Numerical methods in Fortran: Lec. I & II

Devendra Singh Bhakuni

dbhakuni@ictp.it

/afs/ictp/public/d/dbhakuni/NM-Fortran.pdf

October 22, 2023

Topics to be covered

- Random numbers generators, middle square method, linear congruent method, in-built FORTRAN random number generators.

- Non-uniform random number generators: Transformation method, exponential and Gaussian random number generator.

- Rejection method and Monte-Carlo integration.

- Non parametric density estimation.

Topics to be covered

- *Random numbers generators, middle square method, linear congruent method, in-built FORTRAN random number generators.*

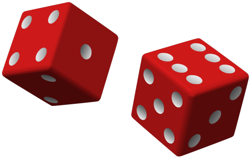
- *Non-uniform random number generators: Transformation method, exponential and Gaussian random number generator.*

- Rejection method and Monte-Carlo integration.

- Non parametric density estimation.

Random number generators and their uses

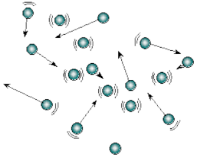
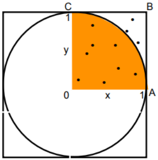
- Generates a sequence of numbers that can not be predicted.

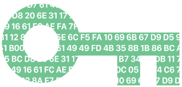
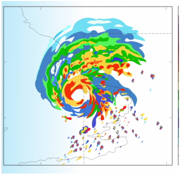


◇ Physical process that contains randomness: radioactive decay, thermal motion.

◇ Monte Carlo integration: multi-dimensional integrals.

◇ Simulation in Classical Statistical and Quantum Mechanics.

True random number generators

◇ Computer must use some external physical variable that is unpredictable, such as radioactive decay of isotopes and atmospheric noise.

◇ Truly random numbers are difficult to generate because they are not cost-efficient.

◇ Slow to generate, no reproducibility.

◇ Required for cryptography.

Pseudo random number generators

◇ Long sequences of numbers $(x_1, x_2, x_3, ...)$ generated by an algorithm on a computer. Look "nearly random" but completely deterministic!

◇ They have statistical properties similar to true random numbers and usually distributed uniformly in range $[0,1]$: uniform distribution:

$$P(x) = \begin{cases} 1 & \text{if } 0 \le x \le 1, \\ 0 & \text{otherwise.} \end{cases} \qquad P(x) = \begin{cases} \frac{1}{(b-a)} & \text{if } a \le x \le b, \\ 0 & \text{otherwise.} \end{cases}$$

◇ To be precise, the algorithm generates integers I_n between 0 and M, and returns a real value. The real value is : $x_n = \frac{\text{float}(I_n)}{M}$.

◇ **Correlations**: $\begin{cases} P(x_n, x_{n+j}) \simeq P(x_n)P(x_{n+j}) \\ P(x_n, x_{n+j}, x_{n+k}) \simeq P(x_n)P(x_{n+j})P(x_{n+k}). \end{cases}$

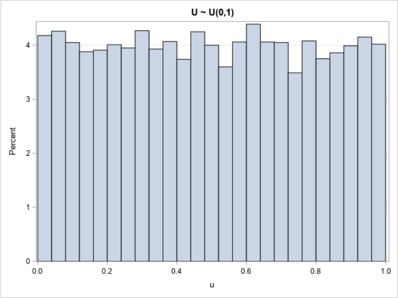
Ideally (no correlations) we have equalities!

Pseudo random numbers generation - Algorithms

Two among the simplest (and oldest) algorithms:

- ◇ The von Neumann method (Middle square algorithm)
- ◇ The Linear Congruential Method (LCM)

We will eventually use the in-built FORTRAN function to generate random numbers!



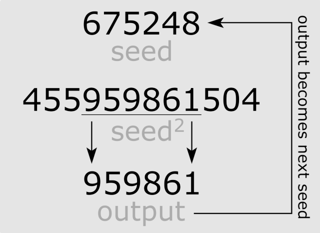
von Neumann (Middle square) algorithm

To generate a d-digit integer sequence:

- take any d digit number
- square it
- take the middle d digits of the result

loop it.

Limitation:
depending on the initial choice, you can be trapped into short loops.



Algorithm

Data: Seed X_0
Result: Random numbers
while *true* **do**
    $X_0 \leftarrow X_0^2$;
    Extract the middle digits as the random number;
    Output the random number;
end

Hint: To get the middle d digits: $M_1 = \frac{M}{10^{d/2}}$ and then $\mathrm{mod}(M_1, 10^{2d})$.

Linear Congruential Method (LCM)

The Linear Congruential Method is defined by the recurrence relation:

$$X_{n+1} = (a \cdot X_n + c) \mod m$$

Where:

X_n is the current pseudorandom number. X_0 - initial value (seed)

a is the multiplier. c is the increment.

m is the modulus.

$m - 1$ is the largest number.

Algorithm

Data: Seed X_0, Multiplier a, Increment c, Modulus m

Result: Random numbers

while *true* **do**

  $X_{n+1} = (a \cdot X_n + c) \mod m$;

  Output X_{n+1} as the random number;

end

Limitation of LCG

• A poor choice for the constants can lead to very poor sequences. Finite period, then sequence repeats itself.

Example:

$$a = c = I_0 = 7 \text{ and } m = 10, \qquad \text{Sequence} : 7, 6, 9, 0, 7, 6, 9, 0, ...$$

• Overflow issue - multiplication of large numbers will be out of bound. Resolution: Schrage's trick (Schrage, 1979)

• m should be as large as possible since the period can never be longer than m

• Typically, one usually chooses m to be near the largest integer than that can be represented. On a 32 bit machine, that is $2^{31} \approx 2 \times 10^9$

• Good choice (a and m are coprime): $c = 0, m = 2^{31} - 1, a = 16807$

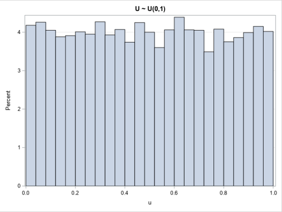
Using in-built FORTRAN function

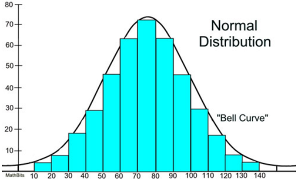
```
program testrand
  implicit none
  integer, parameter :: seed = 12345
  integer :: i, N, M
  real(8), allocatable :: random_array_1d(:), random_array_2d(:,:)
  call srand(seed)
  N = 100
  M = 200
  allocate(random_array_1d(N))
  do i = 1, N
    random_array_1d(i)=rand()
  end do
  allocate(random_array_2d(N,M))
  call random_number(random_array_2d)
  deallocate(random_array_1d)
  deallocate(random_array_2d)
end program testrand
```

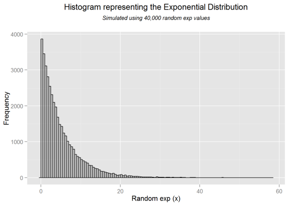
• Other uniform distributions, uniform x in $[a, b]$: $x = a + (b - a)u$, where u is a random number in $[0, 1]$

Non-uniform random number generators

How can we get a random number x distributed with a probability distribution $f(x)$ in the interval $[x_{min}, x_{max}]$ from a uniform random number u?







Methods:

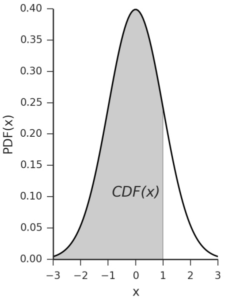
- Transformation method: need a transformation function- $T(u) = X$
- Rejection method

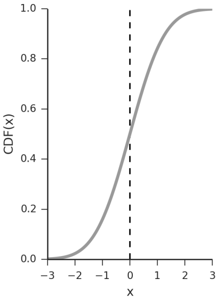
Cumulative Density Function (CDF)

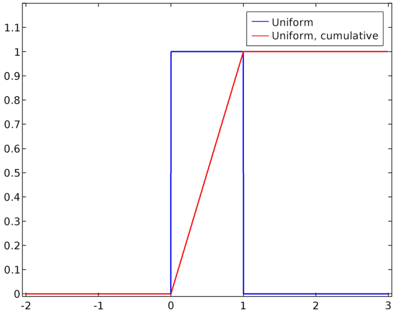
The Cumulative Density Function, denoted as $G_X(x)$, gives the probability that a continuous random variable is less than or equal to x. It is defined as the integral of the PDF from $-\infty$ to x.

$$G_X(x) = \Pr(X \leq x) = \int_{-\infty}^{x} f(t)dt$$

For example, for uniform distribution: $G_X(x) = x$







Transformation method

Desire: Random numbers from a probability density function $f(x)$ using the transformation: $T(u) = X$

Statement: Given a probability density function (PDF) $f(x)$, its corresponding cumulative density function (CDF) G_X, and a uniform variable $u \in [0, 1]$, the random numbers from the distribution $f(x)$ can be generated by $G_X^{-1}(u)$ or the transformation function is $T = F_X^{-1}$.

$$G_X(x) = \Pr(X \leq x) = \Pr(T(u) \leq x) = \Pr(u \leq T^{-1}(x)) = T^{-1}(x)$$

Algorithm

Data: Uniform random number u in the range $[0, 1]$
Data: Desired PDF and its CDF $G_X(x)$ and its inverse $G_X^{-1}(u)$
Result: Non-uniform random number X with the desired distribution

Generate u in the range $[0, 1]$;
Use $X = G^{-1}(u)$ to transform u into X with desired distribution;

Transformation method: exponential distribution

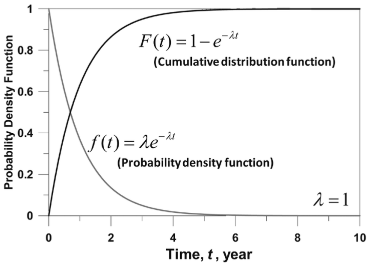
Algorithm

Data: Uniform random number u in the range $[0, 1]$
Data: Desired PDF and its CDF $G_X(x)$ and its inverse $G_X^{-1}(u)$
Result: Non-uniform random number X with the desired distribution

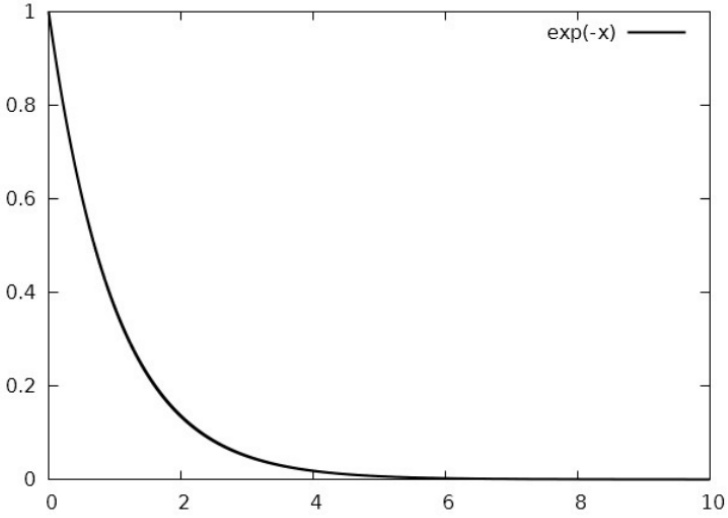
Generate u in the range $[0, 1]$;
Use $X = G^{-1}(u)$ to transform u into X with desired distribution;

- **PDF:** $f(x) = \lambda e^{-\lambda x}$
- **CDF:** $G_X(x) = \int_0^x f(x')dx' = 1 - e^{-\lambda x}$
- **Inverse CDF:**

$$X = G^{-1}(u) = -\frac{1}{\lambda}\ln(1-u)$$

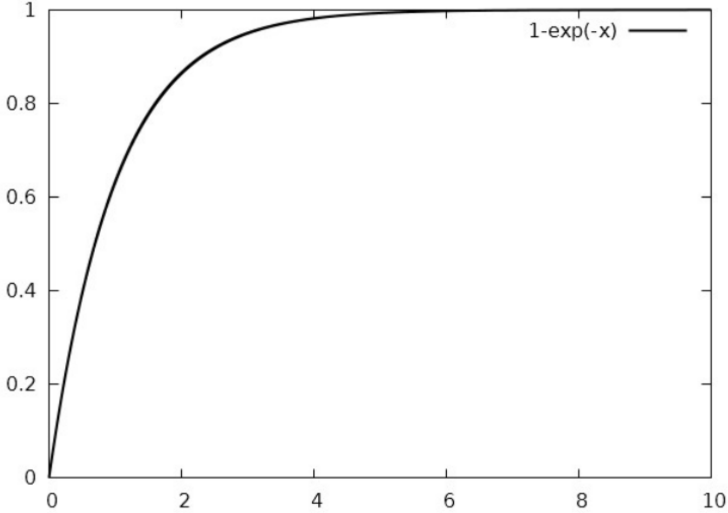


Understanding the transformation method



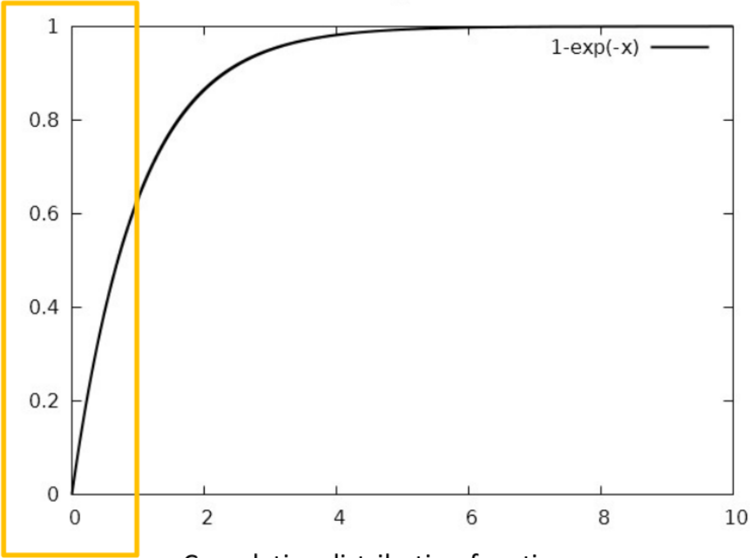
Probability distribution function

Understanding the transformation method



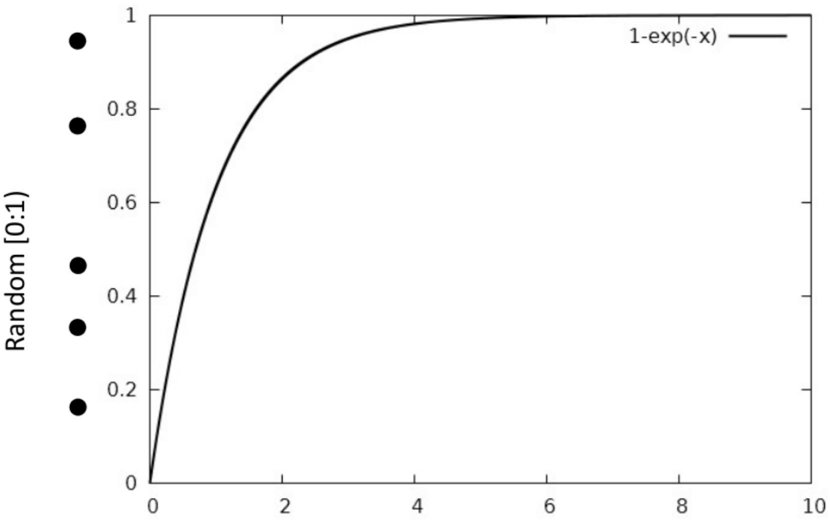
Cumulative distribution function

Understanding the transformation method

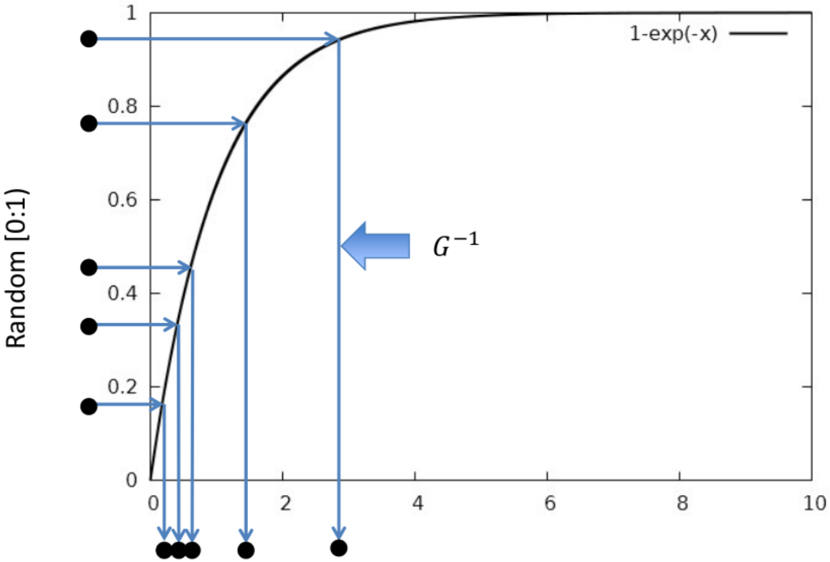


Cumulative distribution function

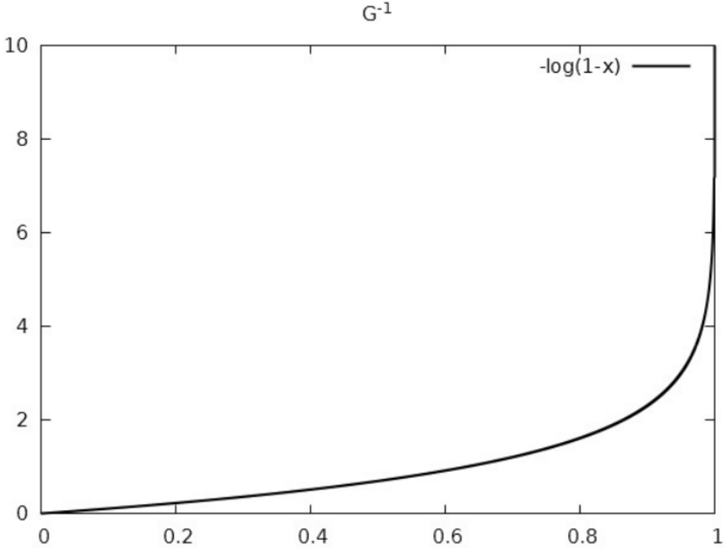
Understanding the transformation method



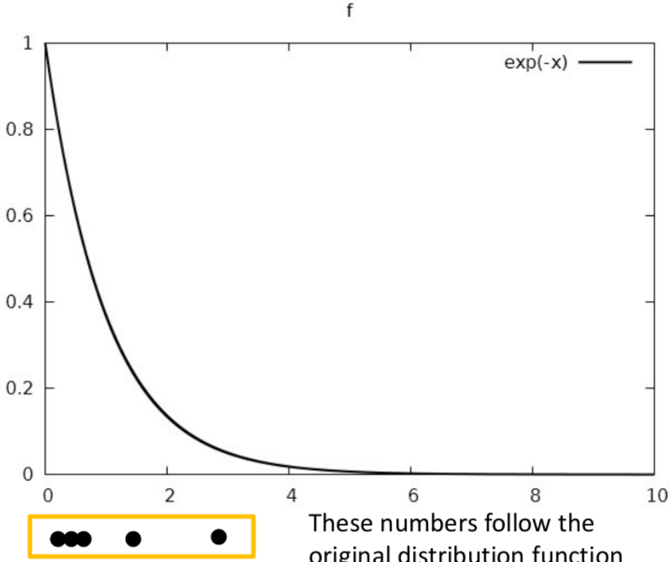
Understanding the transformation method



Understanding the transformation method



Understanding the transformation method



These numbers follow the original distribution function

Gaussian distribution: Box-Muller Method

For Gaussian PDF, it is impossible to invert $u = T^{-1}(x)$ in $1d$ but possible in $2d$. $f(x,y) = \frac{1}{2\pi} \exp\left(-\frac{x^2+y^2}{2}\right)$ and use polar coordinates.

Step 1: Generate Uniform Random Numbers

Generate two uniform random numbers U_1 and U_2 in the range $[0,1]$.

Step 2: Transform Uniforms into Normals

Use the inverse transformation to convert the uniform random numbers into two independent standard normal random numbers Z_1 and Z_2:

$$Z_1 = \sqrt{-2\ln(U_1)}\cos(2\pi U_2)$$

$$Z_2 = \sqrt{-2\ln(U_1)}\sin(2\pi U_2)$$

The generated Z_1 and Z_2 are independent standard normal random numbers $(N(0,1))$.

Assignment-12 & 13

1) Write subroutines for the middle square and the linear congruent method. Use these subroutine to generate N random numbers, use allocatable array to store them. Finally save them in a file. Use "Good choice" of a, m, c.

2) Generate 1000 couples (x, y) of random numbers using Linear Congruential Method (LCM). Plot the generated data on a 2D graph. (use the couples as x, y coordinates of the points)

4) Repeat **(2)** but with the inbuilt FORTRAN function.

3) Use inbuilt function generates 100000 points in the interval $[0, 1]$ from the distribution $f(x) = 3x^2$ using the transformation method.

4) Make a subroutine implementing the Box-Muller method. Generate 10000 random numbers from this distribution and plot the histogram.

Submit your code as ***surname_assignment_12.f90*** to ***dbhakuni@ictp.it*** before the next lesson. Use "gnuplot script.p" to plot histogram.