UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

CIÊNCIA DA COMPUTAÇÃO

MATEMÁTICA COMPUTACIONAL (6900/1)
MÉTODO DE BAILEY 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

No âmbito deste trabalho, iremos calcular a função exponencial de Euler através do Método de Bailey, destacando como as séries de Taylor são empregadas para aproximar o valor dessa função elementar e como a redução do argumento colabora para que erros causados ao multiplicar números no padrão IEEE-754 não se propaguem com tanta facilidade.

3. MÉTODO DE BAILEY 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:

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

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 utilizada por Bailey, 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 $e^{x - \lfloor x \rfloor}$, o método de Bailey prevê a seguinte expressão:

$$e^{x} = 2^{n} \times (e^{r})^{256}$$

Em que e^{r} é calculado através da série de Taylor apenas até o quarto termo, e as constantes r e n são dadas como:

$$r = \frac{x - n \times \ln 2}{2}$$

$$n = \left\lceil \frac{x - \ln 2}{\ln 2} \right\rceil$$

$$e^{r} = 1 + \frac{r}{1!} + \frac{r^{2}}{2!} + \frac{r^{3}}{3!} + \frac{r^{4}}{4!}$$

4. CÓDIGO FONTE

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

'''
MATEMÁTICA COMPUTACIONAL (6900/1)
MÉTODO DE BAILEY PARA CÁLCULO DA EXPONENCIAL DE EULER

Professor Dr. Airton Marco Polidório

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

# Constantes dos intervalos para os valores de argumento
ln2 = log(2)
inicio = -200
fim = 0
passo = 1

# Função para obter a fração da mantissa
def fracao_da_mantissa(mantissa_binaria):
  decimal = 0.0
  expoente = -1

  for bit in mantissa_binaria:
    if bit == '1':
      decimal += 2**expoente
    expoente -= 1

  return decimal

def decimal_para_binario(numero_decimal):
  if numero_decimal == 0:
    return '0'
```

```
  binario = ''
  while numero_decimal > 0:
    resto = numero_decimal % 2
    binario = str(resto) + binario
    numero_decimal //= 2

  return binario

def mantissa_ieee(num: Dec2IEEE):
  bin = decimal_para_binario(num.Fbits.f)
  mantissa = 1 + fracao_da_mantissa(bin)
  return mantissa

# Método de Bailey que calcula a exponencial de Euler
def bailey(x: Dec2IEEE):
  k1 = int(x.x)
  k2_sinal = x.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)

  k2 = abs(k2_sinal)
  n = ceil((k2 - (ln2/2))/ln2)

  r = (k2 - (n * ln2)) / 256
  expr = (1 + r * (1 + r * (0.5 + r * (0.16666666666666666 +
(0.041666666666666664 * r)))))
  expr256 = Dec2IEEE(pow(expr,256))
  expr256.Fbits.e += n

  if k2_sinal<0:
    expr256.x = 1/expr256.x

  expx = expr256.x * res.x

  return expx
```

```python
# Declaração das listas que armazenam os valores necessários
argumentos = []
corretos = []
aproximados = []
erros = []

# Função que monta as listas com os resultados aproximados e
corretos da exponencial de Euler
def calculo_valores():
  for i in range(inicio,fim,passo):
    i /= 100
    num = Dec2IEEE(i)
    num.x = round(num.x, 10)
    aproximado = bailey(num)
    correto = exp(num.x)
    argumentos.append(num.x)
    corretos.append(correto)
    erros.append(abs(correto - aproximado))
    aproximados.append(aproximado)

def grafico_resultados():
  plt.plot(argumentos, corretos, label='Corretos', linestyle='-',
color='red')
  plt.plot(argumentos, aproximados, label='Aproximados',
linestyle='-', color='blue')
  plt.title('Valores obtidos da função exponencial de Euler')
  plt.xlabel('Argumento')
  plt.ylabel('Valores')
  plt.legend()
  plt.grid(True)
  plt.show()

def grafico_erros():
  plt.plot(argumentos, erros, label='Erro')
  plt.title('Análise de Erros da Aproximação de Bailey')
  plt.xlabel('Argumento')
  plt.ylabel('Erro')
  plt.legend()
```

```
    plt.grid(True)
    plt.show()

if __name__ == "__main__":
    calculo_valores()
    grafico_erros()
    grafico_resultados()
```

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 método de Bailey, referenciados pela legenda “Aproximados” no gráfico, foram comparados com os resultados obtidos pela função $exp(x)$, referenciados pela legenda “Corretos”:

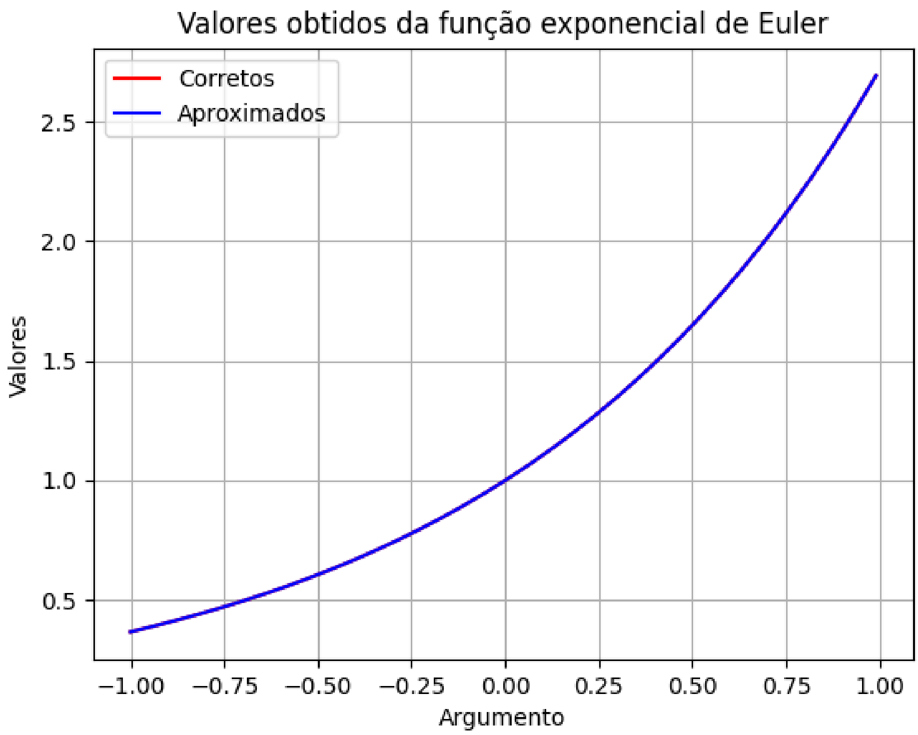

Figura 1: valores obtidos de e^x para valores de argumento de -1 a 1.

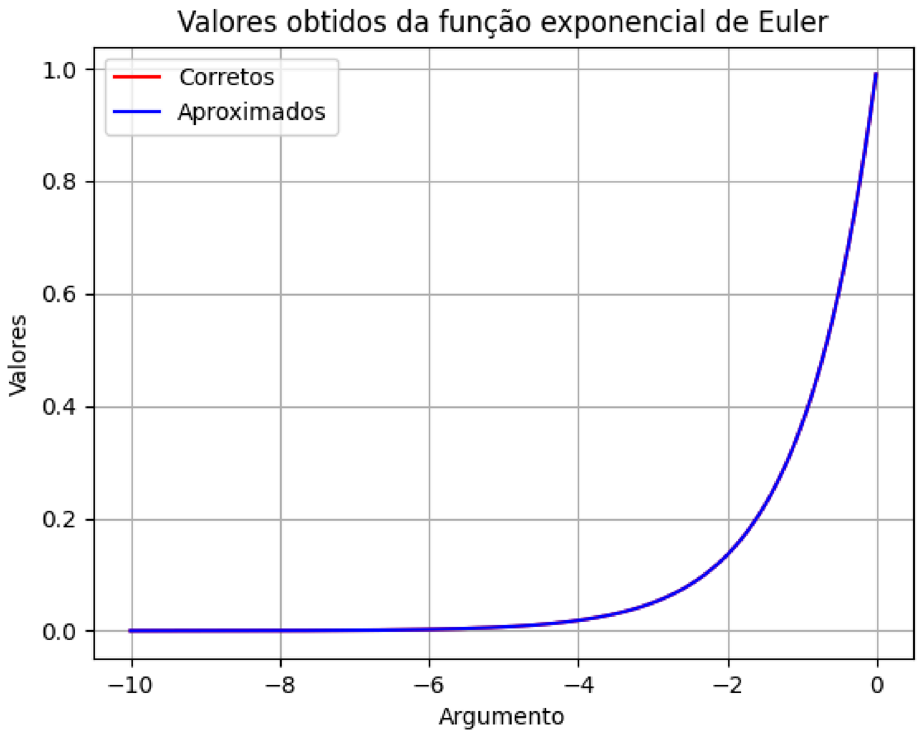

Figura 2: valores obtidos de e^x para valores de argumento de -10 a 0.

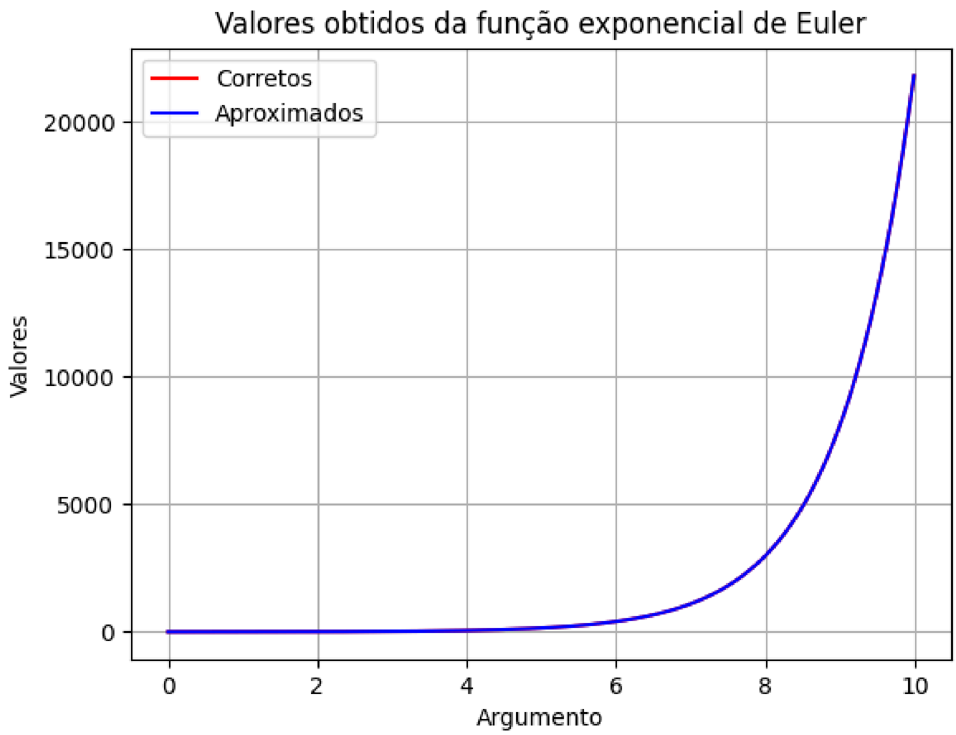

Figura 3: valores obtidos de e^x para valores de argumento de 0 a 10.

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. E assim é possível perceber como o método de Bailey fornece bons resultados até certos valores de argumento.

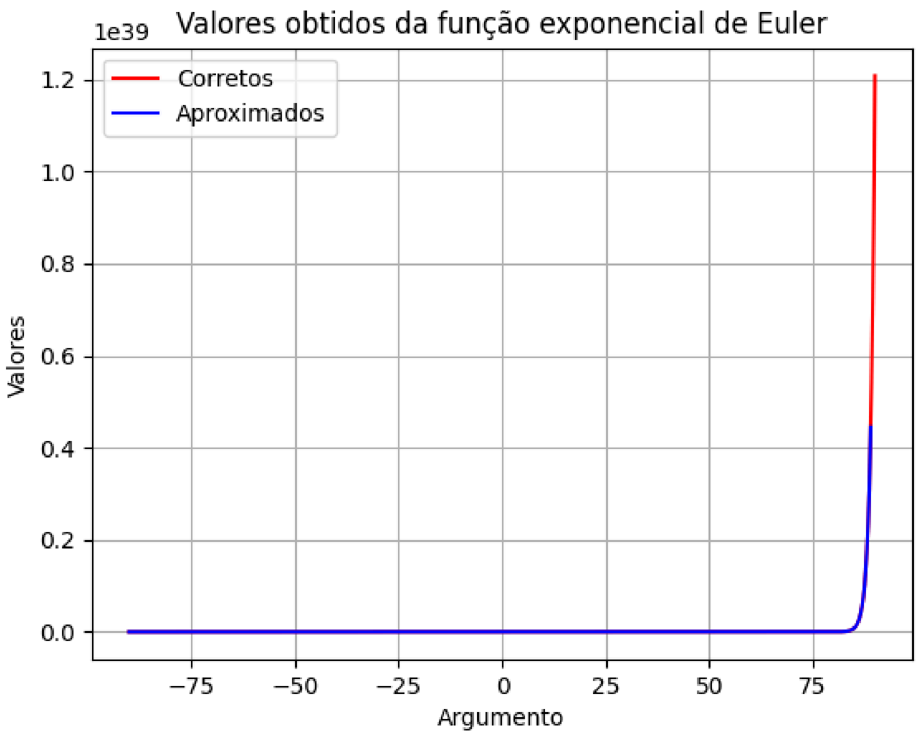

Figura 4: valores obtidos de e^x para valores de argumento de -100 a 100.

Na Figura 4 é visível como os resultados alcançados utilizando o método de Bailey 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 método de Bailey. Sendo assim, é perceptível que os valores calculados pelo método de Bailey não acompanham os valores da função $exp(x)$, porque enquanto ela continua crescendo, a outra 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 método de Bailey e a função $exp(x)$ são descritos através dos gráficos abaixo:

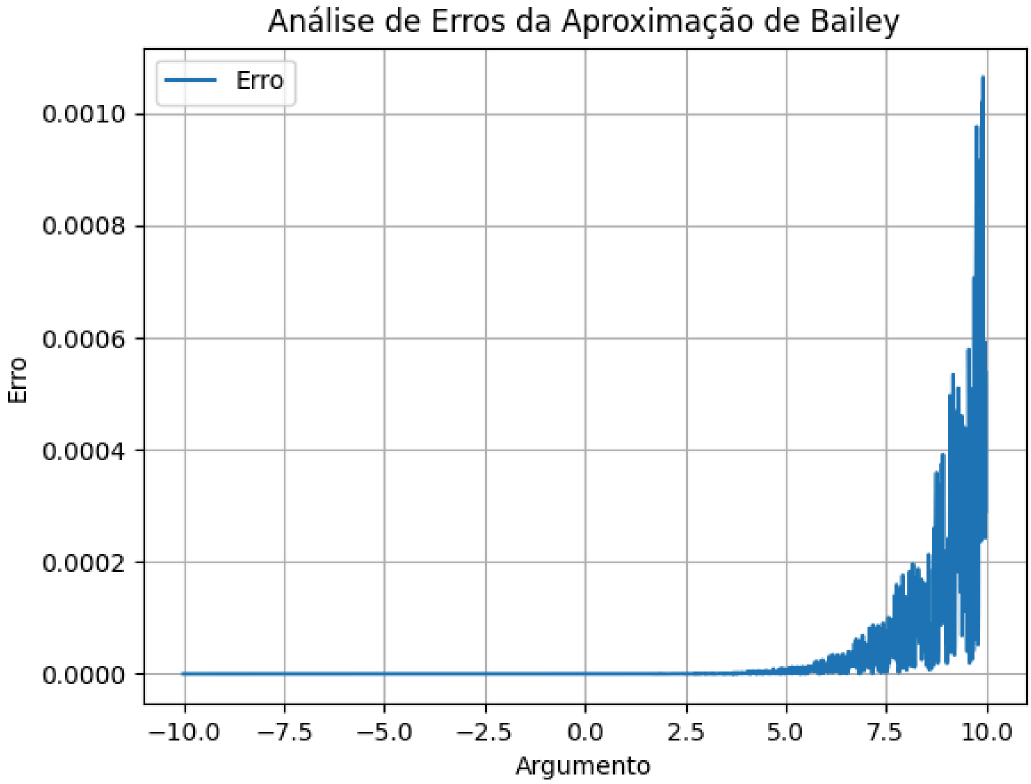

Figura 5: análise do erro para valores de argumento de -10 a 10.

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 de aproximação de Bailey 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 utilizando o método de Bailey, 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:

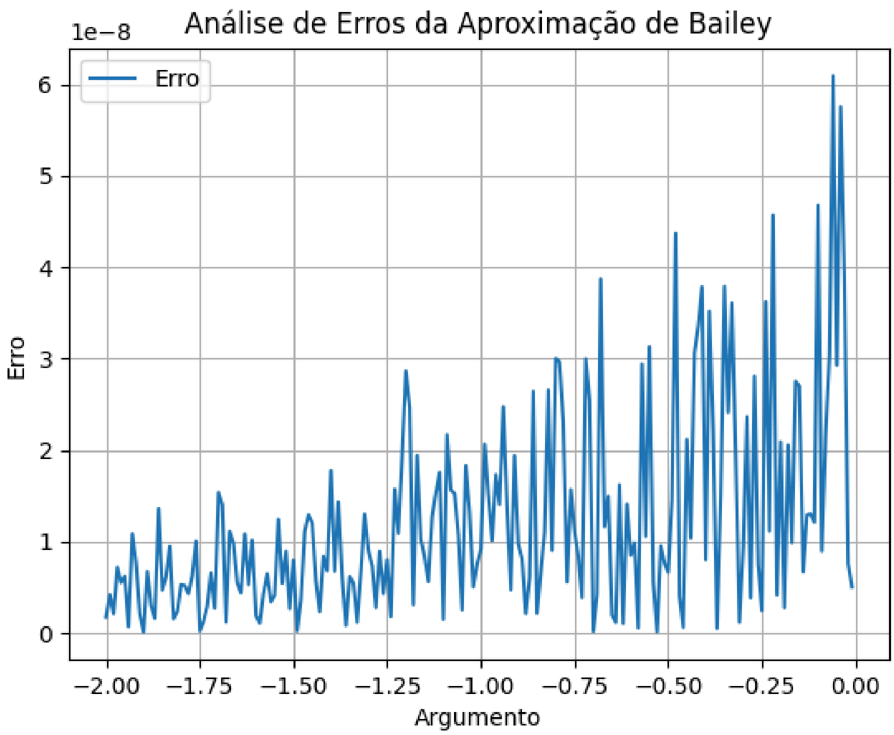

Figura 6: análise do erro para valores de argumento de -2 a 0.

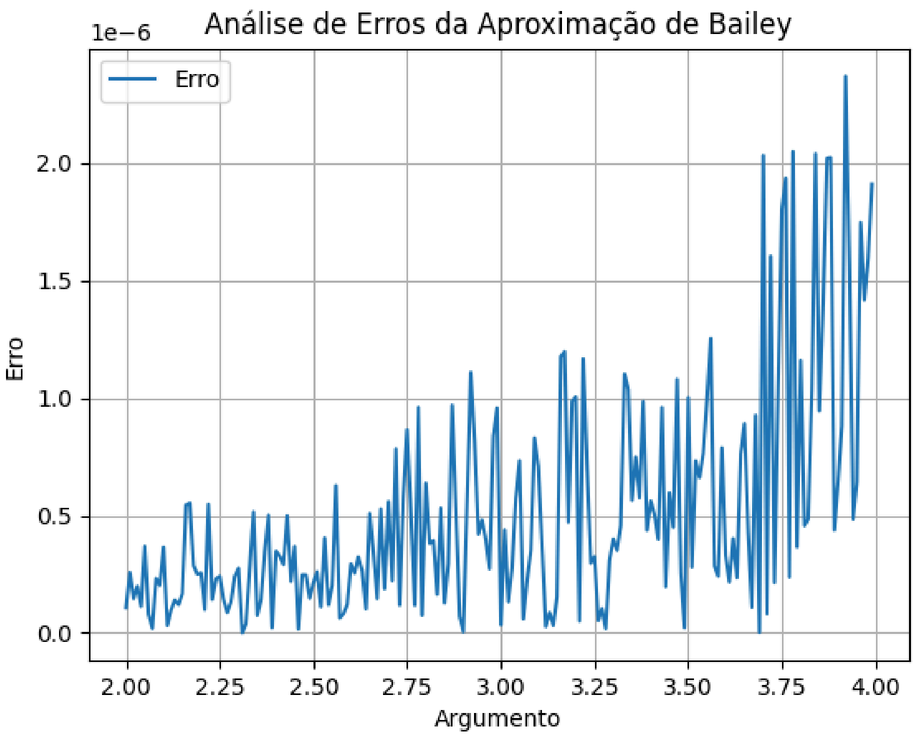

Figura 7: análise do erro para valores de argumento de 2 a 4.

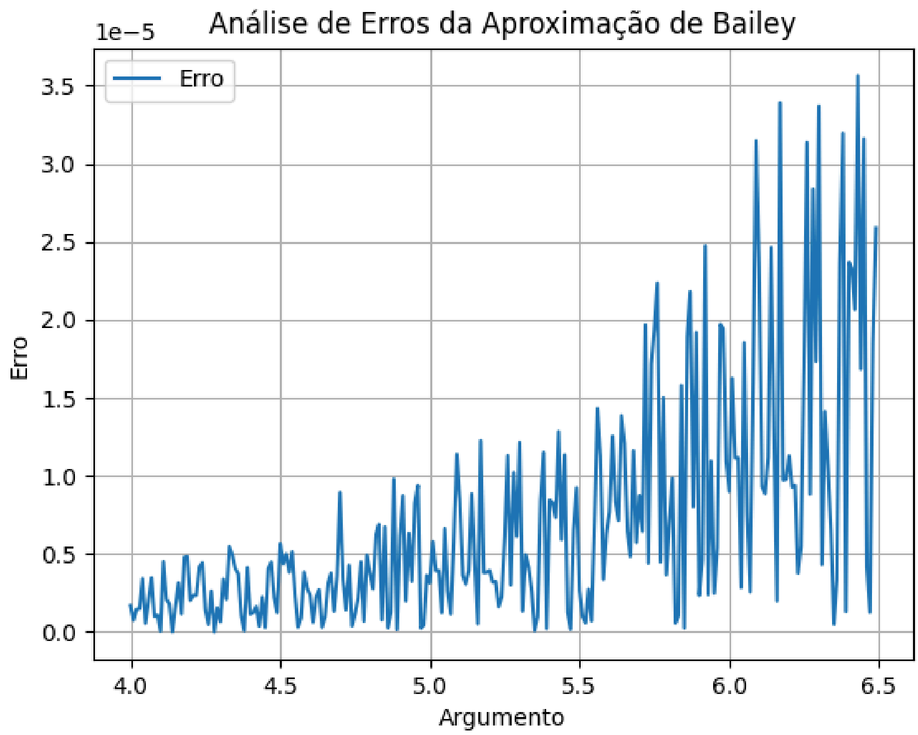

Figura 8: análise do erro para valores de argumento de 4 a 7.

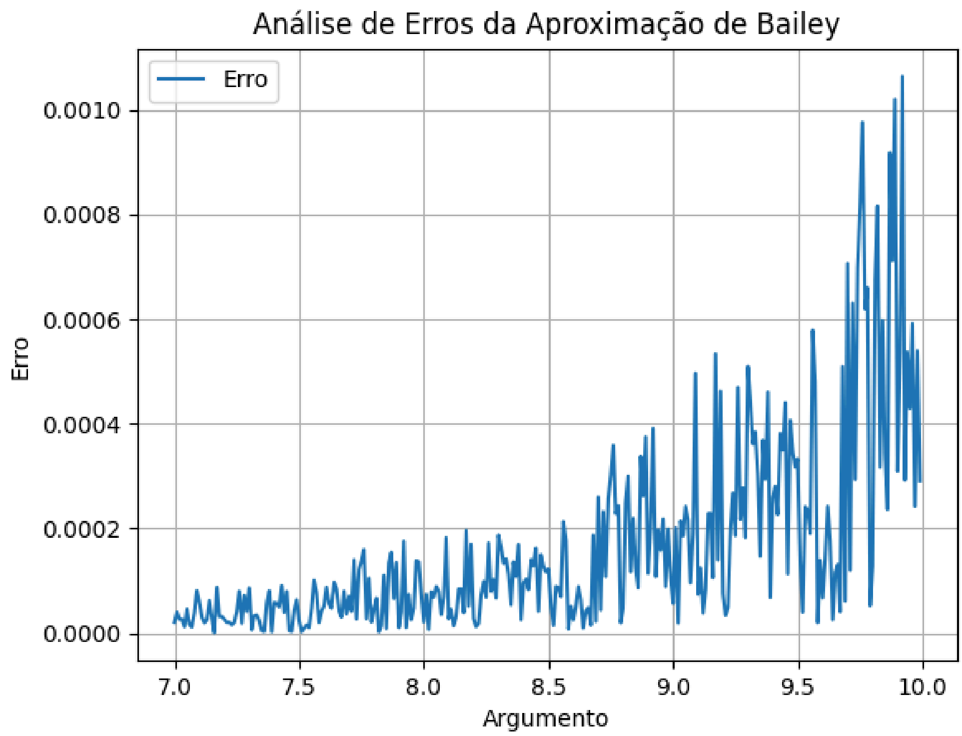

Figura 9: análise do erro para valores de argumento de 7 a 10.

Ao analisar todos os gráficos obtidos, é possível enxergar um padrão no comportamento do erro. O erro se propaga mais rápido conforme o valor do argumento aumenta. Para o intervalo de -2 a 0 (Figura 6), o erro alcançou a oitava casa decimal. Já para o intervalo de 2 a 4 (Figura 7), o erro se propagou para a sexta casa decimal. No intervalo de 4 a 7 (Figura 8), atingiu a quinta casa decimal. E por fim, no intervalo de 7 a 10 (Figura 9), o erro alcançou a quarta e a terceira casa decimal.

7. CONCLUSÃO

Em conclusão, reduzir o valor do argumento da função exponencial e utilizar o método de Bailey é uma estratégia eficaz para alcançar maior precisão em menor tempo, minimizando os erros causados por limitações na representação de números de ponto flutuante.

Os resultados alcançados com o método de Bailey 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 $exp(x)$.

Com a análise do erro vista nas Figuras 6, 7, 8 e 9 conclui-se que o erro se propaga mais rápido conforme o valor do argumento aumenta, este comportamento é esperado, como foi visto teoricamente em sala de aula.

Esses resultados reforçam o estigma aprendido em sala de aula de que “computadores lidam melhor com números pequenos” e indicam que os métodos aproximativos podem ser eficientes em situações onde a precisão máxima não é o foco, mas sim a eficiência computacional.

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.