International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 03, March-2023

Measurement of Computing Times using the Intrinsic Functions of Gfortran

M.A. Sandoval-Hernandez[1], U.A. Filobello-Nino[2], H. Vazquez-Leal[2,3,*], G.C. Velez-Lopez[4], G.J. Morales-Alarcon[5], F. Martinez-Barrios[1], J.C. Vichi-Mendoza[1], J. Chong-Duran[1], C.E. Sampieri-González[2], R. Castaneda-Sheissa[2], J.E. Pretelin-Canela[2], A.E. Gasca-Herrera[2], J.E. Perez-Jacome Friscione[2], N. Bagatella-Flores[6].

[1] Centro de Bachillerato Tecnológico industrial y de servicios No. 190, Av. 15 Col. Venustiano Carranza 2da Sección, Boca del Río, 94297, Veracruz, México.

[2] Facultad de Instrumentación Electrónica, Universidad Veracruzana, Circuito Gonzalo Aguirre Beltrán S/N, Xalapa, 91000, Veracruz, México.

[3] Consejo Veracruzano de Investigación Científica y Desarrollo, Av. Rafael Murillo Vidal No. 1735, Cuauhtémoc, Xalapa, 91069, Veracruz, México.

[4] Instituto Nacional de Astrofísica, Óptica y Electrónica, Luis Enrique Erro 1, Sta. María Tonantzintla, 72840, Puebla, México.

[5] Instituto de Psicología y Educación, Universidad Veracruzana, Agustín Melgar 2, col. 21 de Marzo, Xalapa, 91010 Veracruz, México.

[6] Facultad de Física, Universidad Veracruzana, Paseo No. 112. Desarrollo Habitacional Nuevo Xalapa, Xalapa, 91097, Veracruz, México.

***Abstract*— The primary objective of this scientific article is to emphasize the educational importance of evaluating computation times of mathematical functions within didactic software, particularly in the context of Gfortran's implementation. We argue that understanding the computational efficiency of these functions is crucial to enhance the teaching and learning of mathematics.**

Furthermore, we discuss various alternatives to reduce computation times, particularly when dealing with polynomial expressions. By providing a comprehensive analysis of these alternatives, we aim to offer educators valuable insights into how to optimize mathematical software to improve computational efficiency. Overall, this article contributes to the ongoing conversation on the significance of computation times in the teaching and learning of mathematics and provides practical solutions for optimizing the implementation of didactic software.

Keywords*— *Computing time, transcendental functions, mathematical speech, mathematical functions, GFortran.

I. INTRODUCTION

Generally, in numerical analysis courses, error analysis is important. The accuracy provided by any numerical algorithm is in terms of absolute or relative error [1, 2]. However, in some cases the accuracy obtained is expressed in terms of the significant digits [3, 4].

The basic books of numerical analysis that are used in universities do not mention about the time invested in executing an algorithm. This happens because the books are influenced by the school mathematical discourse (DME in spanish). The DME has been described as a language that is hegemonic, utilitarian, and lacks meaning, arguments, and procedures centered on mathematical objects [5, 6]. In simpler terms, it is a language that remains unchanged despite innovations made in mathematics and teaching. This is because the concepts taught are not modified, only the way they are taught. In fact, many textbooks used in classrooms are based on the DME framework [7, 8, 9].

Knowing the execution times invested by digital devices is important because it allows algorithms to be optimized. Computing times have many implications for power consumption [10]. This is very important in embedded systems which do not have a math coprocessor. For example, the importance of knowing computation times is important in digital signal processing [11]. In this case, the aim is to obtain computation times as short as possible. Several factors intervene to reduce these times that depend on how the hardware is designed [12, 13]. It also depends on the arithmetic that is used. In embedded systems where there is no math co-processor, fixed-point arithmetic is used [14-17]. In [18] some basics of fixed-point arithmetic are presented through didactic examples.

Another alternative to reduce computing times is to use Horner's rule to reduce the multiplications generated by large exponents. Horner's rule is a computational procedure that is highly effective in determining the numerical outcome of a polynomial function. This algorithmic method involves a stepwise approach to evaluate a polynomial by means of a sequence of additions and multiplications, in which each term of the polynomial is factored in a way that reduces the degree of the polynomial by one [19,20]. By utilizing this strategy, Horner's rule allows for the computation of a polynomial's value in a quick and efficient manner, making it a valuable tool in various scientific and mathematical contexts [21].

In [22] the Horner algorithm was used to reduce computing times in polynomial expressions that were the result of solving differential equations. Likewise, it can be shown that using Horner's rule, a time reduction can be obtained in the approximation of the Cumulative distribution function (CDF)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 03, March-2023

obtained in [23] where there is a polynomial of order 9 against the CDF obtained in [24] since a hyperbolic tangent is used

This paper is organized as follows. In Section II, we introduce the some basics of Fortran. In section III presents the discusion about of computing times for intrinsec matematical functions for Fortran. Finally, a concluding remark is given in Section IV.

II. SOME BASICS OF FORTRAN

Fortran has a rich and extensive history, and it continues to evolve to this day. The first proposal for the language was created by J.W. Backus in 1953, and the first international standard was approved in 1966. In 2018, the most recent major standard revision, Fortran 2018, was published, introducing new features. The next major revision, previously known as Fortran 2020 and now referred to as Fortran 202X, is currently under development [25-28].

Fortran, which stands for Formula Translation, is a widely used procedural, imperative programming language that is particularly well-suited for scientific computing and numerical computation. Since 2003, the standard version of Fortran has also supported object-oriented programming. In addition, Fortran 2008 introduced co-arrays that enable Single Program Multiple Data (SPMD) parallel programming. To discuss the Fortran language in general, this tag must be assigned to all relevant questions. Other specific tags can be added for compilers, language revisions, and particular aspects of usage [26-28].

Intrinsic functions are functions that are implemented within the compiler. Some intrinsic functions are included in the runtime library, such as the libgfortran library, so source code for these functions may be available. Other intrinsic functions may not actually exist as a function at all, and instead, the compiler will insert assembly or machine code at the location where the function is called [29].

Although intrinsic functions are not typically implemented in Fortran itself, but in a systems programming language such as C or C++, some intrinsic functions may call code that is written in assembly language for a specific CPU architecture. Additionally, many intrinsic functions call functions from the standard C library, such as the GLIBC implementation [29]. Some intrinsic functions can theoretically be written in Fortran, while others cannot. For example, functions with an indeterminate number of arguments like "min" or type transformation functions like "transfer" cannot be written in Fortran. Many intrinsic functions use internal information that is only available to the compiler, such as the content of the array descriptor or the polymorphic object descriptor.

Table 1 shows some of the intrinsic mathematical functions. Note that the arc functions have not been included. In all cases it is possible to have the real type.

TABLE I. FORTRAN INTRINSEC MATEMATICAL FUNCTIONS

Function	Meaning	Arg type	Return type
ABS(*x*)	absolute value of *x*	INT/REAL	INT/REAL
SQRT(*x*)	square root of *x*	REAL	REAL
SIN(*x*)	sine of *x*	REAL	REAL
COS(*x*)	cosine of *x*	REAL	REAL
TAN(*x*)	tangent of *x*	REAL	REAL
EXP(*x*)	exponential of *x*	REAL	REAL
LOG(*x*)	natural logarithm of *x*	REAL	REAL
SINH(*x*)	hyperbolic sine of *x*	REAL	REAL
COSH(*x*)	hyperbolic cosine of *x*	REAL	REAL
TANH(*x*)	hyperbolic tangent of *x*	REAL	REAL

III. NUMERICAL SUMULATIONS AND DISCUSION

As previously mentioned, there is evidence to suggest that the composition of academic textbooks is influenced by DME [7]. This phenomenon may account for the lack of emphasis on computation times in basic numerical analysis textbooks commonly used in universities. Similarly, when examining the teaching of precalculus [30] in high school, a greater emphasis appears to be placed on the utilization of information technology.

For example, the GeoGebra software is used to facilitate the learning of mathematics such as geometry and the study of functions. However, we consider that if some mathematical software such as Maple, Matlab, C++, Fortran, Python, among others, is used, it is possible to measure the time that the computer takes to execute a built-in function, such as the exponential function. This allows developing the critical thinking of the student since in general they are never questioned about the computing times invested by the didactic tool that is being used for the educational purpose.

In the realm of mathematics education, the utilization of educational technology has become increasingly prevalent. One such example is the GeoGebra software, which has been widely adopted as a tool to facilitate the learning of mathematical concepts, such as geometry and the study of functions. However, we believe that it is essential to expand the scope of mathematical software used in the classroom to include programs such as Maple, Matlab, C++, Fortran, Python, and other related software applications.

By incorporating such software into the curriculum, it is possible to measure the time that the computer takes to execute a built-in function, such as the exponential function. This, in turn, can be used to develop the critical thinking skills of the student. It is crucial to prompt students to question the computing times invested by the didactic tool being utilized for educational purposes, which is often overlooked in traditional learning environments.

Of the functions that are presented in table 1, the computation times have been measured for the circular and hyperbolic trigonometric functions, as well as the logarithm and the exponential function. Figure 1 reports the computing times for all circular trigonometric functions. In this figure it can be seen that the computation times range from 10 to less than 15 nanoseconds.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 03, March-2023

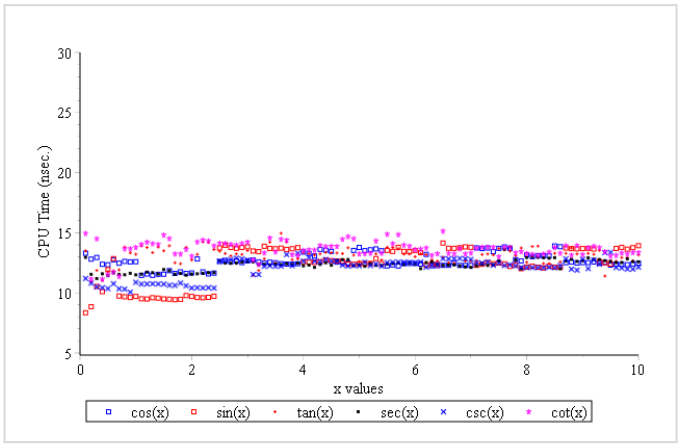

Figure 1. CPU time for circular trigonometric functions.

Figure 2 shows that the time used for the exponential function is on average 11 ns while for the natural logarithm function it is 15 ns.

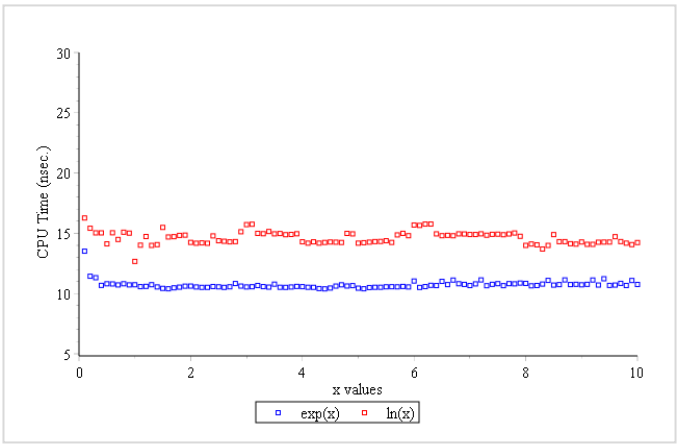

Figure 2. CPU time for exponential and logarithm functions.

Figure 3 shows the computation times for the hyperbolic trigonometric functions. It can be seen that in general the times used to evaluate these functions is greater than in the case of circular trigonometric functions. This is to be expected since hyperbolic functions are found in terms of exponential functions. However, if we wish to make a more significant comparison, it will be possible to verify that the times used to evaluate are greater in hyperbolic, logarithmic and transcendental functions than in circular trigonometric functions; this will happen if we use Taylor series expansions.

In the case of the Taylor series expansion for the exponential and logarithm function, we find that the power series includes both even and odd exponents, which is why the computation times increase.

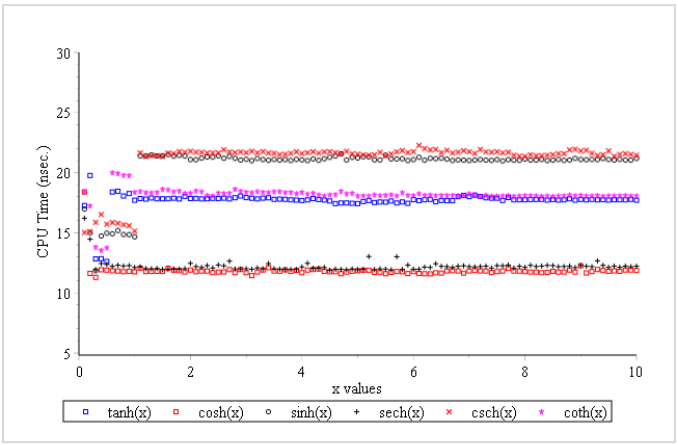

Figure 3. CPU time for hyperbolic trigonometric functions.

Finally, throughout the entirely of the simulations conducted, it was necessary to divide the given interval into one hundred separate parts to achieve the desired level of precision. Moreover, each individual point within this interval was evaluated a total of one hundred thousand times to ensure accuracy in the resulting data. This approach was taken to ensure the reliability of the simulations and to provide comprehensive and accurate insights into behavior of the system under study. Overall, these measures were essential for generating high-quality results and advancing our understanding of the underlying mathematical principles at play.

It is left to the reader or student to determine the computation times used by a computer when implementing a polynomial. Likewise, you are also urged to carry out measurements if the polynomials are modified using Horner's Rule [19, 20] or using fixed-point arithmetic [18].

In this paper the computing system utilized in the simulations comprised an Intel Core Pentium(R) Intel® Core™ i7-7700 Central Processing Unit, operating at a frequency of 3.60 GHz and with 8 processing cores. The operating system employed was Linux Ubuntu 18.04.6 LTS, and the compiler utilized for code compilation was GFortran 7.5.0. Additionally, the system was equipped with a graphics card, specifically the NVIDIA GeForce GTX 1050 Ti/PCIe/SSE2, for improved graphics performance.

IV. CONCLUDING REMARKS

The calculation of computation times is a crucial aspect of evaluating mathematical functions, particularly those that are intrinsic, such as polynomials. In this article, we aim to emphasize the significance of this process and its potential impact on the accuracy of the results obtained. By taking into account the computation time required for a function to execute, researchers and teachers can gain a deeper understanding of the function's behavior and make more informed decisions based on the resulting data. Therefore, it is essential to consider computation times as a key factor in the evaluation of mathematical functions.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 03, March-2023

DECLARATION OF INTERESTS STATEMENT

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

ACKNOWLEDGMENTS

Authors would like to thank Roberto Ruiz Gomez for his contribution to this project. The authors are grateful to the anonymous referee for a careful checking of details and helpful comments that improved this paper.

REFERENCES

[1] Burden, Richard L., J. Douglas Faires, and Annette M. Burden. *Numerical analysis*. Cengage learning, 2015.
[2] Chapra, Steven C., and Raymond P. Canale. *Numerical methods for engineers*. Vol. 1221. New York: Mcgraw-hill, 2011.
[3] Barry, D. A., Culligan-Hensley, P.J. and Barry, S.J., *Real values of the W-function*, ACM Transactions on Mathematical Software (TOMS), ACM New York, NY, USA, vol. 21, no. 2, pp. 161–171, 1995.
[4] Barry, D.A., Barry, S.J. and Culligan-Hensley, P.J., *Algorithm 743: WAPR–A Fortran routine for calculating real values of the W-function*, ACM Transactions on Mathematical Software (TOMS), ACM New York, NY, USA, vol. 21, no. 2, pp. 172–181, 1995.
[5] Soto, D., Gomez, K., Silva, H. y Cordero, F. *Exclusión, cotidiano e identidad: una problemática fundamental del aprendizaje de la matemática*. Comité Latinoamericano de Matemática Educativa, pp. 1041-1048, 2012.
[6] Soto, D. y Cantoral, R. *Discurso matemático escolar y exclusión. Una visión socioepistemológica*, Bolema: Boletim de Educação Matemática, vol. 28 no. 50, pp. 1525–1544,2014.
[7] Uriza, R. C. and Espinosa, G. M. and Gasperini, D. R. *Análisis del discurso matemático escolar en los libros de texto: una mirada desde la teoría socioepistemológica*, Avances de Investigación en Educación Matemática, no. 8, pp. 9–28, 2015.
[8] Sandoval-Hernández, Mario, Hernandez-Mendez, Sergio, Torreblanca-Bouchan, Salvador, Diaz-Arango, Gerardo, *Actualización de contenidos en el campo disciplinar de matemáticas del componente propedéutico del bachillerato tecnológico: el caso de las funciones especiales*. *RIDE. Revista Iberoamericana para la Investigación y el Desarrollo Educativo* 12.23 (2021).
[9] Sandoval-Hernández, M.A., Vázquez-Leal, H., Huerta-Chua, J. Castro-González, F.J., Filobello-Nino, U.A. *Didáctica del graficado de funciones: el caso de las funciones piecewise*. RIDE. Revista Iberoamericana para la Investigación y el Desarrollo Educativo 12.24 (2022).
[10] Anton Cervin, *Fix Point Implementation of Control Algorithms", Lund University. A G Przybył, Andrzej. "Fixed-point arithmetic unit with a scaling mechanism for FPGA-based embedded systems. Electronics* 10.10 1164.(2021):
[11] Roman, Kuc. *Introduction to Digital Signal Processing*, 1982.
[12] J. Le Maire, N. Brunie, F. de Dinechin, J. M. Muller, *Computing floating-point logarithms with fixed-point operations*. 2016 IEEE 23nd Symposium on Computer Arithmetic (ARITH). IEEE, 2016.
[13] Nikola M. Nenadic, and Svetlana B. Mladenovic. *Fast division on fixedpoint DSP processors using Newton-Raphson method*. EUROCON 2005-The International Conference on" Computer as a Tool". Vol. 1. IEEE, 2005.
[14] Bishop, David. *Fixed point package user's guide*. *Packages and bodies for the IEEE* (2006): 1076-2008.
[15] Yates, Randy, Fixed-point arithmetic: *An introduction, Digital Signal Labs*. vol. 81, no. 83, pp. 15, 2009.
[16] Pyeatt, Larry and Ughetta, William, *ARM 64-Bit Assembly Language*, Newnes, 2019.
[17] Application Note 33, *Fixed Point Arithmetic on the ARM*, document number: ARM DAI 0033A, September 1996, https://developer.arm.com/documentation/dai0033/a/
[18] Sandoval-Hernández, M. A., Velez-López, G. C., Vázquez-Leal, H., Filobello-Nino, U. A., Morales-Alarcón, G. J., De-Leo-Baquero, E., Bielma-Pérez, A.C, Sampieri-González, C.E., Pérez-Jácome-Friscione, J.E., Contreras- Hernández, A.D., Álvarez-Gasca, O., Sánchez-Orea, J., and Cuellar-Hernández, L. *Basic Implementation of Fixed-Point Arithmetic in Numerical Analysis*. International Journal of Engineering Research y Technology, 12. 01 (2023), 313-318. 2023.
[19] Uspensky, J. V., Maquieira, J. C., & Varela, J. P. *Teoría de ecuaciones*. Limusa. 1987.
[20] Horner, W. G. *A new method of solving numerical equations of all orders, by continuous approximation*. In Abstracts of the Papers Printed in the Philosophical Transactions of the Royal Society of London (No. 2, pp. 117-117). London: The Royal Society. (1833, December).
[21] T. Fukushima, *Precise and fast computation of Lambert W-functions without transcendental function evaluations*. *Journal of Computational and Applied Mathematics* 244 (2013), 77-89, 2013.
[22] Vazquez-Leal, H., Sandoval-Hernandez, M. A., Filobello-Nino, U., & Huerta-Chua, J. *The novel Leal-polynomials for the multi-expansive approximation of nonlinear differential equations*. Heliyon, 6(4), e03695. 2020.
[23] Sandoval-Hernandez, M. A., Vazquez-Leal, H., Filobello-Nino, U., & Hernandez-Martinez, L. *New handy and accurate approximation for the Gaussian integrals with applications to science and engineering*. Open Mathematics, 17(1), 1774-1793. 2019.
[24] Vazquez-Leal, H., Castaneda-Sheissa, R., Filobello-Nino, U., Sarmiento-Reyes, A., and Sanchez Orea, J. *High accurate simple approximation of normal distribution integral*. Mathematical problems in engineering, 2012.
[25] Backus, John. *The history of Fortran I, II, and III*. ACM Sigplan Notices, 165-180. 13.8 (1978):
[26] Fortran 2003– Last Working Draft. Gnu.Org. Retrieved May 10, 2014.
[27] Fortran 2018. ISO. Retrieved November 30, 2018.
[28] Adams, J. C., Brainerd, W. S., Martin, J. T., Smith, B. T., & Wagener, J. L. *Fortran 90 Handbook*. New York: McGraw-Hill, 1992.
[29] Kong, Soonho, Sicun Gao, and Edmund M. Clarke. *Floating-point Bugs in Embedded GNU C Library*. Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 2013.
[30] Stewart J., *Precálculo*, Thomson, 2012