Computational Physics

Monte Carlo Methods

Korea University
Eunil Won

The Monte Carlo Method

- The Monte Carlo Method

The Monte Carlo method is a numerical technique for calculating probabilities and related quantities by using sequences of random numbers.

1) First, a series of random values $r_1, r_2, \ldots$ is generated according to a uniform distribution interval $0 < r < 1$. That is the p.d.f. $g(r)$ is given by

$$g(r) = \begin{cases} 1 & 0 < r < 1 \\ 0 & \text{otherwise} \end{cases}$$

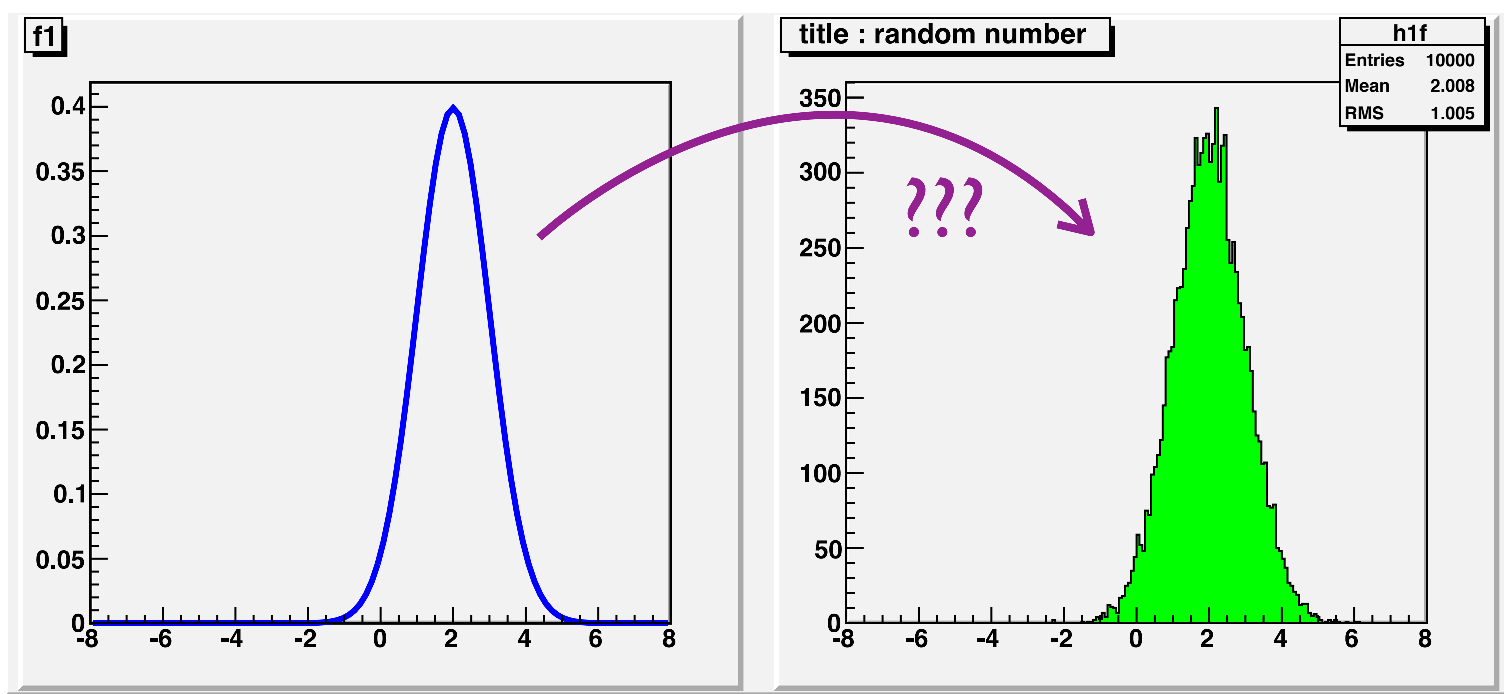
2) Next, the sequences $r_1, r_2, \ldots$ is used to determine another sequence $x_1, x_2, \ldots$ such that x values are distributed according to a p.d.f. $f(x)$ in which one is interested.

ex) A series of random values $r_1, r_2, \ldots$ is generated with a uniform distribution. How can I utilize it to get a series of random values that are distributed according to exp(-x) ?

(Note: it is relatively easy to generate uniform random numbers in computers but difficult to get them for a general p.d.f.)

The Monte Carlo Method

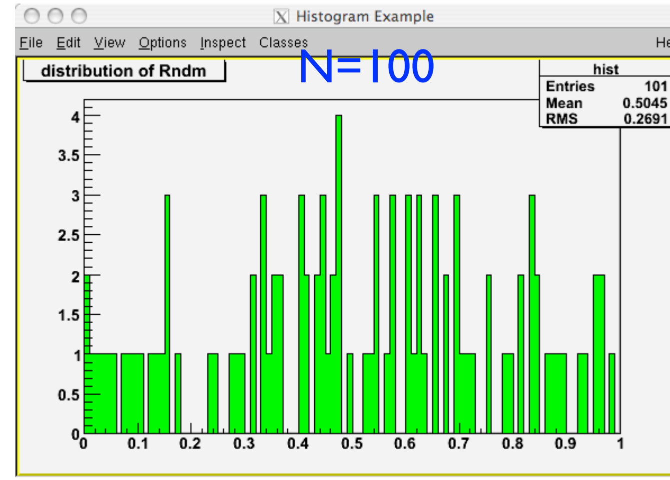
ex) A p.d.f of Gaussian is known. How can I generate random numbers that is populated according to the given p.d.f.?

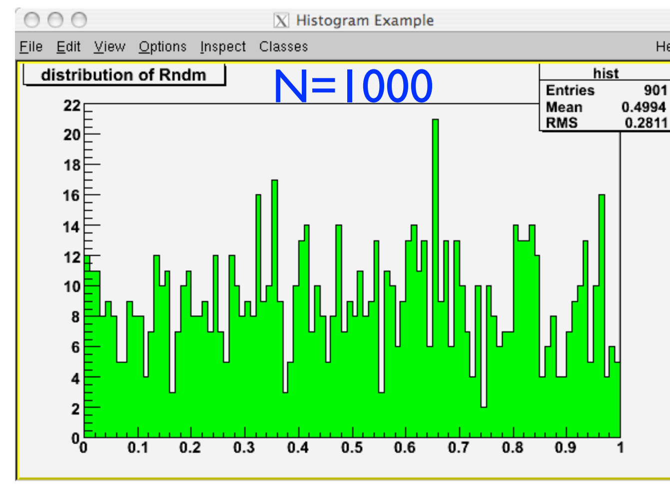


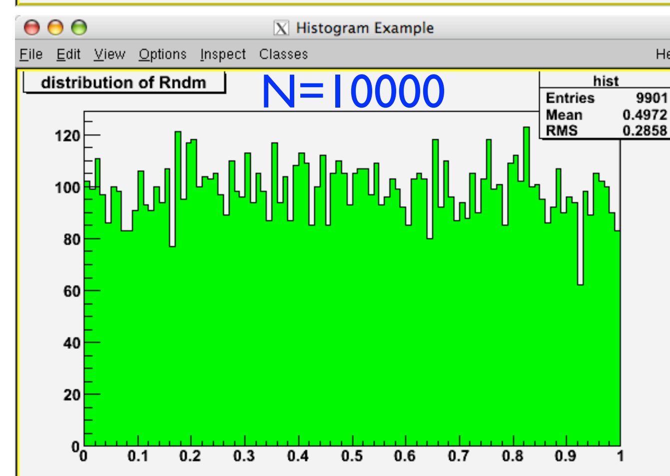
Uniform random numbers

- How to generate uniform random numbers in ROOT?

```
//
// Drawing a histogram with random numbers
//
// E. Won (eunil@hep.korea.ac.kr)

void random()
{
  gROOT->Reset();

  TCanvas* my_canvas = new TCanvas("my_canvas","",200,10,600,400);

  hist = new TH1F("hist","distribution of Rndm",100,0.0,1.0);
  hist->SetFillColor(3);
  hist->Draw();

  // Prepare the random number generator.
  gRandom->SetSeed();

  // Loop generating data according to Rndm
  Float_t data;
  Int_t dummy;
  for ( Int_t i=0; i<10000; i++)
  {
    data = gRandom->Rndm(dummy);
    hist->Fill(data);
    if ( i%100 == 0 ) { my_canvas->Modified(); my_canvas->Update(); }
  }
}
```







The transformation method-1

We would like to get $\{x_i\}$ from $\{r_i\}$

$$\{r_i\} \text{ uniform in } [0,1] \longrightarrow \{x_i\} \text{ distributed according to p.d.f } f(x)$$

and one way of achieving it is called the transformation method that we describe here.

$$\text{The probability of } r \in [r, r+dr] = g(r)dr$$

$$\text{The probability of } x \in [x, x+dx] = f(x)dx$$

Therefore, we require that

$$\int_{-\infty}^{x(r)} f(x')dx' = \int_{-\infty}^{r} g(r')dr' = r \qquad \text{because} \qquad g(r) = \begin{cases} 1 & 0 < r < 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{so} \qquad \int_{-\infty}^{x} f(x')dx' = F(x) - F(-\infty \text{ (or } x_{\min})), \quad F(x) = F(-\infty \text{ (or } x_{\min})) + r$$

$$\therefore \quad x(r) = F^{-1}\Big(F(-\infty \text{ (or } x_{\min})) + r\Big)$$

The transformation method-2

Example)

How can I generate random numbers distributed as *f(x)* = 1/ξ exp(-x/ξ) when a set of uniform random numbers in [0,1] is available (ξ > 0)?

First,

$$f(x) \text{ is a p.d.f since } \int_0^\infty \frac{1}{\xi} e^{-x'/\xi} dx' = -e^{-x/\xi}\Big|_0^\infty = 1$$

From the relation between integral of *f dx* and *g dr*,

$$\int_0^{x(r)} \frac{1}{\xi} e^{-x'/\xi} dx' = r$$

$$1 - e^{-x/\xi} = r$$

$$x(r) = -\xi \log(1 - r)$$

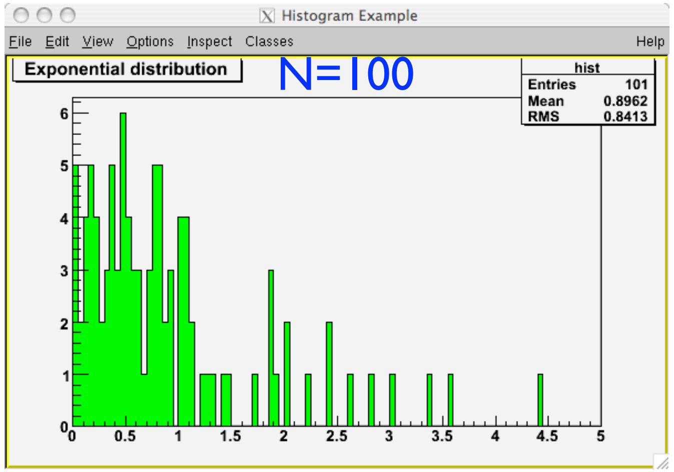
We can replace *r* to *1-r* since *r* is uniform in [0,1], so simplified relation is

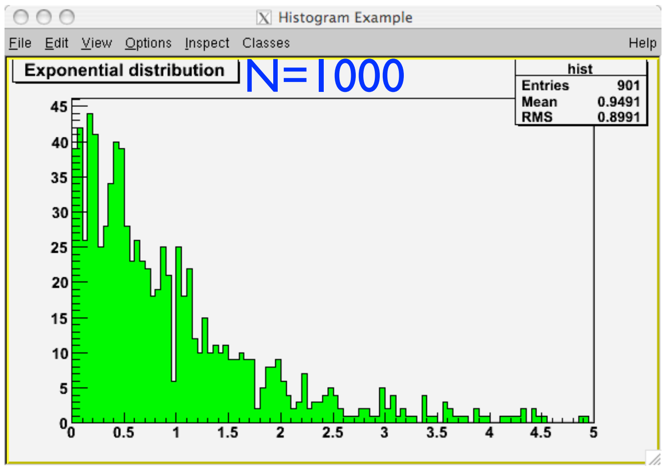
$$x(r) = -\xi \log r$$

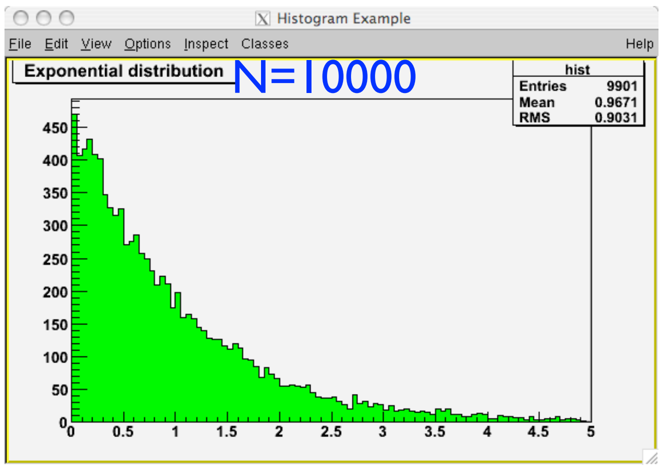
The transformation method-3

Let us verify the result of the example with C++ codes

```
//
// Drawing a histogram with random numbers
//
// E. Won (eunil@hep.korea.ac.kr)

void rexp()
{
  gROOT->Reset();

  TCanvas* my_canvas = new TCanvas("my_canvas","",200,10,600,400);

  hist = new TH1F("hist","Exponential distribution",100,0.,5.);
  hist->SetFillColor(3);
  hist->Draw();

  gRandom->SetSeed();

  // Loop generating data according to an exponential.
  Float_t data;
  Int_t dummy;
  for ( Int_t i=0; i<10000; i++)
  {
    data = - log( gRandom->Rndm(dummy) );
    hist->Fill(data);
    if ( i%100 == 0 ) { my_canvas->Modified(); my_canvas->Update(); }
  }
}
```



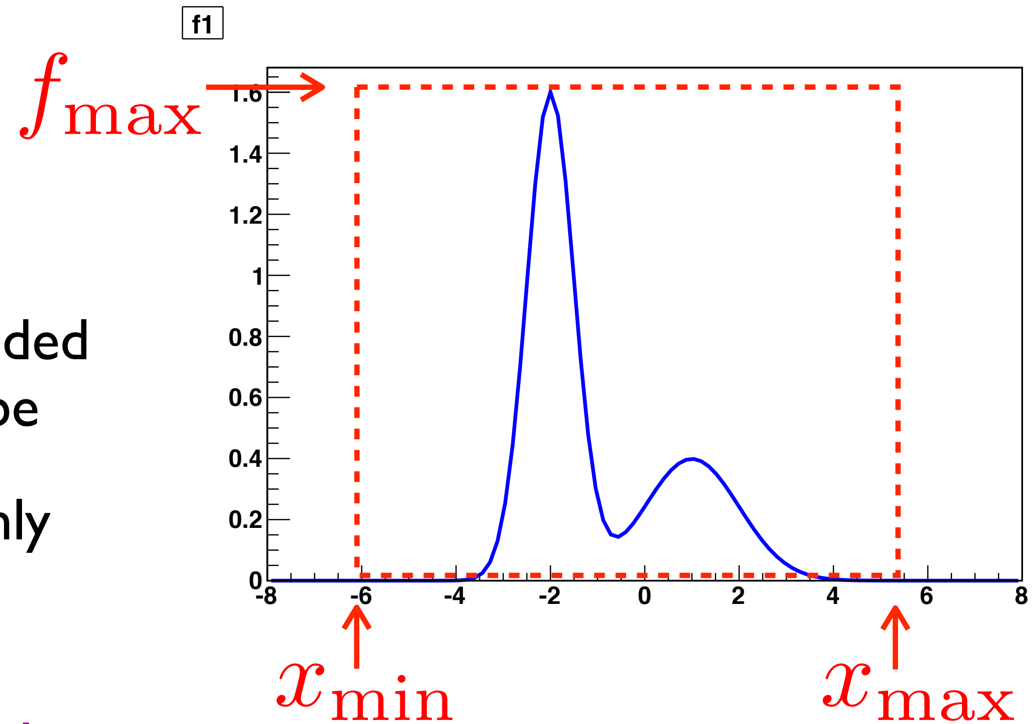




The acceptance-rejection method

It turns out that the transformation method is not possible in some cases (one has to take inverse of the function).

von Neumann's acceptance-rejection method



Consider a p.d.f. f(x) which can be surrounded by a finite box. The desired algorithm can be

1) Generate a random number *x*, uniformly distributed in [x_{min}, x_{max}], i.e.

$$
\begin{aligned}
x &= x_{\min} + r_1(x_{\max} - x_{\min}) \\
&\quad \text{where } r_1 \text{ is uniformly distributed in } [0,1]
\end{aligned}
$$

2) Generate 2nd independent random number *u* in [0,f_{max}], i.e $u = r_2 f_{\max}$

3) If $u < f(x)$, then accept *x*. If not, reject *x* and repeat.

The acceptance-rejection method

ex) $f(x) = \frac{1}{36}\left(\frac{1}{5}x^4 + 1\right)$ in $[-1, 2]$