Empirical analysis of IEEE 754 and Posit in statistical methods

Ștefan-Dan Ciocîrlan*, Resul Ebru*, Răzvan-Victor Rughiniș*, Mihnea Vrejoiu**

**Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, 060042, Romania (e-mail: stefan_dan.ciocirlan@upb.ro).*
***National Institute for Research & Development in Informatics - ICI Bucharest, 011455, Romania*

Abstract: Within statistical methods, mathematical operations play a crucial role in performing meaningful analyzes on processed data sets. However, mathematical operations can lead to unexpected results, which appear due to errors related to rounding and precision, and not due to errors in their implementation. The errors that can occur are related to the limitations of the computer that can only process a finite number of bits or to the limitations of the number representation system (NRS) used. So, for obtaining efficiency, it is necessary to choose a number representation system that can provide good accuracy, using the smallest possible number of bits. This paper aims to investigate the IEEE 754 standard via binary floating-point against the Unum type 3 standard via posit. Several tests were developed for various statistical methods to observe how the two number representation systems perform. The results show that a 16-bit posit number representation system is viable for energy and computing time efficiency.

Keywords: Number Representation Systems, IEEE754, Posit, Statistical Methods, Computer Arithmetic

1. INTRODUCTION

Datasets are one of the most valuable resources available for many fields. These data are often processed using statistical methods, so that they can be interpreted and quantified in various ways. To ensure that the results are correct, measures are taken to ensure the integrity of the data, but often the accuracy of the computations performed on the data is a neglected topic. Unfortunately, unexpected results can sometimes occur if the number representation system accuracy is not considered.

Once the dataset is cleaned and ready for processing, it is expected that the process of extracting the data through various computations to run smoothly. However, embedded microprocessors are becoming increasingly vulnerable to slight errors such as rounding errors, and designing a reliable system is a challenge. A simple one-bit error can have serious consequences in critical applications, such as those on bank systems (Quinn, 1983) or air navigation machines (United States General Accounting Office, 1992). Therefore, implementing reliable, fault-tolerant arithmetic is an essential requirement for computing systems.

Number representation systems affect computer applications due to hardware limitations. The problem arises because any hardware configuration can only represent a finite amount of numbers. Thus, a finite set of numbers must be used to attempt to represent the infinite system of real numbers.

Over the past 30 years, floating point representation has been incredibly important to science, engineering, statistics, in fact, any kind of mathematical computation on a computer. The reason for the popularity of floating point is that in a large number of situations, floating point arithmetic (Goldberg, 1991) closely approximates real number arithmetic. However, an infinity of real numbers cannot fit in 64 bits. So it's no wonder that floating-point arithmetic and real arithmetic sometimes differ. In fact, the gap between the two, called round-off error, leads to numerous problems in mathematical calculations (Goubault, 2001; Goubault et al., 2007; Higham, 2002; Forsythe, 1977).

In the article "Beating Floating Point at its Own Game: Posit Arithmetic" (Gustafson and Yonemoto, 2017b), it was introduced a new type of number representation system: Posit, as a replacement for IEEE Standard 754 floating-point numbers (floats). They claim that this new number representation system offers compelling advantages over floats: higher accuracy, bitwise identical results between systems, simpler hardware, and simpler exception handling. A posit processing unit requires less circuitry than IEEE floating-point, and thus, using similar hardware resources, the posit operations per second supported by a chip are greater than float ones and use less power.

It is stated in this article (Gustafson and Yonemoto, 2017b) the possibility that the IEEE754 number representation system may be completely replaced by the Posit representation, due to the higher accuracy it offers. On the other hand, a recent study led by a group of researchers from the University of Lyon (Dinechin et al., 2019) points out the negative parts of the posit representation. They believe that it is still too early to consider abandoning the floating-point NRS in favor of a posit NRS, as there are some cases where floats still provide greater accuracy.

Posit is still a relatively new number representation system and there is still much research to be done in the field of posit arithmetic before it can begin to be used on a wider scale. Thus, the main purpose of this article is to determine whether the posit format could be a suitable replacement for the IEEE 754 floating-point number in statistical methods. The focus will be on understanding the posit format and determining the differences between posit and the IEEE standard in statistical operations.

From the authors' knowledge, this article is the first research work which targets statistical methods for analysis. Related work is done on artificial intelligence (Carmichael et al., 2019), digital signal processing (Kant and Thakur, 2021) and scientific computing (Klöwer et al., 2019) . They showed that 16 bits posit is a good trade-off regarding accuracy, storage space, resource utilization, computing time and energy consumption.

The article is structured in 5 sections. The first section explains the motivation, the problem and the solution. The second section explains the theoretical concepts behind the number representation systems. The third section presents the implementation of the statistical methods used, followed by the next section where the results are analyzed. The last section offers a conclusion based on the results.

2. BACKGROUND

Statistical methods involve a set of mathematical formulas, more or less complex. Because computer arithmetic has certain limitations compared to real arithmetic, floating-point accuracy problem often occurs (Bao and Zhang, 2013; Benz et al., 2012), which affect the result. Considering that in certain areas, a difference of one hundredth can affect the course of a project, over time several variants have been tried, in order to reach a result as faithful as possible.

The article "An Investigation on Inherent Robustness of Posit Data Representation" (Alouani et al., 2021) presents a number of currently existing solutions and their limitations.

Various architecture-level single error correction and double error detection (SECDED) techniques have been proposed and used. The disadvantage of SECDED is the performance offered, which is low due to the additional latency that comes from the additional memory cells and support circuitry used to detect and correct errors.

To overcome the architecture-level limitations, a circuit-level technique has been proposed, also with the aim of increasing fault tolerance. This is done by slowing down the circuit's response, or by increasing its critical load. It is suggested to strengthen the cell using a pass transistor controlled by a signal. Unfortunately, this technique is problematic because it slows down memory, which is not convenient if there is an emerging computing application.

There are a few studies that have addressed numerical algorithms to characterize applications based on their vulnerabilities in propagating mild errors in floating-point programs. They treat the algorithm as a black box and watch what happens when it injects computational errors.

Although they study the reliability of floating-point applications, none of them discusses the accuracy comparison between IEEE754 and the posit representation. Therefore, this paper will investigate in detail the differences between the two.

2.2 IEEE Floating Point format

The IEEE 754 standard (IEEE Computer Society, 1985) presents four formats for floating-point numbers: the basic, single- or double-width ones, and the extended correspondences of these formats. In 2008, "half precision" (16-bit) and "quad precision" (128-bit) were added to the standard.

Floating point numbers are made up of four fundamental components: *The sign* which indicates if the number is positive or negative, *the base* which is implicitly 2 and it is constant, *the exponent* which indicates how many times is the base multiplied and *the mantissa* which dictates the precision of the number.

$$[sign] \times \left([mantissa] * [base]^{[exponent]}\right) \quad (1)$$

The single-precision format, known as floats, has a length of 32 bits: 1 sign bit, 8 exponent bits and 23 fraction bits. The double-precision format, known as double, has a length of 64 bits, being able to represent a bigger interval of numbers: 1 sign bit, 11 exponent bits and 52 fraction bits.

2.3 Unum Format

This concept of unum (Gustafson, 2015) was proposed as an alternative to the IEEE floating-number standard. The unum format is used for both real numbers and ranges of real numbers and has evolved over the years into 3 different types.

Gustafson describes in his book (Gustafson, 2015) that type I unums can represent either a float or an open range of adjacent floats, when there is no concrete answer following a calculation. To make this possible, they have an extra bit – the ubit (uncertainty bit) – at the end of the mantissa to indicate whether it corresponds to an exact value or a range. Type I unum took the format of the IEEE 754 components: sign, mantissa, and exponent bits. Compared to floats, the length of the mantissa and exponent fields are variable. Their disadvantage is that this variable length requires special hardware implementations.

To solve the shortcomings of the first type, such as the complexity of the hardware implementation, the type II unum was proposed. It is no longer compatible with IEEE floating-point numbers, presenting a mathematical model based on the representation of values on a real projective line. The key concept is that the point where numbers change from positive to negative is the point where positive real numbers turn into negative numbers, and that point represents the value ± ∞. Although it exhibits many ideal mathematical properties, this type is also not very scalable.

The unum type III – Posit (Posit Working Group, 2018) – is based on the same real projective line, such as type II, but the hardware implementation is similar to the one present for IEEE 754 floating-point. Posit has the length of the exponent and of the mantissa variable, and it has 4 components: *the sign* which indicates if the number is positive or negative, *the regime* which is variable and is used to calculate the scale factor of $useed^k$, *the exponent* and *the fraction.*

$$[sign] \times useed^k \times 2^{[exponent]} \times (1 + [\text{fraction}]) \quad (2)$$

There are multiple domains who are interested in implementing the posit number representation system as an alternative to floating point in order to obtain greater accuracy and efficiency (Carmichael et al., 2019; Chen et al., 2018; Johnson, 2018; Langroudi et al., 2018; Montero et al., 2021). Also, the hardware cost should also be taken into consideration

when thinking about adopting a new number representation system (Podobas and Matsuoka, 2018; Uguen et al., 2019).

2.4 Posit vs IEEE floating-point

Posit uses a fixed number of bits, although the number of the exponent or mantissa is flexible: it can be a few bits up to close to size. Like IEEE 754 floats, posit uses the same type of low-level circuitry, but occupies a smaller chip area due to their simplicity. Several FPGA experiments have pointed out significantly reduced latency for posits compared to floats (Gustafson, 2017a).

Lindstrom, Lloyd and Hittinger point out in this article (Lindstrom et al., 2018) some major differences between the two types of representation, differences that favor the posit representation:

- Posits have a variable exponent length, this means that it will have fewer bits for small value exponents, while floats have a fixed length. It can be said that this way the concept of "tapered precision" is used - higher precision is assigned to numbers that appear more frequently, close to unity. This concept was described by Morris 50 years ago (Morris, 1971).
- In posit, the exponent is encoded by an *es* environment variable, which indicates the size of the exponent in bits.
- The posits do not have subnormal values, the only exception being y = 0. Otherwise, any finite value uses the expression $(1 + f)$. Floats also show subnormal values.
- Posits do not have a NaN representation, compared to IEEE floats, which have multiple representation modes. However, $\pm \infty$ can be considered NaN if a certain environment settings is used.

From a computational point of view, (Gustafson and Yonemoto, 2017b) studied the behaviour of the two representations when it comes to basic mathematical operations. The comparison was made between 8-bit posits with 1-bit exponent and 8-bit floats with 4-bit exponent. According to their tests, it can be observed that Type III unums are more accurate, have greater accuracy and dynamic range, and better closure. In some cases they can produce answers with higher accuracy even though they use the same number of bits as floats, or even the same results with fewer bits (Bojnordi, 2020). However, there are some cases that have been studied in this article (Dinechin et al., 2019) that emphasize that floats should be used in favor of posit for certain calculations, especially when it comes to multiplication and division. Therefore, these operations need to be further explored to see if these problems persist if the number of bits used per posit is increased.

3. IMPLEMENTATION

The first part of the application, the one where the elementary statistical methods (addition, multiplication, variance, and standard deviation) were implemented, was written in Python, and the rest of the statistical operations (Pearson, Kendal, Spearmen coefficients, Kolmogorov-Smirnov, Shapiro-Wilk, Anova, Kruskal-Wallis, t-test, Mann-Whitney U, chi-square) were implemented in Scala. The two programming languages were chosen, based on the libraries offered for number representation systems. The switch from Python to Scala was made, due to the number representation library present in the latter, which is more powerful and offers more features.

In Scala, the comparison of the results between the 2 types of number representation systems– Float and Posit – was done by implementing the statistical methods and running these functions on the same dataset for each numeric type, both 16-bit, 32-bit and 64-bit. The results of 16 bits posit (P16), 32 bits posit (P32), 64 bits posit (P64), half-precision IEEE754 (F16), single-precision IEEE754 (F32), double-precision IEEE754 (F64) were then compared with the actual mathematical result to calculate the accuracy of the numerical sets. The number representation system posit uses 2 bits for exponent, while F16 has 5 exponent bits and 10 mantissa bits, F32 has 8 exponent bits and 23 mantissa bits, and F64 has 11 exponent bits and 52 mantissa bits. In python, only P32 and F32 number representations were studied. The F64 results was considered as correct and then, the results obtained from F32 and P32 were compared with the “correct” answer.

The datasets for the methods implemented in Python were taken from the website of the national institute of statistics (Statistics, n.d.). The samples were chosen with values between [0, 1] and >1, and for each of them there were <100, [300, 1500], >10000 number of entries. The datasets used in scala for the rest of the statistical methods, can be found here (Datasets, 2022).

3.1 Pearson Coefficient (Pearson, 1900)

The Pearson correlation coefficient is a number between -1 and 1, which represents the degree of association between 2 continuous variables, as well as the direction of the relationship between them. The higher the absolute value of the coefficient, the more strongly the 2 variables are connected. The calculation formula is as follows:

$$Rxy = \frac{n*\sum xy - \sum x \sum y}{\sqrt{n*\sum x^2 - (\sum x)^2} * \sqrt{n*\sum y^2 - (\sum y)^2}} \quad (3)$$

3.2 Kendall Coefficient (Kendall, 1938)

The Kendall correlation coefficient is a non-parametric coefficient, between 0 and 1, which measures the strength of the dependence between 2 variables. A higher coefficient represents a stronger relationship. This coefficient is based on the ranks and not on the actual values in the data set. So, after calculating the rank of each value, there is the following formula for calculating the coefficient, where C is the number of concordant pairs, and D is the number of disconcordant pairs:

$$\tau = \frac{C-D}{C+D} \quad (4)$$

3.3 Spearman Coefficient (Spearman, 1961)

The Spearman correlation coefficient is a non-parametric coefficient between -1 and 1 that uses ranks to calculate the monotonicity relationship between 2 variables. It is considered the analogue of the Pearson coefficient, but it is calculated using the rank of each value instead of the raw value of the data. A higher absolute value of the coefficient represents a

stronger relationship. After assigning the rank of each value, the formula for calculating the Spearman coefficient is:

$$\rho = \frac{\sum(X-\bar{X})*(Y-\bar{Y})}{n*\sqrt{\frac{1}{n}*\sum(X-\bar{X})^2}*\sqrt{\frac{1}{n}*\sum(Y-\bar{Y})^2}} \quad (5)$$

3.4 Kolmogorov-Smirnov Test (Kolmogorov, 1933; Smirnov, 1948)

The Kolmogorov-Smirnov test is a non-parametric test used to detect whether a set of data is normally distributed using a standard distribution or it can be used to compare 2 sets of data if they are normally distributed to each other. To calculate the test statistic for both one dataset or 2 datasets, the following steps are used:

- the relative frequencies for each dataset is calculated
- for each set, the differences between the relative frequencies with those of the other set or with the absolute ones, if there is only one dataset, are calculated
- the maximum difference is chosen and that represents the Kolmogorov-Smirnov statistic which will then be compared with a threshold to determine whether to reject or accept the null hypothesis

3.5 Shapiro-Wilk Test (Shapiro and Wilk, 1965)

The Shapiro Wilk test is used in the field of statistics to test whether the analyzed data is part of a normal distribution. If the value of the obtained result is below a certain threshold, then the null hypothesis is rejected and it is considered that the data are not normally distributed. Usually this test is used for large datasets (> 30 entries) to be able to get a valid result. The values in the data set must be sorted in ascending order beforehand in order to apply the formula (a is a constant assigned to each variable):

$$W = \frac{(\sum a*x)^2}{\sum(x-\bar{x})^2} \quad (6)$$

3.7 Variance

This metric provides information about how far individual values are from the crowd. The variance of a data set is calculated according to the following formula, where μ is the arithmetic mean:

$$Var(X) = \sum(X-\mu)^2 \quad (7)$$

3.8 Anova test (Fisher, 1921)

The parametric Anova test is an analysis of variance used to determine the existence of a statistically significant difference between 2 or more groups. There are several types of Anova tests, among the most well-known being one-way (when it has only one classification criterion) and two-way (when it has 2 classification criteria). The obtained result is compared with a threshold determined by the degrees of freedom of the classification criteria, to determine the rejection or approval of the null hypothesis. To calculate the one-way or two-way Anova statistic, the steps below are followed:

- The arithmetic mean of each group and the arithmetic mean of all data are calculated
- The deviation of each value from the groups to the mean of the respective group is calculated and then the sum of the squares of the deviations is determined. The same is done for the sum of the squares of the total deviations, which is calculated in the same way, but the total average of the data is used this time
- The degrees of freedom are calculated
- The variance between and within each group is calculated
- The final statistic is given by the ratio of variance between/variance within

3.9 Kruskal-Wallis test (Kruskal and Wallis, 1952)

The non-parametric Kruskal-Wallis test is also called one-way Anova on ranks. It determines whether the analyzed samples come from the same distribution. This is often used in place of the Anova test when the criteria for the Anova null hypothesis are not met. The test determines whether the median of the analyzed groups are equal or different.

$$H = \left(\frac{12}{N*(N+1)}\sum\frac{R^2}{n}\right) - 3*(N+1) \quad (8)$$

3.10 T-test (Helmert, 1876)

The parametric T-test is a method similar to the Anova test, but it is only used for 2 groups. It examines how much the 2 groups differ from each other. The higher the result, the more significant the difference between the 2 groups. This test is only used for data from a normal distribution.

$$t = \frac{\bar{x}-\mu}{\frac{s}{\sqrt{n}}} \quad (9)$$

3.11 Mann-Whitney U test (Mann and Whitney, 1947)

The non-parametric Mann-Whitney test is an alternative to the t-test, which is used for ordinal data or the condition of the t-test that the data belong to a normal distribution is not met. For the calculation, the values in the dataset are assigned ranks. To calculate the Mann-Whitney statistic, the steps below are followed:

- The sum of the ranks of each group is calculated
- Calculate U for each group according to the formula
- The Mann-Whitney statistic is represented by the U with the lowest value

$$U = s - \frac{n*(n+1)}{2} \quad (10)$$

3.12 Chi-square test (Pearson, 1900)

The Chi-square test is a statistical test that determines whether there is a statistically significant difference between the observed and expected data. A small value means that the relationship between the 2 sets is strong

$$c = \sum\frac{(Observed-Expected)^2}{Expected} \quad (11)$$

3.13 Decimal Accuracy

The decimal accuracy is used to create the plots shown in section 4. The following formula was used:

$$decimal\ error = \left|\log_{10}\frac{Xcomputed}{Xexact}\right| \quad (12)$$

$$decimal\ accuracy = \left|\log_{10}\frac{1}{decimal\ error}\right| \quad (13)$$

4. EVALUATION

4.1 Addition

According to Figure 1, where V represents the range of the values and E the number of inputs in the sample, it can be observed that the accuracy of the results obtained in posit is better than those obtained in float, on the same number of bits (32). This can be easily seen in larger datasets. It can also be noted that regardless of the range of the value and the number of entries, P32 outperforms F32. However, on some tests, there is identical accuracy on both P32 and F32. This is probably due to the fact that there were fewer inputs and the numbers were greater than 1. Thus, the bits used by F32 were sufficient to represent the sum for those cases.

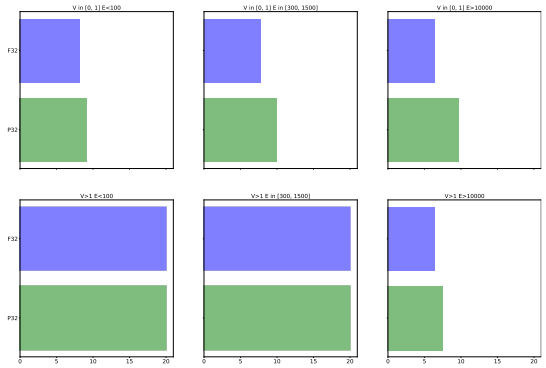

Fig. 1. Decimal accuracy for addition.

4.2 Arithmetic Mean

According to Figure 2, it can be seen that posit has slightly better accuracy on all tests and shows more correct digits, but the differences are not extremely large between the two. Of course, as the number of inputs (including calculations) increases, the results obtained in F32 start to have less accuracy. The higher accuracy of P32 it may be due to the maximum number of bits declared for the exponent, compared to F32 which has a fixed number of bits.

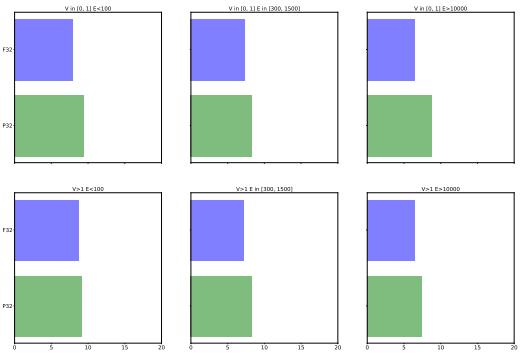

Fig. 2. Decimal accuracy for arithmetic mean.

4.3 Variance

According to Figure 3, it can be observed that F32 has better accuracy than P32 on all tests. On tests where the values in the datasets are between [0, 1], P32 doesn't have as good accuracy as F32, but it's pretty close. It also features a larger number of decimal places. On tests where the values in the data sets are above 1, P32 is very far from the correct answer.

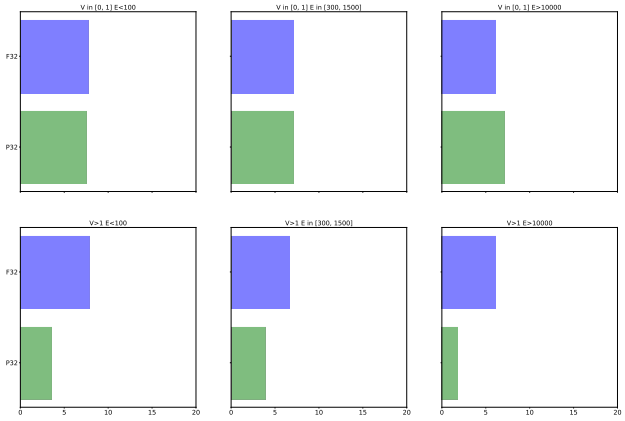

Fig. 3. Decimal accuracy for variance.

4.4 Standard Deviation

The standard deviation is calculated as square root of variance. On tests where the values in the datasets are between [0, 1], P32 has more correct digits. On tests where the values in the datasets are above 1, both P32 and F32 have progressively worse accuracy.

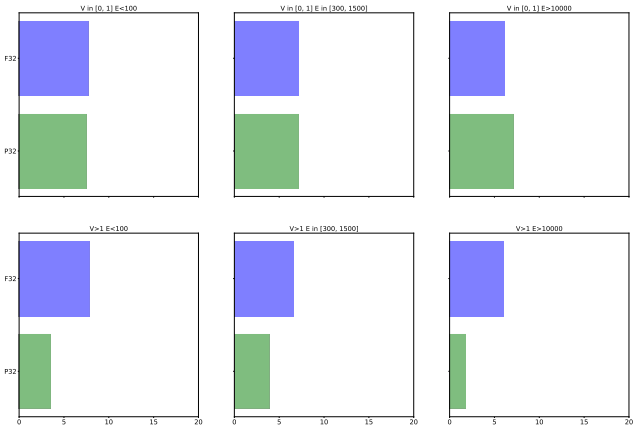

Fig. 4. Decimal accuracy for standard deviation.

4.5 Pearson Coefficient

According to Figure 5, it can be noticed that the accuracy obtained by posit is better than that of the float, on the same number of bits. Also, being able to get valid results on fewer bits, as it can be seen for posit in some cases, is a win in terms of both memory and execution speed. According to the results obtained in 4.1, where it was analyzed the performance of the float versus posit on elementary mathematical operations, it can be considered that the difference in accuracy between posit, float and the real result was expected. This is because posit provides much better accuracy than posit on a smaller number of bits when discussing addition, subtraction, but its accuracy becomes lower when variance or radical calculations are also involved, especially on larger data sets.

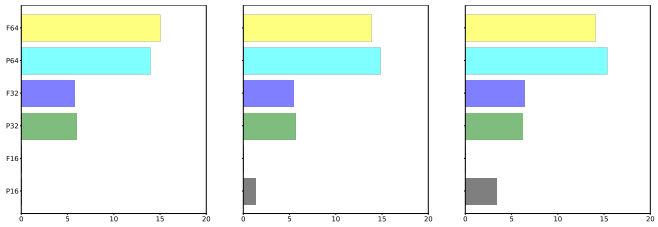

Fig. 5. Decimal accuracy for Pearson coefficient.

4.6 Kendall Coefficient

Kendall's coefficient involves, arithmetically, a subtraction, an addition, and a division. Thus, the mathematical operations are not complex enough, so that there is a big difference between the 2 number representation systems, but the accuracy obtained by posit is better than that of float. It can also be observed in the first case that posit gives 16-bit result and float cannot display the result. This indicates that when the number of entries in the database increases, implicitly the number of concordant and discordant pairs, posit can provide a more accurate result and at the same time on a smaller number of bits. According to the results obtained in *4.1*, a better performance of posit was expected, since they perform better than float on elementary operations.

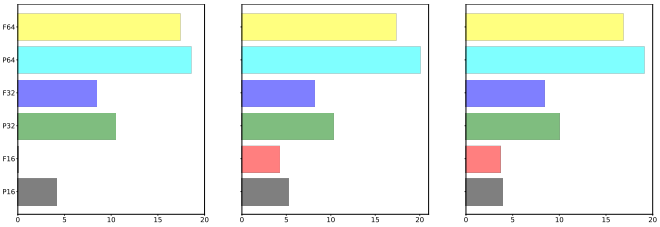

Fig. 6. Decimal accuracy for kendall coefficient.

4.7 Spearman Coefficient

The arithmetic operations of the Spearman coefficient are similar to those of the Pearson coefficient formula, the difference being that the former uses ranks and the latter uses observed values. Thus, it can be noticed that the accuracy of the posit representation is like that of the float representation on a larger number of bits. This happens because of the variance calculation, where posit performs worse. Next, it should be noted that posit also provides 16-bit results, but its accuracy decreases with the increase in the number of entries in the data set. In addition, P32 offers a larger number of decimals than F32.

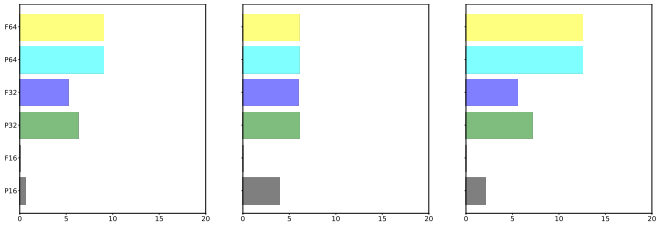

Fig. 7. Decimal accuracy for spearman coefficient.

4.8 Kolmogorov-Smirnov Test

It can be observed that for both one dataset and two datasets, the precision obtained on the different number representation systems is close. Although posit is closer to the actual value of the result at any number of bits, the differences are not extremely large. But they still provide more correct digits than their counterpart. The Kolmogorov-Smirnov test contains only basic mathematical operations (addition, subtraction, multiplication and division). So, on average, for datasets with values greater than 1, the accuracy of the two data types is often similar, and this is also the case in this example. It should be noted that on 16 bits, the bits used by float were not enough to be able to represent the result.

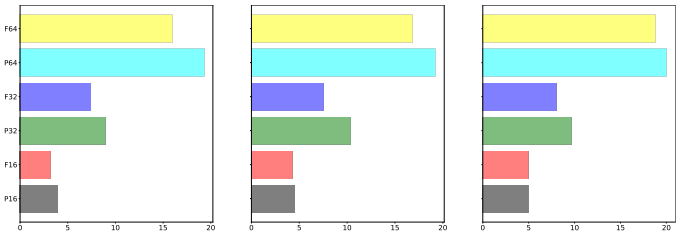

Fig. 8. Decimal accuracy for Kolmogorov-Smirnov Test for one dataset.

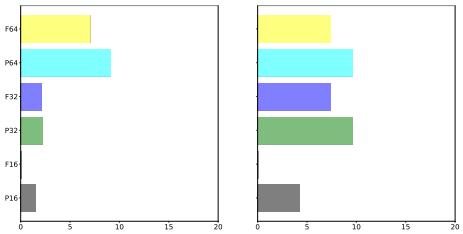

Fig. 9. Decimal accuracy for Kolmogorov-Smirnov Test for two datasets.

4.9 Shapiro-Wilk Test

According to Figure 10, posit has more identical decimal digits with the actual result, compared to float. So, the accuracy of the posit number representation system still provides several significant decimals and overall a result closer to the actual value. The Shapiro-Wilk test contains, in addition to the basic mathematical operations (addition, subtraction, multiplication and division), operations of variance and raising to powers. Although posit presents a problem of accuracy in calculating the variance compared to float, the multitude of the operations in the formula plus the number of bits on which the number is represented, made the posit number representation system to still have better accuracy than its competitor. As the dataset grows, the results of the operations become larger and larger, and the 16 bits (5 exponent and 10 fraction) are no longer enough to represent the number for floats, while posit16 continues to provide relevant results with good accuracy.

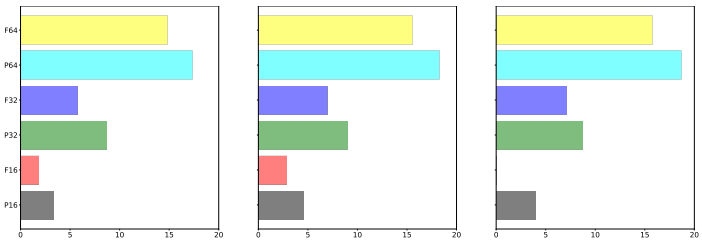

Fig. 10. Decimal accuracy for Shapiro-Wilk test.

4.10 Anova

The mathematical operations for determining the Anova statistic involves, in addition to basic arithmetic, operations for variance. According to the results of *4.3*, Posit gives poorer accuracy results on the variance calculation. However, all this accumulation of operations led to a similar accuracy between posit and float, the difference being the number of decimals displayed, where posit wins. It should be noted that posit accuracy is still better than float for both one-way Anova and two-way Anova.

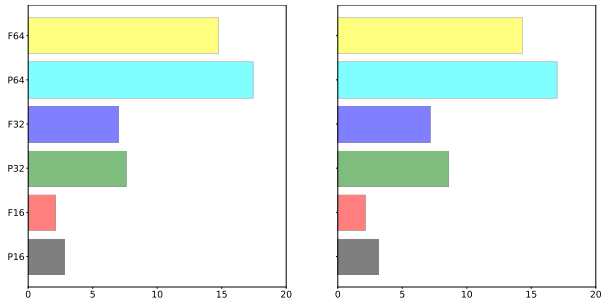

Fig. 11. Decimal accuracy for one-way Anova.

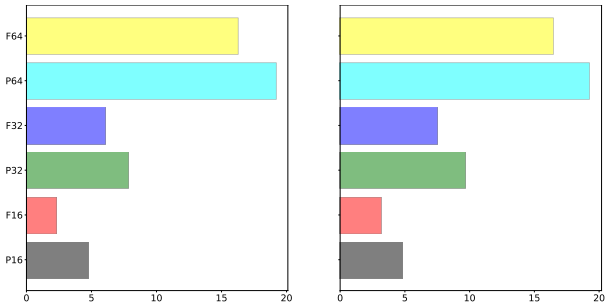

Fig. 12. Decimal accuracy for two-way Anova.

4.11 Kruskal-Wallis Test

According to figure 13, the accuracy of posit is better than that of float, it has a higher number of decimals and the number of exact decimals is higher than that of float. The Kruskal-Wallis test is a rank test, which means that it has values from 1 to N - the number of values in the data set. As a results, for small and medium data sets, the representation problem does not appear even on 16 bits. In addition, the fact that the formula is composed of basic arithmetic operations contributes to the previous idea.

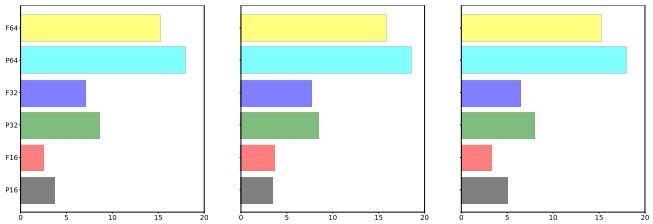

Fig. 13. Decimal accuracy for Kruskal-Wallis test.

4.12 T-test

Within the 2 figures (figure 14 and figure 15), it can be noticed that float has better accuracy than posit and shows a higher number of exact decimals like the actual result. It can be assumed that this is due to the fact that the main operation is a variance calculation. According to *4.3*, posit performs worse on the calculus of variation and square root. So, since the t-test formula mainly consists of these 2 operations, the posit gives a more distant result than its IEEE counterpart.

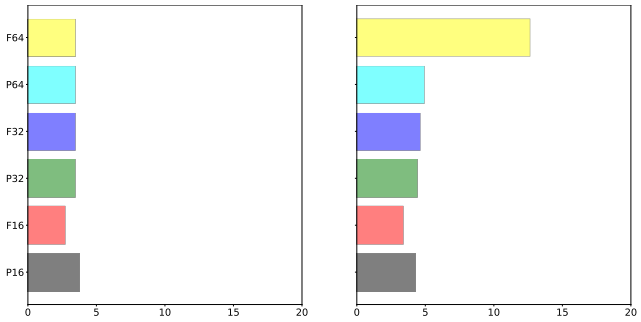

Fig. 14. Decimal accuracy for t-test.

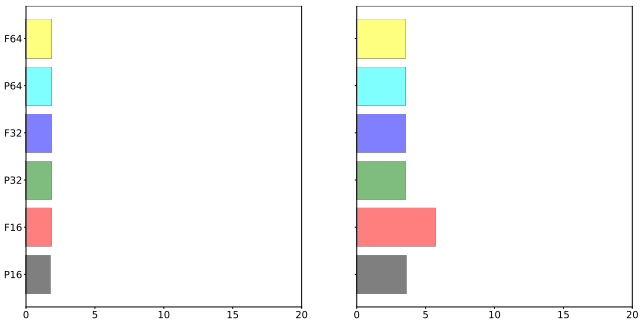

Fig. 15. Decimal accuracy for t-test for pairs.

4.13 Mann-Whitney U Test

The Mann-Whitney test formula is not complex, containing only 3 simple operations: addition, multiplication and division. Additionally, ranks are used and not the values in the dataset, which means that the operations are performed on integers from 1 to N. So, there is no difference in terms of the result in posit or float on the same number of bits. However, it is important to note that as the dataset grows, posit on fewer bits can give a result like a float on more bits, while float on the same number of bits as posit cannot represent the result.

Table 1. Results for Mann-Whitney U test.

Real Result (Q)	Number Representation System	Result
18	Posit: 16	18
	Float: 16	18
	Posit: 32	18
	Float: 32	18
	Posit: 64	18
	Float: 64	18
23305	Posit: 16	23296
	Float: 16	23312
	Posit: 32	23305
	Float: 32	23305
	Posit: 64	23305
	Float: 64	23305
90577	Posit: 16	90496
	Float: 16	Not represented

	Posit: 32	90577
	Float: 32	90577
	Posit: 64	90577
	Float: 64	90577

4.14 Chi-Square Test

Within Chi-Square, it is noticed that there are similar accuracy results as Mann-Whitney. This happens because the arithmetic behind is basic, where both posit and float have similar accuracy, especially on small or medium data sets. So, the main difference that can be said about the fact that posit performs better is that it gives results on fewer bits, while float requires more bits as the dataset grows.

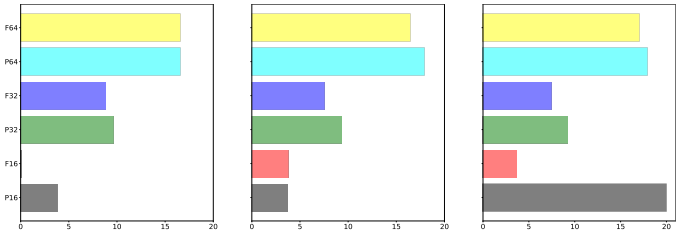

Fig. 16. Decimal accuracy for Chi-Square test.

5. CONCLUSION

This article looked at the effect of changing number representation system within statistical methods. Although at first glance, a difference of a few hundredths may not seem to have much impact on the final result, there are fields, such as biostatistics or astrostatistics, where any extra tenth is considered a significant new information. The statistical functions implemented were chosen because they are relevant functions within the statistical domain. Elementary statistical methods were implemented in Python, and correlation, distribution detection, and statistical differences were written in Scala. The Scala language was used for complex functions because the Posit library in it is more powerful and provides support for elementary operations. The following results were observed in the fight between Posit number representation system with the already established IEEE 754 number representation system.

Python observed results:

- On datasets with values between [0, 1], Posit is at least as accurate as IEEE 754 (standard deviation, variance) and much better than floats on addition and arithmetic mean operations.
- On addition and arithmetic mean operations, Posit outperforms float, regardless of how many entries the data set has.
- On variance and standard deviation operations, on data sets with values greater than 1, IEEE 754 has better accuracy than posit, but the values resulting from float operations also have a fairly large error compared to the correct answer.

Scala Observed Results:

- It can be noticed that the maximum decimal accuracy of posit is higher than that of float on the same number of bits.
- Although the dataset on which the test is performed also matters, posit numbers always provide better accuracy than float on the same number of bits, regardless of the statistical method used.
- Depending on the dataset and the statistical method, IEEE754 numbers represented by float, cannot give a valid result (in this case on 16 bits), while Posit numbers give a good result on the same number of bits; this is due to the flexibility of posit in terms of exponent and fraction, compared to float which has a fixed number (5 bits for exponent and 10 bits for fraction).
- In the case of tests that rely almost entirely on calculating the variance in the formula, such as the t-test, the float has better accuracy than the posit.
- In the case of tests that also contain calculus of variance plus other arithmetic operations, such as Anova, the posit still offers better accuracy, a larger number of decimal digits, as well as better results on fewer bits.
- In Mann-Whitney and Chi-Square, posit and float have a close accuracy, the formula being quite simple from a mathematical point of view and therefore there are no big differences in calculation; however, it can be considered that posit is better than float because it can provide meaningful results on fewer bits.

Following this article, the strengths and weaknesses of the posit numbers within the statistical methods were analyzed, in comparison with the current method of number representation system - IEEE 754 float. Following the implemented operations (basic arithmetic, correlations, statistical differences). The main reason why posit is a strong competitor in the fight to replace the IEEE 754 standard, is the fluctuating character of the exponent and the mantissa. This helps to obtain significant results on a smaller number of bits, thus providing a variant that consumes less memory and is faster in statistical calculations.

In conclusion, Posit can be considered a replacement for IEEE 754 float in statistical methods, often performing better. To see the full power of this new number representation system, more thorough research is still needed.

A first step for future work is to add the remaining statistical methods like: (i) statistical methods to compare the proportions (Fisher exact test, McNemar test, Cochrane Q test, z-test for proportions); (ii) logistic regression analysis; (iii) survival analysis, and (iv) receiver operating characteristics curve method. With these improvements, the proposed statistical methods library will have similar functionalities with scipy.stats. Developers and researchers can use the best number representation system given the accuracy of every method they used. In this way, the accuracy of the final results will be further improved. A second step to increase the dataset size is to integrate the implemented methods with Apache Spark. These will make the proposed library accessible for processing big data information.

ACKNOWLEDGEMENTS

This work was partly supported by the Bitdefender's University PhD Grants Program 2019-2022 and by the Google IoT/Wearables Student Grants 2022. The results presented in this article has been funded by the Ministry of Investments and European Projects through the Human Capital Sectoral Operational Program 2014-2020, Contract no. 62461/03.06.2022, SMIS code 153735.

REFERENCES

Alouani, I., Khalifa, A. B., Merchant, F. & Leupers, R., 2021. An Investigation on Inherent Robustness of Posit Data Representation. *20th International Conference on Embedded Systems (VLSID).*

Bao, T. & Zhang, X., 2013. On-the-fly Detection of Instability problems in floating-point program execution. *ACM SIGPLAN Notices,* Volume 48, pp. 817-832.

Benz, F., Hildebrant, A. & Hack, S., 2012. A Dynamic Program Analysis to Find Floating-point Accuracy Problems. *ACM SIGPLAN Notices ,* 47(6), pp. 453-462.

Bojnordi, P. B. a. M. N., 2020. *Posit: A Potential Replacement for IEEE 754.* [Online] Available at: https://www.sigarch.org/posit-a-potential-replacement-for-ieee-754/ [Accessed 2 February 2023].

Carmichael, Z., Langroudi, H.F., Khazanov, C., Lillie, J., Gustafson, J.L., and Kudithipudi, D., 2019. Performance-efficiency trade-off of low-precision numerical formats in deep neural networks. *Proceedings of the Conference for Next Generation Arithmetic,* Volume 3, pp. 1-9.

Chen, J., Al-Ars, Z. & Peter Hofstee, H., 2018. A matrix-multiply unit for posits in reconfigurable logic leveraging (open)CAPI. *Proceedings of the Conference for Next Generation Arithmetic,* Volume 1, pp. 1-5.

Datasets, 2022. *Scala Statistical Methods Datasets.* [Online] Available at: https://drive.google.com/drive/folders/12NrNdRQb5tS37erJF-z3Ep0WuOXoci2o?usp=sharing, [Accessed 20 11 2022]

Dinechin, F. d., Forget, L., Muller, J.-M. & Uguen, Y., 2019. Posits: the good, the bad and the ugly. *CoNGA'19: Proceedings of the Conference for Next Generation Arithmetic 2019,* Volume 6, pp. 1-10.

Fisher, R. A. et. al., 1921. On the" Probable Error" of a Coefficient of Correlation Deduced from a Small Sample. *Metron,* Volume 1, pp. 3-32.

Forsythe, G. E. e. a., 1977. Computer methods for mathematical computations. s.l.:Prentice-hall.

Goldberg, D., 1991. What Every Computer Scientist Should Know. *ACM Computing Surveys,,* Volume 23, pp. 5-48.

Goubault, E., 2001. Static Analyses of the Precision of Floating-Point Operations. *Proceedings of the 8th International Symposium on Static Analysis,* pp. 234-259.

Goubault, E., Putot, S., Baufreton, P. & Gassino, J., 2007. Static analysis of the accuracy in control systems: Principles and Experiments. *Revised Selected Papers from the 12th International Workshop on Formal Methods for Industrial Critical Systems. ,* pp. 3-20.

Gustafson, J., 2015. *The End of Error: Unum Computing.* s.l.:Chapman & Hall/CRC Computational Science.

Gustafson, J., 2017a. *Posit Arithmetic, [Online] Available at: https://posithub.org/docs/Posits4.pdf [Accessed 15 11 2022]*.

Gustafson, J. L. & Yonemoto, I., 2017b. Beating Floating Point at its Own Game: Posit. *Supercomputing Frontiers and Innovations: an International Journal,* 4(2), pp. 71-86.

Helmert, F. R., 1876. Die genauigkeit der formel von peters zur berechnung des wahrscheinlichen beobachtungsfehlers director beobachtungen gleicher genauigkeit.. *Astronomische Nachrichten,* Volume 88.

Higham, N. J., 2002. *Accuracy and Stability of Numerical Alghorithms.* 2nd ed. Philadelphia, USA: Society for Industrial and Applied Mathematics.

IEEE Computer Society , 1985. *IEEE Standard for Binary Floating-Point Arithmetic,* American National Standards Institute.

Johnson, J., 2018. *Rethinking floating point for deep learning,* New York: Facebook AI Research.

Kant, M. & Thakur, R., 2021. *Implementation and Performance Improvement of POSIT Multiplier for Advance DSP Applications.* s.l., IEEE.

Kendall, M. G., 1938. A New Measure of Rank Correlation. *Biometrika,* Volume 30, pp. 81-93.

Klöwer, M., Düben, P. D. & Palmer, T. N., 2019. Posits as an alternative to floats for weather and climate models., *Proceedings of the Conference for Next Generation Arithmetic*, pp. 1-8

Kolmogorov, A., 1933. Sulla Determinazione Empirica di Una Legge di Distribuzione. *Giornale dell'Istituto Italiano degli Attuari,* pp. 83-91.

Kruskal, W. H. & Wallis, W. A., 1952. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association,,* Volume 47, pp. 583-621.

Langroudi, S. H. F., Pandit, T. & Kudithipudi, D., 2018. Deep Learning Inference on Embedded Devices: Fixed-Point vs Posit. *Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2),* pp. 19-23.

Lindstrom, P., Lloyd, S. & Hittinger, J., 2018. Universal Coding of the Reals: Alternatives to IEEE Floating Point. *CoNGA '18: Proceedings of the Conference for Next Generation Arithmetic,* Volumen 5, pp. 1-14.

Mann, H. B. & Whitney, D. R., 1947. On a test of whether one of two random variables is stochastically larger than the other.. *The annals of mathematical statistics,* pp. 50-60.

Montero, R. M., Del Barrio, A. A. & Botella, G., 2021. Template-Based Posit Multiplication for Training and Inferring in Neural Networks.

Morris, R, 1971. Tapered Floating Point: A New Floating-Point Representation. *IEEE Transactions on Computers ,* C-20(12), pp. 1578-1579.

Pearson, K., 1900. X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling.. *The London, Edinburgh, and Dublin Philosophical*

Magazine and Journal of Science, Volume 50, pp. 157-175.

Podobas, A. & Matsuoka, S., 2018. Hardware Implementation of POSITs and their application in FPGAs. *International Parallel and Distributed Processing Symposium Workshops,* pp. 138-145.

Posit Working Group, 2018. *Posit Standard Documentation*

Quinn, K., 1983. Ever Had Problems Rounding Off Figures? This Stock Exchange Has*, The Wall Street Journal,* 37.

Shapiro, S. Sanford. & Wilk, M. B., 1965. An analysis of variance test for normality (complete samples). *Biometrika,* 52(3/4), pp. 581-611.

Smirnov, N., 1948. Table for estimating the goodness of fit of empirical distributions. *The annals of mathematical statistics,* 19(2), pp. 279-281.

Spearman, C., 1961. The Proof and Measurement of Association Between Two Things. *J. J. Jenkins & D. G. Paterson (Eds.), Studies in individual differences: The search for intelligence,* pp. 45-58.

Statistics, N. I. o., s.f. *National Institute of Statistics.* [Online] Available at: https://insse.ro/cms/ [Accessed: 17 06 2020].

Uguen, Y., Forget, L. & De Dinechin, F., 2019. Evaluating the Hardware Cost of the Posit Number System. *International Conference on Field Programmable Logic and Applications (FPL).*

United States General Accounting Office, 1992. *Patriot Missile Defense: Software Problem Led to System Failure at Dhahran, Saudi Arabia.* [Online] Available at: https://www.gao.gov/products/imtec-92-26, [Accessed: 16 03 2022]