OLD DOMINION UNIVERSITY

Monte Carlo method I

A. Godunov

1. What is Monte Carlo method?
2. Uniform random number generators (RNG)
3. Non-uniform random number generators

updated 1 February 2022

1

Part 1:

What is Monte Carlo method?

2

What is the most probable number for the sum of two dice?

36 possibilities

6 times – for 7

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

3

Deterministic vs. stochastic

Deterministic model – the output is completely determined by given conditions.

Stochastic model – randomness is imbedded when the output cannot be predicted exactly but only as a probability.
Example: thermal motions, radiative decay, …

Monte Carlo methods can be used for solving both stochastic and (complex) deterministic problems.

Monte Carlo methods may solve previously intractable problems by providing generally approximate solutions.

MC methods can be easier to implement comparing to analytical or numerical solutions.

History – why the method is called Monte Carlo method?
Stanislaw Ulam, John von Neumann, Nicholas Metropolis, …

4

4

The Law of Large Numbers

The Law of Large Numbers is the foundation of MC methods: "The results obtained from performing a large number of trials should be close to the expected value. And it will become closer to the true expected value, the more trials you perform."

5

5

Application

- Physical sciences (both classical and quantum systems)
- Engineering (complex systems)
- Risk management
- Finance and business
- Search and rescue
- Cryptography
- Optimization
- … and many more!

6

6

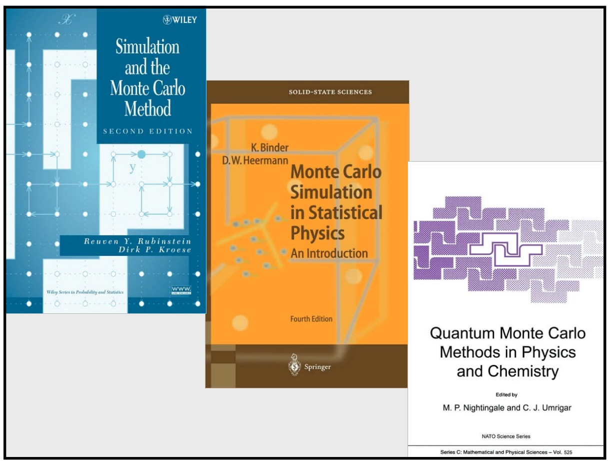
Enormous number of applications

Library of congress: search - books/printed material

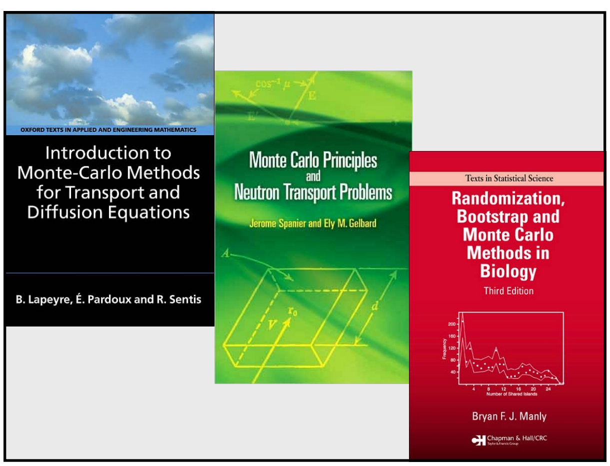
“Monte Carlo method”	1691 results
“Monte Carlo simulation”	640 results
“Monte Carlo physics”	445 results

7

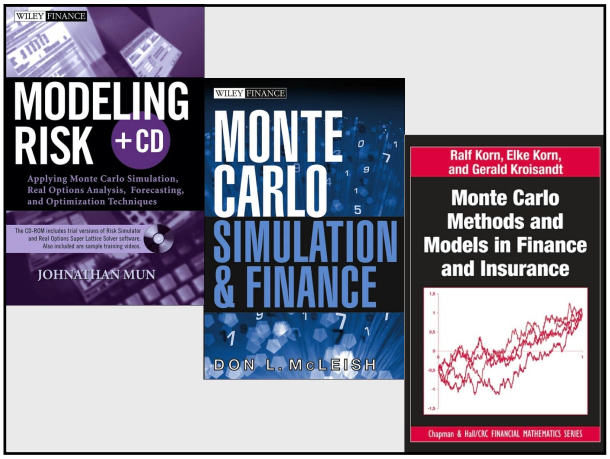
7



8



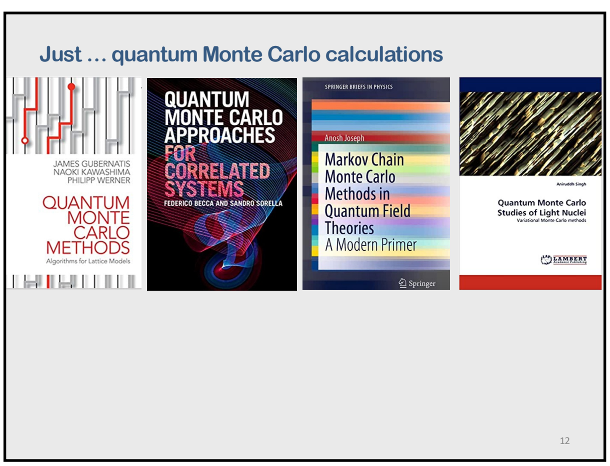
9


WILEY FINANCE
MODELING RISK +CD
Applying Monte Carlo Simulation, Real Options Analysis, Forecasting, and Optimization Techniques
JOHNATHAN MUN
WILEY FINANCE
MONTE CARLO SIMULATION & FINANCE
DON L. McLEISH
Ralf Korn, Elke Korn, and Gerald Kroisandt
Monte Carlo Methods and Models in Finance and Insurance
Chapman & Hall/CRC FINANCIAL MATHEMATICS SERIES

10


Monte Carlo Simulation Methodology for the Reliability of Aircraft Structures
Reliability of Aircraft Structures Under Damage Tolerance Requirements
John Poletti
Optimisation of Source to Image Distance in Diagnostic Radiology
A study using Monte Carlo simulation
Gerhard Winkler
Image Analysis, Random Fields and Markov Chain Monte Carlo Methods
A Mathematical Introduction
Second Edition
Springer

11


Just … quantum Monte Carlo calculations
JAMES GUBERNATIS
NAOKI KAWASHIMA
PHILIPP WERNER
QUANTUM MONTE CARLO METHODS
Algorithms for Lattice Models
QUANTUM MONTE CARLO APPROACHES FOR CORRELATED SYSTEMS
FEDERICO BECCA AND SANDRO SORELLA
SPRINGER BRIEFS IN PHYSICS
Anosh Joseph
Markov Chain Monte Carlo Methods in Quantum Field Theories
A Modern Primer
Springer
Quantum Monte Carlo Studies of Light Nuclei
12

12

. Phys.: Condens. Matter **22** (2010) 023201 (15pp) doi:10.1088/0953-8984/22/2/023201

TOPICAL REVIEW

Continuum variational and diffusion quantum Monte Carlo calculations

R J Needs, M D Towler, N D Drummond and P López Ríos

Theory of Condensed Matter Group, Cavendish Laboratory, Cambridge CB3 0HE, UK

- Three-dimensional electron gas [2–5].
- Two-dimensional electron gas [6–9].
- The equation of state and other properties of liquid ^3He [10, 11].
- Structure of nuclei [12].
- Pairing in ultra-cold atomic gases [13–15].
- Reconstruction of a crystalline surface [16] and molecules on surfaces [17, 18].
- Quantum dots [19].
- Band structures of insulators [20–22].
- Transition metal oxide chemistry [23–25].
- Optical band gaps of nanocrystals [26, 27].
- Defects in semiconductors [28–30].
- Solid-state structural phase transitions [31].
- Equations of state of solids [32–35].
- Binding of molecules and their excitation energies [36–40].
- Studies of exchange–correlation [41–44].

13

Part : 2

Random Number Generators (RNG)

14

Random sequences.

We define a sequence r_1, r_2 … as random if there are no correlations among the numbers. Yet being random does not mean that all the numbers in the sequence are equally likely to occur.

If all the numbers in a sequence are equally likely to occur, then the sequence is called uniform.

Note that 1,2,3,4,… is uniform but not random.

Furthermore, it is possible to have a sequence of numbers that, in some sense, are random but have very short-range correlations among themselves, for example, $r_1, (1 - r_1), r_2, (1 - r_2), r_3, (1 - r_3), \ldots$

Mathematically, the likelihood of a number occurring is described by a distribution function $P(r)$, where $P(r)dr$ is the probability of finding r in the interval $[r, r + dr]$.

A uniform distribution means that $P(r) = a$ constant.

15

15

Sources of Random Numbers

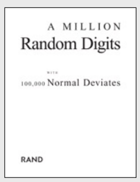
- Tables (in the past)
- Hardware (external sources of random numbers – generates random numbers from a physics process).
- Software (source of pseudorandom numbers)

16

16

Tables …

A Million Random Digits with
100,000 Normal Deviates
by RAND



00000	10097	32533	76520	13586	34673	54876	80959	09117	39292	74945
00001	37542	04805	64894	74296	24805	24037	20636	10402	00822	91665
00002	08422	68953	19645	09303	23209	02560	15953	34764	35080	33606
00003	99019	02529	09376	70715	38311	31165	88676	74397	04436	27659
00004	12807	99970	80157	36147	64032	36653	98951	16877	12171	76833
00005	66065	74717	34072	76850	36697	36170	65813	39885	11199	29170
00006	31060	10805	45571	82406	35303	42614	86799	07439	23403	09732
00007	85269	77602	02051	65692	68665	74818	73053	85247	18623	88579
00008	63573	32135	05325	47048	90553	57548	28468	28709	83491	25624
00009	73796	45753	03529	64778	35808	34282	60935	20344	35273	88435
00010	98520	17767	14905	68607	22109	40558	60970	93433	50500	73998
00011	11805	05431	39808	27732	50725	68248	29405	24201	52775	67851
00012	83452	99634	06288	98083	13746	70078	18475	40610	68711	77817
00013	88685	40200	86507	58401	36766	67951	90364	76493	29609	11062
00014	99594	67348	87517	64969	91826	08928	93785	61368	23478	34113
…..										

17

17

Hardware

Many devices based on physics …

nature > scientific reports > articles > article

Open Access | Published: 04 April 2017

640-Gbit/s fast physical random number generation using a broadband chaotic semiconductor laser

Limeng Zhang, Biwei Pan, Guangcan Chen, Lu Guo, Dan Lu, Lingjuan Zhao & Wei Wang

Scientific Reports **7**, Article number: 45900 (2017) | Cite this article

36 Citations | Metrics

TrueRNG V3 - USB Hardware Random Number Generator
Brand: ubld.it
36 ratings | 17 answered questions
Price: $59.95 & FREE Returns
Available at a lower price from other sellers that may not offer free Prime shipping.

Brand	Ubld.it
Hardware Interface	USB 2.0
Item Dimensions LxWxH	4 x 2 x 0.5 inches

About this item
- High Output Speed: >400 kilobits / second
- Internal Whitening
- Native Windows (XP/8/8.1/10) and Linux Support (CDC Virtual Serial Port)

18

18

Software – pseudo Random Number Generators

- By their very nature, computers are deterministic devices and so cannot create a random sequence.
  Computed random number sequences must contain correlations and in this way cannot be truly random.
- if we know a computed random number r_m and its preceding elements, then it is always possible to figure out r_{m+1}.
  Therefore, computers are said to generate *pseudorandom numbers.*
- While more sophisticated generators do a better job at hiding the correlations, experience shows that if you look hard enough or use pseudorandom numbers long enough, you will notice correlations.

19

19

Good Random Number Generators

Two most important issues:

1. randomness
2. knowledge of the distribution.

Other (still very important) issues

1. long period
2. independent of the previous number
3. produce the same sequence if started with same initial conditions (seed value)
4. fast

20

20

Basic techniques for RNG

The standard methods of generating pseudorandom numbers use modular reduction in congruential relationships.

Two basic techniques for generating uniform random numbers:

1. congruential methods
2. feedback shift register methods.

For each basic technique there are many variations.

The standard random-number generator on computers generates uniform distributions between 0 and 1.

In other words, the standard random-number generator outputs numbers in this interval, each with an equal probability yet each independent of the previous number.

21

21

Linear Congruent Method for a uniform RNG

The linear congruent or power residue method is the common way of generating a pseudorandom sequence of numbers $0 \leq r_i \leq M - 1$ over the interval $[0, M-1]$.

$$x_i = \mathrm{mod}(ax_{i-1} + c, M) = remainder\left(\frac{ax_{i-1} + c}{M}\right) \quad 0 \leq x_{i-1} < M$$

$$\mathrm{mod}(b, M) = b - \mathrm{int}(b/M) * M$$

- starting value x_0 is called "seed"
- coefficients *a* and *c* should be chosen very carefully

the method was suggested by D. H. Lehmer in 1948

22

22

Example:

a=4, c=1, M=9, x_1=3
x_2 = 4
x_3 = 8
x_4 = 6
x_{5-10} = 7, 2, 0, 1, 5, 3

$$x_i = \mathrm{mod}(ax_{i-1} + c, M)$$
$$\mathrm{mod}(b, M) = b - \mathrm{int}(b/M) * M$$

interval: 0-8, i.e. [0,M-1]
period: 9 i.e. M numbers (then repeat)

23

23

Magic numbers for Linear Congruent Method

- M (length of the sequence) must be quite large
- However there must be **no** overflow (therefore for 32 bit machines M=2^{31} ≈ 2*10^9)
- Good "magic" number for linear congruent method (for 32 bit machine):

$$x_i = \mathrm{mod}(ax_{i-1} + c, M)$$

a = 16,807, c = 0, M = 2,147,483,647
for c = 0 "multiplicative congruential generator":

24

24

Random Numbers on interval [A,B]

- Scale results from x_i on [0,M-1] to y_i on [0,1]

$$y_i = x_i / (M-1)$$

- Scale results from x_i on [0,1] to y_i on [A,B]

$$y_i = A + (B-A)x_i$$

25

25

Other Linear Congruential Generators

- Multiple Recursive Generators
  many versions including “Lagged Fibonacci”
- Matrix Congruential Generators
- Add-with-Carry, Subtract-with-Borrow, and Multiply -with-Carry Generators

26

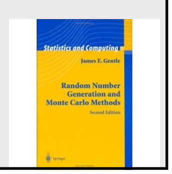
26

Other Generators

- Nonlinear Congruential Generators
- Feedback Shift Register Generators
- Generators Based on Cellular Automata
- Generators Based on Chaotic Systems
- …

James E. Gentle – “Random Number Generation and Monte Carlo Methods

Second edition - 2004



27

Attention!

Before using a random-number generator in your programs, you should check its range and that it produces numbers that "look" random.

Assessing Randomness and Uniformity

1. plots
2. k-th moment of a distribution
3. near-neighbor correlation

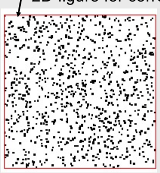
28

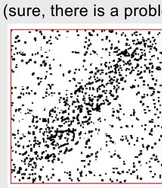
28

1. Plot it.

Plots: Your visual cortex is quite refined at recognizing patterns and will tell you immediately if there is one in your random numbers

- 2D figure, where x_i and y_i are from two random sequences (parking lot test)
- 3D figure (x_i, y_i, z_i)
- 2D figure for correlation (x_i, x_{i+k}) (sure, there is a problem here)





29

29

2. k-th moment

k-th momentum (if the numbers are distributed uniformly)

$$\langle x^k \rangle = \frac{1}{N}\sum_{i=1}^{N} x_i^k \simeq \int_0^1 dx x^k P(x) \simeq \frac{1}{k+1} + O\left(\frac{1}{\sqrt{N}}\right)$$

If the formula above holds for your generator, then you know that the distribution is uniform.

If the deviation varies as $1/\sqrt{N}$, then you also know that the distribution is random because the $1/\sqrt{N}$ result derives from assuming randomness.

30

30

3. Near-neighbor correlation

Taking sums of products for small k:

$$C(k) = \frac{1}{N}\sum_{i=1}^{N} x_i x_{i+k}, \quad (k = 1, 2, \dots)$$

$$\frac{1}{N}\sum_{i=1}^{N} x_i x_{i+k} \simeq \int_0^1 \mathrm{d}x \int_0^1 \mathrm{d}y\, x\, y P(x, y) = \int_0^1 \mathrm{d}y\, x\, y = \frac{1}{4}.$$

If the formula above holds for your random numbers, then you know that they are uniform and independent.

If the deviation varies as $1/\sqrt{N}$, then you also know that the distribution is random.

31

31

Test Suites (most known) for RNG*

the NIST Test Suite (NIST, 2000) includes sixteen tests
http://csrc.nist.gov/groups/ST/toolkit/rng/index.html

"DIEHARD Battery of Tests of Randomness (eighteen tests)
https://en.wikipedia.org/wiki/Diehard_tests

TestU01: includes the tests from DIEHARD and NIST and several other tests that uncover problems in some generators that pass DIEHARD and NIST
http://simul.iro.umontreal.ca/testu01/tu01.html

32

32

Standard RNG in C++

`#include <cstdlib>`	library
`srand(seed)`	is used to initialize the RNG
`rand()`	returns a pseudo random integer in the range 0 to RAND_MAX. RAND_MAX = 32767

Generating integer random numbers in a range i1 – i2:

```
random_i = i1 + (rand()%(i2-i1+1));
```

a better method to do the same

```
random_i = i1 + int(1.0*(i2-i1+1)*rand()/(RAND_MAX-1.0));
```

Generating real random numbers between 0.0 and 1.0

```
drandom = 1.0*rand()/(RAND_MAX-1);
```

33

33

Example: srand and rand in C++

```
// generate integer random numbers between i1 and i2
#include <iostream>
#include <cstdlib>
#include <cmath>
#include <ctime>
using namespace std;

int main ()
{
  int nmax=10;          /* generate 10 random numbers*/
  int i1=1, i2=6, irandom;
  srand (123);          /* initial seed */
//srand(time(NULL)); // better to "randomize" seed values

  for (int i=0; i < nmax; i=i+1)
  {
   irandom = i1+rand()%(i2-i1+1);number between i1 & i2*/
   cout << " " << irandom << endl;
  }
  system("pause");
  return 0;
}
```

```
3
4
6
1
6
2
6
3
5
3
```

34

Example: cont. for float

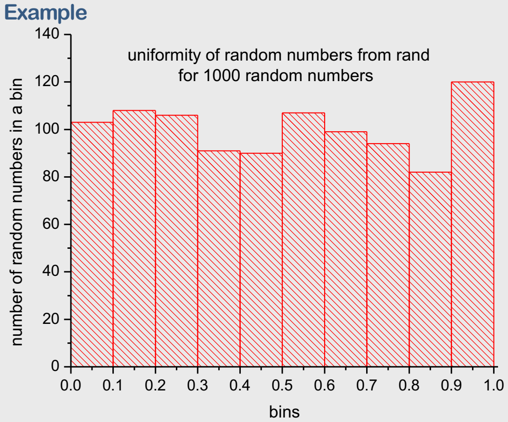
```
/* generate random numbers between 0.0 and 1.0 */
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <cmath>
#include <ctime>
using namespace std;
int main ()
{
  int nmax = 10;    /*generate 10 random number*/
  double drandom;
  cout.precision(4);
  cout.setf(ios::fixed | ios::showpoint);

  srand(4567); /* initial seed value */
  for (int i=0; i < nmax; i=i+1)
  {
      drandom = 1.0*rand()/(RAND_MAX-1);
      cout << "d = " << drandom << endl;
  }
  system("pause");
  return 0;
}
```

```
d = 0.0357
d = 0.7331
d = 0.8495
d = 0.6552
d = 0.1480
d = 0.9866
d = 0.8528
d = 0.3752
d = 0.3467
d = 0.7425
```

35

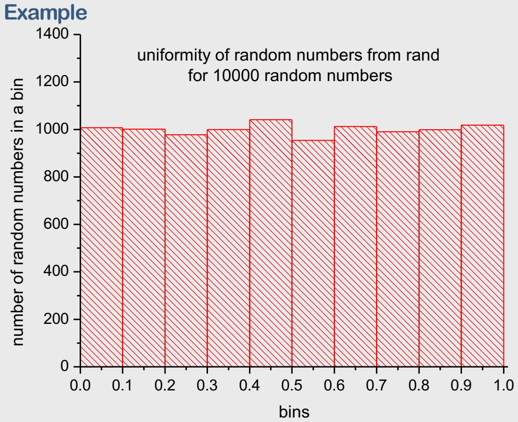
Example



36

36

Example



37

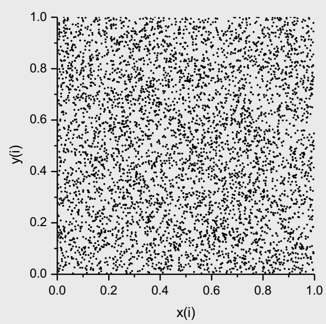
37

Example:

2D distribution for two random sequences x_i and y_i

k-th moment of the random number distribution

two random sequences (parking lot test)



5000 points,
k-th momentum $<x^4>=0.1991$
near-neighbor correlation = 0.2507

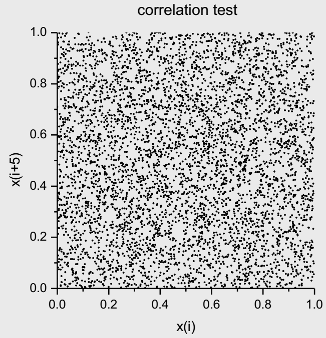
38

38

Example:

2D distribution for correlation (x_i, x_{i+5})

correlation test



5000 points,
k-th momentum $<x^4>=0.1991$
near-neighbor correlation = 0.2507

39

39

Software for RNG

C/C++, Fortran, Python, …
provide built-in uniform random number generators (but for C++ the period is just 2^{31}-1)

but … except for small studies, some of these built-in generators should be avoided.

ATTENTION!
Mersenne Twister* is, by far, today's most popular pseudorandom number generator. It is used by every widely distributed mathematical software package. USE IT!

Period of the generator is $2^{19937}-1$

* developed in 1997 by Makoto Matsumoto and Takuji Nishimura

40

40

Mersenne Twister - RNG in C++

Use an implementation of the Mersenne Twister 19337 algorithm built in <random> header in C++

```
// Create Random Number Generator
random_device rd;
// Used for random seed to generator

mt19937_64 mt(rd());
// Initialize Mersenne twister implementation

uniform_real_distribution<double> dist(xl, xr);
// Set a real uniform distribution over the desired range
```

41

41

Mersenne Twister - Python and MatLab

Python

In Python, ran dom.random() the Mersenne Twister generator.
The best one you can find rather than write your own.

To initialize a random sequence, you need to plant a seed in it.
In Python, the statement random.seed(None) seeds the generator with the system time.

MatLab

In MatLab, rng('default') is the Mersenne Twister generator.

To initialize a random sequence use rng('shuffle') to use seed as current time.

42

42

*Random number generator attacks and defenses

Modern cryptography requires high quality RNG.

Cryptographic attacks that exploit weaknesses in RNGs are known as random number generator attacks.

43

43

Part : 3

Non-uniform Random Number Generators

44

Non-uniform distributions

Most situations in science and engineering demand using random numbers with non-uniform distributions

Examples:

- Radioactive decay (characterized by a Poisson distribution)
- Gauss distribution
- experiments with different types of distributions
- And many more …

45

45

Methods to generate non-uniform distributions

Principal idea: Generating non-uniform random number distributions with a uniform random number generators

Useful methods:

- The transformation method
- The rejection method
- Metropolis algorithm (importance sampling)

46

46

1. The transformation method

The method is based on fundamental property of probabilities.

Consider a collection of variables $\{x_1, x_2, \ldots\}$ that are distributed according to the function $P_x(x)$. Then, the probability to find a value you that lies between x and $x + dx$ is $P_x(x)dx$.

If y is a function of x as $y(x)$, then $|P_x(x)dx| = |P_y(y)dy|$, where $P_y(y)$ is the probability distribution for $\{y_1, y_2, \ldots\}$.

For $P_x = constant = C$ we have

$$\frac{dx}{dy} = \frac{P_y(y)}{C}, \qquad x = \int P_y(y)dy = F(y)$$

Then the non-uniform distribution is the inverse function

$$y(x) = F^{-1}(x)$$

47

47

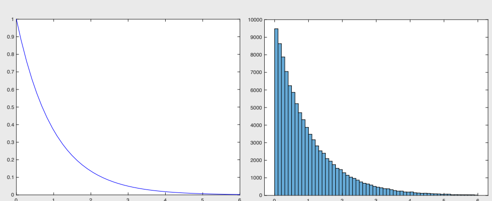
Example 1

1. The Poisson distribution

$$P_y(y) = \exp(-y)$$

Then $x = \int e^{-y} dy = e^{-y}$, $\qquad y = -\ln x$

Thus for a uniform distribution x_i we have $y_i = -\ln x_i$, and the resulting sequence y_i should obey the Poisson distribution



48

48

Example 2

Gaussian distribution is not so easy to derive but here the answer from Box and Muller (Box-Muller method)

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

Let x_1 and x_2 are two independent samples chosen from the uniform distribution on the unit interval $(0, 1)$ then

$$y_1 = \mu + \sigma\sqrt{-2\ln x_1}\cos(2\pi x_2) \text{ or } y_2 = \mu + \sigma\sqrt{-2\ln x_1}\sin(2\pi x_2)$$

49

49

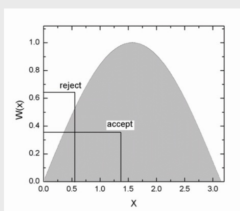
2. The rejection method (von Neuman rejection)

However, very often analytical solutions are not known for the transformation method.

Such situations can be treated by using the rejection method.

Steps:

1. Generate two random numbers x_i on $[x_a, x_b]$ and y_i on $[y_c, y_d]$
2. If $y_i \le w(x_i)$ accept y_i
   If $y_i > w(x_i)$ reject y_i
3. Then y_i so accepted will have the $w(x)$ distribution



50

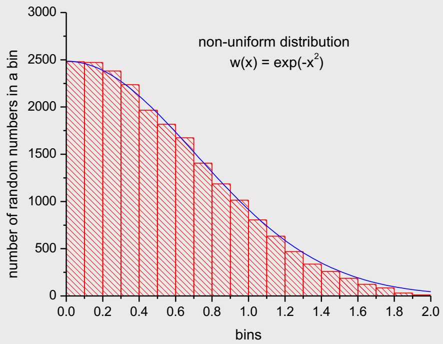
50

Example: w(x)=exp(-x²)

```
double w(double);
int main ()
{
 int nmax = 50000;
 double xmin=0.0, xmax=2.0, ymin, ymax;
 double x, y;
 ymax = w(xmin);
 ymin = w(xmax);
 srand(time(NULL));
 for (double i=1; i <= nmax; i=i+1)
  {
      x = xmin + (xmax-xmin)*rand()/(RAND_MAX+1);
      y = ymin + (ymax-ymin)*rand()/(RAND_MAX+1);
      if (y > w(x)) continue;
      file_3  << " " << x << endl;   /* output to a file */
  }
return 0;
}
/* Probability distribution w(x) */
    double w(double x)
{
    return exp(0.0-1.0*x*x);
}
```

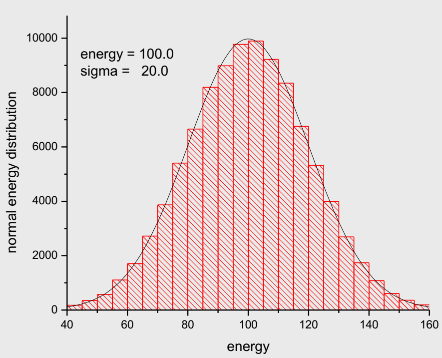
51

calculations



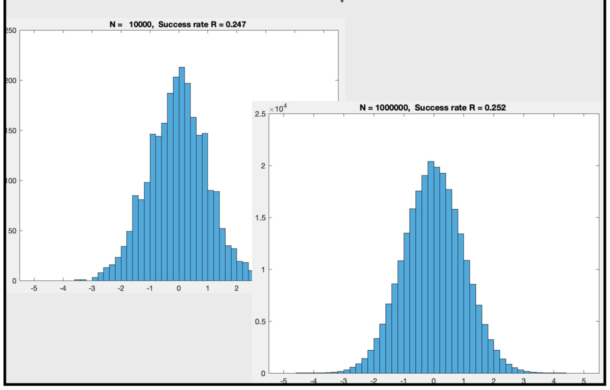
52

calculations



53

Success rate and number of points

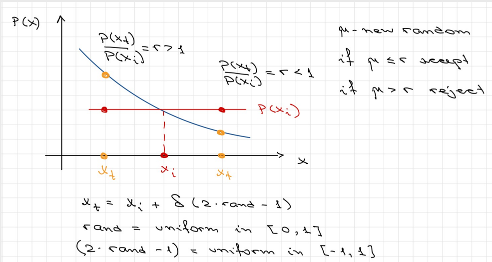


54

3. The Metropolis method

The Metropolis method is a special case of an importance sampling.

Assume that we want to generate random variables $\{x_1, x_2, \ldots\}$ according to $p(x)$. The Metropolis algorithm produces a random walk of points $\{x_i\}$ whose asymptotic probability distribution approaches $p(x)$.



55

55

The algorithm

1. Choose a trial position $x_{trial} = x_i + \delta_i$ where $\delta_i = \delta(2 * rng - 1)$ is a random number in the interval $[-\delta, +\delta]$.
2. Calculate $r = p(x_{trial})/p(x_i)$
   a) If $r \geq 1$ accept the step and let $x_{i+1} = x_{trial}$
   b) If $r < 1$ generate a random number μ between 0 and 1
      i. If $\mu \leq r$ accept the step and $x_{i+1} = x_{trial}$
      ii. If $\mu > r$ reject the step

How do we choose a good step size δ?

- If δ is too large, only a *small fraction* of trail steps will be accepted. If δ is too small, a large fraction of trail steps will it be accepted, but the sampling of the function *will be inefficient.*

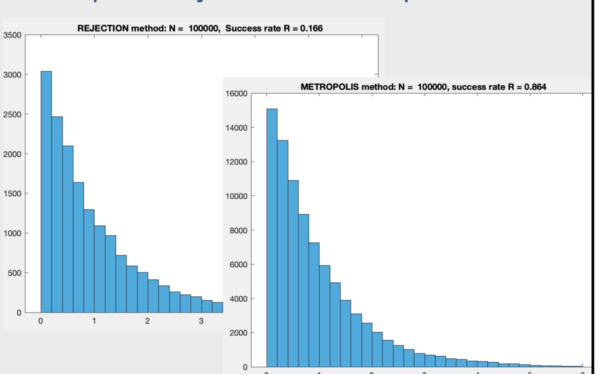
A rough orientation for the magnitude of δ – about a half steps should be accepted.

Also – how to chose x_1? Start at x where $p(x)$ is a maximum.

56

56

Compare the rejection and the Metropolis



57