Metody numeryczne

Wykład nr 12

Dr Piotr Fronczak

Generowanie liczb losowych

Metody Monte Carlo są oparte na probabilistyce – działają dzięki generowaniu liczb losowych.

W komputerach te liczby generowane przez odpowiednie procedury nie są wcale losowe, a raczej deterministyczne i odtwarzalne – choć wyglądają jak ciąg przypadkowych liczb.

Procedury te nazywamy więc **generatorami liczb pseudolosowych**.

Własności generatorów liczb pseudolosowych (GLP)

• **powtarzalność** –sekwencja wylosowanych liczb powinna być powtarzalna przy przyjęciu tej samej pierwszej wylosowanej liczby.

• **losowość** – GLP powinien generować niezależne, jednorodnie rozłożone liczby, które pozytywnie przechodzą wszelkie statystyczne testy na losowość.

• **Długi okres** – okres generatora powinien być znacznie dłuższy niż liczba potrzebnych w symulacji liczb losowych.

• **Nieczułość na warunek początkowy** – okres i własności losowe nie powinny zależeć od wyboru warunku początkowego.

Liniowy generator kongruentny (LCG)

Zaproponowany przez D. H. Lehmera w 1951 roku.

Większość dzisiejszych procedur do generowania liczb pseudolosowych jest oparta na tym generatorze.

.

$$x_{n+1} = (ax_n + b) \pmod{m}$$

Przykład: $x_0 = 7$, $a = 1$, $b = 7$, $m = 10$,

Otrzymamy następującą sekwencję liczb:

7, 4, 1, 8, 5, 2, 9, 6, 3, 0, 7, 4, …

Wybór parametru *m*.

- *m* powinno być jak największe, bo okres nie może być większy niż *m*.
- Zwykle wybieramy *m* najbliższe największej liczbie całkowitej dostępnej w komputerze (dla komputerów 32-bitowych $2^{32} - 1$).

Wady LCG:

Weźmy $x_{n+1} = (5\, x_n + 1) \bmod (16)$

Mamy okres P =16, (najdłuższy możliwy).

	Integer	Binary	Real
x_0:	1	0001	1/16 = 0.0625
x_1:	6	0110	6/16 = 0.3750
x_2:	15	1111	15/16 = 0.9375
x_3:	12	1100	12/16 = 0.7500
x_4:	13	1101	13/16 = 0.8125
x_5:	2	0010	2/16 = 0.1250
x_6:	11	1011	11/16 = 0.6875
x_7:	8	1000	8/16 = 0.5000
x_8:	9	1001	9/16 = 0.5625
x_9:	14	1110	14/16 = 0.8750
x_{10}:	7	0111	7/16 = 0.4375
x_{11}:	4	0100	4/16 = 0.2500
x_{12}:	5	0101	5/16 = 0.3125
x_{13}:	10	1010	10/16 = 0.6275
x_{14}:	3	0011	3/16 = 0.1875
x_{15}:	0	0000	0/16 = 0.0000

Korelacje w ostatnim bicie!

…1010101…

Można pokazać, że

$\langle x \rangle = 1/2$

$\langle (x - \langle x \rangle)^2 \rangle = 1/12 = 0.083.$

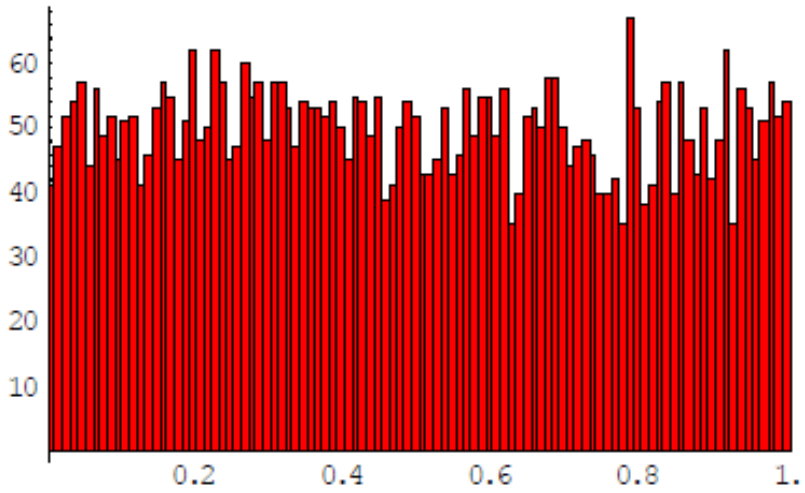
U nas:

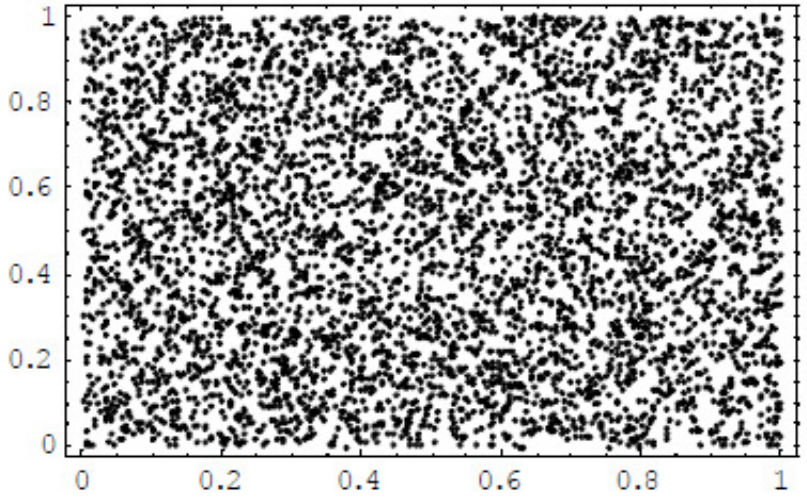
$\langle x \rangle = 0.469$

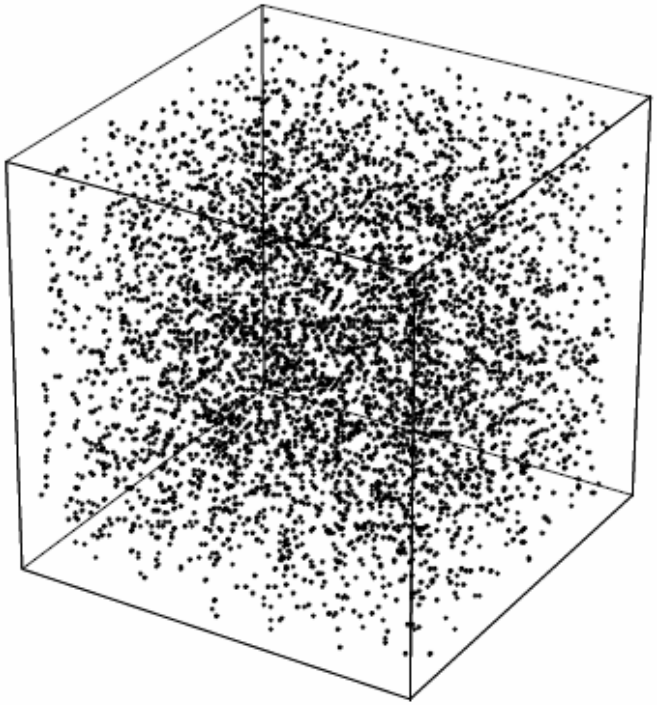
$\langle (x - \langle x \rangle)^2 \rangle = 1/12 = 0.088.$

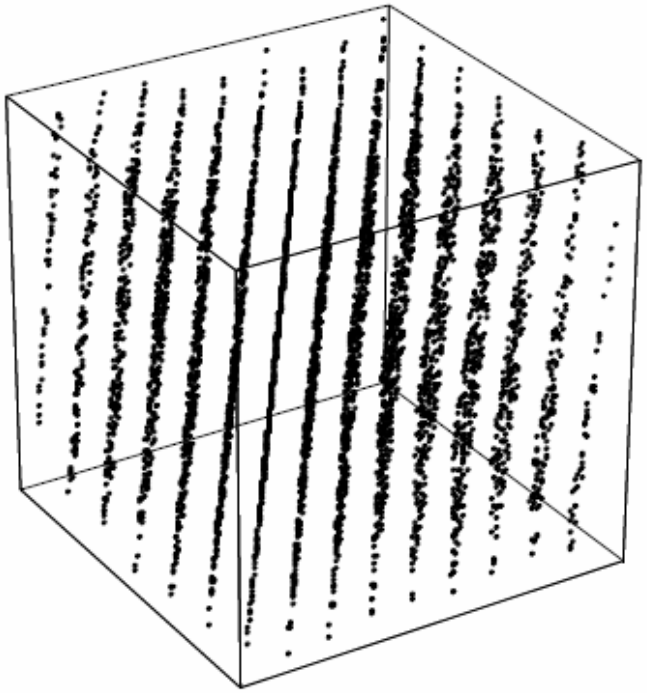
Całkiem nieźle.

Jednorodność w 1D nie oznacza jednorodności w 2D









Można pokazać, że punkty, których współrzędnymi jest kolejnych d liczb z sekwencji generatora LCG leżą w przestrzeni d-wymiarowej na równoległych d-1 wymiarowych hiperpłaszczyznach.

Liczba hiperpłaszczyzn $N \approx \sqrt[d]{m}$

Przykłady współczesnych generatorów LCG

Source	a	b	Output bits in rand()
Borland C/C++	22695477	1	16 - 30
Glibc (GCC)	1103515245	12345	0 - 30
ANSI C	1103515245	12345	16 - 30
Borland Delphi	134775813	1	32 - 63
Microsoft C/C++	214013	2531011	16 - 30

We wszystkich powyższych kompilatorach m = 2^{32} - 1

Generator Fibonacciego (LFG)

Generator *Fibonacciego* jest jednym z wielu wariantów uogólnionego generatora liniowego *liczb losowych*.

LFG stają się popularne w ostatnich latach.

Nazwane tak z powodu podobieństwa do sekwencji Fibonacciego:

$$F_n = F_{n-1} - F_{n-2}$$

0, 1, 1, 2, 3, 5 ,8, 13, …

LFG wykorzystuje do generowania liczb losowych algorytm

$x_n = x_{n-l} + x_{n-k} \pmod{m}$ gdzie $l > k > 0$

z *m wartościami początkowymi* $\{x_0, x_1, \ldots, x_{l-1}\}$.

Generator Fibonacciego (LFG)

W większości aplikacji , m jest potęgą dwójki, $m = 2^M$.

Przy odpowiednim wyborze parametrów l, k i wartości początkowych, okres generatora

$$P = (2l - 1) \times 2M - 1.$$

Dwa najczęściej używane LFG:

$x_n = x_{n-17} + x_{n-5} \pmod{2^{31}}$ okres $\approx 2^{47}$

$x_n = x_{n-55} + x_{n-24} \pmod{2^{31}}$ period $\approx 2^{85}$

W odróżnieniu od LCG, wartość parametru m nie ogranicza okresu generatora.

LFG jest prosty w implementacji:

- dodawanie całkowite
- logiczne AND (operacja mod 2^M)
- tablica (niewielka).

Niejednorodne generatory liczb losowych

Większość generatorów daje liczby rozłożone jednorodnie.

Rozkłady nieliniowe można otrzymać za pomocą:

- metody odwracania dystrybuanty
- metody akceptacji / odrzucenia

Metoda odwracania dystrybuanty

Jeśli zmienna losowa U ma rozkład jednostajny na odcinku [0, 1],

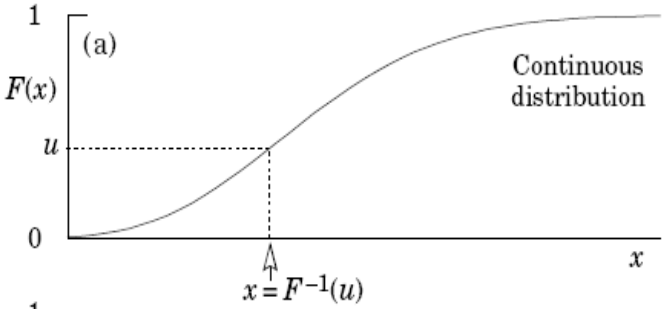
to zmienna losowa $X = F^{-1}(U)$ *ma rozkład o dystrybuancie* $F(x)$.

Dowód:

Ponieważ

$$P\{X \le x\} = P\{F^{-1}(U) \le x\} = P\{U \le F(x)\} = F(x),$$

więc zmienna losowa X ma rzeczywiście rozkład prawdopodobieństwa o dystrybuancie F.



Przykład: rozkład wykładniczy

Gęstość rozkładu wykładniczego

$$f(x) = \lambda e^{-\lambda x} \qquad x \geq 0$$

Liczymy dystrybuantę

$$F(x) = \int_0^x \lambda e^{-\lambda x'} dx' = -e^{-\lambda x'}\Big|_0^x = 1 - e^{-\lambda x}$$

Zatem

$$U = F(X) = 1 - e^{-\lambda X}$$

$$X = -\frac{1}{\lambda}\ln(1-U)$$

Ponieważ U $\in$ [0,1], to kusi, aby liczyć

$$X = -\frac{1}{\lambda}\ln(U)$$

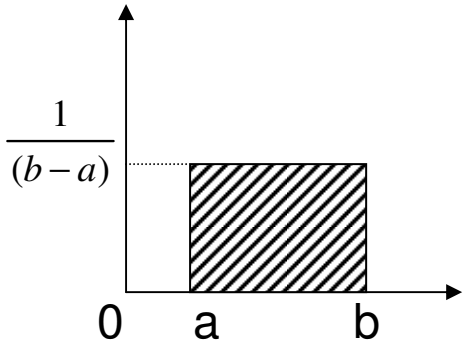
Niebezpieczne!

Przykład: rozkład jednorodny

Rozważmy zmienną losową X jednorodnie rozłożoną na przedziale [a, b].

Funkcja gęstości X ma postać

$$f(x)=\begin{cases}\dfrac{1}{b-a}, & a\le x\le b\\ 0, & \texttt{w pozostałych przyp.}\end{cases}$$



Dystrybuanta ma postać

$$F(x)=\begin{cases}0, & x<a\\ \dfrac{x-a}{b-a}, & a\le x\le b\\ 1, & x>b\end{cases}$$

Zatem

$$F(X)=\frac{X-a}{b-a}=U$$

Odwracając

$$X=a+(b-a)U$$

Zalety:

• Dokładna.

• Prosta i szybka dla niektórych rozkładów.

• Do wygenerowania zmiennej losowej o danym rozkładzie potrzebna jest tylko jedna liczba losowa o rozkładzie U(0, 1).

Wady:

• Na ogół wymagane jest aby dystrybuanta była znana i odwracalna analitycznie → stosunkowo niewielka liczba funkcji!

• W zasadzie, można numerycznie odwracać dystrybuantę – ta metoda jest jednak zwykle znacznie wolniejsza, mniej dokładna i bardziej narażona na niestabilności numeryczne.

Numeryczne odwracanie dystrybuanty

Szukanie miejsca zerowego X_0 funkcji:

$$U - F(X_0) = U - \int_0^{X_0} f(x)dx = 0$$

(np. metodami: siecznych, stycznych itd.)

Generowanie liczb z rozkładu normalnego

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{x^2}{2\sigma^2} \right)$$

Stosując metodę odwracania dystrybuanty dostajemy równanie

$$U = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{x} \exp\left(-\frac{x'^2}{2\sigma^2} \right) dx' = \frac{1}{2}\left[1 + erf\left(\frac{x}{\sqrt{2}\sigma} \right) \right]$$

Niestety, tego równania nie da się odwrócić…

Metoda Boxa-Mullera

Weźmy dwie niezależne liczby losowe *x* oraz *y*, wylosowane z rozkładu normalnego. Prawdopodobieństwo, że punkt (*x,y*) leży w małym obszarze d*x*d*y* płaszczyzny *xy*:

$$f(x,y)dxdy = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2} \right) dxdy$$

Wyrażając to we współrzędnych biegunowych

$$f(r,\theta)drd\theta = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{r^2}{2\sigma^2}\right)rdrd\theta$$

Jeśli wygenerujemy zgodnie z powyższym rozkładem dwie liczby *r* oraz θ,
a następnie przetransformujemy je z powrotem do współrzędnych kartezjańskich *x* i *y,*
otrzymamy dwie liczby losowe z rozkładu Gaussa.

Wygenerowanie θ jest proste – losujemy liczbę z rozkładu jednorodnego [0, 2π).

r generujemy metodą odwrotnej dystrybuanty

$$f(r) = \frac{1}{\sigma^2}\exp\left(-\frac{r^2}{2\sigma^2}\right)r$$

$$U = \frac{1}{\sigma^2}\int_0^r \exp\left(-\frac{r'^2}{2\sigma^2}\right)r'\,dr' = 1-\exp\left(-\frac{r^2}{2\sigma^2}\right)$$

$$r = \sqrt{-2\sigma^2\log(1-U)}$$

Mając *r* oraz θ otrzymujemy dwie liczby losowe z rozkładu Gaussa

$$x = r\sin\theta$$

$$y = r\cos\theta$$

Inna metoda

Centralne twierdzenie graniczne

Jeżeli X_n jest ciągiem niezależnych zmiennych o jednakowym rozkładzie, o wartości przeciętnej <X> i skończonej wariancji σ^2, to ciąg zmiennych losowych

$$Y_n = \frac{\sum_{i=1}^{n} X_i - n\langle X \rangle}{\sigma\sqrt{n}}$$

jest zbieżny do zmiennej losowej o rozkładzie normalnym N(0,1).

$n = 30$ wystarczy, aby Y_{30} traktować jak zmienną losową o rozkładzie normalnym.

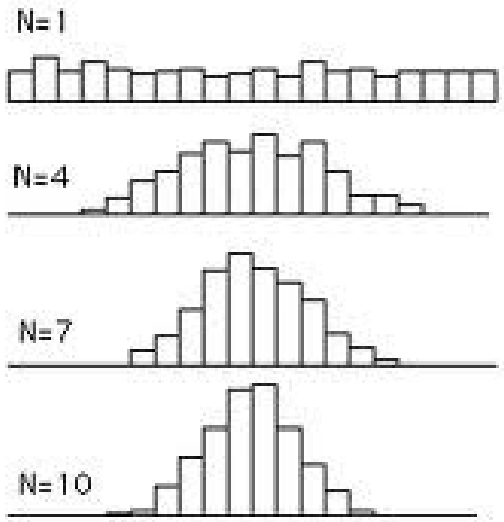
Algorytm

1 Przyjmujemy odpowiednio duże n.

2 Wybieramy rozkład dla X_i o znanej wartości oczekiwanej $<X>$ i znanej wariancji σ^2. Kierujemy się również tym, aby algorytm generowania zmiennych X_i był szybki.

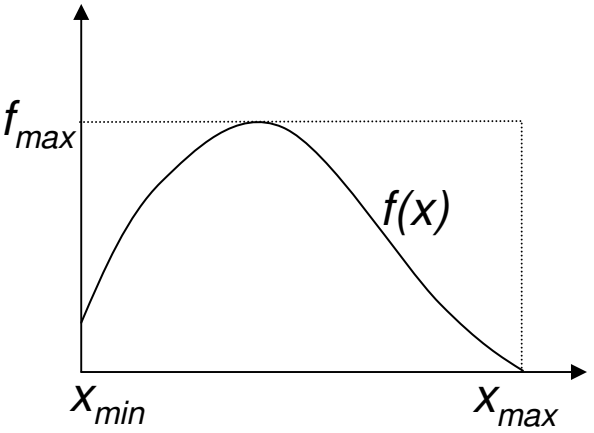
3 Losujemy n realizacji niezależnych zmiennych losowych X_i .

4 Liczymy Y_n i uznajemy je za realizacje zmiennej losowej o rozkładzie normalnym.



Metoda eliminacji

Chcemy wygenerować liczby losowe *x* w przedziału $[x_{min}, x_{max}]$ zgodnie z rozkładem prawdopodobieństwa *f(x).*



Niech f_{max} – maksymalna wartość *f(x)* na przedziale $[x_{min}, x_{max}]$

1. Losujemy liczbę losową x z rozkładu jednorodnego

$$x = x_{\min} + (x_{\max} - x_{\min})U$$

2. Losujemy inną liczbę losową *r* z rozkładu jednorodnego (między 0 i 1)

$$r = U$$

3. Jeśli

$$r < \frac{f(x)}{f_{\max}}$$

akceptujemy liczbę losową *x*.

W przeciwnym wypadku, powracamy do kroku 1.

Dlaczego to działa?

Prawdopodobieństwo wygenerowania liczby z przedziału $x - x+\mathrm{d}x$

$$p_{gen}(x)dx = \frac{dx}{x_{\max} - x_{\min}}$$

Prawdopodobieństwo akceptacji

$$p_{akcept}(x) = \frac{f(x)}{f_{\max}}$$

Zatem prawdopodobieństwo, że nasz generator zwróci liczbę x:

$$p(x)dx = p_{gen}(x) \cdot p_{akcept}(x) = \frac{f(x)dx}{f_{\max}(x_{\max} - x_{\min})}$$

Czyli jest ono proporcjonalne do $f(x)dx$.

Fakt, że czasem odrzucamy wylosowaną liczbę, zmniejsza efektywność algorytmu.

Obliczmy liczbę kroków algorytmu (liczbę wywołania funkcji rand()), by uzyskać średnio jedną liczbę losową.

Prawdopodobieństwo *P*, że wylosujemy jakąkolwiek liczbę w jednym kroku:

$$P = \int_{x_{\min}}^{x_{\max}} p(x)dx = \frac{\int_{x_{\min}}^{x_{\max}} f(x)dx}{f_{\max}(x_{\max} - x_{\min})}$$

A zatem liczba kroków algorytmu potrzebna, by uzyskać jedną liczbę

$$N_{kroków} = \frac{1}{P} = \frac{f_{\max}(x_{\max} - x_{\min})}{\int_{x_{\min}}^{x_{\max}} f(x)dx}$$

Liczba wywołań funkcji rand()

$$N_{rand} = \frac{2 f_{\max}(x_{\max} - x_{\min})}{\int_{x_{\min}}^{x_{\max}} f(x)dx}$$

Porównajmy dwa sposoby generowania liczb z rozkładu normalnego N(0,1).

Jeśli granice zakresu liczb leżą 6 odchyleń standardowych od średniej (1 na 10^7 liczb leży poza tym zakresem), a $\sigma = 1$, to

$$x_{\min} = -6, \quad x_{\max} = 6$$

$$f(x) = \exp\left(-\tfrac{1}{2}x^2\right)$$

$$N_{rand} = \frac{2 f_{\max}(x_{\max} - x_{\min})}{\int_{x_{\min}}^{x_{\max}} f(x)dx} = \frac{24}{\int_{-6}^{6} \exp(-\tfrac{1}{2}x^2)dx} = 9{,}57...$$

Czyli 10 razy więcej niż w metodzie odwracania dystrybuanty.

Hybrydowa metoda eliminacji

Znajdźmy funkcję *g(x)*, która jest podobna do *f(x),* ale jest całkowalna i

$$g(x) \geq f(x) \qquad dla \qquad x_{\min} \leq x \leq x_{\max}$$

Zastosujmy metodę odwrotnej dystrybuanty, by wylosować liczbę losową z rozkładu *g(x)*. Jeśli *g(x)* jest znormalizowana (czyli *f(x)* nie jest), to

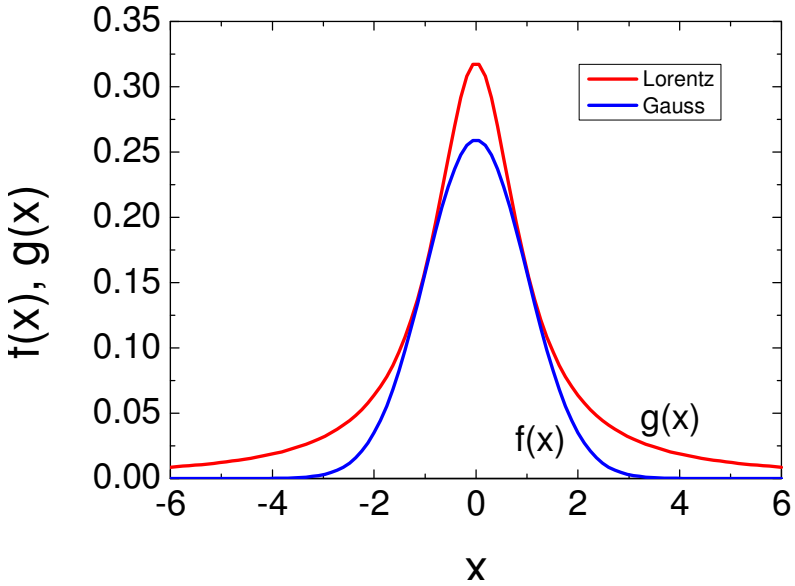
$$p_{gen}(x)dx = g(x)dx$$

$$p_{akcept}(x) = \frac{f(x)}{g(x)}$$

$$p(x)dx = p_{gen}(x) \cdot p_{akcept}(x) = f(x)dx$$

Niech *g(x)* będzie funkcją Lorentza

$$g(x) = \frac{1}{\pi}\frac{1}{1+x^2}$$



Aby *f(x)* leżało poniżej funkcji *g(x),* trzeba je przeskalować (metodą prób i błędów)

$$f_{new}(x) = 0.65\, f_{old}(x)$$

Można pokazać, że dla liczby *x* wylosowanej z rozkładu Lorentza (sprawdź!)

$$x = \tan\left[\pi(U - \tfrac{1}{2})\right]$$

Prawdopodobieństwo wylosowania liczby w jednym pełnym kroku algorytmu

$$P = \int p(x)dx = \int f_{new}(x)dx = 0.65 \cdot \int f_{old}(x)dx = 0.65$$

$$N_{rand} = \frac{2}{P} \approx 3$$

Czyli aby lepiej niż w prostej metodzie eliminacji, ale nadal gorzej niż w metodzie odwrotnej dystrybuanty.