Generators of pseudorandom numbers

- truly random numbers could be obtained in nature by measuring some random process
  - but it is slow, sequence is not reproducible
- therefore, we use pseudorandom numbers generated directly in the computer by some deterministic algoritm starting with a seed
- a good generator has a very large period and satisfies various tests of randomness (uniformity, autocorrelation etc.)

- linear congruential generator (LCG)

$$x_{i+1} = (a x_i + c) \bmod m$$

- it is fast, period can be optimal (largest, depends of integre type)
- historically, a special case of multiplicative congruential generator ($c=0$) was used, but it was found to have problems for applications in more dimensions; d-tuples $(x_i, x_{i+1}, \dots, x_{i+d-1})$ can lie on relatively small number of $(d-1)$-dim. hyperplanes
- as an example - generator RANDU - used for decades

$a = 2^{16}+3 = 65539$, $c=0$, $m=2^{31}$

here we set recurrence relation

$$x_{i+2} = (2^{16}+3)\, x_{i+1} \bmod 2^{31} = (2^{32}+6\cdot 2^{16}+9)\, x_i \bmod 2^{31} =$$
$$= (6 x_{i+1} - 9 x_i) \bmod 2^{31}$$

and it can be shown, that all triples (x_i, x_{i+1}, x_{i+2}) lie on only 15 paralel planes

- details can be found in
  Marsaglia: Random Numbers Fall Mainly in the Planes, PNAS 61 (1968) 25

- currently, the most widely used generators are based on linear recurrences, e.g.
  Mersenne Twister (Mersenne primes as periods
  - uniform distribution to very high dimensions
  or WELL (well equidistributed long-period linear)

Generating random numbers with other distributions

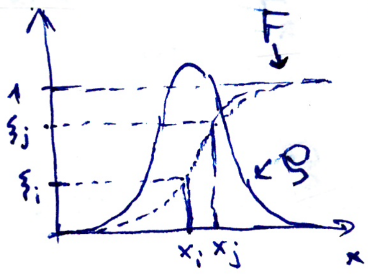
- once we have a pseudorandom number generator which provides uniformly-distributed numbers in the interval $\langle 0,1 \rangle$ (usually built-in functions in programming languages)

  we can generate (pseudo-)random numbers with different distributions

- there are special techniques based on <u>the inversion of the cumulative distribution function</u> (CDF) of a real-valued random variable X

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} \underset{\uparrow}{\varrho}(t)\, dt$$

probability density function



now, if we need random numbers distributed according a probability density function ϱ and we can compute the inversion F^{-1}, then simply take $\xi_i \in \langle 0,1)$ and set $x_i = F^{-1}(\xi_i)$

Examples:

1) exponential distribution on $\langle 0,\infty)$

we take $g(x) = \lambda e^{-\lambda x}$ satisfying $\int_0^\infty g(x)\,dx = 1$

now CDF: $F(x) = 1 - e^{-\lambda x} \Rightarrow F^{-1}(\xi) = -\frac{1}{\lambda}\ln(1-\xi)$

if $\xi \in \langle 0,1)$ uniformly then $x \in \langle 0,\infty)$ with $g(x)$ distribution

2) random points on the unit sphere

the surface element is $dS = \frac{1}{4\pi} R^2 d\rho \sin\theta\, d\varphi$

for unit sphere we would like ~~to~~ to generate points with the PDF

$$g(\theta,\varphi) = \frac{\sin\theta}{4\pi} \leftarrow \text{normalization } \int_0^{2\pi} d\varphi \int_0^{\pi} d\theta \frac{\sin\theta}{4\pi} = 1$$

the CDF is

$$F(\theta,\varphi) = \int_0^\varphi d\varphi \int_0^\theta d\theta \frac{\sin\theta}{4\pi} = \frac{\varphi}{2\pi}\cdot\frac{1-\cos\theta}{2} = F_\varphi \cdot F_\theta$$

thus we can generate points by

choosing $\xi_1 \in \langle 0,1\rangle \Rightarrow \varphi_i = F_\varphi^{-1}(\xi_1) = 2\pi\xi_1$,

choosing $\xi_2 \in \langle 0,1\rangle \Rightarrow \theta_i = F_\theta^{-1}(\xi_2) = \arccos(1-2\xi_2)$

3) <u>normal distribution</u>

a) we could use the central limit theorem and calculate sums of uniformly distributed points

b) unfortunately F^{-1} has no simple analytical form, but we can use a trick of Box, Muller: Ann Math Stat 29 (1958) 610 based on a two-dimensional normal distribution

$$g(x,y) = \frac{1}{2\pi} e^{-\frac{x^2+y^2}{2}}$$

and use the polar coordinates to get

$$g(x,y)\,dx\,dy = g_p(r,\varphi)\,dr\,d\varphi = \frac{1}{2\pi} e^{-r^2/2}\, r\,dr\,d\varphi$$

the CDF in polar coordinates is

$$F_\rho(r,\varphi) = \frac{\varphi}{2\pi}\int_0^r r' e^{-\frac{r'^2}{2}} dr' = \frac{\varphi}{2\pi}\left(1-e^{-r^2}\right) = F_\varphi \cdot F_r$$

thus by

choosing $\xi_1 \in \langle 0,1\rangle \Rightarrow r_i = \sqrt{-\ln(1-\xi_1)}$

choosing $\xi_2 \in \langle 0,1\rangle \Rightarrow \varphi_i = 2\pi\,\xi_2$

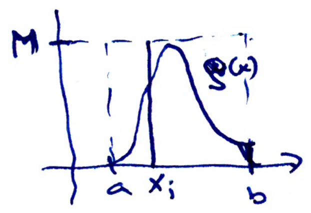
we get two points from the normal distribution:

$x_i = r_i \cos\varphi_i \quad , y_i = r_i \sin\varphi_i$

- for a general distribution which has no simple F^{-1} or we don't know any tricks we can use

a) von Neumann method

- illustrated in 1D, but it is straightforward to extend it to higher dimen. (not very efficient!)



we generate two random variables $\xi_i, \eta_i \in \langle 0,1\rangle$ uniformly

and map them to $\langle a,b\rangle$ or $\langle 0,M\rangle$

$x_i = a + \xi_i(b-a)$

$y_i = M\eta_i$

where M must be chosen to satisfy $w(x) < M$ for all $x \in \langle a,b\rangle$

- now, if $y_i \le \rho(x_i)$ then we use x_i

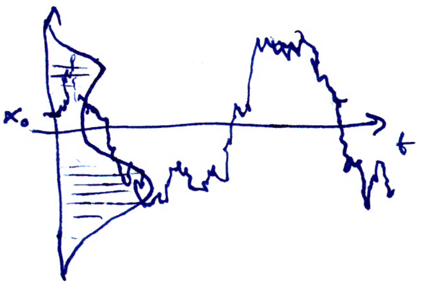
if $y_i > \rho(x_i)$ then we throw it away

thus the probability of accepting a point in $\langle x, x+dx\rangle$ is proportional to $\rho(x)dx$ for ξ_i is uniformly distributed on $\langle a,b\rangle$

- the problem is that for $\rho(x)$ localised is relatively small region we throw away a lot of points and the method is not efficient

b) Metropolis-Hastings algorithm

- basic idea : walk randomly in space of variables according to the probability density function, i.e. from x_k we try to ~~do~~ do a step $\delta x_k = d \cdot \xi_k$ where ξ_k is uniform on $\langle -1, 1 \rangle$ and d is a parameter (the longest step)



we accept the step if $g(x_k + \delta x_k) \geq g(x_k)$

or if $g(x_k + \delta x_k) < g(x_k)$ then we will generate $\eta \in \langle 0, 1 \rangle$ and we accept the step if $\frac{g(x_k+\delta x_k)}{g(x_k)} > \eta$

if we do not accept the step then $x_{k+1} = x_k$

- in D-dimensinal space the algorithm is

subroutine returning the next point strarting from x(:)

```
r(:) = random(:)        ← D random numbers each from <0,1>
xtrial(:) = x(:) + d(2r(:) - 1)     ← shift to <-1,1>
ratio = g(xtrial) / g(x)
if (ratio ≥ 1 or ratio > random) then
    x = xtrial
return x(:)
```

- the crutial parameter is d

if it is too short we cannot jump over regions of small probability density and points are strongly correlated

if it is too large we can throw away too many tries and the algorithm is not efficient

optimal choice depends on $g(x)$

- in the limit of long random walks this algorithm for reasonable d really generates points distributed according to $g(x)$

- in general, this can be shown using the theory of Markov chains (as briefly discussed below) but first we give a „physical" explanation

- let $N_n(x)$ denote a density of random independent walkers who get to x after n steps
- the change of $N(x)$ in the next step due to going to y minus coming from y is

$$\Delta N(x) = N_n(x) P(x \to y) - N_n(y) P(y \to x) =$$

$$= N_n(y) P(x \to y) \left[\frac{N_n(x)}{N_n(y)} - \frac{P(y \to x)}{P(x \to y)} \right]$$

where $P(x \to y)$ is probability of going from $x \to y$

- equilibrium occurs when

$$\frac{N_n(x)}{N_n(y)} = \frac{P(y \to x)}{P(x \to y)}$$

and for large n it can be shown that $N_n(x) \to N_e(x)$ which, according to M-H algorithm, should be $N_e(x) \sim \varrho(x)$

- this is because we can express

$$P(x \to y) = \underset{\substack{\uparrow \\ \text{probability of doing step } x \to y}}{T(x \to y)} \underset{\substack{\uparrow \\ \text{probability of accepting step } x \to y}}{A(x \to y)}$$

and for reasonable step d when $y \in \langle x-d, x+d \rangle$ will be $A(x \to y) = A(y \to x)$

and for $\varrho(x) \geq \varrho(y)$: $T(y \to x) = 1$ and $T(x \to y) = \frac{\varrho(y)}{\varrho(x)}$

and for $\varrho(x) < \varrho(y)$: $T(x \to y) = 1$ and $T(y \to x) = \frac{\varrho(x)}{\varrho(y)}$

thus together $\quad \frac{T(y \to x)}{T(x \to y)} = \frac{\varrho(x)}{\varrho(y)}$

and $\quad \frac{N_e(x)}{N_e(y)} = \frac{\varrho(x)}{\varrho(y)}$ at least for all $x, y : |x-y| < d$ but the result is valid also if we can get from x to y in finite number of steps