EP578 Computing for Physicists

Topic 10

Monte Carlo Simulations

Department of
Engineering Physics

University of Gaziantep

Course web page
`www.gantep.edu.tr/~bingul/ep578`

Nov 2011

Introduction

The deterministic systems are described by some mathematical rule. But some systems are not deterministic known as random or stochastic.

Monte Carlo refers to any procedure that makes use of random numbers and it is opposed to deterministic algorithms.

There are lots of applications:

- **Mathematics**, Numerical Analysis
- **Physics**, Simulation of Natural Phenomena
- **Engineering**, Simulation of Experimental Apparatus
- **Biology**, Cell Simulations
- **Statistics**, Distribution Functions
- **Economy**, Modelling Stock Exchange
- . . .

- Monte Carlo methods (or Monte Carlo experiments) are a class of computational algorithms that rely on repeated random sampling to compute their results.

- Monte Carlo methods are often used in simulating physical and mathematical systems.

- To illustrate the implementation of this kind of algorithm in MATLAB we will look at just a few basic applications and relatively advanced applications.

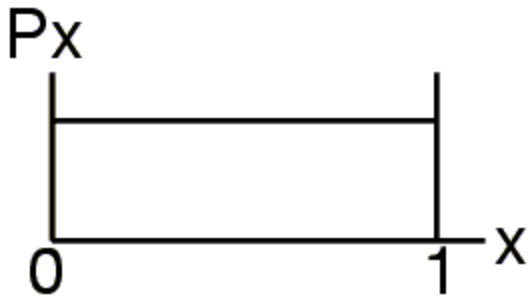
- See also:
  `http://en.wikipedia.org/wiki/Monte_Carlo_method`

Random Numbers

- Key to the Monte Carlo method is the generation of sequences of random numbers.

- A **Random Number** is a number chosen as if by chance from some specified distribution.

- In a uniform distribution of Random Numbers in the range [0,1], every number have the same chance of turning up.



- A **Random Number Generator** is a computer sub-program that produce sequence of random numbers.

- Today, all computer languages contain a mechanism for producing a sequence of random numbers which are uniformly distributed in [0,1].

  - in Fortran `CALL RANDOM_NUMBER(x)`
  - in MATLAB `x = rand;`
  - in C/C++ `x = rand()/(RAND_MAX+1.0);`
  - in ROOT `x = gRandom->Uniform();`
    (or use objects from **`TRandom3`** class)

Basic Simulations

- *Simulation of tossing a coin*.

```
// MC simulation of tossing a coin
void mccoin()
{
   int m = 0, n = 1000;
   double r, p;

   for(int i=0; i<n; i++){
     r = gRandom->Uniform();
     if(r<0.5) m++;  // head!
   }

   p = double(m)/n;

   cout << n <<" "<< m << " "<< p << endl;
}
```

- Simulation of tossing 4 coins .

```
// MC simulation of tossing 4 coins. Find the
// probability of getting at least 2 heads.
void mccoin4()
{
   int m = 0, n = 10000;
   double r, p;

   for(int i=0; i<n; i++){
     int k=0;
     if(gRandom->Uniform()<0.5) k++;
     if(gRandom->Uniform()<0.5) k++;
     if(gRandom->Uniform()<0.5) k++;
     if(gRandom->Uniform()<0.5) k++;
     if(k>=2) m++;
   }

   p = double(m)/n;

   cout << n <<" "<< m << " "<< p << endl;
}
```

- *Simulation of tossing a pair of dice*.

```
// MC simulation of tossing a pair of dice coins.
// Find the probability of getting at least 6:6.
void mcdice()
{
   int m = 0, n = 10000;
   double r, p;

   for(int i=0; i<n; i++){
     int d1 = 1 + int(6*gRandom->Uniform());
     int d2 = 1 + int(6*gRandom->Uniform());
     if(d1==6 && d2==6) m++;  // 6:6
   }

   p = double(m)/n;

   cout << n <<" "<< m << " "<< p << endl;
}
```

Radioactive Decay Simulation

This is a truly random processes. The probability of decay is constant. The probability that a nucleus undergoes radioactive decay in time *dt* is *p*:

$$p = \lambda\, dt \quad (\text{for } \lambda\, dt << 1)$$

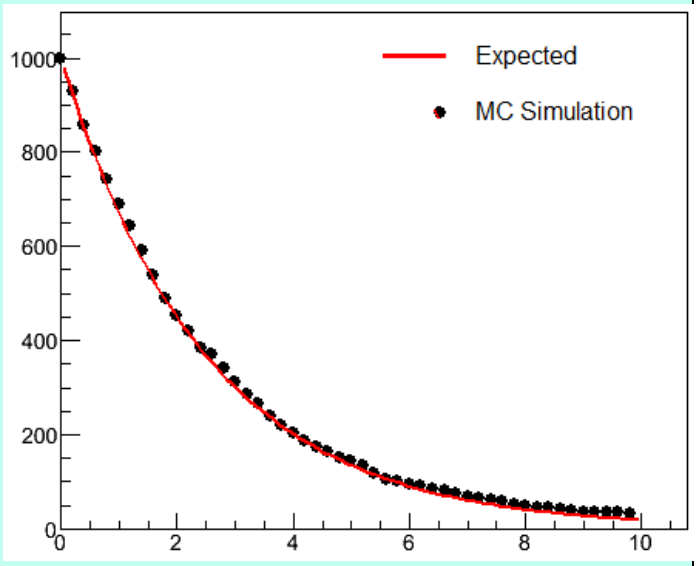
where λ (decay constant) is probability per unit time for the decay of each nucleus of a given nuclide.

Consider a system initially having N_0 unstable nuclei at time $t = 0$. How does the number of parent nuclei, *N*, change with time?

Theoretically, the number of undecayed nuclei at time *t* is given by:

$$N = N_0 \exp(-\lambda t)$$

```
// MC simulation of radioactive decay
void mcrd()
{
  gROOT->Reset();
  double const dt=0.2, n0=1000;
  double const tmax=10, L=0.4;
  int    const N = tmax/dt;
  double n=n0, ns[N]={n0}, t[N]={0.0};

  for(int j=1; j<N; j++){
     t[j] = j*dt;
     for(int i=1; i<=n; i++){
        double r = gRandom->Uniform();
        if(r<L*dt) n--;
     }
     ns[j] = n;
  }

  TGraph *grs = new TGraph(N, t, ns);
  TF1    *grt = new TF1("theory","1000*exp(-0.4*x)",0, tmax);
  grs->SetMarkerStyle(20);
  grs->Draw("AP");
  grt->Draw("Same");
}
```



Special Distribution Functions

Binomial Distribution Function

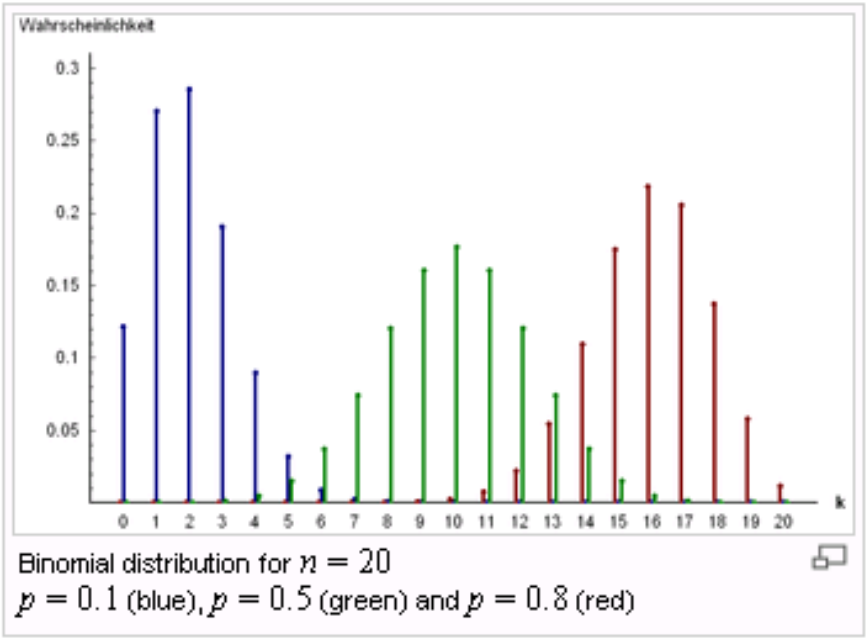
The binomial distribution function specifies the number of times (k) that an event occurs in *n* independent trials where *p* is the probability of the event occurring in a single trial.

$$p_{binom}(n,k,p) = \binom{n}{k} p^k (1-p)^k$$

$$\text{mean}: \quad < k >= np$$

$$\text{std.dev}: \quad \sigma = \sqrt{np(1-p)}$$

where $\binom{n}{k} = \frac{n!}{k!(n-k)!}$



Binomial distribution for $n = 20$
$p = 0.1$ (blue), $p = 0.5$ (green) and $p = 0.8$ (red)

Example: A coin is tossed 6 times.

The probability of getting exactly four heads:

$$\binom{6}{2}\left(\frac{1}{2}\right)^2\left(\frac{1}{2}\right)^4 = \frac{15}{64} = 0.234375$$

The probability of getting at least four heads:

$$\binom{6}{4}\left(\frac{1}{2}\right)^4\left(\frac{1}{2}\right)^2 + \binom{6}{5}\left(\frac{1}{2}\right)^5\left(\frac{1}{2}\right) + \binom{6}{5}\left(\frac{1}{2}\right)^6 = \frac{11}{32} = 0.34375$$

where $\binom{n}{r} = \frac{n!}{r!(n-r)!}$

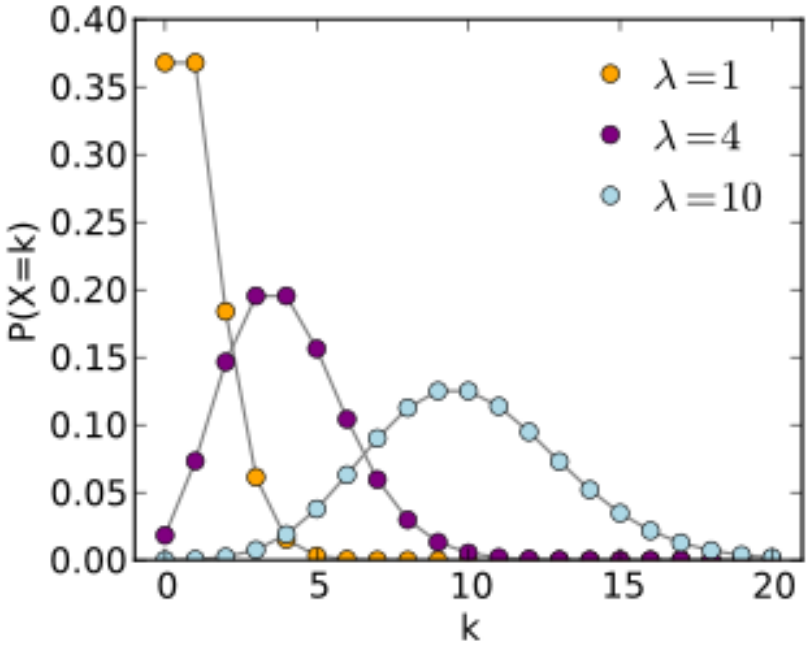
Poisson Distribution Function

In the binomial equation, if the probability *p* is so small then the distribution of events can be approximated by the Poisson distribution.

$$p_{poisson}(k,\lambda) = \frac{e^{-\lambda}\lambda^{k}}{k!}$$

$$\text{mean}: \quad \lambda = np$$

$$\text{std.dev}: \quad \sigma = \sqrt{\lambda}$$



lim Binomial Distribution = Poisson Distribution

p->0

Example: Birthday problem:

Probability of one person to have birthday in any day is 1/365 = 0.00274. Calculate the probability that 4 people share a birthday in a group of 150 people.

Mean: $\lambda = 150*0.00274 = 0.411$

Probability: $p = \exp(-0.411)*0.411^4/4! = 0.002$

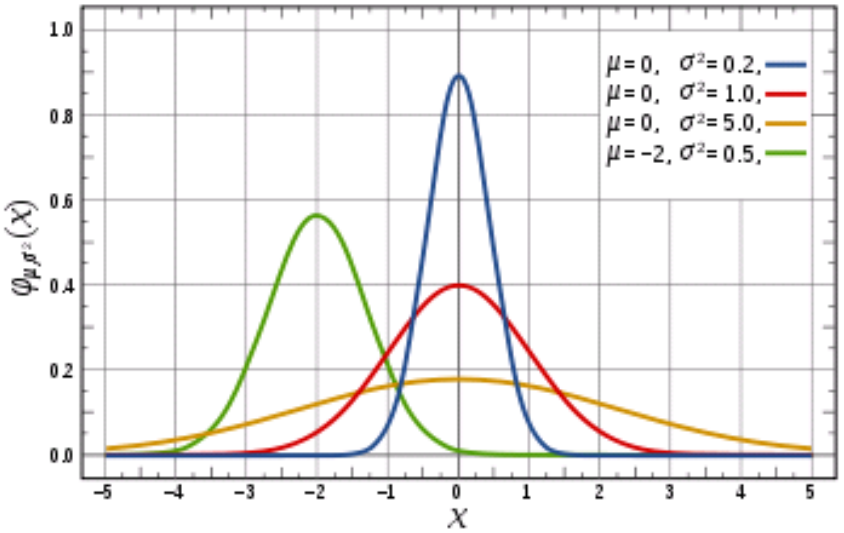
Gaussian (Normal) Distribution Function

If the number of events (n) is very large, then the Gaussian distribution function may be used to describe physical events. The Gaussian distribution is a continuous function which approximates the exact binomial distribution of events.

$$p_{gauss}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

mean : μ

std.dev : σ



lim Binomial Distribution = Gaussian Distribution

n->inf

Example: Mean weight of 500 male students at a certain university is 72 kg and the standard deviation is 5 kg. Assuming that the weights are normally distributed, find how many students weigh:

(a) between 66 and 75 kg

(b) more than 80 kg.

Normal distribution function:

$$f(x) = \frac{1}{5\sqrt{2\pi}} e^{-\frac{1}{2}((x-72)/5)^2} = 0.079788 e^{-0.02(x-72)^2}$$

(a) $p = \int_{66}^{75} f(x)dx = 0.6107$

# of students weighing between 66-75 kg is 500*0.6107 = **305**.

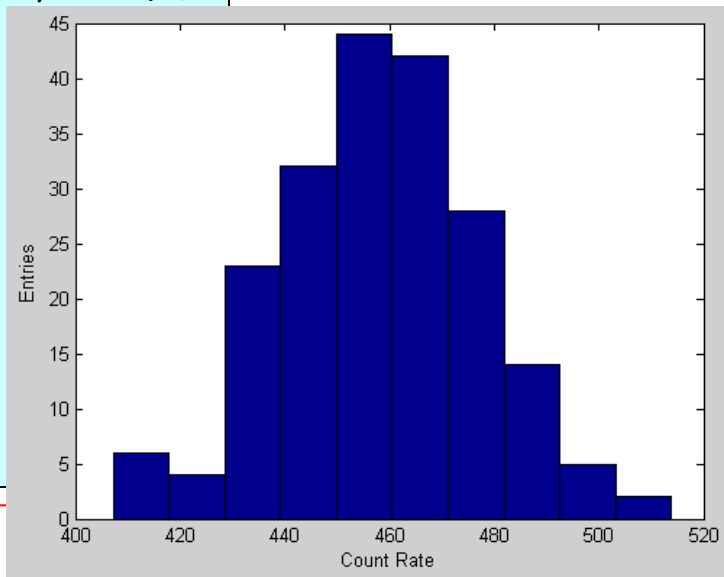
(b) $p = \int_{80}^{\infty} f(x)dx = 0.0548$

# of students weighing more than 80 kg is 500*0.0548 = **27**.

Example :

In an experiment a Geiger counter is used to count the radioactive emissions of cobalt 60 over a 10-second period. After a large number of such readings are taken, the count rate is estimated to be normally distributed with a mean of 460 and a standard deviation of 20. Simulate such an experiment 200 times by generating 200 random numbers with this mean and standard deviation. Plot the histogram (use 10 bins).

```
void mcgm()
{
  gROOT->Reset();
  TH1F *h = new TH1F("h","Entries",10,400,550);

  for(int i=1;i<=200;i++) {
     double r = gRandom->Gaus(460,20);
     h->Fill(r);
  }

  h->Draw();
}
```



Example :

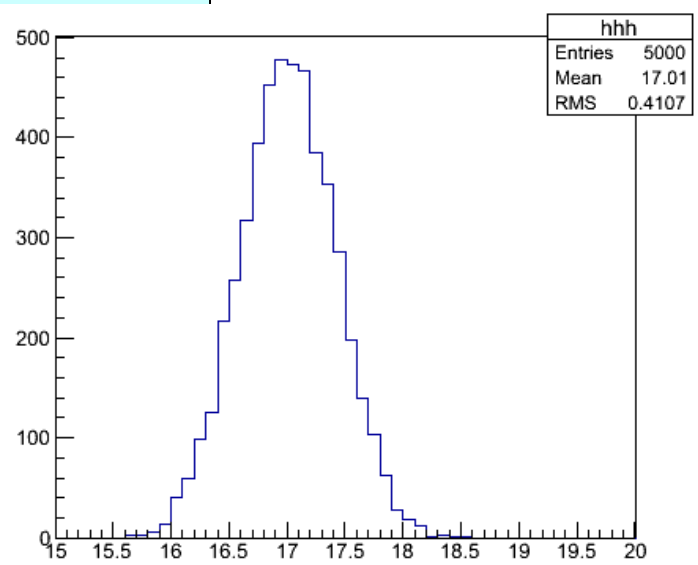
Energy resolution of a photon detector is given by:

$$\sigma_E = 0.1\sqrt{E}$$

where E is measured in keV. 5000 X-rays of wavelength 73 pm (17 keV) are generated and send on the detector.

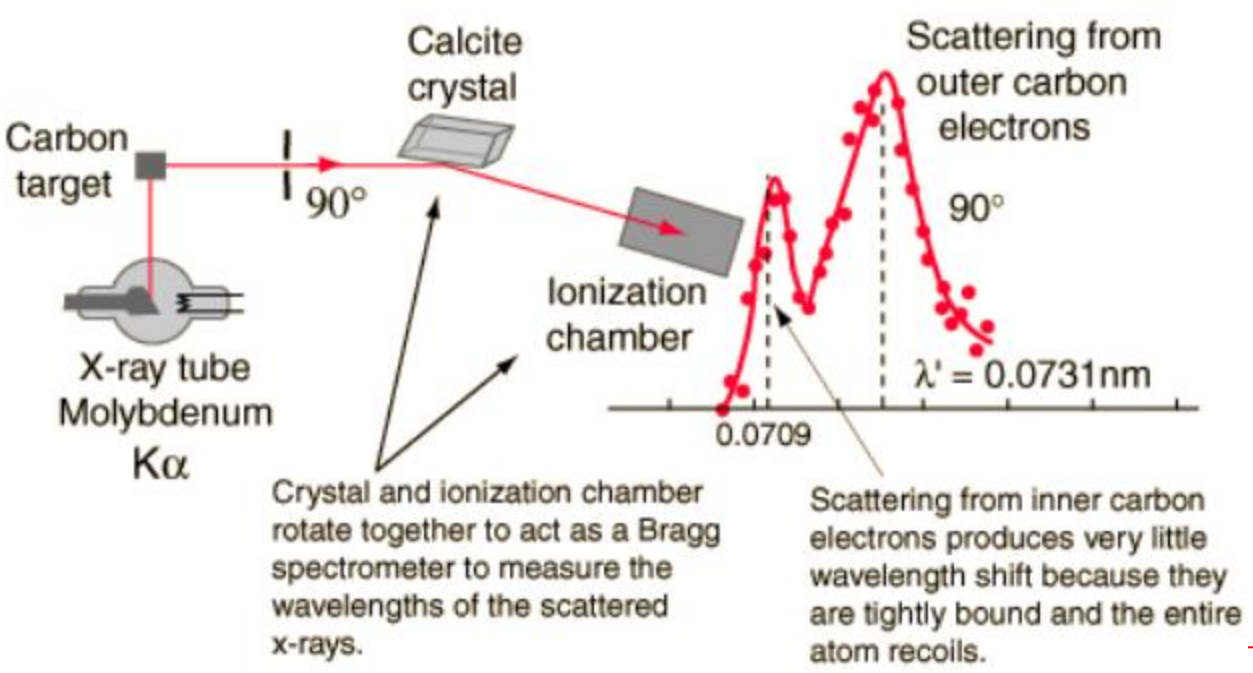
What is the measured energy spectrum?

```
void mcdet()
{
  gROOT->Reset();
  TH1F *h = new TH1F("hhh","",50,15,20);

  for(int i=1;i<=5000;i++) {
     double s = 0.1*sqrt(17);
     double r = gRandom->Gaus(17,s);
     h->Fill(r);
  }

  h->Draw();
}
```



Compton Scattering

Compton's original experiment made use of molybdenum K-alpha x-rays, which have a wavelength of 0.0709 nm. These were scattered from a block of carbon and observed at different angles with a Bragg spectrometer. The spectrometer consists of a rotating framework with a calcite crystal to diffract the x-rays and an ionization chamber for detection of the x-rays. Since the spacing of the crystal planes in calcite is known, the angle of diffraction gives an accurate measure of the wavelength.



Using the detector in the previous example simulate the experiment!

Homework

Solve the following problems. You have to prepare a pdf document and sent it to me until next lecture.

E-mail: bingul[at]gantep.edu.tr (*replace* [at] *with* @)

1. Bacteria are grown in culture dishes in a laboratory. Experience tells us that on average in this lab 20% of the dishes become contaminated by unwanted bacteria (thus spoiling the culture). Assume that the lab is growing bacteria in ten dishes. Write a MC simulation program to evaluate the probability that more than half of the dishes will become contaminated.

2:

A machine produces bolts which are 5% defective. Write a MC simulation program to find the probability, that a in a random sample of 400 bolts produced by this machine, less than 30 are defective.

3.

Table shows a data obtained from a radioactive substance. Here *t* is the time in seconds and *N* is the number of surviving nuclei. Using ROOT fitting tool, determine the half life and identify the nucleus. Assume that each measurement of *N* has an associated counting error of sqrt(*R*). **e.g**, for *R* = 300 Bq then measurement error is sqrt(300) = 17.3 Bq.

t(s)	N
40	300
100	245
140	210
200	165
240	127
300	110
340	90
400	85
440	58
500	45

4.

Simulate the experiment in problem 3. Compare your result with the experiment. Plot the experimental and your simulated data on the same axis.