# Chapter 5
# Monte Carlo method

## 5.1. Central problem of the Monte Carlo method

- Central problem of the Monte Carlo (MC) method
- Lindeberg–Lévy central limit theorem
- Numerical calculation of the mean and variance of random variables
- Calculation of an integral by means on sampling of random variables
- Reduction of an integral to a form of expectation
- Monte Carlo calculation of an integral based on the uniform distribution

## 5.1. Central problem of the Monte Carlo method

**Monte Carlo (MC) method** is a general numerical method for a variety of mathematical problems based on computer generation of (pseudo) random numbers and probability theory. The MC method is usually used in two cases:

- If the solution can be represented in a form of a definite integral (for *calculations*).
- If the solution is a random state of a system that can be predicted by sampling random transitions between various states (for *imitation* of real systems).

When applied to rarefied gas flows in the form of a DSMC method, the MC method is used for both imitation and calculations.

### Central problem of the Monte Carlo (MC) method

The central problem of the MC method is the numerical calculation of a definite integral $I$

(5.1.1)
$$I = \int_a^b g(x)dx \, .$$

The idea of numerical evaluation of $I$ with random numbers is based on the **central limit theorem** (C.L.T., Eq. (4.9.1)-(4.9.3)). The C.L.T. allows one to find values of an integral represented in a form of expectation (mean) of some random variable.

Let's assume that we have a continuous random variable $X$ with PDF $f(x)$ and introduce a new variable $Y = h(X)$, where $h(X)$ is some function. Then **mean** of $Y$ is equal to (Eq. (4.8.1))

(5.1.2)
$$E(Y) = E\big(h(X)\big) = \int_{-\infty}^{+\infty} h(x)f(x)dx \, .$$

## 5.1. Central problem of the Monte Carlo method

**Lindeberg–Lévy Central limit theorem**

Let $Y_n$ $(n = 1,2,\ldots)$ be independent random variables with the same distribution and therefore the same mean $\mu$ and variance $\sigma^2$. Let's introduce a random variables $Y_N$ and $Z_N$:

(5.1.3) $$E_N = \frac{Y_1 + Y_2 + Y_3 + \cdots + Y_N}{N}, \qquad Z_N = \frac{E_N - \mu}{\sigma/\sqrt{N}}.$$

Then variable $Z_N$ is *asymptotically normal* with mean 0 and variance 1, i.e. the distribution function $F_N(x)$ of $Z_N$ satisfies

$$\lim_{N\to\infty} F_N(x) = \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-u^2} du\,.$$

**Consequence**: If $Y = h(X)$, then we can re write Eq. (5.1.2) and (5.1.3) in the form

$$\int_{-\infty}^{+\infty} h(x) f(x) dx = E_N - \frac{\sigma}{\sqrt{N}} Z_N.$$

With increasing $N$, the last term decreases as $1/\sqrt{N}$ (because variance of $Z_N$ approaches 1) and one can estimate the integral in Eq. (5.1.2) as

(5.1.4) $$\int_{-\infty}^{+\infty} h(x) f(x) dx \approx E_N = \frac{Y_1 + Y_2 + Y_3 + \cdots + Y_N}{N} = \frac{h(X_1) + h(X_2) + \cdots + h(X_N)}{N}.$$

Eq. (5.1.4) is the central equation of the MC method: It allows one to approximately calculate value of an integral as an arithmetic mean of multiple random variables distribution.

## 5.1. Central problem of the Monte Carlo method

**Numerical calculation of the mean and variance of random variables**

One can use the C.L.T. in order to numerically estimate the mean and variance (or standard deviation) of a random variable $X$ which values $X_i$ were obtained (e.g., measured in an experiment) $N$ times.

The expectation $E(X)$ can be calculated directly based on Eq. (5.1.3) with $Y = X$:

(5.1.5)
$$E(X) \approx \frac{1}{N}\sum_{i=1}^{N} X_i.$$

$V(X)$ can be also represented in the form of expectation (see Eq. (4.6.4)):

$$V(X) = E([X - E(X)]^2).$$

And, thus, estimated with Eq. (5.1.3) if $Y = [X - E(X)]^2$, i.e.

(5.1.6)
$$V(X) \approx \frac{1}{N}\sum_{i=1}^{N}[X_i - E(X)]^2 = \frac{1}{N}\sum_{i=1}^{N} X_i^2 - 2E(X)\frac{1}{N}\sum_{i=1}^{N} X_i + E^2(X) = \frac{1}{N}\sum_{i=1}^{N} X_i^2 - E^2(X).$$

Eq. (5.1.6) is convenient to use in order to simultaneously estimate $E(X)$ and $V(X)$. Assuming that **N** random values are stored in the array **X**, it can be done with the following C++ code:

```
double E = 0.0, V = 0.0;
for ( int i = 0; i < N; i++ ) {
     E += X[i];
     V += X[i] * X[i];
}
E /= N;
V = V / N – E * E;
```

## 5.1. Central problem of the Monte Carlo method

**Calculation of an integral by means on sampling of random variables**

Calculation of an integral based on the C.L.T. reduces to obtaining multiple independent random variables with the same distribution. The process of obtaining multiple independent realizations of a random variables is referred to as **sampling**. The obtained vector of values $S_N = (Y_1, Y_2, Y_3, \dots, Y_N)$ is called the **sample**, and the number of variables in the sample is called the **sample size**. Special algorithm/code used to sample a random variable with a given distribution is called the **generator** of this random variable.

In order to find $I$ given by Eq. (5.1.1) we need to make three steps:

1. Find such PDF $f(x)$ and function $h(x)$, that

(5.1.7)
$$I = \int_a^b g(x)dx = E\big(h(X)\big) = \int_{-\infty}^{+\infty} h(x)f(x)dx\,.$$

2. Develop a generator of variable $X$ with given PDF $f(x)$.

3. Generate sample of the large size and use Eq. (5.1.4).

The C.L.T. provides us with the estimate for the error of the numerically calculated integral :

(5.1.8)
$$|\mu - E_N| \sim \frac{\sigma}{\sqrt{N}}\,.$$

There are two general ways to decrease the error of the MC estimate: To increase $N$ or to decrease $\sigma$. The convergence of $E_N$ to $\mu$ with increasing $N$ is slow.

## 5.1. Central problem of the Monte Carlo method

Obviously, the choice of functions $f(x)$ and $h(x)$ in Eq. (5.1.7) is non-unique. One can use this freedom and look for such $f(x)$ and $h(x)$ that provides minimum $\sigma$.

**Example**: Use the MC method in order to calculate the integral

$$I = \int_0^1 \cos\left(\frac{\pi}{2}x\right) \sin\left(\frac{\pi}{2}x\right) dx\,.$$

Let's use $X$ in the form of a **standard continuous variable** with uniform distribution in the interval from 0 to 1 and PDF equal to (see slide 23 in Chapter 4)

$$f(x) = \begin{cases} 1, & 0 < x < 1; \\ 0, & x \le 0 \text{ or } x \ge 1. \end{cases}$$

$$\text{Then} \qquad I = \int_0^1 h(x) f(x) dx\,, \qquad \text{where} \qquad h(x) = \cos\left(\frac{\pi}{2}x\right) \sin\left(\frac{\pi}{2}x\right).$$

Let's assume that $X$ can be generated with the computer function **`brng()`**.

C++ implementation of the code for this problem:

```
double Integral ( int N ) // N is the sample size
{
double    E = 0.0, M_PI = 3.1415926;
          for ( int i = 0; i < N; i++ ) {
                    double X = brng ();
                    E += cos ( M_PI * X / 2.0 ) * sin ( M_PI * X / 2.0 );
          }
          return E / N;
}
```

## 5.1. Central problem of the Monte Carlo method

### Reduction of an integral to a form of expectation

Let's consider the general approach to the choice of $h(x)$ and $f(x)$.

(5.1.9) $$I = \int_a^b g(x)dx = E\big(h(X)\big) = \int_{-\infty}^{+\infty} h(x)f(x)dx\,.$$

Let's consider an arbitrary random variable $\tilde{X}$ with a PDF $\tilde{f}(x)$ *which has a non zero values in the interval from $a$ to $b$*: $\tilde{f}(x) > 0$ at $a \le x \le b$. Then let's introduce a new random variable $X$ with the PDF

(5.1.10) $$f(x) = \begin{cases} \dfrac{\tilde{f}(x)}{\int_a^b \tilde{f}(x)dx} & a < x < b \\ 0 & x \le a \text{ or } x \ge b \end{cases}.$$

Here we need to dived by $\int_a^b \tilde{f}(x)dx$ in order to satisfy the normalization condition for PDF $f(x)$, Eq. (4.6.3). Then integral $I$ can be represented in the form

(5.1.11) $$I = \int_a^b g(x)dx = \int_{-\infty}^{+\infty} h(x)f(x)dx, \qquad \text{where} \qquad h(x) = \frac{g(x)}{f(x)}.$$

If possible, $\tilde{f}(x)$ must be chosen in order to reduce variance $\sigma^2$ of $h(X)$.

## 5.1. Central problem of the Monte Carlo method

Two practical requirements to the choice of $f(x)$:

- We should be able to easily sample values of the random variable $X$ with given PDF.
- Random variable must have as small as possible variance.

If, however, large variance is not a concern in the problem under consideration and $a$ and $b$ are finite, then one can use the uniform distribution in Eq. (5.1.11).

**Monte Carlo calculation of an integral based on the uniform distribution**

Let's assume that $a$ and $b$ in Eq. (5.1.9) are finite, then one can always introduce the random variable $X$ with the **uniform distribution** between $a$ and $b$ with PDF (see Eq. (4.7.1))

(5.1.12)
$$f(x) = \begin{cases} 1/(b-a) & a < x < b \\ 0 & x \le a \text{ or } x \ge b \end{cases}$$

Then we can re-write the integral in the form

$$I = \int_a^b g(x)dx = (b-a)\int_{-\infty}^{+\infty} g(x)f(x)dx$$

and the MC estimate for such $f(x)$ takes the form

(5.1.13)
$$I \approx I_N = \frac{b-a}{N}\sum_{i=1}^{N} g(X_i).$$

The estimate $I_N$ can be viewed as an average value of the integrand within the integration interval multiplied by the length of the interval $(b-a)$

## 5.1. Central problem of the Monte Carlo method

Note that Eq. (5.1.13) is similar to the equation for the **rectangle quadrature rule** on a mesh with equal spacing:

$$(5.1.14)\quad I_M = \Delta x \sum_{i=1}^{M} g(x_i), \qquad \text{where} \qquad \Delta x = \frac{b-a}{M}, \qquad x_i = a + (i-1)\Delta x.$$

The MC estimate, Eq. (5.1.13), cannot be more computationally "effective" than the rectangle rule. The MC method, however, can be preferable if

- We need to calculate multidimensional integrals (such integrals regularly appear, e.g., in the statistical physics). When we calculate an $n$-dimensional integral with $M$ nodes in every coordinate direction, the total number of quadrature nodes is equal to $M^n$.
- The domain of integration of multidimensional integrals is geometrically complex.

Under such conditions, the MC estimate can be more effective, i.e. it allows one to find integrals with smaller sample size $N$ than the number of quadrature nodes $M^n$ in Eq. (5.1.14).

Additionally, the MC method can be used to estimate integrals under conditions when $f(x)$ is unknown, but random variables $X_i$ can be obtained as a result of MC imitation of the evolving **random process** (system). This principle is used, in particular, in the Direct Simulation Monte Carlo method for rarefied gas flows.

## 5.2. Random and pseudo-random numbers

- Standard approach to sampling of random variable with given distribution
- Random numbers
- Pseudo-random numbers
- Basic random number generator based on Multiply With Carry (MWC) algorithm

## 5.2. Random and pseudo-random numbers

**Standard approach to sampling of random variable with given distribution**

The standard approach for sampling of random variable $X$ with given distribution is usually based on the transformation of random variables and includes two steps:

1. First, one can develop a method (approach, algorithm, generator) for sampling of **random numbers** $\gamma$, i.e. standard continuous random variable with uniform distribution in the interval from 0 and 1 (See slide 23 in Chapter 4).
2. Next, one can develop an equation (method, algorithm, generator) that allows one to find $X$ by transforming random numbers, i.e. as a function of single or multiple random numbers

$$X = G(\gamma_1, \gamma_2, \gamma_3, \dots). \tag{5.2.1}$$

**Random numbers**

**Random number** is the standard random variable $\gamma$ with uniform distribution is distributed in the interval [0,1] with the PDF

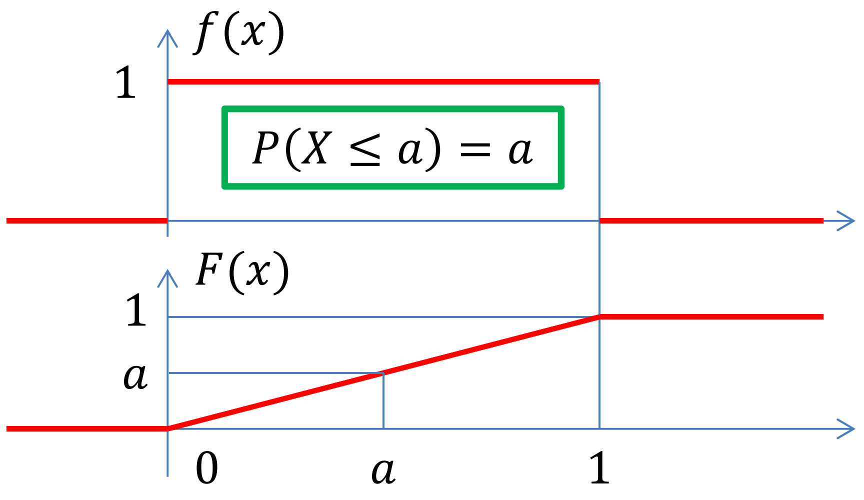
$$f(x) = \begin{cases} 1, & 0 \le x \le 1; \\ 0, & x < 0 \text{ or } x > 1, \end{cases} \qquad F(x) = \begin{cases} 0, & x < 0; \\ x, & 0 \le x \le 1; \\ 1, & x > 1. \end{cases}$$

## 5.2. Random and pseudo-random numbers

Major properties of random numbers:

1. If $0 \le a \le 1$, then

(5.2.2) $$P(\gamma < a) = a.$$



2. Moments of random numbers

(5.2.3) $$M_n(\gamma) = \int_0^1 x^n f(x)dx = \frac{1}{n+1}.$$

3. Mean

(5.2.4) $$E(\gamma) = \int_0^1 x f(x)dx = \frac{1}{2}.$$

3. Variance

(5.2.5) $$\sigma^2 = V(X) = \int_0^1 x^2 f(x)dx - \left(\frac{1}{2}\right)^2 = \frac{1}{3} - \frac{1}{4} = \frac{1}{12}.$$

4. If a random number is written in decimal form, $\gamma = 0.D_1D_2D_3D_4D_5$ …, then every individual digit $D_i$ is a discrete random variable with uniform distribution, i.e. every decimal number 0, 1, 2, 3, …, 9 must occurs in every $D_i$ with the same frequency.

## 5.2. Random and pseudo-random numbers

Practical implementation of true random number with digital computers is a difficult problem. Generation of true random numbers is possible by using carefully designed physical devices. In calculations with digital computers, the true random numbers are replaces with the so-called pseudo-random numbers.

### Pseudo-random numbers

**Pseudo-random numbers** are calculated using some deterministic (not stochastic) approach, and, thus, *they are not random* at all. But the statistical properties of such pseudo-random numbers are close to corresponding properties of true random numbers. In particular, the moments of distribution of pseudo-random numbers are close to moments of random numbers given by Eq. (5.2.3). Practically, it means that large sets (samples) of pseudo-random numbers behave like large samples of true random numbers.

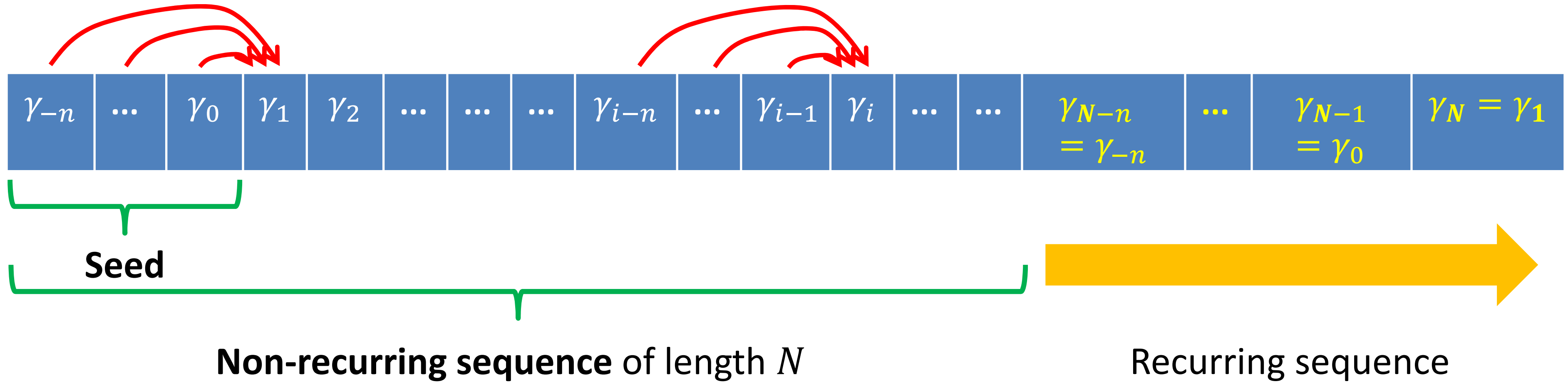
Generators of pseudo-random numbers are often called **basic random number generators** (BRNGs). Usually the BRNGs are implemented in the form of a deterministic equation (algorithm) that allows one to calculate new realization $\gamma_i$ of a random number based on the previous realizations $\gamma_{i-1}, \gamma_{i-2}, \dots$

(5.2.6)
$$\gamma_i = \Gamma(\gamma_{i-1,}\gamma_{i-2}, \dots, \gamma_{i-n}).$$

Every time when we use the same $\gamma_{i-1,}\gamma_{i-2}, \dots, \gamma_{i-n}$ in Eq. (5.2.6), it produces the same $\gamma_i$ and, thus, $\gamma_i$ is not a random variable.

## 5.2. Random and pseudo-random numbers

The process of calculation of pseudo-random number with Eq. (5.2.6) can be illustrated with the following sketch



In order to start calculations at $i = 1$ we need to set values of $\gamma_{i-1,}\gamma_{i-2}, \ldots, \gamma_{i-n}$. These values are called the **seed**. One can prove that for any algorithm with finite $n$ implemented on digital computers, there is only a finite **non-recurring sequence** of length $N$ of pseudo-random numbers. At $i > N$, all numbers will be repeated with **period** $N$. It is dangerous to use a BRNG in order to generate more than $N$ numbers. In this case the results can be affect by the deterministic nature of pseudo-random numbers.

Implementations of BRNGs in computer codes usually include two functions:

**`void SetSeed ( int Seed )`**: To set value of the seed. The seed is often set based on the current system computer time.

**`double brng ()`**: To generate a new pseudo-random numbers.

## 5.2. Random and pseudo-random numbers

Usually, the software packages initialize the seed with a constant value at the start of the code. That is why if **`SetSeed ()`** is not applied, every individual run of the code will provide the same sequence of random numbers. It is convenient for debug of the code, since at every run we will have the same results. For final calculations, **`SetSeed ()`** should be used if simulations are supposed to be restarted. This function is usually called only once at the beginning of the execution of the code. Then after a restart, the simulation will be continued with different sequences of random numbers. The typical C++ code then looks like

```
int main ( int argc, char **argv )
{
time_t    t;
          SetSeed ( unsigned ( time ( &t ) ) );
          // No calls of SetSeed below
float     v1 = brng ();
          ...
}
```

In MATLAB, the BRNG is implemented in the form of functions :

To set seed: **`rng ( seed )`**, where seed is the nonnegative integer seed.

To generate random number: **`rand ()`**

In C++, the standard library does not contain a BRNG.

## 5.2. Random and pseudo-random numbers

The "quality" of various BRNGs is defined by a few factors including:

- Difference of moments of BRNG from moments given by Eq. (5.2.3). Usually, the error increases with increasing order $n$ of moments.
- Difference in distributions of individual digits from uniform distributions if the random number is represented in the form $= 0.D_1D_2D_3D_4D_5\ldots$. Usually, the degree of deviation of distribution for $D_n$ from uniform increases with increasing $n$.
- Size of the period.
- Sensitivity of aforementioned quantities to the choice of the initial seed.

Good BRNGs can be obtained even with $n = 1$ in Eq. (5.2.6), when it reduces to

(5.2.7) $$\gamma_i = \Gamma(\gamma_{i-1}).$$

In particular the practically important generator can be obtained with the simple equation

(5.2.8) $$\gamma_i = \{K\gamma_{i-1}\},$$

where $\{x\}$ denotes the fractional part of real number $x$ and $K = 5^{17}$.

Such simple methods, however, are sensitive to the initial seed, and provide good sequence of pseudo-random number only for specific $\gamma_0$. In addition, Eq. (5.2.8) is not convenient for real calculations, because it requires manipulations with large integer values which do not fit to the integer variable types in programming languages. For real calculations, some equivalent form of Eq. (5.2.8) is usually used.

## 5.2. Random and pseudo-random numbers

**Basic random number generator based on Multiply With Carry (MWC) algorithm**

Simple, but effective and reliable Multiply With Carry (MWC) algorithm was proposed by George Marsaglia. It can be implemented in the form of C++ code as follows:

```
// Variables containing current state of the generator
// Values here correspond to the default state
unsigned int m_u = 521288629, m_v = 362436069;
void SetSeed ( unsigned int u, unsigned int v = unsigned (362436069 ) )
{
         m_u = u;
         m_v = v;
}
unsigned int IntUniform ( unsigned int& u, unsigned int& v )
{
         v = 36969*(v & 65535) + (v >> 16);
         u = 18000*(u & 65535) + (u >> 16);
         return (v << 16) + u;
};
double brng()
{
unsigned  int z = IntUniform ( m_u, m_v );
         return z*2.328306435996595e-10;
}
```

This code initially generates a pseudo-random integer distributed uniformly from 0 to $2^{32}$-1, so the period for this generator cannot be larger than $2^{32}$.

This BRNG in a slightly general form is implemented in the file BRNG.cxx posted on the blackboard.

## 5.2. Random and pseudo-random numbers

Currently, in many software packages, the BRNG implementation is based on the **MT (Mersenne Twister) algorithm** suggested in

M. Matsumoto and T. Nishimura, "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator", ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, p. 3-30.

There are a lot of free C++ implementations of this algorithm available in the internet, see, e.g.,

http://pages.cs.wisc.edu/~stjones/proj/randistrs.c

https://github.com/frt/mtwist/blob/master/randistrs.c

## 5.3. Sampling of discrete random variables

- Standard method of sampling of discrete random variables
- Uniform distribution
- Poisson distribution

## 5.3. Sampling of discrete random variables

### Standard method of sampling of discrete random variables

A random variable $X$ is called **discrete** if it accepts finite or countable number of discrete values. The probability distribution of a discrete random variable is given by the **table of probabilities**

| i | 1 | 2 | 3 | 4 | 5 | 6 | … | i | … |
|---|---|---|---|---|---|---|---|---|---|
| Value | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | … | $x_i$ | … |
| Probability | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ | $p_6$ | … | $p_i$ | … |

Let's find a method for sampling of values of a discrete random variable with given table of probabilities. First, let's introduce **cumulative probabilities** $P_i$

$$P_i = p_1 + \cdots + p_i = \sum_{j=1}^{i} p_j\,, \qquad i = 1,2,\ldots$$

and add to this list a point $P_0 = 0$.

Then the **standard method of sampling of a discrete random variable** can be formulated as follows:

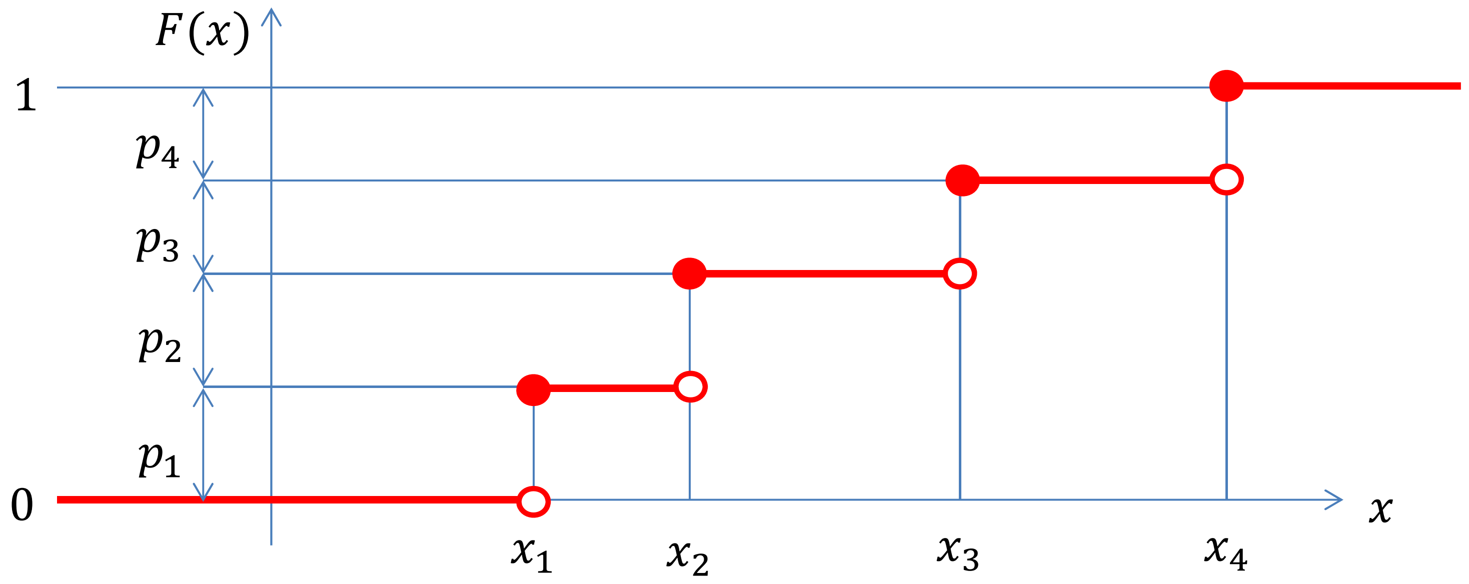
1. Generate a random number $\gamma$.

(5.3.1) 2. Find such $i$ that $P_{i-1} \le \gamma < P_i$.

3. Accept $X = x_i$.

## 5.3. Sampling of discrete random variables

D.F. of $X$ with 4 possible values



In order to prove this algorithm of the standard method, let's simply note that

$$P(X = x_i) = P(P_{i-1} \leq \gamma < P_i) = P(\gamma < P_i - P_{i-1}) = P(\gamma < p_i) = p_i.$$

The standard method can be implemented in the following C++ code:

```
// Pcum is an array of cumulative probabilities
// Individual values are indexed by the index i = 0,...,N-1

double frand_discrete_std ( int N, double *X, double *Pcum )
{
int       i = 0;
double    Gamma = brng ();
          while ( Gamma > Pcum[i] ) i++;
          return X[i];
}
```

## 5.3. Sampling of discrete random variables

For specific distributions, other methods exist. We consider only two examples.

**Uniform distribution**

Let's consider a discrete random variable $K$ which takes integer values from $0$ to $N-1$ with the **uniform distribution**, i.e. with equal probabilities,  i.e. with the probability table

(5.3.2) $$x_i = i - 1, \qquad p_i = \frac{1}{N}.$$

Then such a random variable can be sample with equation

(5.3.3) $$K = [N\gamma],$$

where $[x]$ means the integer part of number $x$.

C++ implementation of this method:

```
int irand_uniform ( int N )
{
          return int ( N * brng () );
}
```

## 5.3. Sampling of discrete random variables

**Poisson distribution**

Let's consider a discrete random variable $K$ which takes integer values from $0$ to $\infty$ with the **Poisson distribution**, i.e. with probabilities (see Slides 16 and 17 in Chapter 4)

(5.3.4) $$p_k = P(K = k) = \frac{a^k}{k!} e^{-a},$$

where parameter $a = E(K) = V(K)$.

It can be proved that such a random variable can be sampled with equation

(5.3.5) $$K = \min \left\{ k = 1,2,\ldots \,\middle|\, \prod_{i=1}^{k} \gamma_i < e^{-a} \right\} - 1.$$

C++ implementation of this method:

```
int irand_Poisson ( double E )
{
double    expE = exp ( - E );
double    prod = 1.0;
int       i = 0;
          do {
                    i++;
                    prod *= brng ();
          } while ( prod >= expE );
          return i - 1;
}
```

## 5.4. Sampling of continuous random variables

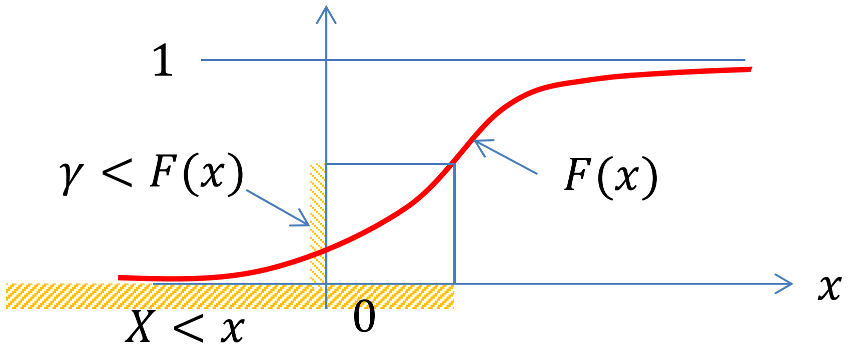
- Standard method of sampling of continuous random variables
- Uniform distribution in a finite interval
- Rayleigh distribution
- Sampling of random vectors with independent coordinates
- Gaussian distribution
- Sampling of random velocities of gas molecules with the Maxwell-Boltzmann distribution
- Acceptance and rejection method
- Calculation of two-dimensional integrals with the acceptance and rejection method
- Acceptance and rejection method for sampling of continuous random variables
- Random vectors with uniform distribution of directions (Isotropic vectors)

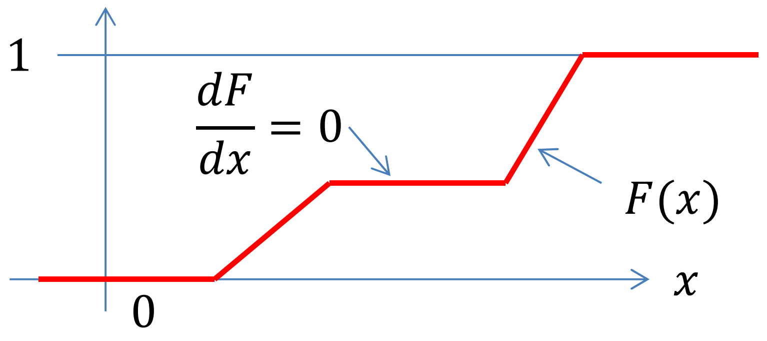
## 5.4. Sampling of continuous random variables

**Standard method of sampling of continuous random variables**

Let' assume that we want to sample a continuous random variable $X$ with a given distribution function $F(x)$ that satisfied the condition

(5.4.1) $$dF/dx > 0 \text{ for all } x \text{ where } 0 < F(x) < 1.$$





It means, in particular, that there is a function $G(y) = F^{-1}(x)$ **inverse** to $F(x)$, i.e.

$$G(F(x)) = x, \qquad F(G(y)) = y.$$

Let's show then that the random variable $X$ can be sampled with the equation

(5.4.2) $$X = G(\gamma) = F^{-1}(\gamma),$$

where $\gamma$ is the random number. In order to prove it, let's show that variable $X$ defined by Eq. (5.4.2) has the distribution function $F(x)$:

$$P(X < x) = P(F^{-1}(\gamma) < x) = P(F(F^{-1}(\gamma)) < F(x)) = P(\gamma < F(x)) = F(x).$$

Equation (5.4.2) defines the **standard method of sampling of continuous random variables**. It can be applied only if condition given by Eq. (5.4.1) is met.

## 5.4. Sampling of continuous random variables

In applications of this method, it is convenient to find equation for $X$ by solving equation

(5.4.3) $$F(X) = \gamma.$$

**Uniform distribution in a finite interval**

Let' assume that we want to sample a random variable $X$ with a uniform distribution in an finite interval $[a, b]$. This random variable has a PDF

(5.4.4) $$f(x) = \begin{cases} 1/(b-a), & a \le x \le b; \\ 0, & x < a \text{ or } x > b, \end{cases}$$

and the distribution function

$$\text{For} \quad a \le x \le b \ : \qquad F(x) = \int_a^x \frac{dx}{b-a} = \frac{x-a}{b-a}.$$

This distribution function satisfies Eq. (5.4.1). Then we need to solve equation

$$F(X) = \gamma$$

with respect to $X$:

$$\frac{X-a}{b-a} = \gamma.$$

Then

(5.4.5) $$X = a + (b-a)\gamma.$$

## 5.4. Sampling of continuous random variables

**Rayleigh distribution**

We say that a non-negative continuous variable has the **Rayleigh distribution** with parameter $\sigma$ if its PDF at $x \geq 0$ is equal to

(5.4.6) $$f(x) = \frac{x}{\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right).$$

Then the distribution function is equal to

$$F(x) = \int_0^x \frac{x}{\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right) dx = 1 - \exp\left(-\frac{x^2}{2\sigma^2}\right).$$

This $F(x)$ satisfies Eq. (5.4.1) and then in order to find $X$ we need to solve the equation

$$F(X) = \gamma \qquad \text{or} \qquad 1 - \exp\left(-\frac{X^2}{2\sigma^2}\right) = \gamma.$$

Then

$$-\frac{X^2}{2\sigma^2} = \log(1-\gamma).$$

But $\gamma$ and $1-\gamma$ have the same distribution, so that

(5.4.7) $$X = \sigma\sqrt{-2\log\gamma}.$$

## 5.4. Sampling of continuous random variables

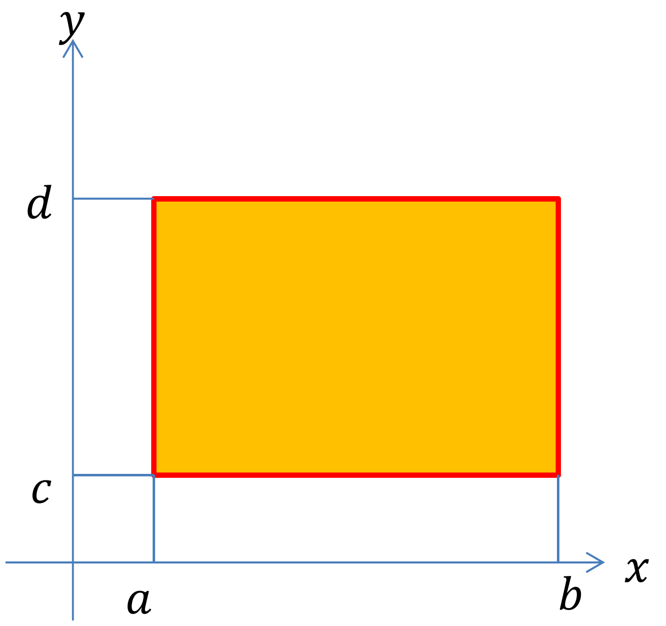
**Sampling of random vectors with independent coordinates**

Let's consider a 2D random vector $\boldsymbol{Z} = (X, Y)$ where individual coordinate $X$ and $Y$ are an **independent** continuous random variables. According to the definition given by Eq. (4.8.8) it means that

(5.4.8)
$$f_{\boldsymbol{Z}}(x, y) = f_X(x) f_Y(x).$$

The individual independent components of random vectors can be sampled independently.

**Example 1**: Let's assume that we want to sample a point with independent coordinates that is uniformly distributed in a rectangle $[a, d] \times [c, d]$.



$$dP = \frac{dxdy}{(b-a)(d-c)} = f_{(X,Y)}(x, y) dxd = f_X(x) f_Y(x) dxdy.$$

$$f_X(x) = \frac{1}{b-a}, \qquad f_Y(x) = \frac{1}{d-c}.$$

Then random coordinates of the point are

(5.4.9)
$$X = a + (b-a)\gamma_1,$$
$$Y = c + (d-c)\gamma_2.$$

## 5.4. Sampling of continuous random variables

**Example 2**: Let's assume that we want to sample a random point uniformly distributed in a circle of radius $R$ with center in the point $(a, b)$.

In order to solve the problem, we can introduce the random variables $\mathcal{R}$ and $E$ corresponding to polar coordinates on the plane $(X - a, Y - b)$.

(5.4.9) $$x = a + r\cos\varepsilon\,, \qquad y = b + r\sin\varepsilon\,, \qquad dA = dxdy = rdrd\varepsilon.$$

Since the point has uniform distribution, then

$$dP = \frac{dA}{\pi R^2} = f_{(X,Y)}(x, y)dxdy = f_{(\mathcal{R},E)}\,(r\cos\varepsilon\ , r\sin\varepsilon)rdrd\varepsilon = f_{(\mathcal{R},E)}(r, \varepsilon)drd\varepsilon,$$

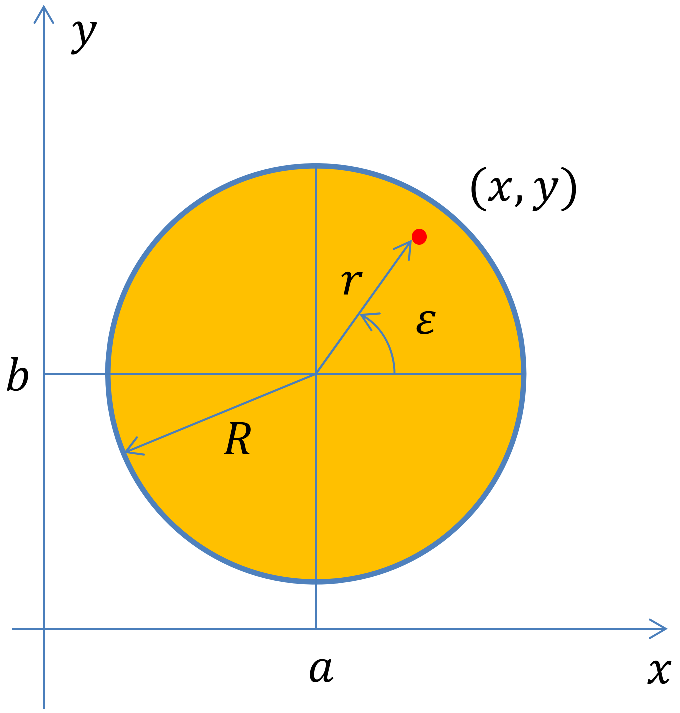
$$f_{(\mathcal{R},E)}(r, \varepsilon) = \frac{r}{\pi R^2} = \frac{2r}{R^2}\frac{1}{2\pi} = f_{\mathcal{R}}(r)f_E(\varepsilon),$$

where

$$f_{\mathcal{R}}(r) = \frac{2r}{R^2}, \qquad f_E(\varepsilon) = \frac{1}{2\pi}$$

Thus, according to Eq. (5.4.8), random variables $\mathcal{R}$ and $E$ are independent. The distribution functions of these variables are equal to

$$F_{\mathcal{R}}(r) = \int_0^r f_{\mathcal{R}}(r)dr = \frac{r^2}{R^2}, \qquad F_E(\varepsilon) = \int_0^\varepsilon f_E(\varepsilon)d\varepsilon = \frac{\varepsilon}{2\pi}.$$



## 5.4. Sampling of continuous random variables

Then equations for sampling $\mathcal{R}$ and $E$ can be found from by solving equations

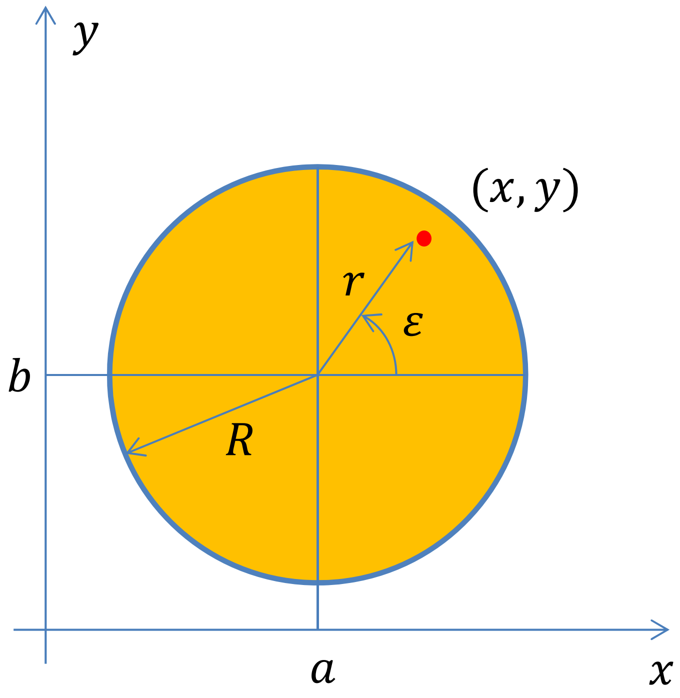
$$\frac{\mathcal{R}^2}{R^2} = \gamma_1, \qquad \frac{E}{2\pi} = \gamma_2,$$

which results to

(5.4.10) $$\mathcal{R} = R\sqrt{\gamma_1}, \qquad E = 2\pi\gamma_2.$$

The equations for $X$ and $Y$ can be found from Eq. (5.4.9):

(5.4.11) $$X = a + \mathcal{R}\cos E\,, \qquad Y = b + \mathcal{R}\sin E\,.$$



C++ implementation of this algorithm (see file Point2DUniformCircleRNG.cxx ):

```
void v2rand_circle ( double &X, double &Y, double a, double b, double R )
{
double     M_PI = 3.1415926;
double     R1 = R * sqrt ( brng () );
double     E1 = 2.0 * M_PI * brng ();
           X = a + R1 * cos ( E1 );
           Y = b + R1 * sin ( E1 );
}
```

## 5.4. Sampling of continuous random variables

### Gaussian distribution

An explicit equation for modelling of the Gaussian distribution cannot be obtained based on the standard method, because there is no explicit equation for $F(x)$. In order to find an equation for sampling a Gaussian random variable let's first consider a vector of two independent Gaussian variables $(X, Y)$ with zero mean, variance equal to 1, and PDFs equal to

$$f_X(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}, \qquad f_Y(y) = \frac{1}{\sqrt{2\pi}} e^{-y^2/2}.$$

Then

$$f_{(X,Y)}(x, y) = f_X(x) f_Y(y) = \frac{1}{2\pi} e^{-(x^2+y^2)/2}.$$

Now let's introduce random polar coordinates $\mathcal{R}$ and $E$ on the plane $(X, Y)$: Then

$$x = r\cos\varepsilon\,, \qquad y = r\sin\varepsilon\,, \qquad dxdy = rdrd\varepsilon, \qquad f_Z(x, y)dxdy = f_Z(r, \varepsilon)drd\varepsilon.$$

$$f_{(\mathcal{R},E)}(r, \varepsilon) = \frac{r}{2\pi} e^{-r^2/2} = f_{\mathcal{R}}(r) f_E(\varepsilon), \qquad f_{\mathcal{R}}(r) = re^{-r^2/2}, \qquad f_E(\varepsilon) = \frac{1}{2\pi}.$$

Thus, $\mathcal{R}$ and $E$ are independent random variables. $\mathcal{R}$ has a Rayleigh distribution with parameter $\sigma = 1$ and $E$ is distributed uniformly between 0 and $2\pi$. Then

(5.4.12) $$\mathcal{R} = \sqrt{-2\log\gamma_1}\,, \qquad E = 2\pi\gamma_2, \qquad x = \mathcal{R}\cos E\,, \qquad y = \mathcal{R}\sin E\,.$$

These equations allow one to obtain two independent samples of variable $X$ simultaneously. One can use both or only one of them as needed.

## 5.4. Sampling of continuous random variables

A Gaussian random variable $Y$ with mean $\mu$ and standard deviation $\sigma$ can be obtained from random variable $X$, which has a Gaussian distribution with mean 0 and standard deviation 1 as follows (see slides 25 and 28 in Chapter 4):

(5.4.13)
$$Y = \mu + \sigma X.$$

Eqs. (5.4.12) and (5.4.13) give the standard method of sampling of Gaussian random variables.

C++ implementation of this method (see file GaussianRNG.cxx):

```
double frand_Gaussian ( double E, double V ) // Here V is the variance
{
double    M_PI = 3.1415926;
          return E + sqrt ( - 2.0 * V * log ( brng () ) ) * cos ( 2.0 * M_PI * brng () );
}
```

### Sampling of random velocities of gas molecules with the Maxwell-Boltzmann distribution

Let's assume that we want to generate a random velocity vector $\mathbf{V}$ of a molecule in the gas in the equilibrium state, where the velocity distribution function is equal to the Maxwell-Boltzmann distribution function given by Eq. (3.6.9):

$$f_M(\mathbf{v}) = \frac{n}{(2\pi RT)^{3/2}} \exp\left[-\frac{(\mathbf{v}-\mathbf{u})^2}{2RT}\right].$$

## 5.4. Sampling of continuous random variables

In order to develop a method for sampling of random components of the velocity vector $\mathbf{V} = (V_x, V_y, V_z)$, the Maxwell-Boltzmann distribution function must be first turned into the form of a joint PDF. Any PDF must satisfy the normalization condition, Equation (4.6.2). It means that the PDF of $\mathbf{V}$ is equal to

$$f_{(V_x,V_y,V_z)}(v_x, v_y, v_z) = \frac{f_M(\mathbf{v})}{n} = \frac{1}{(2\pi RT)^{3/2}} \exp\left[-\frac{(\mathbf{v}-\mathbf{u})^2}{2RT}\right] = f_{V_x}(v_x) f_{V_y}(v_y) f_{V_z}(v_z),$$

(5.4.14)

$$f_{V_x}(v_x) = \frac{1}{(2\pi RT)^{1/2}} \exp\left[-\frac{(v_x-u_x)^2}{2RT}\right].$$

These equations show that

- Velocity components $\mathbf{V} = (V_x, V_y, V_z)$ are independent random variables;
- PDF of an individual velocity component, e.g., $f_{V_x}(v_x)$ is the Gaussian PDF with expectation $u_x$ and standard deviation $\sqrt{RT}$.

Sampling of velocity components from the Maxwell-Boltzmann distribution reduces to sampling of Gaussian random variables and can be performed with the following C++ code:

```
void vrand_MB ( double *v, double m, double *u, double T )
{
double  RT = BOLTZMANN_CONSTANT * T / m;
        v[0] = frand_Gaussian ( u[0], RT );
        v[1] = frand_Gaussian ( u[1], RT );
        v[2] = frand_Gaussian ( u[2], RT );
}
```

## 5.4. Sampling of continuous random variables

### Acceptance and rejection method

Let's consider a point $\boldsymbol{P}$ with random coordinates $(X, Y)$ that is *uniformly* distributed in a finite domain $D$ of complex shape on the plane $(x, y)$. The **uniformity** of distribution means that

(5.4.14) $$P(\text{point } \boldsymbol{P} \text{ inside } dA) = \frac{dA}{A},$$

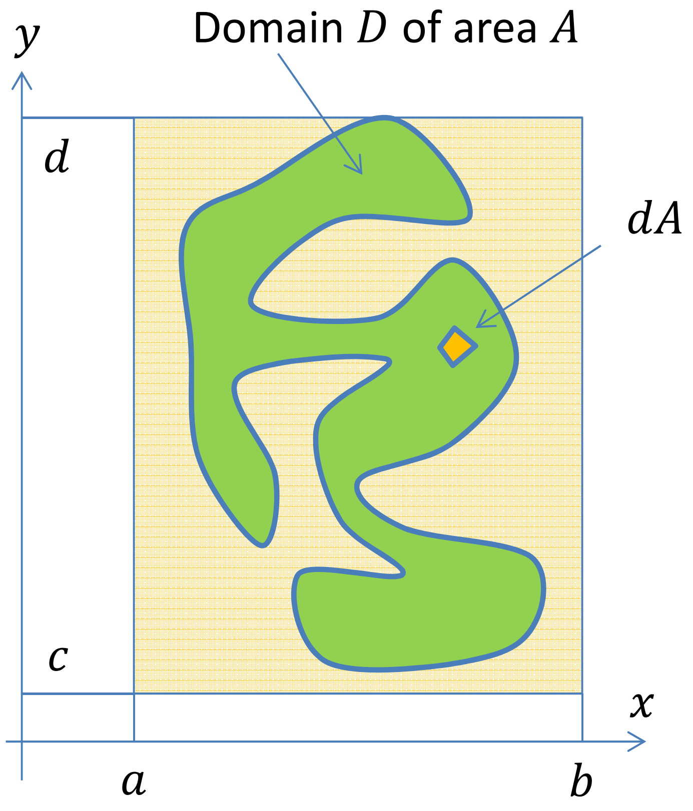
where $A$ is the area of $D$. If $D$ is not a rectangle with faces parallel to axes $Ox$ and $Oy$, then random variables $X$ and $Y$ are not independent (see example in slide 32 of Chapter 4). But they can be sampled with the **acceptance and rejection method** as follows

1. Let's find such rectangle $[a, b] \times [c, d]$ that includes the full domain $D$.

2. Generate a random point $(X, Y)$ with uniform distribution inside the rectangle $[a, b] \times [c, d]$:

(5.4.15) $$X = a + (b - a)\gamma_1, \quad Y = c + (d - c)\gamma_2.$$

3. Check the position of the point $(X, Y)$ with respect to $D$. If the point inside $D$, then $(X, Y)$ is *accepted* as the next realization of coordinates of the random point. If not, the point is *rejected* and algorithm returns to step 2. Random points must be generated in a loop until a point inside $D$ is found.



## 5.4. Sampling of continuous random variables

**Calculation of two-dimensional integrals with the acceptance and rejection method**

$$I = \iint_D g(x,y)dxdy$$

First, let's represent this integral in the form of an expectation:

$$I = A \iint_D g(x,y) f_{(X,Y)}(x,y)dxdy, \qquad f_{(X,Y)}(x,y) = \frac{1}{A}.$$

Here $A$ is the area of $D$ and $f_{(X,Y)}(x,y)$ is the joint PDF of $(X,Y)$ with uniform distribution in $D$. Then the integral $I$ can be estimated as

(5.4.16) $$I \approx I_N = \frac{A}{N}\sum_{i=1}^{N} g(X_i,Y_i),$$

The estimate $I_N$ can be viewed as an average value of the integrand within the domain $D$ multiplied by the area $A$ of the domain: Compare with Eq. (5.1.13)

where $(X_i,Y_i)$ are coordinates of a random point distributed uniformly in the domain $D$. These random coordinates can be generated with the acceptance and rejection method. The area of the domain D can be also calculated with the acceptance and rejection method as follows:

$$\frac{A}{(b-a)(d-c)} = P(\text{Point } (X,Y) \text{ with coordinates in Eq. (5.4.15) is inside } D),$$

(5.4.17) $$A = (b-a)(d-c)\frac{\text{Number of accepted points } N}{\text{Total number of points } N_{tot} \text{ generated with Eq. (5.4.15)}}.$$

## 5.4. Sampling of continuous random variables

Then Eq. (5.4.16) can be re-written as

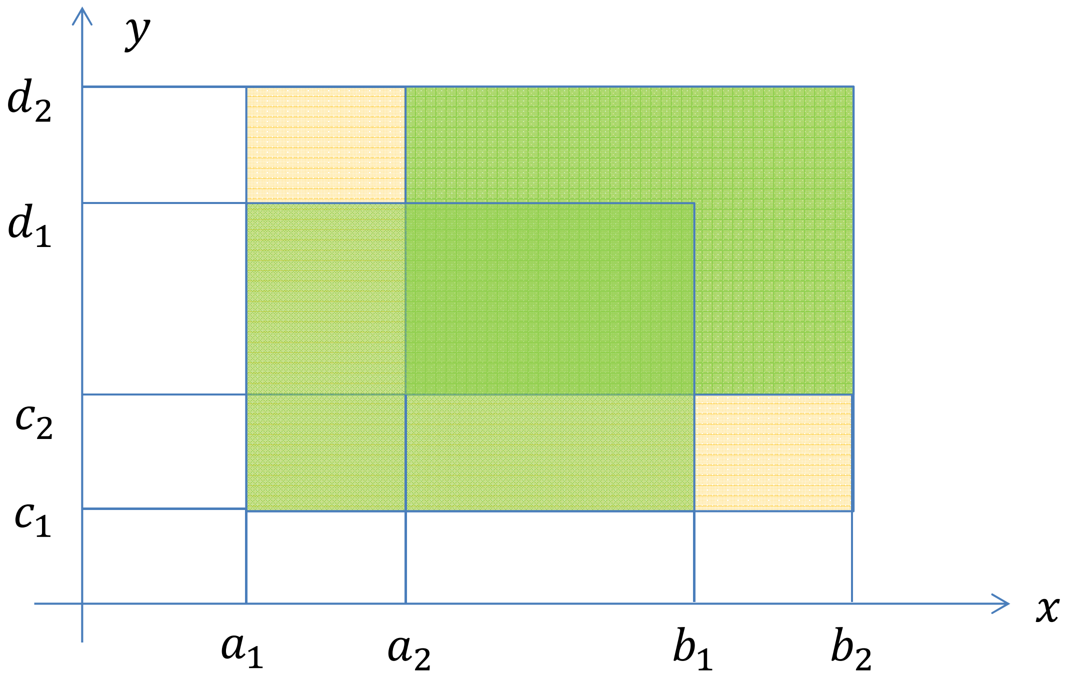
(5.4.18)
$$I \approx I_N = \frac{(b-a)(d-c)}{N_{tot}} \sum_{i=1}^{N} g(X_i, Y_i) \, .$$

In order to implement this method for practical calculations we only need to develop of computer function that checks whether a point with given coordinates $(x, y)$ is inside $D$ or not.

**Example**: Calculate the integral

$$I = \iint_D (x+y)^2 \sin e^{-x} \, dxdy \, ,$$

where $D$ is the superposition of two partially overlapping rectangles as shown in the sketch by green. In this case point $(x, y)$ belongs to $D$ if and only if it belongs at least to one of rectangles.



## 5.4. Sampling of continuous random variables

Then the MC estimate of $I$ can be obtained at arbitrary $a_i$, $b_i$, $c_i$, and $d_i$ $(i = 1,2)$ with the following C++ code:

```
int     N = 100000; // Desired size of the sample of accepted points
double  a[2], b[2], c[2], d[2]; // Coordinates of the rectangle edges
bool IsPointInsideD ( double X, double Y ) ////////////////////////////////////////////////
{
        for ( int i = 0; i < 2; i++ )
                if ( X >= a[i] && X <= b[i] && Y >= c[i] && Y <= d[i] ) return true;
        return false;
}
double Integral() // This function returns value of I_N ////////////////////////////////////
{
double  x_min = ( a[0] < a[1] ) ? a[0] : a[1];
double  x_max = ( b[0] > b[1] ) ? b[0] : b[1];
double  y_min = ( c[0] < c[1] ) ? c[0] : c[1];
double  y_max = ( d[0] > d[1] ) ? d[0] : d[1];
int     Ntot = 0; // Total number of generated points in the acceptance and rejection method
double  IN = 0.0;
        for ( int i = 0; i < N; i++ ) {
                double X, Y;
                do {
                    X = x_min + ( x_max - x_min ) * brng ();
                    Y = y_min + ( y_max - y_min ) * brng ();
                    Ntot++;
                } while ( !IsPointInsideD ( X, Y ) );
                IN += ( X + Y ) * ( X + Y ) * sin ( exp ( - X ) );
        }
        return (( x_max - x_min ) * ( y_max - y_min ) * IN / Ntot ); // Eq. (5.4.18)
}
```

## 5.4. Sampling of continuous random variables

**Acceptance and rejection method for sampling of continuous random variables**

Let's consider a continuous random variable $X$ with given PDF $f(x)$, which satisfies two conditions:

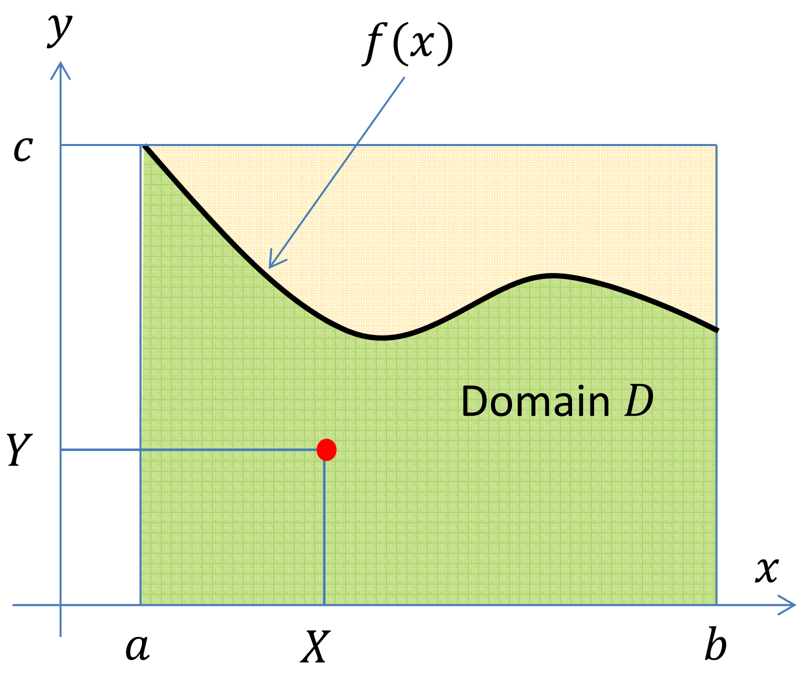
- $f(x) \neq 0$ in a finite interval [a,b] and $f(x) = 0$ outside $[a, b]$.
- Values of $f(x)$ are finite inside $[a, b]$, i.e. such constant $c$ exists that

$$f(x) < c \quad \text{for all} \quad a \leq x \leq b.$$

One can prove that if we introduce a random vector $(X, Y)$ which corresponds to a point distributed uniformly in the domain $D$ (greed area in the sketch) bounded by the axis $Ox$, plot of the function $f(x)$, and vertical lines $x = a$ and $x = b$, then random coordinate $X$ of this point has PDF $f(x)$.

The proof of this statement can be performed based on the notions of conditional probabilities and conditional PDFs.

Then the sampling of random variable $X$ reduces to the sampling of a point in the domain $D$ with the acceptance and rejection method.



## 5.4. Sampling of continuous random variables

If one preliminary found $c$, then the algorithm of the acceptance and rejection method for sampling of $X$ can be formulated as follows:

1. Generate a random point with coordinates $(X, Y)$

$$X = a + (b - a)\gamma_1, \qquad Y = c\gamma_2.$$

2. Check the condition

$$Y \le f(X). \tag{5.4.19}$$

If this condition is satisfied, then $X$ is accepted as the value of random variable with PDF $f(x)$, otherwise the algorithm returns to step 1 and new random point is generated .

C++ implementation of the acceptance and rejection method (see file ARMRNG.cxx):

```
double frand_arm ( double a, double b, double c, double (*pdf) ( double X ) )
{
double    X, Y;
          do {
                    X = a + ( b - a ) * brng ();
                    Y = c * brng ();
          } while ( Y > pdf ( X ) ); // Condition opposite to Eq. (5.4.19)
          return X;
}
```

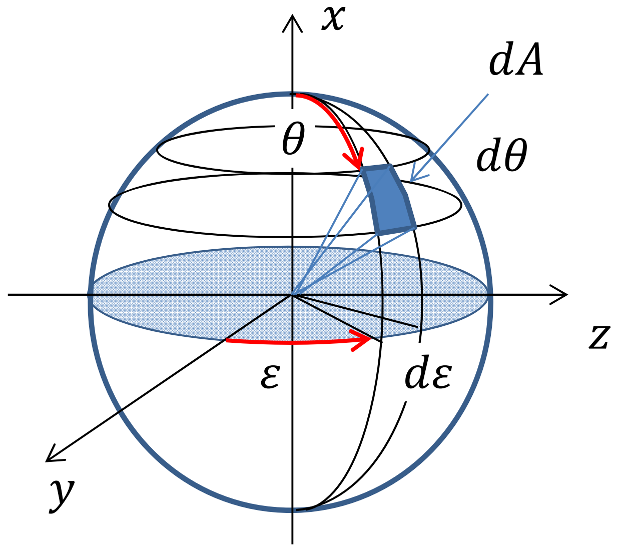
## 5.4. Sampling of continuous random variables

**Random vectors with uniform distribution of directions (Isotropic vectors)**

2D unit random vector $\boldsymbol{R} = (X, Y)$ has **uniform distribution of directions**, it its end is distributed uniformly at the circumference of a unit circle. Since in this case the polar angle $E$ has a uniform distribution, components of $\boldsymbol{R}$ can be sampled with equations

$$E = 2\pi\gamma,$$

$$X = \cos E\,, \qquad Y = \sin E\,.$$



3D unit random vector $\boldsymbol{R} = (X, Y, Z)$ has **uniform distribution of directions**, it its end is distributed uniformly at the surface of a unit sphere. If we introduce spherical angles $(\theta, \varepsilon)$ that characterize the direction of unit vector with components $(x, y, z)$, then

$$x = \cos\theta\,, \qquad y = \sin\theta\cos\varepsilon\,, \qquad z = \sin\theta\sin\varepsilon\,.$$

Then the direction of $\boldsymbol{R}$ can be characterized by random values $(T, E)$ of angles $(\theta, \varepsilon)$. Uniformity of the directional distribution of $\boldsymbol{R}$ means that

$$P(\theta < T \le \theta + d\theta, \varepsilon < E \le \varepsilon + d\varepsilon) = \frac{dA}{4\pi} = \frac{\sin\theta d\theta d\varepsilon}{4\pi} = f_{(T,E)}(\theta, \varepsilon) d\theta d\varepsilon.$$

Then the PDF of $(T, E)$ is

$$f_{(T,E)}(\theta, \varepsilon) = f_T(\theta) f_E(\varepsilon), \qquad f_T(\theta) = \frac{\sin\theta}{2}, \qquad f_E\,(\varepsilon) = \frac{1}{2\pi}.$$

## 5.4. Sampling of continuous random variables

Thus, $T$ and $E$ are independent random variables with distribution functions:

$$F_T(\theta) = \int_0^\theta f_T(\theta)\,d\theta = 1 - \frac{\cos\theta}{2}, \qquad F_E(\theta) = \int_0^\varepsilon f_E(\varepsilon)\,d\varepsilon = \frac{\varepsilon}{2\pi}.$$

Thus, $\cos T$ and $E$ have uniform distribution and random components of $\boldsymbol{R} = (X, Y, Z)$ can be sampled as follows:

(5.4.19)

$$\cos T = 1 - 2\gamma_1, \qquad E = 2\pi\gamma_2,$$
$$\sin T = \sqrt{1 - \cos^2 T},$$
$$X = \cos T, \qquad Y = \sin T \cos E, \qquad Z = \sin T \sin E.$$

C++ implementation of this algorithm (see file IsotropicVectorRNG.cxx):

```
void v3rand_isotropic ( double *R )
{
double     M_PI = 3.1415926;
double     cosT = 1.0 - 2.0 * brng ();
double     sinT = sqrt ( 1.0 - cosT * cosT );
double     E = 2.0 * M_PI * brng ();
           R[0] = cosT;
           R[1] = sinT * cos ( E );
           R[2] = sinT * sin ( E );
}
```

Random vectors with uniform distribution of directions are sometimes called **isotropic**.

## 5.5. Sampling of random events

### Sampling of random events

Let's assume that in the course of a simulation some random event $A$ (change of state of a system under consideration) occurs with given probability $P(A)$. How can we sample/draw such random event during the simulations process?

**Examples**:

1. We have two molecules $i$ and $j$ which collide each other during time $\Delta t$ with given probability $P_{ij} < 1$. How can we decide whether collision happens or not during $\Delta t$?

2. We simulate an excited atom that can emit a photon during time $\Delta t$ with known probability $P_\nu < 1$. How can we decide whether the random emission occurs or not?

Sampling of random evens during Monte Carlo simulations is based on the major property of random numbers given by Eq. (5.2.2):

$$\text{If} \quad 0 \le a \le 1 \quad \text{then} \quad P(\gamma < a) = a.$$

We can apply this property to $a = P(A)$:

(5.5.1) $$P(\gamma < P(A)) = P(A),$$

i.e. probabilities of random events $A$ and $\gamma < P(A)$ are identical. It means that every time when we need to decide whether the random even occurs or not, we can use the following approach:

1. Sample $\gamma$ and check whether $\gamma < P(A)$ or not.

2. If it is true, then the random even occurs, otherwise it does not occur.