****

**Universidade Estadual de Santa Cruz - UESC**

**Segundo Relatório de Implementações de Métodos da Disciplina Análise Numérica**

Relatório de implementações realizadas por Igor Lima Rocha

Disciplina Análise Numérica.

Curso Ciência da Computação

Semestre 2023.2

Professor Gesil Sampaio Amarante II

**Ilhéus - BA  
2023**

# Índice

[**Índice 2**](#_ob0knhvlg74y)

[**Lista de Figuras 5**](#_nxee0sysp8my)

[Método da Regressão linear 5](#_mecpvsk1jl82)

[Método da Interpolação Lagrange 5](#_e4c6bryq78i5)

[Método de Diferenças Divididas de Newton 5](#_933dddu0j4p9)

[Método da Derivadas de 1a ordem 5](#_m0z6ulq50hzu)

[Método da Derivadas de 2a ordem 5](#_hw6wm85jqy2p)

[Método da Integração Trapézio Simples 5](#_q50czbpvw155)

[Método de Integração Trapézio Múltiplo 5](#_2ayivbuyylzi)

[Método de Simpson 1/3 5](#_e9t26mxorsrs)

[Método de Simpson 3/8 5](#_k1181kmk7yov)

[**Linguagem escolhida e justificativas 6**](#_p00saff8igu5)

[**Instruções gerais 7**](#_32zlbquu6755)

[**Método 1 (Regressão Linear) 8**](#_dma3x5k4n4l0)

[Estratégia de Implementação 8](#_utleu9crzr9)

[Estrutura dos Arquivos de Entrada/Saída 8](#_i1lb7t7zu33m)

[Problemas teste 9](#_rq194vanr7hc)

[Dificuldades enfrentadas 10](#_yh17tvwqcyhb)

[**Método 2 (Interpolação Lagrange) 11**](#_xz9o95xavbqh)

[Estratégia de Implementação 11](#_kdrw6modmxwc)

[Estrutura dos Arquivos de Entrada/Saída 11](#_85y7gf80e83f)

[Problemas teste 11](#_cmyqzbksabub)

[Dificuldades enfrentadas 12](#_re1fdsi7kdn2)

[**Método 3 (Diferenças Divididas de Newton) 13**](#_e8664xfs4heq)

[Estratégia de Implementação 13](#_uur6udqptj1)

[Estrutura dos Arquivos de Entrada/Saída 13](#_61sghl6h7a33)

[Problemas teste 13](#_buxeg7oqur1e)

[Dificuldades enfrentadas 14](#_t6d1c2j9diku)

[**Método 4 (Derivadas de 1a ordem) 15**](#_xbk15250eszv)

[Estratégia de Implementação 15](#_1n3w14s63u0k)

[Estrutura dos Arquivos de Entrada/Saída 15](#_ov8n5wgb0zn5)

[Problemas teste 15](#_pjsdza6xwujt)

[Dificuldades enfrentadas 16](#_2ov1kewfnpuo)

[**Método 5 (Derivadas de 2a ordem) 17**](#_xu8ai4rntedl)

[Estratégia de Implementação 17](#_o4mjsawnz8hi)

[Estrutura dos Arquivos de Entrada/Saída 17](#_43eyy0kw6rmd)

[A saída é o valor aproximado da derivada no ponto xi, que é retornado pela função solve\_by\_derivada\_segunda. 17](#_ymhzcxma19fg)

[Problemas teste 17](#_2tnri6m8wrv)

[Dificuldades enfrentadas 18](#_eh1pmq4ycxcw)

[**Método 6 (Integração Trapézio Simples) 19**](#_3cc6sxb3lqdt)

[Estratégia de Implementação 19](#_e8yazgvwsqh5)

[Estrutura dos Arquivos de Entrada/Saída 19](#_7nc96t58gncd)

[A saída é o valor aproximado da derivada no ponto xi, que é retornado pela função solve\_by\_trapezio\_simples. 19](#_h9orbw2o4w43)

[Problemas teste 19](#_6jmffvqhpsx7)

[Dificuldades enfrentadas 20](#_v0qxs2pvhr5b)

[**Método 7 (Integração Trapézio Múltiplo) 21**](#_ct9ljc1t7k4d)

[Estratégia de Implementação 21](#_9tbmed41rzqx)

[Estrutura dos Arquivos de Entrada/Saída 21](#_7vvy353cmwno)

[A saída é o valor aproximado da derivada no ponto xi, que é retornado pela função solve\_by\_trapezio\_multiplo. 21](#_v9fpm8mkmdpp)

[Problemas teste 21](#_eqn1nzvhz3e2)

[Dificuldades enfrentadas 22](#_x172xr1gmlh4)

[**Método 8 (Simpson 1/3) 23**](#_mqiqh1nofz2x)

[Estratégia de Implementação 23](#_7s89i7rt0ofg)

[Estrutura dos Arquivos de Entrada/Saída 23](#_uqjqq9enjao)

[A saída é o valor aproximado da integral no ponto xi, no intervalo especificado, que é retornado pela função solve\_by\_simpson\_1\_3. 23](#_scphf1wt3crm)

[Problemas teste 23](#_w30quw1dkigi)

[Dificuldades enfrentadas 24](#_vav7kxsqyl07)

[**Método 9 (Simpson 3/8) 25**](#_mre99ts0dqxr)

[Estratégia de Implementação 25](#_oacjjbg8eupf)

[Problemas teste 25](#_dopfrhuedsci)

[A saída é o valor aproximado da integral no ponto xi, no intervalo especificado, que é retornado pela função solve\_by\_simpson\_3\_8. 25](#_l1b2i4js40fh)

[Problemas teste 25](#_b4ssh3hxybsr)

[Dificuldades enfrentadas 26](#_d6rfqbvl6jmc)

[**Método 10 (Extrapolação de Richards) 27**](#_xml4ugfakd4q)

[Estratégia de Implementação 27](#_9g6akcejv0ij)

[Esta técnica é usada para extrapolar valores de uma função conhecida em pontos além do conjunto de dados original. A implementação envolve a resolução de um sistema linear para obter os coeficientes da função extrapoladora. A função aceita três parâmetros: um conjunto de pontos conhecidos x e y e um conjunto de pontos X para os quais desejamos extrapolar os valores Y. 27](#_d4jhkltrv320)

[Estrutura dos Arquivos de Entrada/Saída 27](#_d7h2l3cpi645)

[Problemas teste 27](#_xteifnxzhyyt)

[Dificuldades enfrentadas 27](#_yxmajcitdy8h)

[**Método 11 (Quadratura de Gauss) 28**](#_erxmli55a341)

[Estratégia de Implementação 28](#_qsed7ujny4v3)

[Este método é utilizado para calcular a integral de uma função em um intervalo específico. A estratégia adotada envolveu o uso de coeficientes e pesos de Gauss pré-definidos para o cálculo da integral, juntamente com uma transformação dos limites do intervalo de integração para o intervalo padrão [−1,1]. 28](#_5leoj4mktued)

[Estrutura dos Arquivos de Entrada/Saída 28](#_2vaqgqxafk8f)

[A saída é o valor aproximado da integral no intervalo especificado, que é retornado pela função solve\_by\_gauss. 28](#_bqs5ca4f8yuo)

[Problemas teste 28](#_5m4v238mxlb)

[Dificuldades enfrentadas 28](#_rzela38ab1qr)

[**Conclusão 29**](#_vat7uub5s5ow)

# Lista de Figuras

## Método da Regressão linear

1. Exemplo de arquivo de entrada do método bissecção
2. Retornos gerais das iterações do algoritmo

## Método da Interpolação Lagrange

1. Retornos gerais das iterações do algoritmo

## Método de Diferenças Divididas de Newton

1. Retornos gerais das iterações do algoritmo

## Método da Derivadas de 1a ordem

1. Retornos gerais das iterações do algoritmo

## Método da Derivadas de 2a ordem

1. Retornos gerais das iterações do algoritmo

## Método da Integração Trapézio Simples

1. Retornos gerais das iterações do algoritmo

## Método de Integração Trapézio Múltiplo

1. Retornos gerais das iterações do algoritmo

## Método de Simpson 1/3

1. Retornos gerais das iterações do algoritmo

## Método de Simpson 3/8

1. Retornos gerais das iterações do algoritmo

# Linguagem escolhida e justificativas

Nesse contexto, optei por utilizar a linguagem de programação Python, juntamente com as bibliotecas pandas e sympy, devido a várias razões que destacam suas vantagens significativas.

A escolha de Python como linguagem principal baseia-se na sua crescente popularidade na comunidade de desenvolvimento e na sua ampla adoção na área científica e de análise numérica. Python oferece uma sintaxe clara e legível, o que torna a implementação dos algoritmos mais intuitiva e facilita a colaboração entre membros da equipe. Além disso, sua vasta coleção de bibliotecas especializadas simplifica a execução de tarefas complexas, economizando tempo e esforço de desenvolvimento.

A inclusão da biblioteca pandas no projeto é justificada pela sua capacidade de manipular e analisar dados tabulares, como os presentes em arquivos CSV. Através do uso do pandas, conseguimos importar facilmente os dados do arquivo e manipulá-los em estruturas de dados flexíveis, tornando a exploração dos dados e a extração de informações relevantes uma tarefa mais eficiente.

A biblioteca sympy desempenha um papel fundamental na nossa abordagem, já que nos permite realizar cálculos simbólicos e manipulação algébrica, essenciais para a solução de equações e determinação de zeros de funções. Através do sympy, podemos definir variáveis simbólicas, criar equações e resolver algebricamente, garantindo uma abordagem precisa e confiável na determinação dos zeros de funções em intervalos específicos. Além disso, o sympy oferece integração com outras bibliotecas numéricas, permitindo-nos combinar os benefícios da manipulação simbólica com os métodos numéricos disponíveis.

Em resumo, a escolha de Python como linguagem principal, combinada com as bibliotecas pandas e sympy, oferece uma abordagem abrangente e eficiente para a implementação dos algoritmos de cálculo numérico. Essas ferramentas nos permitem aproveitar as vantagens da simplicidade de desenvolvimento, manipulação de dados tabulares e cálculos simbólicos, garantindo a precisão e eficiência na determinação dos zeros de funções, como requerido no escopo deste projeto.

# Instruções gerais

Separei as partes principais do programa em componentes diferentes, para cada parte ficar com sua responsabilidade.

O programa dentro de “**main.py**” engloba todos os métodos. Para executar, você deve rodar o comando  **python main.py**  e selecionar o método desejado, assim como algumas configurações gerais.

Criei uma classe geral, que recebe a string da equação, e a transforma em uma equação do **SymPy** (biblioteca escolhida para tratar matematicamente as funções). **Essa classe vai ser utilizada em quase todos os métodos.**

Os arquivos de entrada devem estar na pasta **“entradas”**, da forma que é enunciado no tópico do método.

Já os arquivos de saída serão gerados em pastas separadas, dentro da pasta “**saidas**”.

# 

# Método 1 (Regressão Linear)

## Estratégia de Implementação

Para tratar exclusivamente do método de bissecção, criei uma função que recebe dois parâmetros: x e y, que são listas contendo os valores.

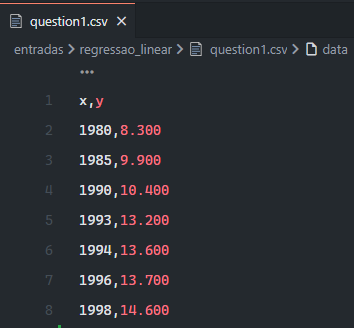
Na função, o cálculo dos parâmetros A e B da regressão linear y = ax + b é feito usando as fórmulas:

O método retorna os valores de a e b, bem como um histórico contendo os cálculos intermediários.

## Estrutura dos Arquivos de Entrada/Saída

Um único arquivo de entrada é necessário, deve chamá-lo de “**linear\_regression.csv**” e estar dentro da pasta **“entradas”**, o arquivo deve ser separado por vírgulas, e ter os seguintes cabeçalhos:

* **x**: Valores da variável independente.
* **y**: Valores da variável dependente.



*Imagem 1.1 - Exemplo de arquivo de entrada do método regressão linear*

Já a saída está no terminal, e vai ser mostrado os seguintes dados:

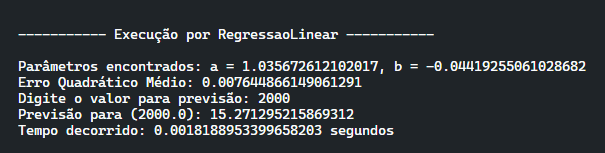
* **a**: Coeficiente angular da reta.
* **b**: Coeficiente linear da reta.
* **EQM**: Erro quadrático médio.

## Problemas teste

Foram realizados testes utilizando os princípios e técnicas descritos no livro **"Cálculo Numérico" de Neide Franco**. A implementação foi avaliada em relação a três problemas apresentados no livro. Os resultados obtidos foram consistentes com métodos amplamente reconhecidos, evidenciando a eficácia da implementação.

As questões escolhidas foram:

* **8.1** da página 268
* **8.5** da página 269
* **8.11** da página 273

  
*Imagem 1.2 - Retornos gerais das iterações do algoritmo*

## Dificuldades enfrentadas

A principal dificuldade foi no cálculo do Erro Quadrático Médio (EQM) e sua interpretação para validar a qualidade do ajuste da reta aos pontos dados.

# Método 2 (Interpolação Lagrange)

## Estratégia de Implementação

A estratégia principal foi criar polinômios Li para cada ponto dado e combinar esses polinômios para formar o polinômio interpolador final.

## Estrutura dos Arquivos de Entrada/Saída

Para cada sistema de equações, deve ser criado um arquivo de entrada, deve chamá-lo de “question{index}.csv”, e deve colocá-lo na pasta **“entradas/interpolacao\_lagrange/”**. O arquivo deve ser separado por vírgulas, e ter os seguintes cabeçalhos:

* **x**: Valores da variável independente.
* **y**: Valores da variável dependente.

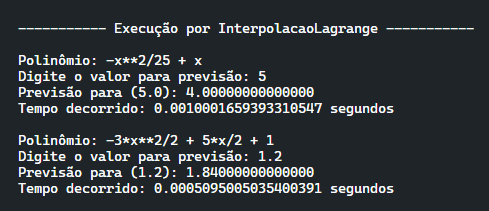
A saída é o polinômio interpolador, que é retornado pela função **solve\_by\_lagrange\_interpolation**.

## Problemas teste

Foram realizados testes utilizando os princípios e técnicas descritos no livro **"Cálculo Numérico" de Neide Franco**. A implementação foi avaliada em relação a três problemas apresentados no livro. Os resultados obtidos foram consistentes com métodos amplamente reconhecidos, evidenciando a eficácia da implementação.

As questões escolhidas foram:

* **10.2** da página 316
* **10.6** da página 318
* **10.9** da página 318

  
*Imagem 2.1 - Retornos gerais das iterações do algoritmo*

## Dificuldades enfrentadas

Não houveram dificuldades.

# Método 3 (Diferenças Divididas de Newton)

## Estratégia de Implementação

Para implementar o método da diferença dividida de Newton, foi desenvolvida uma função chamada solve\_by\_diferenca\_newton. Esta função aceita três parâmetros: uma lista de valores de X, uma lista correspondente de valores de f(X), e um valor x para o qual desejamos estimar f(x) utilizando o polinômio interpolador. A função calcula primeiramente as diferenças divididas com base nos pontos dados e, em seguida, constrói o polinômio interpolador utilizando essas diferenças. Finalmente, utiliza o polinômio para estimar o valor de f(x) para o x dado.

## Estrutura dos Arquivos de Entrada/Saída

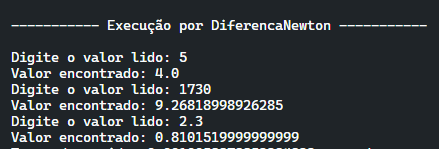
Para cada sistema de equações, deve ser criado um arquivo de entrada, deve chamá-lo de “question{index}.csv”, e deve colocá-lo na pasta **“entradas/diferenca\_newton/”**. O arquivo deve ser separado por vírgulas, e ter os seguintes cabeçalhos:

* **x**: Valores da variável independente.
* **y**: Valores da variável dependente.

A saída é o polinômio interpolador, que é retornado pela função **solve\_by\_newton**.

## Problemas teste

Foram realizados testes utilizando os princípios e técnicas descritos no livro **"Cálculo Numérico" de Neide Franco**. A implementação foi avaliada em relação a três problemas apresentados no livro. Os resultados obtidos foram consistentes com métodos amplamente reconhecidos, evidenciando a eficácia da implementação.

  
*Imagem 3.1 - Retornos gerais das iterações do algoritmo*

## Dificuldades enfrentadas

Não houveram dificuldades.

# 

# Método 4 (Derivadas de 1a ordem)

## Estratégia de Implementação

A função utiliza o método de diferenças centrais, uma aproximação numérica para a derivada.

A equação é processada usando a classe Equation que converte a string em uma função que pode ser avaliada.

## Estrutura dos Arquivos de Entrada/Saída

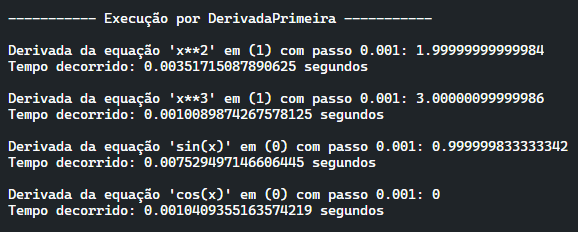
Um único arquivo de entrada é necessário, deve chamá-lo de “**derivada\_primeira.csv**” e estar dentro da pasta **“entradas”**, o arquivo deve ser separado por vírgulas, e ter os seguintes cabeçalhos:

* **equation**: A string que contém a função a ser avaliada. Deve estar nos padrões do SymPy, então:
  + A única variável deve ser “x”
  + deve ser escrito como **x\*\*3**
  + **cosseno de X** deve ser escrito como **cos(x)**
  + **seno de X** deve ser escrito como **sin(x)**
  + números reais devem ser escritos com ponto: cinco vírgula cinco = 5.5
* **x:** O ponto no qual a derivada será calculada.
* **step:** O passo incremental utilizado na aproximação da derivada.

A saída é o valor aproximado da derivada no ponto xi, que é retornado pela função solve\_by\_derivada\_primeira.

## Problemas teste

Foram realizados testes utilizando os princípios e técnicas descritos no livro **"Cálculo Numérico" de Neide Franco**. A implementação foi avaliada em relação a três problemas apresentados no livro. Os resultados obtidos foram consistentes com métodos amplamente reconhecidos, evidenciando a eficácia da implementação.

  
*Imagem 4.1 - Retornos gerais das iterações do algoritmo*

## Dificuldades enfrentadas

Não houveram dificuldades.

# Método 5 (Derivadas de 2a ordem)

## Estratégia de Implementação

A função utiliza a fórmula de diferenças centrais para derivadas de segunda ordem, que é uma aproximação numérica.

A equação é processada usando a classe Equation que converte a string em uma função que pode ser avaliada.

## Estrutura dos Arquivos de Entrada/Saída

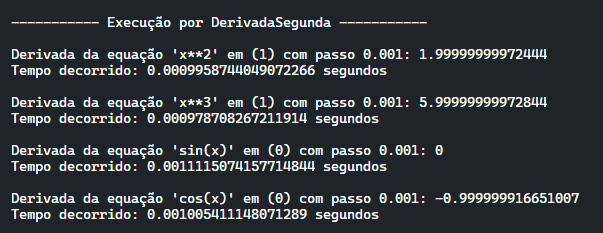
Um único arquivo de entrada é necessário, deve chamá-lo de “**derivada\_segunda.csv**” e estar dentro da pasta **“entradas”**, o arquivo deve ser separado por vírgulas, e ter os seguintes cabeçalhos:

* **equation**: A string que contém a função a ser avaliada. Deve estar nos padrões do SymPy, então:
  + A única variável deve ser “x”
  + deve ser escrito como **x\*\*3**
  + **cosseno de X** deve ser escrito como **cos(x)**
  + **seno de X** deve ser escrito como **sin(x)**
  + números reais devem ser escritos com ponto: cinco vírgula cinco = 5.5
* **x:** O ponto no qual a derivada será calculada.
* **step:** O passo incremental utilizado na aproximação da derivada.

## A saída é o valor aproximado da derivada no ponto xi, que é retornado pela função solve\_by\_derivada\_segunda.

## Problemas teste

Foram realizados testes utilizando os princípios e técnicas descritos no livro **"Cálculo Numérico" de Neide Franco**. A implementação foi avaliada em relação a três problemas apresentados no livro. Os resultados obtidos foram consistentes com métodos amplamente reconhecidos, evidenciando a eficácia da implementação.

  
*Imagem 5.1 - Retornos gerais das iterações do algoritmo*

## Dificuldades enfrentadas

Não houveram dificuldades.

# 

# Método 6 (Integração Trapézio Simples)

## Estratégia de Implementação

A função calcula a área sob a curva da função entre A e B utilizando a fórmula do Trapézio Simples.

A equação é processada usando a classe Equation que converte a string em uma função que pode ser avaliada.

## Estrutura dos Arquivos de Entrada/Saída

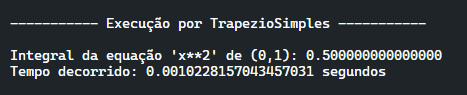
Um único arquivo de entrada é necessário, deve chamá-lo de “**trapezio\_simples.csv**” e estar dentro da pasta **“entradas”**, o arquivo deve ser separado por vírgulas, e ter os seguintes cabeçalhos:

* **equation**: A string que contém a função a ser avaliada. Deve estar nos padrões do SymPy, então:
  + A única variável deve ser “x”
  + deve ser escrito como **x\*\*3**
  + **cosseno de X** deve ser escrito como **cos(x)**
  + **seno de X** deve ser escrito como **sin(x)**
  + números reais devem ser escritos com ponto: cinco vírgula cinco = 5.5
* **a:** O limite inferior de integração.
* **b:** O limite superior de integração.
* **h:** O intervalo de discretizações.

## A saída é o valor aproximado da derivada no ponto xi, que é retornado pela função solve\_by\_trapezio\_simples.

## Problemas teste

Foram realizados testes utilizando os princípios e técnicas descritos no livro **"Cálculo Numérico" de Neide Franco**. A implementação foi avaliada em relação a três problemas apresentados no livro. Os resultados obtidos foram consistentes com métodos amplamente reconhecidos, evidenciando a eficácia da implementação.

  
*Imagem 6.1 - Retornos gerais das iterações do algoritmo*

## Dificuldades enfrentadas

Não houveram dificuldades.

# 

# Método 7 (Integração Trapézio Múltiplo)

## Estratégia de Implementação

A função calcula a área sob a curva da função utilizando a fórmula do Trapézio Múltiplo, que divide o intervalo de integração em N subintervalos iguais e aplica a regra do Trapézio em cada subintervalo.

## Estrutura dos Arquivos de Entrada/Saída

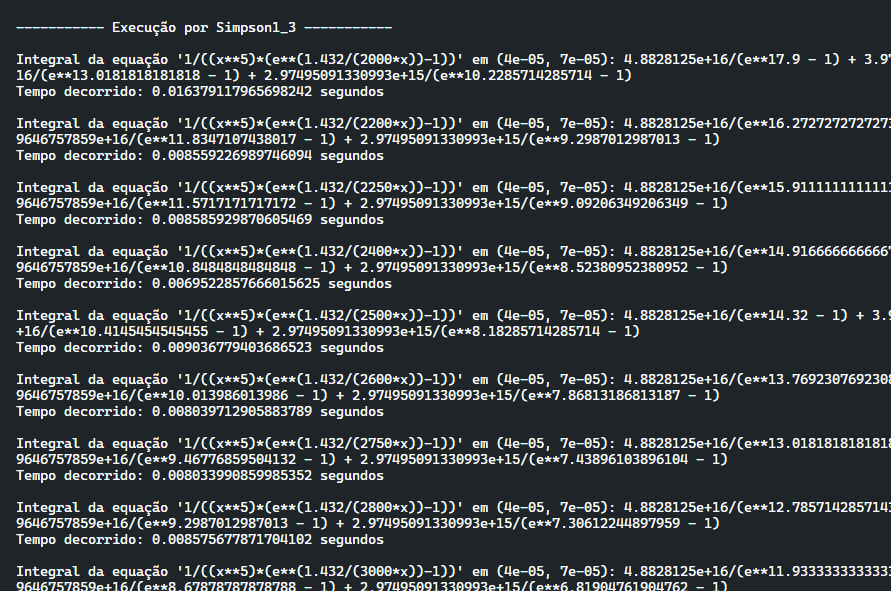
Um único arquivo de entrada é necessário, deve chamá-lo de “**trapezio\_multiplo.csv**” e estar dentro da pasta **“entradas”**, o arquivo deve ser separado por vírgulas, e ter os seguintes cabeçalhos:

* **equation**: A string que contém a função a ser avaliada. Deve estar nos padrões do SymPy, então:
  + A única variável deve ser “x”
  + deve ser escrito como **x\*\*3**
  + **cosseno de X** deve ser escrito como **cos(x)**
  + **seno de X** deve ser escrito como **sin(x)**
  + números reais devem ser escritos com ponto: cinco vírgula cinco = 5.5
* **a:** O limite inferior de integração.
* **b:** O limite superior de integração.
* **h:** O intervalo de discretizações.

## A saída é o valor aproximado da derivada no ponto xi, que é retornado pela função solve\_by\_trapezio\_multiplo.

## Problemas teste

Foram realizados testes utilizando os princípios e técnicas descritos no livro **"Cálculo Numérico" de Neide Franco**. A implementação foi avaliada em relação a três problemas apresentados no livro. Os resultados obtidos foram consistentes com métodos amplamente reconhecidos, evidenciando a eficácia da implementação.

  
*Imagem 7.1 - Retornos gerais das iterações do algoritmo*

## Dificuldades enfrentadas

Não houveram dificuldades.

# Método 8 (Simpson 1/3)

## Estratégia de Implementação

Este método é uma técnica de integração numérica que proporciona uma aproximação mais precisa da integral de uma função em um dado intervalo. A função recebe como entrada a equação em formato de string e os limites de integração A e B.

## Estrutura dos Arquivos de Entrada/Saída

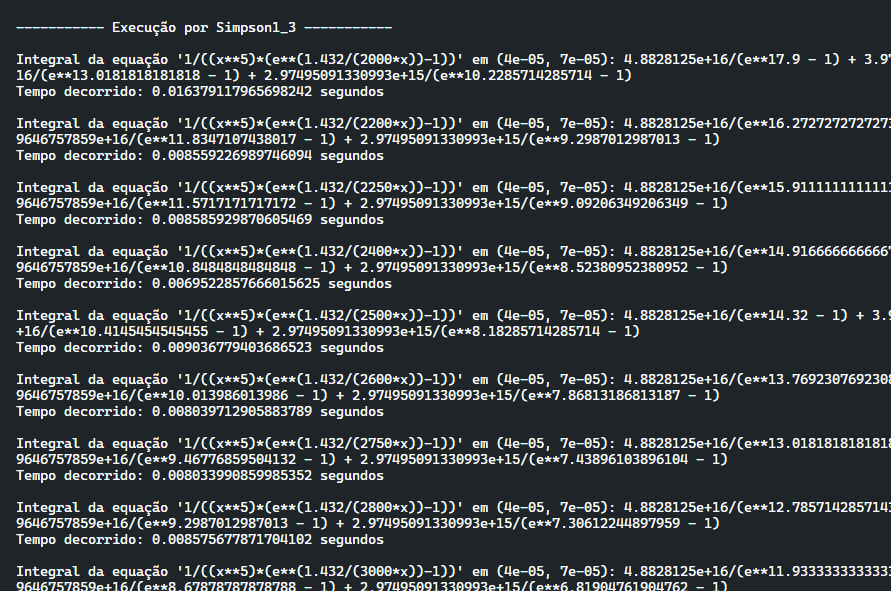
Um único arquivo de entrada é necessário, deve chamá-lo de “**simpson\_1\_3.csv**” e estar dentro da pasta **“entradas”**, o arquivo deve ser separado por vírgulas, e ter os seguintes cabeçalhos:

* **equation**: A string que contém a função a ser avaliada. Deve estar nos padrões do SymPy, então:
  + A única variável deve ser “x”
  + deve ser escrito como **x\*\*3**
  + **cosseno de X** deve ser escrito como **cos(x)**
  + **seno de X** deve ser escrito como **sin(x)**
  + números reais devem ser escritos com ponto: cinco vírgula cinco = 5.5
* **a:** O limite inferior de integração.
* **b:** O limite superior de integração.
* **h:** O intervalo de discretizações.

## A saída é o valor aproximado da integral no ponto xi, no intervalo especificado, que é retornado pela função solve\_by\_simpson\_1\_3.

## Problemas teste

Foram realizados testes utilizando os princípios e técnicas descritos no livro **"Cálculo Numérico" de Neide Franco**. A implementação foi avaliada em relação a três problemas apresentados no livro. Os resultados obtidos foram consistentes com métodos amplamente reconhecidos, evidenciando a eficácia da implementação.

  
*Imagem 8.1 - Retornos gerais das iterações do algoritmo*

## Dificuldades enfrentadas

Não houveram dificuldades.

# Método 9 (Simpson 3/8)

## Estratégia de Implementação

Este método é uma técnica avançada de integração numérica, que fornece uma aproximação mais precisa para a integral de uma função em um intervalo especificado. A função aceita como entrada a equação em formato de string e os limites de integração A e B. Utilizando a fórmula de Simpson de 3/8:

## Problemas teste

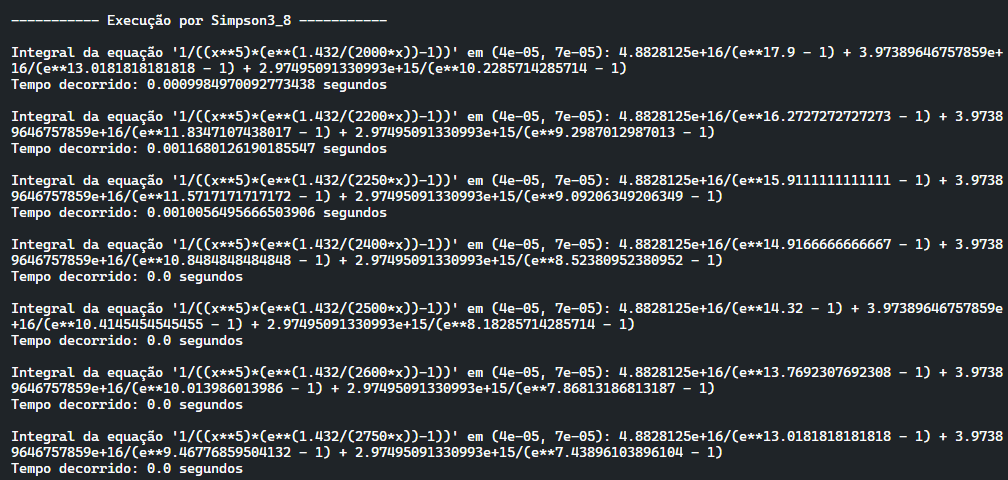
Um único arquivo de entrada é necessário, deve chamá-lo de “**simpson\_3\_8.csv**” e estar dentro da pasta **“entradas”**, o arquivo deve ser separado por vírgulas, e ter os seguintes cabeçalhos:

* **equation**: A string que contém a função a ser avaliada. Deve estar nos padrões do SymPy, então:
  + A única variável deve ser “x”
  + deve ser escrito como **x\*\*3**
  + **cosseno de X** deve ser escrito como **cos(x)**
  + **seno de X** deve ser escrito como **sin(x)**
  + números reais devem ser escritos com ponto: cinco vírgula cinco = 5.5
* **a:** O limite inferior de integração.
* **b:** O limite superior de integração.
* **h:** O intervalo de discretizações.

## A saída é o valor aproximado da integral no ponto xi, no intervalo especificado, que é retornado pela função solve\_by\_simpson\_3\_8.

## Problemas teste

Foram realizados testes utilizando os princípios e técnicas descritos no livro **"Cálculo Numérico" de Neide Franco**. A implementação foi avaliada em relação a três problemas apresentados no livro. Os resultados obtidos foram consistentes com métodos amplamente reconhecidos, evidenciando a eficácia da implementação.

  
*Imagem 9.1 - Retornos gerais das iterações do algoritmo*

## Dificuldades enfrentadas

Não houveram dificuldades.

# 

# Método 10 (Extrapolação de Richards)

## Estratégia de Implementação

## Esta técnica é usada para extrapolar valores de uma função conhecida em pontos além do conjunto de dados original. A implementação envolve a resolução de um sistema linear para obter os coeficientes da função extrapoladora. A função aceita três parâmetros: um conjunto de pontos conhecidos x e y e um conjunto de pontos X para os quais desejamos extrapolar os valores Y.

## Estrutura dos Arquivos de Entrada/Saída

Para cada sistema de equações, deve ser criado um arquivo de entrada, deve chamá-lo de “**question{index}.csv**”, e deve colocá-lo na pasta “**entradas/richard/**”. O arquivo deve ser separado por vírgulas, e ter os seguintes cabeçalhos:

* **x**: Valores da variável independente.
* **y**: Valores da variável dependente.

## Problemas teste

Foram realizados testes utilizando os princípios e técnicas descritos no livro **"Cálculo Numérico" de Neide Franco**. A implementação foi avaliada em relação a três problemas apresentados no livro.

## Dificuldades enfrentadas

Não consegui finalizar esse método por conta de um problema com algum erro com a linguagem.

# Método 11 (Quadratura de Gauss)

## Estratégia de Implementação

## Este método é utilizado para calcular a integral de uma função em um intervalo específico. A estratégia adotada envolveu o uso de coeficientes e pesos de Gauss pré-definidos para o cálculo da integral, juntamente com uma transformação dos limites do intervalo de integração para o intervalo padrão [−1,1].

## Estrutura dos Arquivos de Entrada/Saída

Um único arquivo de entrada é necessário, deve chamá-lo de “**gauss.csv**” e estar dentro da pasta **“entradas”**, o arquivo deve ser separado por vírgulas, e ter os seguintes cabeçalhos:

* **equation**: A string que contém a função a ser avaliada. Deve estar nos padrões do SymPy, então:
  + A única variável deve ser “x”
  + deve ser escrito como **x\*\*3**
  + **cosseno de X** deve ser escrito como **cos(x)**
  + **seno de X** deve ser escrito como **sin(x)**
  + números reais devem ser escritos com ponto: cinco vírgula cinco = 5.5
* **a:** O limite inferior de integração.
* **b:** O limite superior de integração.

## A saída é o valor aproximado da integral no intervalo especificado, que é retornado pela função solve\_by\_gauss.

## Problemas teste

Foram realizados testes utilizando os princípios e técnicas descritos no livro **"Cálculo Numérico" de Neide Franco**. A implementação foi avaliada em relação a três problemas apresentados no livro. Os resultados obtidos foram consistentes com métodos amplamente reconhecidos, evidenciando a eficácia da implementação.

## Dificuldades enfrentadas

Não houveram dificuldades.

# Conclusão

Ao concluir este projeto, emergem reflexões importantes sobre a diversidade e a aplicabilidade dos métodos numéricos. Exploramos uma variedade de técnicas, desde a interpolação e extrapolação até a integração e derivação numérica, cada uma com suas particularidades e contextos de aplicação. A implementação desses métodos não só enriqueceu nosso conhecimento técnico, mas também nos proporcionou ferramentas práticas para abordar problemas complexos do mundo real.

Através deste trabalho, ficou evidente que o campo da análise numérica é vasto e multifacetado. A escolha do método apropriado em cada situação requer uma compreensão detalhada do problema em questão, bem como das limitações e pontos fortes de cada técnica. A precisão, a eficiência computacional e a facilidade de implementação são fatores cruciais que influenciam essa decisão.

Além disso, a realização de testes com problemas variados demonstrou a relevância de uma abordagem prática para validar e compreender os métodos. Essa experimentação prática é fundamental para ganhar confiança nas ferramentas numéricas e na sua capacidade de fornecer soluções confiáveis.

Em suma, este projeto não apenas ampliou nosso repertório de métodos numéricos, mas também reforçou a importância da análise numérica no contexto moderno, onde a computação e a modelagem matemática são essenciais para a solução de desafios emergentes em diversas áreas. Este trabalho serve como uma base sólida para futuras investigações e aplicações, abrindo caminho para descobertas e inovações no campo da análise numérica.