Simulations with ROOT

Thanks to Prof. Massimo Masera for the next slides

Monte Carlo technique

- Monte Carlo refers to any procedure that makes use of random numbers.
- Monte Carlo methods are used in:
  - Simulation of natural phenomena
  - Simulation of experimental apparatus
  - Numerical analysis (e.g. Integral evaluation)
- What is a Random Number?
  - Is e.g. 3 a random number?
  - A sequence of random numbers is a set of numbers that have nothing to do with the other numbers in the sequence

Monte Carlo technique

- General procedure:
  - A sequence of m random numbers with uniform distribution in the [0,1] interval is extracted.
  - This sequence is used to produce a second sequence distributed according to a generic f(x), which is the pool of simulated data

Pseudo-Random numbers

- The sequence is reproducible, because the algorithm for the generation is deterministic.
- General characteristic of a random generator:
  - Statistical independence
  - “Long” repetition period
  - The sequence looks random, when indeed it is not

Random generators in ROOT

Are implemented in the **TRandom** class: fast generator with a short (10^9) period, based of the linear congruential method.

- TRandom1: inherits from TRandom and implements RANLUX
- TRandom2: inherits from TRandom has a period of 1026, but only 3 32 bits word
- TRandom3:inherits from TRandom and implements the Mersenne-Twister generator which has a period of $2^{19937}-1$ ($\approx 10^{6002}$).

Here are the CPU times obtained using the four random classes on an lxplus machine with an Intel 64 bit architecture and compiled using gcc 3.4:

	TRandom (ns/call)	**TRandom1** (ns/call)	**TRandom2** (ns/call)	**TRandom3** (ns/call)
`Rndm()`	-	-	6	9
`Gaus()`	31	161	35	42
`Rannor()`	116	216	126	130
`Poisson(m=10)`	147	1161	162	239
`Poisson(m=10)UNURAN`	80	294	89	99

BAD

SLOW

FAST

GOOD (default)

Simulation of a radioactive decay of a single nucleus

- Radioactive decay is an intrinsic random process: the probability of decay is constant (independent of the age of nuclei)
- The probability that a nucleus decays in the time Δt is p :

$$p = \alpha \Delta t \ \ (\text{per } \alpha \Delta t \ll 1)$$

- Let's consider a system initially having N_0 unstable nuclei: how does the number of nuclei vary with time?

Simulation of a radioactive decay of a single nucleus

- The algorithm which has to be implemented is the following

LOOP from t=0 to T, step Δt

Reset the number of decayed nuclei (N_{dec}) in the current time bin

LOOP over each remaining parent nucleus (from 0 to N_{tot})

Decide if the nucleus decays:

if(gRandom->Rndm()<$\alpha\Delta t$)

increment the number of decayed nuclei in this time bin

endif

END loop over nuclei

$N_{tot}=N_{tot}-N_{dec}$

Plot N_{tot} vs t

END loop over time

END

Esercitazione 10 – Esercizio 2 (Decay.C)

- Write a macro to implement the algorithm shown in the previous slide. Show the number of remaining nuclei as a function of time for the two following cases:
  - N_0=1000, α=0.01 s^{-1}, Δt=1 s
  - N_0=50000, α=0.03 s^{-1}, Δt=1 s
- Show the results in linear and logarithmic scale, for t between 0 and 300 seconds
- Compare the results with the expected result

$$dN = -N\alpha dt$$

$$N = N_0 e^{-\alpha t}$$

Possible Solution

- In the next slide a possible solution implemented using a macro in ROOT is shown
- The ROOT classes are used to generate random numbers, to store informations in the histograms and for input/output operations
- The macro (decay.C) is composed by two funtions
- It can be interpret or exectuded by CLANG (CINT)

```
#if !defined(__CINT__) || defined(__MAKECINT__)
#include <TF1.h>
#include <TFile.h>
#include <TH1D.h>
#include <TMath.h>
#include <TRandom3.h>
#include <Riostream.h>
#endif

// Declare function
Double_t exponential(Double_t *x, Double_t *par);
//_____________________________________________________
void Decay(Int_t n0 = 50000, Double_t alpha = 0.03, Double_t Delt = 1.0, Double_t ttot = 300, Int_t seed = 95689){

  gRandom->SetSeed(seed);
  Int_t Nbins = static_cast<Int_t>(ttot/Delt); // number of time intervals
  cout << "Numersof bins: "<<Nbins<<" di ampiezza "<<Delt<<" s";
  Double_t timetot = Delt*Nbins;  // totale time = ttot
  cout<<" Tempo totale "<<timetot<<endl;
  // histogram booking
  TH1D *h1 = new TH1D("h1","Remaining nuclei",Nbins+1,-Delt/2.,timetot+Delt/2.);
  h1->SetFillColor(kOrange-4);
  //Theoretical function
  TF1 *fteo = new TF1("fteo",exponential,0.,timetot,2);
  fteo->SetLineColor(kRed);
  Double_t N0 = n0;
  Double_t ALPHA = alpha;
  fteo->SetParameters(N0,ALPHA);
  fteo->SetParNames("normalizzazione","coefficiente");

  Double_t prob = alpha*Delt; //probability
  h1->Fill(0.,static_cast<double>(n0));
  for(Double_t time=Delt; time<timetot+Delt/2.; time+=Delt){
    Int_t ndec = 0;
    for(Int_t nuclei=0; nuclei<n0;nuclei++)if(gRandom->Rndm()<prob)ndec++;
    n0-=ndec;
    h1->Fill(time,static_cast<double>(n0));
  }

  TFile *file = new TFile("decay.root","recreate");
  h1->Write();
  fteo->Write();
  h1->Draw("histo");
  fteo->Draw("same");
  file->Close();
}

//_____________________________________________________
Double_t exponential(Double_t *x, Double_t *par){
  Double_t xx = x[0];
  return par[0]*exp(-par[1]*xx);
}
```

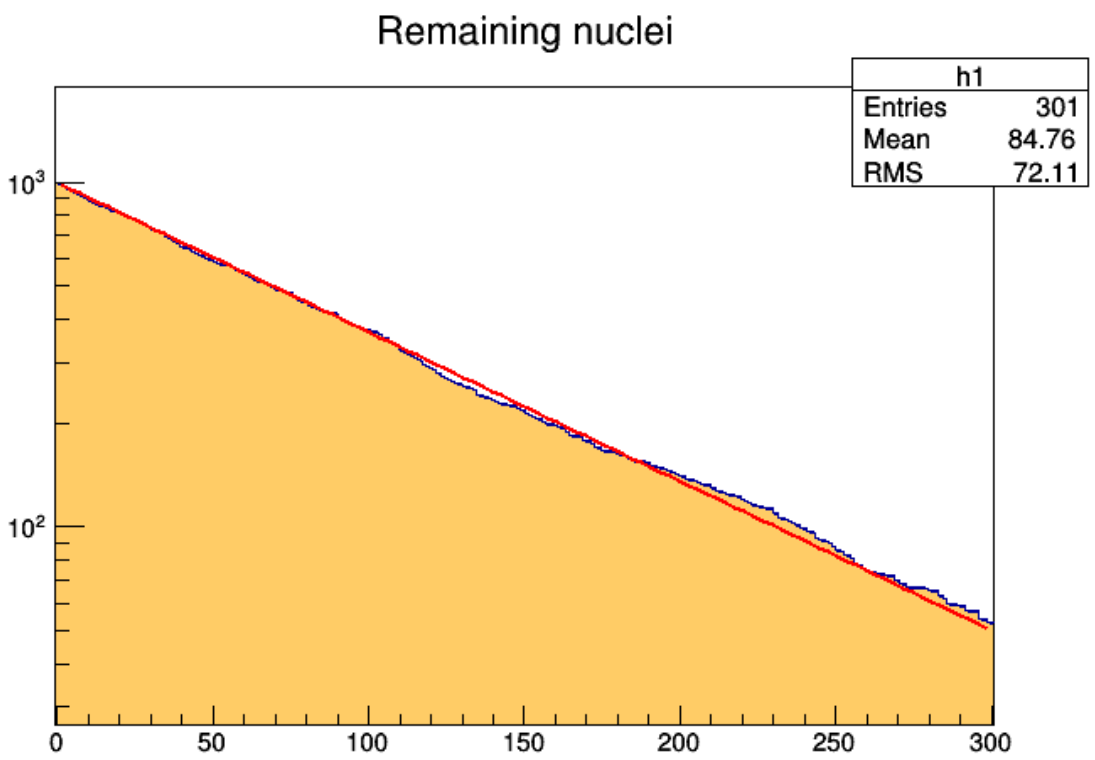
Header files, need to compile the macro

```
#if !defined(__CINT__) || defined(__MAKECINT__)
#include <TF1.h>
#include <TFile.h>
#include <TH1D.h>
#include <TMath.h>
#include <TRandom3.h>
#include <Riostream.h>
#endif

// Declare function
Double_t exponential(Double_t *x, Double_t *par);
//
void Decay(Int_t n0 = 50000, Double_t alpha = 0.03, Double_t Delt = 1.0, Double_t ttot = 300, Int_t seed = 95689){

  gRandom->SetSeed(seed);
  Int_t Nbins = static_cast<Int_t>(ttot/Delt); // number of time intervals
  cout << "Numersof bins: "<<Nbins<<" di ampiezza "<<Delt<<" s";
  Double_t timetot = Delt*Nbins;  // totale time = ttot
  cout<<" Tempo totale "<<timetot<<endl;
  // histogram booking
  TH1D *h1 = new TH1D("h1","Remaining nuclei",Nbins+1,-Delt/2.,timetot+Delt/2.);
  h1->SetFillColor(kOrange-4);
  //Theoretical function
  TF1 *fteo = new TF1("fteo",exponential,0.,timetot,2);
  fteo->SetLineColor(kRed);
  Double_t N0 = n0;
  Double_t ALPHA = alpha;
  fteo->SetParameters(N0,ALPHA);
  fteo->SetParNames("normalizzazione","coefficiente");

  Double_t prob = alpha*Delt; //probability
  h1->Fill(0.,static_cast<double>(n0));
  for(Double_t time=Delt; time<timetot+Delt/2.; time+=Delt){
    Int_t ndec = 0;
    for(Int_t nuclei=0; nuclei<n0;nuclei++)if(gRandom->Rndm()<prob)ndec++;
    n0-=ndec;
    h1->Fill(time,static_cast<double>(n0));
  }

  TFile *file = new TFile("decay.root","recreate");
  h1->Write();
  fteo->Write();
  h1->Draw("histo");
  fteo->Draw("same");
  file->Close();
}

//_____________________________________________________
Double_t exponential(Double_t *x, Double_t *par){
  Double_t xx = x[0];
  return par[0]*exp(-par[1]*xx);
}
```

Functions are declared before their implementation. Default values can be Passed as “default” argument of the function

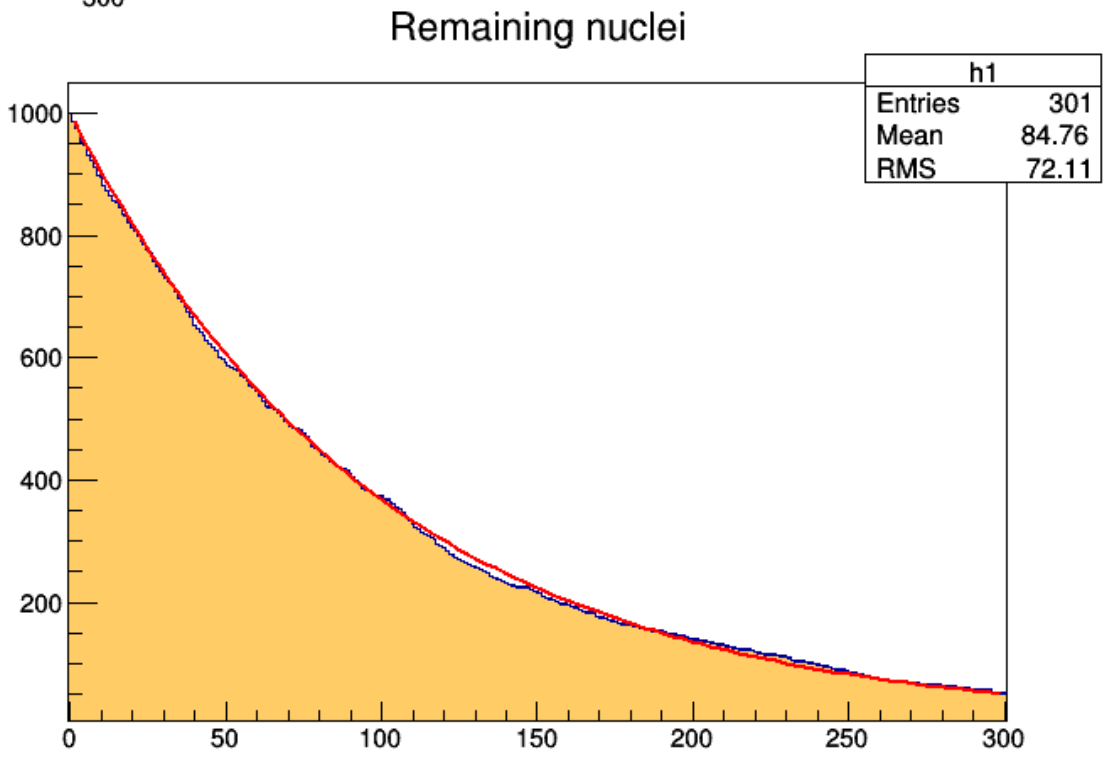
```
#if !defined(__CINT__) || defined(__MAKECINT__)
#include <TF1.h>
#include <TFile.h>
#include <TH1D.h>
#include <TMath.h>
#include <TRandom3.h>
#include <Riostream.h>
#endif

// Declare function
Double_t exponential(Double_t *x, Double_t *par);
//_____________________________________________________
void Decay(Int_t n0 = 50000, Double_t alpha = 0.03, Double_t Delt = 1.0, Double_t ttot = 300, Int_t seed = 95689){

  gRandom->SetSeed(seed);
  Int_t Nbins = static_cast<Int_t>(ttot/Delt); // number of time intervals
  cout << "Numersof bins: "<<Nbins<<" di ampiezza "<<Delt<<" s";
  Double_t timetot = Delt*Nbins;  // totale time = ttot
  cout<<" Tempo totale "<<timetot<<endl;
  // histogram booking
  TH1D *h1 = new TH1D("h1","Remaining nuclei",Nbins+1,-Delt/2.,timetot+Delt/2.);
  h1->SetFillColor(kOrange-4);
  //Theoretical function
  TF1 *fteo = new TF1("fteo",exponential,0.,timetot,2);
  fteo->SetLineColor(kRed);
  Double_t N0 = n0;
  Double_t ALPHA = alpha;
  fteo->SetParameters(N0,ALPHA);
  fteo->SetParNames("normalizzazione","coefficiente");

  Double_t prob = alpha*Delt; //probability
  h1->Fill(0.,static_cast<double>(n0));
  for(Double_t time=Delt; time<timetot+Delt/2.; time+=Delt){
    Int_t ndec = 0;
    for(Int_t nuclei=0; nuclei<n0;nuclei++)if(gRandom->Rndm()<prob)ndec++;
    n0-=ndec;
    h1->Fill(time,static_cast<double>(n0));
  }

  TFile *file = new TFile("decay.root","recreate");
  h1->Write();
  fteo->Write();
  h1->Draw("histo");
  fteo->Draw("same");
  file->Close();
}

//_____________________________________________________
Double_t exponential(Double_t *x, Double_t *par){
  Double_t xx = x[0];
  return par[0]*exp(-par[1]*xx);
}
```

Definition of the “exponential function”
That can be used in the TF1 definition



Continuous line: expected exponential.

Histogram: simulation result.



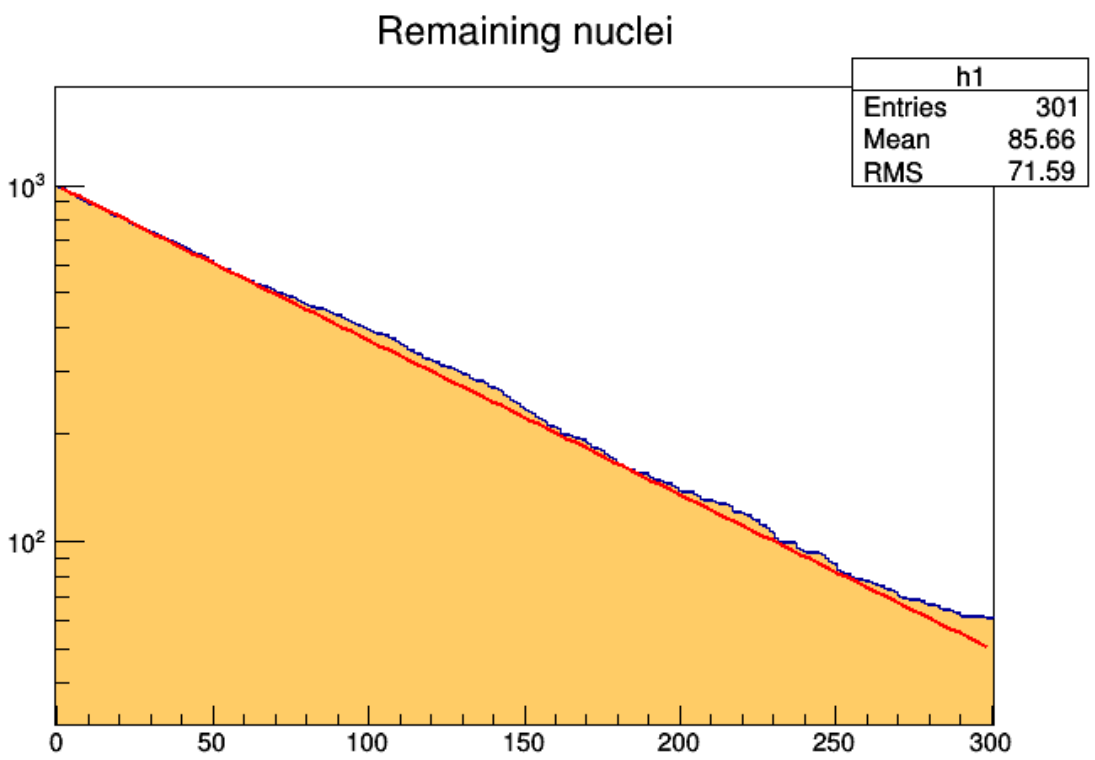
Result for:

N_0=100,

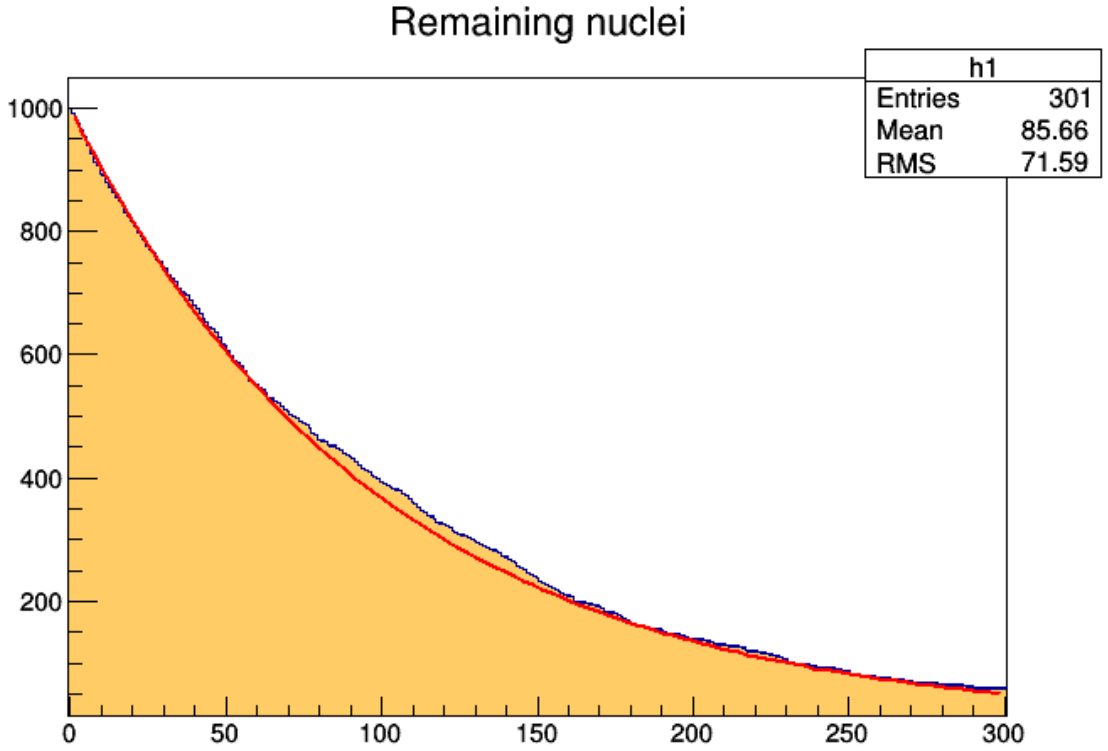
$\alpha=1\times10^{-2}\ s^{-1}$

Δt=1 s

Statistical fluctuation are very important



Same parameters as before, but different seed: the results are different!

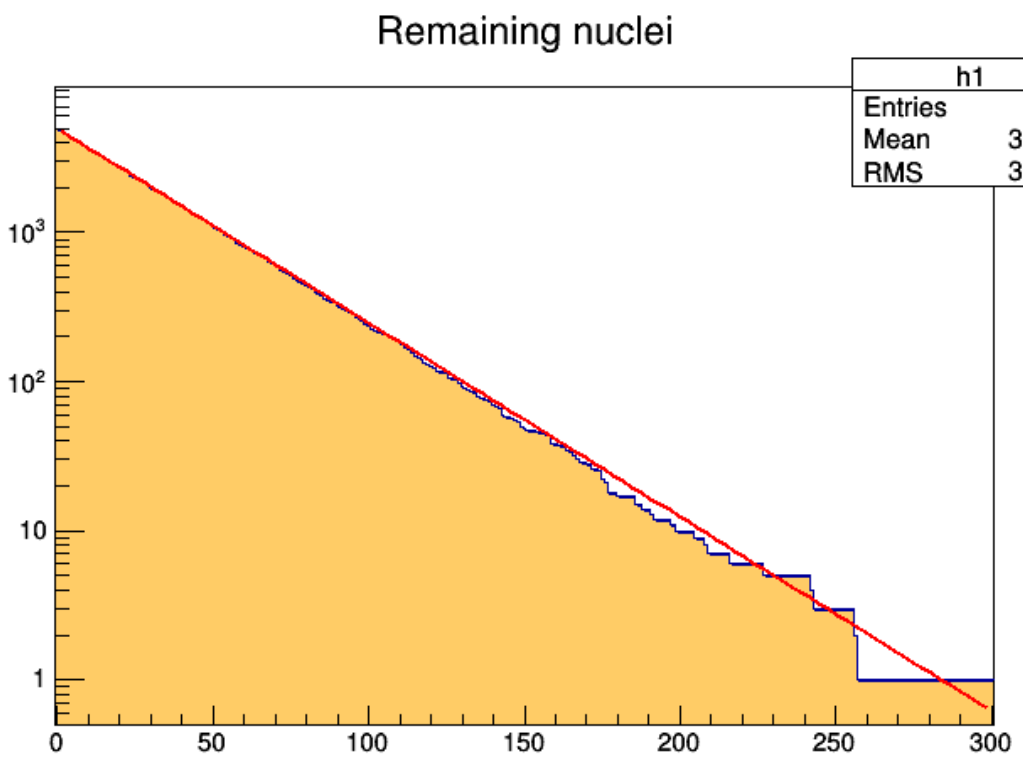


Risultato per:

N_0=100,

$\alpha=1\times10^{-2}$ s^{-1}

Δt=1 s



The importance of fluctuations depends on the number N of residual nuclei

Result for:

N_0=5000,

$\alpha=3\times10^{-2}\ s^{-1}$

Δt=1 s