Revista de Engenharias da Faculdade Salesiana
n. 6(2017)pp. 2-9

http://www.fsma.edu.br/RESA

Confiabilidade numérica na simulação de Sistemas Caóticos utilizando Matlab e C++

Wilson Rocha Lacerda Junior, Maurício Silva Barbosa,
Erivelton Geraldo Nepomuceno e Samir Angelo Milani Martins

***Abstract*—This work compares the numerical reliability in simulating chaotic systems using C++ and Matlab. The Lower Bound Error was used as a parameter to measure how reliable the simulations are. The chaotic systems of Lorenz and logistic map were used as case studies. With the results one could note that, as expected, C++ is faster than Matlab. However, this paper's main contribution is that, for the same precision (64 bits), C++ shows a more reliabe result in terms of the quality of the simulation in such a way that, accoringly to the proposed metric (simulation time), resulted in a gain up to 26% for the Lorenz system and up to 54% for the Logistic map, when compared to Matlab results.**

***Index Terms*—Discrete-time systems, chaos, numerical simulation, Lower Bound Error.**

***Resumo*—Esse trabalho compara a confiabilidade numérica na simulação de sistemas caóticos utilizando C++ e Matlab. A técnica do Limite Inferior do Erro foi utilizada como parâmetro para medir a confiabilidade de cada simulação. Os sistemas caóticos de Lorenz e Mapa Logístico foram usados como estudos de caso. Como foi possível observar, e o que é esperado na literatura, é que o C++ apresenta uma velocidade muito superior ao Matlab. Entretanto, o que este artigo traz de contribuição é que para usando a mesma precisão (64 bits), o C++ apresentou uma resultado superior na qualidade da simulação, que segundo a métrica proposta (tempo de simulação) resultou em um ganho de até 26% para o sistema de Lorenz e de 54% para o Mapa Logístico quando comparado aos resultados do Matlab.**

***Palavras-chave*—Sistemas discretos no tempo, caos, simulação numérica, limite inferior do erro.**

Lacerda Júnior, W. R.[1,2] e-mail: (wilson@outlook.com). Barbosa, M. S.[2] e-mail: (mauriciospta@gmail.com). Nepomuceno, E. G.[1,2] e-mail: (nepomuceno@ufsj.edu.br). Martins, S. A. M.[1,2] e-mail: (martins@ufsj.edu.br).

[1]GCOM - Grupo de Controle e Modelagem, Departamento de Engenharia Elétrica. URL: (http://www.ufsj.edu.br/gcom/)

[2]Universidade Federal de São João del-Rei, Praça Frei Orlando 170 - Centro, 36307-352, São João del-Rei, Minas Gerais, Brasil.

I. Introdução

Simulações numéricas desempenham um papel fundamental no estudo de sistemas dinâmicos não lineares [3, 4, 8, 19]. Para esse fim, os pesquisadores têm utilizado computadores e softwares cada vez mais acessíveis para obter as soluções de seus experimentos [14]. Por exemplo, tem-se o Matlab® como uma ferramenta frequentemente utilizada para obter soluções de sistemas dinâmicos não lineares devido à sua capacidade de computar equações complexas e processos iterativos por meio de uma linguagem de alto nível.

Outras ferramentas muito utilizadas são os ambientes de desenvolvimento em C++, que oferecem maior versatilidade para otimização dos códigos por parte do pesquisador [5], além de uma velocidade de processamento maior do que a do Matlab [1].

Diante disso, o trabalho de alguns autores [3] mostra que, devido à propriedades inerentes do computador, os resultados encontrados por meio de simulações numéricas são, em muitos casos, imprecisos. Além disso, Lozi [8] relata que há muitos trabalhos publicados em que a confiança numérica dos resultados não é devidamente verificada. O mesmo autor afirma que ”no caso simples de um sistema discreto dinâmico (do mapa de Hénon), há dúvidas quanto à natureza dos resultados computacionais: pseudo-órbitas instáveis longas ou atratores estranhos?”.

Uma das causas dos erros computacionais são relacionadas aos erros de arredondamento [18]. Além dos erros de arredondamento, há os erros introduzidos durante a integração de sistemas contínuos no tempo usando métodos numéricos. Esses erros se propagam na simulação dos sistemas e, consequentemente, diminuem a confiabilidade dos resultados obtidos [15]. Nesse contexto, Nepomuceno [12] mostrou que uma simples sequência de iterações do Mapa Logístico discreto pode gerar um resultado em regime permanente que não é o esperado teoricamente. Além disso, [6, 13] apontaram que o uso de diferentes métodos de discretização resultam em duas soluções numéricas diferentes.

Com a finalidade de investigar o erro numérico na simulação de sistemas complexos, Nepomuceno e Martins [11] introduziram um método para calcular o limite inferior do erro (LBE) baseado no fato de que extensões matemáticas equivalentes podem gerar respostas diferentes em uma simulação no computador. Entretanto, as análises não consideraram a influência da linguagem de programação na confiabilidade das soluções numéricas. E essa é a principal proposta desse trabalho: apresentar um método para comparar precisão e desempenho de diferentes linguagens de programação na simulação de sistemas caóticos e, assim, permitir a escolha daquela que garante uma maior confiabilidade numérica nos resultados. Isto é relevante devido à natureza de sensibilidade às condições iniciais desta classe de sistemas, em que pequenos erros numéricos podem conduzir a grandes divergências ao longo do tempo. Dessa forma, utilizando a técnica para o cálculo do LBE, foi realizada uma comparação da confiabilidade numérica na simulação de sistemas não lineares com comportamento caótico utilizando as linguagens de programação C++ e Matlab.

O restante do trabalho está organizado como se segue: na seção 2 são levantados os conceitos preliminares, suficientes para o entendimento do trabalho. Na seção 3 são apresentados os métodos utilizados para se chegar aos resultados discutidos na seção 4. A seção 5 trata da conclusão do trabalho e propostas de possíveis trabalhos futuros.

II. Conceitos Preliminares

Nesta seção são apresentados os conceitos teóricos necessários para a realização das análises deste trabalho.

A. *Extensão Intervalar*

Funções racionais reais de n variáveis reais possuem extensões intervalares "naturais"[10]. Dada uma expressão racional em variáveis reais, pode-se substituir as variáveis reais por variáveis de intervalo correspondentes e substituir as operações aritméticas por operações aritméticas intervalares para obter uma função de intervalo racional que é uma extensão natural da função racional real. Por outro lado, duas expressões racionais que são equivalentes em aritmética real, podem não ser equivalentes em aritmética intervalar. As definições 1 e 2 apresentam as condições para a equivalência no contexto de aritmética intervalar.

Definição 1. *(Extensão Intervalar- R. E. Moore. [10]) Seja f uma função da variável real x. Uma extensão intervalar de f é uma função de intervalo F de uma variável de intervalo X com a propriedade*

$$F(x) = f(x) \text{ para argumentos reais },$$

em que um intervalo é dado por um conjunto fechado de números reais $x \in \mathbb{R}$ tal que $X = [\underline{X}, \overline{X}] = \{x : \underline{X} \leq x \leq \overline{X}\}$.

Para um melhor entendimento, veja o exemplo 1:

Examplo 1. *Seja $x_{n+1} = f(x_n) = rx_n(1 - x_n)$. Alguns exemplos de extensão intervalar de f são:*

$$\begin{aligned} X_{n+1} = F(X_n) &= r(X_n(1 - X_n)), & (1)\\ &= rX_n - rX_n^2, & (2)\\ &= r(X_n - X_n^2), & (3)\\ &= rX_n - (rX_n)X_n. & (4)\end{aligned}$$

No exemplo 1, as equações 1-4 são matematicamente equivalentes, mas possuem diferentes sequências nas operações aritméticas básicas [17, 20].

Definição 2. *G e H são extensões intervalares equivalentes se*

$$G(X) = H(X) \text{ para argumentos intervalares.}$$

Considere o seguinte exemplo da definição 2.

Examplo 2. *Sejam as seguintes extensões intervalares:*

$$\begin{aligned} G(X) &= rX(1 - X)),\\ H(X) &= r(X(1 - X)),\\ L(X) &= rX - rX^2.\end{aligned}$$

Considerando $r = 10$ e $X = [0{,}3; 0{,}6]$, tem-se:

$$\begin{aligned} G([0{,}3;0{,}6]) &= 10[0{,}3;0{,}6](1 - [0{,}3;0{,}6])) = [1{,}2;4{,}2],\\ H([0{,}3;0{,}6]) &= 10([0{,}3;0{,}6](1 - [0{,}3;0{,}6])) = [1{,}2;4{,}2],\\ L([0{,}3;0{,}6]) &= 10[0{,}3;0{,}6] - 10([0{,}3;0{,}6])^2 = [-0{,}6;5{,}1].\end{aligned}$$

No exemplo 2.2, apenas $G(X)$ e $H(X)$ são extensões intervalares equivalentes.

B. *Órbitas e pseudo-órbitas*

Definição 3. *Uma órbita é uma sequência de de valores de um mapa, representada por $\{x_n\} = [x_0, x_1, \ldots, x_n]$.*

Definição 4. *Seja $i \in \mathbb{N}$ uma pseudo-órbita definida por uma condição inicial, uma extensão intervalar de f, um hardware, software, padrão de precisão e método de discretização específicos. Uma pseudo-órbita é uma aproximação de uma órbita e pode ser representada por*

$$\{\hat{x}_{i,n}\} = [\hat{x}_{i,0}, \hat{x}_{i,1}, \ldots, \hat{x}_{i,n}],$$

tal que

$$|x_n - \hat{x}_{i,n}| \leq \delta_{i,n}, \tag{5}$$

em que $\delta_{i,n} \in \mathbb{R}$ é o erro $\delta_{i,n} \geq 0$.

Uma pseudo-órbita define um intervalo no qual a verdadeira órbita se situa. Assim, um intervalo associado a cada valor de uma pseudo-órbita pode ser definido como:

$$I_{i,n} = [\hat{x}_{i,n} - \delta_{i,n}, \hat{x}_{i,n} + \delta_{i,n}]. \tag{6}$$

Das equações 5 e 6, tem-se que:

$$x_n \in I_{i,n}, \forall i \in \mathbb{N}. \tag{7}$$

Não há apenas uma pseudo-órbita, uma vez que existem diferentes hardwares, softwares, padrões de precisão numérica e métodos de discretização, que podem produzir diferentes resultados para cada extensão intervalar.

C. Sistemas caóticos e o expoente de Lyapunov

Desde o trabalho de Lorenz [7], sistemas dinâmicos caóticos têm sido extensivamente estudados. Entre as definições de caos disponíveis na literatura, uma bastante aceita é a proposta por [2].

Definição 5. *(Caos - J. Banks et. al. [2]). Seja $f : X \to X$ um sistema dinâmico. Esse sistema é caótico se possui três propriedades, a saber:*

1) *f é transitivo;*
2) *as órbitas periódicas de f são densas em X;*
3) *f é sensível as condições iniciais.*

A primeira condição implica que para quaisquer sub conjuntos de U e V em X, existe um número inteiro positivo k, tal que $f^k(U) \cap V$ é um conjunto não vazio.

A segunda condição trata-se da densidade de órbitas periódicas de f em X. Diz-se que V é denso em U se para qualquer ponto $u \,\epsilon\, U$ e para qualquer $\epsilon > 0$, há um ponto $v \,\epsilon\, V$, tal que $||u - v|| > \epsilon$.

Por último, entende-se que f é sensível às condições iniciais se h um número real positivo $\delta > 0$, tal que para qualquer ponto $x \in X$ e toda vizinhança N de x existe um ponto $y \in N$ e $n \in \mathbb{N}^+$, tais que $d(f^n(x), f^n(y)) > \epsilon$ onde d a distância em X.

No contexto de sistemas caóticos, o cálculo do expoente de Lyapunov (EL) é considerado uma importante solução para o problema de identificação de caos em sistemas dinâmicos [16]. O expoente de Lyapunov descreve a velocidade de fase com a qual dois pontos da mesma vizinhança no espaço fásico convergem ou divergem um do outro. Dessa forma, valores positivos indicam que o sistema possui comportamento caótico. Nesse trabalho, o calculo do EL é realizado utilizando extensões intervalares por meio do método proposto em [9].

D. Limite Inferior do Erro

Definição 6. *(Lower Bound Error, [11]) Sejam $\hat{x}_{a,n}$ e $\hat{x}_{b,n}$ duas pseudo-órbitas derivadas de duas extensões intervalares. Seja $\delta_{\alpha,n} = |\hat{x}_{a,n} - \hat{x}_{b,n}|/2$ o limite inferior do erro de um mapa $f(x)$, então $\delta_{a,n} \geq \delta_{\alpha,n}$ ou $\delta_{b,n} \geq \delta_{\alpha,n}$.*

A prova da definição 6 encontra-se em [11] e não será reproduzida aqui. A definição 6 estabelece que pelo menos uma das duas pseudo-órbitas deve ter um erro maior ou igual ao limite inferior do erro. Sendo assim, uma simulação deve ser parada caso o Limite Inferior do Erro seja maior que a precisão definida. Isso tem um significado prático. Se esse limite inferior do erro for maior do que a precisão requerida, não se deve prosseguir a simulação sem uma análise posterior. Para garantir a confiança na simulação, deve-se provar que uma das pseudo-órbitas garante a precisão estabelecida.

III. Metodologia

Inicialmente, definimos as linguagens de programação a serem comparadas no estudo a ser realizado. Matlab e C++ foram as linguagens escolhidas para comparação em razão de serem amplamente utilizadas para cálculos numéricos, sendo a primeira frequentemente empregada para estudos de sistemas caóticos [4].

Os modelos escolhidos para comparação foram as equações de Lorenz e Mapa Logístico, sendo que seus respectivos parâmetros foram definidos para que os sistemas apresentem comportamento caótico.

Para realizar as simulações foi utilizado um computador com processador I5 5200 2.20 GHz e sistema operacional Manjaro Linux KDE Edition, versão 17.0.2. Todas as rotinas [1] foram executadas nos softwares Matlab®, versão R2016a e NetBeans IDE, versão 8.2, utilizando o pacote de compiladores gcc 7.1.1-4. Destaca-se que os métodos RK4 e RK5 foram implementados para garantir uma equivalência entre as linguagens propostas, não sendo utilizadas funções ou bibliotecas pré-programadas. Além disso, é importante ressaltar que o compilador utilizado não manipula as operações aritméticas (o que poderia causar diferenças no desempenho), respeitando, assim, a forma com que cada extensão matemática foi proposta.

Para cada modelo proposto foram utilizadas duas extensões intervalares equivalentes implementadas em cada linguagem proposta. Para uma análise mais rigorosa, tanto o Runge Kutta de quarta ordem como o de quinta ordem (método de Butcher) foram utilizados para resolver as equações de Lorenz.Vale ressaltar que em todos os casos os sistemas foram simulados utilizando os mesmos parâmetros e condições iniciais, tanto em Matlab quanto em C++, bem como as mesmas extensões intervalares.

Como parâmetro de parada para um contexto no qual a simulação não satisfaz uma precisão predefinida, utilizou-se o seguinte critério, no qual $\varepsilon_{\alpha,n}$ é a precisão relativa na iteração n:

$$\varepsilon_{\alpha,n} = \frac{|\hat{x}_{a,n} - \hat{x}_{b,n}|}{|\hat{x}_{a,n} + \hat{x}_{b,n}|} \tag{8}$$

Como critério de parada foi utilizado $\varepsilon = 0{,}0001$ para as equações de Lorenz e $\varepsilon = 0{,}00001$ para o Mapa Logístico. A escolha desses valores se deve ao fato de serem pequenos, garantindo uma simulação rigorosa em relação aos erros computacionais. No entanto a escolha desses valores depende da aplicação e dos critérios de cada abordagem. Por fim, os modelos foram simulados de forma a obter o tempo de simulação que atenda à precisão predefinida e realizada uma análise comparativa de cada resultado alcançado.

[1] https://ufsj.edu.br/martins/rotinas.php

IV. Resultados

A. Equações de Lorenz utilizando RK4

Considere as equações de [7] definidas na Eq. 9 e sua respectiva extensão intervalar, dada pela Eq. 10:

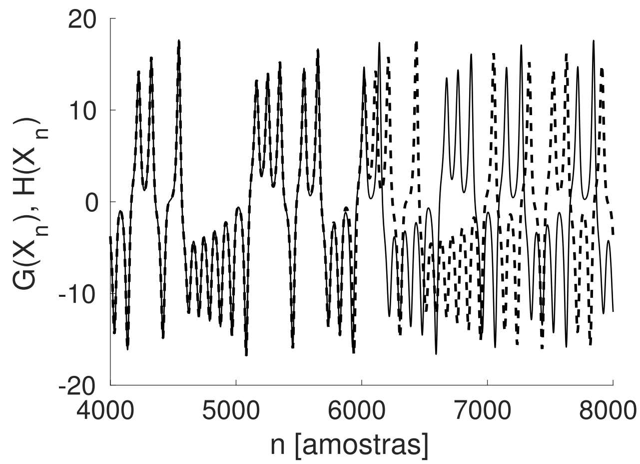
$$\begin{aligned}\frac{dx_l}{dt} &= \underline{\sigma(y_l - x_l)}\\ \frac{dy_l}{dt} &= \underbrace{x_l(\rho - z_l) - y_l}\\ \frac{dz_l}{dt} &= x_l y_l - \beta z_l\end{aligned} \tag{9}$$

$$\begin{aligned}\frac{dx_{l1}}{dt} &= \underline{\sigma y_{l1} - \sigma x_{l1}}\\ \frac{dy_{l1}}{dt} &= \underbrace{x_{l1}\rho - x_{l1} z_{l1} - y_{l1}}\\ \frac{dz_{l1}}{dt} &= x_{l1} y_{l1} - \beta z_{l1}\end{aligned} \tag{10}$$

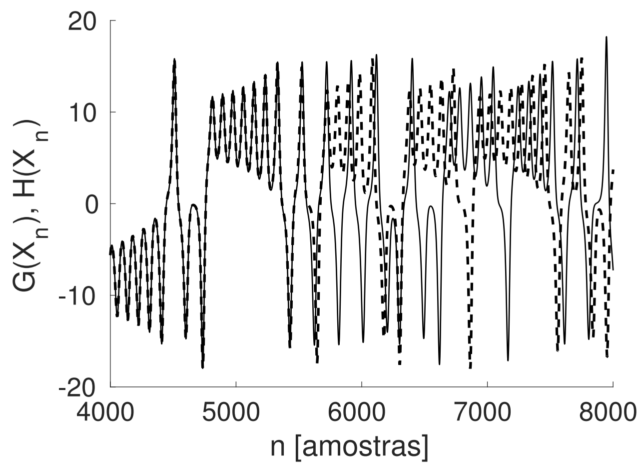
Como pode ser observado, as equações 9 e 10 são matematicamente equivalentes, diferindo uma da outra apenas nos termos sublinhados. Considere, então, a solução das equações de Lorenz dado $\sigma = 10$, $\rho = 28$ e $\beta = 8/3$, sendo estes os mesmos valores utilizados em [7], condições iniciais $(x_0,y_0,z_0) = (8,1,1)$ e passo de integração $h = 0{,}008$ para ambas as extensões intervalares. Dessa forma, verificando o número de iterações que satisfaz $\varepsilon = 0{,}0001$, foi estabelecido qual linguagem de programação garante a maior confiabilidade numérica numérica na simulação.

As figuras 1a e 1b mostram os resultados obtidos referente às variáveis x_l e x_{l1} das equações de Lorenz, considerando uma janela de simulação para $n \in [4000{,}8000]$. Como pode ser observado, apenas o fato das simulações terem sido executadas em softwares diferentes, fez com que a dinâmica obtida apresentasse diferenças. Além disso, é evidente a divergência entre os resultados considerando o mesmo software, porém extensões intervalares diferentes.

As figuras 2a e 2b mostram a evolução do erro relativo às pseudo-órbitas, na qual os valores estão em escala logarítmica. Como pode ser observado, o erro cresce exponencialmente e rapidamente se torna grande o suficiente para causar saídas muito diferentes, extrapolando o critério de confiabilidade previamente estabelecido. Desse forma, não se deve confiar nos resultados que não atendam o critério de parada predefinido, uma vez que não se sabe qual seria a trajetória verdadeira observando apenas as pseudo-órbitas computacionalmente obtidas. Considerando a critérios estabelecidos, o Matlab apresentou confiabilidade numérica até $n = 3530$, que equivale a 28,24 de simulação em LTU (unidade de tempo de Lorenz). A simulação em C++ garantiu uma confiabilidade até $n = 4465$, que equivale a 35,72 LTU, o que representa uma confiança de 26,5% maior em relação ao Matlab.



(a) C++.



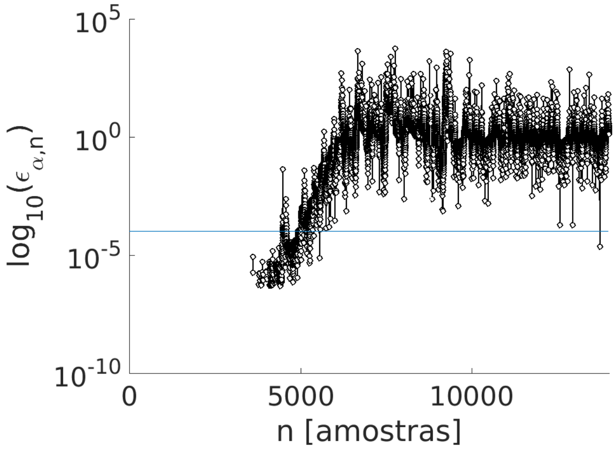
(b) Matlab.

Figura 1: Evolução temporal da simulação das equações 9 e 10 por meio do RK4. x_1 (-) e x_{11} (- -) foram obtidos utilizando as mesma condições iniciais ($(x_0,y_0,z_0) = (8,1,1)$), mesmos parâmetros ($\sigma = 10$, $\rho = 28$ e $\beta = 8/3$) e mesmo passo de integração ($h = 0{,}008$).

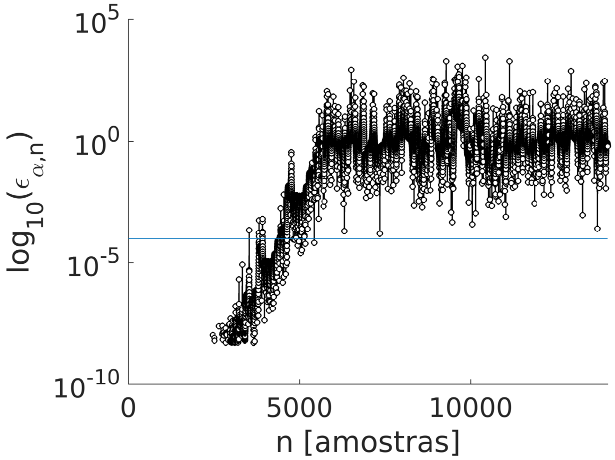
B. Equações de Lorenz utilizando RK5

De forma análoga ao caso anterior, o procedimento foi realizado substituindo o RK4 pelo RK5. A solução das equações de Lorenz utilizando RK5 também mostrou superioridade do C++ em relação ao Matlab, que garantiram, respectivamente, 36,41 LTU e 31,95 LTU dentro dos critérios propostos, significando uma vantagem para o C++ de 14,0%. As figuras 3a e 3b mostram os resultados obtidos referente às variáveis x_l e x_{l1} das equações de Lorenz, cujas solução foi obtida por meio do RK5.

As figuras 4a e 4b mostram a evolução do erro relativo às pseudo-órbitas. Como pode ser observado, a linguagem C++ mostrou-se mais apropriada, apesar da diferença ter diminuído em relação ao RK4. O tempo de processamento expressivamente melhor do C++ em relação ao Matlab

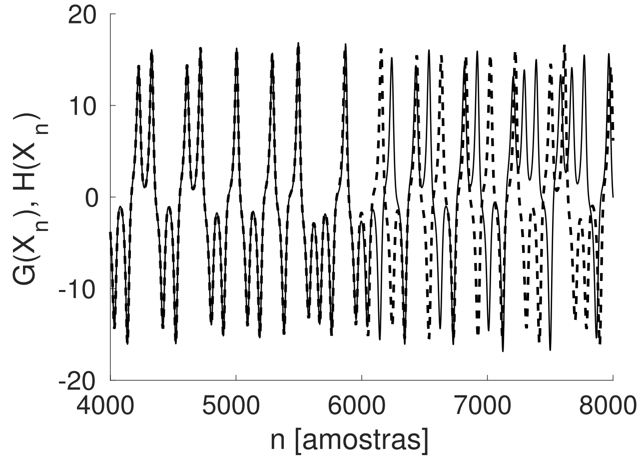


(a) C++.

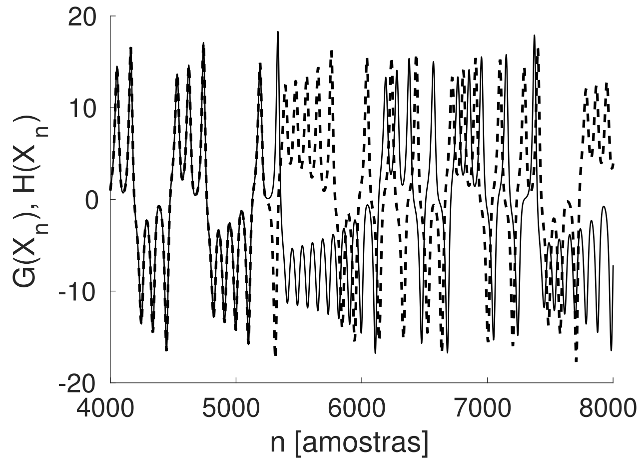


(b) Matlab.

Figura 2: Crescimento exponencial da diferença entre as duas pseudo-órbitas x_l e x_{l1}, comparado à precisão requerida $\varepsilon = 0{,}0001$. O erro relativo entre as pseudo órbitas se torna grande o suficiente para que não se possa confiar nos resultados a partir de determinada iteração, uma vez que não se sabe qual seria a trajetória verdadeira observando apenas as pseudo-órbitas computacionalmente obtidas.



(a) C++.



(b) Matlab.

Figura 3: Evolução temporal da simulação das equações 9 e 10 por meio do RK5. x_1 (-) e x_{11} (- -) foram obtidos utilizando as mesma condições iniciais ($(x_0,y_0,z_0) = (8,1,1)$), mesmos parâmetros ($\sigma = 10$, $\rho = 28$ e $\beta = 8/3$) e mesmo passo de integração ($h = 0{,}008$).

constatado em todos os casos, como mostra a tabela I, é justificado pelo fato do Matlab ser uma linguagem interpretada e o C++, compilada.

C. Mapa Logístico

Considere o Mapa Logístico representado pela Eq. 11 e sua respectiva extensão intervalar, dada pela Eq. 12

$$x_{1_{n+1}} = r x_{1_n}(1 - x1_n) \qquad (11)$$

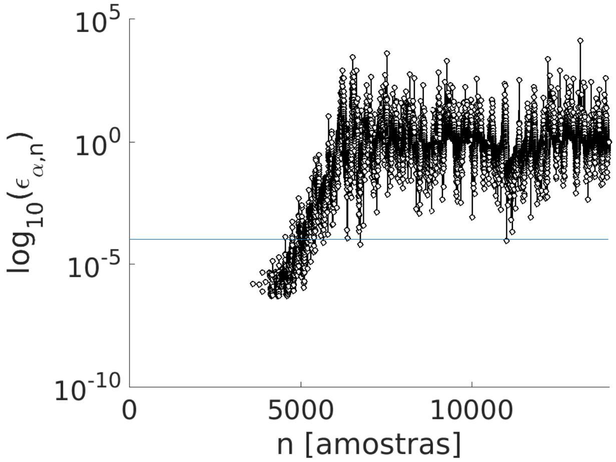
$$x_{11_{n+1}} = r x_{11_n} - r x_{11_n} x_{11_n} \qquad (12)$$

Como pode ser observado, as equações 11 e 12 são matematicamente equivalentes. Considere, então, a solução da equação do Mapa Logístico dado $r = 4$, e condições iniciais $(x(1)) = (0.8)$ para ambas as extensões intervalares. As figuras 5a e 5b mostram os resultados obtidos referente às variáveis x_l e x_{l1} das extensões intervalares do Mapa Logístico, considerando uma janela de simulação para $n \in [30{,}80]$.

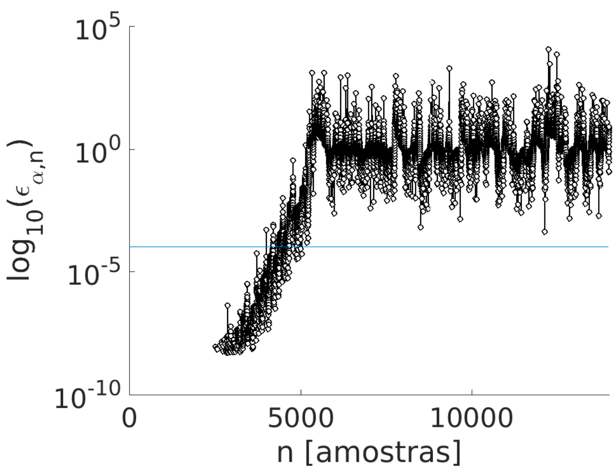
De forma análoga ao caso de Lorenz, considerando o número de iterações que satisfaz $\varepsilon = 0{,}00001$, foi verificado que a simulação em Matlab apresentou confiabilidade até $n = 35$, considerando a critérios estabelecidos. A simulação em C++ garantiu uma confiabilidade até $n = 54$, o que representa uma confiança de 54,3% maior em relação ao Matlab. As figuras 6a e 6b mostram a evolução do erro às pseudo-órbitas, na qual os valores estão em escala logarítmica.

Tabela I: Tempo de processamento.

Atrator de Lorenz	Matlab RK4		Matlab RK5		C++ RK4		C++ RK5	
	Eq. 9	Eq. 10	Eq. 9	Eq. 10	Eq. 9	Eq. 10	Eq. 9	Eq. 10
Tempo de simulação (s)	70,08	72,33	70,04	75,2	2	2	4	4



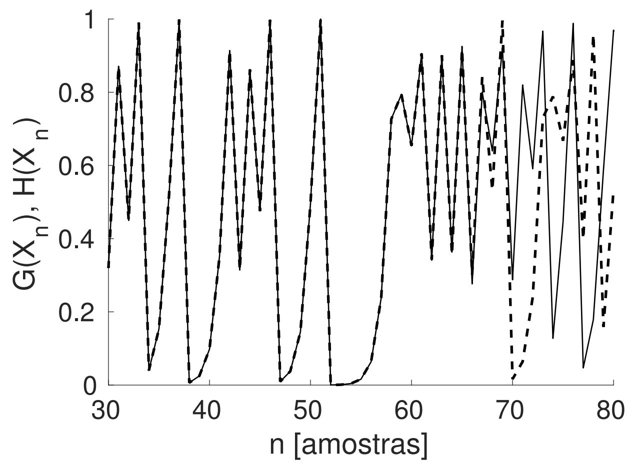
(a) C++.



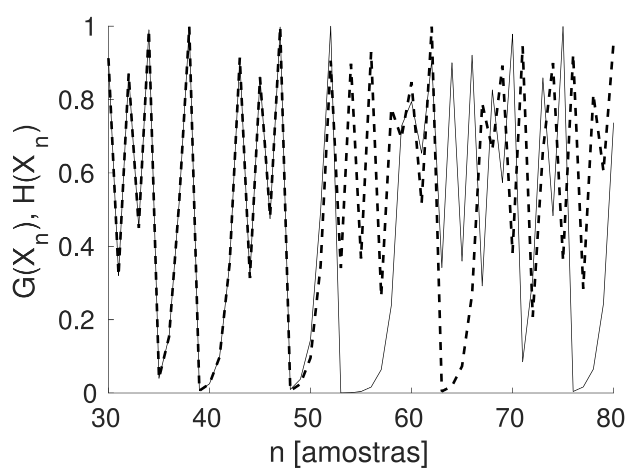
(b) Matlab.

Figura 4: Crescimento exponencial da diferença entre as duas pseudo-órbitas x_l e x_{l1}, como visto na figura 3, comparado à precisão requerida $\varepsilon = 0{,}0001$.

A tabela II apresenta os valores dos expoentes de Lyapunov calculados para cada caso. Como pode ser observado os expoentes obtidos são todos positivos, o que caracteriza os sistemas como caóticos. A Tabela III apresenta uma síntese dos resultados. Nos três casos estudados, utilizando a mesma precisão, ou seja, 64 bits, o C++ não apenas foi mais rápido, mas também apresentou um resultado qualitativo superior, chegando a cerca de 54% para o caso do Mapa Logístico. Interessante observar também, a variação do resultado para o método de discretização, o que certamente merece investigações futuras.



(a) C++.



(b) Matlab.

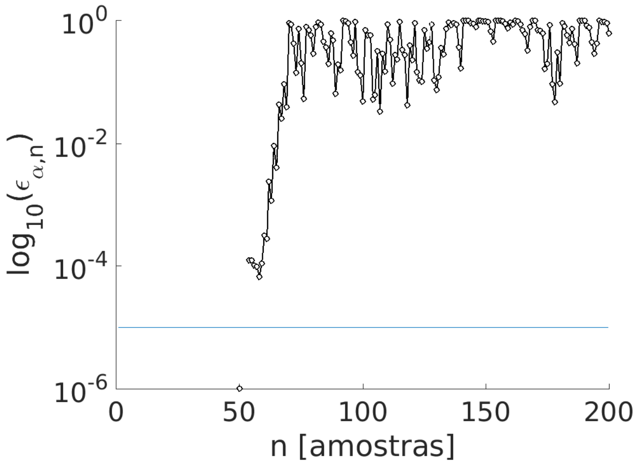
Figura 5: Evolução temporal da simulação das equações 11 e 12. x_1 (-) e x_{11} (–) foram obtidos utilizando as mesma condições iniciais ($x_1 = 0{,}8$), mesmo parâmetro ($r = 4$).

Tabela II: Resultado do cálculo do expoente de Lyapunov para os modelos propostos.

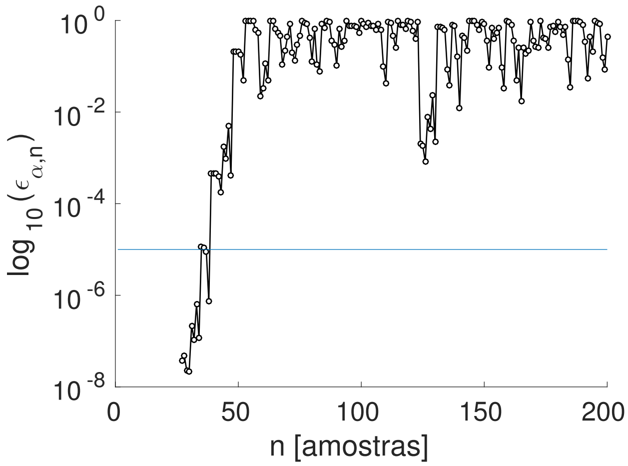
Sistema	Linguagem	Expoente de Lyapunov
Lorenz	Matlab RK4	0.8735
Lorenz	Matlab RK5	0.9419
Lorenz	C++ RK4	0.8207
Lorenz	C++ RK5	0.9620
Mapa Log.	Matlab	0.6652
Mapa Log.	C++	0.72921

V. Conclusão

Esse trabalho apresentou uma comparação entre as linguagens de programação Matlab e C++ no que con-



(a) C++.



(b) Matlab.

Figura 6: Resultado da simulação das equações 11 e 12. x_1 (-) e x_{11} (–) foram obtidos utilizando as mesma condições iniciais ($x_1 = 0{,}8$), mesmo parâmetro ($r = 4$).

Tabela III: Tempo de Simulação. Para o caso do Lorenz o tempo é medido em LTU, enquanto para o caso do Mapa Logístico está em número de iterações.

Sistema	Matlab	C++	Ganho do C++
Lorenz (RK4)	28,24	35,72	26,5%
Lorenz (RK5)	31,95	36,41	14,0%
Mapa Logístico	35	54	54,3%

cerne à confiabilidade numérica na simulação de sistemas dinâmicos não lineares com comportamento caótico.

Foram apresentados dois estudos de caso: análise das equações de Lorenz e do Mapa Logístico discreto. Para as equações de Lorenz, foram aplicados os métodos de discretização RK4 e RK5. Em todos os casos pode-se observar uma significativa superioridade da linguagem C++ em relação ao Matlab, apresentando uma robustez maior nas simulações, bem como um tempo de processamento muito menor.

Considerando RK4 e RK5, pode-se perceber que a diferença do tempo de simulação alcançado entre as duas linguagens de programação, dentro dos critérios estabelecidos, diminuiu. Contudo, visto que o Matlab apresenta um maior custo computacional devido à sua construção e resultados inferiores no que refere-se à confiabilidade, a linguagem C++ seria a mais indicada para simulação das equações de Lorenz e do Mapa Logístico.

Deve-se ressaltar, no entanto, que os resultados obtidos não podem ser generalizados, uma vez que não sabemos como o erro pode influenciar na resposta de um sistema caótico simulado no computador. Sendo assim, para a simulação de outros sistemas se faz necessária a aplicação do método proposto a fim de verificar qual seria a linguagem de programação mais adequada.

Para trabalhos futuros pretende-se aplicar o método proposto e comparar outros sistemas caóticos representados por meio de equações diferenciais, bem como utilizar outros métodos de discretização e diferentes linguagens de programação.

Agradecimentos

Os autores agradecem à UFSJ e às agências CAPES, CNPq e FAPEMIG pelo apoio.

Referências

[1] T. Andrews. Computation time comparison between Matlab and c++ using launch windows. 2012.

[2] J. Banks, J. Brooks, G. Cairns, G. Davis, and P. Stacey. On devaney's definition of chaos. *The American mathematical monthly*, 99(4):332–334, 1992.

[3] 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*, 13(3):35–52, 2013.

[4] S. M. Hammel, J. A. Yorke, and C. Grebogi. Do numerical orbits of chaotic dynamical processes represent true orbits? *Journal of Complexity*, 3(2):136–145, 1987.

[5] H. J. Lee and W. E. Schiesser. *Ordinary and partial differential equation routines in C, C++, Fortran, Java, Maple, and Matlab*. CRC Press, 2003.

[6] S. Liao. On the reliability of computed chaotic solutions of non-linear differential equations. *Tellus A*, 61(4):550–564, 2009.

[7] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 20(2):130–141, 1963.

[8] R. Lozi. *Can we trust in numerical computations of chaotic solutions of dynamical systems*, volume 84. World Scientific Series in Nonlinear Science Series A, 2013.

[9] E. 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*, 26(13):1650226, 2016.

[10] R. E. Moore. *Methods and applications of interval analysis*. SIAM, 1979.

[11] E. Nepomuceno and S. Martins. A lower bound error for free-run simulation of the polynomial narmax. *Systems Science & Control Engineering*, 4(1):50–58, 2016.

[12] E. G. Nepomuceno. Convergence of recursive functions on computers. *The Journal of Engineering*, 1(1), 2014.

[13] E. G. Nepomuceno and E. M. Mendes. On the analysis of pseudo-orbits of continuous chaotic nonlinear systems simulated using discretization schemes in a digital computer. *Chaos, Solitons & Fractals*, 95:21–32, 2017.

[14] E. Ott. *Chaos in dynamical systems*. Cambridge university press, 2002.

[15] S. Qin and S. Liao. Influence of round-off errors on the reliability of numerical simulations of chaotic dynamic systems. *arXiv preprint arXiv:1707.04720*, 2017.

[16] 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*, 65(1-2):117–134, 1993.

[17] M. Rudolph-Lilith and L. E. Muller. On a representation of the verhulst logistic map. *Discrete Mathematics*, 324:19–27, 2014.

[18] I. Sandberg. Floating-point-roundoff accumulation in digital-filter realizations. *Bell system technical journal*, 46(8):1775–1791, 1967.

[19] T. Sauer, C. Grebogi, and J. A. Yorke. How long do numerical chaotic solutions remain valid? *Physical Review Letters*, 79(1):59, 1997.

[20] M. Yabuki and T. Tsuchiya. Double precision computation of the logistic map depends on computational modes of the floating-point processing unit. *arXiv preprint arXiv:1305.3128*, 2013.