Estimativa de Seno
Matemática Computacional

Felipe Gabriel Comin Scheffel[1]

[1]Universidade Estadual de Maringá (UEM)

{ra117306}@uem.br

Abstract. *The main objective of this article is to present and compare more than one implementation of the sine estimation problem, analyzing aspects such as error due to estimation, error due to truncation and performance based on the number of operations realized.*

Resumo. *Este trabalho tem como objetivo principal apresentar e comparar mais de uma implementação do problema da estimativa do seno, analisando aspectos como erro devido a estimativa, erro por truncamento e performance por número de operações realizadas.*

1. Introdução

Como a função seno é uma função transcendental (ou seja, não pode ser expressa como uma expressão algébrica finita), não é possível calcular seu valor exato para a maioria dos argumentos usando um número finito de operações aritméticas. A estimativa do seno é o processo de aproximar o valor da função seno em uma determinada entrada usando métodos numéricos. É um problema comum em muitos campos, incluindo matemática, física, engenharia e ciência da computação. Na prática, a escolha do método de aproximação do seno depende dos requisitos específicos da aplicação, como a precisão necessária, a faixa de valores de entrada e os recursos computacionais disponíveis. Vários algoritmos foram desenvolvidos para estimar o valor do seno, variando de simples expansões da série de Taylor a métodos iterativos mais complexos.

2. Metodologia

Para a realização da implementação e os estudos seguintes, foram implementadas várias versões de dois processos de estimativa do seno: o primeiro utiliza uma Série de Maclauring (versão especial da uma Série de Taylor), enquanto o segundo utiliza a aproximação de Padè. O programa tem a capacidade de comparar os resultados das estimativas com o valor ”real”do seno, utilizando uma biblioteca padrão da linguagem, para calcular o erro.

Também foi realizada a implementação das funções utilizando o método de Horner para reduzir o número de operações complexas realizadas (multiplicações e divisões), visando aumentar a performance e reduzir ainda mais o erro. Existe a funcionalidade de comparar as funções implementadas entre si em termos de tempo de execução e minimização de erro.

2.1. Fundamentação Teórica

A fundamentação teórica para a estimativa de seno tem como base o fato de a função seno ser transcendental. Ou seja, apesar de ser periódica e contínua, mas não poder ser

representada exatamente por um número finito de termos em uma série polinomial, como as funções lineares, por exemplo. Portanto, aproximar a função seno envolve encontrar uma série de termos polinomiais que se aproximem da função seno gradativamente ao se aproximar do infinito.

2.1.1. Série de Taylor

A abordagem mais comum para aproximar a função seno é através da expansão em série de Taylor. Essa é uma técnica matemática que permite a expressão de uma função como uma série de potências de suas derivadas avaliadas em um ponto específico. Ou seja, partindo de um ponto central, são calculadas as distâncias para os próximos pontos da série, derivando e normalizando o valor. Segue a fórmula:

$$P(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

onde $f(x)$ é a função sendo aproximada, $f^{(n)}(a)$ é a $n$ésima derivada de $f(x)$ avaliada em $x = a$, a é o centro da série e ! denota o fatorial de n.

Quando o ponto cetral $a = 0$, como no caso da estimativa do seno, a série é denominada um Série de Maclaurin. Partindo do fato de que as primeiras derivadas de $sinx$ seguem:

$$\begin{aligned}
\frac{d}{dx}\sin x &= \cos x \\
\frac{d^2}{dx^2}\sin x &= -\sin x \\
\frac{d^3}{dx^3}\sin x &= -\cos x \\
\frac{d^4}{dx^4}\sin x &= \sin x
\end{aligned}$$

É facilmente perceptível o caráter cíclico das derivações. E, visto o resultado das funcões para x = 0:

$$\begin{aligned}
\sin 0 &= 0 \\
\cos 0 &= 1
\end{aligned}$$

é possível adaptar a série base de Taylor para uma série especializada:

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!}x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

Essa especialização, como toda série de Taylor, é uma ferramenta excelente para aproximar a função seno quando o valor de x é pequeno. No entanto, à medida que o valor de x aumenta, o erro na aproximação aumenta de maneira explosiva após certo ponto.

Nesse trabalho, será utilizada a série de Maclaurin truncada em 6 termos, ou seja, a série de Taylor de ordem 11 (ou 12, visto que tem o mesmo valor).

$$P_{11}(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!}$$

2.1.2. Aproximação de Padè

A aproximação de Padè é uma generalização da série de Taylor, representando uma função a partir da divisão entre dois polinômios[Baker et al. 1996]. Quando a centralização é no ponto $x = 0$, a aproximação pode ser representada pela fórmula:

$$R(x) \approx \frac{\sum_{i=0}^{m} a_i x^i}{1 + \sum_{j=1}^{n} b_j x^j}$$

Sendo representada pela notação $[n/m]$, representando a ordem dos polinômios. Para obter os coeficientes, é comum utilizar sequências de potências de uma função de aproximação (para a centralição em $x = 0$, é possível utilizar uma série de Maclaurin) truncadas em certa ordem, gerando um sistema de equações lineares.

Por exemplo, para a aproximação de Pade $[2/2]$, expandindo os polinômios, temos:

$$\sin(x) \approx \frac{p_0 + p_1 x + p_2 x^2}{1 + q_1 x + q_2 x^2}$$

E, pela fórmula apresentada anteriormente, obtida pela série de Maclaurin:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

Para a ordem $[m, n]$, é necessário realizar o truncamento no termo de grau $m + n + 1$, ou seja, para a ordem $[2, 2]$, será utilizado até o termo de grau 5. Então, igualando as fórmulas, chega-se à equação:

$$p_0 + p_1 x + p_2 x^2 = (1 + q_1 x + q_2 x^2)(x - \frac{x^3}{3!} + \frac{x^5}{5!})$$

E, então, resolvendo o sistema linear para os coeficientes, têm-se que:

$$
\begin{aligned}
p_0 &= 0 \\
p_1 &= 1 \\
p_2 &= 0 \\
q_1 &= 0 \\
q_2 &= \frac{1}{6}
\end{aligned}
$$

E substituindo na fórmula:

$$\sin(x) \approx \frac{x}{1 + \frac{1}{6}x^2}$$

A aproximação de Padè geralmente pode fornecer estimativas mais precisas do que aproximações polinomiais. No entanto, o método requer mais computação e é mais difícil de analisar do que a série de Taylor, e as propriedades de convergência das aproximações de Padè podem ser afetadas pelas propriedades da função que está sendo aproximada.

Nesse trabalho, será utilizada a aproximação de Padè de ordem $[7, 4]$, que tem a forma:

$$R_{7,4}(x) = \frac{p_0 + p_1x + p_2x^2 + p_3x^3 + p_4x^4 + p_5x^5 + p_6x^6 + p_7x^7}{1 + q_1x + q_2x^2 + q_3x^3 + q_4x^4}$$

Igualando a $P_{11}(x)$, resolve-se o sistema linear para chegar aos coeficientes:

$$
\begin{aligned}
p_0 &= 0 \\
p_1 &= 1 \\
p_2 &= 0 \\
p_3 &= -241/1650 \\
p_4 &= 0 \\
p_5 &= 601/118800 \\
p_6 &= 0 \\
p_7 &= -121/2268000 \\
q_1 &= 0 \\
q_2 &= 17/825 \\
q_3 &= 0 \\
q_4 &= 19/118800
\end{aligned}
$$

E assim, substituindo na fórmula:

$$R_{7,4}(x) = \frac{x - \frac{241x^3}{1650} + \frac{601x^5}{118800} - \frac{121x^7}{2268000}}{1 + \frac{17x^2}{825} + \frac{19x^4}{118800}}$$

2.1.3. Método de Horner

Para reduzir o número de operações realizadas no cálculo de ambos os métodos de aproximação, foi utilizada a forma de Horner. Resumidamente, se trata de uma maneira de reorganizar os termos de um polinômio, mantendo os termos de menor em grau em maior evidência para que possa ser realizado o aproveitamento de valores já calculados nos termos de maior grau[Burrus et al. 2003].

Por exemplo, o polinômio

$$p(x) = 2x^3 - 5x^2 + 3x - 1$$

pode ser refatorado para a forma de Horner:

$$p(x) = ((2x - 5)x + 3)x - 1$$

Aplicando esse processo na série de Maclaurin para sen(x) truncada em 6 termos, chega-se em:

$$P_{11}(x) = x\left(1 + x^2\left(-\frac{1}{3!} + x^2\left(\frac{1}{5!} + x^2\left(-\frac{1}{7!} + x^2\left(\frac{1}{9!} - \frac{x^2}{11!}\right)\right)\right)\right)\right)$$

É importante notar que nesse caso específico as expressões ”internas”utilizam como repetição o valor x^2, e no fim é relizada uma multiplicação ”extra”por x. Essa característica será relevante na implementação.

2.2. Implementação

Nessa seção, será comentado sobre o processo de desenvolvimento do script, além de serem apresentados trechos relevantes de código e instruções para sua utilização. Nos trechos de código, serão omitidos comentários, tipagens e validações a fim de apresentar a lógica principal de forma mais limpa. Por fim, será apresentado o código em sua totalidade.

Foi desenvolvida a implementação utilizando a linguagem python[1], visto que se trata apenas de um script imperativo com estrutura pouco robusta, tendo sido utilizadas algumas bibliotecas:

[1]https://www.python.org/

- **numpy**[2] e **matplotlib.pyplot**[3], para a geração dos gráficos
- **timeit**, nativa, para as análises de tempo de execução

O programa completo está disponível no github, em *schelip/sine-estimation*.

2.2.1. Entrada de dados

O programa funciona de modo interativo, mas precisa inicialmente receber uma flag indicando qual o propósito da execução. Estão disponíveis as seguintes opções:

- **-t –time-profile**: imprime os resultados, tempo de execução e erro das funções de estimativa implementadas
- **-v –plot-values**: plota os resultados das funções de estimativa implementadas
- **-e –plot-error**: plota o erro das funções de estimativa implementadas
- **-x –export**: exporta os resultados das funções de estimativa implementadas e erro para um arquivo de texto
- **-h –help**: mostra esta informação de uso

Para a opção `--plot-values`, é possível realizar o plot para apenas um dos métodos de aproximação utilizando um argumento extra `[-m | --maclaurin, -p | --pade]`

Após iniciar o programa, o usuário será questionado para os valores máximo, mínimo e de passo para os valores de x a terem o seno estimado. Para esses valores, é possível utilizar `pi` ou operações matemáticas, e o código interpretará a operação utilizando `eval`.

2.2.2. Implementação da Série de Taylor

Foram implementadas duas versões do algoritmo. A primeira, não otimizada, simplesmente aplica a fórmula para x, truncando para um certo número de termos definido por uma constante.

```
MACLAURIN_TERMS = 6

def sin_taylor_unopt(x):
    result = 0
    for i in range(MACLAURIN_TERMS):
        result += (pow(-1, i) * pow(x, 2 * i + 1) /
           factorial(2 * i + 1))
    return result
```

É facíl ver que essa versão está muito pouco otimizada. Uma certa iteração i irá realizar

- i multiplicações para positivar/negativar o termo;
- 2i multiplicações para calcular o fatorial;

[2] https://numpy.org/
[3] https://matplotlib.org/

- 2i multiplicações para calcular a potência;
- 1 multiplicação entre os fatores do termo;
- 1 divisão entre os fatores do termo;

Totalizando 5i + 2 operações por iteração.

Essa quantidade de multiplicações traz uma queda tanto de performance, quanto de precisão, visto a impossibilidade de representar com precisão um número com ponto flutuante com casas decimais finitas.

Portanto, segue a versão otimizada da aproximação utilizando a série de Taylor:

```
coefs_maclaurin = [1, -1/6, 1/120, -1/5040, 1/362880,
   -1/39916800]

def sin_taylor_opt(x: float) -> float:
    return horner(x, coefs_maclaurin, MACLAURIN_TERMS)
```

Essa versão realiza exatamente 7 operações (uma para coeficiente e uma extra para o membro de grau 1).

2.2.3. Método de Horner

Foi realizada a otimização do número de operações utilizando a forma de Horner. Para tanto, os coeficientes são pré-calculados e armazenados em uma lista, que é então percorrida de forma reversa para cálculo dos membros mais internos da forma.

Foi criada uma função helper para o cálculo de uma solução de um polinômio na forma de Horner. É importante notar que essa implementação utiliza x^2 ao invés de x, visto que essa é a forma simplificada obtida da série de Taylor, contando com um argumento `extra` que dita ser realizada uma multiplicação por x ao fim da execução.

```
def horner(x, coefs, n_terms, extra = True):
    x_2 = x * x
    result = coefs[n_terms - 1]
    for i in range(n_terms - 2, -1, -1):
        if coefs[i] != 0:
            result *= x_2
            result += coefs[i]
    if coefs[0] == 0: result *= x_2
    if extra: result *= x
    return result
```

2.2.4. Aproximação de Padè

Nesse caso, também foram implementadas duas versões do algoritmo, utilizando ou não o método de Horner. A implementação não otimizada simplesmente realiza as somas dos membros de cada polinômio e então a divisão:

```
PADE_ORDER = (7, 4)
coefs_pade_num = [1, 0, -241/1650, 0, 601/118800, 0,
   -121/2268000]
coefs_pade_den = [0, 17/825, 0, 19/118800]

def sin_pade_unopt(x):
    num = sum([coefs_pade_num[i] * pow(x, i + 1) for i in
       range(PADE_ORDER[0])])
    den = 1 + sum([coefs_pade_den[i] * pow(x, i + 1) for i
       in range(PADE_ORDER[1])])
    return num / den
```

O termo i de cada polinômio conta com uma exponenciação de grau i e uma multiplicação pelo coeficiente. Assim, são realizadas $\sum_{i=1}^{7} i$ multiplicações para o numerador e $\sum_{i=1}^{4} i$, totalizado $28 + 10 = 38$ multiplicações e uma divisão, resultando em 39 operações.

A versão otimizada utiliza as mesmas listas de coeficientes, mas na forma de Horner, fazendo uso da função útil já apresentada. Note que o denominador não realiza a multiplicação extra pelo elemento com grau 1, e é necessário realizar uma soma com o elemento de grau 0 (ou seja, 1).

```
def sin_pade_opt(x):
    num = horner(x, coefs_pade_num, PADE_ORDER[0])
    den = 1 + horner(x, coefs_pade_den, PADE_ORDER[1],
       False)
    return num / den
```

No total, são realizadas 8 operações: uma para cada coeficiente diferente de 0, uma extra no numerador e uma divisão.

3. Discussão

Nesse tópico, serão apresentados os resultados obtidos pela implementação desenvolvida para diferentes conjuntos de dados, além de ser realizada uma análise sobre os erros no resultado.

Todas as análises foram realizadas considerando grau 11 para a série de Taylor (ou seja, 6 termos) e grau $[7/4]$ para a aproximação de Padè.

3.1. Performance

Foi desenvolvida uma função que utiliza a `timeit` para comparar os tempos de execução. Seguem os resultados obtidos para $x = pi/8$:

Função	Resultado	Erro
math.sin (valor ”real”)	0.3826834323650898	0
taylor não-otimizado	0.3826834323650889	8.881784197001252e-16
taylor otimizado	0.3826834323650889	8.881784197001252e-16
padè não-otimizado	0.38268343236508934	4.440892098500626e-16
padè otimizado	0.38268343236508934	4.440892098500626e-16

Função	Tempo de execução (10000x)
taylor não-otimizado	0.04874589999963064
taylor otimizado	0.014976499995100312
padè não-otimizado	0.05690310000500176
padè otimizado	0.023631500007468276

Percebe-se facilmente a melhora no tempo de execução sem alteração no erro. Mesmo sendo realizadas um número muito baixo de iterações por função, para 10000 execuções, o tempo é cortado no mínimo pela metade na versão otimizada.

3.2. Intervalo pequeno

Inicialmente, foi realizada a análise utilizando um conjunto pequeno e próximo de zero, iniciando em $-\pi/4$ até $\pi/4$ com passo de 0.1.

Foi gerado o gráfico com os valores obtidos e também do erro cálculado. Além da geração de gráficos, o programa apresenta uma opção para exportar os resultados para um arquivo de texto, que apresentaram os resultados demonstrados nas tabelas abaixo.

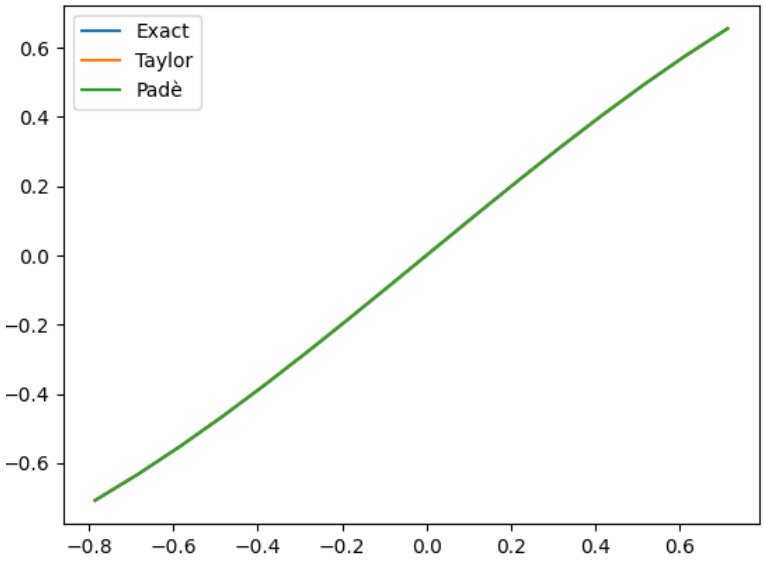
Percebe-se que o erro geral é de ordem baixa, tendo ordem máxima de 10^{-11} tanto para Taylor quanto Padè, e portanto ambos os métodos foram considerados adequados para esse intervalo.

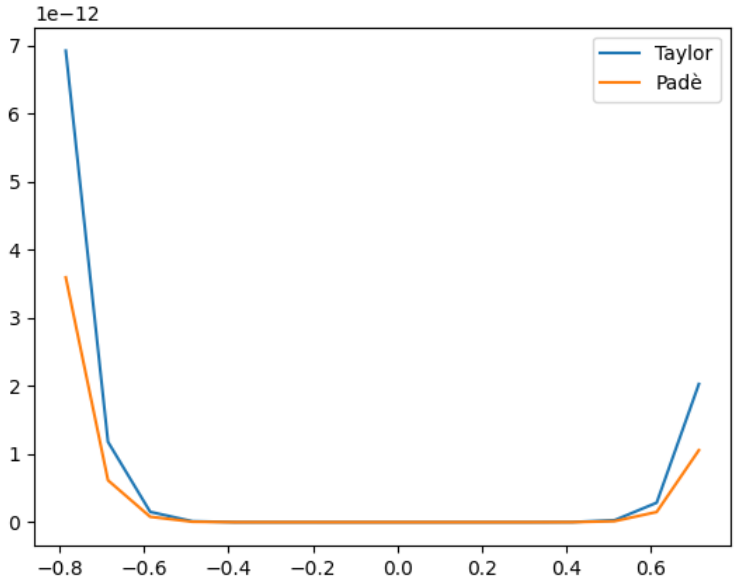
Resultados para série de Maclaurin:

Valor x	sen(x)	Taylor (ordem 11)	Erro de Taylor
-0.7853981633974483	-0.7071067811865476	-0.7071067811796194	0.0000000000069281
-0.6853981633974483	-0.6329813066769582	-0.6329813066757778	0.0000000000011804
-0.5853981633974483	-0.5525312921868542	-0.5525312921867022	0.0000000000001520
-0.4853981633974483	-0.4665605676677813	-0.4665605676677680	0.0000000000000133
-0.3853981633974484	-0.3759281241809911	-0.3759281241809904	0.0000000000000007
-0.2853981633974484	-0.2815395311427008	-0.2815395311427007	0.0000000000000001
-0.1853981633974484	-0.1843378881738284	-0.1843378881738284	0.0000000000000000
-0.0853981633974484	-0.0852944019603276	-0.0852944019603276	0.0000000000000000
0.0146018366025515	0.0146013177229801	0.0146013177229801	0.0000000000000000
0.1146018366025515	0.1143511458661537	0.1143511458661537	0.0000000000000000
0.2146018366025515	0.2129584151592960	0.2129584151592960	0.0000000000000000
0.3146018366025514	0.3094378743628594	0.3094378743628592	0.0000000000000001
0.4146018366025515	0.4028255326123511	0.4028255326123494	0.0000000000000017
0.5146018366025515	0.4921882912963978	0.4921882912963694	0.0000000000000284
0.6146018366025514	0.5766332672696289	0.5766332672693427	0.0000000000002862
0.7146018366025513	0.6553167142459179	0.6553167142438878	0.0000000000020300

Resultados para aproximação de Padè:

Valor x	sen(x)	Padè $[7/4]$	Erro de Padè
-0.7853981633974483	-0.7071067811865476	-0.7071067811829503	0.0000000000035972
-0.6853981633974483	-0.6329813066769582	-0.6329813066763423	0.0000000000006158
-0.5853981633974483	-0.5525312921868542	-0.5525312921867745	0.0000000000000796
-0.4853981633974483	-0.4665605676677813	-0.4665605676677743	0.0000000000000070
-0.3853981633974484	-0.3759281241809911	-0.3759281241809908	0.0000000000000003
-0.2853981633974484	-0.2815395311427008	-0.2815395311427007	0.0000000000000001
-0.1853981633974484	-0.1843378881738284	-0.1843378881738284	0.0000000000000000
-0.0853981633974484	-0.0852944019603276	-0.0852944019603276	0.0000000000000000
0.0146018366025515	0.0146013177229801	0.0146013177229801	0.0000000000000000
0.1146018366025515	0.1143511458661537	0.1143511458661538	0.0000000000000000
0.2146018366025515	0.2129584151592960	0.2129584151592960	0.0000000000000000
0.3146018366025514	0.3094378743628594	0.3094378743628592	0.0000000000000001
0.4146018366025515	0.4028255326123511	0.4028255326123502	0.0000000000000009
0.5146018366025515	0.4921882912963978	0.4921882912963829	0.0000000000000149
0.6146018366025514	0.5766332672696289	0.5766332672694792	0.0000000000001498
0.7146018366025513	0.6553167142459179	0.6553167142448602	0.0000000000010577

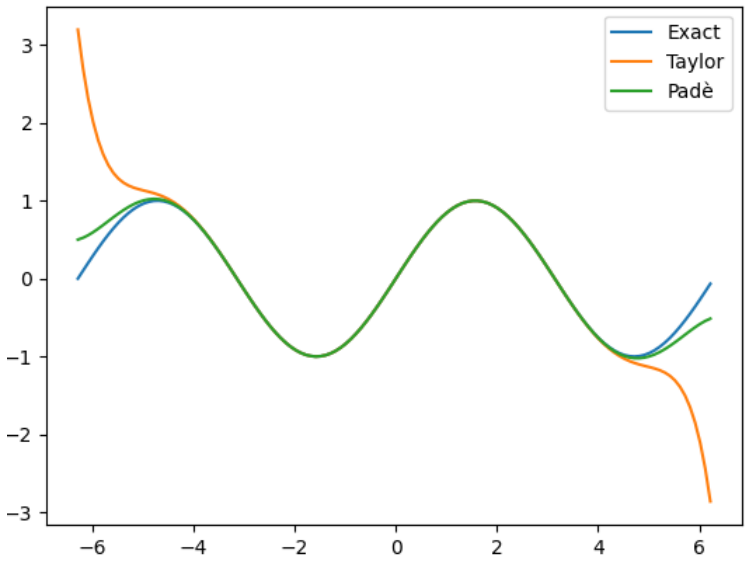

Exact
Taylor
Padè
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6

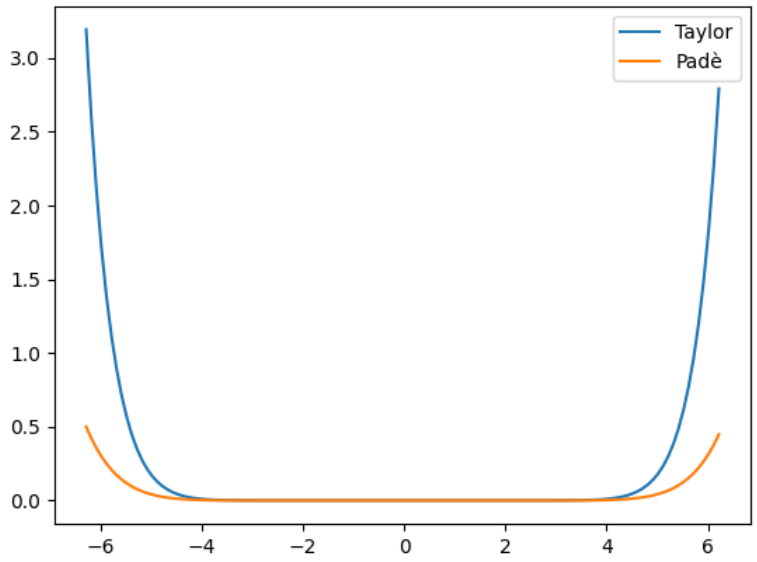

1e−12
Taylor
Padè
7
6
5
4
3
2
1
0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6

3.3. Intervalo grande

Em seguida, foi realizado o cálculo para um intervalo com mesmo passo, mas uma aplitude maior, indo de -2π a 2π, ou seja, para todo o ciclo inicial da função seno em ambas as direções. Apenas os gráficos serão apresentados, visto a quantidade grande de linhas que são exportadas para o arquivo de texto.

3.3.1. Resultados





Já é possível perceber a vantagem da aproximação de Padè. Ao ultrapassar certo valor, a série de Taylor desaba, enquanto a aproximação de Padè, apesar de também apresentar bastante erro, já passando da ordem de 10^0, continua convergindo no formato da função.

4. Conclusão

No decorrer das atividade, foi possível desenvolver a implementação de um script que realize o cálculo da estimativa do seno com duas técnicas diferentes e realizar uma comparação tanto entre os resultados quanto entre os próprios processos. Com isso, também foi possível realizar uma análise sobre o erro e seus impactos em métodos de estimação e no contexto geral da computação.

Referências

[Baker et al. 1996] Baker, G. A., Baker Jr, G. A., Graves-Morris, P., and Baker, S. S. (1996). *Pade Approximants: Encyclopedia of Mathematics and It's Applications, Vol. 59 George A. Baker, Jr., Peter Graves-Morris*, volume 59. Cambridge University Press.

[Burrus et al. 2003] Burrus, C. S., Fox, J. W., Sitton, G. A., and Treitel, S. (2003). Horner's method for evaluating and deflating polynomials. *DSP Software Notes, Rice University, Nov*, 26.