

UNIVERSIDADE FEDERAL DE MINAS GERAIS
Escola de Engenharia
Programa de Pós-Graduação em Engenharia Elétrica

Priscila Fernanda da Silva Guedes

**EFFICIENT FOURTH-ORDER FIXED-STEP RUNGE-KUTTA DISCRETIZATION SCHEME
FOR NONLINEAR SYSTEMS IN FLOATING-POINT AND POSIT ARITHMETIC**

Belo Horizonte

2023

Priscila Fernanda da Silva Guedes

**EFFICIENT FOURTH-ORDER FIXED-STEP RUNGE-KUTTA DISCRETIZATION SCHEME
FOR NONLINEAR SYSTEMS IN FLOATING-POINT AND POSIT ARITHMETIC**

A thesis presented to the Graduate Program in Electrical Engineering (PPGEE) of the Federal University of Minas Gerais (UFMG) in partial fulfillment of the requirements to obtain the degree of Doctor in Electrical Engineering.

Advisor: Prof. Dr. Eduardo Mazoni Andrade Marçal
Mendes

Co-Advisor: Prof. Dr. Erivelton Geraldo Nepomuceno

Belo Horizonte

2023

G924e	<p>Guedes, Priscila Fernanda da Silva. Efficient fourth-order fixed-step runge-kutta discretization scheme for nonlinear systems in floating-point and posit arithmetic [recurso eletrônico] / Priscila Fernanda da Silva Guedes. - 2023. 1 recurso online (73 f. : il., color.) : pdf.</p> <p>Orientador: Eduardo Mazoni Andrade Marçal Mendes. Coorientador: Erivelton Geraldo Nepomuceno.</p> <p>Tese (doutorado) - Universidade Federal de Minas Gerais, Escola de Engenharia.</p> <p>Bibliografia: f. 66-73. Exigências do sistema: Adobe Acrobat Reader.</p> <p>1. Engenharia elétrica - Teses. 2. Sistemas dinâmicos - Teses. 3. Runge-Kutta, Fórmulas de - Teses. 4. Lorenz, Equações de - Teses. 5. Computação - Matemática - Teses. 6. Sistemas de tempo discreto - Teses. I. Mendes, Eduardo Mazoni Andrade Marçal. II. Nepomuceno, Erivelton Geraldo. III. Universidade Federal de Minas Gerais. Escola de Engenharia. IV. Título.</p> <p style="text-align: right;">CDU: 621.3(043)</p>
-------	--



UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FOLHA DE APROVAÇÃO

"EFFICIENT FOURTH-ORDER FIXED-STEP RUNGE-KUTTA DISCRETIZATION SCHEME FOR NONLINEAR SYSTEMS IN FLOATING-POINT AND POSIT ARITHMETIC"

Priscila Fernanda da Silva Guedes

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Doutor em Engenharia Elétrica. Aprovada em 02 de maio de 2023. Por:

Prof. Ph.D. Eduardo Mazoni Andrade Marçal Mendes
DELT (UFMG) - Orientador

Prof. Dr. Erivelton Geraldo Nepomuceno
Coorientador - (Depto. of Electronic Engineering
(Maynooth University - Ireland)

Prof. Dr. Danton Diego Ferreira
Departamento de Automática (UFLA)

Prof. Dr. Márcio Júnior Lacerda
Departamento de Engenharia Elétrica (UFSJ)

Prof. Dr. Janier Arias García
DELT (UFMG)

Prof. Dr. Leonardo Antônio Borges Torres
DELT (UFMG)



Documento assinado eletronicamente por **Frederico Gadelha Guimaraes, Coordenador(a) de curso de pós-graduação**, em 02/10/2023, às 18:34, conforme horário oficial de Brasília, com fundamento no art. 5º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2678398** e o código CRC **EB0150B7**.

To my family.

Acknowledgements

Initially, I'd like to thank my family for their support and understanding throughout this period.

To my advisor, Eduardo, I would like to thank you for giving me this opportunity and for your teachings, guidance and patience over these years.

I would also like to thank my co-advisor, Erivelton, for their dedication, patience and valuable contributions, which were very important in defining the direction of this work.

To my colleagues, Alesi, Caio and Pedro, thank you for the discussions and the good moments during the doctorate.

I would also like to thank the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES)* for the financial support during the doctoral period and PPGEE/UFMG.

"I would rather have questions that can't be answered than answers that can't be questioned."

Richard Feynman

Resumo

Pesquisadores têm estudado sistemas dinâmicos não lineares em diversas áreas da ciência e da engenharia. Utilizando para entender seu comportamento, descrito por equações diferenciais, um método de discretização para análise numérica. Um dos métodos mais citados e conhecidos para investigar tais sistemas é o método Runge-Kutta de quarta ordem. Embora o poder e o avanço computacional tenham crescido rapidamente nas últimas décadas, problemas embarcados e de grande escala têm motivados pesquisas significativas para melhorar a eficiência computacional.

Poucos estudos focaram na limitação da precisão finita em esquemas de discretização devido a efeitos de arredondamento na representação de números de ponto flutuante. Normalmente, os resultados do método de discretização são considerados como verdadeiros para as pesquisas, sem se importar com a limitação computacional de representar esses termos. Portanto, o primeiro resultado desta pesquisa é um esquema computacional para a discretização efetiva de sistemas dinâmicos não lineares. Usando um teorema, mostra-se que os termos de alta ordem no método Runge-Kutta de quarta ordem podem ser desprezados sem perda de precisão. Esta abordagem foi aplicada a sistemas bem conhecidos na literatura, nomeadamente o sistema Rössler, as equações de Lorenz e o sistema Sprott B. Observou-se uma redução significativa no número de operações matemáticas e no tempo de simulação desses sistemas, mantendo a precisão, a observabilidade dos sistemas dinâmicos e o Maior Expoente de Lyapunov.

Posteriormente, houve a necessidade de aplicar o resultado apresentado a uma outra computação. Ou, de maneira mais geral, aritmética de número universal, que é uma alternativa à aritmética de ponto flutuante. Portanto, o teorema aplicado à aritmética de ponto flutuante foi alterado com a precisão da aritmética de Posit para obter o resultado efetivo da discretização ao usar a aritmética de Posit. Da mesma forma, observou-se uma redução significativa no número de operações matemáticas, com redução de 98,57% das operações para as equações de Lorenz, preservando as características dos sistemas.

Além disso, com a crescente preocupação com os fatores climáticos, principalmente a pegada de

carbono, tornou-se imperativo desenvolver algoritmos mais eficientes que possam resolver problemas com uma pegada de carbono menor. Com o teorema proposto aplicado ao método de discretização de Runge-Kutta de quarta ordem para obter uma discretização efetiva resultando em uma redução dos monômios, dos números de operações e do o tempo de simulação, há uma redução significativa na sua pegada de carbono, sem sacrificar as características dos sistemas. Em particular, as equações de Lorenz obtiveram uma redução de aproximadamente 99% na pegada de carbono utilizando a aritmética Posit.

Palavras-chave: Sistema Dinâmicos não-lineares, Discretização, Método de Runge-Kutta de Quarta Ordem.

Abstract

Researchers have been studying nonlinear dynamical systems in several fields of science and engineering. Using to understand their behavior, described by differential equations, a discretization method for numerical analysis. One of the most widely cited and well-known methods for investigating such systems is the fourth-order Runge-Kutta method. Although computational power and advancement have grown rapidly in recent decades, large-scale and embedded problems have motivated significant research to improve computational efficiency.

Few studies have focused on the limitation of finite precision in discretization schemes due to rounding effects in the representation of floating-point numbers. Typically, discretization method results are considering as true for the researches, without caring about the computational limitation of representing these terms. Therefore, the first result of this research is a computational scheme for the effective discretization of nonlinear dynamic systems. Using a theorem, it is shown that high-order terms in the fourth-order Runge-Kutta method can be neglected without loss of precision. This approach was applied to well-known systems in the literature, namely the Rössler system, the Lorenz equations, and the Sprott B system. A significant reduction was observed in the number of mathematical operations and simulation time of these systems while maintaining the accuracy, observability of dynamical systems, and the Largest Lyapunov Exponent.

Subsequently, there was a need to apply the presented result to another arithmetic computation known as Posit arithmetic. Or more generally, universal number arithmetic, which is an alternative to floating-point arithmetic. Therefore, the theorem applied to floating-point arithmetic was altered with the precision of Posit arithmetic to obtain the effective result of discretization when using Posit arithmetic. Similarly, a significant reduction was observed in the number of mathematical operations, with a reduction of 98.57% of operations for the Lorenz equations, while preserving the characteristics of the systems.

Furthermore, with the growing concern about climate factors, particularly the carbon footprint, it has become imperative to develop more efficient algorithms that can solve problems with a smaller carbon footprint. With the proposed theorem applied to the fourth-order Runge-Kutta discretization method to obtain an effective discretization resulting in a reduction of the monomials, the number of operations, and the simulation time, there is a significant reduction in its carbon footprint without sacrificing system features. In particular, the Lorenz equations achieved an approximately 99% reduction in carbon footprint using Posit arithmetic.

Keywords: Nonlinear Dynamical Systems, Discretization, Fourth Order Runge-Kutta Method.

List of Symbols

x_{k+1}	Recursive function;
\mathbb{N}	Set of natural numbers;
\in	It is an element of;
\mathbb{R}	Set of real numbers;
\mathbb{Z}_+	Set of non-negative integer numbers;
\mathbb{R}_+	Set of non-negative real numbers;
x_k	Output obtained from iteration k of a map;
$\hat{x}_{i,k}$	Element of the pseudo-orbit of i , corresponding to iteration k ;
I	Interval;
\pm	More or less;
S	Significant or Mantissa;
\times	Indicates multiplication;
B	Adopted basis;
E	Exponent;
b_0	Hidden bit;
b_n	Bits of the signifier;
p	System accuracy;
ϵ	Machine epsilon;
$\ell_{\Omega,k}$	Lower Bound Error;
s	sign.

List of Abbreviations

UFMG	Universidade Federal de Minas Gerais.
IEEE	Institute of Electrical and Electronics Engineers.
IEEE 754	Standard for Binary Floating-Point Arithmetic.
LLE	Largest Lyapunov Exponent.
LBE	Lower Bound Error.
RK4	Fourth Order Runge-Kutta.

Contents

1	Introduction	16
1.1	Motivation	18
1.2	Objectives	19
1.3	Thesis outline and contributions	19
2	Theoretical Foundation	20
2.1	Dynamic Systems	21
2.1.1	Continuous Dynamic Systems	21
2.1.2	Discrete Dynamic Systems	21
2.1.3	Recursive Functions	21
2.1.4	Interval Extensions	21
2.1.5	Orbits and Pseudo-Orbits	22
2.1.6	Lyapunov Exponent	23
2.1.6.1	Wolf's Algorithm	23
2.1.6.2	Rosenstein's Algorithm	24
2.1.6.3	Mendes and Nepomuceno's Algorithm	24
2.2	Observability of Dynamical Systems	26
2.3	Fourth-Order Runge-Kutta Method	28
2.4	Arithmetic Computing	31
2.4.1	IEEE 754 Floating Point Standard	32
2.4.1.1	Representation of Numbers on the Computer	32
2.4.1.2	Computational Representation in Different Formats of the IEEE 754 Standard	33

2.4.1.3	Accuracy of a Floating Point System	33
2.4.1.4	Rounding	34
2.4.1.5	Computational Errors	35
2.4.2	Universal Number (<i>Unum</i>)	35
2.4.2.1	<i>Unum</i> Type I and Type II	35
2.4.2.2	Posit	36
2.5	Green Algorithm	38
3	Simulation of continuous-time systems using RK4 and floating point arithmetic (IEEE - 754)	39
3.1	Rössler System	44
3.2	Lorenz Equations	50
3.3	Sprott B	52
3.4	Discussion	54
4	Simulation of continuous-time systems using RK4 and Posit arithmetic	56
4.1	Rössler System	58
4.2	Lorenz Equations	60
4.3	Sprott B	62
4.4	Discussion	63
5	Conclusion	64
	Bibliography	66

CHAPTER 1

Introduction

Most real systems can be described by nonlinear continuous-time models, that in turn, may be written as a set of differential equations. For a better computational analysis and understanding of their behavior, discretization methods that transform these continuous-time systems into discrete-time systems are required. Once they are fully determined, they are almost always used in computational systems [1–6]. In the literature there are several methods of discretization, some better known and others not so much. Methods based on the expansion of the Taylor series, such as Euler, Heunn and Runge-Kutta [7, 8] are among the most known methods, with the fourth-order Runge-Kutta being the most cited in the literature [9–14].

In Euler’s method there is a truncation of the Taylor series after the first term [7]. Heunn’s method, when compared to Euler’s method, performs an extra evaluation of the function, that is, it performs a second-order approximation, obtaining more accurate results than Euler’s method [7]. One of the most used numerical integration methods is the fourth-order Runge-Kutta, in which there are four evaluations of the function in each step [7, 8]. Other methods use information from the previous steps to obtain higher-order approximations, notably the Adams – Bashforth and Adams – Bashforth – Moulton methods [7, 15]. There are also non-standard methods, such as the Monaco and Normand-Cyrot method, which is based on representing the solution of the system in terms of the Lie exponential expansion [16] and the Mickens method that uses finite differences in building the model [17].

Numerical methods for nonlinear dynamical systems are expected to be accurate and efficient for most or preferably all solutions [18]; otherwise, a technique is unreliable and therefore unlikely to be used in applications, even if any alternative is not as accurate and efficient [18]. In other words, a method must be robust.

There is no doubt that great advances have been made in the simulation of nonlinear dynamical systems. However, concerns about finite precision effects are still present in the literature [4, 5]. Computers

have properties and characteristics that result in inaccurate numerical simulations. Nepomuceno [19] shows that the logistic map simulation may not converge to the exact fixed point due to a computer limitation. This limitation comes from the finite representation of real numbers, as computers are not capable of representing all real numbers. If an initial condition is represented, after some mathematical operations, the result may no longer be representative, so the computer rounds off each computational step, accumulating the error in the outcome [4]. Similar results can be found in [20], where rounding errors are under investigation in the simulations, but using a method based on range extension. Nepomuceno et al. [21] used the lower bound error to simulate a chaotic system with lower and upper bounds. It has been demonstrated that the widths of these limits do not diverge, which is an advantage compared to other techniques based on arithmetic intervals [22]. Several works have been dedicated to the investigation of the finite precision effects of the computer [4, 5, 19, 23–27].

With the advancement of research, new ideas emerge as a replacement for existing ones or simply to complement the ideas already presented. The same happened with floating-point arithmetic; which is governed by the IEEE (Institute of Electrical Engineering and Electronics) 754 standard for floating-point arithmetic [28, 29], which presents rules and standards to be followed by computer and software manufacturers in the computation treatment of floating point numbers [30].

Gustafson [31] developed an arithmetic as a replacement for floating-point arithmetic, called universal number (Unum). In particular, Posit - Unum type 3 is claimed as a possible replacement for floating point arithmetic. Posit arithmetic offers some benefits over floating point numbers, including better dynamic range and accuracy in the same bit field and more accurate arithmetic calculations [32]. As it is a new arithmetic, some studies using Posit compare their results with those presented by the IEEE. In [33] there was a comparison of the precision representation, dynamic range and performance of implemented Posit FPU (Floating-Point Unit) with IEEE 754 floating-point IP cores. Posit exhibits superiority in precision representation and dynamic range than IEEE 754, and through further optimization of the implementation, Posit can be a good candidate for floating-point IP cores. In [34] a comparative study of the robustness of the 32-bit representations for Posit and IEEE-754 for machine learning was carried out. It was shown that in 100% of the machine learning applications tested, the accuracy of the implemented systems in Posit it is greater than in floating point. In [35], it was shown, by studying a neural networks, that the use of Posit numbers allowed for the training of more complex neural networks while still achieving similar precisions to floating point format.

The advancement of science has brought forth new developments, both in the form of new tools and technologies, such as Posit, as well as new concerns. One of these concerns is the issue of climate change.

Numerous human activities, such as data centers and other sources of large-scale computing, contribute significantly to greenhouse gas emissions, making it imperative to find ways to reduce these emissions. In [36], software was developed to calculate the emission of greenhouse gases, without altering the algorithms of the simulations, by estimating the emission based on the characteristics of the simulations and the simulation time.

Quantitative estimates of the carbon footprint of urban ecosystems are crucial for developing low-carbon policies to mitigate climate change. In [37], a model of the carbon footprint in conceptual design based on indeterminate numbers was presented, demonstrating the effectiveness of carbon footprint calculation. The authors of [38] reviewed over 195 studies related to urban carbon footprint and carbon footprint. The ongoing problem of greenhouse gas emissions is a major concern in light of the crisis of climate change [39, 40]. The impact of human activities on these emissions, particularly in the field of large-scale computing, is significant and calls for immediate action to mitigate it. While various efforts have been made to reduce these emissions, there is still a lack of understanding on how computational precision impacts the carbon footprint of simulations [41–43].

In most articles that use discretization methods, the authors seem to assume their results are computationally sound without worrying about limitations of computers. Therefore, to fill this gap, in this work an analysis of the discretization results will be performed taking into account the computer's accuracy in floating point arithmetic. With the advancement of research and knowledge about Posit arithmetic, the same analysis of the discretization of systems will be performed, including computational accuracy and reduction of the carbon footprint.

1.1 Motivation

As a motivation, the initial step of this research was then to analyze the simulation of continuous-time systems. Due to characteristics of nonlinear continuous-time chaotic systems, they were used as examples. For the simulation of these systems, their discretization was performed using methods based on the Taylor series. In the discretization process using a well-known simulation method, the fourth-order Runge-Kutta method, a large number of monomials was created and because of that it was decided to focus the study on this method.

The discretization by the fourth order Runge-Kutta method presented a large amount of monomials as an step-size raised to high numbers depending on the system under analysis. It was arbitrarily decided to exclude monomials that presented an step-size raised to powers greater than seven, and as the results

obtained it was verified that the characteristics of the systems remained the same, with this, it was concluded that the computer in the calculations performed did not use these monomials. Therefore, from this initial analysis it was possible to define the objectives of this research.

1.2 Objectives

Obtain an effective discretization of nonlinear dynamical systems when discretized by the fourth-order Runge-Kutta method. Effective because it is possible to exclude some monomials based on the precision of the computational arithmetic used, and yet the characteristics of continuous systems are preserved.

The characteristics analyzed were the attractor in the xy plane, the observability and the Lyapunov exponent. In addition to comparing quantitative characteristics, such as the number of monomials for each equation, the number of mathematical operations performed in the simulation at each iteration, simulation time, as well as the reduction in the carbon footprint.

1.3 Thesis outline and contributions

The Thesis's organization and the related contributions of each chapter are:

Chapter 2 presents an overview of the main concepts used in the Thesis, such as dynamical systems, Lyapunov exponent, observability, fourth-order Runge-Kutta and computational arithmetic.

Chapter 3 shows the theorem for obtaining an effective discretization for floating point arithmetic. The results was published in the journal paper entitled "Effective computational discretization scheme for nonlinear dynamical systems" [44].

Chapter 4 derives the results for Posit arithmetic, with the change in the precision value. The article "Posit Number Impact on Carbon Footprint Reduction in Simulation of NonLinear Dynamical Systems" was submitted for publication in a journal IEEE Access [45].

Finally, Chapter 5 presents concluding remarks and suggestions for future works.

CHAPTER 2

Theoretical Foundation

This chapter brings the theoretical foundation that supports the studies carried out in this thesis. First, the study of dynamic systems is presented, exposing concepts of recursive functions, orbits and pseudo-orbits, as well as the calculation of the Largest Lyapunov exponent. The fourth-order Runge-Kutta method is also portrayed as a discretization method. Finally, concepts of computational arithmetic are presented, both floating point and Posit.

The diagram of Figure 2.1 shows how these concepts will contribute to the discussion in the next sections.

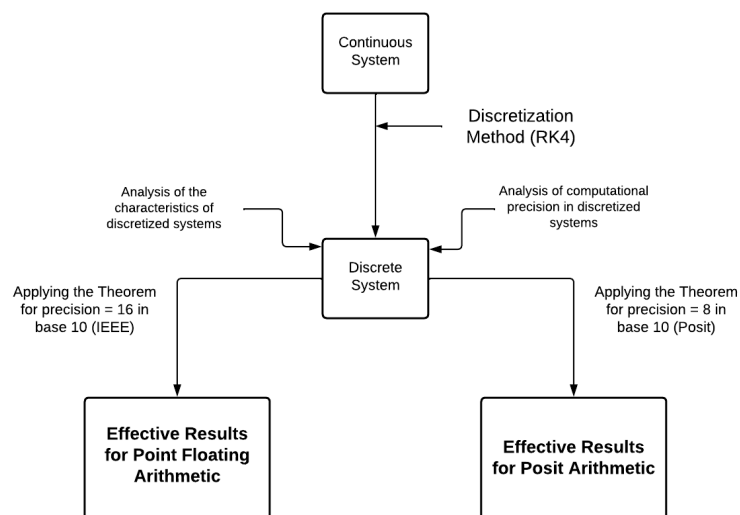


Figure 2.1: Diagram of the study.

In words, there is a continuous system under study that needs to be discretized for computational simulation. With the discretized system (discrete system), all sorts of analyses can be performed such as stability, controllability, observability, the number of monomials generated during the discretization process, their computational precision and the exponents of each variable. Taking these aspects into

account, a theorem is proposed considering the accuracy and characteristics of the computer so that there is an effective discretization.

2.1 Dynamic Systems

According to Monteiro [46], a system can be defined as a set of objects brought together by some interrelationship, with a cause and effect relationship existing between the elements of this set. When the magnitudes that quantify this system vary over time, we have the concept of a dynamic system.

2.1.1 Continuous Dynamic Systems

Continuous-time dynamical systems are governed by differential equations. A system is continuous-time if time t is a real number. Normally, $t \in \mathbb{R}_+$ is considered, that is, t is a non-negative real number [46].

2.1.2 Discrete Dynamic Systems

A discrete-time system is governed by difference equations. A system is discrete-time when time t is an integer. Generally, $t \in \mathbb{Z}_+$ is considered, that is, t is a non-negative integer [46].

2.1.3 Recursive Functions

Let $k \in \mathbb{N}$, $M \subset \mathbb{R}$ be a metric space, the relation

$$x_{k+1} = f(x_k), \quad (2.1)$$

where $f : M \rightarrow M$ is the recursive function defined in state space and x_k denotes the state in discrete time k [47].

2.1.4 Interval Extensions

The intervals are normally represented by capital letters [22]. The lower and upper bounds of an interval X are denoted by \underline{X} and \overline{X} , respectively.

$$X = [\underline{X}, \overline{X}]. \quad (2.2)$$

For a function f defined for a real variable x , Moore and Bierbaum [48] provided the following definition.

Definition 1. *An interval extension of f is an interval function F defined to map an interval variable X , such that for real arguments one has*

$$F(x) = f(x). \quad (2.3)$$

Definition 2 ([49]). *An interval function $F : \mathbb{IR} \rightarrow \mathbb{IR}$ (\mathbb{IR} is the real interval) represents the real function $f : \mathbb{R} \rightarrow \mathbb{R}$ when for all $x \in [a, b]$ we have $f(x) \in F([a, b])$.*

Arithmetic operations with intervals will follow the rules described by Moore and Bierbaum [48].

Example 1 ([50]). *Some interval extensions of $f(x) = rx(1 - x)$ can be given by:*

$$\begin{aligned} F(X) &= rX(1 - X) \\ G(X) &= r(X(1 - X)) \\ H(X) &= rX - rX^2 \end{aligned} \quad (2.4)$$

Consider $r = 3$ and $X = [0.3, 0.4]$, then we have:

$$\begin{aligned} F([0.3, 0.4]) &= 3[0.3, 0.4](1 - [0.3, 0.4]) = [0.54, 0.84], \\ G([0.3, 0.4]) &= 3([0.3, 0.4](1 - [0.3, 0.4])) = [0.54, 0.84], \\ H([0.3, 0.4]) &= 3[0.3, 0.4] - 3([0.3, 0.4]^2) = [0.42, 0.93]. \end{aligned}$$

It is noteworthy that the equations in the set (2.4) are mathematically equivalent, but have different sequences of elementary operations [51].

2.1.5 Orbits and Pseudo-Orbits

The sequence defined by $\{x_0, x_1, x_2, \dots, x_k\}$ obtained at each iteration of a function is called the orbit of x_0 [52].

However, when a computer is used to calculate the results at each iteration of a map, we have a pseudo-orbit. That is, an approximation of the true orbit due to computer limitations. A specific pseudo-orbit $i \in \mathbb{N}$ is represented by $\{\hat{x}_{i,0}, \hat{x}_{i,1}, \dots, \hat{x}_{i,k}\}$, such that,

$$|x_k - \hat{x}_{i,k}| \leq \delta_{i,k}, \quad (2.5)$$

where $\delta_{i,k} \in \mathbb{R}$ is an error and $\delta_{i,k} \geq 0$. Then, an interval associated with each value of the pseudo-orbit is defined,

$$I_{i,k} = [\hat{x}_{i,k} - \delta_{i,k}, \hat{x}_{i,k} + \delta_{i,k}]. \quad (2.6)$$

From the inequation (2.5) and equation (2.6) it is evident that

$$x_k \in I_{i,k} \quad \text{for all } i \in \mathbb{N}. \quad (2.7)$$

2.1.6 Lyapunov Exponent

The Lyapunov exponents measure the average divergence or convergence of nearby trajectories along certain directions in state space. In chaotic systems the states of two copies of the same system will separate exponentially with time despite very similar initial conditions [53].

The calculation of the Largest Lyapunov Exponent is considered one of the best methods for detecting chaos in dynamic systems, because in addition to presenting a simpler and more accessible methodology in relation to other indicator methods, it can be used to quantify how chaotic is a dynamic system, so that the higher the exponent found, the more chaotic and less predictable is the system under analysis [54].

2.1.6.1 Wolf's Algorithm

Wolf et al. [55] was the forerunner in investigating the exponential divergence of trajectories obtained from very close initial conditions with the aim of calculating the Largest Lyapunov Exponent.

According to Wolf et al. [55], consider an unknown time series $\{x(t)\}$ with N samples and its reconstructed version in a space of dimension d . If τ is the time delay used for reconstruction, then each trajectory point is a d -dimensional vector of the form $\vec{x}(t) = \{x(t), x(t + \tau), x(t + 2\tau), \dots, x(t + (d - 1)\tau)\}$. If we select a point \vec{x}_0 that represents the state of the system at t_0 and identify its closest spatial neighbour \vec{x}_n , then, the high density of the system trajectories in the phase space allows us to make the assumption that these points belong to different neighbouring trajectories. Let us denote the distance $\|\vec{x}_0 - \vec{x}_n\|$ as $L(t_0)$. If we consider these points as the initial conditions of two trajectories and let the system evolve in time, then, at time t_1 the distance between the new trajectory points \vec{x}'_0 and \vec{x}'_n will be $L'(t_1) = \|\vec{x}'_0 - \vec{x}'_n\|$. At this point, the trajectory point \vec{x}'_n is substituted by another point \vec{x}'_m such that the distance $L(t_1) = \|\vec{x}'_0 - \vec{x}'_m\|$ is less than $L'(t_1)$.

If the algorithm reaches the last point of the trajectory, then the maximum largest Lyapunov exponent, λ_1 , is calculated by the equation:

$$\lambda_1 = \frac{1}{t_M - t_0} \sum_{k=1}^M \log_2 \frac{L'(t_k)}{L(t_{k-1})} \quad (2.8)$$

where t_M is the time of the last substitution and M the number of substitutions.

2.1.6.2 Rosenstein's Algorithm

Rosenstein et al. [54] developed an algorithm based on the method by Wolf et al. [55]. The algorithm is similar to that of Kantz [56], with the distance between the trajectories is defined as the Euclidean distance in the space of reconstructed states, also using only a neighboring trajectory. Kantz [56] mentions that when using only one neighbor, within a neighborhood of points, can induce statistical errors in noisy signals.

In Rosenstein's algorithm, the Lyapunov exponent is estimated as:

$$\lambda_1(i) = \frac{1}{i\Delta t} \frac{1}{M-i} \sum_{j=1}^{M-i} \ln \frac{d_j(i)}{d_j(0)}, \quad (2.9)$$

where $M = N - (m - 1)J$, J is the lag, m is the embedding dimension, N is the number of point of the serie, Δt is the sampling period of the time series and $d_j(i)$ is the distance between the j -th pair of nearest neighbors after i discrete-time steps.

2.1.6.3 Mendes and Nepomuceno's Algorithm

Lower Bound Error - LBE

The lower bound error was introduced by Nepomuceno and Martins [50] and improved in Nepomuceno et al. [57]. It is based on the fact that interval extensions are mathematically equivalent, but may present different results in computer simulations, due to the representation of real numbers as floating point numbers, as evidenced in Figure 2.2.

Example 2. Consider the Hénon map described by the equations:

$$x_{k+1} = 1 - ax_k^2 + y_k, \quad (2.10)$$

$$y_{k+1} = bx_k. \quad (2.11)$$

Let be the following interval extensions:

$$F(X_k) = 1 - aX_k^2 + Y_k \quad (2.12)$$

$$G(X_k) = 1 - aX_kX_k + Y_k. \quad (2.13)$$

If $a = 1.4$, $b = 0.3$, $x_0 = 0.3$ and $y_0 = 0.3$.

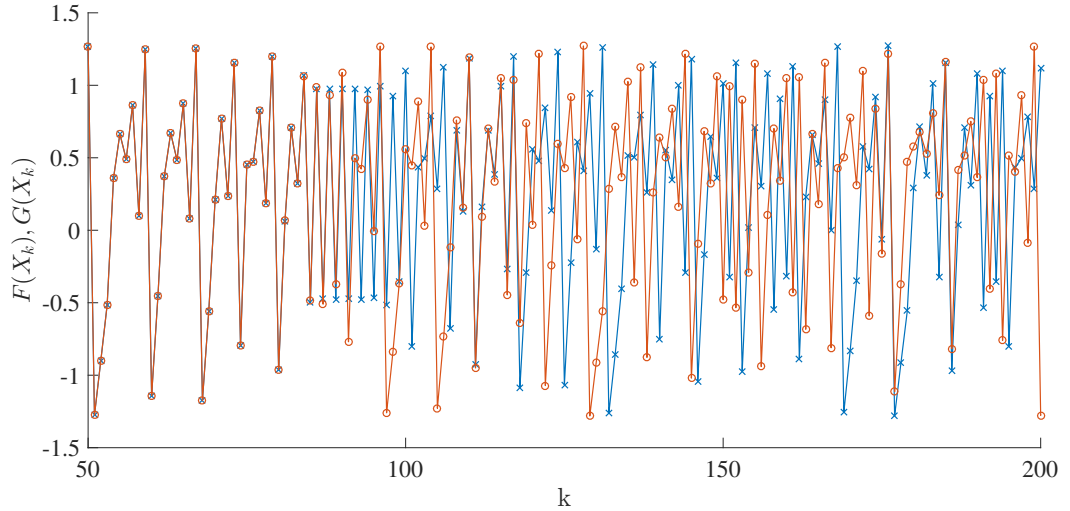


Figure 2.2: Free Simulation of the Hénon Map, with $a = 1.4$, $b = 0.3$, $x_0 = 0.3$ and $y_0 = 0.3$, for two different extensions, (-x-) for $F(X_k)$ and (-o-) for $G(X_k)$.

The interval extensions represented by Equations (2.12) and (2.13) are mathematically equivalent, but represent a different sequence of operations. This causes the accumulated error at each iteration to be different, as a result, after a few iterations, the interval extensions diverge.

The definition of LBE is presented in Theorem 1.

Theorem 1. Let two pseudo-orbits $\hat{x}_{a,n}$ and $\hat{x}_{b,n}$ derived from two interval extensions. Let $\delta_{\alpha,n} = |\hat{x}_{a,n} - \hat{x}_{b,n}|/2$ be the lower bound error of a map $f(x)$, then $\delta_{a,n} \geq \delta_{\alpha,n}$ or $\delta_{b,n} \geq \delta_{\alpha,n}$.

Theorem 1 states that at least one of the two pseudo-orbits must have an error greater than or equal to the lower bound of the error. If the LBE is greater than the required accuracy, the simulation must be carefully analyzed. So, to guarantee the reliability of the simulations, it must be proved that at least one of the pseudo-orbits has the necessary accuracy.

In Nepomuceno et al. [57], the LBE concept was refined. The concept of the associative property of multiplication was used to determine the natural interval extensions. Thus, the LBE was defined as follows:

Theorem 2. *Let $\{\hat{x}_{a,n}\}$ and $\{\hat{x}_{b,n}\}$ be two pseudo-orbits derived from two arithmetic interval extensions. Let $\ell_{\Omega,n} = |\hat{x}_{a,n} - \hat{x}_{b,n}|/2$ be the lower bound error associated to the set of pseudo-orbits $\Omega = [\{\hat{x}_{a,n}\}, \{\hat{x}_{b,n}\}]$ of a map, then $\gamma_{a,n} = \gamma_{b,n} \geq \ell_{\Omega,n}$.*

Calculating the Largest Lyapunov Exponent

The method proposed by Mendes and Nepomuceno [53] uses the concept of lower bound error (LBE) and consists of the following steps:

- Choose two interval extensions from the system under investigation.
- With the same initial condition, the same step-size and the same discretization scheme, simulate the two interval extensions.
- Compute the LBE.
- Use the method of least squares to fit a line to the slope of the natural logarithm of the lower bound error. The slope of the line is the LLE.

In the example 2, we have the simulation of the two interval extensions. So, based on this example, Figure 2.3 shows the LBE curve for the Henon map and the associated Lyapunov exponent; the value of the Lyapunov exponent found was 0.4062 nat/k (nat is a unit of information, based on natural logarithms) and is in agreement with that found in the literature [58]. The Lyapunov exponent was calculated using the method proposed by Mendes and Nepomuceno [53], in which a straight line is fitted to the LBE curve, so that the slope of the line represents the Lyapunov exponent and the independent term represents the maximum precision of the simulated system. The line is empirically adjusted, usually until the simulation precision is lost.

2.2 Observability of Dynamical Systems

In the Thesis, the examples focus on systems with three dimensions, then the concept of observability will be developed for these systems.

Let be the dynamical system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \quad r = g(\mathbf{x}), \quad (2.14)$$

where $\mathbf{x} \in \mathbb{R}^3$ is the state vector, \mathbf{f} is nonlinear vector field and r is the observable acquired through the measurement function $g : \mathbb{R}^3 \mapsto \mathbb{R}$.

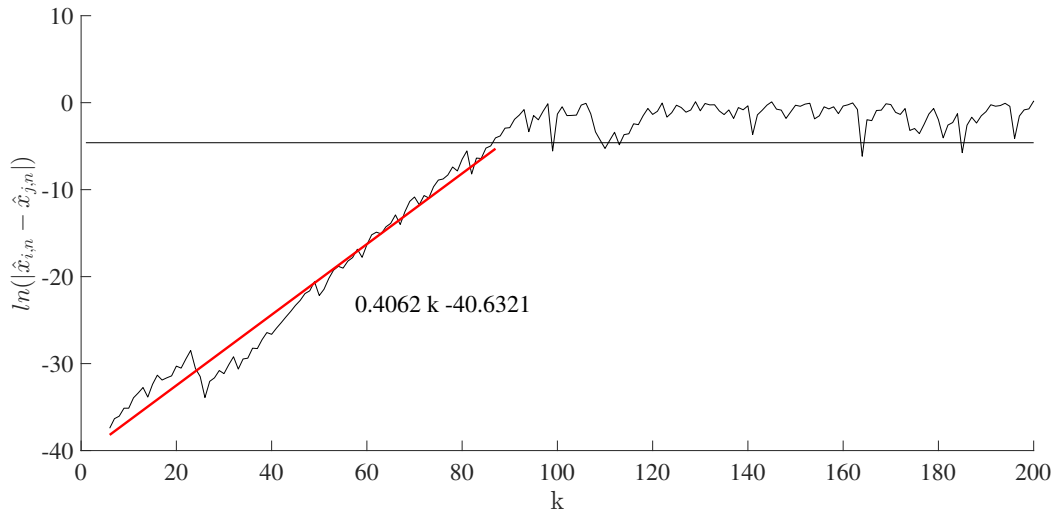


Figure 2.3: Evolution of the LBE of the Hénon Map and the associated Lyapunov exponent. Where the x axis represents the iterations and the y axis, the natural logarithm (\ln) of the LBE.

The portrait that is reproduced may be spanned by the derivative coordinates, such as

$$X = r, \quad Y = \dot{r}, \quad W = \ddot{r}. \quad (2.15)$$

It is possible to define a coordinate transformation Φ , when original states (x,y,w) and derivative coordinates (X,Y,W) are related. If $r = x$, then the transformation Φ is equivalent to

$$X = r, \quad Y = f_r, \quad W = \frac{\partial f_r}{\partial x} f_x + \frac{\partial f_r}{\partial y} f_y + \frac{\partial f_r}{\partial w} f_w, \quad (2.16)$$

where f_x , f_y , and f_w are the components of \mathbf{f} [59, 60].

The system can be rewritten as an explicit system

$$\dot{X} = Y, \quad \dot{Y} = W, \quad \dot{W} = F_r(X, Y, W), \quad (2.17)$$

where $F_r(X, Y, W)$ is the jerk equation [60–62]. For a given original system, the jerk equation F_r can be analytically derived using the coordinate transformation Φ [63].

Example 3. Let the Rössler system be represented by:

$$\begin{cases} \dot{x} = -y - z, \\ \dot{y} = x + ay, \\ \dot{z} = b + z(x - c). \end{cases}$$

And Φ_y , the transformation coordinate of the variable y , built from equations (2.15).

$$\Phi_y = \begin{cases} X = y, \\ Y = \dot{y} = ay + x, \\ Z = a\dot{y} + \dot{x} = -y - z + ax + a^2y. \end{cases} \rightarrow \Phi_y = \begin{cases} y = X, \\ x = Y - ax, \\ z = -X + aY - Z. \end{cases} \quad (2.18)$$

$$F_r = -y - z + ax + a^2y.$$

The jerk equation for variable y .

$$\begin{aligned} F_y &= \frac{\partial F_r}{\partial x} f_x + \frac{\partial F_r}{\partial y} f_y + \frac{\partial F_r}{\partial z} f_z \\ &= a(-y - z) + (a^2 - 1)(ay + x) - (b + z(x - c)) \\ &= aZ - Y - b + XY - aX^2 - cX - aY^2 + a^2XY + acY + YZ - aXZ - cZ. \end{aligned}$$

The same is done with the variables x and z .

2.3 Fourth-Order Runge-Kutta Method

Runge-Kutta methods are a family of implicit and explicit iterative methods used as approximations for solutions of ordinary differential equations [7].

An implicit Runge-Kutta method has the form:

$$x_{k+1} = x_k + h \sum_{i=1}^m b_i k_i, \quad (2.19)$$

where

$$k_i = f \left(t_k + c_i h, x_k + h \sum_{j=1}^m a_{i,j} k_j \right), \quad i = 1, 2, \dots, m.$$

where $a_{i,j}$, b_i and c_i are constants of the numerical scheme.

The difference for the implicit method is that in the explicit method the summation in j in this expression goes up to $i - 1$.

Fourth-order Runge Kutta (RK4) method is one of the most used and well known [8, 64] methods. Consider the initial value problem determined by:

$$\dot{x} = f(t, x), \quad x(t_0) = x_0, \quad (2.20)$$

where x is the state variable. Let the integration step $h > 0$ in RK4 be expressed by:

$$x_{k+1} = x_k + \frac{h}{6} (k_1 + 2k_2 + 2k_3 + k_4), \quad (2.21)$$

where

$$\begin{aligned} k_1 &= f_k, \\ k_2 &= f\left(t_k + \frac{h}{2}, x_k + \frac{h}{2}k_1\right), \\ k_3 &= f\left(t_k + \frac{h}{2}, x_k + \frac{h}{2}k_2\right), \\ k_4 &= f(t_{k+1}, x_k + hk_3). \end{aligned} \quad (2.22)$$

For systems that do not depend on time explicitly, a discrete model can be directly written from the continuous counterpart. Therefore, the equations (2.22) can be rewritten as follows, which will be used in this thesis.

$$\begin{aligned} k_1 &= f_k, \\ k_2 &= f\left(x_k + \frac{h}{2}k_1\right), \\ k_3 &= f\left(x_k + \frac{h}{2}k_2\right), \\ k_4 &= f(x_k + hk_3). \end{aligned} \quad (2.23)$$

Example 4. Consider the discretization of the following system:

$$\begin{cases} \dot{x} = y, \\ \dot{y} = -x + y. \end{cases} \quad (2.24)$$

For the variable x we have:

$$\begin{aligned} k_{1_x} &= f(\dot{x}) = y, \\ k_{2_x} &= f\left(\dot{x} + \frac{h}{2}\dot{y}\right) = y + \frac{1}{2}hy - \frac{1}{2}hx, \\ k_{3_x} &= f\left(\dot{x} + \frac{h}{2}\dot{y} + \frac{h^2}{4}\ddot{y} - \frac{h^2}{4}\ddot{x}\right) = y - \frac{1}{2}hx + \frac{1}{2}hy - \frac{1}{4}h^2x, \\ k_{4_x} &= f\left(\dot{x} + h\dot{y} - \frac{h^2}{2}\ddot{x} + \frac{h^2}{2}\ddot{y} - \frac{h^3}{4}\dddot{x}\right) = y - hx + hy - \frac{1}{2}h^2x - \frac{1}{4}h^3y. \end{aligned}$$

Then,

$$x_{k+1} = x_k + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) = x_k + hy_k + \frac{1}{2}h^2y_k - \frac{1}{2}h^2x_k - \frac{1}{6}h^3x_k - \frac{1}{24}h^4y_k. \quad (2.25)$$

Likewise, for the variable y we have:

$$\begin{aligned} k_{1_y} &= f(\dot{y}) = -x + y, \\ k_{2_y} &= f\left(\dot{y} + \frac{h}{2}(-\dot{x} + \dot{y})\right) = -x + y - \frac{1}{2}hx, \\ k_{3_y} &= f\left(\dot{y} + \frac{h}{2}(-\dot{x} + \dot{y} - \frac{h}{2}\ddot{x})\right) = -x + y - \frac{1}{2}hx - \frac{1}{4}h^2y, \\ k_{4_y} &= f\left(\dot{y} + h(-\dot{x} + \dot{y} - \frac{h}{2}\ddot{x} - \frac{h^2}{4}\ddot{y})\right) = -x + y - hx - \frac{1}{2}h^2y + \frac{1}{4}h^3x - \frac{1}{4}h^3y. \end{aligned}$$

Then,

$$y_{k+1} = y_k + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) = y_k - hx_k + hy_k - \frac{1}{2}h^2x_k - \frac{1}{6}h^3y_k + \frac{1}{24}h^4x_k - \frac{1}{24}h^4y_k. \quad (2.26)$$

Therefore, the discretization of the continuous-time system using the RK4 method can be given by

$$\begin{cases} x_{k+1} = x_k + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) = x_k + hy_k + \frac{1}{2}h^2y_k - \frac{1}{2}h^2x_k - \frac{1}{6}h^3x_k - \frac{1}{24}h^4y_k, \\ y_{k+1} = y_k + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) = y_k - hx_k + hy_k - \frac{1}{2}h^2x_k - \frac{1}{6}h^3y_k + \frac{1}{24}h^4x_k - \frac{1}{24}h^4y_k. \end{cases} \quad (2.27)$$

By iterating Eq. (2.27), the Runge-Kutta solution of the system in Eq. (2.24) is obtained. Note that the discrete model is rather complex when compared to the continuous counterpart.

In the next example, a non-linearity in the variable y dynamics will be considered.

Example 5. Consider the following system:

$$\begin{cases} \dot{x} = y, \\ \dot{y} = x^2 + y. \end{cases} \quad (2.28)$$

Applying Eq. (2.21) and Eq. (2.23) to Eq. (2.28) yields to the following nonlinear discrete Runge-Kutta model.

$$\begin{cases} x_{k+1} = \frac{(x_k^2 + y_k)^2 h^6}{96} + \frac{(4x_k^2 y_k + 6y_k^2) h^5}{96} + \frac{(8y_k^2 + (16x_k + 4)y_k + 8x_k^3 + 4x_k^2) h^4}{96} + \\ + \frac{((32x_k + 16)y_k + 16x_k^2) h^3}{96} + \frac{(48x_k^2 + 48y_k) h^2}{96} + h y_k + x_k \\ y_{k+1} = \frac{h^9 y_k^4}{384} + \frac{((8x_k + 4)y_k^3 + 4x_k^2 y_k^2) h^8}{384} + \\ + \frac{(8y_k^3 + (24x_k^2 + 16x_k + 4)y_k^2 + (16x_k^3 + 8x_k^2)y_k + 4x_k^4) h^7}{384} + \\ + \frac{(16y_k^3 + (32x_k + 20)y_k^2 + (32x_k^3 + 40x_k^2)y_k + 20x_k^4) h^6}{384} + \\ + \frac{((80x_k + 80)y_k^2 + 96x_k^2 y_k + 24x_k^4) h^5}{384} + \\ + \frac{(128y_k^2 + (160x_k^2 + 96x_k + 16)y_k + 64x_k^3 + 16x_k^2) h^4}{384} + \\ + \frac{(128y_k^2 + (256x_k + 64)y_k + 128x_k^3 + 64x_k^2) h^3}{384} + \\ + \frac{((384x_k + 192)y_k + 192x_k^2) h^2}{384} + \frac{(384x_k^2 + 384y_k) h}{384} + y_k \end{cases} \quad (2.29)$$

It is important to point out that the variable x presented a linear component, but with the discretization several non-linear components were introduced. For the variable y that had only one non-linear component, other non-linear components were introduced. Furthermore, comparing the continuous system and the discrete system, it can be stated that the discrete system is more complex, taking into account the amount of monomials and nonlinearities.

2.4 Arithmetic Computing

Most numerical problems are solved using computational tools. However, computers are not able to represent all real numbers, resulting in divergent results. Therefore, this section presents some essential points of the IEEE 754 floating point standard [28], responsible for guaranteeing standards for the representation and operations with real numbers in computers.

2.4.1 IEEE 754 Floating Point Standard

Floating point arithmetic have been used in computers since the mid-1950s. From that date on, each company adopted a different standard, resulting in program incompatibility on different machines. In order to solve this problem, in the 1970s and 1980s, scientists and engineers, led by the Institute of Electrical and Electronics Engineers – IEEE, developed a standard for representing and operating floating point numbers [30].

Therefore, the IEEE 754 standard presents rules and standards to be followed by computer and software manufacturers in the treatment of computational arithmetic for floating point numbers. Some of these concepts will be detailed below.

2.4.1.1 Representation of Numbers on the Computer

A real number is represented on the computer as floating point number. The term "floating point" is due to the fact that the decimal point of the number moves to the position after the first non-null digit [65]. A number x is represented as floating point number as follows:

$$x = \pm S \times B^E, \quad (2.30)$$

where S is called the signifier or mantissa, E is the exponent and B is the base used.

Most current computers use the binary base ($B = 2$), but to facilitate the iteration of man with the machine, the computer converts numerical values from the decimal input, in addition to converting the output to decimal, regardless of the base used [66].

In current computers that follow the IEEE 754 standard, base 2 is used internally, with digits 0 and 1 [67]. The binary expansion of the signifier of a non-zero number x is given by

$$S = (b_0, b_1 b_2 \dots) \quad \text{with} \quad b_0 = 1. \quad (2.31)$$

The IEEE 754 standard determines that the first bit of the signifier is always different from zero, as seen in the equation (2.31). Usually, this first bit is called a hidden bit because the computer does not use its memory to store it. The number zero cannot be normalized and has a special representation, with zero mantissa and the lowest possible exponent [68].

2.4.1.2 Computational Representation in Different Formats of the IEEE 754 Standard

The IEEE 754 standard specifies two essential formats: the single format, which uses 32 bits, and the double format which uses 64 bits. The single format has 1 bit for the sign, 8 bits for the exponent and 23 bits for the significand. While the double format provides 1 bit for the sign, 11 bits for the exponent and 52 bits for the significand. The sign bit is 0 for positive numbers and 1 for negative numbers [30]. Table 2.1 presents the arrangement of bits.

Table 2.1: Bit arrangement according to IEEE standard for 32 and 64 bit systems.

System	Signal (\pm)	Exponent (E)	Mantissa (S)
<i>single</i> (32 bits)	1 bit	8 bits	23 bits
<i>double</i> (64 bits)	1 bit	11 bits	52 bits

The exponent uses a biased representation. For the single format, the exponent is the binary representation of $E + 127$. The range of E goes from binary number from 1 to 254, which corresponds to $E_{min} = -126$ and $E_{max} = 127$ [69].

The range of values the single format can represent goes from approximately 1.2×10^{-38} to 3.4×10^{38} , while the 64-bit system presents a range of values from approximately 2.2×10^{-308} to 1.8×10^{308} .

Numbers that are smaller than the smallest possible value to be represented by the format are considered equal to zero by the computer (this rounding is called underflow). Similarly, numbers that exceed the upper limit are considered infinite and this event is called overflow. This range of values is symmetrical, valid for both positive and negative numbers [28].

2.4.1.3 Accuracy of a Floating Point System

The precision of a floating point system is associated with the amount of bits representable by the format, given by the number of bits of the significand, including the hidden bit [30]. A floating point with precision p can be expressed by:

$$x = \pm(1.b_1b_2\dots b_{p-2}b_{p-1})_2 \times 2^E. \quad (2.32)$$

Using this representation, the following definition about precision is presented.

Definition 3. Precision (ρ) denotes the number of bits of the mantissa. The double precision ($\rho = 53$) corresponds to approximately $\rho_{10} = \log_{10}(2^{53}) \approx 16$ decimal digits [30].

The smallest floating point x greater than 1 is given by:

$$(1,00\dots01)_2 = 1 + 2^{-(p-1)}. \quad (2.33)$$

The distance between this number and 1 is called the machine's epsilon (ϵ):

$$\epsilon = (0,000\dots01)_2 = 2^{-(p-1)}. \quad (2.34)$$

2.4.1.4 Rounding

A real number in the IEEE 754 floating point standard can be expressed by $x = \pm(b_0, b_1 b_2 \dots b_{p-2} b_{p-1})_2 \times 2^E$, where p is system precision, $b_0 = 1$ and $E_{min} \leq E \leq E_{max}$ for a normalized number or $b_0 = 0$ and $E = E_{min}$ for a subnormalized number.

N_{min} and N_{max} are denoted as the smallest and largest normalized positive values that the machine can represent, respectively. According to Overton [30], a real number is not a floating point if either of the two statements are true:

- x is outside the normalized range, i.e. it is greater than N_{max} , less than $-N_{max}$, is between 0 and N_{min} or even between $-N_{max}$ and 0. For example, $x = 2^{130}$ or $x = 2^{-130}$ are outside the normalized range on a system that adopts the 32-bit single format.
- Binary number expansion requires more than p bits to specify exactly what number, i.e. the floating point precision p is not enough to represent the number x exactly. For example the number $x = 1 + 2^{-25} = (1.000000000000000000000001)_2$ requires more bits than the single format allows.

In both cases, it is necessary to approximate x by a different representation of the original real number, rounding this value to a floating point. According to the IEEE 754 standard [28], rounding the number x can be performed following one of the rules below:

- Round to nearest: the real number x is rounded to the nearest floating point.
- Negative or rounding down: the real number x is rounded to the nearest smaller floating point.
- Round positive or up: the real number x is rounded to the closest floating point that is greater.
- Rounding towards zero: works the same as negative rounding if $x > 0$ and positive rounding if $x < 0$.

Rounding to the nearest is generally adopted as the rounding standard and, if the nearest floating points are equally distanced from the real number, the one with the last significant bit equal to zero is chosen. The notation used to demonstrate the rounding of x will be $round(x)$ and if x is floating point then $round(x) = x$.

2.4.1.5 Computational Errors

Even before the invention of the computer, error analysis and care with numerical precision have been a relevant subject [70]. With the use of the computer, there were significant advances in numerical methods involving this area [71]. Although many programs and algorithms present results with precision, it can happen that computational errors considerably influence the numerical computation results [72].

The difference between the main types of computational errors is important for understanding the behavior of numerical algorithms and the factors that affect the accuracy of the results. Computational errors can be divided into truncation errors and rounding errors [66].

2.4.2 Universal Number (*Unum*)

The universal number was created based on the principles of floating point (IEEE-754), but differs from this one in the ability to vary the precision and number of bits, and also determines whether a number is exact or within a range. This section aims to understand Posit (*Unum* type III).

2.4.2.1 *Unum* Type I and Type II

The concept of *unum* was proposed by Gustafson [31] as an alternative to IEEE floating point arithmetic.

Unum type I can represent either an exact floating point or an open gap between two adjacent floating points, when the computer cannot provide an exact result. For this, *unum* includes an uncertainty bit (*ubit*) as a value of **0** for exact result, and **1** otherwise.

The rest of the *unum* Type I format is similar to the IEEE-754 floating point scheme [28], which features the sign, exponent, and fraction (or mantissa) *bits*. Also, in *unum* format, it presents fields for exponent and fraction length. The components of the *unum* Type I format are presented in the Definition 4, which can also be seen in Figure 2.4.

Definition 4 ([31]). *A unum is a variable-length bit string that has six subfields: Sign Bit, Exponent, Fraction, Uncertainty Bit, Exponent Size, and Fraction Size.*

Therefore, *unum* can be represented as:

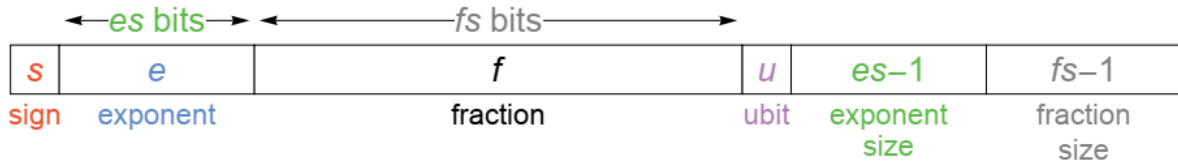


Figure 2.4: *Unum* Type I. Source [31].

The last two fields indicate the number of bits in the exponent and the fraction, allowing both to change with each calculation.

The *unum* Type II format was proposed with the aim of solving some problems presented by Type I, such as the fact that certain values can be represented in different ways. Thus, this Type II is no longer compatible with IEEE-754.

Unum Type II is based on mapping values onto the real projective line, which is the set $\hat{\mathbb{R}} = \mathbb{R} \cup \{\infty\}$. The key concept is that the point where signed (two's complement) numbers change from positive to negative was defined to be the point where positive real numbers turn into negative numbers, and the same ordering, and that point represents the value $\pm\infty$. The Type II format is shown in Figure 2.5. The first quadrant (upper right) presents an ordered set of real numbers, while the fourth quadrant (upper left), the negative numbers.

The bottom half of the circle contains the reciprocals of the numbers in the top half, a reflection of the horizontal axis. In this way, given a certain value, we can obtain the opposite and reciprocal values by vertical and horizontal reflections, respectively.

Unum Type II has a representation limitation to 20 bits, for current technology [32]. With this, the Type III format (*Posit*) was developed.

2.4.2.2 Posit

Posit is based on Type II, which uses the real projective line, but the implementation in *hardware* is similar to the logic used in IEEE-754 floating point arithmetic [73]. This is achieved by relaxing the perfect reflection rule for obtaining the reciprocals. In *Posit* this rule is only for the 0, $\pm\infty$ and integer powers of 2 [32]. Therefore, all numbers are of the form $m \cdot 2^k$, where m and k are integers, and there are no open intervals.

The structure of *Posit* with n bits and es exponent bits is shown in Figure 2.6. The *Posit* format is formed by the sign, regime, exponent and fraction (mantissa) fields.

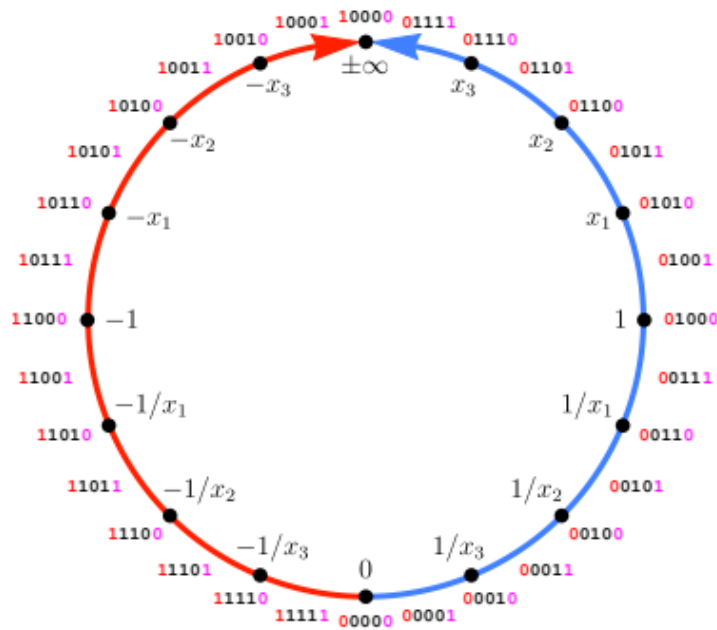


Figure 2.5: *Unum* Type II. Source [32].

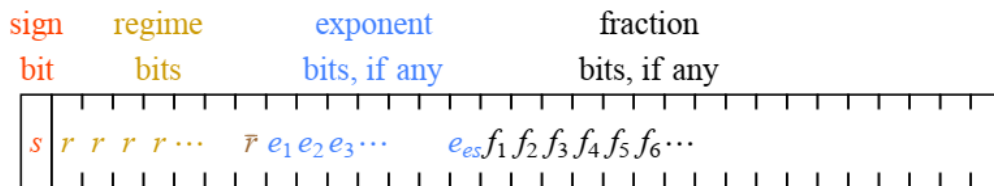


Figure 2.6: Generic posit format for finite, nonzero values. Source [31].

Binary	0000	0001	001x	01xx	10xx	110x	1110	1111
Numerical meaning, k	-4	-3	-2	-1	0	1	2	3

Figure 2.7: Decimal values of regime bits. Source [31].

The regime field is unique to this format. It is used to calculate a scale factor named $useed^k$, where $useed = 2^{2^{es}}$. The k value is determined by the number of identical bits, terminated with an opposite bit, if any. Let m be the number of identical bits; if the regime field consists of leading 0's, then $k = -m$; if they are 1, then $k = m - 1$.

The bits of the exponent (blue color coded) encode the value e and represent the scale factor 2^e . As the regime length is variable, there can be up to es exponent bits, as the first bit of this field is located right after the regime field (therefore, there is a possibility that there are no exponent bits).

The remaining bits after the exponent correspond to the fraction field and represent the fraction value f .

The decimal value of a *Posit* is given by:

$$(-1)^s \times useed^k \times 2^e \times f, \quad (2.35)$$

where s is the sign bit, k is the integer represented by the regime bits, e is the integer represented by the exponent bits, and f is the fraction (including the hidden bit, which is always 1).

Consider the example of representing the number 3.55383×10^{-6} in posit, taken from the book [31].

Example 6. Representation of the number 3.55383×10^{-6} in Posit.

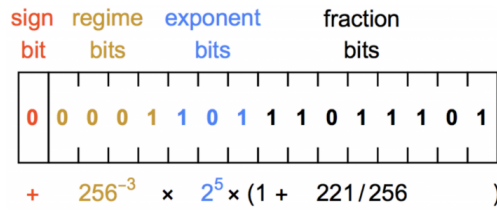


Figure 2.8: Example of a posit number.

$$(-1)^{sign} \times useed^k \times 2^e \times f, \quad (2.36)$$

where $useed = 2^{2^3} = 256$, $k = -3$, $e = 5$ and $f = 1 + 221/256$.

2.5 Green Algorithm

Several human activities are responsible for significant greenhouse gas emissions, including data centers and other sources of large-scale computing. While many important scientific milestones have been reached thanks to the development of high-performance computing, the resulting environmental impact is underestimated. In Lannelongue et al. [36], a methodology for the carbon footprint of any computational activity is presented. An online tool was developed called Green Algorithm, in which with minimal information it is possible to estimate the carbon footprint of computer simulations. The program needs basic information from the computer on which the algorithm is being simulated, such as the processor and memory, as well as the simulation time and from this data it is possible to estimate the carbon footprint.

CHAPTER 3

Simulation of continuous-time systems using RK4 and floating point arithmetic (IEEE - 754)

This chapter brings the simulation of continuous systems that will be discretized by the fourth order Runge-Kutta method, the result of this discretization will be simulated in floating point arithmetic. Due to the discretization result and computational precision, some monomials may be excluded based on Theorem 3 that will be presented below, but first some definitions will be presented to substantiate and understand the proposed theorem.

Definition 5. *Monomial is an algebraic expression formed by a real number, or a variable, or by a multiplication of numbers (coefficients) and variables.*

Example 7. *The algebraic expressions in (3.1)-(3.4) are examples of monomials*

$$2, \tag{3.1}$$

$$4.7 \times 10^{-3}, \tag{3.2}$$

$$5x^2y, \tag{3.3}$$

$$3.2 \times 10^2xyz. \tag{3.4}$$

Addition and subtraction of numbers in scientific notation can only be performed when there are equal exponents. The monomial in (3.2) can be represented by mantissa ($S = 4.7$), base ($B = 10$) and exponent ($E = -3$).

Example 8. Consider three monomials,

$$x_1 = -1.2 \times 10^2, \quad (3.5)$$

$$x_2 = 2.5 \times 10^3, \quad (3.6)$$

$$x_3 = 3.3 \times 10^{-1}. \quad (3.7)$$

To add them, it is necessary to convert their representations to the same exponent. In this case, the conversion is performed for the highest value exponent ($E = 3$).

$$\begin{aligned} S &= x_1 + x_2 + x_3 \\ &= -0.12 \times 10^3 + 2.5 \times 10^3 + 0.00033 \times 10^3 \\ &= (-0.12 + 2.5 + 0.00033) \times 10^3 \\ &= 2.38033 \times 10^3. \end{aligned}$$

The key point is to analyse monomials of systems discretized by the fourth-order Runge-Kutta method. As some monomials can be of high-order, it can be noticed that round-off operation may turn some of these monomials negligible. Thus, it is possible to safely determine that such monomials may be excluded or not. The result of excluding monomials is presented in the following theorem.

The theorem is general for any computational arithmetic, being necessary to consider the precision value of each arithmetic. That is, just adjust the value of ρ in base 10. For the proof of the theorem, floating point arithmetic for the double format (IEEE-754 2019) was considered.

Theorem 3 ([44]). Let γ be a set of monomials, that is, $\gamma = \{\alpha_1 10^{\beta_1}, \alpha_2 10^{\beta_2}, \dots, \alpha_n 10^{\beta_n}\}$, with $1 \leq \alpha_n \leq 9$ and with $\beta_n > \beta_{n-1} > \beta_{n-2} > \dots > \beta_1$. Let Ω be the set of differences between β_n and other exponents, that is, $\Omega = \{\Omega_1, \Omega_2, \dots, \Omega_{n-1}\} = \{(\beta_n - \beta_1), (\beta_n - \beta_2), \dots, (\beta_n - \beta_{n-1})\}$. If $\Omega_i > \rho$, then the monomial γ_i may be excluded in the implementation of the discretization scheme.

Proof. Definition 3 shows that a number represented in the decimal base for double precision is approximately $\rho \approx 16$ decimal digits. Then, if the necessary adjustment to perform the sum (or subtraction) is greater than 16, the number will be represented by zeros multiplied by 10^β , that is, $0.0000000000000000 \times 10^\beta$, which confirms its exclusion. This rationale can be applied for any number of bits or decimal digits. \square

Example 9. Consider the following equation

$$\begin{aligned} X &= 0.05 + 6.5104 \times 10^{-20} - 3.90625 \times 10^{-10} \\ &= 5 \times 10^{-2} + 6.5104 \times 10^{-20} - 3.90625 \times 10^{-10} \end{aligned}$$

According to Theorem 3, the set of monomials are as follows:

$$\begin{aligned} \gamma_1 &= \alpha_1 10^{\beta_1} = 6.5104 \times 10^{-20} \\ \gamma_2 &= \alpha_2 10^{\beta_2} = -3.90625 \times 10^{-10} \\ \gamma_3 &= \alpha_3 10^{\beta_3} = 5.0 \times 10^{-2}. \end{aligned}$$

The set Ω is given by

$$\begin{aligned} \Omega_1 &= \beta_3 - \beta_1 = -2 - (-20) = 18 \\ \Omega_2 &= \beta_3 - \beta_2 = -2 - (-10) = 8. \end{aligned}$$

Since $\Omega_1 > 16$ (for double precision), γ_1 may be excluded without loss of accuracy. This is verified in Eq. (3.8). Fig. 3.1(a) and 3.1(b) present this operation using Matlab for long and hexadecimal formats, respectively. As it can be seen, the hexadecimal remains the same after the exclusion of the monomial.

$$\begin{aligned} X &= 5 \times 10^{-2} + 0.0000000000000000 \times 10^{-2} \\ &\quad - 0.0000000390625 \times 10^{-2} \\ &= 4.9999999609375 \times 10^{-2} \end{aligned} \tag{3.8}$$

To show what happens when there are monomials with the same exponent value, an example is now given (Example 10).

Example 10. Consider the following equation

$$\begin{aligned} X &= 0.04 + 6.5 \times 10^{-20} - 3.9 \times 10^{-10} + 0.02 \\ &= 5 \times 10^{-2} + 6.5 \times 10^{-20} - 3.9 \times 10^{-10} + 2 \times 10^{-2} \end{aligned}$$

<pre>>> format long >> 0.05 + 6.5104e-20-3.90625e-10 ans = 0.0499999999609375 >> 0.05-3.90625e-10 ans = 0.0499999999609375</pre>	<pre>>> format hex >> 0.05 + 6.5104e-20-3.90625e-10 ans = 3fa99999963e9b47 >> 0.05-3.90625e-10 ans = 3fa99999963e9b47</pre>
---	--

(a) Format Long

(b) Format Hexadecimal

Figure 3.1: Example 9 was run on Matlab™. The result is presented in long and hexadecimal formats, as described in Eq. (3.8). The exclusion of monomial $\gamma_1 = 6.5104 \times 10^{-20}$ does not change the final result.

According to Theorem 3, the set of monomials are as follows:

$$\begin{aligned}\gamma_1 &= \alpha_1 10^{\beta_1} = 6.5 \times 10^{-20} \\ \gamma_2 &= \alpha_2 10^{\beta_2} = -3.9 \times 10^{-10} \\ \gamma_3 &= \alpha_3 10^{\beta_3} = 2.0 \times 10^{-2} \\ \gamma_4 &= \alpha_4 10^{\beta_4} = 5.0 \times 10^{-2}.\end{aligned}$$

The set Ω is given by

$$\begin{aligned}\Omega_1 &= \beta_4 - \beta_1 = -2 - (-20) = 18 \\ \Omega_2 &= \beta_4 - \beta_2 = -2 - (-10) = 8 \\ \Omega_3 &= \beta_4 - \beta_3 = -2 - (-2) = 0.\end{aligned}$$

Since $\Omega_1 > 16$ (for double precision), γ_1 may be excluded without loss of accuracy. The monomial represented by Ω_3 will be kept in the representation, since only monomials where $\Omega > 16$ will be excluded.

Based on Theorem 3 and on the examples above, one can summarize the steps to obtain an effective discretization in:

1. Discretization of the continuous system by the fourth-order Runge-Kutta method;

2. Apply to each monomial the values of the initial conditions, parameters and step-size, in order to obtain a numeral monomial in base 10;
3. Evaluate each of the monomials based on the Theorem 3 and exclude the monomials that do not respect the precision;
4. Get the effective discretization.

First, the continuous system is discretized using the fourth-order Runge-Kutta method in the Maple software, version 18. The result of the discretization will be symbolic, that is, depending on the variables, parameters and step-size.

For floating point arithmetic (IEEE-754 2019), the discretization result will be transferred to Matlab software, version R2016a - 64 bits. For each function $(x_{k+1}, y_{k+1}, z_{k+1})$ a vector will be created, which each position of the vector will be occupied by a monomial resulting from the dicretization. Once this is done, the initial condition, parameters and step-size will be considered, in order to obtain monomials in base 10. Afterwards, it is possible to exclude some monomials based on the Theorem 3.

The proposed approach is illustrated for three systems: the Rössler system [74], the Lorenz equations [75] and Sprott B [76]. For each system, the reduced RK4 based on floating point arithmetic, here named as RRK4, according to Theorem 3 is calculated. The quality of RRK4 is evaluated by means of the observability of the dynamical systems, plot of the projections on the xy plane of the discretized systems and computation of the largest Lyapunov exponent. The step size h is chosen according to the usual values found in the literature. In addition, all parameters used for the systems under study are shown in Table 3.1.

Table 3.1: Parameters of the systems under study. 1) Rössler: $(x_0, y_0, z_0) = (-1.0, 1.0, 1.0)$, $a = 0.15$, $b = 0.20$ and $c = 10.0$, and step-size 10^{-2} ; 2) Lorenz: $(x_0, y_0, z_0) = (1.0, 0.5, 0.9)$, $\sigma = 16.0$, $\beta = 4.0$, $\rho = 45.92$, $h = 10^{-3}$; 3) Sprott B: $(x_0, y_0, z_0) = (0.05, 0.05, 0.05)$ and $h = 10^{-2}$.

System	Initial condition (x_0, y_0, z_0)	Parameters	Step-size
Rössler	$(-1.0, 1.0, 1.0)$	$a = 0.15, b = 0.20$ and $c = 10.0$	10^{-2}
Lorenz	$(1.0, 0.5, 0.9)$	$\sigma = 16.0, \beta = 4.0, \rho = 45.92$	10^{-3}
Sprott B	$(0.05, 0.05, 0.05)$	-	10^{-2}

3.1 Rössler System

Let the Rössler system be represented by [74]:

$$\begin{cases} \dot{x} = -y - z, \\ \dot{y} = x + ay, \\ \dot{z} = b + z(x - c), \end{cases} \quad (3.9)$$

where bifurcation parameters are described by (a, b, c) . Rössler equations were discretized by the Runge-Kutta scheme of fourth-order. Theorem 3 was applied considering initial condition $(x_0, y_0, z_0) = (-1.0, 1.0, 1.0)$, parameters $a = 0.15$, $b = 0.20$ and $c = 10.0$, and step-size of 10^{-2} . The number of monomials for each equation of the Rössler system is shown in Table 3.2. RK4 represents the discretization performed by the fourth-order Runge-Kutta method, and RRK4 represents the fourth-order Runge-Kutta method when Theorem 3 is applied and some terms are excluded.

Table 3.2: Number of monomials for each of the discretized equations for the systems Lorenz, Rössler and Sprtt B. The comparison is made between conventional RK4 and Reduced RK4 (RRK4). The highest reduction in the number of monomials occurs for the Lorenz equations, where the total is decreased in 75.5%.

Equations	Rössler		Lorenz		Sprtt B	
	RK4	RRK4	RK4	RRK4	RK4	RRK4
x_{k+1}	117	101	116	79	175	41
y_{k+1}	38	38	886	192	38	26
z_{k+1}	903	290	963	210	167	37

The exclusions of monomials are related to the observability of the equations. To investigate the observability of each dynamical variable, it is necessary to begin with a measurement component so that $r = g(x, y, z) = y$. The coordinate transformation Φ_y peruses at that point as

$$\Phi_y = \begin{cases} X = y, \\ Y = ay + x, \\ Z = -y - z + ax + a^2y, \end{cases} \quad (3.10)$$

and the equivalent jerk equation F_y is

$$F_y = -b - cX + (ac - 1)Y + (a - c)Z - aX^2 + (a^2 + 1)XY - aXZ - aY^2 + YZ. \quad (3.11)$$

When the observable is variable x of the Rössler system, the coordinate transformation Φ_x is as follows

$$\Phi_x = \begin{cases} X = x, \\ Y = -y - z, \\ Z = -x - ay - z(x - c) - b, \end{cases} \quad (3.12)$$

and the corresponding jerk equation F_x is

$$F_x = ab - cX + X^2 - aXY + acY + XZ + (a - c)Z - \frac{(a + c + Z - aY + b)}{a + c - X}. \quad (3.13)$$

For variable z of the Rössler system as the observable, the coordinate transformation Φ_z is as follows

$$\Phi_z = \begin{cases} X = z, \\ Y = z(x - c) + b, \\ Z = (b + z(x - c))(x - c) + z(-y - z), \end{cases} \quad (3.14)$$

and the associated jerk equation is

$$F_z = b - cX - Y + aZ + aX^2 - XY + \frac{(ab + 3Z)Y - aY^2 - bZ}{Z} + \frac{2bY^2 - 2Y^3}{X^2}. \quad (3.15)$$

Analysing Eqs. (3.11), (3.13) and (3.15) it is possible to state that Eq. (3.15) is more complex than Eq. (3.13) and therefore is more complex than Eq. (3.11). Similar to [62], we consider complexity associated with the number of monomials included in the jerk equation, order of the nonlinearities and poles. Note that the jerk equation indicates the connections between the states of the original system “seen from one observable point of view”.

Based on the complexity of F_x , F_y e F_z , it can be said that variable y is more observable than variable

x , which is more observable than variable z , that is, $y \triangleright x \triangleright z$. From the results presented in Table 3.2, it is conceivable to evaluate the observability and compare it with what has already been exposed before. Variable y is more observable than variable x , which is more observable than the variable z . Since variable z is the one with the greatest exclusion of terms, it does not contribute much to the description of the system. This statement corroborates the observability analysis carried out previously.

Figures 3.2(a) and 3.2(b) show the projections on the xy plane of the Rössler equations for the fourth-order Runge-Kutta method and the reduced fourth-order Runge-Kutta method. The trajectories exhibit very close values in both cases, so Figures 3.2(a) and 3.2(b) are practically the same, illustrating Theorem 3 consequences.

The largest Lyapunov exponent was calculated using the method proposed by Nepomuceno and Mendes [20] for both RK4 and RRK4, as shown in Table 3.3. This method uses the interval extensions to calculate the lower bound error which is a measure of the distance between the simulated dynamical systems (or pseudo-orbit) and the real orbit. When a system behaves chaotically the distance between these two orbits is exponentially divergent, and therefore a slope in a logarithm plot of the lower bound error captures the divergence and quantifies it as a number which is the positive Lyapunov exponent. The positive exponent was found to be 0.0897 and 0.0909 for RK4 and RRK4, respectively, which is in a good agreement with the literature value of 0.090 nat/iter [77]. Table 3.4 shows the number of points needed to calculate the Lyapunov exponent using the method proposed in [20]. It is possible to observe that RK4 uses a slightly larger number for this estimation.

Table 3.3: Calculation of Lyapunov exponent. The expected values are obtained in [77] for Rössler and Lorenz equations and in [76] for Sprott B. The unit of the Lyapunov exponent is indicated in nat/iter.

System	Literature λ	Calculated λ for RK4	Calculated λ for reduced RK4
Rössler	0.0900	0.0897	0.0909
Lorenz	1.5000	1.4702	1.4778
Sprott B	0.2100	0.1710	0.1854

Table 3.4: Number of points needed to estimate the Lyapunov exponent for each system.

System	RK4	RRK4
Rössler	35218	33331
Lorenz	21276	19678
Sprott B	14893	12687

In order to analyse the computational complexity of the systems under study, the mathematical operations were counted and the simulation time has been calculated.

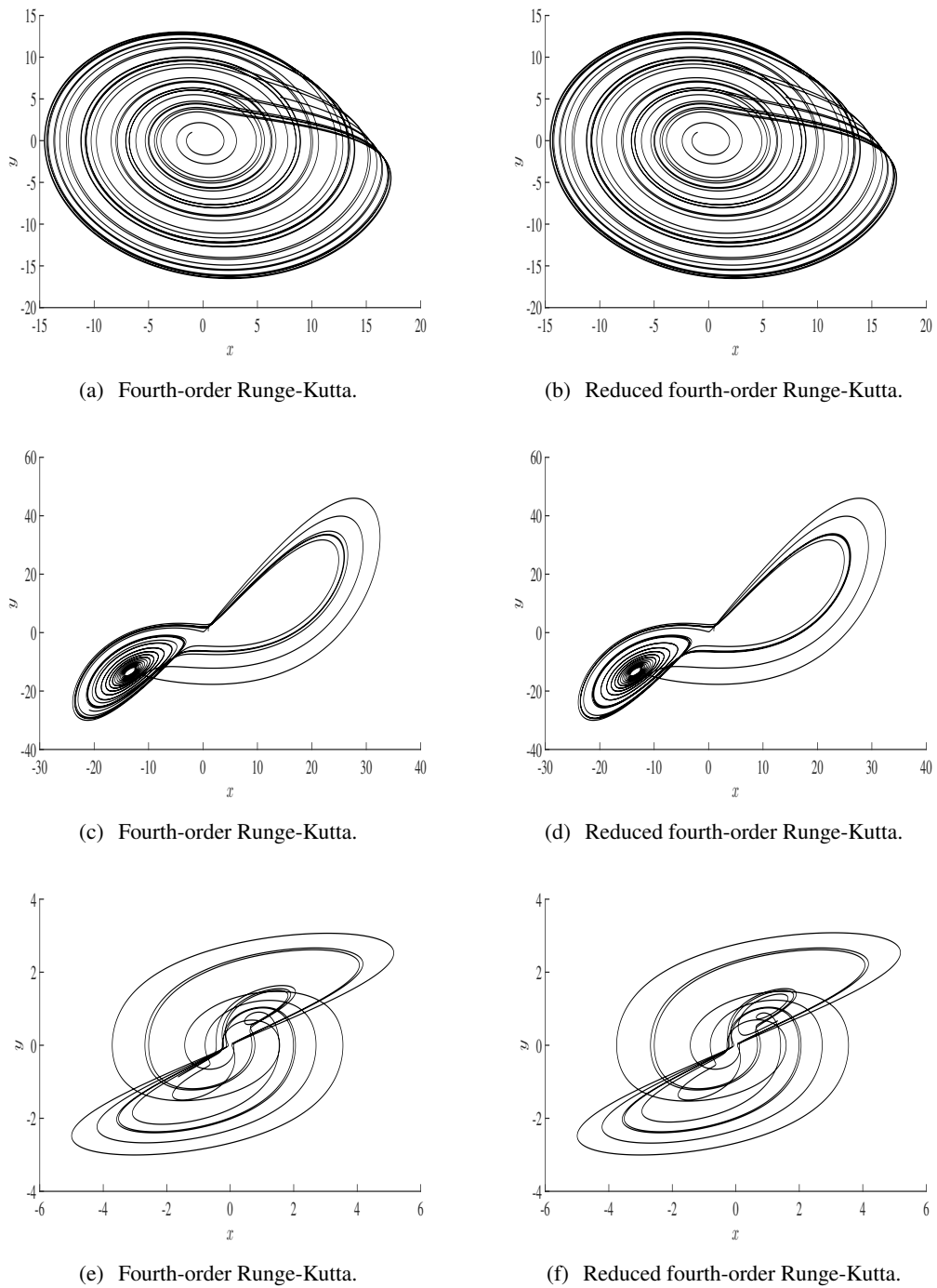


Figure 3.2: (a) and (b): Projections on the xy plane of the discretized Rössler equations. (c) and (d): Projections on the xy plane of the discretized Lorenz equations. (e) and (f): Projections on the xy plane of the Sprott B.

Table 3.5 shows the summary of the basic operations used in the calculations. The number of operations is per iteration that outfits a computational complexity of $O(n)$. Rössler equations present a total of 9080 operations per iteration using RK4 whereas, using RRK4, the number of operations was 3095, representing a reduction of approximately 65.9% of operations performed per iteration.

Table 3.5: Summary of computational complexity. The basic operations used throughout each system were analysed, that is, Sum/Subtraction, Multiplication/Division and Power. For each system, all the operators for variables x, y, z are added and the reduction was calculated. The proposed method can reduce up to 81.1% of the required operations for the Lorenz equations.

	Operations	RK4			Reduced RK4			Reduction
		x_{k+1}	y_{k+1}	z_{k+1}	x_{k+1}	y_{k+1}	z_{k+1}	
Rössler	Sum/Subtraction	116	37	902	100	37	289	
	Multiplication/Division	527	141	4878	445	141	1269	
	Power	192	47	2240	161	47	606	
	Summation of operators	835	225	8020	706	225	2164	
	Total			9080			3095	65.9%
Lorenz	Sum/Subtraction	115	885	962	78	191	209	
	Multiplication/Division	605	5260	5739	325	877	980	
	Power	289	2881	3190	172	425	514	
	Summation of operators	1009	9026	9891	575	1493	1703	
	Total			19926			3771	81.1%
Sprott B	Sum/Subtraction	174	37	166	40	25	36	
	Multiplication/Division	726	139	710	118	64	115	
	Power	404	60	393	62	33	64	
	Summation of operators	1304	236	1269	220	122	215	
	Total			2809			557	80.2%

Table 3.6: Average time in seconds of a thousand attempts to execute the proposed algorithm. We have also presented one standard deviation in order to consider the intrinsic fluctuation of time consumption in a computer.

System	$RK4$	$RRK4$	Reduction
Rössler	4.1729 ± 0.5054	1.4500 ± 0.0991	65.3%
Lorenz	18.7971 ± 0.1078	1.7481 ± 0.0994	90.7%
Sprott B	1.6427 ± 0.0567	0.2540 ± 0.0085	84.5%

The simulation time is shown in Table 3.6. Each system was simulated a thousand times and the time shown in the table is the average over the outcome of the simulations. For the reduced fourth-order Runge-Kutta it is evident that there was a significant reduction in the simulation time. Since the decrease in time is directly related to the amount of operations performed, the contribution to the decrease in computational cost is worth considering.

Table 3.7 shows the average simulation time for various iterations. As the number of iterations increases, the time also increases linearly for both RK4 and RRK4. Hence, the simulation time can be related to the number of iterations using the equation for RK4:

$$t(k) = 0.0005k + 0.6973, \quad (3.16)$$

and for RRK4:

$$t(k) = 0.0001k + 0.0995, \quad (3.17)$$

where $t(k)$ represents the simulation time in seconds for k iterations.

Table 3.7: Average time in seconds of a hundred attempt to execute the proposed algorithm. Initial conditions, parameters and step size for each system are as follows: 1) Rössler: $(x_0, y_0, z_0) = (-1.0, 1.0, 1.0)$, $a = 0.15$, $b = 0.20$ and $c = 10.0$, and step-size of 10^{-2} ; 2) Lorenz: $(x_0, y_0, z_0) = (1.0, 0.5, 0.9)$, $\sigma = 16.0$, $\beta = 4.0$, $\rho = 45.92$, $h = 10^{-3}$; 3) Sprott B: $(x_0, y_0, z_0) = (0.05, 0.05, 0.05)$ and $h = 10^{-2}$.

Iteration	Rössler		Lorenz		Sprott B	
	RK4	RRK4	RK4	RRK4	RK4	RRK4
500	0.9391	0.1800	2.6733	0.2048	0.1622	0.0294
1000	1.1450	0.2439	3.4366	0.2855	0.2351	0.0423
1500	1.3790	0.3196	4.3002	0.3769	0.3181	0.0550
2000	1.6072	0.3926	5.1826	0.5393	0.4029	0.0681
2500	1.8093	0.4813	6.0595	0.5520	0.4873	0.0797
3000	2.0389	0.5360	6.9338	0.6390	0.5689	0.0968
3500	2.2802	0.6143	7.7909	0.7257	0.6510	0.1069
4000	2.4932	0.6904	8.6913	0.8227	0.7329	0.1239
4500	2.6933	0.7648	9.5379	0.9037	0.8098	0.1329
5000	2.9826	0.8267	10.4094	0.9924	0.8957	0.1475
5500	3.2316	0.9050	11.2939	1.0876	0.9785	0.1608
6000	3.4198	0.9700	12.1794	1.1691	1.0620	0.1716
6500	3.6160	1.0470	13.0695	1.2572	1.1436	0.1863
7000	3.8412	1.1218	14.0214	1.3443	1.2639	0.1970
7500	4.0879	1.1989	14.8371	1.4378	1.3052	0.2121
8000	4.3457	1.2722	15.6907	1.5242	1.3904	0.2297
8500	4.5597	1.3437	16.5518	1.6121	1.4750	0.2412
9000	4.7452	1.4097	17.4318	1.6973	1.5640	0.2500
9500	4.9484	1.4943	18.2519	1.7946	1.6349	0.2615
10000	5.2406	1.5631	19.1772	1.8750	1.7258	0.2745

With the estimates presented by equations (3.16) and (3.17), it is possible to observe that with RRK4 there is a reduction in the simulation time, as a consequence of the reduction in the number of operations. And with the growing concern about climate factors, especially the carbon footprint, this reduction contributes to the development of more efficient algorithms from this point of view. Table 3.8 shows the reduction in carbon footprint. To find the simulation time, equations (3.16) and (3.17) were considered, for k equal to one million iterations, which would be the approximate number of iterations needed to build the bifurcation diagram. For the Rössler system there was a reduction of approximately 75%.

Table 3.8: Carbon Footprint for the systems of Rössler, Lorenz and Sprott B for the original and reduced systems using the Floating Point arithmetic.

		Time	Carbon Footprint	
			Decreased (%)	gCO_2e
Rössler	RK4	8 min 21 sec	-	1.39
	RRK4	1 min 40 sec	75.04%	0.347
Lorenz	RK4	28 min 21 sec	-	4.86
	RRK4	3 min 20 sec	89.3%	0.52
Sprott B	RK4	3 min 20 sec	-	0.52
	RRK4	30 sec	66.54%	0.174

3.2 Lorenz Equations

Consider the Lorenz equations [75]

$$\begin{cases} \dot{x} = \sigma(y - x), \\ \dot{y} = x(\rho - z) - y, \\ \dot{z} = xy - \beta z, \end{cases} \quad (3.18)$$

where σ , ρ and β are parameters. Lorenz equations were discretized by the fourth-order Runge-Kutta method. To apply Theorem 3, initial condition $(x_0, y_0, z_0) = (1.0, 0.5, 0.9)$, parameters $\sigma = 16.0$, $\beta = 4.0$ and $\rho = 45.92$, and step-size of 10^{-3} were considered. The number of monomials for each equation of the Lorenz equations is shown in Table 3.2. As it occurred to Rössler's equations, the reduced RK4 varies as iterations of Lorenz equations occur.

The observability of each dynamic variable was investigated, and the associated model for variable x is

$$F_x = X\beta\rho\sigma - X^3\sigma - X\beta\sigma - \beta\sigma Y - X^2Y - \beta Y - \beta Z - \sigma Z + \frac{\sigma Y^2}{X} - Z + \frac{Y^2}{X} + \frac{YZ}{X} \quad (3.19)$$

whereas for the other coordinates the respective equations have a large number of monomials and, more

importantly they are of the following implicit form

$$\dot{X} = Y, \quad \dot{Y} = Z, \quad F_r(X, Y, Z, \dot{Z}) = 0. \quad (3.20)$$

Equations relating to F_x , F_y , and F_z have 12, 588, and 211 monomials, respectively. Based on the complexity of the jerk equations, it can be concluded that variable x is more observable than variable z , which is more observable than variable y , that is, $x \triangleright z \triangleright y$. This is slightly different from the results in Table 3.2, variable x is more observable than variable y , which is more observable than variable z , that is, $x \triangleright y \triangleright z$. The result is in accordance with what is stated in [62, 78] and indicates that the variable z is the least observable, based on symmetry considerations. Nonetheless the result presented in Table 3.2 is in accordance with the observability analysis performed previously using the jerk equations, which determines that the variable x is the most observable.

Figures 3.2(c) and 3.2(d) show the projections on the xy plane of the Lorenz equations. The trajectories exhibit very close values in both cases. The Lyapunov exponent was calculated as shown in Table 3.3 and Table 3.4 shows the number of points needed to calculate the Lyapunov exponent. Values obtained from RK4 and RRK4 are similar to the values found in the literature.

Table 3.5 shows the summary of the basic operations used in the calculations. The Lorenz equations presented 19926 and 3771 mathematical operations per iteration, when discretized using RK4 and RRK4, respectively. The reduction in the number of mathematical operations was 81.1%.

Table 3.6 shows the average time of simulation of Lorenz equations. As there was a significant reduction in the number of operations, evidently the simulation time for the reduced equations was shorter.

The average simulation time for different iterations is shown in Table 3.7. It is observed that the time grows linearly as the number of iterations increases. Hence, the time-per-iteration relationship can be represented by the following equations for RK4 (Equation (3.21)) and RRK4 (Equation (3.22)), respectively:

$$t(k) = 0.0017k + 1.7055, \quad (3.21)$$

$$t(k) = 0.0002k + 0.123, \quad (3.22)$$

where $t(k)$ is the simulation time in seconds for k iterations.

The carbon footprint of the systems was estimated using Equations (3.21) and (3.22). For the Lorenz

equations, there was a significant reduction of approximately 89.3% in the carbon footprint, as shown in Table 3.8.

3.3 Sprott B

Consider the Sprott B system [76]:

$$\begin{cases} \dot{x} = yz, \\ \dot{y} = x - y, \\ \dot{z} = 1 - xy. \end{cases} \quad (3.23)$$

The Sprott B system was discretized using the fourth-order Runge-Kutta method. In order to apply Theorem 3, $(x_0, y_0, z_0) = (0.05, 0.05, 0.05)$ and step-size of 10^{-2} were considered as indicated in [76, 79, 80]. The number of monomials for each equation of the Sprott B system is shown in Table 3.2. RRK4, as shown in Table 3.2, is achieved for some iterations since, as the Sprott B equations are iterated, their values change. Thus, the values of the monomials are also changed, causing the number of monomials to vary for each equation.

For the investigation of the observability of each dynamic variable, consider $r = g(x, y, z) = y$ as and coordinate transformation Φ_y such as

$$\Phi_y = \begin{cases} X = y, \\ Y = x - y, \\ Z = -x + (1 + z)y, \end{cases} \quad (3.24)$$

and the corresponding jerk equation F_y is

$$F_y = -Z + (1 - XY + X^2)X - (Y + Z)\frac{Y}{X}. \quad (3.25)$$

When the observable is variable z of the Sprott B, the coordinate transformation Φ_z reads as

$$\Phi_z = \begin{cases} X = z, \\ Y = 1 - xy, \\ Z = xy - x^2 - y^2z, \end{cases} \quad (3.26)$$

and the corresponding jerk equation F_z is

$$F_z = \frac{1}{2X} \{8X^2Y - 8X^2 - 2XY - 4XZ + Y^2 + YZ + 2X - Y \pm (2X - Y)(-4XY^2 + 8XY + Y^2 + 2YZ + Z^2 - 4X - 2Y - 2Z + 1)^{\frac{1}{2}}\}. \quad (3.27)$$

The final case is to consider the variable x of the Sprott B as the observable. The coordinate transformation Φ_x reads as

$$\Phi_x = \begin{cases} X = x, \\ Y = yz, \\ Z = (x - y)z + (1 - xy)y, \end{cases} \quad (3.28)$$

and the associated jerk equation is

$$\begin{aligned} F_x^3 X^3 + (4X^3Y - 3X^4 + 7X^3Z - 3X^2Y^2 - 3X^2YZ - X^2 + XY)F_x^2 + (22X^3Y^2 + 27X^6Y + 9X^6Z - 18X^4Y \\ - 22X^4Z + 28X^3YZ + 15X^3Z^2 + 4X^3 - 8X^2Y^3 - 22X^2Y^2Z - 14X^2YZ^2 - 5X^2Y - Y^3 - 3X^2Z + 3XY^4 \\ + 6XY^3Z + 3XY^2Z^2 + 2XY^2 + 4XYZ - Y^2Z)F_x + 27X^9Y - 45X^7Y - 18X^7Z + 4X^6 + 31X^6Y^2 + 39X^6YZ \\ + 9X^6Z^2 - 27X^5Y^3 - 36X^5Y^2Z - 9X^5YZ^2 + 17X^5Y + 16X^5Z - 22X^4Y^2 - 42X^4YZ - 27X^4Z^2 - 4X^4 \\ + 40X^3YZ^2 + 5X^3Y + 9X^3Z^3 + 6X^3Z + 31X^3Y^3 + 60X^3Y^2Z - 19X^2Y^4 - 44X^2Y^3Z - 40X^2Y^2Z^2 - 5X^2Y^2 \\ - 15X^2YZ^3 - 9X^2YZ - 2X^2Z^2 + 4XY^5 + 15XY^4Z + 18XY^3Z^2 + 4XY^3 + 7XY^2Z^3 + 6XY^2Z - 3Y^5Z \\ - 3Y^4Z^2 + 4XYZ^2 - Y^6 - Y^3Z^3 - 3Y^3Z - 2Y^2Z^2 = 0. \end{aligned} \quad (3.29)$$

It can be seen that Eq. (3.29) is more complex than Eq. (3.27), just as this equation is more complex than Eq. (3.25). Therefore, one can guarantee that the variable y is more observable than the variable z , which is more observable than the variable x , that is, $y \triangleright z \triangleright x$. According to the results presented in Table 3.2, it is possible to assess the observability of the system. Variable y is the most observable since there is a lesser exclusion of terms, followed by variables z and x . All this is in conformity with the observability analysis carried out previously.

Figure 3.2 shows the projections on the xy plane of the Sprott B for RK4 and RRK4. Note that Figures 3.2(e) and 3.2(f) appear to be the same since their pseudo-orbits exhibit very close values. This highlights the concept of monomial exclusion.

In addition, using both RK4 and RRK4, a Lyapunov exponent of 0.1710 nat/iter and 0.1854 nat/iter was found, respectively, as shown in Table 3.3, which is also in good agreement with literature. Table 3.4 shows the number of points needed to calculate the Lyapunov exponent. It is possible to observe again

that RK4 needs a slightly larger number of iterations to estimate the exponent.

Table 3.5 shows the number of operations required to simulate the Sprott B system per iteration. The Sprott B system presented a total of 2809 operations when discretized using RK4 and 557 operations using RRK4. Therefore, there was a reduction of approximately 80.2% in the number of mathematical operations when Theorem 3 was applied.

Table 3.6 shows the average time of simulation, as shown in other systems, the number of mathematical operations reduced and consequently the average time of simulation also reduced.

Table 3.7 shows the average simulation time for different iterations. As the number of iterations increases, the time grows linearly, as demonstrated by the relationships described by Equations (3.30) and (3.31). The time shown in Table 3.8 is an estimate for building the bifurcation diagram for one million iterations. Thus, the reduction in the carbon footprint was approximately 66.54%.

$$t(k) = 0.0002k + 0.0729, \quad (3.30)$$

$$t(k) = 0.00003k + 0.0166, \quad (3.31)$$

where $t(k)$ represents the simulation time in seconds for k iterations.

3.4 Discussion

Theorem 3 was applied to a discretization scheme, namely the fourth-order Runge-Kutta scheme with fixed step-size, for the still widespread use and also for better visualisation of the amount of excluded (or neglected) terms (or monomials).

It could be argued that the proposed computational discretization scheme does not improve the performance of classical Runge-Kutta numerical methods since the removal of the neglected terms in the method is automatically performed by the computer and therefore the advantage of not including them in the general scheme has no effect on the results. Unfortunately that is not the case since the computer will perform the operations related to the neglected terms anyhow before returning zero. That certainly has an impact on the overall performance as shown here.

Stretching this argument a bit further, it could be also argued that the advantage related to the number of operations performed by computers could be considered as not very important, if their current performance is taken into account. That is not entirely true, if a discretization scheme is to be implemented in

a FPGA device for instance, the code length is certainly a factor to be taken into consideration. A shorter algorithm will certainly have an appeal in this case.

Another criticism is the use of a fixed step size Runge-Kutta of low order (considered outdated) when a variety of discretization schemes with variable step size are currently available. The change in the step will aggravate the problem when its value becomes smaller. More terms will be neglected and the final operation will be performed by far less terms which again shows the importance of the present study.

The results presented by floating point arithmetic were significant from the point of view of reducing the number of monomials, reducing operations performed per iteration and consequently reducing simulation time. It is important to emphasize that the results obtained with the effective discretization maintained the characteristics of the systems, such as attractors, Lyapunov exponent and Observability.

CHAPTER 4

Simulation of continuous-time systems using RK4 and Posit arithmetic

The idea of excluding terms due to computer precision will be applied to Posit arithmetic. Theorem 3 will consider the accuracy of the Posit.

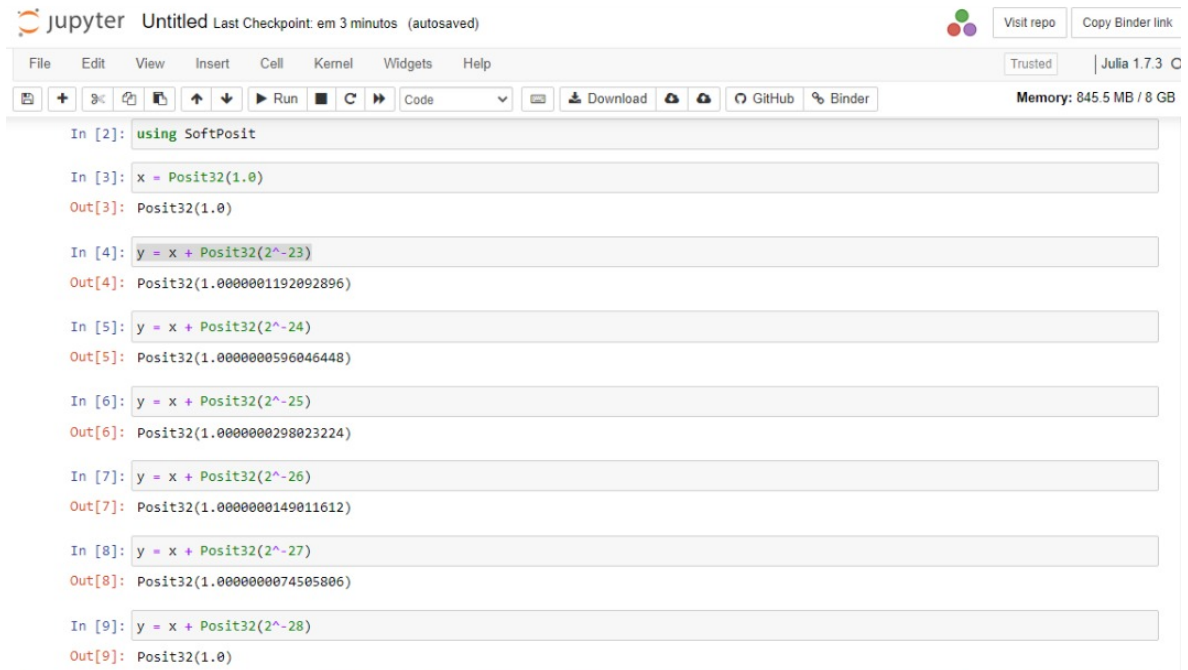
Consider Figure 4.1, a screenshot of the computer window with a Jupyter notebook. This figure brings the sum of successive numbers with a chosen number x . The computer works with base 2, so adding successive numbers is the same as adding x to 2^y , where y are successive numbers. Therefore, when $x + 2^y$ is equal to x , it can be stated that for this value of y there is a loss of computational precision.

According to Figure 4.1, the SoftPosit library used to implement Posit arithmetic loses precision for mantissa equal to 28 bits, that is, $\rho = 28$; which corresponds to approximately 8 decimal places. In this chapter, a value of $\rho_{10} \approx 8$ will be considered. Therefore, the same theorem presented for floating point arithmetic in Theorem 3 will be applicable to Posit arithmetic, with the only difference being the value of ρ_{10} .

Moreover, for the Posit arithmetic, the result of the discretization carried out in the Maple software was transferred to the Python software, version 3.7.1 and the SoftPosit library was used, and the same steps to obtain an effective discretization in floating point arithmetic will be done in Posit arithmetic.

Example 11. Consider the following equation

$$\begin{aligned} X &= 0.05 + 6.51 \times 10^{-5} + 3.96 \times 10^{-12} \\ &= 5 \times 10^{-2} + 6.51 \times 10^{-5} + 3.96 \times 10^{-12} \end{aligned}$$



```

jupyter Untitled Last Checkpoint: em 3 minutos (autosaved)
File Edit View Insert Cell Kernel Widgets Help
+ 3x Copy Paste Undo Redo Run Stop Refresh Code Download GitHub Binder
Memory: 845.5 MB / 8 GB

In [2]: using SoftPosit
In [3]: x = Posit32(1.0)
Out[3]: Posit32(1.0)
In [4]: y = x + Posit32(2^-23)
Out[4]: Posit32(1.0000001192092896)
In [5]: y = x + Posit32(2^-24)
Out[5]: Posit32(1.0000000596046448)
In [6]: y = x + Posit32(2^-25)
Out[6]: Posit32(1.0000000298023224)
In [7]: y = x + Posit32(2^-26)
Out[7]: Posit32(1.0000000149011612)
In [8]: y = x + Posit32(2^-27)
Out[8]: Posit32(1.0000000074505806)
In [9]: y = x + Posit32(2^-28)
Out[9]: Posit32(1.0)

```

Figure 4.1: Mantissa bits preview for Posit.

According to Theorem 3, the set of monomials are as follows:

$$\begin{aligned}\gamma_1 &= \alpha_1 10^{\beta_1} = 6.51 \times 10^{-5} \\ \gamma_2 &= \alpha_2 10^{\beta_2} = 3.96 \times 10^{-12} \\ \gamma_3 &= \alpha_3 10^{\beta_3} = 5.0 \times 10^{-2}.\end{aligned}$$

The set Ω is given by

$$\begin{aligned}\Omega_1 &= \beta_3 - \beta_1 = -2 - (-5) = 3 \\ \Omega_2 &= \beta_3 - \beta_2 = -2 - (-12) = 10.\end{aligned}$$

Since $\Omega_1 > 8$ (for Posit), γ_2 may be excluded without loss of accuracy. Fig. 4.2 present this operation using Python.

```

>>> import softposit as sp
>>> sp.posit32(0.05)+sp.posit32(6.651*10**-5)
0.05006650974974036
>>> sp.posit32(0.05)+sp.posit32(6.651*10**-5)+sp.posit32(3.96*10**-12)
0.05006650974974036

```

Figure 4.2: Example 11 was run on Python. The exclusion of monomial $\gamma_1 = 3.96 \times 10^{-12}$ does not change the final result.

As with floating-point arithmetic, the monomial exclusion proposal for Posit arithmetic will be illustrated for three systems: the Rössler system [74], the Lorenz equations [75] and Sprott B [76]. For each system, the reduced RK4 based on Posit arithmetic, here named as RPosit, according to Theorem 3, considering $\rho_{10} = 8$, is calculated.

The quality of RPosit is evaluated by means of the observability of the dynamical systems, computation of the largest Lyapunov exponent, simulation time and the reduction of the carbon footprint. The step size h is chosen according to the usual values found in the literature. In addition, all parameters used for the systems under study are shown in Table 3.1.

4.1 Rössler System

Let the Rössler system be represented by [74], according to equation (3.9).

Theorem 3 was applied to Posit, considering the parameters highlighted in Table 3.1.

The number of monomials for each equation of the Rössler system is presented in Table 4.1. Posit represents the discretization performed by the fourth-order Runge-Kutta method and RPosit represents the system reduced by the Posit.

Based on the complexity, as noted for floating point arithmetic, it can be concluded that y is more observable than x , which is more observable than z , that is, $y \triangleright x \triangleright z$. This statement is supported by the results in Table 4.1, which indicate that the variable y is more observable than x , which is more observable than z . Since variable z has the largest exclusion of terms, it does not seem to contribute much to the description of the system, which is consistent with the observability analysis in [44, 62].

Table 4.1: Number of monomials for each discretized equation for Lorenz, Rössler and Sprott B systems.

Equations	Rössler		Lorenz		Sprott B	
	Posit	RPosit	Posit	RPosit	Posit	RPosit
x_{k+1}	117	18	116	13	175	8
y_{k+1}	38	10	886	24	38	7
z_{k+1}	903	27	963	15	167	6

As another criterion for evaluating the results, the Lyapunov exponent was calculated as shown in Table 4.2. The Lyapunov exponent was calculated using the method proposed by Nepomuceno and Mendes [20]. For RPosit a value of 0.0914 was found.

Table 4.3 shows the number of operations necessary for each iteration to simulate each system. The Rössler equations have a total of 9080 operations per iteration when calculated using Posit, whereas

Table 4.2: Calculation of the Lyapunov exponent. The expected values were obtained in [77] for the Rössler and Lorenz equations, and in [76] for Sprott B. The unit of the Lyapunov exponent is nat/iter.

System	Literature	Posit	RPosit
Rössler	0.0900	0.0855	0.0914
Lorenz	1.5000	1.5443	1.4592
Sprott B	0.2100	0.2081	0.2372

RPosit has a total of 269 operations, representing a reduction of approximately 97.04% of operations performed per iteration.

Table 4.3: Computational complexity. The basic operations used in each system were analyzed, that is, Addition/Subtraction, Multiplication/Division and Power. For each system, all operators for the variables x, y, z are summed and the reduction is calculated.

		RK4			RPosit			Reduction
Operations		x_{k+1}	y_{k+1}	z_{k+1}	x_{k+1}	y_{k+1}	z_{k+1}	
Rössler	Sum/Subtraction	116	37	902	17	9	26	
	Multiplication/Division	527	141	4878	44	21	84	
	Power	192	47	2240	20	8	40	
	Summation of operators	835	225	8020	81	38	150	
	Total		9080			269		97.04%
Lorenz	Sum/Subtraction	115	885	962	12	23	14	
	Multiplication/Division	605	5260	5739	39	82	48	
	Power	289	2881	3190	16	30	21	
	Summation of operators	1009	9026	9891	67	135	83	
	Total		19926			285		98.57%
Sprott B	Sum/Subtraction	174	37	166	7	6	5	
	Multiplication/Division	726	139	710	17	11	10	
	Power	404	60	393	7	4	5	
	Summation of operators	1304	236	1269	31	21	20	
	Total		2809			72		97.44%

Table 4.4 shows the average simulation time for various iterations. As the number of iterations increases, the time also increases linearly for both Posit and RPosit. Hence, the simulation time can be related to the number of iterations using the equation for Posit:

$$t(k) = 0.0664k - 2.3902, \quad (4.1)$$

and for RPosit:

$$t(k) = 0.0022k + 0.0568, \quad (4.2)$$

where $t(k)$ represents the simulation time in seconds for k iterations.

Table 4.4: Average time in seconds of a hundred attempt to execute the proposed algorithm. Initial conditions, parameters and step size for each system are as follows: 1) Rössler: $(x_0, y_0, z_0) = (-1.0, 1.0, 1.0)$, $a = 0.15$, $b = 0.20$ and $c = 10.0$, and step-size of 10^{-2} ; 2) Lorenz: $(x_0, y_0, z_0) = (1.0, 0.5, 0.9)$, $\sigma = 16.0$, $\beta = 4.0$, $\rho = 45.92$, $h = 10^{-3}$; 3) Sprott B: $(x_0, y_0, z_0) = (0.05, 0.05, 0.05)$ and $h = 10^{-2}$.

Iteration	Rössler		Lorenz		Sprott B	
	Posit	RPosit	Posit	RPosit	Posit	RPosit
500	32.7201	1.1048	69.4772	1.2363	9.2679	0.2966
1000	65.6605	2.2123	137.7913	2.5067	18.6230	0.5860
1500	99.4124	3.4367	203.6965	3.7021	25.8499	1.0080
2000	129.3818	4.4892	275.7256	4.8307	37.4983	1.1437
2500	161.7611	5.8145	349.1869	6.1712	46.9779	1.4515
3000	197.3239	6.6830	410.4117	7.4568	57.0728	2.0481
3500	229.0141	7.7063	470.5010	8.9101	64.3457	2.0497
4000	259.6650	9.2248	545.3113	9.6726	73.5389	2.4020
4500	305.0579	10.2199	621.4583	10.6799	85.8576	2.8105
5000	327.1820	11.3886	690.6847	11.8862	95.5799	2.9853
5500	364.8461	12.0575	736.2958	13.7164	106.0473	3.4338
6000	390.2489	13.2096	801.7055	14.7777	112.7767	3.7445
6500	431.6851	14.8487	898.2372	15.8668	125.5166	4.0322
7000	460.0903	15.8024	945.9377	17.2628	134.0342	4.1393
7500	478.8445	16.9259	1031.7592	18.1602	144.6224	4.4603
8000	539.1163	18.2183	1101.5918	18.7389	154.4316	4.8905
8500	552.9877	19.0536	1153.0226	20.0133	160.3803	5.2086
9000	589.7269	20.0061	1228.4084	21.0094	173.5347	5.5616
9500	621.5443	21.1598	1302.0462	22.6732	180.6069	5.8339
10000	683.1889	22.3663	1370.7029	23.3023	190.5237	5.8952

With the estimates presented by equations (4.1) and (4.2), it is possible to observe that with RPosit there is a significant reduction in the simulation time, as a consequence of the reduction in the number of operations. And with the growing concern about climate factors, especially the carbon footprint, this reduction contributes to the development of more efficient algorithms from this point of view. Table 4.5 shows the reduction in carbon footprint, for the Rossler system there was a reduction of approximately 96.64%.

4.2 Lorenz Equations

Consider the Lorenz equations [75], described according to the equation (3.18).

Theorem 3 was applied using Posit, considering the parameters highlighted in Table 3.1. And, a $\rho_{10} = 8$ was considered. The number of monomials for each Lorenz equation is shown in Table 4.1. The reduction in the number of monomials was significant.

Table 4.5: Carbon Footprint for the systems of Rossler, Lorenz and Sprott B for the original and reduced systems using the Posit arithmetic.

		Time	Carbon Footprint	
			Decreased (%)	gCO_2e
Rössler	Posit	18 h 26 min	-	45.60
	RPosit	37 min	96.64%	1.53
Lorenz	Posit	37 h 56 min	-	156.70
	RPosit	38 min	99%	1.57
Sprott B	Posit	5 h 22 min	-	13.27
	RPosit	10 min	96.9%	0.412

For observability analysis, and based on the results in Table 4.1, variable x is more observable than variable y , which is more observable than variable z . This result is in accordance with previous findings [44, 62, 78] and indicates that variable z is the least observable due to symmetry considerations.

Table 4.2 brings the Lyapunov exponent, using Posit a value of 1.5443 was entered and for RPosit, the value entered was 1.4592 which is very close to that indicated in the literature of 1.5000 nat/iter [77].

In terms of computational complexity, the number of operations needed to simulate the Lorenz equations was calculated, as shown in Table 4.3. The Lorenz equations had 19926 and 285 mathematical operations per iteration when calculated using Posit and RPosit, respectively. This represents a reduction of 98.57% in the number of mathematical operations.

The average simulation time for different iterations is shown in Table 4.4. Just like in Rössler's system, it is observed that the time grows linearly as the number of iterations increases. Hence, the time-per-iteration relationship can be represented by the following equations for Posit (Equation (4.3)) and RPosit (Equation (4.4)), respectively:

$$t(k) = 0.1366k + 0.0493, \quad (4.3)$$

$$t(k) = 0.0023k + 0.3349, \quad (4.4)$$

where $t(k)$ is the simulation time in seconds for k iterations.

The carbon footprint of the systems was estimated using the Green Algorithms model and the online tool [36]. The simulation time used was based on the estimated value that will be used to construct the bifurcation diagram, approximately one million iterations. Hence, the time can be estimated using Equations (4.3) and (4.4). For the Lorenz equations, there was a significant reduction of approximately 99% in the carbon footprint, as shown in Table 4.5.

4.3 Sprott B

Consider the Sprott B system [76], according to equations (3.23). Theorem 3 was applied to Posit, considering the parameters highlighted in Table 3.1 and $\rho_{10} = 8$.

The number of monomials in each equation of the Sprott B system was reduced by approximately 94.47% using the Posit language and the proposed theorem, as shown in Table 4.1. The observability analysis performed in [44] revealed that the variable y is the most observable, followed by z and then x . This is also reflected in the results in Table 4.1.

The Lyapunov exponent for the Sprott B system was found to be 0.2081 nat/iter using Posit and 0.2372 nat/iter using RPosit, as shown in Table 4.2. These results are in good agreement with literature.

Table 4.3 shows the computational complexity of the Sprott B system. When discretized using Posit, it required 2809 operations, whereas when Theorem 3 was applied using RPosit, it required only 72 operations, resulting in a reduction of approximately 97.44% in the number of mathematical operations.

Table 4.4 shows the average simulation time for different iterations. As the number of iterations increases, the time grows linearly, as demonstrated by the relationships described by Equations (4.5) and (4.6). The time shown in Table 4.5 is an estimate for building the bifurcation diagram for one million iterations. Thus, the reduction in the carbon footprint was approximately 96.9%.

$$t(k) = 0.0193k - 1.3582, \quad (4.5)$$

$$t(k) = 0.0006k + 0.0237, \quad (4.6)$$

where $t(k)$ represents the simulation time in seconds for k iterations.

4.4 Discussion

As mentioned for floating-point arithmetic, it can be argued that the proposed computational discretization scheme does not improve the performance of classical Runge-Kutta numerical methods, since the removal of neglected terms in the method is performed automatically by the computer and, therefore, the advantage of not including them in the overall scheme has no effect on the results. Unfortunately, this is not the case, as the computer will also perform operations related to the neglected terms before returning zero. This certainly has an impact on overall performance, as shown here.

In the results presented for Posit arithmetic there was a drastic reduction in the number of monomials, resulting in a reduction in the number of operations and, consequently, in the reduction of simulation time, which contributes to the reduction of the carbon footprint, which is currently in focus. It is important to emphasize that the results obtained with the effective discretization maintained the characteristics of the systems, such as Lyapunov exponent and Observability.

CHAPTER 5

Conclusion

The focus of this thesis was the study of the fourth-order Runge-Kutta discretization method, an effective computational discretization method for the solution of nonlinear dynamic systems. Based on a theorem, it has been shown how to safely exclude monomials in an implemented Runge-Kutta of fourth-order with almost no accuracy loss. The reduced fourth-order Runge-Kutta using floating-point arithmetic (RRK4) has been successfully applied to three classical nonlinear dynamical systems: the Rössler equations, Lorenz equations, and Sprott B. The quality of simulations has been verified by means of the concept of observability of nonlinear dynamical systems, projections on the xy plane of the attractors, and computation of the largest Lyapunov exponent. The figures of attractors in the xy plane reinforce the efficiency of RRK4 by showing practically identical trajectories. To demonstrate the computational effectiveness of the proposed technique, the number of mathematical operations between RK4 and RRK4 was used for comparison. The reduction achieved is up to 81.8%. Consequently, simulation time has also significantly decreased, reaching 90.7% for the Lorenz case. It was shown that decreasing the step-size, the number of neglected terms increases, further reducing the number of operations performed. This is a striking outcome that allows the user of such systems to speed up simulations and save energy and time in many applications.

Later, the precision was modified for Posit arithmetic. It has been demonstrated how to delete monomials efficiently and with minimal loss of precision. The reduced fourth-order Runge-Kutta (RPosit) is applied to three classical nonlinear systems: the Rössler equations, the Lorenz equations, and Sprott B. A carbon footprint analysis was also included.

For comparison, the number of mathematical operations between Posit and RPosit was used to demonstrate the computational efficiency of Posit. As shown, a reduction of up to 98% was achieved. The relationship between simulation time and carbon footprint reduction is depicted, which shows that a decrease in the number of operations also resulted in a decrease in simulation time and a significant

reduction in the carbon footprint. This finding is promising and should be applied to other nonlinear dynamical systems and extended to different discretization schemes. It is also believed that the reduced discretizations would be beneficial for embedded applications, such as image encryption.

As future work, we plan to evaluate the use of the theorem in variable step-size discretization and in other discretization schemes. In addition, we plan to further optimize the algorithm introduced to make it more versatile and applicable to a wider range of nonlinear dynamic systems. In particular, we plan to develop a tool that can automatically calculate the carbon footprint reduction for any given nonlinear dynamic system. This would allow for a more efficient and environmentally-friendly approach to simulating and analyzing nonlinear dynamic systems. As we move towards a more sustainable future, tools like this will play an increasingly important role in reducing the carbon footprint of scientific and engineering simulations.

Bibliography

- [1] V. Ardourel and J. Jebeile, “On the presumed superiority of analytical solutions over numerical methods,” *European Journal for Philosophy of Science*, vol. 7, no. 2, pp. 201–220, 2017.
- [2] S. M. Hammel, J. A. Yorke, and C. Grebogi, “Do numerical orbits of chaotic dynamical processes represent true orbits?” *Journal of Complexity*, vol. 3, no. 2, pp. 136–145, 1987.
- [3] T. b. Sauer, C. d. Grebogi, and J. Yorke, “How long do numerical chaotic solutions remain valid?” *Physical Review Letters*, vol. 79, no. 1, pp. 59–62, 1997.
- [4] Z. Galias, “The Dangers of Rounding Errors for Simulations and Analysis of Nonlinear Circuits and Systems? and How to Avoid Them,” *IEEE Circuits and Systems Magazine*, vol. 13, no. 3, pp. 35–52, jan 2013.
- [5] R. Lozi, “Can we trust in numerical computations of chaotic solutions of dynamical systems?” in *Topology and dynamics of Chaos: In celebration of Robert Gilmore’s 70th birthday*. World Scientific, 2013, pp. 63–98.
- [6] X. Zhuang, Q. Wang, and J. Wen, “Numerical Dynamics of Nonstandard Finite Difference Method for Nonlinear Delay Differential Equation,” *International Journal of Bifurcation and Chaos*, vol. 28, no. 11, p. 1850133, oct 2018.
- [7] J. C. Butcher and N. Goodwin, *Numerical methods for ordinary differential equations*. Wiley Online Library, 2008, vol. 2.
- [8] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical mathematics*. Springer Science & Business Media, 2010, vol. 37.

-
- [9] I. Singh, B. Kaur, and K. K. Khun, “Simulating longitudinal vibrations of coupled oscillator using the fourth-order runge–kutta method by programming spreadsheet,” *European Journal of Physics*, vol. 40, no. 4, p. 045003, 2019.
- [10] A. Sinhababu and S. Ayyalasomayajula, “An improved dealiasing scheme for the fourth-order runge-kutta method: Formulation, accuracy and efficiency analysis,” *International Journal for Numerical Methods in Fluids*, vol. 93, no. 3, pp. 559–589, 2021.
- [11] M. Habibi, M. Safarpour, and H. Safarpour, “Vibrational characteristics of a fg-gplrc viscoelastic thick annular plate using fourth-order runge-kutta and gdq methods,” *Mechanics Based Design of Structures and Machines*, vol. 50, no. 7, pp. 2471–2492, 2022.
- [12] M. Shao, X. Zhao, J. Wu, Q. Wu, J. Qing, and X. Guo, “Nonlinear vibration of moving orthotropic films under oblique support,” *International Journal of Structural Stability and Dynamics*, p. 2350179, 2023.
- [13] J. Wang, X. Li, S. Shi, and X. Guo, “Multiscale nonlinear dynamics analysis of defective graphene reinforced pmma composite plates under aerodynamic pressure,” *Nonlinear Dynamics*, pp. 1–34, 2023.
- [14] H. M. Hasan, S. S. Alkhfaji *et al.*, “Nonlinear dynamic buckling behavior of axially loaded functionally graded graphene-enhanced composite laminated cylindrical shells in thermal conditions,” *Journal of Applied Nonlinear Dynamics*, vol. 12, no. 02, pp. 213–230, 2023.
- [15] E. Süli and D. F. Mayers, *An introduction to numerical analysis*. Cambridge university press, 2003.
- [16] S. Monaco and D. Normand-Cyrot, “A combinatorial approach of the nonlinear sampling problem,” *Lecture Notes in Control and Information Sciences*, vol. 144, pp. 788–797, 1990.
- [17] R. E. Mickens, “Nonstandard finite difference schemes for differential equations,” *The Journal of Difference Equations and Applications*, vol. 8, no. 9, pp. 823–847, 2002.
- [18] J. V. Lambers and A. C. Sumner, *Explorations in Numerical Analysis*. Hattiesburg: The University of Southern Mississippi, 2016.
- [19] E. G. Nepomuceno, “Convergence of recursive functions on computers,” *The Journal of Engineering*, vol. 2014, no. 10, pp. 560–562, 2014.

- [20] E. G. Nepomuceno and E. M. A. M. Mendes, “On the analysis of pseudo-orbits of continuous chaotic nonlinear systems simulated using discretization schemes in a digital computer,” *Chaos, Solitons & Fractals*, vol. 95, pp. 21–32, 2017.
- [21] E. G. Nepomuceno, P. F. Guedes, A. M. Barbosa, M. Perc, and R. Repnik, “Soft computing simulations of chaotic systems,” *International Journal of Bifurcation and Chaos*, vol. 29, no. 08, p. 1950112, 2019.
- [22] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to Interval Analysis*. Philadelphia: SIAM, 2009.
- [23] L.-S. Yao, “Computed chaos or numerical errors,” *arXiv preprint nlin/0506045*, 2005.
- [24] D. Faranda, M. F. Mestre, and G. Turchetti, “Analysis of Round Off Errors With Reversibility Test As a Dynamical Indicator,” *International Journal of Bifurcation and Chaos*, vol. 22, no. 09, p. 1250215, 2012.
- [25] A. Hasan, E. C. Kerrigan, and G. A. Constantinides, “Control-theoretic forward error analysis of iterative numerical algorithms,” *IEEE Transactions on Automatic Control*, vol. 58, no. 6, pp. 1524–1529, 2013.
- [26] T. Karimov, D. Butusov, and A. Karimov, “Comparison of analog and numerical chaotic system simulation,” in *2015 XVIII International Conference on Soft Computing and Measurements (SCM)*. IEEE, 2015, pp. 81–83.
- [27] E. G. Nepomuceno, S. A. Martins, B. C. Silva, G. F. Amaral, and M. Perc, “Detecting unreliable computer simulations of recursive functions with interval extensions,” *Applied Mathematics and Computation*, vol. 329, pp. 408–419, 2018.
- [28] Institute of Electrical and Electronics Engineers (IEEE), *754-2008 – IEEE standard for floating-point arithmetic*. IEEE, 2008, p. 1–58.
- [29] ———, “IEEE Standard for Floating-Point Arithmetic,” *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pp. 1–84, 2019.
- [30] M. L. Overton, *Numerical Computing with IEEE Floating Point Arithmetic*. Society for Industrial and Applied Mathematics : Philadelphia, jan 2001.
- [31] J. L. Gustafson, *The end of error: Unum computing*. Chapman and Hall/CRC, 2017.

- [32] J. L. Gustafson and I. T. Yonemoto, "Beating floating point at its own game: Posit arithmetic," *Supercomputing frontiers and innovations*, vol. 4, no. 2, pp. 71–86, 2017.
- [33] J. Hou, Y. Zhu, S. Du, and S. Song, "Enhancing accuracy and dynamic range of scientific data analytics by implementing posit arithmetic on FPGA," *Journal of Signal Processing Systems*, vol. 91, pp. 1137–1148, 2019.
- [34] I. Alouani, A. B. Khalifa, F. Merchant, and R. Leupers, "An investigation on inherent robustness of posit data representation," in *2021 34th International Conference on VLSI Design and 2021 20th International Conference on Embedded Systems (VLSID)*. IEEE, 2021, pp. 276–281.
- [35] R. Murillo, A. A. Del Barrio, and G. Botella, "Deep pensieve: A deep learning framework based on the posit number system," *Digital Signal Processing*, vol. 102, p. 102762, 2020.
- [36] L. Lannelongue, J. Grealey, and M. Inouye, "Green algorithms: quantifying the carbon footprint of computation," *Advanced science*, vol. 8, no. 12, p. 2100707, 2021.
- [37] B. He, Z. Deng, S. Huang, and J. Wang, "Application of unascertained number for the integration of carbon footprint in conceptual design," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 229, no. 11, pp. 2088–2092, 2015.
- [38] K. Chen, M. Yang, X. Zhou, Z. Liu, P. Li, J. Tang, and C. Peng, "Recent advances in carbon footprint studies of urban ecosystems: Overview, application, and future challenges," *Environmental Reviews*, vol. 30, no. 2, pp. 342–356, 2022.
- [39] IPCC, "Global warming of 1.5c. an ipcc special report on the impacts of global warming," Intergovernmental Panel on Climate Change, Tech. Rep., 2018.
- [40] United Nations Framework Convention on Climate Change, "Emissions gap report 2020," UNFCCC, Tech. Rep., 2020.
- [41] R. Jain, M. Radojevic, and R. Buyya, "Green cloud computing: Balancing energy in processing, data storage, and transport," *IEEE Transactions on Sustainable Computing*, vol. 1, no. 1, pp. 3–14, 2016.
- [42] H. Dai, J. Li, W. Li, Y. Li, and X. Wang, "A review of energy-efficient techniques for large-scale data centers," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 4, pp. 2689–2712, 2018.

-
- [43] L. Gao, W. Fan, and H. Chen, “Data center carbon emissions and energy-saving opportunities: A big data analytics perspective,” *IEEE Transactions on Sustainable Computing*, vol. 4, no. 2, pp. 192–204, 2019.
- [44] P. F. Guedes, E. M. Mendes, and E. Nepomuceno, “Effective computational discretization scheme for nonlinear dynamical systems,” *Applied Mathematics and Computation*, vol. 428, p. 127207, 2022.
- [45] —, “Posit Number Impact on Carbon Footprint Reduction in Simulation of NonLinear Dynamical Systems,” *IEEE Access*, vol. submitted, 2023.
- [46] L. H. A. Monteiro, *Sistemas dinâmicos, 3a ed.* Editora Livraria da Física: São Paulo, SP, 2011.
- [47] R. Lozi, “Can we trust in numerical computations of chaotic solutions of dynamical systems?” *Topology and Dynamics of Chaos*, pp. 63–98, 2013.
- [48] R. E. Moore and F. Bierbaum, *Methods and applications of interval analysis.* SIAM: Philadelphia, 1979, vol. 2.
- [49] R. H. N. Santiago, B. R. C. Bedregal, and B. M. Acióly, “Formal aspects of correctness and optimality of interval computations,” *Formal Aspects of Computing*, vol. 18, no. 2, pp. 231–243, 2006.
- [50] E. Nepomuceno and S. Martins, “A lower bound error for free-run simulation of the polynomial narmax,” *Systems Science & Control Engineering*, vol. 4, no. 1, pp. 50–58, 2016.
- [51] M. Rudolph-Lilith and L. E. Muller, “On a representation of the Verhulst logistic map,” *Discrete Mathematics*, vol. 324, pp. 19–27, 2014.
- [52] R. Gilmore and M. Lefranc, *The topology of chaos: Alice in stretch and squeezeland.* John Wiley & Sons, 2012.
- [53] E. M. A. M. Mendes and E. G. Nepomuceno, “A very simple method to calculate the (positive) Largest Lyapunov Exponent using interval extensions,” *International Journal of Bifurcation and Chaos*, vol. 26, no. 13, 2016.
- [54] M. T. Rosenstein, J. J. Collins, and C. J. De Luca, “A practical method for calculating largest Lyapunov exponents from small data sets,” *Physica D: Nonlinear Phenomena*, vol. 65, no. 1-2, pp. 117–134, 1993.

-
- [55] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, "Determining Lyapunov exponents from a time series," *Physica D: Nonlinear Phenomena*, vol. 16, no. 3, pp. 285–317, jul 1985.
- [56] H. Kantz, "A robust method to estimate the maximal Lyapunov exponent of a time series," *Physics Letters A*, vol. 185, no. 1, pp. 77–87, 1994.
- [57] E. Nepomuceno, S. Martins, G. Amaral, and R. Riveret, "On the lower bound error for discrete maps using associative property," *Systems Science & Control Engineering*, vol. 5, no. 1, pp. 462–473, 2017.
- [58] M. T. Rosenstein, J. J. Collins, and C. J. D. Luca, "A practical method for calculating largest Lyapunov from small data sets," *Physica D: Nonlinear Phenomena*, vol. 65, pp. 117–134, 1993.
- [59] C. Letellier, L. A. Aguirre, and J. Maquet, "Relation between observability and differential embeddings for nonlinear dynamics," *Physical Review E*, vol. 71, no. 6, p. 066213, 2005.
- [60] C. Letellier and L. A. Aguirre, "Interplay between synchronization, observability, and dynamics," *Physical Review E*, vol. 82, no. 1, p. 016204, 2010.
- [61] G. Gouesbet and J. Maquet, "Construction of phenomenological models from numerical scalar time series," *Physica D: Nonlinear Phenomena*, vol. 58, no. 1-4, pp. 202–215, 1992.
- [62] C. Letellier and L. A. Aguirre, "Investigating nonlinear dynamics from time series: The influence of symmetries and the choice of observables," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 12, no. 3, pp. 549–558, 2002.
- [63] C. Letellier, J. Maquet, L. Le Sceller, G. Gouesbet, and L. Aguirre, "On the non-equivalence of observables in phase-space reconstructions from recorded time series," *Journal of Physics A: Mathematical and General*, vol. 31, no. 39, p. 7913, 1998.
- [64] A. Quarteroni and F. Saleri, "Scientific computing with matlab and octave," *Computational Science and Engineering*, 2006.
- [65] M. A. Campos and P. F. Lima, *Introdução ao tratamento da informação nos Ensinos Fundamental e Médio*. Editora FGV, 2005, vol. 1.
- [66] M. T. Heath, *Scientific computing*. McGraw-Hill : New York, 2002.
- [67] R. F. Weber, *Fundamentos de arquitetura de computadores*. Sagra Luzzatto, 2000, vol. 1.

-
- [68] G. V. R. Viana, “Padrão IEEE 754 para aritmética binária de ponto flutuante,” *Revista CT*, vol. 1, no. 1, pp. 29–33, 1999.
- [69] J.-J. Fernandez, I. García, and E. M. Garzón, “Educational issues on number representation and arithmetic in computers: An undergraduate laboratory,” *IEEE Transactions on Education*, vol. 46, no. 4, pp. 477–485, 2003.
- [70] L. F. Richardson, “IX The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam,” *Philosophical Transactions of the Royal Society of London*, vol. 210, no. 459-470, pp. 307–357, 1911.
- [71] C. J. Freitas, “The issue of numerical uncertainty,” *Applied Mathematical Modelling*, vol. 26, pp. 237–248, 2002.
- [72] T. Hickey, Q. Ju, and M. H. V. Emden, “Interval Arithmetic : from Principles to Implementation,” *Journal of the ACM (JACM)*, 2001.
- [73] A. A. Del Barrio, N. Bagherzadeh, and R. Hermida, “Ultra-low-power adder stage design for exascale floating point units,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 13, no. 3s, pp. 1–24, 2014.
- [74] O. E. Rössler, “An equation for continuous chaos,” *Physics Letters A*, vol. 57, no. 5, pp. 397–398, 1976.
- [75] E. N. Lorenz, “Deterministic nonperiodic flow,” *Journal of the atmospheric sciences*, vol. 20, no. 2, pp. 130–141, 1963.
- [76] J. C. Sprott, “Some simple chaotic flows,” *Physical review E*, vol. 50, no. 2, p. R647, 1994.
- [77] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, “Determining Lyapunov exponents from a time series,” *Physica D: Nonlinear Phenomena*, vol. 16, no. 3, pp. 285–317, 1985.
- [78] L. A. Aguirre, S. B. Bastos, M. A. Alves, and C. Letellier, “Observability of nonlinear dynamics: Normalized results and a time-series approach,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 18, no. 1, p. 013123, 2008.
- [79] Q. Lai and S. Chen, “Generating multiple chaotic attractors from Sprott B system,” *International Journal of Bifurcation and Chaos*, vol. 26, no. 11, p. 1650177, 2016.

-
- [80] Q. Lai, G. Xu, and H. Pei, “Analysis and control of multiple attractors in Sprott B system,” *Chaos, Solitons & Fractals*, vol. 123, pp. 192–200, 2019.