Scientific Research Publishing

Journal of Applied Mathematics and Physics, 2019, 7, 2712-2722
https://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

An Algorithm for Generating Random Numbers with Normal Distribution

Pirooz Mohazzabi, Michael J. Connolly

Department of Mathematics and Physics, University of Wisconsin-Parkside, Kenosha, WI, USA
Email: mohazzab@uwp.edu

How to cite this paper: Mohazzabi, P. and Connolly, M.J. (2019) An Algorithm for Generating Random Numbers with Normal Distribution. *Journal of Applied Mathematics and Physics*, **7**, 2712-2722.
https://doi.org/10.4236/jamp.2019.711185

Received: October 6, 2019
Accepted: November 5, 2019
Published: November 8, 2019

Abstract

A new algorithm is suggested based on the central limit theorem for generating pseudo-random numbers with a specified normal or Gaussian probability density function. The suggested algorithm is very simple but highly accurate, with an efficiency that falls between those of the Box-Muller and von Neumann rejection methods.

Keywords

Random Numbers, Central Limit Theorem, Normal Distribution, Gaussian Distribution

1. Introduction

Random numbers are extremely important in all areas of computational science. As a result, any programming compiler comes with some sort of pseudo-random number generator that normally generates random numbers that are uniformly distributed in the interval $[0,1)$. However, in many situations, one requires random numbers that have nonuniform distribution.

Let us suppose that we want to generate random numbers that are distributed according to the probability density function $p(x)$. This can be systematically achieved by first finding the cumulative probability distribution function $P(x)$ according to [1] [2],

$$P(x)=\int_{-\infty}^{x} p(u)\,\mathrm{d}u \tag{1}$$

and then solving the equation

$$P(x)=r \tag{2}$$

for x in terms of r, where r is a uniform random number in the unit interval $0\le r<1$. The random numbers x that are generated by this so-called *inverse* *transform* method are distributed according to the probability density function $p(x)$.

For example, consider the exponential probability density function,

$$p(x) = \begin{cases} \frac{1}{\lambda} e^{-x/\lambda} & \text{if } x \ge 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{3}$$

where λ is a constant. We have

$$P(x) = \frac{1}{\lambda} \int_0^x e^{-x/\lambda} dx = 1 - e^{-x/\lambda} \tag{4}$$

Then solving

$$1 - e^{-x/\lambda} = r \tag{5}$$

we get

$$x = -\lambda \ln(1 - r) \tag{6}$$

Thus, if *r* is a uniformly distributed random number in the unit interval $0 \le r < 1$, then *x* is a random number distributed according to the exponential density function (3).

In principle, the inverse transform method generates random numbers with any probability density function. However, the method relies on two conditions. First, the integral in Equation (1) should be evaluated analytically and, second, one should be able to solve Equation (2) explicitly for *x* in terms of *r*. But these are not always possible. For example, the Gaussian or normal probability density function,

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2} \tag{7}$$

is a function for which the integral in Equation (1) cannot be evaluated analytically. However, based on the Gaussian density function, one can generate a two-dimensional probability density, and then transform it into plane polar coordinates. After some simple algebraic manipulations, one obtains the random numbers

$$x = (2\rho)^{1/2} \cos\theta \quad \text{and} \quad y = (2\rho)^{1/2} \sin\theta \tag{8}$$

each of which is distributed according to the Gaussian probability density (7) [1]. Here ρ is a random number generated according to the exponential distribution (3) and θ is uniformly distributed in the interval $0 \le \theta < 2\pi$. This algorithm is known as the *Box-Muller* method.

A second method for generating random numbers with non-uniform probability density function if the inverse transform of $p(x)$ fails is the *von Neumann rejection* method (also called acceptance-rejection method). This algorithm uses a so called *comparison function* $c(x)$ such that $c(x) \ge p(x)$ over the entire interval of the definition of $p(x)$ [3] [4] [5]. Clearly, the simplest comparison function in most cases is a constant as shown in **Figure 1**. Then, a pair of random numbers (x, y) is generated such that $a \le x < b$ and $0 \le y < c$. This

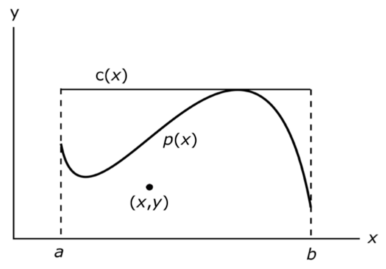

Figure 1. Probability density function $p(x)$ and a constant comparison function $c(x)$ for generating random numbers by acceptance-rejection method.

defines a point within the rectangle shown in **Figure 1**. If $y < p(x)$, this point is below the curve $p(x)$ and the random number x is accepted, otherwise, it is rejected. The process is then repeated. The random numbers x thus generated have the probability density function $p(x)$. Clearly the rejection method works for any arbitrary probability density function.

Because the normal or Gaussian probability distribution is the most encountered distribution in nature, in this work we suggest yet another algorithm for generating random numbers with normal probability density function that is based on the central limit theorem.

The organization of the contents of this article is as follows: In Section 2, we develop the theoretical foundation for the investigation. In Section 3, we discuss random numbers with normal distribution, and suggest a new algorithm for their generation followed by computer simulation results. In Section 4, we compare the suggested algorithm with other algorithms in terms of accuracy and speed. Finally, in Section 5, we present a summary of the current investigation followed by concluding remarks.

2. Theory

Consider a random number x that is distributed according to the normalized probability density function $p(x)$. The expected value or the mean value of x and x^2 are given by [6]

$$\langle x \rangle = \int_{\mathscr{D}} x p(x) \mathrm{d}x \tag{9}$$

and

$$\langle x^2 \rangle = \int_{\mathscr{D}} x^2 p(x) \mathrm{d}x \tag{10}$$

where $\mathscr{D}$ is the domain of $p(x)$. Then, the variance of the distribution is given by

$$\sigma_x^2 = \langle x^2 \rangle - \langle x \rangle^2 \tag{11}$$

Now let us pick n of these random numbers and find their average

$$y_1 = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{12}$$

Then, repeat this process a large number of times with the same number n to generate $y_1, y_2, y_3, \cdots$. The question is, are the new random numbers y as random as the numbers x? In other words, is the probability density function for y the same as $p(x)$? The answer is no. The new random numbers y are not as random as the numbers x. For example, consider a random number x that is uniformly distributed in the interval $0 \le x < 1$. Figure 2(a) shows the graph of 1000 of these random numbers. We then generate 1000 pairs of these numbers and plot the average of them, which is shown in Figure 2(b). As can be seen from these figures, although both systems have a mean of 0.5, the random numbers generated by the average of pairs of the uniformly distributed numbers are generally closer to their mean than the former ones, and hence they have a different probability distribution function.

Let $x, y, z, \cdots$ be several uncorrelated random variables with corresponding probability density functions $f(x), g(y), h(z), \cdots$. Let the variances of these random variables be $\sigma_x^2, \sigma_y^2, \sigma_z^2, \cdots$, respectively. Then according to the Bienaymé formula [7],

$$\sigma_{x+y+z+\cdots}^2 = \sigma_x^2 + \sigma_y^2 + \sigma_z^2 + \cdots \tag{13}$$

where $\sigma_{x+y+z+\cdots}^2$ is the variance of the random numbers generated by the sum of the random variables $x, y, z, \cdots$. From this equation, we find the variance of the average $(x+y+z+\cdots)/n$,

$$\sigma_{(x+y+z+\cdots)/n}^2 = \sigma_{x/n}^2 + \sigma_{y/n}^2 + \sigma_{z/n}^2 + \cdots \tag{14}$$

where n is the number of random variables to be averaged. But according to Equation (11), we have

$$\sigma_{x/n}^2 = \left\langle \left(\frac{x}{n}\right)^2 \right\rangle - \left\langle \frac{x}{n} \right\rangle^2 = \frac{1}{n^2}\left(\langle x^2 \rangle - \langle x \rangle^2\right) = \frac{\sigma_x^2}{n^2} \tag{15}$$

with similar results for $\sigma_{y/n}^2, \sigma_{z/n}^2, \cdots$. Therefore, Equation (14) reduces to

$$\sigma_{(x+y+z+\cdots)/n}^2 = \frac{1}{n^2}\left(\sigma_x^2 + \sigma_y^2 + \sigma_z^2 + \cdots\right) \tag{16}$$

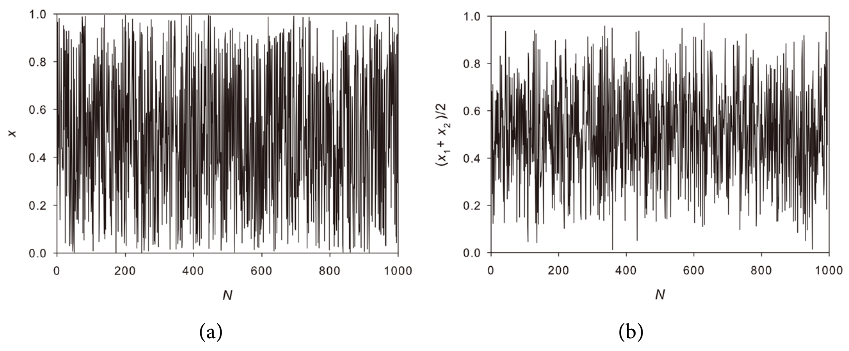

Figure 2. (a) Uniformly distributed random numbers in the interval $[0,1)$. (b) Average of two uniformly distributed random numbers in the same interval.

Finally, if all the random variables have the same distribution, we get

$$\sigma_x^2+\sigma_y^2+\sigma_z^2+\cdots=\sigma^2+\sigma^2+\sigma^2+\cdots=n\sigma^2 \tag{17}$$

And, if for the sake of simplicity we denote $\sigma^2_{(x+y+z+\cdots)/n}$ by σ_n^2, Equation (14) reduces to

$$\sigma_n=\frac{\sigma}{\sqrt{n}} \tag{18}$$

Furthermore, it can also be shown that the mean of these random numbers is the same as the mean of the original random numbers, *i.e.*,

$$\langle x_n\rangle=\langle x\rangle \tag{19}$$

As an example, the normalized probability density function for a random number that is uniformly distributed over the interval $[0,1)$ is $p(x)=1$. Then,

$$\langle x\rangle=\frac{1}{2} \tag{20}$$

and

$$\langle x^2\rangle=\frac{1}{3} \tag{21}$$

Therefore,

$$\sigma=\sqrt{\frac{1}{3}-\left(\frac{1}{2}\right)^2}=\frac{1}{2\sqrt{3}} \tag{22}$$

Therefore,

$$\sigma_2=\frac{\sigma}{\sqrt{2}}=\frac{1}{2\sqrt{6}} \tag{23}$$

This explains why the random numbers in **Figure 2(b)** are less random than those in **Figure 2(a)**. Likewise, if we average n of the uniformly distributed random numbers in the interval $[0,1)$ to obtain new random numbers, we obtain

$$\langle x_n\rangle=\frac{1}{2}\quad\text{and}\quad\sigma_n=\frac{1}{2\sqrt{3n}} \tag{24}$$

According to the central limit theorem, the random numbers thus obtained tend toward normal or Gaussian distribution as $n\to\infty$ [8] [9] [10] [11]. In fact, according to this theorem, the shape of the initial distribution of the random numbers is immaterial and the distribution of the mean approximates a normal distribution if the sample size is sufficiently large, with a mean and standard deviation given by Equations (18) and (19) [12]. It is also worth noting that as $n\to\infty$, $\sigma_n\to 0$, the resulting normal distribution approaches the Dirac delta function [13] [14] [15].

3. Random Numbers with Specific Normal Distribution

Most common computer programming languages come with some sort of built-in pseudo-random number generator that generates numbers in some interval

[16] [17]. However, most generators use the *linear congruential* or *power residue* method and generate pseudo-random numbers in the interval $[0,1)$, which is perhaps the most convenient interval in many applications [18] [19] [20].

Figures 3-5 show the results of computer simulations for progression toward a normal or Gaussian distribution for $n=2$, $n=3$, and $n=20$ starting with a uniform initial distribution in the interval $[0,1)$. In each case 10^6 random numbers are generated. The corresponding probability density function for a normal distribution with mean and standard deviation given by Equations (24) is also shown for comparison in each case. As can be seen from **Figure 5**, the agreement between the sample mean of 20 random numbers with Gaussian distribution is nearly perfect. Therefore, excellent Gaussian approximations can be achieved with fairly small samples.

In **Figure 5**, the mean and standard deviation of $p(x)$ for the sample means and the Gaussian distribution are both $\mu=1/2$ and $\sigma=1/\sqrt{240}$, respectively. This is due to the fact that the mean and the standard deviation in this case are given by Equations (24). Furthermore, because n is an integer, the standard deviation is quantized, with values given by

$$\sigma_n=\frac{1}{2\sqrt{3n}},\quad n=1,2,3,\cdots \tag{25}$$

But this is certainly a limitation because we may need to generate random numbers that have Gaussian distribution with an arbitrary mean μ and standard deviation σ that are not necessarily one of these quantities.

Let us suppose that we need to generate random numbers with Gaussian distribution given by Equation (7), with arbitrary values of the mean μ and standard deviation σ. We start by generating uniform random numbers in the interval $[0,1)$, and take the average of n of these numbers, say 20 of them, to generate new random numbers with Gaussian distribution as in **Figure 5**. The random numbers thus generated have a mean 1/2 and a standard deviation $1/2\sqrt{3n}$.

To adjust the generated random numbers to match the specified Gaussian distribution, we have to change two things; the mean μ and the standard deviation σ. The standard deviation needs to be adjusted first. To do so, we divide each random number by $1/2\sqrt{3n}$ and multiply by the required σ. Thus, we multiply each of the random numbers by a factor α, where

$$\alpha=\frac{\sigma}{1/2\sqrt{3n}}=2\sqrt{3n}\sigma \tag{26}$$

This changes the standard deviation to the required value. But this also changes the mean from 1/2 to

$$\mu=\alpha\left(\frac{1}{2}\right)=\sqrt{3n}\sigma \tag{27}$$

We now have to shift the mean to the required value μ. This is done by shifting each of the new random numbers by δx, given by

$$\delta x=-\sqrt{3n}\sigma+\mu \tag{28}$$

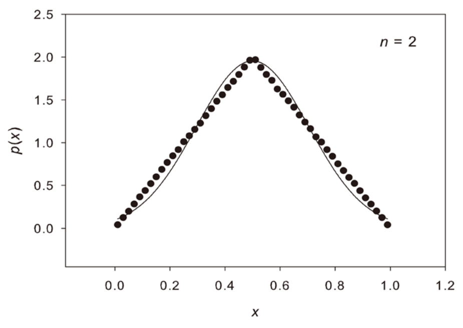

Figure 3. Probability density generated by averaging two random numbers from a uniform distribution in the interval $[0,1)$ (the bullets). The solid curve is the Gaussian function with the same mean and standard deviation calculated from Equation (24).

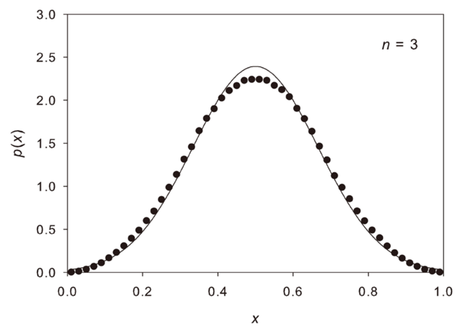

Figure 4. Same legends as in **Figure 3** but with averaging three random numbers.

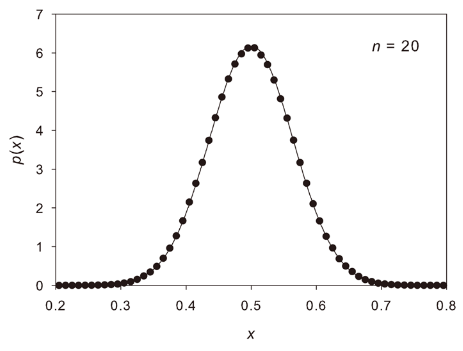

Figure 5. Same legends as in **Figure 3** but with averaging twenty random numbers.

The random numbers thus generated have the required normal or Gaussian distribution.

Combining the above transformations, if x is the average of n uniformly distributed random numbers in the interval $[0,1)$, then x' obtained through the transformation

$$x' = \sqrt{3n}\sigma(2x-1)+\mu \tag{29}$$

is distributed according to the normalized Gaussian distribution

$$p(x') = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x'-\mu)^2/2\sigma^2} \tag{30}$$

As an example, suppose we want to generate random numbers having a normal probability distribution with $\mu = -2$ and $\sigma = 1.0$, *i.e.*,

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-(x+2)^2/2} \tag{31}$$

To do so, we generate random numbers x so that each is the average of 20 uniformly distributed random numbers in the interval $[0,1)$. We then transform these into new random numbers x' according to Equation (29), which reduces to

$$x' = 2\left[\sqrt{15}(2x-1)-1\right] \tag{32}$$

These calculations can be performed using any computer programming language, such as Python. The random numbers x' thus generated have the required probability distribution, as shown in Figure 6.

4. Comparison with Other Algorithms

As stated earlier, throughout the computer simulations 10^6 random numbers were generated. These numbers were then placed in bins, each of width $dx = 0.01$. The bin contents were then normalized and plotted in the figures described above.

In order to compare the suggested algorithm with the Box-Muller and the von Neumann rejection methods, we investigated two measures; accuracy and efficiency. For accuracy, we considered the root-mean-square deviation between the bin contents and the values of the corresponding normal distribution for each algorithm. In doing so, we compared the probability density functions in different intervals about the mean of the distribution, namely, $\mu \pm 0.5\sigma$, $\mu \pm \sigma$, $\mu \pm 2\sigma$, $\mu \pm 3\sigma$, $\mu \pm 4\sigma$, and $\mu \pm 5\sigma$. We have considered several intervals because the fluctuation of the bin contents is normally higher near the peak of the distribution. The results are shown in Table 1 for $\mu = 0$, which clearly shows the accuracy of the three algorithms is nearly identical in all cases.

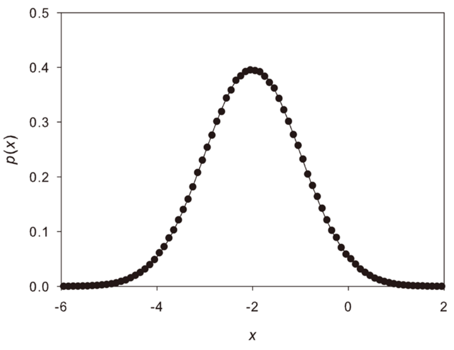

Figure 6. Probability density function for a Gaussian distribution with $\mu = -2$ and $\sigma = 1$ generated by the algorithm described in this article. The solid curve is the corresponding Gaussian function given by Equation (31).

Table 1. Root-mean-square deviation between the bin contents and the values of the corresponding normal distribution for the suggested algorithm, the Box-Muller, and the von Neumann rejection algorithms.

	Root-Mean-Square Deviation		
3-6 Test Interval	This Work	Box-Muller	von Neumann
$\mu \pm (1/2)\sigma$	6.09×10^{-1}	6.09×10^{-1}	6.09×10^{-1}
$\mu \pm \sigma$	1.58×10^{-1}	1.59×10^{-1}	1.59×10^{-1}
$\mu \pm 2\sigma$	1.30×10^{-2}	1.35×10^{-2}	1.35×10^{-2}
$\mu \pm 3\sigma$	4.20×10^{-3}	4.08×10^{-3}	4.15×10^{-3}
$\mu \pm 4\sigma$	4.20×10^{-3}	3.56×10^{-3}	3.53×10^{-3}
$\mu \pm 5\sigma$	3.26×10^{-3}	3.13×10^{-3}	3.11×10^{-3}

As a measure of efficiency, we compared the central processing unit (CPU) times of the computer during the execution of the program using each algorithm. The results consistently showed that our algorithm is slower than the Box-Muller method by about a factor of two, but faster that the von Neumann rejection method by about a factor of four.

5. Summary and Conclusions

Since random numbers are extensively used in various fields, such as computational physics and other science, we have reviewed them in this work. But specifically we have focused on machine generated pseudo-random numbers that are in most cases uniformly distributed over the interval $[0,1)$.

However, in many situations non-uniform random numbers, often with normal or Gaussian distribution, are needed. These random numbers are generated by two common methods, namely, the Box-Muller and the von Neumann rejection methods, which we have briefly reviewed in this article.

We have then suggested a third, yet very simple and efficient, algorithm for generating random numbers with a specified normal or Gaussian distribution. The suggested algorithm relies on a modified central limit theorem, and uses the average of uniformly distributed random numbers in the interval $[0,1)$. The algorithm generates random numbers with the required normal probability distribution function very accurately using a small number of uniform random numbers in the averaging process.

In comparison with other methods, the suggested algorithm is as accurate as the Box-Muller and the von Neumann rejection algorithms. In terms of the efficiency or the speed with which the random numbers are generated, however, our algorithm is slower than the Box-Muller method by about a factor of two, but faster than the von Neumann rejection method by about a factor of four.

Considering the accuracy, simplicity, and the efficiency of the suggested algorithm, it can justifiably compete with the existing methods.

At this time, the authors intend no future work based on the current study.

Acknowledgements

This work was supported by a URAP grant from the University of Wisconsin-Parkside.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Gould, H. and Tobochnik, J. (1996) An Introduction to Computer Simulation Methods. 2nd Edition, Addison-Wesley, New York, 358-361.

[2] Landau, R.H. and Páez, M.J. (1997) Computational Physics. John Wiley and Sons, New York, 104-105.

[3] Gould, H. andTobochnik, J. (1996) An Introduction to Computer Simulation Methods. 2nd Edition, Addison-Wesley, New York, 371-372.

[4] Giordano, N.J. (1997) Computational Physics. Prentice Hall, Upper Saddle River, NJ, 161-163.

[5] Allen, M.P. and Tildesley, D.J. (1987) Computer Simulation of Liquids. Clarendon Press, Oxford, Great Britain, 349-351.

[6] Grinstead, C.M. and Snell, J.L. (1997) Introduction to Probability. 2nd Revised Edition, American Mathematical Society, Providence, RI, 269-270.

[7] Wikipedia (2019) Variance. https://en.wikipedia.org/wiki/Variance

[8] Pitman, J. (1993) Probability. Springer-Verlag, New York, 196. https://doi.org/10.1007/978-1-4612-4374-8

[9] Wani, J.K. (1971) Probability and Statistical Inference. Appleton Century Crofts, Meredith Corporation, New York, 154-156.

[10] Wallace, P.R. (1984) Mathematical Analysis of Physical Problems. Dover Publications, Inc., New York, 431.

[11] Burington, R.S. and May Jr., D.C. (1970) Handbook of Probability and Statistics with Tables. 2nd Edition, McGraw-Hill, New York, 190.

[12] Witte, R.S. and Witte, J.S. (2007) Statistics. 8th Edition, John Wiley & Sons, Hoboken, NJ, 199-204.

[13] Wikipedia (2019) Dirac Delta Function. https://en.wikipedia.org/wiki/Dirac_delta_function

[14] Byron Jr., F.W. and Fuller, R.W. (1970) Mathematics of Classical and Quantum Physics. Dover Publications, New York, 447-448.

[15] Liboff, R.L. (2003) Introductory Quantum Mechanics. 4th Edition, Addison Wesley, New York, 74-75.

[16] Press, W.H., Flannery, B.P., Teukolsky, S.A. and Vetterling, W.T. (1986) Numerical Recipes. Cambridge University Press, New York.

[17] Haile, J.M. (1997) Molecular Dynamics Simulation. John Wiley & Sons, New York, 376-377.

[18] Gould, H. and Tobochnik, J. (1996) An Introduction to Computer Simulation Methods. 2nd edition, Addison-Wesley, New York, 401-403.

[19] Landau, R.H. and Páez, M.J. (1997) Computational Physics. John Wiley and Sons, New York, 84-86.

[20] Harrison, P. (2001) Computational Methods in Physics, Chemistry and Biology. John Wiley & Sons, New York, 116-117.