Simulations with ROOT- 2

General distributions

- In general it is not sufficient to have uniform random numbers.
- In many problems it is necessary to have number distributed according to other p.d.f. (e.g. Gaussian, exponential, Poisson, ...)
- IN ROOT are available in the TRandom class generators with several p.d.f.
  - Binomial
  - BreitWigner
  - Circle
  - Exp
  - Gauss
  - Landau
  - Poisson
  - Rannor
  - Rndm
  - Sphere
  - Uniform

General distributions

- But, what if you want to generate a number x_i distributed according to a certain distribution $\mathbf{f}(\mathbf{x})$?

- It is possible to use at least two techniques:

  – Rejection

  – Inversion

Inversion Method

- Inversion method
  - This method is applicable for relatively simple (i.e. can be easily inverted) distribution functions:
    - Normalize the distribution function, so that it becomes a “probability distribution function”
    - Integrate the PDF analytically from minimum x (x_{min}) to an arbitrary x (x)
      - ➢ This represents the probability of choosing a value less than x
    - Equate this to a uniform random number and solve for x, given a uniform random number λ

$$\frac{\int_{x_{min}}^{x} f(x)\,dx}{\int_{x_{min}}^{x_{max}} f(x)\,dx} = \lambda$$

This method is fully efficient, since each random number λ gives an x value

Inversion Method

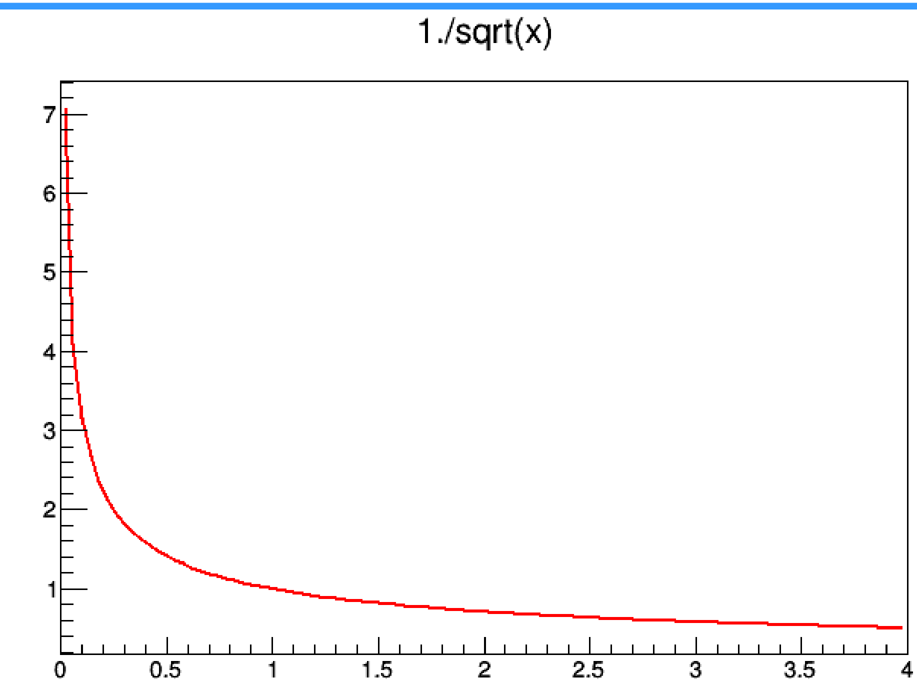
- Example: Generate x between 0 and 4 according to:

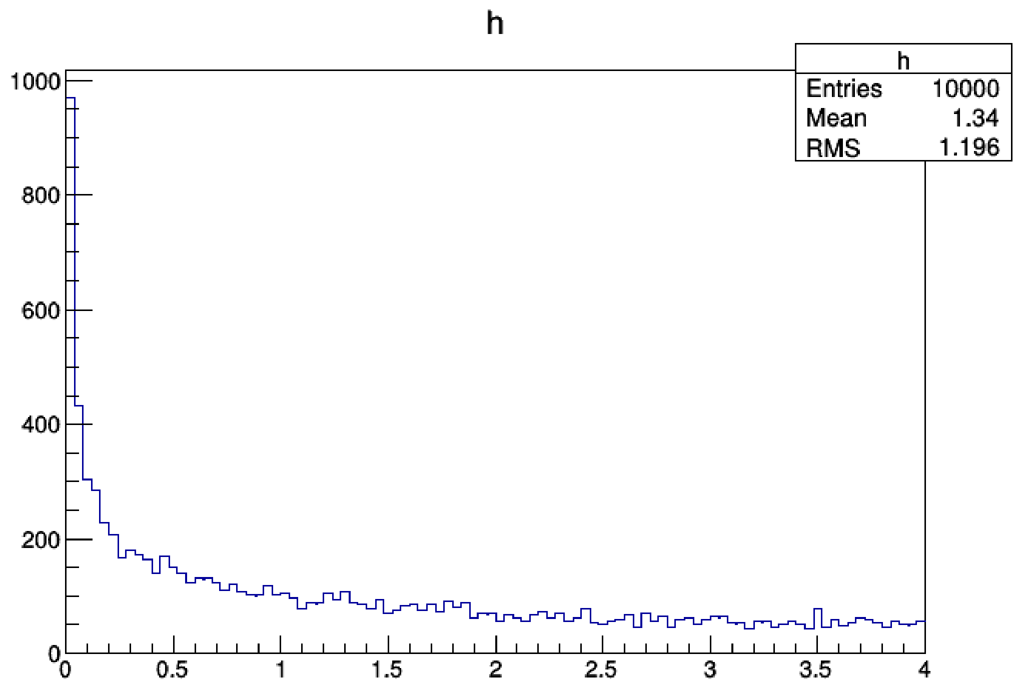
$$f(x) = \frac{1}{\sqrt{x}}$$

$$\frac{\int_{x_{min}}^{x} x^{-\frac{1}{2}}\,dx}{\int_{0}^{4} x^{-\frac{1}{2}}\,dx} = \lambda$$

$$\frac{x_{min}^{1/2} - 2\,x^{1/2}}{0 - 2\cdot 4^{1/2}} = \lambda = \frac{x^{1/2}}{2}$$

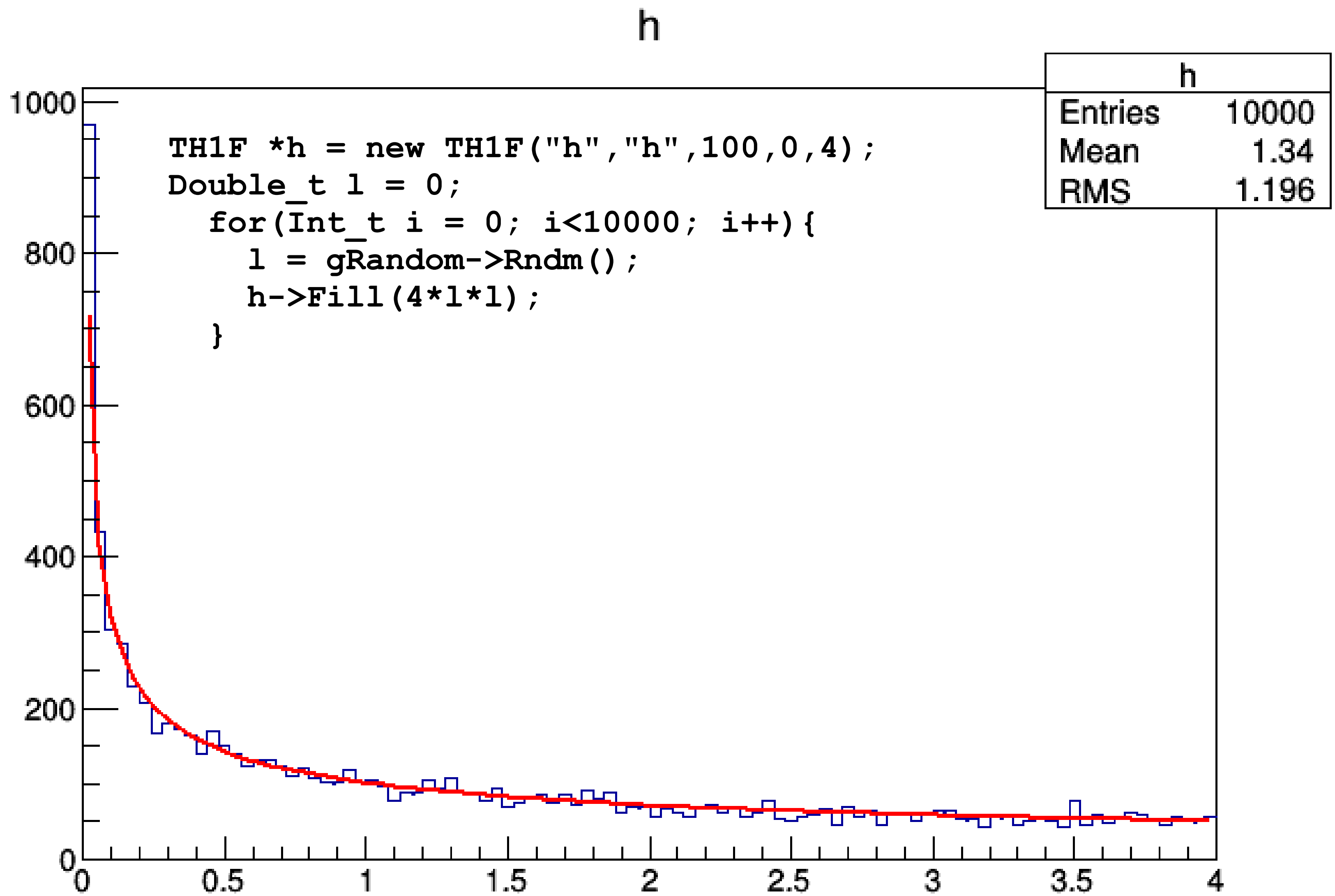
$\Rightarrow$ Generate x according to $\mathbf{x} = 4\lambda^2$





Inversion.C

Inversion Method : Results for 10000 trials



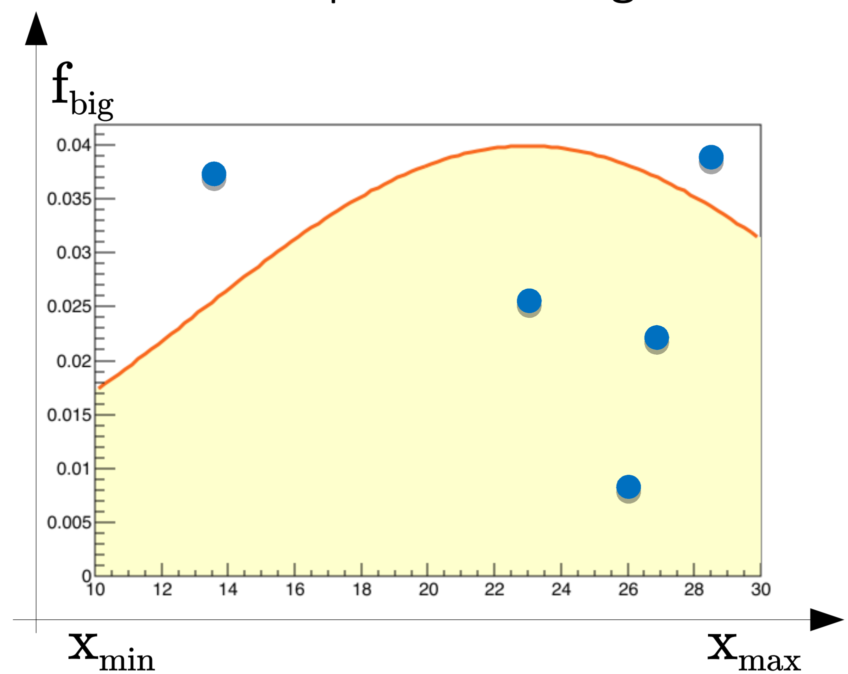
Rejection Method

- Algorithm:
  - Chose trial x, given a uniform random number λ_1:

$$\mathrm{x}_{\mathrm{trial}} = \mathrm{x}_{\mathrm{min}} + (\mathrm{x}_{\mathrm{max}} - \mathrm{x}_{\mathrm{min}})\ \lambda_1$$

  - Decide whether to accept the trial value:
    - If $\mathrm{f}(\mathrm{xtrial})$ > $\lambda_2 \mathrm{f}_{\mathrm{big}}$ then accept

    Where $\mathrm{f}_{\mathrm{big}} \geq \mathrm{f}(\mathrm{x})$ for all $\mathrm{x},\ \ \mathrm{x}_{\mathrm{min}} \leq \mathrm{x} \leq \mathrm{x}_{\mathrm{max}}$.
  - Repeat the algorithm until the trial value is accepted. This algorithm can be visualized as throwing darts



Rejection Method

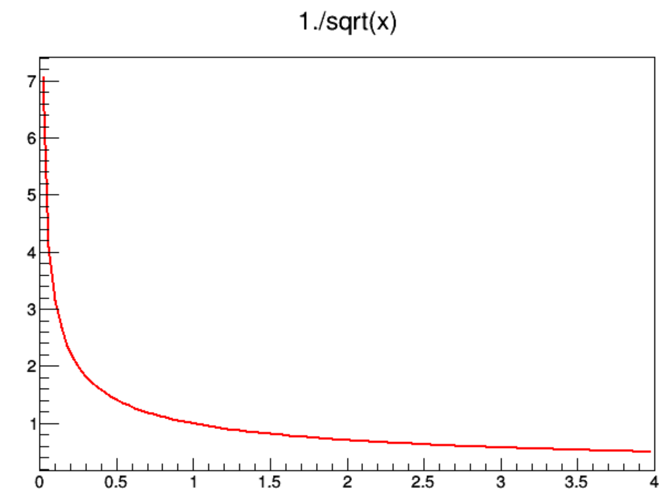
- u_1, u_2 are two numbers distributed according to a uniform distribution in [0,1]
  x_T, y_T are extracted:
  - $x_T = x_{min} + (x_{max} - x_{min})\, u_1$
  - $y_T = f_{big}\, u_2$, with $f_{big} \geq f(x)\ \forall\ x \in [x_{min}, x_{max}]$
- x_T accepted if $f(x_T) > y_T$

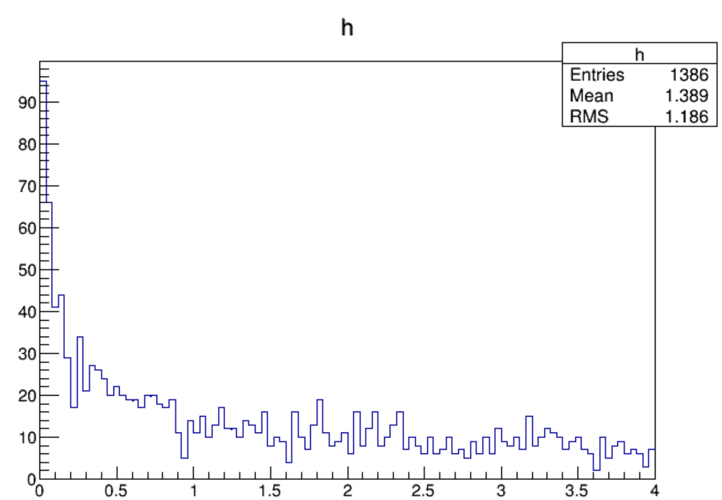
Rejection Method : Example

- Example: Generate x between 0 and 4 according to:

$$f(x) = \frac{1}{\sqrt{x}}$$

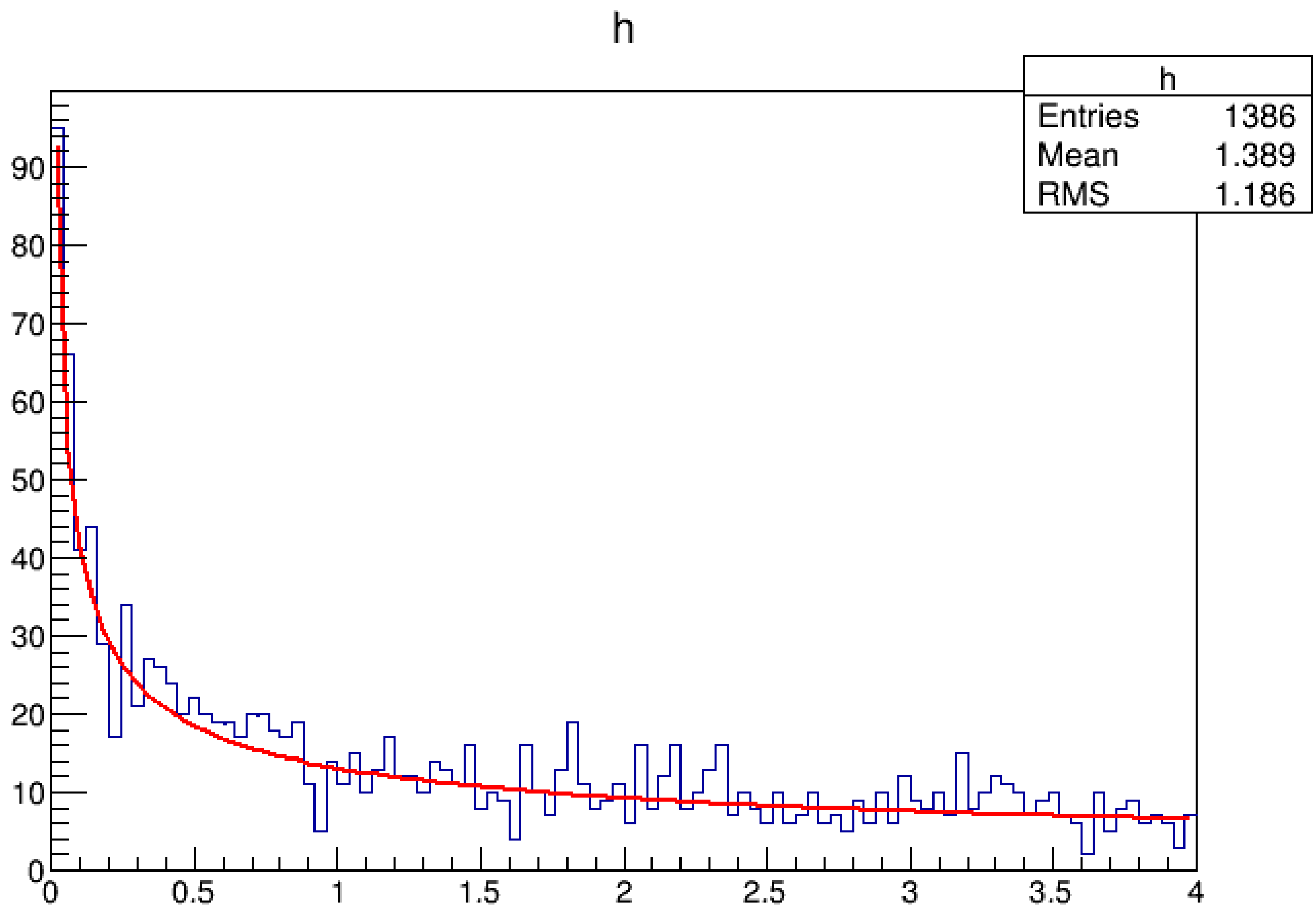
```
TF1 *f1 = new
TF1("f1","1./sqrt(x)",0,4);
Double_t fMax = 4;

Double_t u1 = 0;
Double_t u2 = 0;

Double_t xT = 0;
Double_t yT = 0;
TH1F *h = new TH1F("h","h",100,0,4);
for(Int_t i = 0; i<10000; i++){
   u1 = gRandom->Rndm();
   u2 = gRandom->Rndm();

   xT = xmin + (xmax-xmin)*u1;
   yT = u2*fMax;
   if(f1->Eval(xT) > yT)
    h -> Fill(xT);
}
```





Reiezione.C

Rejection Method : Results for 10000 trials



Rejection Method : Integral

- This procedure also gives an estimate of the integral of f(x)

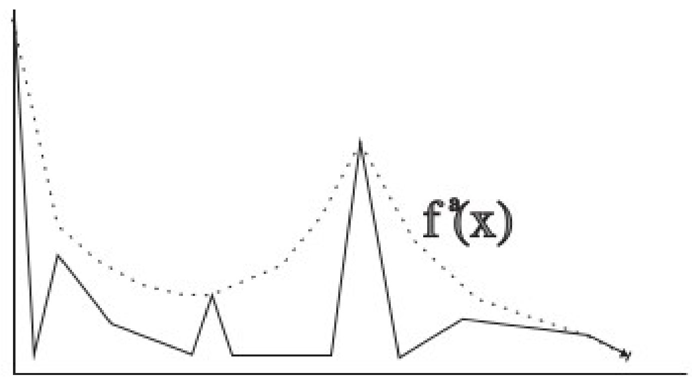
$$I = \int_{x_{min}}^{x_{max}} f(x)\,dx \approx \frac{n_{accept}}{n_{trial}} f_{big}(x_{min} - x_{max})$$

Limits of the rejection method

- In general this method has a limited efficiency
- Is not suited if the function presents peaks
- Cannot be used if the function have poles or integration limits that tend to ∞
  - What if the rejection technique is impractical and you can't invert the integral of the distribution function?

Importance sampling

- **Importance Sampling**: replace the distribution function f(x) by an approximate form $f^a(x)$ for which the inversion technique can be applied.
- Generate trial values for x with inversion technique according to $f^a(x)$, and accept the trial value with the probability proportional to the weight:



- The rejection technique is just a special case where $f^a(x)$ is chosen to be constant

Esercitazione 12 - Exercise 1

- Write a class that inherits with `public inheritance` from the ROOT `TRandom3` class. In the class, the inversion and rejection methods for the function

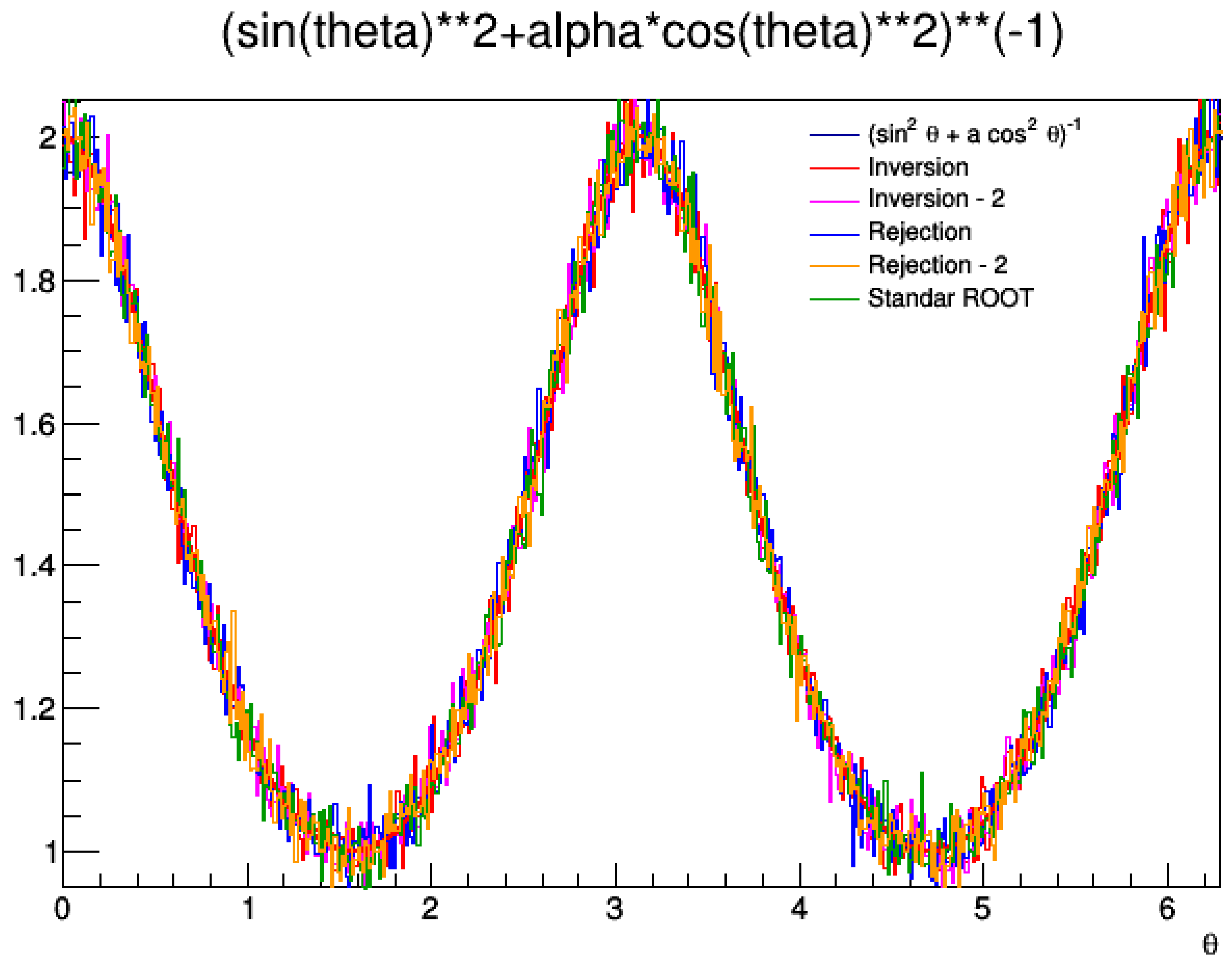
$$\mathrm{f}(\theta) = (\sin^2\theta + a\cos^2\theta)^{-1}$$

  in the range $0 \leq \theta \leq 2\pi$ have to be implemented as two *class function.*

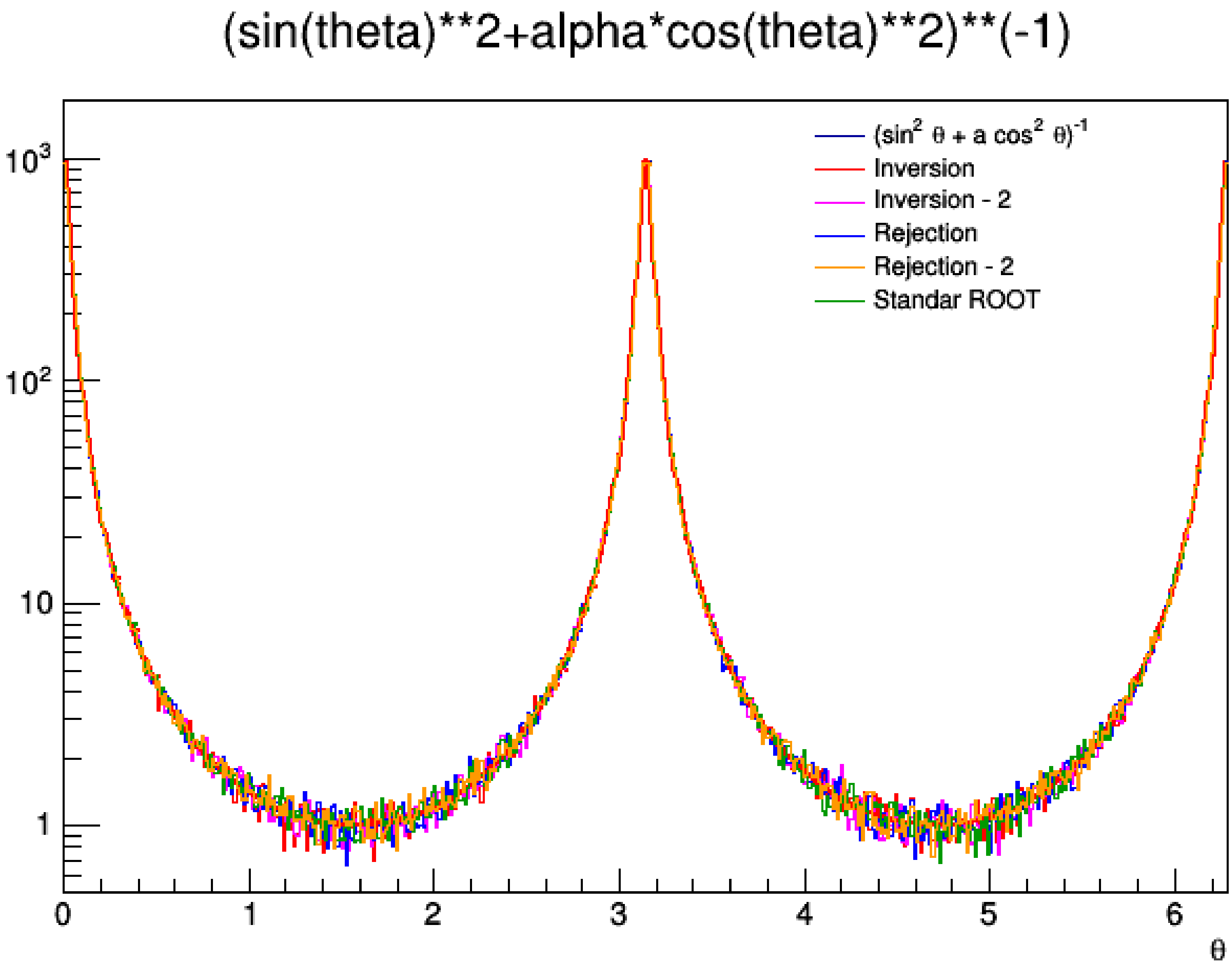
- Write a macro that uses the implemented class and compare the rejection and the inversion technique:
  - Generate 1000000 values for each method using a = 0,5 and a = 0,001
  - Plot the results obtained for each a and overlay the distribution curves $\mathrm{f}(x)$ properly normalized
  - Compare the CPU time request for the 4 runs (hint: in ROOT it is possible the use the `TStopwatch` class)

`MyRandom3.{h, cxx} InversionRejection.C`

Result for a = 0,5



Result for a = 0,001



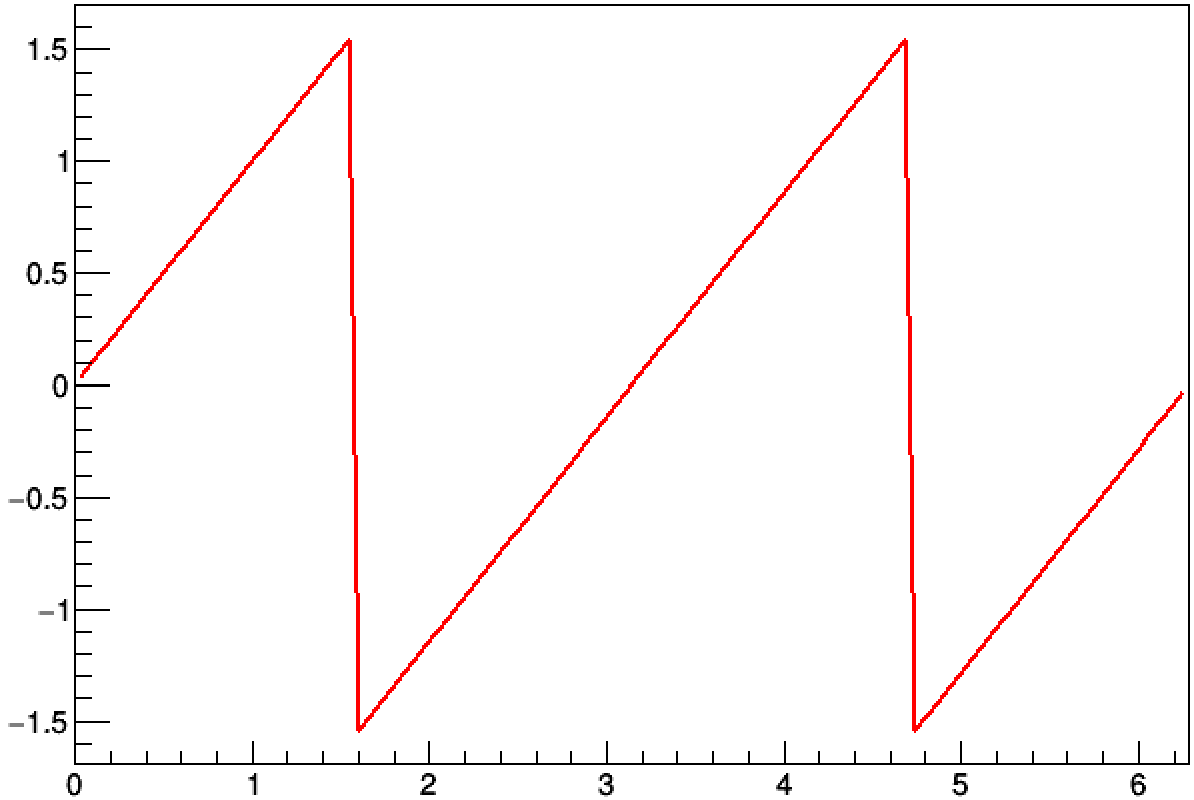
Execution time

```
root [0] .L MyRandom3.cxx+
root [1] .L InversionRejection.C+
root [2] InversionRejection(0.5)
Pararameter alpha = 0.5
Number of bins= 500, Bin size = 0.0125664
Number of extracted numbers: 1e+06
CPU time inversion method (assolute / relative) 0.3/0.9375
CPU time inversion method BIS                   0.23/0.71875
CPU time rejection method                       0.32/1
CPU time rejection method (recursive)           0.32/1
CPU time standard ROOT via TF1                  0.09/0.28125
root [3] Info in <TCanvas::Print>: file /home/ramona/Dropbox/C++/Esercizi/Esercitazione11/

root [3] .q
ramona@ramona-SVS13A1X9ES ~/Dropbox/C++/Esercizi/Esercitazione11 $ root -l
root [0] .L InversionRejection.C+
root [1] InversionRejection(0.001)
/bin/root.exe: symbol lookup error: /home/ramona/Dropbox/C++/Esercizi/Esercitazione11/./In
ramona@ramona-SVS13A1X9ES ~/Dropbox/C++/Esercizi/Esercitazione11 $ root -l
root [0] .L MyRandom3.cxx+
root [1] .L InversionRejection.C+
root [2] InversionRejection(0.001)
Pararameter alpha = 0.001
Number of bins= 500, Bin size = 0.0125664
Number of extracted numbers: 1e+06
CPU time inversion method (assolute / relative) 0.31/0.0461997
CPU time inversion method BIS                   0.22/0.0327869
CPU time rejection method                       6.71/1
CPU time rejection method (recursive)           6.78/1.01043
CPU time standard ROOT via TF1                  0.08/0.0119225
```

Notes of the Inversion method (exercise)

The integral function contain the $\arctan$ function: this function return values between -π/2 e π/2.

If we represent the function we have a periodic function:



Notes of the Inversion method (exercise)

- The function $\mathbf{f(x)}$ is periodic and has to be integrated with an appropriated normalization factor

$$F(x) = k\int_{-\frac{\pi}{2}}^{x} \frac{d\theta}{a\cos^2\theta + \sin^2\theta} = \frac{k}{a}\int_{-\frac{\pi}{2}}^{x} \frac{d\theta}{a\cos^2\theta\left(1 + \frac{\tan^2\theta}{a}\right)}$$

$$z \equiv \frac{\tan\theta}{\sqrt{a}} \Rightarrow dz = \frac{1}{\sqrt{a}\cos^2\theta} d\theta$$

$$F(x) = \frac{k}{\sqrt{a}} \int_{-\infty}^{\frac{\tan x}{\sqrt{a}}} \frac{dz}{1+z^2} = \frac{k}{\sqrt{a}} atan\left(\frac{\tan x}{\sqrt{a}}\right) + \frac{k}{\sqrt{a}}\frac{\pi}{2}$$

Notes of the Inversion method (exercise)

- The normalization constant is

$$F\left(x \rightarrow \frac{\pi}{2}\right) = \frac{k}{\sqrt{x}}\frac{\pi}{2} + \frac{k}{\sqrt{x}}\frac{\pi}{2} \equiv 1 \Rightarrow k = \frac{\sqrt{a}}{\pi}$$

- If you extract u with a uniform distribution between 0 and 1 you can obtain a requested function as

$$u = \frac{1}{\pi}\arctan\left(\frac{\tan x}{\sqrt{a}}\right) + \frac{1}{2} \Rightarrow x = \arctan\left[\sqrt{a}\tan\left(\pi u - \frac{\pi}{2}\right)\right]$$

- To move the function in the [0,2π] interval :
    - Extract a second number w uniformly distributed in [0,1]
        - If w < 0.5 → x = x+π (2nd and 3rd quadrant)
        - Else
            - if x<0 → x+=2π (4th quadrant)
        - Else
            - if X>=0 → x = x (1st quadrant)

```
#ifndef MYRANDOM3_H
#define MYRANDOM3_H


#include "TRandom3.h"

class MyRandom3 : public TRandom3 {
////////////////////////////////////////////////////////////////////////
// Class used to generate random numbers.
// New function(s) to be sampled are added w.r.t. TRandom3
// Origin: M.Masera 17/10/2002
// Last mod. 16/10/2017
////////////////////////////////////////////////////////////////////////
 public:
  //
  MyRandom3();
  MyRandom3(double alpha, UInt_t seed=65539);
  // Funct1(theta,alpha) returns the value of f(x,a)=1/(sin(x)**2+a*cos(x)**2)

  virtual ~MyRandom3(); // descructor
  double Funct1(double theta);

  // returns a random number distributed according to Funct1
  // with the inversion method
  double Funct1RndmByInversion();

  // returns a random number distributed according to Funct1
  // with the inversion method (another implementation)
  double Funct1RndmByInversion2();

  // returns a random number distributed according to Funct1
  // with the rejection method
  double Funct1RndmByRejection();
  double Funct1RndmByRejection2();

  //
 private:
  // private methods
  void Init();  // set alpha parameter

  // data members
  double fAlpha;      //! parameter of the function
  double fPi;         //! PI
  double fBig;        //! used by rejection method
  Double_t fSqrtAlpha;  //! sqrt(fAlpha)

  ClassDef(MyRandom3,1)
};


#endif
```

Public Inheritance from the TRandom3 class

Default and Copy Constructor

Function definition

Alternative method to initialize data members outside the constructor

Exercise 1 - Easy version

- Write a <u>macro</u> that implement the inversion and rejection method for the function

$$\mathbf{f}(\theta) = (\sin^2\theta + a\cos^2\theta)^{-1}$$

  in the range $0 \leq \theta \leq 2\pi$.

- Compare the rejection and the inversion technique:
  - Generate 1000000 values for each method using $\mathbf{a}$ = 0,5 and $\mathbf{a}$ = 0,001
  - Plot the results obtained for each $\mathbf{a}$ and overlay the distribution curves $\mathbf{f}(\mathbf{x})$ properly normalized
  - Compare the CPU time request for the 4 runs (hint: in ROOT it is possible the use the `TStopwatch` class)