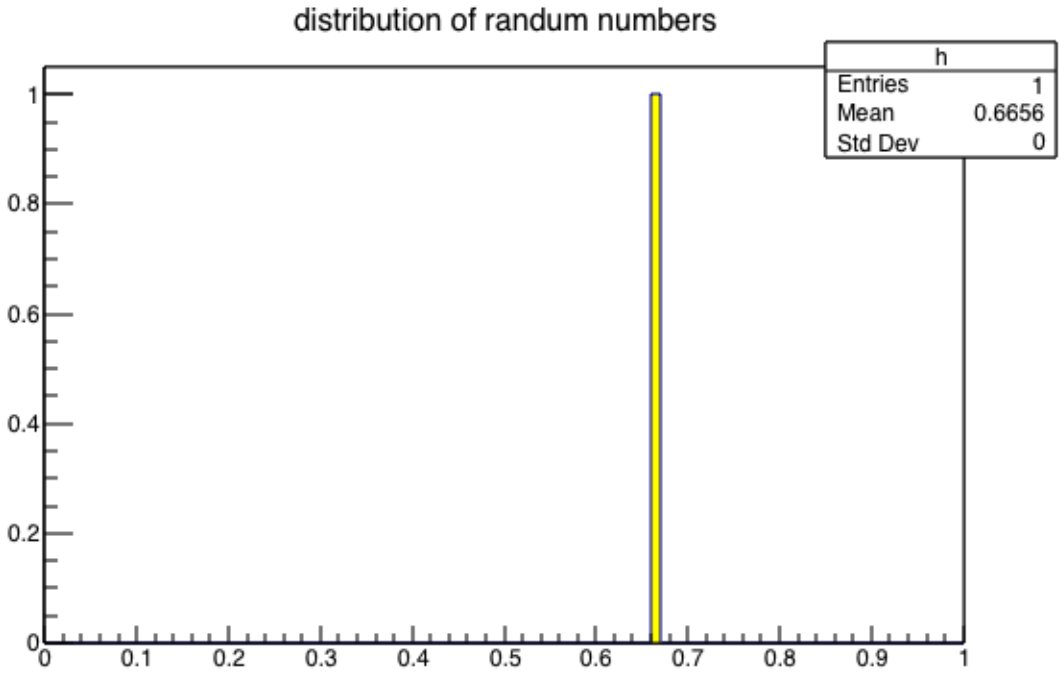
Random number & Monte Carlo (MC) Method

Uniform random numbers

- A series of random values x_1, x_2, …, x_n is generated according to a uniform distribution interval [0,1]. The probability density function (pdf), f(x) is given by

$$f(x) = \begin{cases} 1 & (0 < x < 1) \\ 0 & otherwise \end{cases}$$

- We have been using the program ran0.c for generating uniform random numbers
  - How to generate uniform random numbers in ROOT?



Random number generation

- ✓ ran0.c in the Numerical Recipes

```c
#define IA 16807
#define IM 2147483647
#define AM (1.0/IM)
#define IQ 127773
#define IR 2836
#define MASK 123459876

float ran0(long *idum)
{
  long k;
  float ans;

  *idum ^= MASK;
  k=(*idum)/IQ;
  *idum=IA*(*idum-k*IQ)-IR*k;
  if (*idum < 0) *idum += IM;
  ans=AM*(*idum);
  *idum ^= MASK;
  return ans;
}
#undef IA
#undef IM
#undef AM
#undef IQ
#undef IR
#undef MASK
/* (C) Copr. 1986-92 Numerical Recipes Software 31191a+)-. */
```

Random number generators

Implementations and recommendations

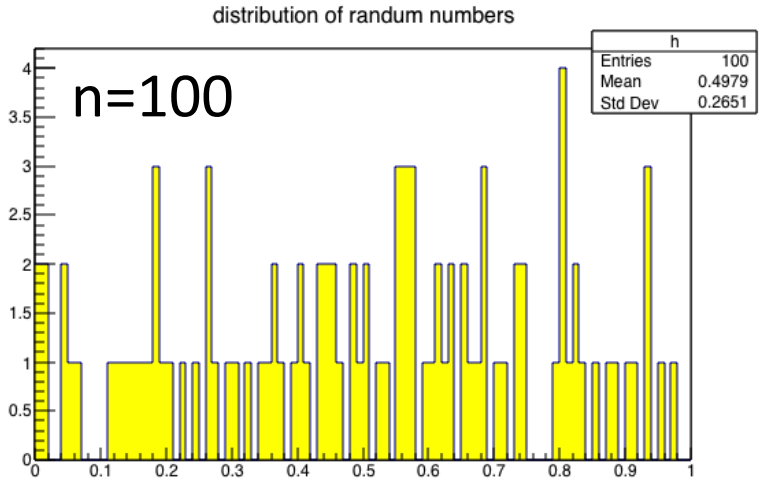
NR: Numerical Recipes
GSL: GNU Scientific Library

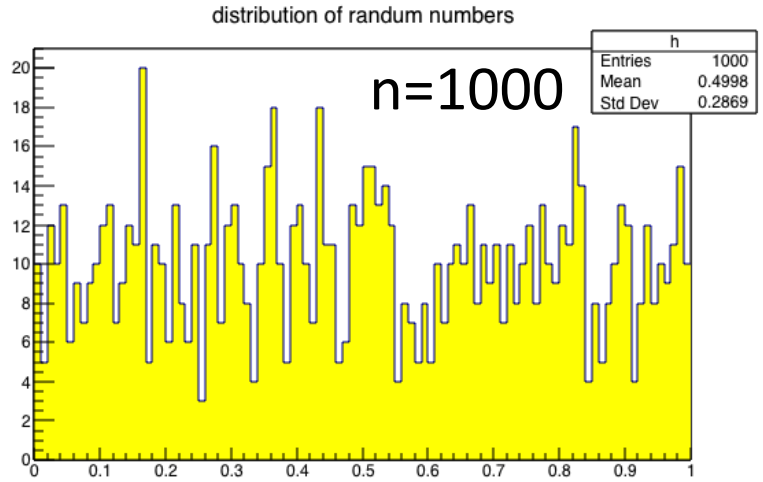
Library	Generator	Relative speed	Period
NR	RAN0	1.0	$\sim 2^{31}$
NR	RAN1	1.3	$\sim 2^{36}$
NR	RAN2	2.0	$\sim 2^{62}$
NR	RANQD2	0.25	$\sim 2^{30}$
GSL	MT19937	0.8	$\sim 2^{19937}$
GSL	TAUS	0.6	$\sim 2^{88}$
GSL	RANLXD2	8.0	$\sim 2^{400}$

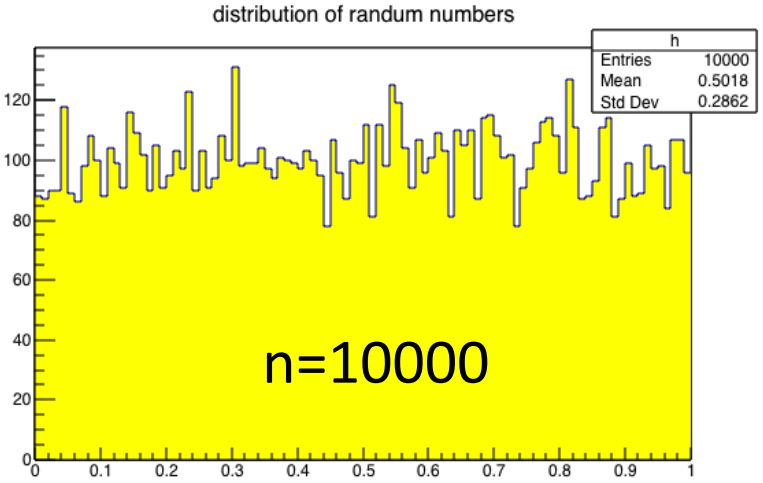
Always use GSL! See the GSL doc for the many more algorithms available

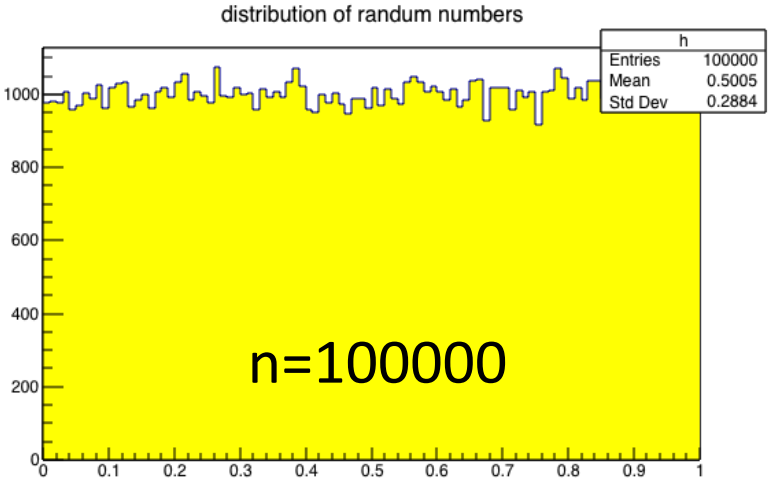
Eg1) Uniform distribution with random numbers

- Make a program to produce a uniform distribution [0,1] for n=100, 1000, 10000 and 100000
  - Use 'ran0' function in ran0.c file, e.g.) x = ran0(&seed);
  - gRandom->Rndm() in ROOT

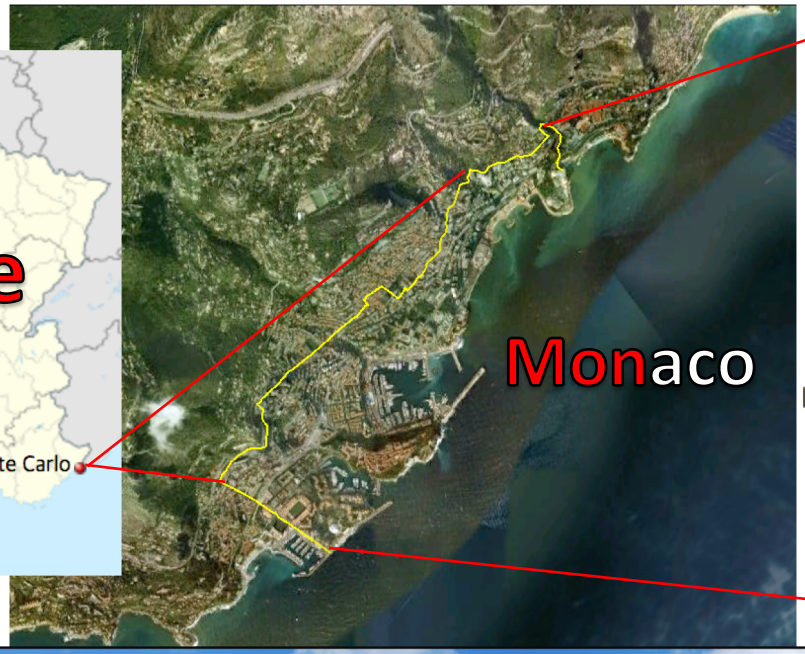


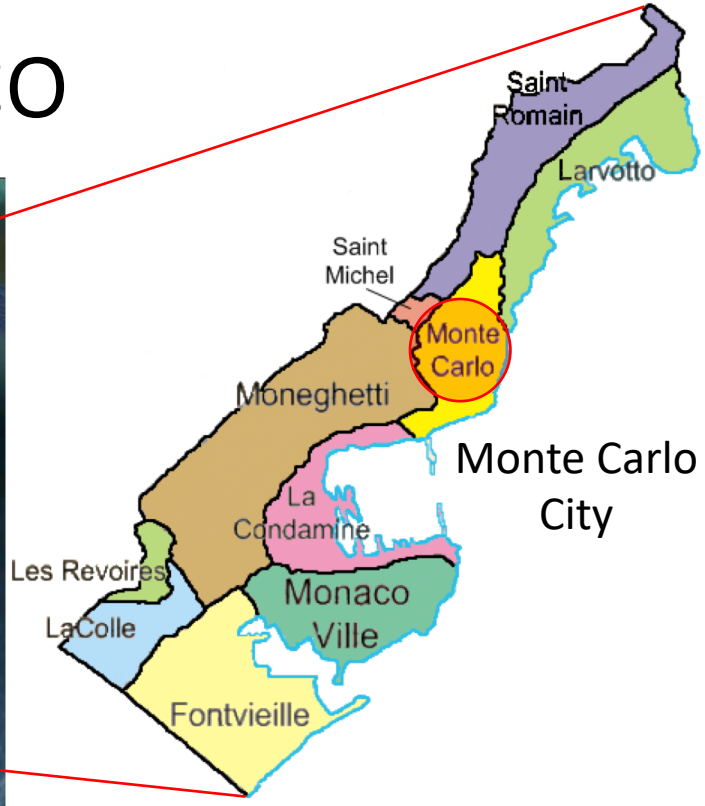






Monte Carlo at Monaco





Monte Carlo (MC) Method

- Broad class of computational algorithms that rely on repeated random sampling to obtain numerical results.
  - Their essential idea is using randomness to solve problems that might be deterministic in principle.
- In physics-related problems, Monte Carlo methods are useful for simulating systems with many coupled degrees of freedom, such as fluids, disordered materials, strongly coupled solids, and particle physics.
- Monte Carlo method was invented in the late 1940s by Stanislaw Ulam and Von Neumann, while they was working on nuclear weapons projects at the Los Alamos National Laboratory
  - The project was top secret so they chose the name Monte Carlo in reference to the Casino in Monaco.

Monte-Carlo methods generally follow the following steps:

1. Determine the statistical properties of possible inputs
2. Generate many sets of possible inputs which follows the above properties
3. Perform a deterministic calculation with these sets
4. Analyze statistically the results

✓ The error on the results typically decreases as $1/\sqrt{n}$ if n experiments are performed.

Why use the Monte Carlo method?

- multiple integration

$$I = \int_c^d \int_a^b f(x, y) dx dy \, , \ldots$$

  - If the number of grids in each dimension is *n*, the number of grids in N dimension is n^N.
  - The calculation time increases drastically with respect to *n* and *N*. e. g.) If the calculation time of 1 point is 1 msec and the number of grids in each dimension is 1000,
    - 1 sec for 1D
    - 17 mins for 2D
    - 12 days for 3D
    - 31 years for 4D

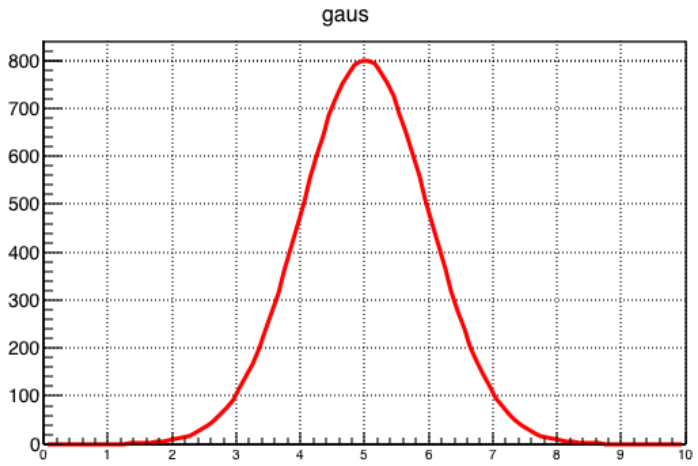
Monte Carlo method

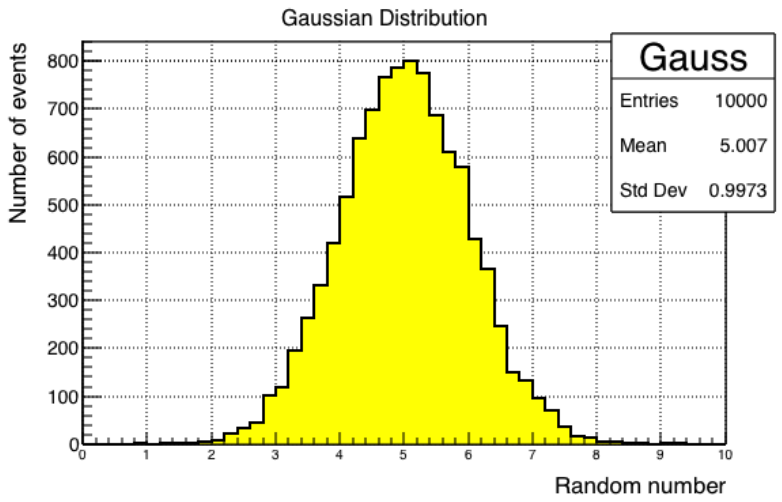
- Do not calculate over all the grid points
  - ➢ but calculate the values of randomly-selected grids until the result converges to a value.
  - ➢ Saving CPU time significantly.

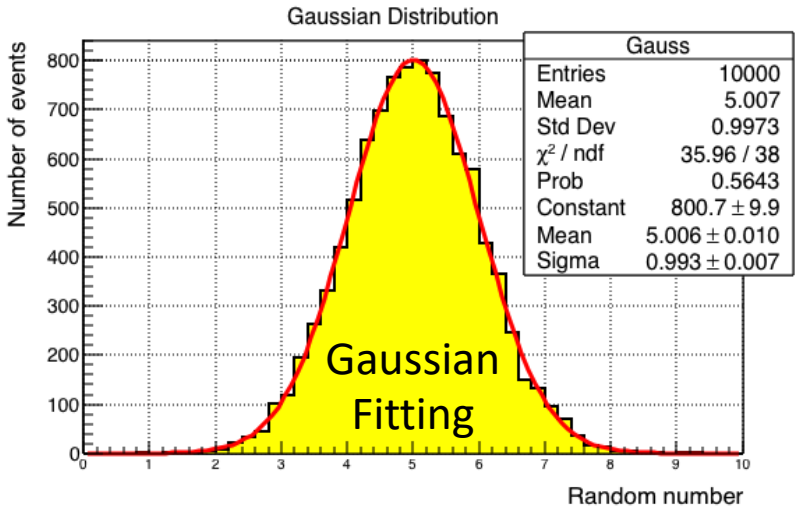
- The calculation errors are proportional to $N^{1/2}$ for N grid points.

Simple example for MC Method

- Gaussian distribution with mean μ=5, σ=1. Choose random numbers in the probability function of the Gaussian distribution.







Monte Carlo method

- Example: calculation of π [Program: lect5-2.c]

a) (x, y) is randomly generated within the range $-1 \le x \le 1,\ -1 \le y \le 1$

→ Use 'ran0' function in ran0.c file, e.g.) x = ran0(&seed);

b) Count the number n of the cases that satisfy $x^2 + y^2 \le 1$

c) Calculate the ratio n / N, where N is total trial number.

d) Repeat (a)-(c) until n / N converges to a value.

Examples

- *Weibull distribution*. Let $\alpha, \lambda > 0$ then the Weibull cdf is given by

$$F(x) = 1 - \exp\left(-\lambda x^{\alpha}\right), \ x \geq 0.$$

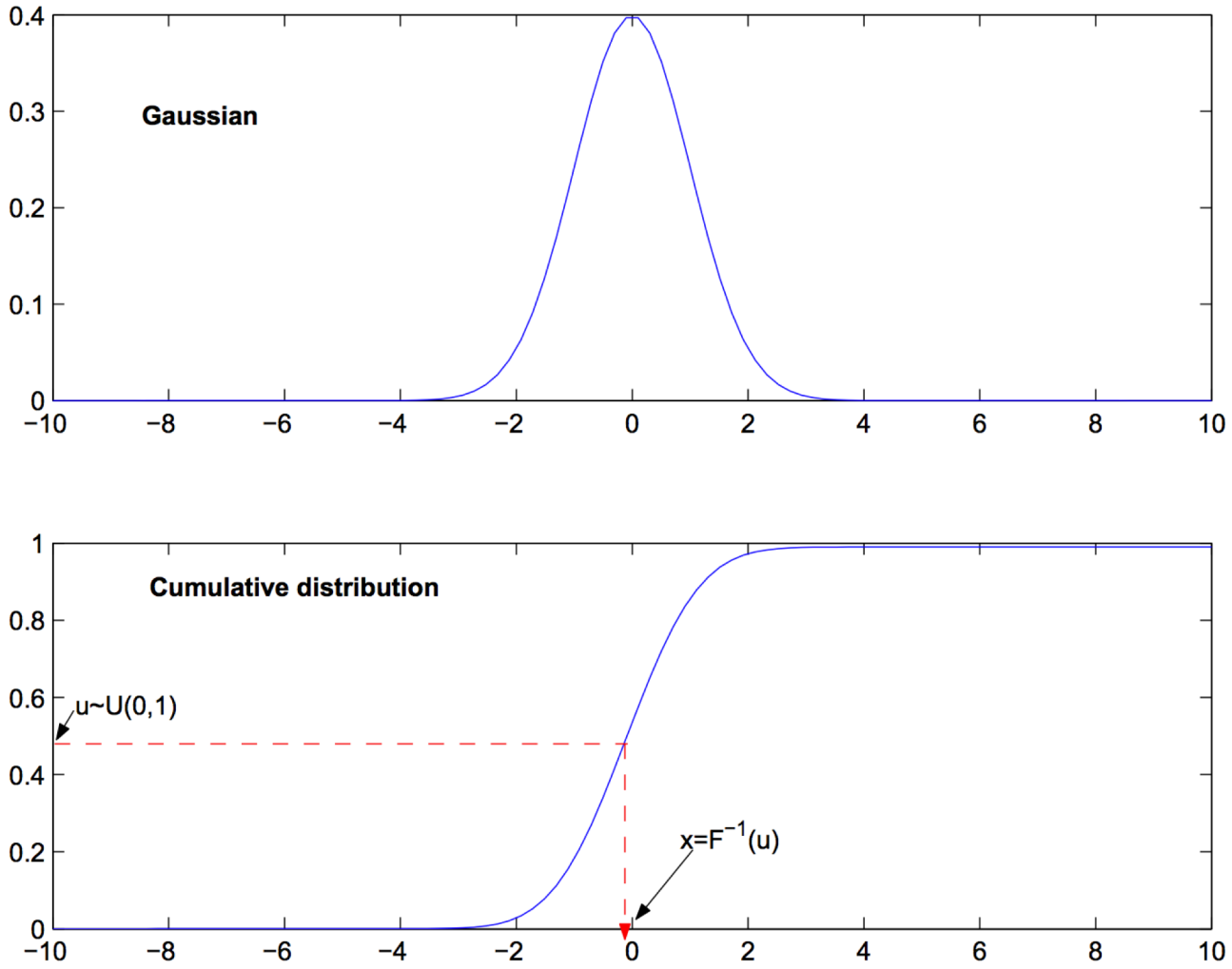
We calculate

$$\begin{aligned} u &= F(x) \Leftrightarrow \log(1-u) = -\lambda x^{\alpha} \\ &\Leftrightarrow x = \left(-\frac{\log(1-u)}{\lambda}\right)^{1/\alpha}. \end{aligned}$$

- As $(1-U) \sim \mathcal{U}[0,1]$ when $U \sim \mathcal{U}[0,1]$ we can use

$$X = \left(-\frac{\log U}{\lambda}\right)^{1/\alpha}.$$

The transformation method



Top: pdf of a normal, bottom: associated cdf.

(cumulative distribution function)

The transformation method

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

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

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

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

The transformation method

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

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

Example 1: $f(y) = ae^{-ay},\ y \in [0, \infty)$

$$\left|\frac{dx}{dy}\right| = ae^{-ay}, \quad \text{or,} \quad x = e^{-ay}, \quad \text{i.e.,} \quad \boxed{y = \frac{-\ln(x)}{a}}$$

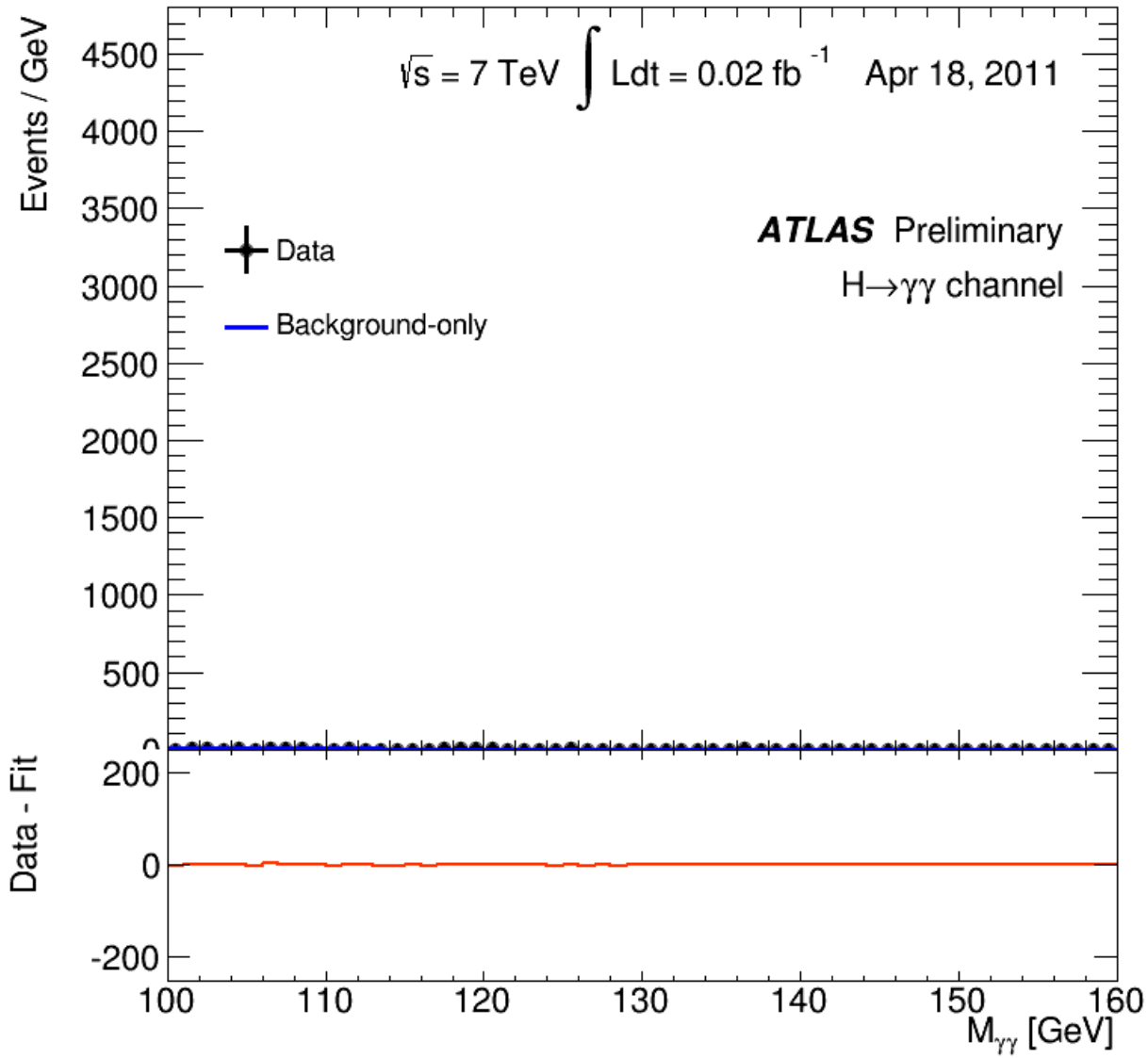
Example 2: $f(y) = \frac{y^{-1/2}}{2},\ y \in [0, 1]$

$$\left|\frac{dx}{dy}\right| = \frac{y^{-1/2}}{2}, \quad \text{or} \quad x = y^{1/2}, \quad \text{i.e.,} \quad \boxed{y = x^2}$$

Note that in this case we are sampling a probability density that is infinite at 0, but that is OK!

H->γγ : Example ATLAS (CMS similar)

Transformation method

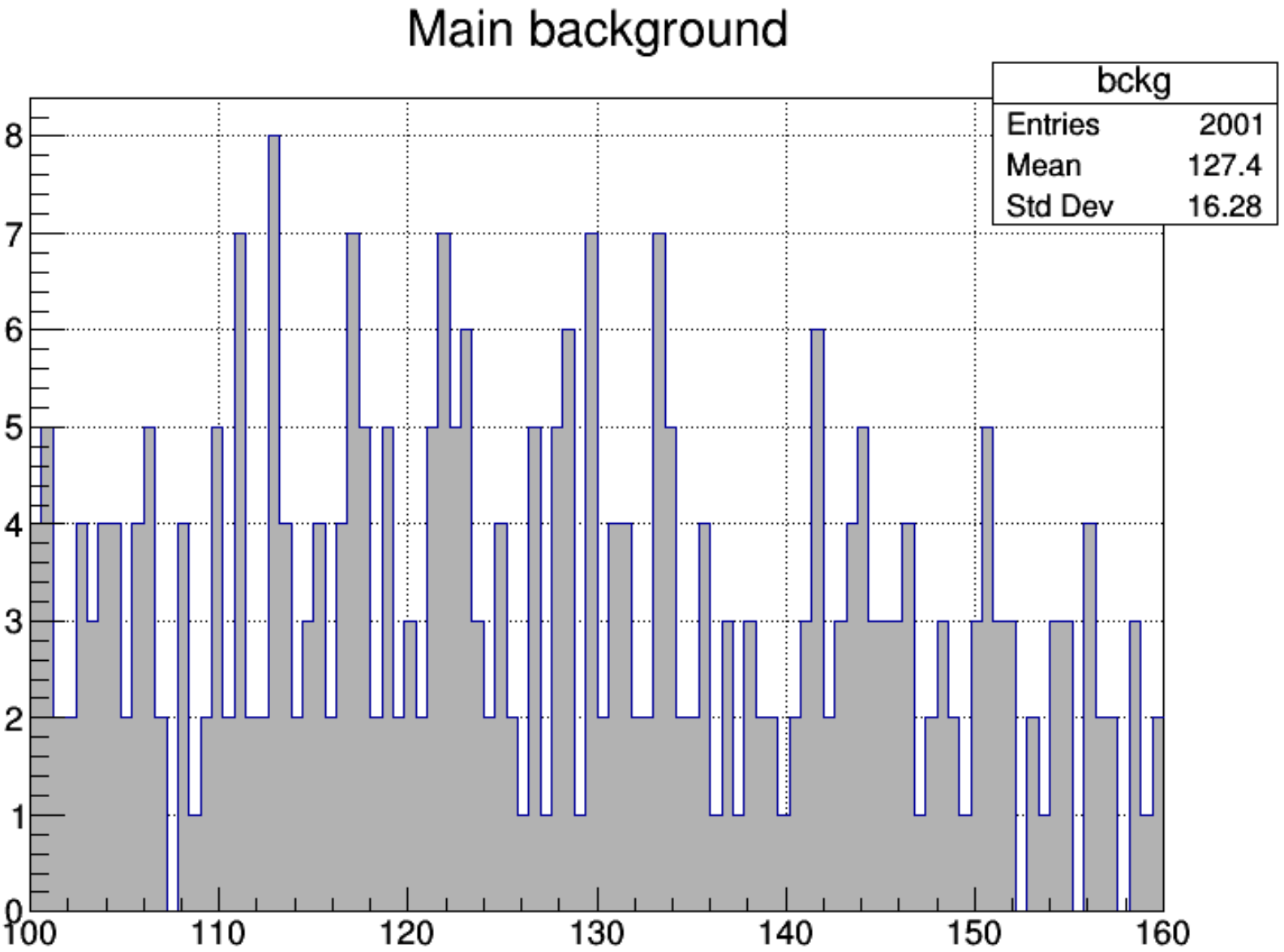


Eg2) transformation method

- Program lect5_3_bckg.C

```
1  void lect5_3_bckg()
2  {
3    TCanvas *c1 = new TCanvas("c1","The MC example",200,10,800,600);
4    c1->SetGrid();
5
6    TH1F *bckg  = new TH1F("bckg","Main background",100,100,160);
7    bckg->SetFillColor(16);
8    gSystem->Unlink("bckg.gif"); // delete old file
9
10   // Fill histograms randomly
11   gRandom->SetSeed();
12   const int kUPDATE = 2000;
13   float xbckg;
14   int seed;
15   for (int i=0; i<1000000; i++) {
16     xbckg = -log ( gRandom->Rndm(seed) ) * 100.;
17     bckg->Fill(xbckg);
18     if (i && (i%kUPDATE) == 0) {
19       if (i == kUPDATE) {
20         bckg->SetMinimum(0);
21         bckg->Draw("same");
22         c1->Update();
23       }
24       c1->Modified();
25       c1->Update();
26       c1->Print("bckg.gif+");
27       printf("i = %d\n", i);
28     }
29   }
30   bckg->Draw("hist"); // to redraw axis hidden by the fill area
31   c1->Modified();
32   c1->Print("bckg.gif++");
33 }
```

- % root lect5_3_bckg.C

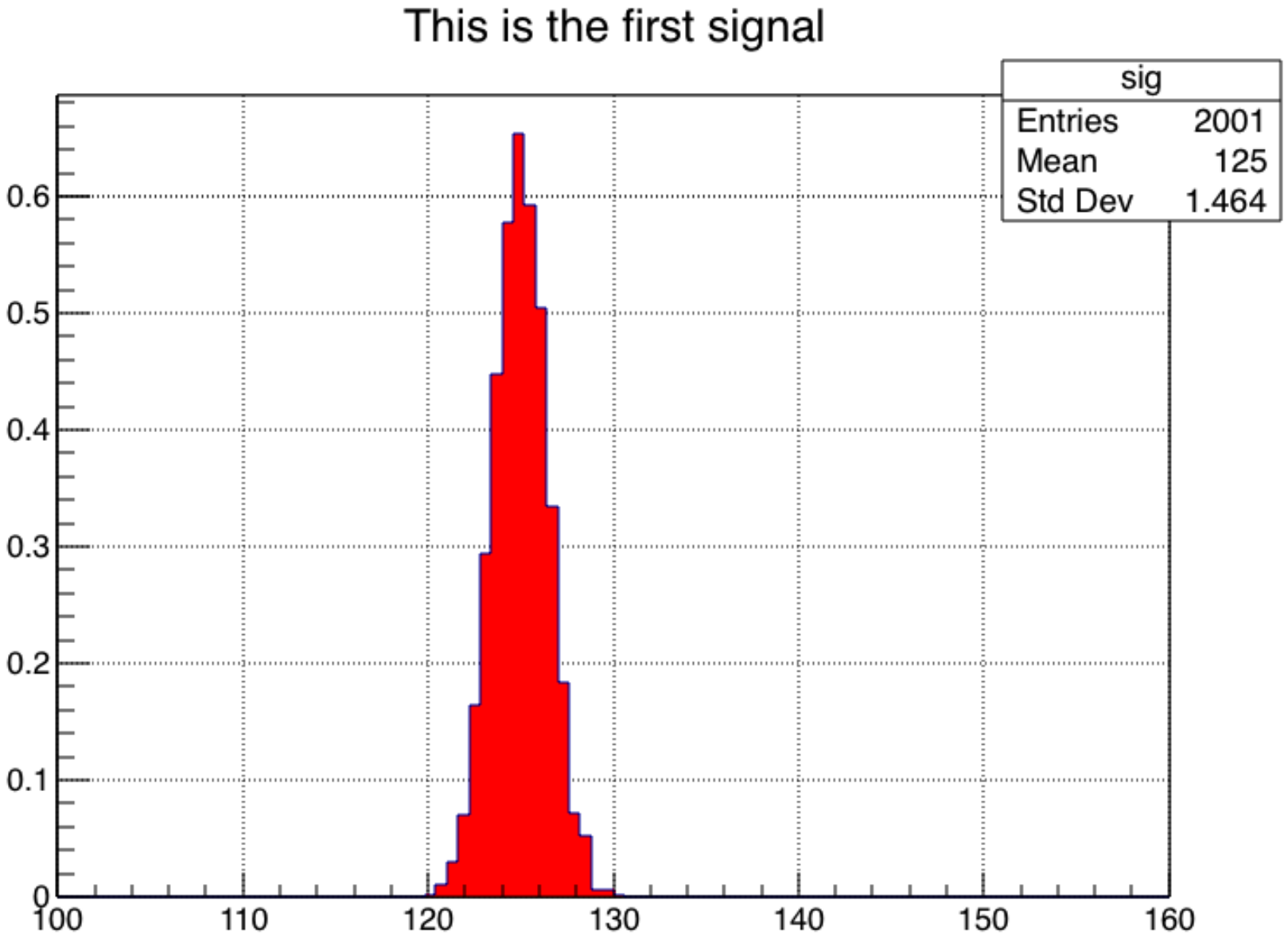


Eg3) transformation method

- Program lect5_3_sig.C

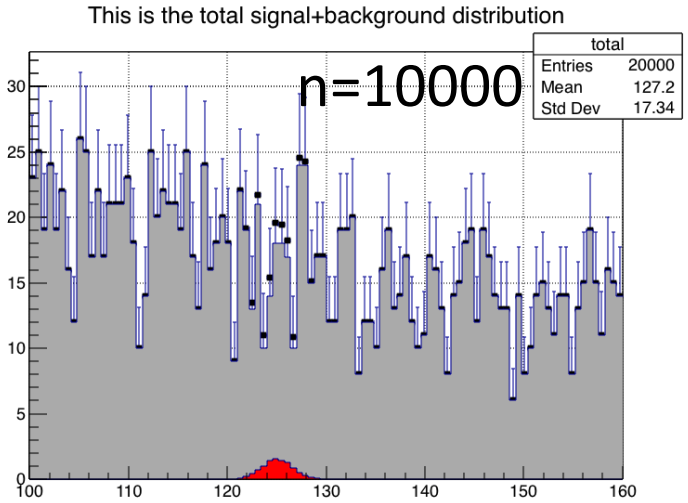
```
1  void lect5_3_sig()
2  {
3    TCanvas *c1 = new TCanvas("c1","The MC example",200,10,800,600);
4    c1->SetGrid();
5
6    TH1F *sig = new TH1F("sig","This is the first signal",100,100,160);
7    sig->SetFillColor(2);
8    gSystem->Unlink("sig.gif"); // delete old file
9
10   // Fill histograms randomly
11   gRandom->SetSeed();
12   const int kUPDATE = 2000;
13   float xsig;
14   int gifcnt = 0;
15   for (int i=0; i<100000; i++) {
16     xsig   = gRandom->Gaus(125,1.5);
17     sig->Fill(xsig,0.002);
18     if (i && (i%kUPDATE) == 0) {
19       if (i == kUPDATE) {
20         sig->Draw("hist same");
21         c1->Update();
22       }
23       c1->Modified();
24       c1->Update();
25       c1->Print("sig.gif+");
26       printf("i = %d\n", i);
27     }
28   }
29   sig->Draw("hist"); // to redraw axis hidden by the fill area
30   c1->Modified();
31   // make infinite animation by adding "++" to the file name
32   // You can view the animated file sig.gif with Netscape/IE or mozilla
33   c1->Print("sig.gif++");
34 }
```

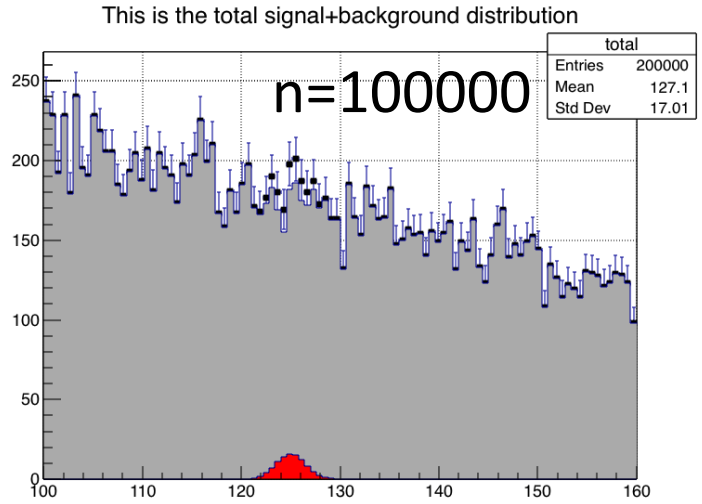
- % root lect5_3_sig.C

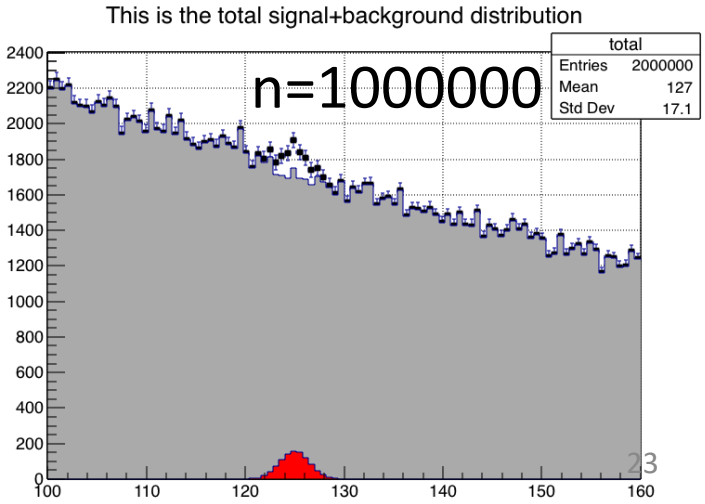


Eg4) transformation method

- Produce below signal in the Gaussian distribution and background in the Exponential distribution using ROOT framework

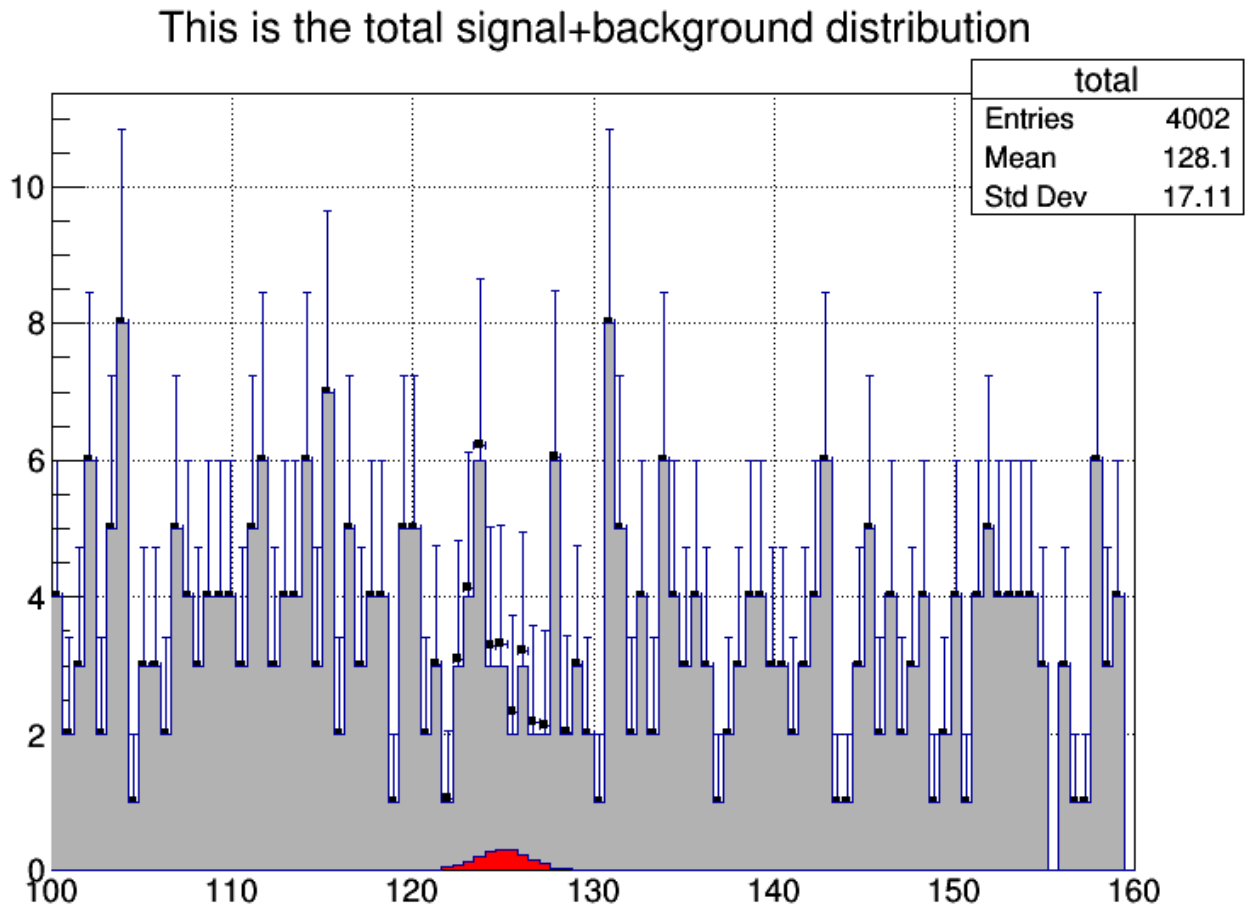






Eg4) transformation method

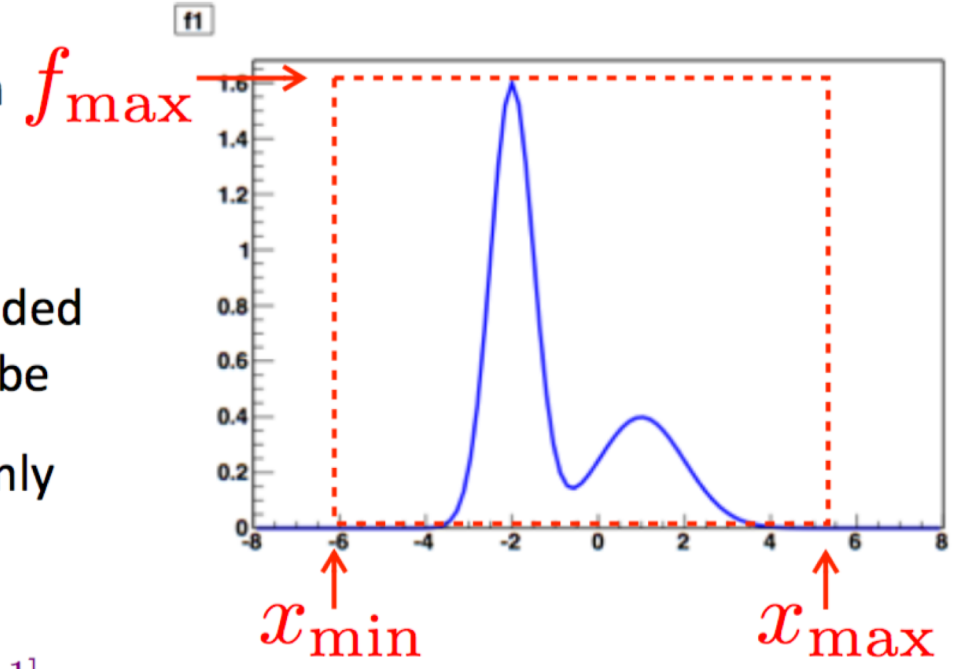
- Produce below signal in the Gaussian distribution and background in the Exponential distribution using ROOT framework



The 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.

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

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

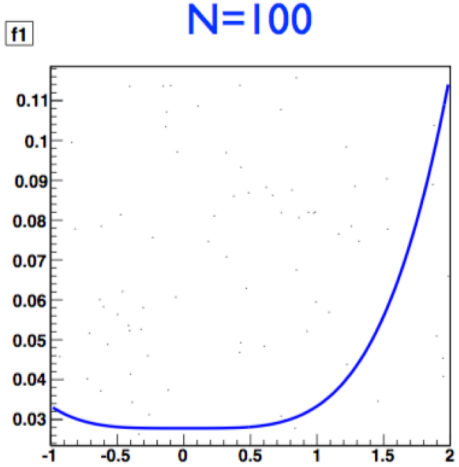
$$u = r_2 f_{\max}$$

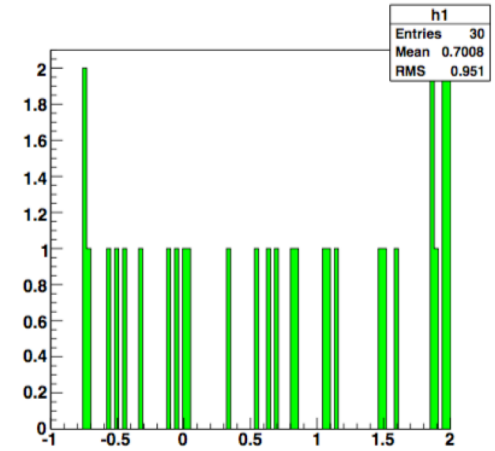
3) If $u < f(x)$, then accept *x*. If not, reject *x* and repeat.

The rejection method

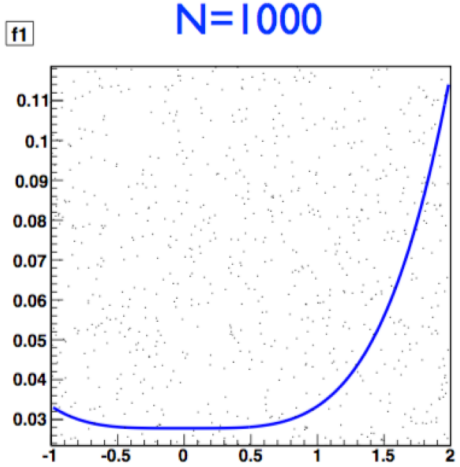
ex) $f(x) = \frac{1}{36}\left(5x^4 + 1\right)$ in $[-1, 2]$

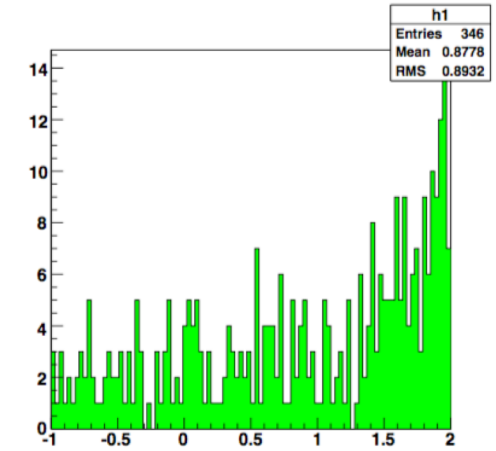
N=100



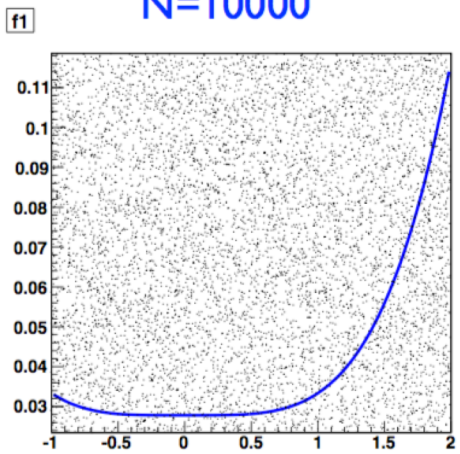


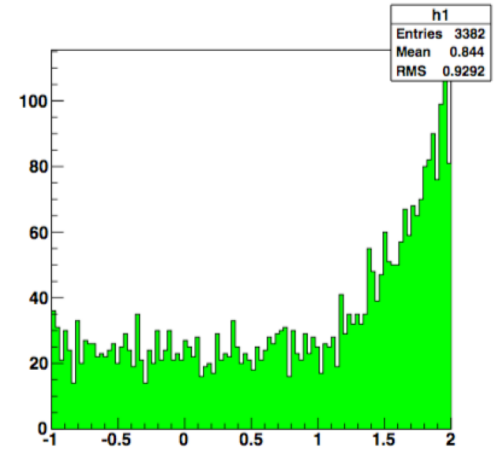
N=1000





N=10000





Eg5) transformation method

- Calculate integral of f(x) using the rejection method.

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