Υπολογιστικές Μέθοδοι

http://eclass.uoa.gr/courses/PHYS186/

Διδάσκοντες:
Φ. Διάκονος
Δ. Φασουλιώτης

- Μέθοδοι Monte-Carlo και εφαρμογές

Μέθοδοι Monte Carlo

Τι είναι:

Οποιαδήποτε αριθμητική μέθοδος
χρησιμοποιεί ψευδοτυχαίους αριθμούς

Που χρησιμοποιείται:

Παντού

Ολοκλήρωση

Επίλυση διαφορικών εξισώσεων

Προσομοίωση στοχαστικών συστημάτων

Γιατί;

Εύκολη στην εφαρμογή

Εφαρμόσιμη και με τις πιο περίπλοκες συνοριακές συνθήκες

Αποδοτική σε πολυδιάστατα προβλήματα

Μέθοδοι MC

Πως υλοποιούνται:

- Επιλογή της συνάρτησης πυκνότητας πιθανότητας (pdf) του υποθετικού πληθυσμού.

- Γέννηση τυχαίου δείγματος από μια ακολουθία ψευδοτυχαίων αριθμών σύμφωνα με την παραπάνω πιθανότητα.

- Στατιστική επεξεργασία για την εκτίμηση των υπό εξέταση παραμέτρων από το τυχαίο δείγμα.

Μέθοδοι MC: Τυχαίοι αριθμοί

- **Τυχαίοι αριθμοί**
  Φυσικές στοχαστικές διαδικασίες
  (ηλεκτρονικός θόρυβος, πυρηνικές διασπάσεις, κ.λ.π.)

- **Ψευδοτυχαίοι αριθμοί**
  Παράγονται αιτιοκρατικά από υπολογιστικούς αλγορίθμους
  Εύκολοι στην παραγωγή
  Επαναλήψιμη διαδικασία
  Παρέχουν ομοιόμορφη κάλυψη φασικού χώρου

- **Σχεδόν Ψευδοτυχαίοι αριθμοί**
  Παράγονται αιτιοκρατικά από υπολογιστικούς αλγορίθμους
  Αλυσίδες Markov
  Παρέχουν προμελετημένη κάλυψη φασικού χώρου

Μέθοδοι MC: Τυχαίοι αριθμοί

Αλγόριθμοι (Γεννήτορες) Ψευδοτυχαίων Αριθμών

Ομοιόμορφη κατανομή [0.,1.]

Χαρακτηριστικά:

- **Περίοδος**
- **Συσχέτιση**
- **Απόδοση**

Υλοποίηση:

$I_{j+1} = \mathrm{mod}\,(aI_j + c, m)$

... και ποικίλες παραλλαγές

Ποιοτικός έλεγχος:

- **Θεωρητικός έλεγχος**
- **Πρακτικός έλεγχος**

Περισσότερες πληροφορίες

http://random.mat.sbg.ac.at

Numerical Recipes

κ.λ.π

Παράδειγμα γεννήτορα ψευδο-τυχαίων αριθμών

$$x_{n+1} = (a\, x_n + c) \bmod m$$

με a =1103515245, c = 12345 and m =2^{31}.

```
int x=1717;  //seed – αρχική τιμή της ακολουθίας ψευδο-τυχαίων αριθμών
             // μπορεί να επιλέγεται με διάφορους τρόπους
int m=pow(2,31);
x=(1103515245*x+12345)%m;
double random=double(x)/m;
```

Numerical Recipes

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

```
float ran0(long *idum)
```

"Minimal" random number generator of Park and Miller. Returns a uniform random deviate between 0.0 and 1.0. Set or reset `idum` to any integer value (except the unlikely value `MASK`) to initialize the sequence; `idum` must not be altered between calls for successive deviates in a sequence.

```
{
    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;
}
```

XORing with `MASK` allows use of zero and other simple bit patterns for `idum`.

Compute `idum=(IA*idum) % IM` without overflows by Schrage's method.

Convert `idum` to a floating result.

Unmask before return.

Numerical Recipes

```
#define IA 16807
#define IM 2147483647
#define AM (1.0/IM)
#define IQ 127773
#define IR 2836
#define NTAB 32
#define NDIV (1+(IM-1)/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0-EPS)

float ran1(long *idum)
```

"Minimal" random number generator of Park and Miller with Bays-Durham shuffle and added safeguards. Returns a uniform random deviate between 0.0 and 1.0 (exclusive of the endpoint values). Call with `idum` a negative integer to initialize; thereafter, do not alter `idum` between successive deviates in a sequence. `RNMX` should approximate the largest floating value that is less than 1.

```
{
    int j;
    long k;
    static long iy=0;
    static long iv[NTAB];
    float temp;

    if (*idum <= 0 || !iy) {                Initialize.
        if (-(*idum) < 1) *idum=1;          Be sure to prevent idum = 0.
        else *idum = -(*idum);
        for (j=NTAB+7;j>=0;j--) {           Load the shuffle table (after 8 warm-ups).
            k=(*idum)/IQ;
            *idum=IA*(*idum-k*IQ)-IR*k;
            if (*idum < 0) *idum += IM;
            if (j < NTAB) iv[j] = *idum;
        }
        iy=iv[0];
    }
    k=(*idum)/IQ;                           Start here when not initializing.
    *idum=IA*(*idum-k*IQ)-IR*k;             Compute idum=(IA*idum) % IM without over-
    if (*idum < 0) *idum += IM;                 flows by Schrage's method.
    j=iy/NDIV;                              Will be in the range 0..NTAB-1.
    iy=iv[j];                               Output previously stored value and refill the
    iv[j] = *idum;                              shuffle table.
    if ((temp=AM*iy) > RNMX) return RNMX;   Because users don't expect endpoint values.
    else return temp;
}
```

Τυχαίοι αριθμοί: Έλεγχος

Γεννήτορες Ψευδοτυχαίων Αριθμών: **Πρακτικός έλεγχος**

Υπολογισμός των ροπών k τάξης της κατανομής τυχαίων αριθμών:

$$\left\langle x^k \right\rangle = \frac{1}{n}\sum_{i=1}^{n} x_i^k$$

Αν οι τυχαίοι αριθμοί κατανέμονται με ομοιόμορφη πιθανότητα p(x)

$$\frac{1}{n}\sum_{i=1}^{n} x_i^k \approx \int_0^1 dx x^k\, p(x) + \propto (1/\sqrt{n}) \approx \frac{1}{k+1} + \propto (1/\sqrt{n})$$

Τυχαίοι αριθμοί: Έλεγχος

Γεννήτορες Ψευδοτυχαίων Αριθμών: **Πρακτικός έλεγχος**

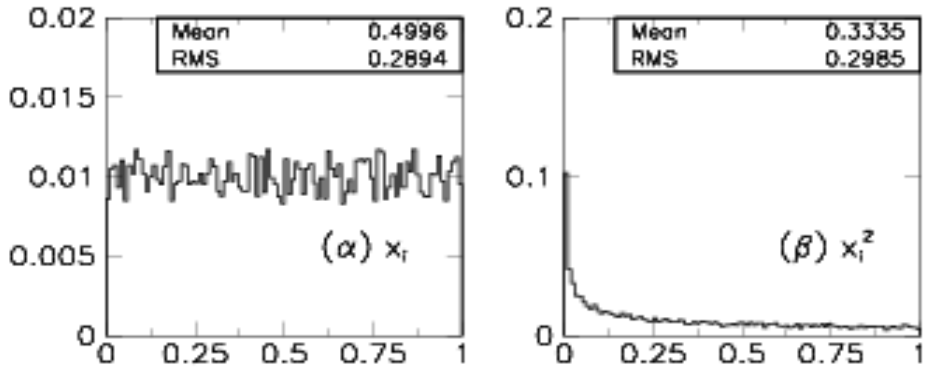
Υπολογισμός των Συσχετίσεων: $$C(k) = \frac{1}{n}\sum_{i=1}^{n} x_i x_{i+k}$$

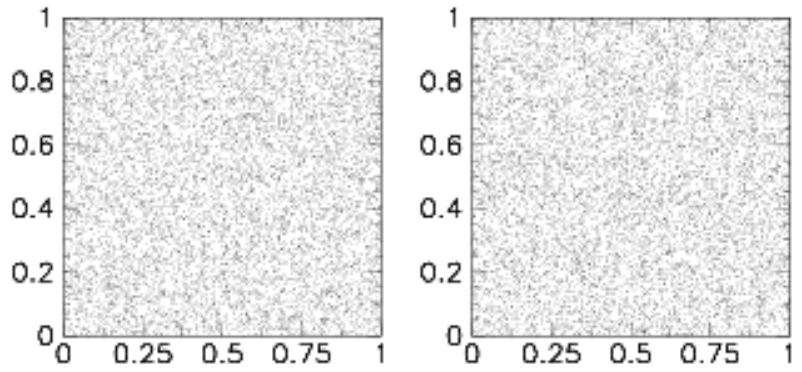
Αν οι τυχαίοι αριθμοί είναι ανεξάρτητοι και κατανέμονται ομοιόμορφα

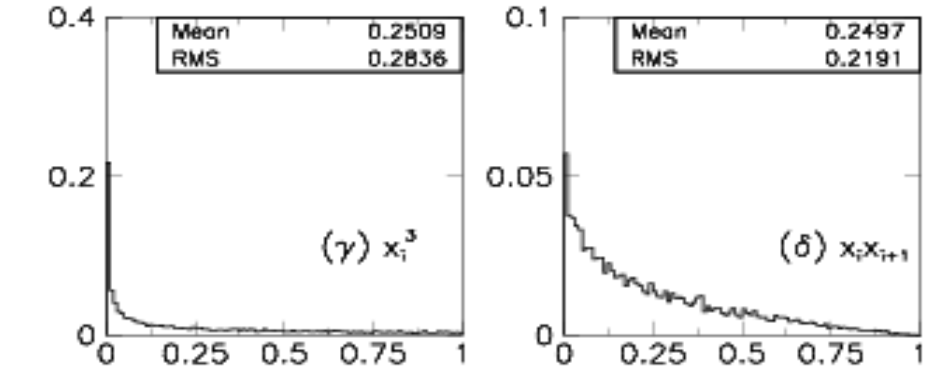
$$\frac{1}{n}\sum_{i=1}^{n} x_i x_{i+k} \approx \int_0^1 dx \int_0^1 dy xy p(x,y) = \frac{1}{4}$$

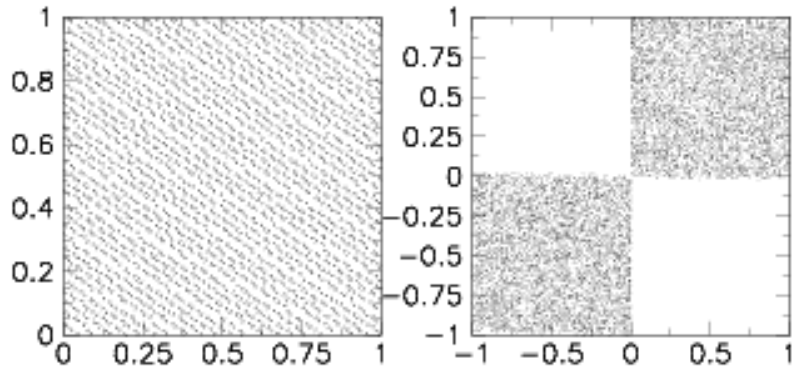
Τυχαίοι αριθμοί: Έλεγχος

Γεννήτορες Ψευδοτυχαίων Αριθμών: Πρακτικός έλεγχος





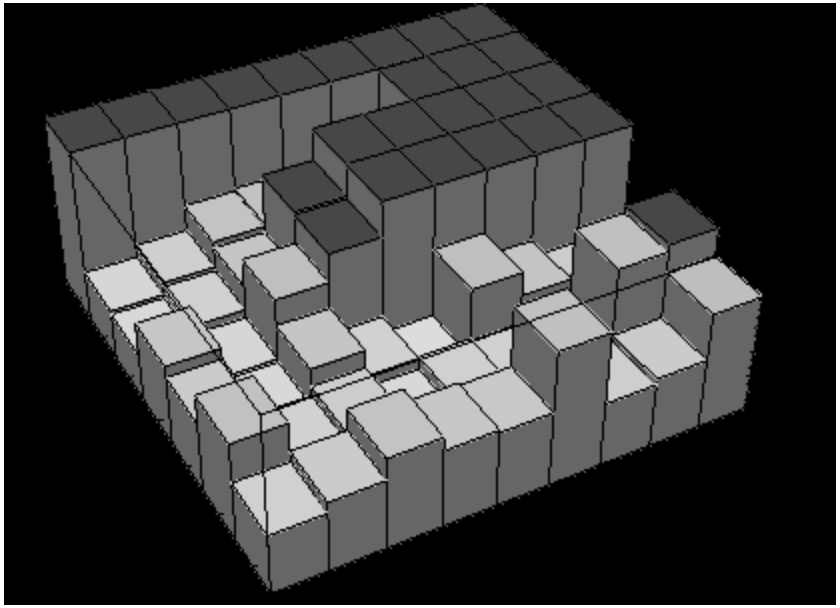




Τυχαίοι αριθμοί: Έλεγχος

Γεννήτορες Ψευδοτυχαίων Αριθμών: **Πρακτικός έλεγχος**

Πιο σύνθετοι έλεγχοι:
π.χ. Κατανομή κ αριθμών σε επίπεδα κ-1 διαστάσεων



Τελικός έλεγχος:
Επανάληψη της μελέτης με άλλο γεννήτορα τυχαίων αριθμών

Γεννήτορες Ψευδοτυχαίων Αριθμών μη ομοιόμορφης κατανομής
Μέθοδος μετασχηματισμού

Έστω x τυχαία μεταβλητή ομοιόμορφα κατανεμημένη στο διάστημα [0 , 1]

$$p(x) = \begin{cases} 1, & 0 \le x \le 1 \\ 0, & \text{αλλού} \end{cases}$$

Αν μετασχηματίσουμε τη μεταβλητή x → y, η πιθανότητα πρέπει να διατηρείται

Οπότε $p(y)dy = p(x)dx$

και δεδομένου ότι ***p(x)=1***, τότε

$$p(y)dy = dx \Rightarrow x(y) = \int_0^y p(y')dy'$$

Παράδειγμα 1

Έστω ότι θέλουμε y να κατανέμεται ομοιόμορφα στο διάστημα [a, b]

$$p(y) = \frac{1}{b-a}, \quad a \le y \le b$$

τότε

$$p(y)dy = \frac{dy}{b-a} = dx \Rightarrow$$

$$x(y) = \int_a^y \frac{dy'}{b-a} \Rightarrow x = \frac{1}{b-a}(y-a) \Rightarrow$$

$$\boxed{y = a + (b-a)x}$$

Παράδειγμα 2

Έστω ότι θέλουμε y να κατανέμεται γραμμικά

$$p(y) \propto y$$
$$p(y) = 2y, \quad 0 \le y \le 1$$

τότε

$$p(y)dy = 2ydy = dx \Rightarrow$$
$$x(y) = \int_0^y 2y'dy' \Rightarrow x = y^2 \Rightarrow$$
$$\boxed{y = \sqrt{x}}$$

Παράδειγμα 3

Έστω ότι θέλουμε y να κατανέμεται εκθετικά

$$p(y)dy = e^{-y}$$

τότε

$$p(y)dy = e^{-y}dy = dx \Rightarrow$$

$$x(y) = \int_0^y e^{-y'}dy' \Rightarrow x = 1 - e^{-y} \Rightarrow$$

$$\boxed{y = -\ln(1-x)}$$

Μέθοδος μετασχηματισμού – Κανονική κατανομή

Οι μέθοδοι μετασχηματισμού γενικεύονται και σε περισσότερες διαστάσεις

$$p(y_1, y_2, \ldots) dy_1 dy_2 \ldots = p(x_1, x_2, \ldots) \left| \frac{\partial(x_1, x_2, \ldots)}{\partial(y_1, y_2, \ldots)} \right| dy_1 dy_2 \ldots$$

Για παράδειγμα αν

$$y_1 = \sqrt{-2 \ln x_1} \cos 2\pi x_2 \qquad y_2 = \sqrt{-2 \ln x_1} \sin 2\pi x_2$$

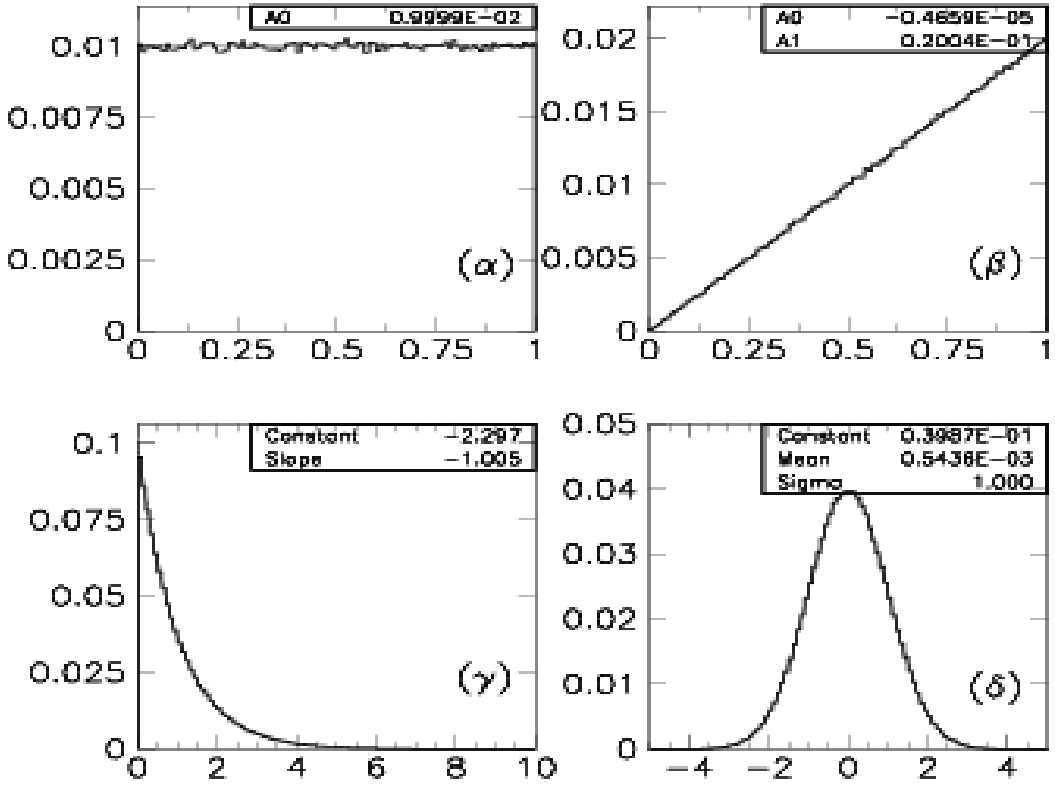
τότε

$$\left| \frac{\partial(x_1, x_2)}{\partial(y_1, y_2)} \right| = -\left(\frac{1}{\sqrt{2\pi}} e^{-y_1^2/2} \right) \left(\frac{1}{\sqrt{2\pi}} e^{-y_2^2/2} \right)$$

Τυχαίοι αριθμοί: Μεθ. Μετασχηματισμού

Γεννήτορες Ψευδοτυχαίων Αριθμών μη ομοιόμορφης κατανομής

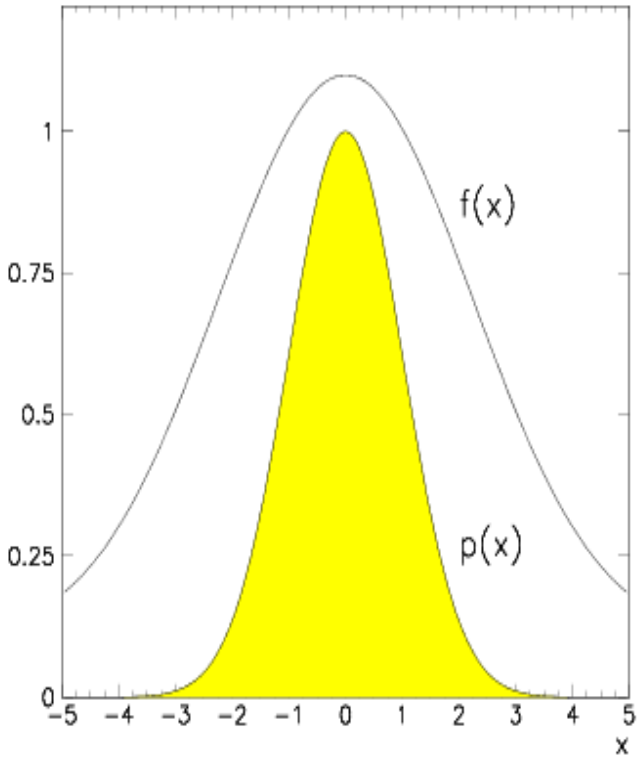
Μέθοδος μετασχηματισμού



Τυχαίοι αριθμοί: Μεθ. Απόρριψης

Γεννήτορες Ψευδοτυχαίων Αριθμών μη ομοιόμορφης κατανομής

Μέθοδος απόρριψης



- Γεννάμε σημεία σε δύο διαστάσεις τα οποία είναι ομοιόμορφα κάτω από τη συνάρτηση σύγκρισης.
- Όποτε το σημείο βρίσκεται κάτω από τη συνάρτηση πιθανότητας το κρατάμε, αν όχι το απορρίπτουμε και γεννάμε καινούριο σημείο.
- Με αυτή τη διαδικασία έχουμε καταφέρει να γεννάμε ομοιόμορφα σημεία κάτω από την p(x).

Τυχαίοι αριθμοί: Μεθ. Απόρριψης

Γεννήτορες Ψευδοτυχαίων Αριθμών μη ομοιόμορφης κατανομής

Μέθοδος απόρριψης

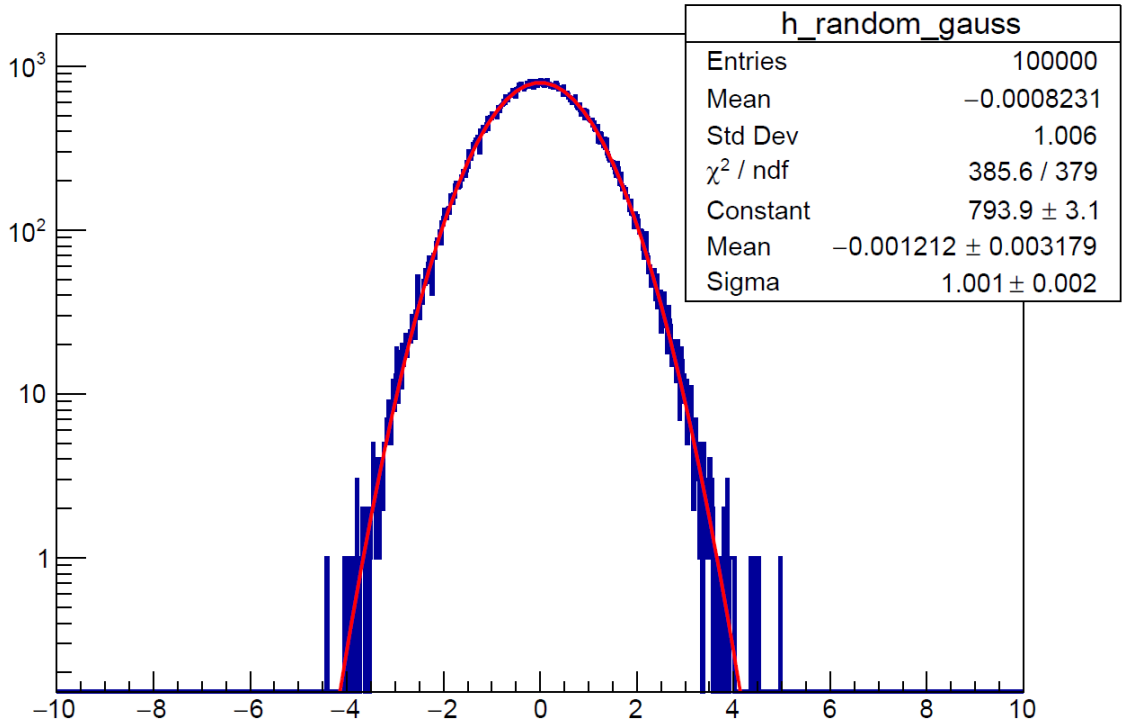
```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <TFile.h>
#include <TH1F.h>
#include <TRandom.h>

void random_gauss(int N, int seed)
{
  TH1F *h=new TH1F("h_random_gauss"," ",1000,-10.,10.);
  TRandom *ran=new TRandom3(seed);
  int i=0;
  while(i<N)
    {
     double x=10.*ran->Rndm()-5.;
     double y=ran->Rndm();
     double rg=exp(-0.5*x*x);
     if(rg>y){ i++; h->Fill(x);}
    }
  h->SetLineWidth(2.);
  //h->SetLineColor(kRed);
  h->Draw("");
  h->Fit("gaus");
}
```

Τυχαίοι αριθμοί: Μεθ. Απόρριψης

Γεννήτορες Ψευδοτυχαίων Αριθμών μη ομοιόμορφης κατανομής

Μέθοδος απόριψης

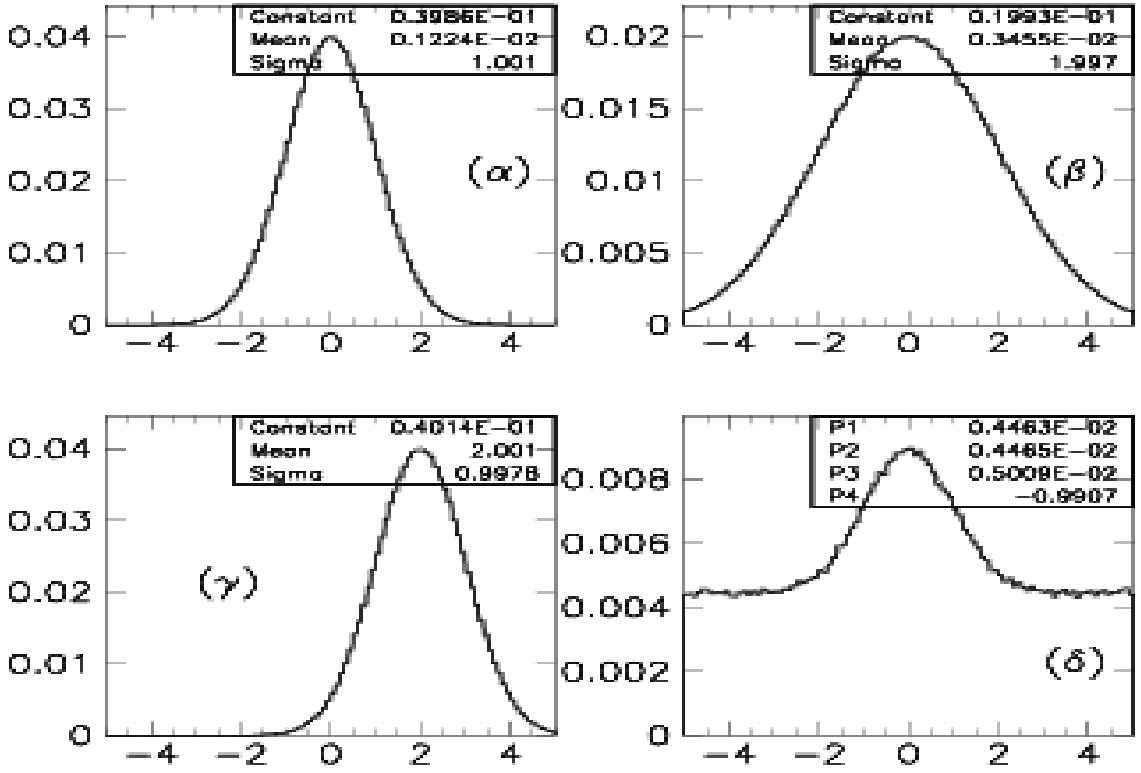
```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <TFile.h>
#include <TH1F.h>
#include <TRandom.h>

void random_gauss(int N, int see
{
  TH1F *h=new TH1F("h_random_gau
  TRandom *ran=new TRandom3(seed
  int i=0;
  while(i<N)
    {
     double x=10.*ran->Rndm()-5.
     double y=ran->Rndm();
     double rg=exp(-0.5*x*x);
     if(rg>y){ i++; h->Fill(x);}
    }
  h->SetLineWidth(2.);
  //h->SetLineColor(kRed);
  h->Draw("");
  h->Fit("gaus");
}
```



Τυχαίοι αριθμοί: Μεθ. Απόρριψης

Γεννήτορες Ψευδοτυχαίων Αριθμών μη ομοιόμορφης κατανομής

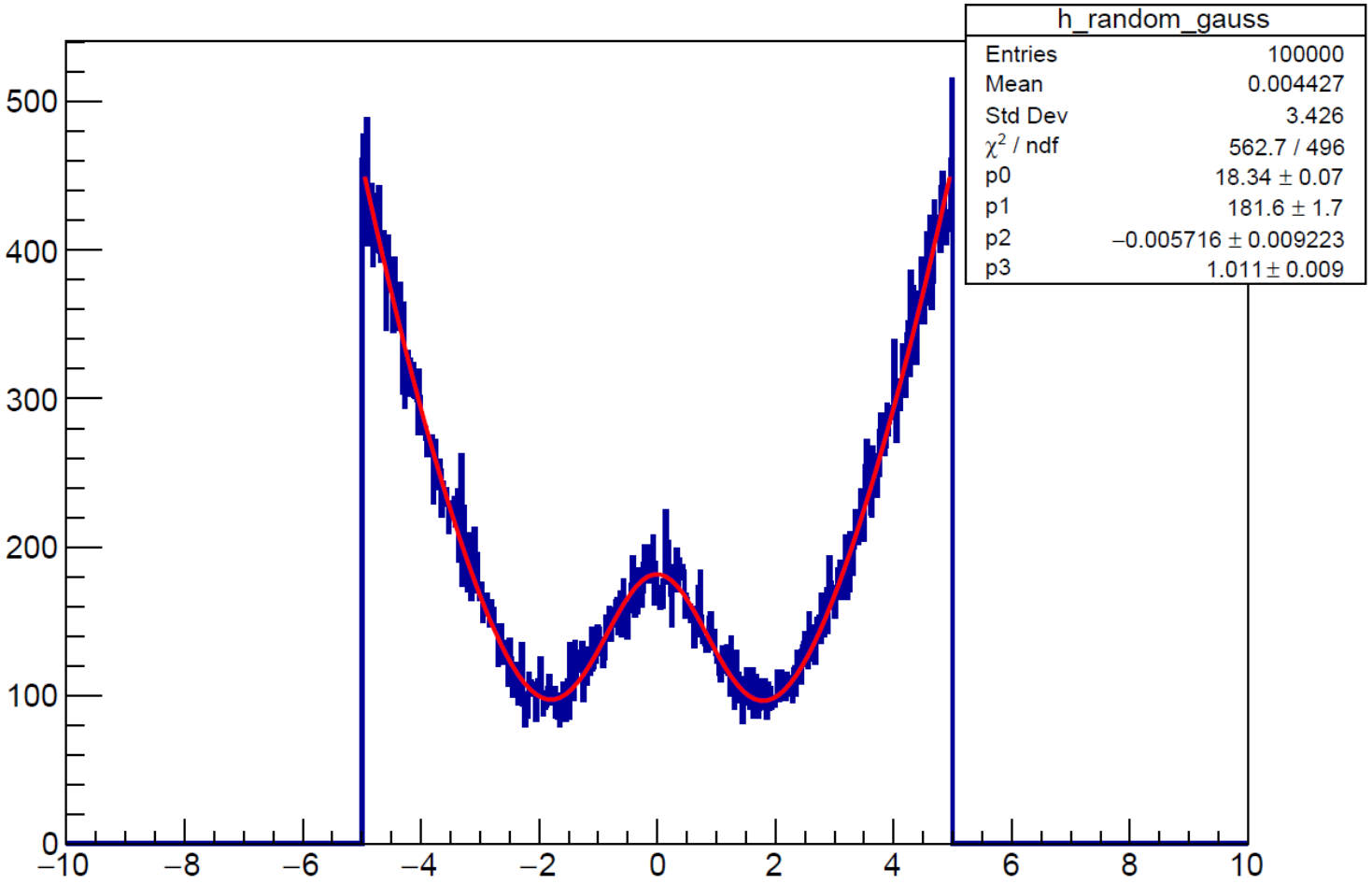
Μέθοδος απόρριψης



Τυχαίοι αριθμοί: Μεθ. Απόρριψης

Γεννήτορες Ψευδοτυχαίων Αριθμών μη ομοιόμορφης κατανομής

Μέθοδος απόρριψης

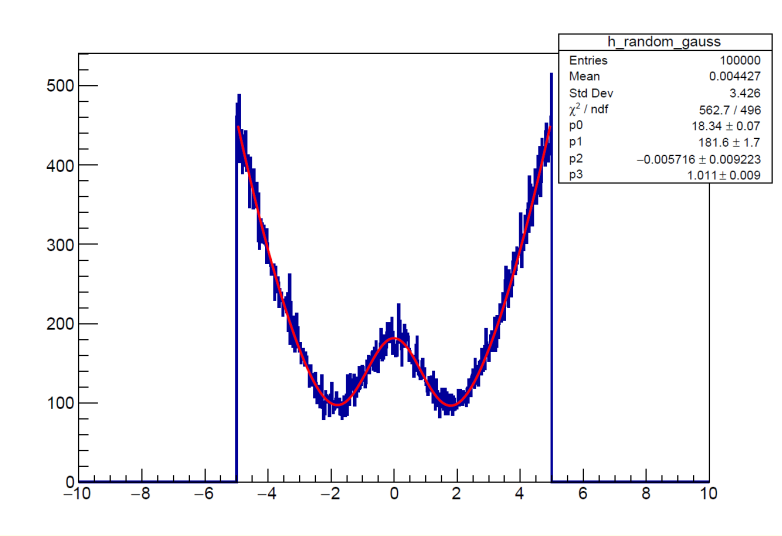


Τυχαίοι αριθμοί: Μεθ. Απόρριψης

Γεννήτορες Ψευδοτυχαίων Αριθμών μη ομοιόμορφης κατανομής

Μέθοδος απόρριψης

```
double fitf(double *x,double *p) {
   double val=p[0]*x[0]*x[0]+p[1]*exp(-0.5*(x[0]-p[2])*(x[0]-p[2])/p[3]/p[3]);
   return val;
}

void random_gauss(int N, int seed)
{
  TH1F *h=new TH1F("h_random_gauss"," ",1000, -10., 10.);
  TRandom *ran=new TRandom3(seed);
  int i=0;
  while(i<N)
    {
     double x=10.*ran->Rndm()-5.;
     double y=2.5*ran->Rndm();
           double rg=0.1*x*x+exp(-0.5*x*x);
     if(rg>y){
               i++; h->Fill(x);
             }
    }
  h->SetLineWidth(2.);
  h->Draw("");
  TF1 *func=new TF1("func",fitf, -5., 5., 4);
  func->SetParameters(100.,100.,0.,1.);
  h->Fit("func"," "," ",-5.,5.);
}
```



MC Ολοκλήρωση: Crude

Απλοϊκό (crude) Monte Carlo

$$I = \int_a^b y(x)dx \quad \Rightarrow \quad I = \frac{b-a}{n}\sum_{i=1}^{n} y(x_i)$$

Όπου τα x_i δεν είναι ισαπέχοντα πλέον, αλλά τυχαία κατανεμημένα

$$\mu_y = E[y] = \frac{1}{b-a}\int_a^b y(x)dx = \frac{I}{b-a} \qquad \sigma_y^2 = V[y] = \frac{1}{b-a}\int_a^b y^2 dx - \mu_y^2$$

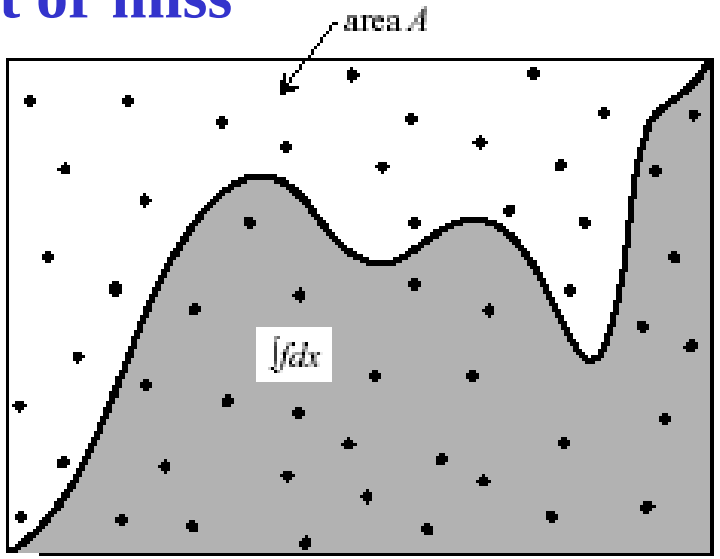
$$\Rightarrow \quad I = \frac{b-a}{n}\sum_{i=1}^{n} y(x_i) \pm (b-a)\frac{\sigma_y}{\sqrt{n}}$$

MC Ολοκλήρωση: Hit or miss

Monte Carlo hit or miss

Γεννάμε δύο ομοιόμορφα τυχαίους αριθμούς. Έναν $\mathbf{x_i}$ στο διάστημα [α,b] και έναν $\mathbf{y_i}$ στο διάστημα $[y_{min},y_{max}]$. Το γεννημένο σημείο θεωρείται:

- επιτυχές (hit) αν $y_i<y(x_i)$
- άστοχο (miss) αν $y_i>y(x_i)$



$$I = \frac{n_{hit}}{n}(b-a)(y_{\max}-y_{\min})+y_{\min}(b-a)$$

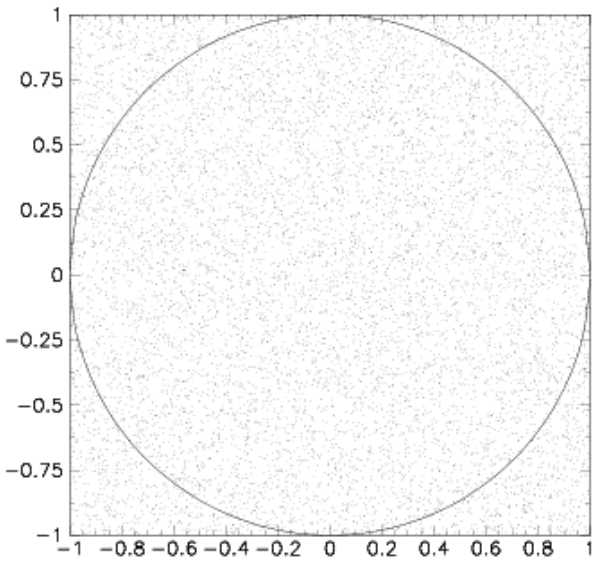
$$V[I] = \frac{1}{n^2}V[n_{hits}](b-a)^2(y_{\max}-y_{\min})^2$$

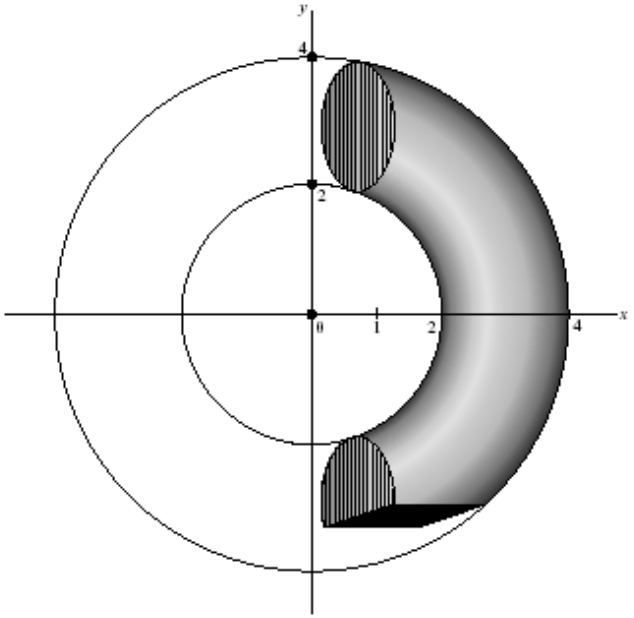
με $V[n_{hits}]=np(1-p)$

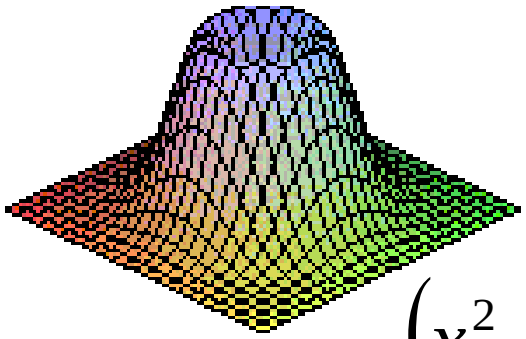
$$\frac{\Delta I}{I} = \frac{1}{\sqrt{n_{hit}}}\sqrt{\left(1-\frac{n_{hit}}{n}\right)}$$

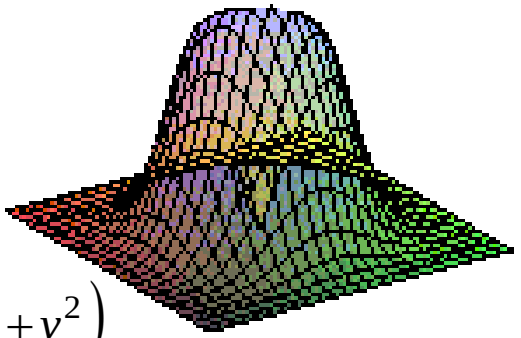
MC Ολοκλήρωση: Παραδείγματα

Monte Carlo hit or miss









$$\left(x^2+y^2\right)e^{-\left(x^2+y^2\right)}$$

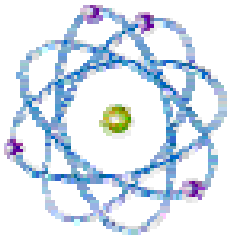
MC Ολοκλήρωση: Σύγκριση

Μέθοδος	Αβεβαιότητα σαν συνάρτηση του αριθμού των σημείων	
	1 διάσταση	D διαστάσεις
Monte Carlo	$n^{-1/2}$	$n^{-1/2}$
Τραπεζίου	n^{-2}	$n^{-2/d}$
Simpson	n^{-4}	$n^{-4/d}$
Gauss τάξης m	n^{-2m+1}	$n^{-(2m-1)/d}$

MC Ολοκλήρωση: Παραδείγματα

Multidimensional Integration

Example: Atomic Physics



$$I = \int_0^1 dx_1 \int_0^1 dx_2 \ldots \int_0^1 dx_{12}\, f(x_1, x_2, \ldots x_{12})$$

^{4}Be

3 Dimension/electron * 4 electrons = 12 Dimensions

For 100 points in each integration there are $100^{12} = 10^{24}$ calculations

Assuming 1 Giga evaluations/sec

It would take over 10^7 years!!!!

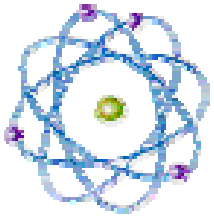
MC Ολοκλήρωση: Παραδείγματα

Multidimensional Integration

via MC mean value

Example: Atomic Physics

$$I = \int_0^1 dx_1 \int_0^1 dx_2 \ldots \int_0^1 dx_{12}\, f(x_1, x_2, \ldots x_{12})$$



^{4}Be

3 Dimension/electron * 4 electrons = 12 Dimensions

$$\simeq (1-0)^{12} * \frac{1}{N} \sum_t^N f(x_1^t \;\; x_2^t \;\; \ldots \;\; x_{12}^t)$$

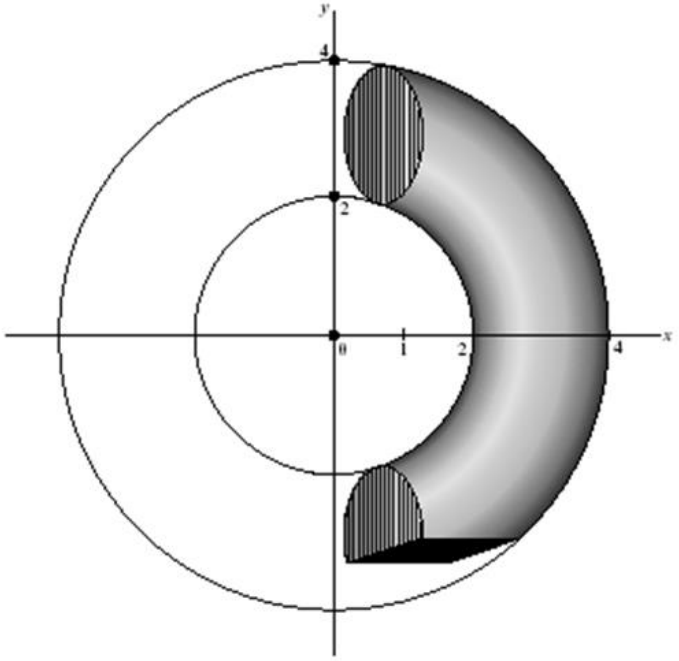
For N=10^6 random points in the MC integration there are ~10^6 calculations

Assuming 1 Giga evaluations/sec

It would take ~ 10^{-3} sec

compare to 7yrs

Μέθοδος απόρριψης - Ολοκλήρωμα με περίεργο σύνορο



$$\int \rho dxdydz$$

$$\int x\rho dxdydz$$

$$\int y\rho dxdydz$$

$$\int z\rho dxdydz$$

Υπολογισμός μάζας και κέντρου μάζας «περίεργου αντικειμένου»

Έστω την τομή τοροειδούς με ορθογώνιο κουτί

$$z^2 + \left(\sqrt{x^2 + y^2} - 3\right)^2 \le 1$$

$$x \ge 1$$

$$y \ge -3$$

Όρια στα οποία κυμαίνονται τα x, y, z

$$4 \ge x \ge 1$$

$$4 \ge y \ge -3$$

$$1 \ge z \ge -1$$

MC Ολοκλήρωση: Παραδείγματα

```
void mc_integral2()
{
  ran=new TRandom3(37999);
  int N=10000;
  double x,y,z,ma,mx,my,mz,em,emx,emy,emz;
  double sm=0,smx=0,smy=0,smz=0,  vm=0,vmx=0,vmy=0,vmz=0;
  double vol=3.*7.*2.;
  for(int i=0; i<N; i++)
    {
      x= 1.+3.*(ran->Rndm());
      y=-3.+7.*(ran->Rndm());
      z=-1.+2.*(ran->Rndm());
      if(z*z+(sqrt(x*x+y*y)-3.)*(sqrt(x*x+y*y)-3.)<1.)
          {
          double den=1.;
          sm+=den; smx+=x*den; smy+=y*den; smz+=z*den;
          vm+=den*den;       vmx+=x*den*x*den;
          vmy+=y*den*y*den; vmz+=z*den*z*den;
          }
    }
  ma=vol*sm/N;
  mx=vol*smx/N/ma; my=vol*smy/N/ma; mz=vol*smz/N/ma;
  em=vol*sqrt((vm/N-sm/N*sm/N)/N);
  emx=vol*sqrt((vmx/N-smx/N*smx/N)/N)/ma;
  emy=vol*sqrt((vmy/N-smy/N*smy/N)/N)/ma;
  emz=vol*sqrt((vmz/N-smz/N*smz/N)/N)/ma;
  cout << "mass = "<< ma <<" +- "<< em  <<endl;
  cout << "xcen = "<< mx <<" +- "<< emx <<endl;
  cout << "ycen = "<< my <<" +- "<< emy <<endl;
  cout << "zcen = "<< mz <<" +- "<< emz <<endl;
}
```

$$\sigma = \sqrt{\langle x^2 \rangle - \langle x \rangle^2}$$

$$\langle x \rangle = \frac{\sum_{i=1}^{N} x}{N}, \quad \langle x^2 \rangle = \frac{\sum_{i=1}^{N} x^2}{N}$$

$$er\left[\langle x \rangle\right] = \sigma / \sqrt{N}$$

```
mass = 21.80 +- 0.21
xcen = 2.418 +- 0.025
ycen = 0.162 +- 0.025
zcen = 0.005 +- 0.007
```

MC Ολοκλήρωση: Μείωση διασποράς

Ολοκλήρωση σε υποδιαστήματα (stratification)

Στην περίπτωση αυτή το υπολογιζόμενο ολοκλήρωμα θα είναι το άθροισμα δύο ή περισσοτέρων ολοκληρωμάτων και η διασπορά του το άθροισμα των διασπορών.
Για να πετύχουμε καλύτερη διασπορά χρειάζεται να γνωρίζουμε τη συμπεριφορά της συνάρτησης.
Παίρνοντας ίσα υποδιαστήματα, τουλάχιστον στη μονοδιάστατη περίπτωση, δεν κινδυνεύουμε να αυξήσουμε τη διασπορά. Βέβαια δεν είναι σίγουρο ότι θα έχουμε και αξιοσημείωτη βελτίωση.

MC Ολοκλήρωση: Μείωση διασποράς

Σημαντική δειγματοληψία (Importance Sampling)

Στην τεχνική λοιπόν αυτή, αλλάζουμε τη μεταβλητή ολοκλήρωσης ώστε να έχουμε ένα ολοκλήρωμα μικρότερης διασποράς

$$I = \int_a^b y(x)dx = \int_a^b \frac{y(x)}{g(x)} g(x)dx = \int_{G(a)}^{G(b)} \frac{y(x)}{g(x)} dG(x) \qquad G(x) = \int_a^x g(x)dx$$

Επομένως πρέπει να βρούμε μία συνάρτηση g(x) τέτοια ώστε:

- Η g(x) να είναι συνάρτηση πυκνότητας πιθανότητας, δηλαδή να είναι παντού θετική και κανονικοποιημένη ώστε G(b)=1.
- Η G(x) να είναι γνωστή αναλυτικά
- Ή η G(x) να μπορεί να λυθεί ως προς x, ή να υπάρχει γεννήτορας τυχαίων αριθμών που να γεννά σημεία x σύμφωνα με τη g(x)
- Ο λόγος y(x)/g(x) να είναι επαρκώς περισσότερο σταθερός από την y(x) ώστε να μειωθεί η διασπορά

Ασταθές σχήμα αν g(x)<<

MC Ολοκλήρωση: Μείωση διασποράς

Importance Sampling: Ένα παράδειγμα

$$I = \int_0^1 e^{-x^2} dx$$

$$g(x) = ae^{-x} \Rightarrow G(x) = a\int_0^x g(x)dx = a\left(1 - e^{-x}\right)$$

όπου $$a = \frac{1}{1 - e^{-1}}$$

Λύνοντας ως προς x, έχουμε:

$$x = -log(1 - G/a)$$

Όπου δειγματοληπτούμε ομοιόμορφα τη G.

Οπότε το ολοκλήρωμα γίνεται:

$$I = \int_0^1 e^{-x^2} dx = \frac{1}{n}\sum_{i=1}^{n} \frac{e^{-x_i^2}}{e^{-x_i}/a}$$

MC Ολοκλήρωση: Μείωση διασποράς

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <TRandom.h>

void mc_integ(int N, int nstr)
{
  ran=new TRandom3(17179);
  double sum=0., sum2=0.;
  double  sum_is=0., sum2_is=0.;
  double volume = 1.;
  double a=1./(1.-exp(-1.*volume));

  for(int i=0;i<N;i++)
    {
      double x=ran->Rndm()*volume;
      double y=exp(-x*x);
      sum+= y; sum2+=y*y;
// Importance sampling
      x=ran->Rndm();
      double x_is=-log(1.-x/a);
      y=exp(-x_is*x_is)/exp(-x_is)/a;
      sum_is+= y; sum2_is+=y*y;
    }
   double  Inte = volume*(sum/N);
   double eInte = volume*sqrt((sum2/N-sum*sum/N/N)/N);
   double  Inte_is = sum_is/N;
   double eInte_is = sqrt((sum2_is/N-sum_is*sum_is/N/N)/N);
   cout<<N<<" "<<Inte<<" +- "<<eInte<<"  " <<Inte_is<<" +- "<<eInte_is< <endl;
```

MC Ολοκλήρωση: Μείωση διασποράς

```
// Stratification
  const int nn=nstr;
  double  sum_st[nn]={0.}, sum2_st[nn]={0.};
  double h=volume/nstr;
  double Inte_st=0;
  double eInte_st=0;
  for(int j=0; j<nstr; j++)
  {
    for(int i=0;i<N/nstr;i++)
    {
      double x=ran->Rndm()*h+j*h;
      double y=exp(-x*x);
      sum_st[j]+= y; sum2_st[j]+=y*y;
    }
  }
  for(int j=0;j<nstr;j++)
  {
    Inte_st +=sum_st[j]*nstr/N;
    eInte_st+=(sum2_st[j]*nstr/N-sum_st[j]*sum_st[j]*nstr*nstr/N/N)*nstr/N;
  }
  Inte_st=h*Inte_st;
  eInte_st=h*sqrt(eInte_st);

  cout<<N<<" "<<nstr<<" "<<Inte_st<<" +- "<<eInte_st<<"  "<<endl;
}
```

MC Ολοκλήρωση: Μείωση διασποράς

Πραγματική τιμή : 0.74682

Importance sampling: Σύγκριση

n	I_{flat}	ΔI_{flat}	I_{exp}	ΔI_{exp}
10^2	**0.76216**	**0.01880**	**0.75082**	**0.00589**
10^3	**0.74683**	**0.00612**	**0.75151**	**0.00172**
10^4	**0.74562**	**0.00199**	**0.74800**	**0.00055**
10^5	**0.74766**	**0.00063**	**0.74690**	**0.00017**

Stratification: Σύγκριση

# διαστημ.	1	2	4	8
$n=10^4$	**0.00199**	**0.00097**	**0.00049**	**0.00025**

MC Ολοκλήρωση: Μείωση διασποράς

Control Variates

$$I = \int y(x)dx = \int [y(x) - g(x)]dx + \int g(x)dx$$

Ευσταθές σχήμα και για g(x)<<

Antithetic Variates

$V[y_1(x)+y_2(x)]=V[y_1(x)]+V[y_2(x)]+2\mathrm{cov}\,[y_1(x),y_2(x)]$

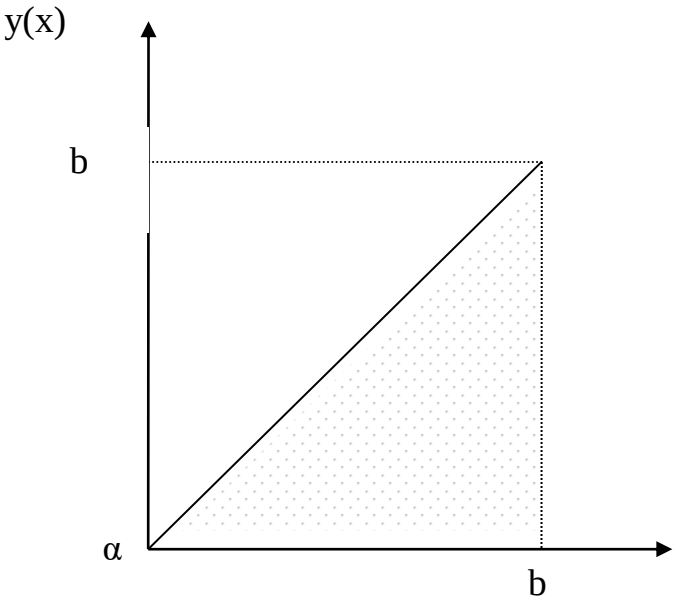
$$I = \int y(x)dx = \int [y_1(x) + y_2(x)]dx$$

$$y_1 = \frac{1}{2}y(x) \qquad y_2 = \frac{1}{2}y(b-(x-a))$$

Για μονοτονικές y(x)

MC Ολοκλήρωση: Εφαρμογή

$$I = \int_a^b dx \int_a^x dy g(x,y)$$



I) Ο προφανής τρόπος:

A) Επιλέγουμε x_i =R[α,b]

B) Επιλέγουμε y_i =R[α,x_i]

Γ) Αθροίζουμε τα $g(x_i,y_i)$ ώστε τελικά να έχουμε: $$I = \frac{(b-a)^2}{2n} \sum_{i=1}^{n} g(x_i, y_i)$$

Παρόλο που αυτή η μέθοδος φαίνεται προφανής, είναι **λανθασμένη!** Ο λόγος είναι ότι τα σημεία (x_i,y_i) δεν είναι ομοιόμορφα κατανεμημένα σε όλη την περιοχή της ολοκλήρωσης.

Υπάρχει περίπου ο ίδιος αριθμός σημείων στην περιοχή α<x<(α+b)/2 όπως και στην περιοχή (α+b)/2<x<b παρόλο που η δεύτερη είναι τριπλάσια σε μέγεθος.

MC Ολοκλήρωση: Εφαρμογή

II) Η μέθοδος της απόρριψης:

Α) Επιλέγουμε x_i=R[α,b] και y_i=R[α,b]

Β) Δημιουργούμε μια νέα συνάρτηση z(x,y), η οποία ορίζεται σε όλο το δισδιάστατο χώρο στον οποίο γεννάμε τους τυχαίους αριθμούς, αλλά έχει το ίδιο ολοκλήρωμα με τη g(x,y) ως:

$z(x_i,y_i) = 0$ αν $x_i<y_i$

$= g(x_i,y_i)$ αν $x_i>y_i$

Το ολοκλήρωμα σ` αυτήν την περίπτωση γίνεται:

$$I = \frac{(b-a)^2}{n}\sum_{i=1}^{n} z(x_i, y_i)$$

MC Ολοκλήρωση: Εφαρμογή

III) Η μέθοδος της απόρριψης *(με γνωστή την περιοχή ολοκλήρωσης)* :

Ένας άλλος τρόπος θα ήταν να υπολογίσουμε το λόγο r του χώρου που καταλαμβάνει η περιοχή ολοκλήρωσης σε σχέση με την περιοχή δειγματοληψίας. Αν γνωρίζαμε το λόγο αυτό, θα μπορούσαμε να περιορίσουμε το σφάλμα, απορρίπτοντας τα σημεία που βρίσκονται εκτός της περιοχής ολοκλήρωσης. Δηλαδή, θα ακολουθούσαμε τα εξής βήματα:

Α) Επιλέγουμε $x_i=R[\alpha,b]$ και $y_i=R[\alpha,b]$

Β) Απορρίπτουμε το σημείο αν δεν ανήκει στην περιοχή ολοκλήρωσης, δηλαδή, αν $y_i>x_i$.

Γ) Αθροίζουμε τα $g(x_i,y_i)$, αντικαθιστώντας τον όγκο δειγματοληψίας με τον όγκο ολοκλήρωσης $r(\alpha-b)^2$. Στο συγκεκριμένο παράδειγμα γνωρίζουμε ότι r=1/2 άρα:

Το ολοκλήρωμα σ` αυτήν την περίπτωση γίνεται:

$$I = \frac{1}{2}\frac{(b-a)^2}{n'}\sum_{i=1}^{n} g(x_i, y_i)$$

MC Ολοκλήρωση: Εφαρμογή

IV) Η μέθοδος της αναδίπλωσης (ένα τρικ)

A) Επιλέγουμε u_i=R[α,b] και v_i=R[α,b]

B) Θέτουμε x_i=max(u_i, v_i) και y_i=min(u_i, v_i)

Το ολοκλήρωμα δίνεται από:

$$I = \frac{1}{2}\frac{(b-a)^2}{n}\sum_{i=1}^{n} g(x_i, y_i)$$

Αυτή η μεθοδολογία ισοδυναμεί με την ομοιόμορφη δειγματοληψία όλου του τετραγώνου και στη συνέχεια την αναδίπλωσή του γύρω από τη διαγώνιο, έτσι ώστε όλα τα σημεία να πέσουν στην περιοχή της ολοκλήρωσης. Η πυκνότητα των σημείων παραμένει ομοιόμορφη.

MC Ολοκλήρωση: Εφαρμογή

V) Η μέθοδος βάρους

Τα πρώτα δύο βήματα είναι ίδια με την πρώτη (λανθασμένη) μέθοδο που αναφέραμε.

Α) Επιλέγουμε x_i =R[α,b]

Β) Επιλέγουμε y_i =R[α,x_i]

Γ) Αλλά, στο τρίτο βήμα πραγματοποιούμε το άθροισμα με βάρος (κάποιο συντελεστή). Το βάρος διορθώνει την ανομοιόμορφη πυκνότητα σημείων (στο παράδειγμα αυτό 1/(x-α)). Επομένως το ολοκλήρωμα δίνεται από:

$$I = \frac{(b-a)}{n}\sum_{i=1}^{n}(x_i - a)g(x_i, y_i)$$

Η μέθοδος αυτή είναι πρακτικά μια εφαρμογή της μεθόδου σημαντικής δειγματοληψίας. Μπορεί να είναι πιο αποδοτική ή όχι από τη μέθοδο αναδίπλωσης, ανάλογα με το αν η (x-α)g ή η g έχει μικρότερη διασπορά.