UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

CIÊNCIA DA COMPUTAÇÃO

MATEMÁTICA COMPUTACIONAL (6900/1)
ALGORITMO BASEADO EM LUT PARA CÁLCULO DA EXPONENCIAL DE EULER

Professor Dr. Airton Marco Polidório

Discentes:
Luca Mattosinho Teixeira RA 124316
Paula Fernandes Torres RA 123565

MARINGÁ
2024

1. INTRODUÇÃO

A busca por métodos eficientes e precisos na computação de funções matemáticas fundamentais é uma constante na área da Matemática Computacional, cujo principal objetivo é efetuar cálculos com precisão e menor custo de tempo possível. No entanto, alcançar esse objetivo muitas vezes é um desafio, já que as máquinas possuem limitações físicas e muitos problemas matemáticos, por natureza, exigem processamentos que extrapolam a capacidade de armazenar números muito grandes ou muito pequenos, os quais as máquinas enfrentam dificuldades para manipular e armazenar com precisão. Assim, os métodos existentes na Matemática Computacional servem para que seja possível contornar essas limitações. Essa busca constante por soluções mais eficientes é vital não apenas para o avanço da Matemática Computacional, mas também para sua aplicação prática em diversos campos onde cálculos precisos e rápidos são essenciais.

2. OBJETIVO

Neste trabalho iremos calcular a função exponencial de Euler através de um algoritmo baseado em Look-up Table (LUT), utilizando Nice Numbers. A implementação de *lookup tables*, ou tabelas de busca, surge como uma estratégia adicional para aprimorar a eficiência computacional. Ao pré-computar e armazenar valores de função em uma tabela, reduzimos a carga computacional durante a execução, agilizando consideravelmente o processo.

3. MÉTODO PARA CÁLCULO DA EXPONENCIAL DE EULER

Para a aproximação da função e^x, para qualquer x, utilizaremos a estratégia de reduzir o valor do argumento x, para conseguir computá-la utilizando valores

menores e assim atingir maior precisão em menor tempo. Sabemos que é possível calcular o valor de e^x através da série de Taylor:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots + \frac{x^n}{n!}$$

Entretanto, vimos em sala de aula que o erro propaga-se mais rápido no computador quando valores grandes são utilizados em multiplicações e divisões. Por isso, a estratégia é reduzir o valor do argumento através da seguinte expressão:

$$e^x = e^{\lfloor x \rfloor} \times e^{x-\lfloor x \rfloor}$$

Assim, temos que $\lfloor x \rfloor = k,\ k \in \mathbb{Z}$. Portanto, $\lfloor x \rfloor$ representa a parte inteira e $x - \lfloor x \rfloor$ representa a parte fracionária do número x. A resolução de e^k é separada em dois casos. Caso k seja par, temos que:

$$e^k = (e^{\frac{k}{2}})^2$$

Caso k seja ímpar:

$$e^k = e \times (e^{\frac{k-1}{2}})^2$$

E ainda, caso $x < 0$:

$$e^k = \frac{1}{e^k}$$

Agora para o cálculo da parte fracionária $e^{x-\lfloor x \rfloor}$, a maneira utilizada para aproximar o cálculo da exponencial de Euler é utilizando uma tabela de consulta, em inglês, Look-up Table (LUT) preenchida com Nice Numbers. O termo “Nice Number” geralmente é associado a valores que, ao serem utilizados em um contexto, servem de forma agradável e conveniente. Os Nice Numbers são representados como $2^{-n} + 1$, sendo $n \in \mathbb{Z}$. Uma LUT consiste em uma estrutura de dados utilizada para armazenar valores e associá-los a uma chave específica. No nosso caso, a chave será um número inteiro $x = 1, 2, \dots, n$ até que o Nice Number relacionado a n, ou seja, $2^{-n} + 1$ seja igual a 1. Os números associados a essa chave x serão:

- k, o valor do nice number $2^{-n} + 1$;

- $ln\ k$, o logaritmo natural do nice number k.

Após a construção da Look-up Table, para conseguirmos utilizá-la com o propósito de aproximar o cálculo da exponencial de Euler, precisamos fornecer um valor de argumento x ao algoritmo:

```python
def algoritmo(x, lut):
  k1 = int(x)
  k2 = x - k1

  if k1%2:
    res = Dec2IEEE(pow(exp(abs(k1)/2), 2))
  else:
    res = Dec2IEEE(exp(1)*(pow(exp((abs(k1)-1)/2), 2)))
  if k1 < 0:
    res = Dec2IEEE(1/res.x)

  j = 0
  xj = abs(k2)
  yj = 1
  tl = len(lut)
  while xj != 0 and j < tl:
    # Pesquisar na LUT o maior valor k tal que xj - k >= 0
    linha = lut[j]
    if xj >= linha[2]:
      max_k = linha[2]
      xj = xj - max_k
      yj = yj * linha[1]

    j += 1

  # Recuperação do resíduo
  yj = (1 + xj) * yj

  if k2<0:
    yj = 1/yj

  # O resultado aproximado é armazenado em yj
```

```
    yj *= res.x
    return yj
```

Observa-se que, assim como no procedimento feito por Bailey, dividimos o argumento em duas partes, uma inteira, que pode ser calculada de forma exata, e a parte fracionária, calculada com a LUT.

4. CÓDIGO FONTE

Python

```python
from IEEE754 import Dec2IEEE
from math import log, exp
import matplotlib.pyplot as plt

'''
MATEMÁTICA COMPUTACIONAL (6900/1)
ALGORITMO BASEADO EM LUT PARA CÁLCULO DA EXPONENCIAL DE EULER

Professor Dr. Airton Marco Polidório

Discentes:
Luca Mattosinho Teixeira RA 124316
Paula Fernandes Torres RA 123565
'''

# Intervalos para os valores de argumento
inicio = -110
fim = 110
passo = 1

# Função que gera a LUT com nice numbers
def cria_lut() -> [[]]:
  lut = []
  n = 1
  while True:
```

```
    nice_number = 2**(-n) + 1
    lut.append([n, nice_number, log(nice_number)])
    n += 1
    if nice_number == 1:
      break
  return lut

# Algoritmo aproximativo que calcula a exponencial de Euler
def algoritmo(x, lut):
  k1 = int(x)
  k2 = x - k1

  if k1%2:
    res = Dec2IEEE(pow(exp(abs(k1)/2), 2))
  else:
    res = Dec2IEEE(exp(1)*(pow(exp((abs(k1)-1)/2), 2)))
  if k1 < 0:
    res = Dec2IEEE(1/res.x)

  j = 0
  xj = abs(k2)
  yj = 1
  tl = len(lut)
  while xj != 0 and j < tl:
    # Pesquisar na LUT o maior valor k tal que xj - k >= 0
    linha = lut[j]
    if xj >= linha[2]:
      max_k = linha[2]
      xj = xj - max_k
      yj = yj * linha[1]

    j += 1

  # Recuperação do resíduo
  yj = (1 + xj) * yj

  if k2<0:
    yj = 1/yj
```

```
  # O resultado aproximado é armazenado em yj
  yj *= res.x
  return yj

# Declaração das listas que armazenam os valores necessários
valores_aproximados = []
valores_exatos = []
erros = []
argumentos = []

# Função que monta as listas com os resultados aproximados e
corretos da exponencial de Euler
def calculoValores():
  lut = cria_lut()

  for i in range(inicio,fim,passo):
    i /= 100
    x = i
    resultado_aproximado = algoritmo(x, lut)
    valores_aproximados.append(resultado_aproximado)
    valores_exatos.append(exp(x))
    erros.append(abs(exp(x)-resultado_aproximado))
    argumentos.append(x)

def graficoResultados():
  plt.plot(argumentos, valores_exatos, label='Exp(x)',
linestyle='-', color='red')
  plt.plot(argumentos, valores_aproximados, label='Aproximados',
linestyle='-', color='blue')
  plt.title('Valores obtidos da função exponencial de Euler com
LUT')
  plt.xlabel('Argumento')
  plt.ylabel('Valores')
  plt.legend()
  plt.grid(True)
  plt.show()
```

```
def graficoErros():
  plt.plot(argumentos, erros)
  plt.title('Análise de erros da função exponencial de Euler
com LUT')
  plt.xlabel('Argumento')
  plt.ylabel('Erro')
  plt.show()

# Chamadas das funções para calcular os valores e gerar os
gráficos dos erros e resultados
if __name__ == "__main__":
  calculoValores()
  graficoErros()
  graficoResultados()
```

O código foi feito na linguagem Python contendo como única dependência a biblioteca *matplotlib* que pode ser obtida para Windows ou Linux através do comando no terminal:

```
Unset
pip install matplotlib
```

5. AVALIAÇÃO DOS RESULTADOS

Os valores de e^x calculados utilizando o algoritmo implementado com LUT, referenciados pela legenda “Aproximados” no gráfico, foram comparados com os resultados obtidos pela função $exp(x)$, referenciados pela legenda “Exp(x)”:

Figura 1: valores obtidos para valores de argumento de -1 a 1.

Figura 2: valores obtidos para valores de argumento de -10 a 0.

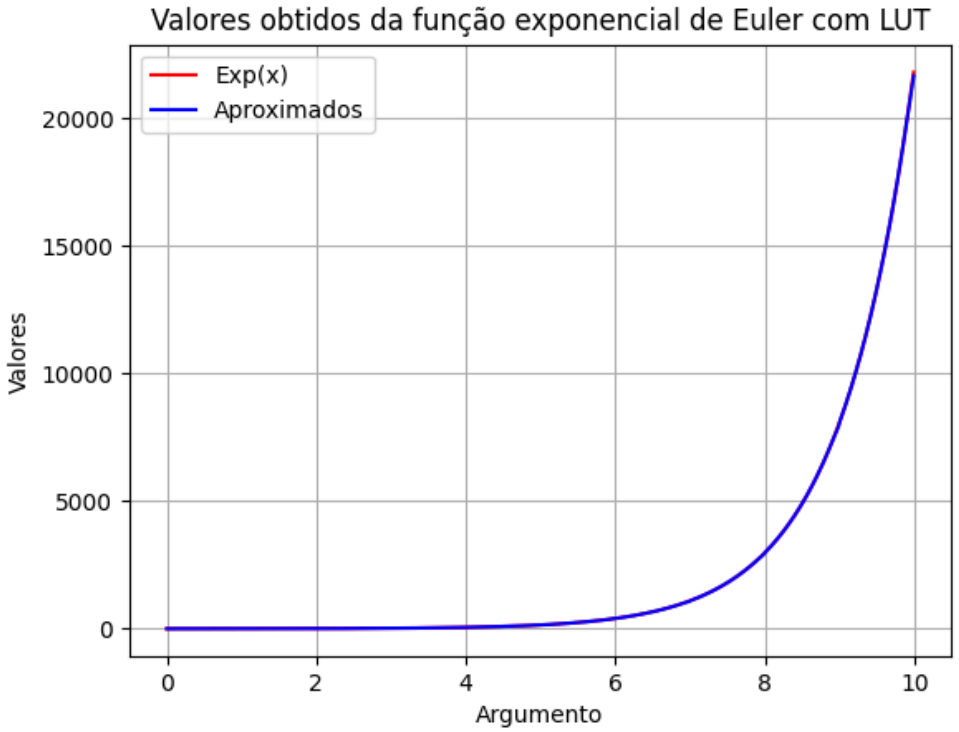

Figura 3: valores obtidos para valores de argumento de 0 a 10.

Nas Figuras 1, 2 e 3 é possível ver como a função utilizando o método de Bailey alcançou os mesmos resultados da função $exp(x)$. Nos gráficos é possível ver como as funções se sobrepõem, ou seja, como os valores aproximados seguem os valores corretos. Assim, é possível perceber como o método de Bailey fornece bons resultados até certos valores de argumento.

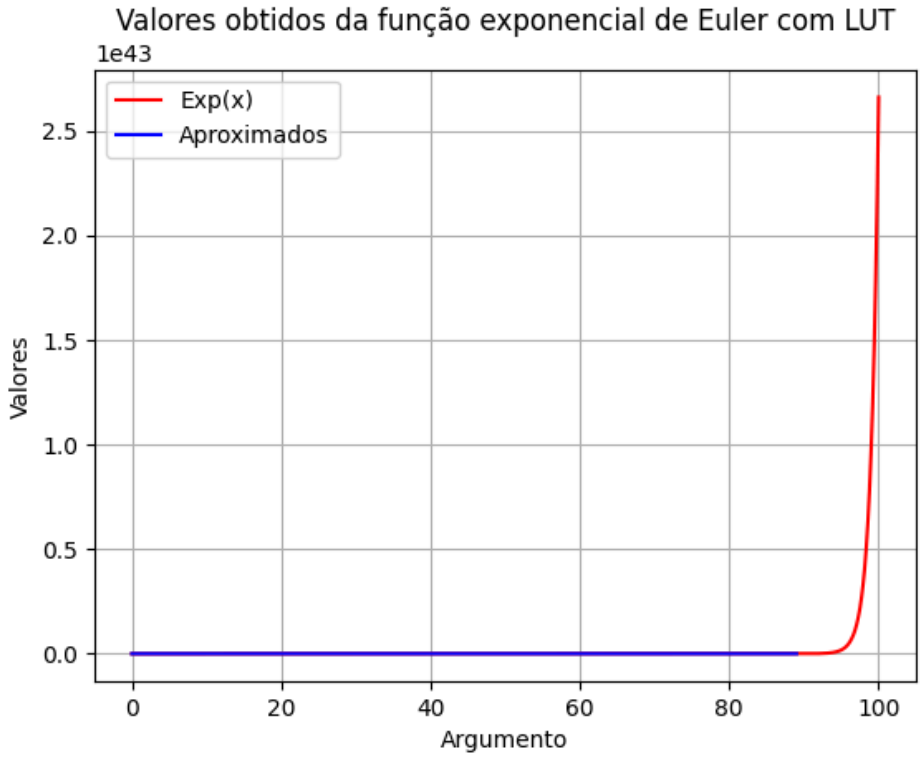

Figura 4: valores obtidos para valores de argumento de 0 a 100.

Na Figura 4 é visível como os resultados alcançados seguem os resultados da função $exp(x)$. Porém, quando e^x se torna muito grande, a linguagem Python utiliza uma notação de número infinito, *inf,* para os valores calculados pelo algoritmo. Sendo assim, é perceptível que os valores calculados pelo algoritmo com LUT não acompanham os valores da função $exp(x)$, porque enquanto $exp(x)$ continua crescendo, a função que utiliza o algoritmo com LUT somente retorna *inf,* que não é um número comparativo no gráfico.

6. AVALIAÇÃO DO ERRO

Os resultados obtidos através da análise do erro calculado entre os valores calculados pela função utilizando o algoritmo com LUT e a função $exp(x)$ são descritos através dos gráficos abaixo:

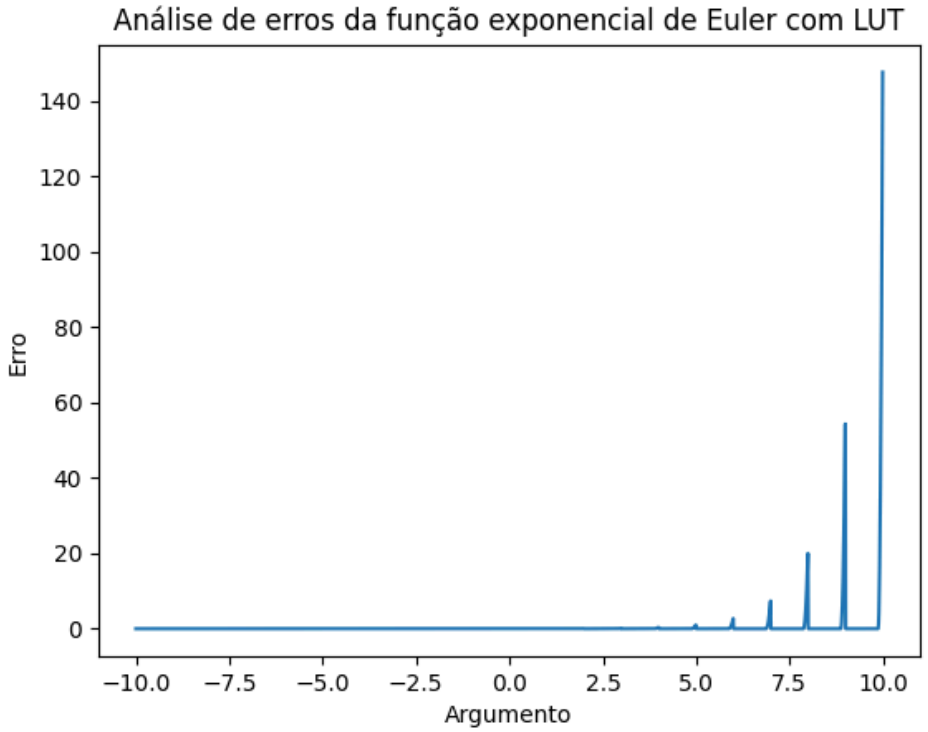

Figura 5: análise do erro para valores de argumento de -10 a 10.

O gráfico aparenta ter um aumento significativo do erro conforme o aumento do argumento, isso se dá pelo comportamento da função e^x e pela escala do gráfico. Como $\lim_{x \to \infty} e^x = \infty$, a linguagem Python define o valor de e^x como *inf* quando ele se torna muito grande. Além disso, como o $\lim_{x \to -\infty} e^x = 0$, e^x é calculado como 0 para valores menores de argumento. Portanto, conforme o argumento tende a ficar menor, o erro calculado sempre é 0, pois a função do algoritmo com LUT e a função $exp(x)$ retornam 0 para e^x. Já quando o argumento tende a ficar maior, o erro é calculado pela diferença entre o resultado obtido pela função $exp(x)$ e a função aproximativa, e assim o erro é maior do que zero. No gráfico, essa diferença aparenta ser grande justamente porque para valores menores de argumento, e^x é sempre zero e portanto o erro também é sempre 0. Nos gráficos abaixo é possível ver os valores detalhados para alguns intervalos pela mudança de escala:

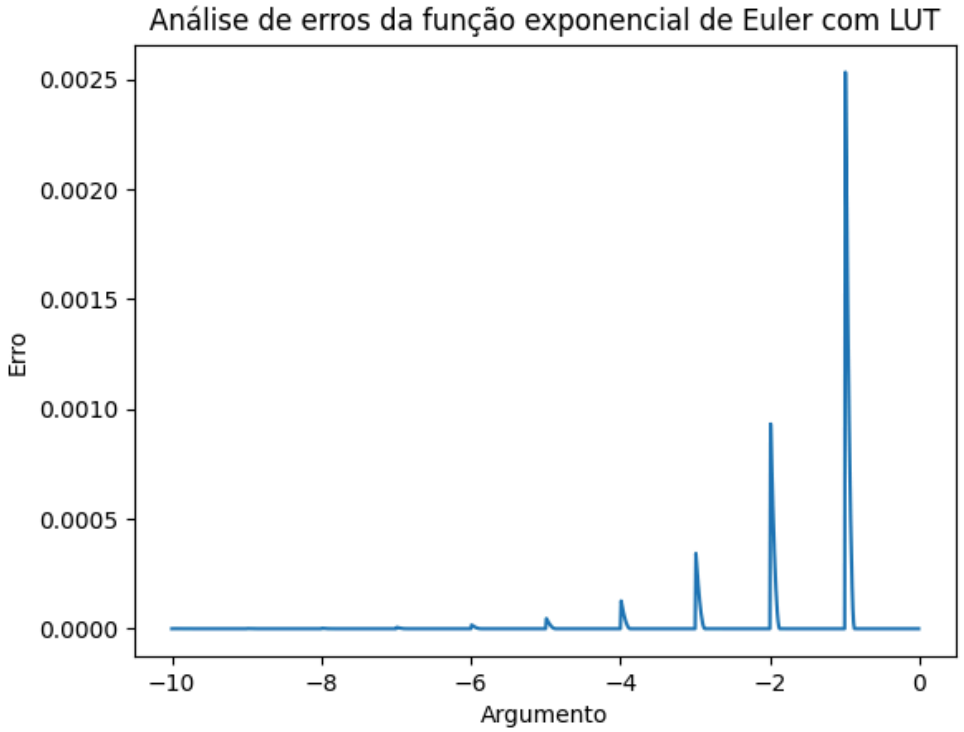

Figura 6: análise do erro para valores de argumento de -10 a 0.

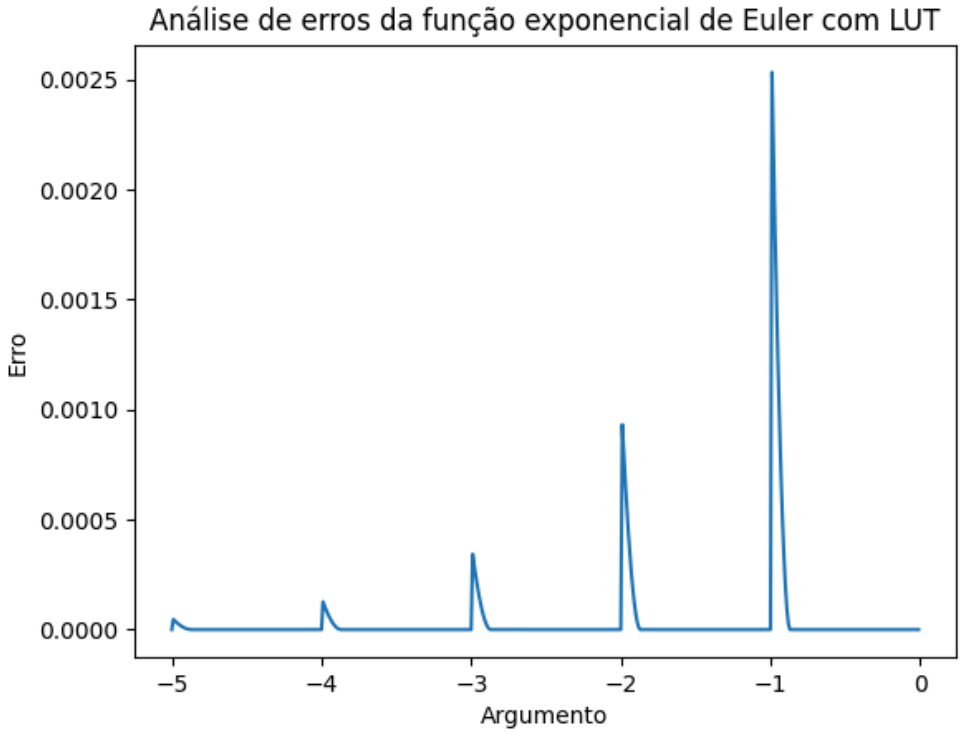

Figura 7: análise do erro para valores de argumento de -5 a 0.

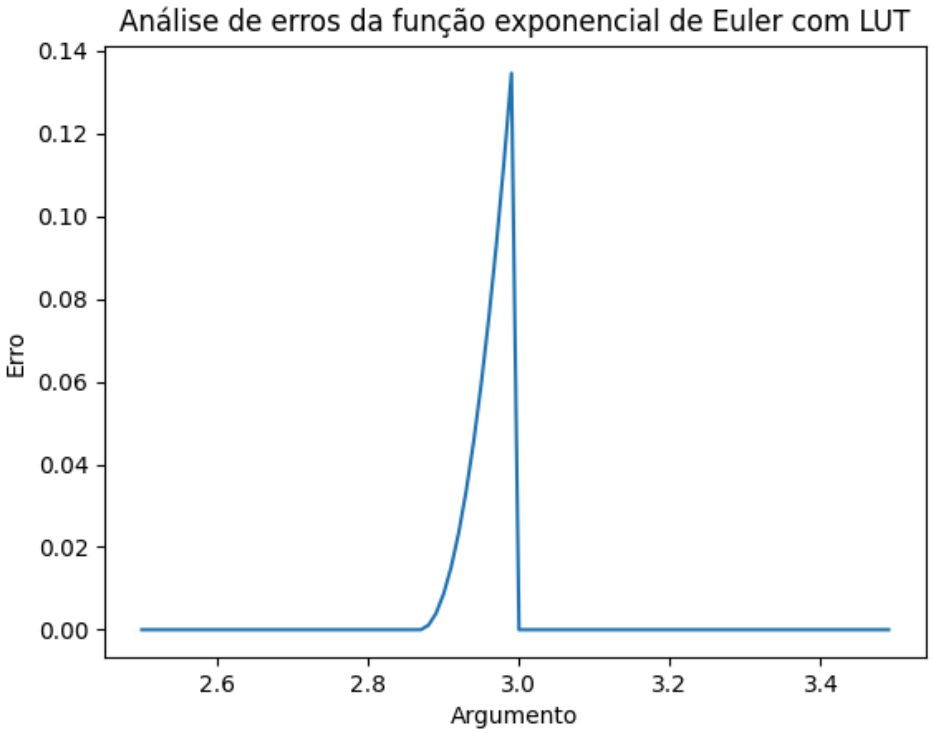

Figura 8: análise do erro para valores de argumento de 2.6 a 3.4.

As Figuras 6, 7 e 8 mostram o comportamento da propagação do erro, em que em determinados pontos o erro se propaga para mais casas decimais rapidamente, gerando esses aumentos dramáticos nos gráficos. Este comportamento é explicado no gráfico abaixo, e ocorre pela limitação dos valores disponíveis na LUT:

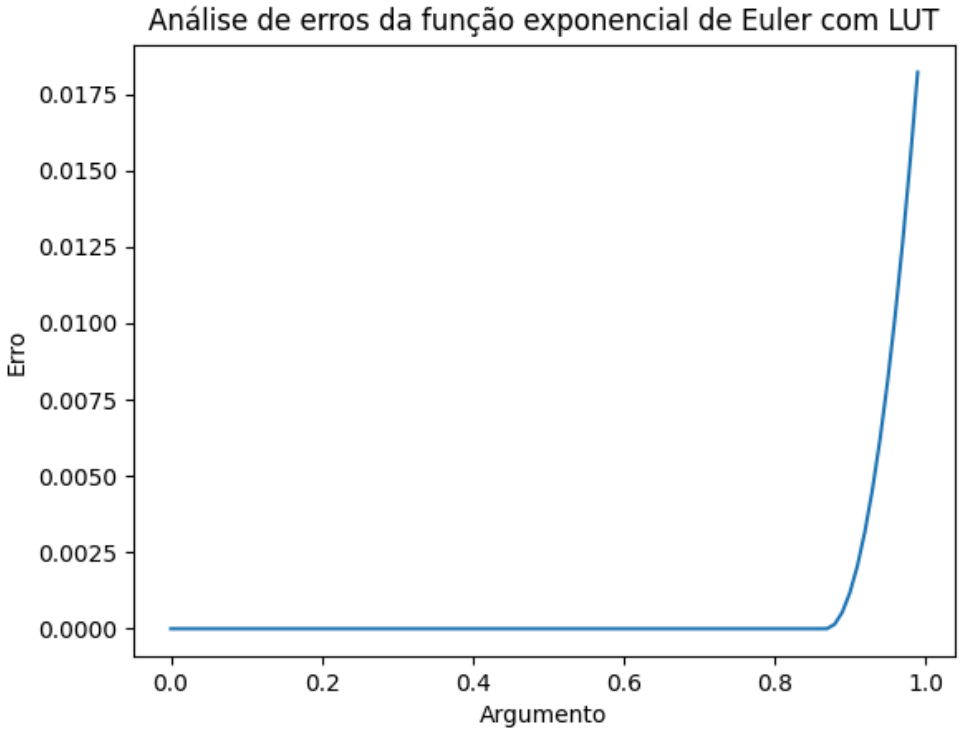

Figura 9: análise do erro para valores de argumento de 0 a 1.

Na Figura 9 é visível que o erro passa a se propagar rapidamente para mais casas decimais conforme o valor do argumento se aproxima de 1. Isto acontece sempre que o valor de argumento tende a um valor inteiro, como pode ser visto nas Figuras 6, 7 e 8. Como a construção da LUT é feita utilizando Nice Numbers e é limitada pelo tamanho de ponto flutuante, a precisão alcançada é reduzida para valores de argumento em pontos extremos que são anteriores a um número inteiro, como o 0,9.

7. CONCLUSÃO

Em conclusão, a redução do argumento utilizando a implementação de um algoritmo que faz uso de Look-up Table com Nice Numbers resultou em um método eficiente para calcular a exponencial de Euler para uma gama de valores de argumento. No geral, essa abordagem atendeu ao objetivo de efetuar os cálculos da exponencial de Euler com precisão, gerando bons resultados para os erros cometidos em maior parte dos valores de argumento.

Os resultados alcançados com o algoritmo aproximativo correspondem aos resultados obtidos pela função exponencial de Euler, ainda que os resultados sejam muito grandes, pois caso seja necessário é possível calcular o valor retornado representado somente como *inf* pelo Python e compará-lo ao $exp(x)$. Porém, a precisão alcançada é reduzida quando o valor do argumento tende a um número inteiro. Essa redução da precisão é ocasionada pelo tamanho da LUT, que é limitada ao atingir o valor 1.

8. REFERÊNCIAS

[1] American National Standards Institute / Institute of Electrical and Electronics Engineers: IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std 754-1985, New York, 1985.