vec{r}^N) \approx \frac{1}{K} \sum_{k=1}^{K} f(\vec{r}^{N,(k)})$$

where $\vec{r}^{N,(k)}$ is a random vector with a probability density $\propto \mathrm{e}^{-\beta U(\vec{r}^N)}$.

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 $\exp(-\beta\Delta U)$

Why? Because then it holds for the probability ratio:

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

(moves there and back are compared, always the probability of one move = 1, and of the other = Boltzmann probability)

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

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

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 $\pi_i = \dfrac{\exp[-\beta U(A_i)]}{\sum_i \exp[-\beta U(A_i)]}$

Conditions:

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

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

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

$$\Uparrow$$

$$\pi_i W_{i\to j} = \pi_j W_{j\to i}$$

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 again

One of solutions (Metropolis):

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

Equivalent form:

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

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

Version 1	Version 2	Version 3

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

  Sequence: move molecule A – move molecule B – change volume – $\cdots$

- Randomly

Chaos is better than bad control

Heat-bath method

good for lattice models:

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

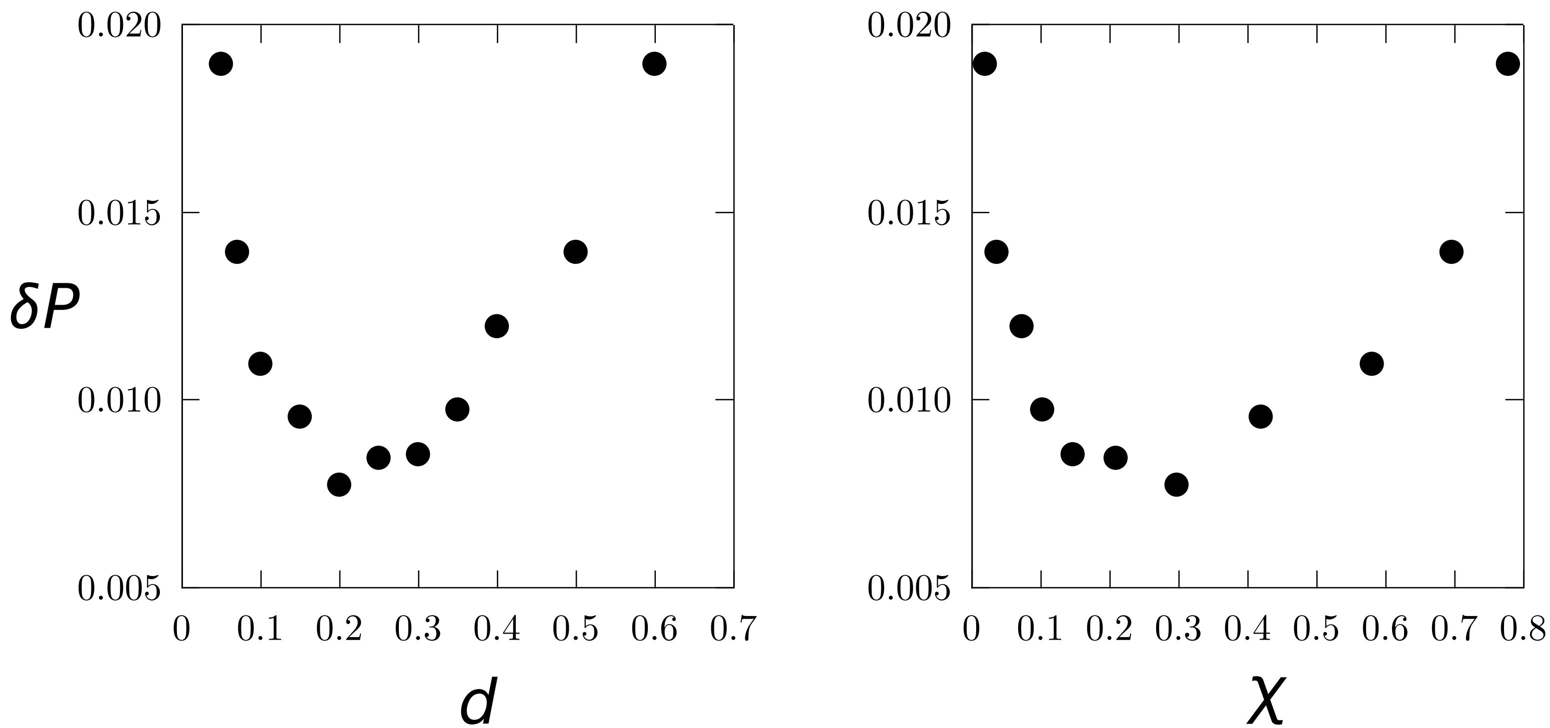
Usually:

- choose spin/lattice site
- choose a new spin value with a Boltzmann probability
  (it depends on the neighbourhood)

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...



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