Random Numbers Generation and Goodness-of-Fit Testing for Literal Neutrosophic Numbers

Mohamed Bisher Zeina, Mohamad Taher Anan, Yousef Marjamak

Faculty of Science, Department of Mathematical Statistics, University of Aleppo, Aleppo, Syria
Emails: bisher.zeina@gmail.com; mtanan200988@gmail.com; yousefmarjamak20@gmail.com
* Correspondence: bisher.zeina@gmail.com; Tel.: (+963969602951)

Abstract

This paper presents an algorithm for generating random numbers that follow literal neutrosophic distributions using algebraic isomorphisms. The algorithm was applied to three distributions: literal neutrosophic uniform distribution, literal neutrosophic exponential distribution, and literal neutrosophic normal distribution. We also proposed a development of the Kolmogorov-Smirnov test to analyze goodness-of-fit for literal neutrosophic data. To evaluate the effectiveness of the proposed method, a simulation study was conducted, and the power results showed that increasing the sample size improved the test power. Our research contributes to the development of statistical methods for analyzing literal neutrosophic data, which has applications in a wide range of fields.

Keywords: AH-Isometry; Neutrosophic Field of Reals; Neutrosophic Random Variables; Random Numbers Generation; Goodness of Fit.

1. Introduction

Random numbers generation is the process of generating a sequence of numbers that appear to be random, with no predictable pattern or structure. Random numbers are a fundamental component of many applications in probability theory and statistics, as well as in various fields such as finance, engineering, computer science, and cryptography. [1]

In probability theory, random numbers play a critical role in simulation and modeling. Simulations are often used to study the behavior of complex systems or to estimate the probability of rare events. To perform such simulations, random numbers are used to represent the various sources of uncertainty in the system being modeled. For example, in a simulation of a manufacturing process, random numbers might be used to represent the variation in the size or weight of raw materials, or the variability in machine settings or operator behavior. The use of random numbers in simulation allows the modeler to study the behavior of the system under a wide range of possible scenarios, and to estimate the likelihood of various outcomes. [2-3]

Random numbers also play a critical role in statistical inference, which is the process of drawing conclusions about a population based on a sample of data. Random sampling is a common method used in statistics to select a subset of individuals or units from a larger population for analysis. By using random sampling, it is

Doi: https://doi.org/10.54216/GJMSA.040201
Received: November 23, 2022 Accepted: March 17, 2023

possible to ensure that the sample is representative of the population, and to make statistical inferences about the population based on the characteristics of the sample. In addition, random numbers are used in various statistical tests to determine whether an observed pattern or result is likely to have occurred by chance or is statistically significant. [4]

There are many methods for generating random numbers, ranging from simple coin flips and dice rolls to more sophisticated algorithms based on complex mathematical formulas or physical processes. The choice of a particular method depends on the application and the desired properties of the random numbers, such as their distribution, independence, and reproducibility. [5]

In neutrosophic sets theory many goodness of fit tests were generalized to handle uncertain interval valued data like [6-10], but until this moment there is no test can handle neutrosophic literal numbers of the form $N = a + bI\,; I^2 = I$. So, in this paper we will first represent a method for generating random numbers follow a specific literal neutrosophic probability distribution then we represent a goodness of fit approach that assure the null hypothesis of having no significant difference between the generated numbers and the generator distribution.

2. Preliminaries

Definition 2.1 [11]

Let $R(I)$ be the literal neutrosophic real numbers set, we say that $N \in R(I)$ if $N = a + bI\,; a, b \in R$ & $I^2 = I$.

Definition 2.2 [12]

Let $R(I)$ be the literal neutrosophic real numbers set, transformation T defined below is called the AH-isometry:

$$T: R(I) \to R^2 \,:\, T(a + bI) = (a, a + b) \qquad (1)$$

And its inverse is defined by:

$$T^{-1}: R^2 \to R(I) \,:\, T^{-1}(a, b) = a + (b - a)I \qquad (2)$$

Note:

Let $N_1, N_2 \in R(I)$ and T be the AH-Isometry, T is an algebraic isomorphism and it satisfies the following properties:

1. $T(x_N + y_N) = T(x_N) + T(y_N)$
2. $T(x_N \cdot y_N) = T(x_N) \cdot T(y_N)$
3. T is correspondence one-to-one.
4. T preserves distances.

Definition 2.3 [12]

Let $N_1 = a_1 + b_1 I, N_2 = a_2 + b_2 I \in R(I)$, we say that $N_1 \leq N_2$ if:

$$a_1 \leq a_2 \;\&\; a_1 + b_1 \leq a_2 + b_2$$

Definition 2.4 [13]

Let $f: R(I) \to R(I); f = f(x_N)$ where $x_N = x_1 + x_2 I \in R(I)$ then f is called a neutrosophic real function with one neutrosophic variable.

Doi: https://doi.org/10.54216/GJMSA.040201
Received: November 23, 2022 Accepted: March 17, 2023

Definition 2.5 [14-15]

Literal neutrosophic random variable X_N is defined as follows:

$$X_N: \Omega_1 \times \Omega_2(I) \to R(I)$$

Where:

$$X_N = X_1 + X_2 I$$

And:

X_1, X_2 are both real valued crisp random variables defined on Ω_1, Ω_2 respectively and taking values in R.

Definition 2.6 [7]

Literal neutrosophic probability P_N is defined as follows:

$$P_N = p_1 + p_2 I \,; p_1 \in [0,1] \,, p_1 + p_2 \in [0,1] \,, I^2 = I$$

1. **Proposed Algorithm for Random Literal Neutrosophic Numbers Generating**

Say that $P_N = p_1 + p_2 I$ is a literal neutrosophic probability that describes the literal neutrosophic cumulative probability distribution $F_N(x_N; \Theta_N)$ where $\Theta_N = \Theta_1 + \Theta_2 I$ is a vector of distribution's parameters, i.e., $P_N = F_N(x_N; \Theta_N)$, then to generate random literal neutrosophic numbers following distribution F_N we can follow the following steps:

1. Solve the neutrosophic equation $P_N = F_N(x_N; \Theta_N)$ for x_N to get an equation of the form:
$$x_N = \Phi_N(P_N; \Theta_N) \quad (3)$$
2. Take the isometry to equation (3).
3. You will have two equations of the form:
$$x_1 = \Phi(p_1; \Theta_1) \qquad (4)$$
$$x_1 + x_2 = \Phi(p_1 + p_2; \Theta_1 + \Theta_2) \qquad (5)$$
4. Generate a random number for p_1 in $[0,1]$ and for $p_1 + p_2$ in $[0,1]$ (a crisp uniformly distributed number).
5. Solve equations (4-5) with respect to x_1, x_2.
6. The random literal neutrosophic number will be:
$$x_N = x_1 + x_2 I$$
7. Repeat steps 4 to 6 until you have the sufficient set of numbers.

3.1 Random Numbers Generation of Literal Neutrosophic Uniform Distribution (LNUD)

Literal neutrosophic cumulative distribution function (LNCDF) of LNUD is defined in [7] as:

$$P_N = F(x_N) = \frac{x_N - a_N}{b_N - a_N}; a_N = a_1 + a_2 I, b_N = b_1 + b_2 I \,, x_N = x_1 + x_2 I \,; I^2 = I \,(6)$$

Solving equation (6) with respect to x_N gives:

$$x_N = a_N + (b_N - a_N) P_N \,(7)$$

Doi: https://doi.org/10.54216/GJMSA.040201
Received: November 23, 2022 Accepted: March 17, 2023

Taking isometric image of (7) results:

$$T[x_N] = (a_1 + (b_1 - a_1)p_1, a_1 + a_2 + (b_1 + b_2 - a_1 - a_2)(p_1 + p_2)) = (x_1, x_1 + x_2)$$

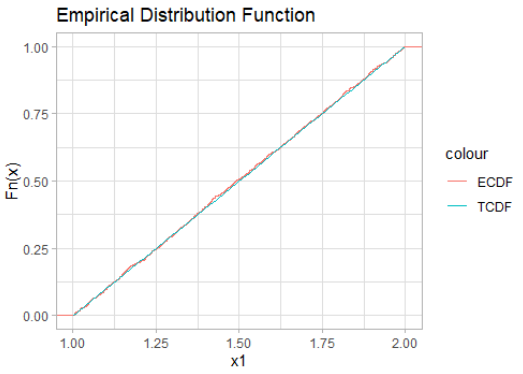
Which means depending on (2) and the properties of AH-Isometry that:

$$x_N = a_1 + (b_1 - a_1)p_1 + [a_1 + a_2 + (b_1 + b_2 - a_1 - a_2)(p_1 + p_2) - a_1 + (b_1 - a_1)p_1]I \quad (8)$$

Table 1: Random Literal Neutrosophic Numbers following LNUD with $a_N = 1 - I, b_N = 2 + 2I$

x_1	x_2	x_N
1.478	−0.9226	$1.478 - 0.9226\,I$
1.758	−0.8263	$1.758 - 0.8263\,I$
1.216	0.6474	$1.216 + 0.6474\,I$
1.318	−0.2543	$1.318 - 0.2543\,I$
1.232	2.2	$1.232 + 2.2\,I$
1.143	−0.9595	$1.143 - 0.9595\,I$
1.415	0.3543	$1.415 + 0.3543\,I$
1.414	1.782	$1.414 + 1.782\,I$

A sample of $n = 1000$ random numbers were generated using R code presented in the appendix and figure (1-2) are the corresponding ECDF compared to theoretical CDF of $x_1, x_1 + x_2$ respectively:



: ECDF and TCDF of X_1 of LNUD

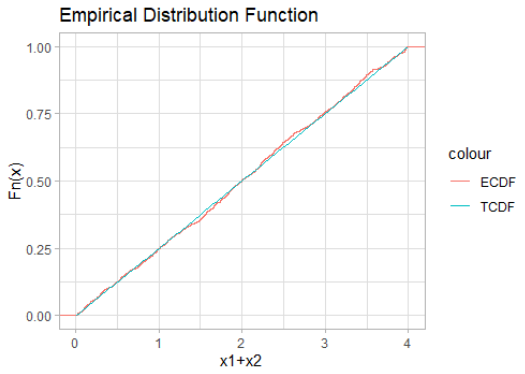


Figure 1: ECDF and TCDF of $X_1 + X_2$ of LNUD

3.2 Random Numbers Generation of Literal Neutrosophic Exponential Distribution (LNED)

Literal neutrosophic cumulative distribution function (LNCDF) of LNED is defined in [7] as:

$$P_N = F(x_N) = 1 - e^{-\lambda_N x_N}; \lambda_N = \lambda_1 + \lambda_2 I\,, x_N = x_1 + x_2 I\,; I^2 = I \quad (9)$$

Solving equation (9) with respect to x_N gives:

$$x_N = -\frac{\ln(1-P_N)}{\lambda_N} \quad (10)$$

Doi: https://doi.org/10.54216/GJMSA.040201
Received: November 23, 2022 Accepted: March 17, 2023

Taking isometric image of (10) results:

$$T[x_N] = \left(-\frac{\ln(1-p_1)}{\lambda_1}, -\frac{\ln(1-p_1-p_2)}{\lambda_1+\lambda_2}\right) = (x_1, x_1+x_2)$$

Which means depending on (2) and the properties of AH-Isometry that:

$$x_N = -\frac{\ln(1-p_1)}{\lambda_1} + \left[\frac{\ln(1-p_1)}{\lambda_1} - \frac{\ln(1-p_1-p_2)}{\lambda_1+\lambda_2}\right] I \qquad (11)$$

Table 2: Random LNNs following LNED with $\lambda_N = 2.5 - 1.5I$

x_1	x_2	x_N
0.3374	0.6674	$0.3374 + 0.6674\,I$
0.2306	0.2496	$0.2306 + 0.2496\,I$
0.5316	−0.2506	$0.5316 - 0.2506\,I$
0.01263	0.3645	$0.01263 + 0.3645\,I$
0.02248	0.1658	$0.02248 + 0.1658\,I$
0.1266	0.7232	$0.1266 + 0.7232\,I$
0.1257	1.438	$0.1257 + 1.438\,I$
0.05811	0.4207	$0.05811 + 0.4207\,I$

A sample of $n = 1000$ random numbers were generated using R code presented in the appendix and figure (3-4) are the corresponding ECDF compared to theoretical CDF of $x_1, x_1 + x_2$ respectively:

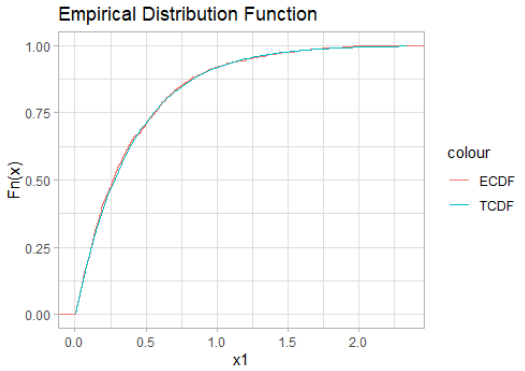


Figure 3: ECDF and TCDF of X_1 of LNED

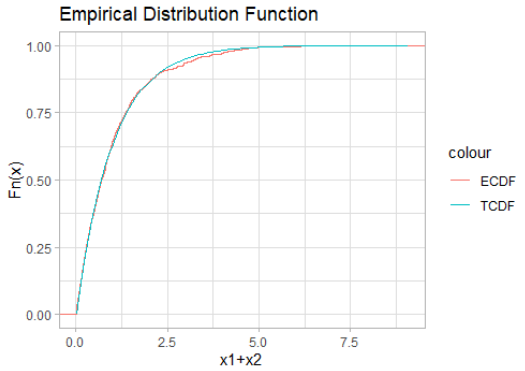


Figure 4: ECDF and TCDF of $X_1 + X_2$ of LNED

3.3 Random Numbers Generation of Literal Neutrosophic Normal Distribution (LNND)

Literal neutrosophic cumulative distribution function (LNCDF) of LNND is defined in [7] as:

$$P_N = \frac{1}{\sigma_N\sqrt{2\pi}} \int_{-\infty}^{x_N} e^{-\frac{1}{2}\left(\frac{x_N-\mu_N}{\sigma_N}\right)^2} ; \sigma_N = \sigma_1 + \sigma_2 I, \mu_N = \mu_1 + \mu_2 I\,, x_N = x_1 + x_2 I\,; I^2 = I$$

Doi: https://doi.org/10.54216/GJMSA.040201
Received: November 23, 2022 Accepted: March 17, 2023

Or simply write:

$$P_N = F_N(x_N; \mu_N, \sigma_N) \qquad (12)$$

Solving equation (12) with respect to x_N gives:

$$x_N = \psi(P_N, \mu_N, \sigma_N) \ (13)$$

Taking isometric image of (13) results:

$$T[x_N] = (\psi(p_1, \mu_1, \sigma_1), \psi(p_1 + p_2, \mu_1 + \mu_2, \sigma_1 + \sigma_2)) = (x_1, x_1 + x_2)$$

Which means depending on (2) and the properties of AH-Isometry that:

$$x_N = \psi(p_1, \mu_1, \sigma_1) + [\psi(p_1 + p_2, \mu_1 + \mu_2, \sigma_1 + \sigma_2) - \psi(p_1, \mu_1, \sigma_1)]I \qquad (14)$$

Table 3: Random LNNs following LNND with $\mu_N = 2I, \sigma_N = 1 + I$

x_1	x_2	x_N
$\mathbf{-0.5277}$	-2.596	$-0.5277 - 2.596\,I$
$\mathbf{-1.441}$	3.937	$-1.441 + 3.937\,I$
$\mathbf{-1.957}$	3.146	$-1.957 + 3.146\,I$
$\mathbf{0.02866}$	0.4835	$0.02866 + 0.4835\,I$
$\mathbf{1.538}$	-3.55	$1.538 - 3.55\,I$
$\mathbf{1.635}$	-0.04629	$1.635 - 0.04629\,I$
$\mathbf{-0.5628}$	-0.7353	$-0.5628 - 0.7353\,I$
$\mathbf{-0.697}$	2.666	$-0.697 + 2.666\,I$

A sample of $n = 1000$ random numbers were generated using R code presented in the appendix and figure (5-6) are the corresponding ECDF compared to theoretical CDF of $x_1, x_1 + x_2$ respectively:

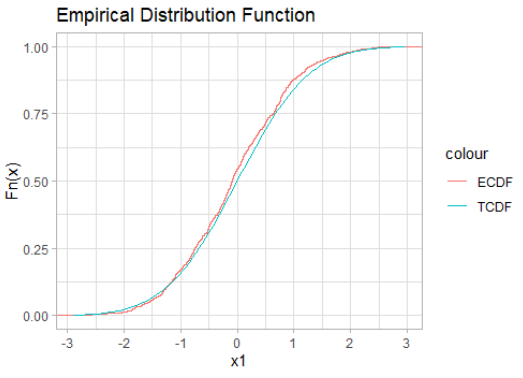


Figure 5: ECDF and TCDF of X_1 of LNED

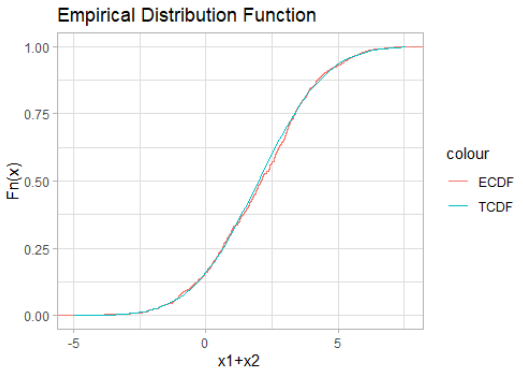


Figure 6: ECDF and TCDF of $X_1 + X_2$ of LNED

Doi: https://doi.org/10.54216/GJMSA.040201
Received: November 23, 2022 Accepted: March 17, 2023

2. Literal Neutrosophic Kolmogorov-Smirnov Goodness of Fit Test

The Kolmogorov-Smirnov (KS) test is a statistical test used to determine whether a sample of data comes from a specified distribution, such as a normal distribution or an exponential distribution. The KS test can be used as a goodness-of-fit test to test the hypothesis that a sample of data comes from a specified distribution.

In this section we are going to develop the classical KS test to handle literal neutrosophic numbers proposing the following algorithm:

1. Determine the literal neutrosophic distribution to which you want to compare your data.
2. Define the null hypothesis and the alternative hypothesis. The null hypothesis is that the sample of data comes from the specified distribution, while the alternative hypothesis is that the sample of data does not come from the specified distribution.
3. Calculate the LNCDF of the specified distribution.
4. Calculate the literal neutrosophic empirical distribution function (LNEDF) of your sample data, here you will use neutrosophic ordering relation in definition 2.3.
5. Calculate the test statistic, which is the maximum absolute difference between the LNCDF and LNEDF.

$$D_N = \max |F_N - S_N| \qquad (15)$$

Where:

F_N is the LNCDF of the specified distribution.

S_N is the LNEDF of the sample data.

6. Calculate neutrosophic p-value (pv_N) of the test using the formula:

$$pv_N = 1 - \sum_{i=1}^{\infty} (-1)^{i-1} e^{-2i^2 D_N^2} \qquad (16)$$

7. Compare the calculate pv_N with desired significance level $\alpha_N = \alpha_1 + \alpha_2 I$ (can be crisp significance level when $\alpha_2 = 0$). If pv_N is less than α_N, reject the null hypothesis; otherwise, fail to reject the null hypothesis.

3. Simulation Study

1. Based on equation (8), a simulation study was performed $N = 10000$ times to generate different samples of sizes $n = 10,30,50,100$ each is following literal neutrosophic uniform distribution with parameters $a_N = 1 - I, b_N = 2 + 2I$ and tested whether these random samples are drawn from literal neutrosophic uniform distribution with parameters $a_N = 1.5 - I, b_N = 2.5 + 2I$ or not at significance level $\alpha = 0.05$. The following table shows simulation results:

Table 4: Test power of Kolmogorov Smirnov for literal neutrosophic uniform distribution

n	Test Power
10	0.1568
30	0.4040
50	0.6176
100	0.9509

Doi: https://doi.org/10.54216/GJMSA.040201
Received: November 23, 2022 Accepted: March 17, 2023

We see that test power increases when sample size increases. Proposed Kolmogorov-Smirnov test works better with samples of size larger than 50 for literal neutrosophic uniform distribution detection.

2. Based on equation (11), a simulation study was performed $N = 10000$ times to generate different samples of sizes $n = 10,30,50,100$ each is following literal neutrosophic exponential distribution with parameter $\lambda_N = 2.5 - 1.5I$ and tested whether these random samples are drawn from literal neutrosophic exponential distribution with parameter $\lambda_N = 4 - I$ or not at significance level $\alpha = 0.05$. The following table shows simulation results:

Table 5: Test power of Kolmogorov Smirnov for literal neutrosophic exponential distribution

n	Test Power
10	0.1485
30	0.5100
50	0.7318
100	0.9504

We see that test power increases when sample size increases. Proposed Kolmogorov-Smirnov test works better with samples of size larger than 30 for literal neutrosophic exponential distribution detection.

3. Based on equation (14), a simulation study was performed $N = 10000$ times to generate different samples of sizes $n = 10,30,50,100$ each is following literal neutrosophic normal distribution with parameters $\mu_N = 2I, \sigma_N = 1 + I$ and tested whether these random samples are drawn from literal neutrosophic normal distribution with parameters $\mu_N = I, \sigma_N = 2 + I$ or not at significance level $\alpha = 0.05$. The following table shows simulation results:

Table 6: Test power of Kolmogorov Smirnov for literal neutrosophic normal distribution

n	Test Power
10	0.0042
30	0.2136
50	0.7463
100	0.9988

We see that test power increases when sample size increases. Proposed Kolmogorov-Smirnov test works better with samples of size larger than 30 for literal neutrosophic normal distribution detection.

4. Conclusions and future research directions

In this study, we developed a new random number generator algorithm and applied it to find analytical formulas for three different literal neutrosophic probability distributions: uniform, exponential, and normal. We also developed the Kolmogorov-Smirnov test for goodness of fit to evaluate the performance of literal neutrosophic data. Our results show that the power of the test increases as the sample size grows, indicating that the test is effective in detecting differences between the generated data and the target distributions.

Our study has several strengths, including the use of literal neutrosophic distributions and the potential applications of the developed algorithm in various fields. However, further research is needed to explore the properties of these distributions and to compare the results of our method with those from other methods.

In conclusion, our study provides a new tool for generating random numbers from literal neutrosophic probability distributions and for evaluating the goodness of fit of the generated data. Future research in this area could further improve the performance and applicability of these methods.

Funding: "This research received no external funding"

Doi: https://doi.org/10.54216/GJMSA.040201
Received: November 23, 2022 Accepted: March 17, 2023

Acknowledgments: Authors are very thankful to Editors-In-Chief for their assistance and management, authors also thank reviewers for their valuable edits and suggestions.

Conflicts of Interest: "The authors declare no conflict of interest."

References

[1] S. M. Ross, Introduction to probability models, 10 ed., United States: Academic Press, 2019.

[2] L. Devroye, Non-Uniform Random Variate Generation, New York: Springer, 1986.

[3] G. S. Fishman, Monte Carlo: concepts, algorithms, and applications, New York: Springer, 1996.

[4] D. E. Knuth, The Art of Computer Programming, vol. 2, United States: Addison-Wesley, 1998.

[5] P. L'Ecuyer and R. Simard, "TestU01: A C library for empirical testing of random number generators," *ACM Transactions on Mathematical Software,* vol. 33, no. 4, pp. 1-40, 2007.

[6] M. Aslam, "Introducing Kolmogorov–Smirnov Tests under Uncertainty: An Application to Radioactive Data," *ACS Omega,* vol. 5, pp. 914-917, 2020.

[7] M. Aslam, "A new goodness of fit test in the presence of uncertain parameters," *Complex & Intelligent Systems,* vol. 7, pp. 359-365, 2021.

[8] M. Ahsan-ul-Haq, "A new Cramèr–von Mises Goodness-of-fit test under Uncertainty," *Neutrosophic Sets and Systems,* vol. 49, pp. 262-268, 2022.

[9] M. Aslam, "Chi-square test under indeterminacy: an application using pulse count data," *BMC Med Res Methodology,* vol. 21, no. 1, 2021.

[10] M. Jdid, R. Alhabib and A. A. Salama, "Fundamentals of Neutrosophical Simulation for Generating Fundamentals of Neutrosophical Simulation for Generating Random Numbers Associated with Uniform Probability Random Numbers Associated with Uniform Probability Distribution," *Neutrosophic Sets and Systems,* vol. 49, pp. 92-102, 2022.

[11] M. B. Zeina and M. Abobala, "A Novel Approach of Neutrosophic Continuous Probability Distributions using AH-Isometry with Applications in Medicine," in *Cognitive Intelligence with Neutrosophic Statistics in Bioinformatics*, Elsevier, 2023.

[12] M. B. Zeina , O. Zeitouny , F. Masri , F. Kadoura and S. Broumi, "Operations on Single-Valued Trapezoidal Neutrosophic Numbers using (α,β,γ)-Cuts "Maple Package"," *International Journal of Neutrosophic Science,* vol. 15, no. 2, pp. 113-122, 2021.

[13] M. Abobala and M. B. Zeina, "A Study of Neutrosophic Real Analysis by Using the One-Dimensional Geometric AH-Isometry," *Galoitica: Journal of Mathematical Structures and Applications,* vol. 3, no. 1, pp. 18-24, 2023.

[14] M. B. Zeina and A. Hatip, "Neutrosophic Random Variables," *Neutrosophic Sets and Systems,* vol. 39, pp. 44-52, 2021.

[15] A. Astambli, M. B. Zeina and Y. Karmouta, "On Some Estimation Methods of Neutrosophic Continuous Probability Distributions Using One-Dimensional AH-Isometry," *Neutrosophic Sets and Systems,* vol. 53, pp. 641-652, 2023.

Doi: https://doi.org/10.54216/GJMSA.040201
Received: November 23, 2022 Accepted: March 17, 2023

Appendix – R Code

 # Generating Random Numbers Code

```
rm(list=ls())
library(ggplot2)
set.seed(123)
options("scipen"=100, "digits"=4)
n<-1000
#######################################################
#Uniform
x1<-runif(n,min=1,max=2)
x12<-runif(n,min=0,max=4)
for (i in 1:n)
{
if(x12[i]-x1[i]>=0)
cat(x1[i],"+",x12[i]-x1[i],"I\n")
else
cat(x1[i],"-",abs(x12[i]-x1[i]),"I\n")
}
for (i in 1:n)
{
cat(x1[i],"\n")
}
for (i in 1:n)
{
cat(x12[i]-x1[i],"\n")
}
```


Doi: https://doi.org/10.54216/GJMSA.040201
Received: November 23, 2022 Accepted: March 17, 2023

```
ggplot(data.frame(x1), aes(x1)) +
  stat_ecdf(geom = "step", aes(colour = "ECDF")) +
  stat_function(
    fun = ~ (.x-1)
    ,aes(colour = "TCDF")
  ) +
  theme_light() +
  labs(
    x = "x1",
    y = "Fn(x)",
    title = "Empirical Distribution Function"
  )
ggplot(data.frame(x12), aes(x12)) +
  stat_ecdf(geom = "step", aes(colour = "ECDF")) +
  stat_function(
    fun = ~ (.x/4)
    ,aes(colour = "TCDF")
  ) +
  theme_light() +
  labs(
    x = "x1+x2",
    y = "Fn(x)",
    title = "Empirical Distribution Function"
  )
#######################################################
#Exponential
x1<-rexp(n,rate=2.5)
x12<-rexp(n,rate=1)
```


Doi: https://doi.org/10.54216/GJMSA.040201
Received: November 23, 2022 Accepted: March 17, 2023

```
for (i in 1:n)
{
if(x12[i]-x1[i]>=0)
cat(x1[i],"+",x12[i]-x1[i],"I\n")
else
cat(x1[i],"-",abs(x12[i]-x1[i]),"I\n")
}
for (i in 1:n)
{
cat(x1[i],"\n")
}
for (i in 1:n)
{
cat(x12[i]-x1[i],"\n")
}
ggplot(data.frame(x1), aes(x1)) +
  stat_ecdf(geom = "step", aes(colour = "ECDF")) +
  stat_function(
    fun = ~ (1-exp(-2.5*.x))
    ,aes(colour = "TCDF")
  ) +
  theme_light() +
  labs(
    x = "x1",
    y = "Fn(x)",
    title = "Empirical Distribution Function"
  )
ggplot(data.frame(x12), aes(x12)) +
```


Doi: https://doi.org/10.54216/GJMSA.040201
Received: November 23, 2022 Accepted: March 17, 2023

```
  stat_ecdf(geom = "step", aes(colour = "ECDF")) +
  stat_function(
    fun = ~ (1-exp(-.x))
    ,aes(colour = "TCDF")
  ) +
  theme_light() +
  labs(
    x = "x1+x2",
    y = "Fn(x)",
    title = "Empirical Distribution Function"
  )
#####################################################################
n<-1000
#Normal
x1<-rnorm(n,0,1)
x12<-rnorm(n,2,2)
for (i in 1:n)
{
if(x12[i]-x1[i]>=0)
cat(x1[i],"+",x12[i]-x1[i],"I\n")
else
cat(x1[i],"-",abs(x12[i]-x1[i]),"I\n")
}
for (i in 1:n)
{
cat(x1[i],"\n")
}
for (i in 1:n)
```


Doi: https://doi.org/10.54216/GJMSA.040201
Received: November 23, 2022 Accepted: March 17, 2023

```
{
cat(x12[i]-x1[i],"\n")
}
ggplot(data.frame(x1), aes(x1)) +
  stat_ecdf(geom = "step", aes(colour = "ECDF")) +
  stat_function(
    fun = ~ (pnorm(.x,mean=0,sd=1))
    ,aes(colour = "TCDF")
  ) +
  theme_light() +
  labs(
    x = "x1",
    y = "Fn(x)",
    title = "Empirical Distribution Function"
  )
ggplot(data.frame(x12), aes(x12)) +
  stat_ecdf(geom = "step", aes(colour = "ECDF")) +
  stat_function(
    fun = ~ (pnorm(.x,mean=2,sd=2))
    ,aes(colour = "TCDF")
  ) +
  theme_light() +
  labs(
    x = "x1+x2",
    y = "Fn(x)",
    title = "Empirical Distribution Function"
  )
```


Doi: https://doi.org/10.54216/GJMSA.040201
Received: November 23, 2022 Accepted: March 17, 2023

 # Simulation Code

```
rm(list=ls())
library(ggplot2)
set.seed(123)
options("scipen"=100, "digits"=4)
N<-10000
nSet<-c(10,30,50,100)
#######################################################
#Uniform
for (n in nSet)
{
p<-0
for (i in 1:N)
{
x1<-runif(n,min=1,max=2)
x12<-runif(n,min=0,max=4)
p1<-ks.test(x1,"punif",1.5,2.5)$p
p2<-ks.test(x12,"punif",0.5,4.5)$p
if ((p1<0.05) & (p1+p2<0.05))
p<-p+1
}
cat("power of n = ",n," is: ",p/N,"\n")
}
#######################################################
#Exponential
for (n in nSet)
{
```


Doi: https://doi.org/10.54216/GJMSA.040201
Received: November 23, 2022 Accepted: March 17, 2023

```
p<-0
for (i in 1:N)
{
x1<-rexp(n,rate=2.5)
x12<-rexp(n,rate=1)
p1<-ks.test(x1,"pexp",4)$p
p2<-ks.test(x12,"pexp",3)$p
if ((p1<0.05) & (p1+p2<0.05))
p<-p+1
}
cat("power of n = ",n," is: ",p/N,"\n")
}
#################################################################
#Normal
for (n in nSet)
{
p<-0
for (i in 1:N)
{
x1<-rnorm(n,0,1)
x12<-rnorm(n,2,2)
p1<-ks.test(x1,"pnorm",0,2)$p
p2<-ks.test(x12,"pnorm",1,3)$p
if ((p1<0.05) & (p1+p2<0.05))
p<-p+1
}
cat("power of n = ",n," is: ",p/N,"\n")
}
```


Doi: https://doi.org/10.54216/GJMSA.040201
Received: November 23, 2022 Accepted: March 17, 2023