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

Lista de Figuras

Método da Regressão linear

Método da Interpolação Lagrange

Método de Diferenças Divididas de Newton

Método da Derivadas de 1a ordem

Método da Derivadas de 2a ordem

Método da Integração Trapézio Simples

Método de Integração Trapézio Múltiplo

Método de Simpson 1/3

Método de Simpson 3/8

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:

$$a = \frac{n^*(\sum xy) - (\sum x)(\sum y)}{n^*(\sum x^2) - (\sum x)^2}$$

$$b = \frac{(\sum y) - a^*(\sum x)}{n}$$

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.

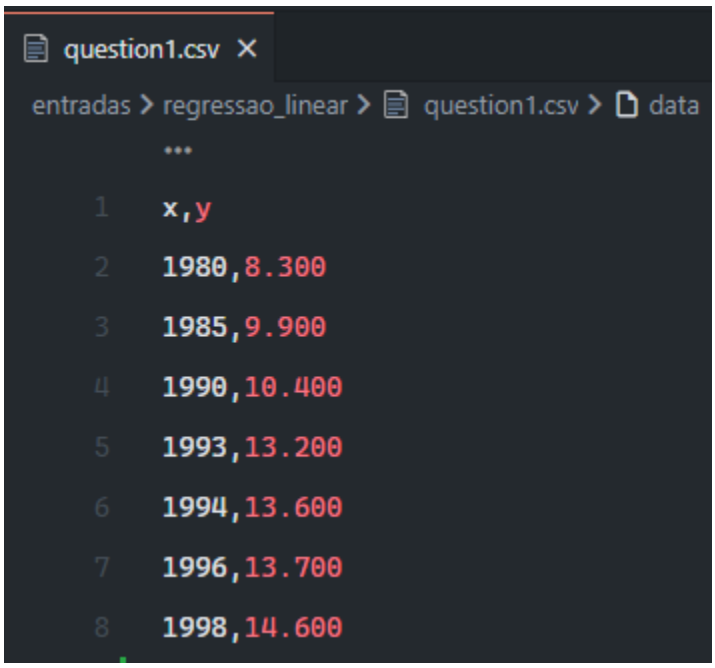

Imagem 1.1 - Exemplo de arquivo de entrada do método regressão linear

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

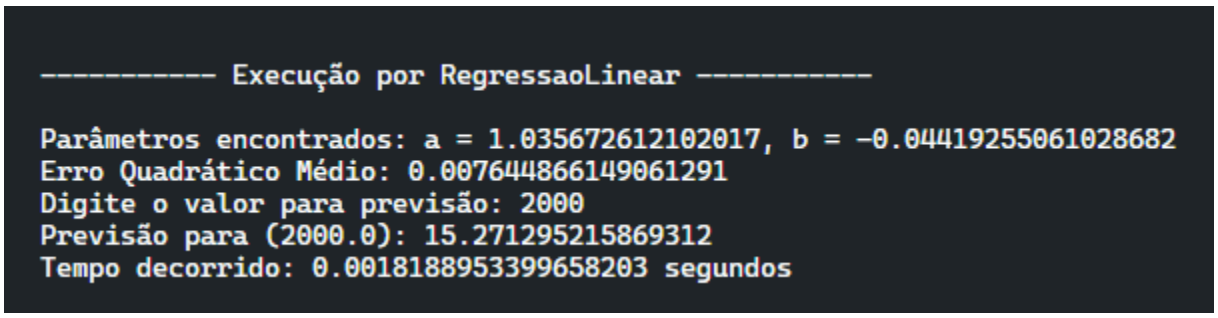

Imagem 1.2 - Retornos gerais das iterações do algoritmo

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

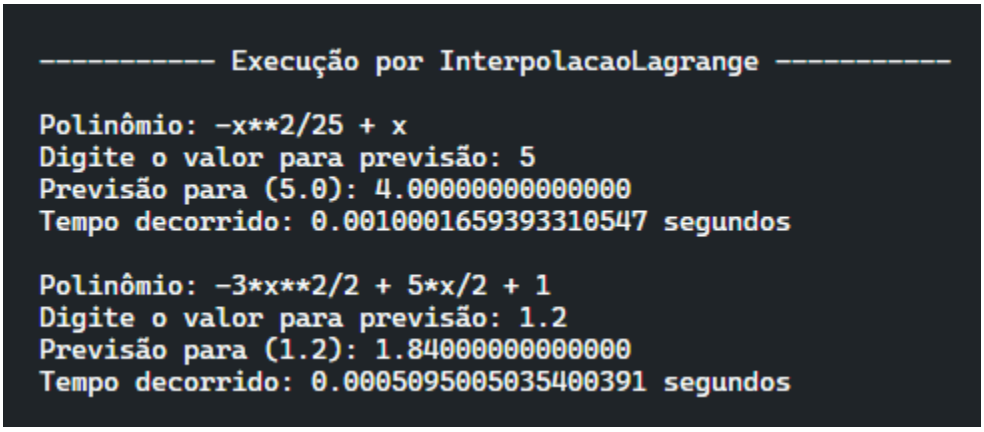

Imagem 2.1 - Retornos gerais das iterações do algoritmo

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.

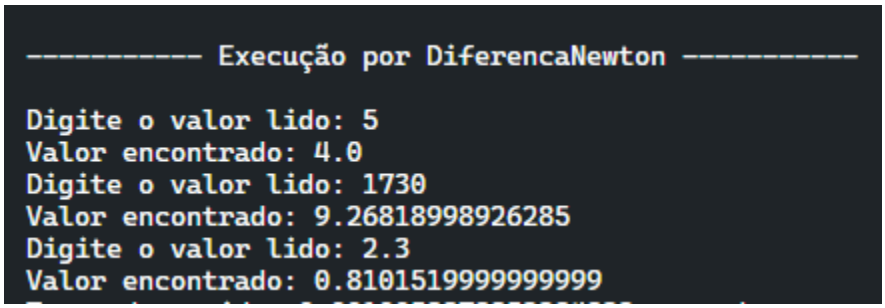

Imagem 3.1 - Retornos gerais das iterações do algoritmo

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"
    - x^3 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.

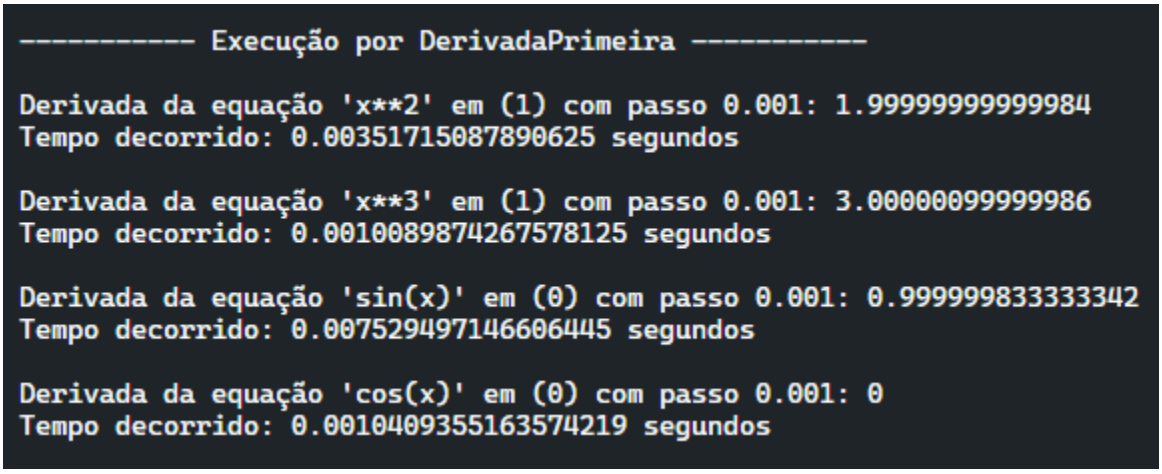

Imagem 4.1 - Retornos gerais das iterações do algoritmo

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"
  - x^3 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.

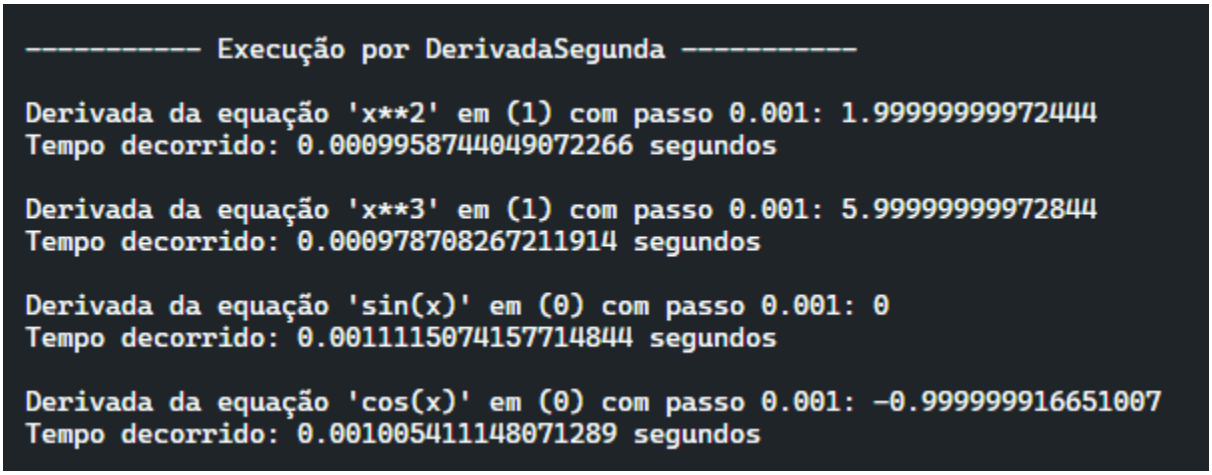

Imagem 5.1 - Retornos gerais das iterações do algoritmo

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"
    - x^3 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.

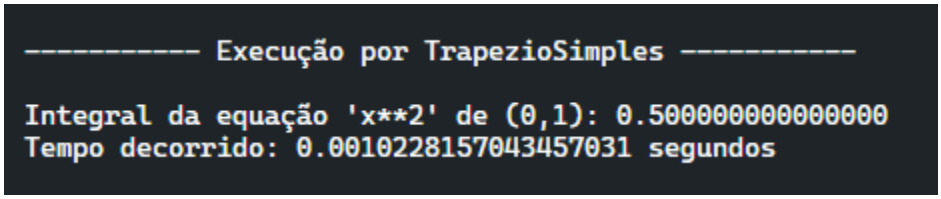

Imagem 6.1 - Retornos gerais das iterações do algoritmo

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"
  - x^3 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.

```
---------- Execução por Simpson1_3 ----------

Integral da equação '1/((x**5)*(e**(1.432/(2000*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**17.9 - 1) + 3.97
16/(e**13.0181818181818 - 1) + 2.97495091330993e+15/(e**10.2285714285714 - 1)
Tempo decorrido: 0.016379117965698242 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2200*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**16.272727272727
9646757859e+16/(e**11.8347107438017 - 1) + 2.97495091330993e+15/(e**9.2987012987013 - 1)
Tempo decorrido: 0.008559226989746094 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2250*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**15.911111111111
9646757859e+16/(e**11.5717171717172 - 1) + 2.97495091330993e+15/(e**9.09206349206349 - 1)
Tempo decorrido: 0.008585929870605469 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2400*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**14.916666666666
9646757859e+16/(e**10.84848484848848 - 1) + 2.97495091330993e+15/(e**8.52380952380952 - 1)
Tempo decorrido: 0.0069522857666015625 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2500*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**14.32 - 1) + 3.9
+16/(e**10.4145454545455 - 1) + 2.97495091330993e+15/(e**8.18285714285714 - 1)
Tempo decorrido: 0.009036779403686523 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2600*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**13.769230769230
9646757859e+16/(e**10.013986013986 - 1) + 2.97495091330993e+15/(e**7.86813186813187 - 1)
Tempo decorrido: 0.008039712905883789 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2750*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**13.018181818181
9646757859e+16/(e**9.46776859504132 - 1) + 2.97495091330993e+15/(e**7.43896103896104 - 1)
Tempo decorrido: 0.008033990859985352 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2800*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**12.785714285714
9646757859e+16/(e**9.2987012987013 - 1) + 2.97495091330993e+15/(e**7.30612244897959 - 1)
Tempo decorrido: 0.008575677871704102 segundos

Integral da equação '1/((x**5)*(e**(1.432/(3000*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**11.933333333333
9646757859e+16/(e**8.678787878787878 - 1) + 2.97495091330993e+15/(e**6.8190476190476? - 1)
```

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"
  - x^3 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.

```
---------- Execução por Simpson1_3 ----------

Integral da equação '1/((x**5)*(e**(1.432/(2000*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**17.9 - 1) + 3.97
16/(e**13.0181818181818 - 1) + 2.97495091330993e+15/(e**10.2285714285714 - 1)
Tempo decorrido: 0.016379117965698242 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2200*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**16.272727272727
9646757859e+16/(e**11.8347107438017 - 1) + 2.97495091330993e+15/(e**9.2987012987013 - 1)
Tempo decorrido: 0.008559226989746094 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2250*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**15.911111111111
9646757859e+16/(e**11.5717171717172 - 1) + 2.97495091330993e+15/(e**9.09206349206349 - 1)
Tempo decorrido: 0.008585929870605469 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2400*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**14.916666666666
9646757859e+16/(e**10.84848484848848 - 1) + 2.97495091330993e+15/(e**8.52380952380952 - 1)
Tempo decorrido: 0.0069522857666015625 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2500*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**14.32 - 1) + 3.9
+16/(e**10.4145454545455 - 1) + 2.97495091330993e+15/(e**8.18285714285714 - 1)
Tempo decorrido: 0.009036779403686523 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2600*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**13.769230769230
9646757859e+16/(e**10.013986013986 - 1) + 2.97495091330993e+15/(e**7.86813186813187 - 1)
Tempo decorrido: 0.008039712905883789 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2750*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**13.018181818181
9646757859e+16/(e**9.46776859504132 - 1) + 2.97495091330993e+15/(e**7.43896103896104 - 1)
Tempo decorrido: 0.008033990859985352 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2800*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**12.785714285714
9646757859e+16/(e**9.2987012987013 - 1) + 2.97495091330993e+15/(e**7.30612244897959 - 1)
Tempo decorrido: 0.008575677871704102 segundos

Integral da equação '1/((x**5)*(e**(1.432/(3000*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**11.933333333333
9646757859e+16/(e**8.678787878787878 - 1) + 2.97495091330993e+15/(e**6.8190476190476? - 1)
```

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"
    - x^3 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.

```
---------- Execução por Simpson3_8 ----------

Integral da equação '1/((x**5)*(e**(1.432/(2000*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**17.9 - 1) + 3.97389646757859e+
16/(e**13.0181818181818 - 1) + 2.97495091330993e+15/(e**10.2285714285714 - 1)
Tempo decorrido: 0.0009984970092773438 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2200*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**16.2727272727273 - 1) + 3.9738
96467578 59e+16/(e**11.83471074380 17 - 1) + 2.97495091330993e+15/(e**9.2987012987013 - 1)
Tempo decorrido: 0.0011680126190185547 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2250*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**15.9111111111111 - 1) + 3.9738
96467578 59e+16/(e**11.5717171717172 - 1) + 2.97495091330993e+15/(e**9.09206349206349 - 1)
Tempo decorrido: 0.0010056495666503906 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2400*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**14.9166666666667 - 1) + 3.9738
96467578 59e+16/(e**10.84848484848848 - 1) + 2.97495091330993e+15/(e**8.52380952380952 - 1)
Tempo decorrido: 0.0 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2500*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**14.32 - 1) + 3.973896467578 59e
+16/(e**10.41454545455455 - 1) + 2.97495091330993e+15/(e**8.18285714285714 - 1)
Tempo decorrido: 0.0 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2600*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**13.7692307692308 - 1) + 3.9738
96467578 59e+16/(e**10.013986013986 - 1) + 2.97495091330993e+15/(e**7.86813186813187 - 1)
Tempo decorrido: 0.0 segundos

Integral da equação '1/((x**5)*(e**(1.432/(2750*x))-1))' em (4e-05, 7e-05): 4.8828125e+16/(e**13.0181818181818 - 1) + 3.9738
96467578 59e+16/(e**9.46776859504132 - 1) + 2.97495091330993e+15/(e**7.43896103896104 - 1)
Tempo decorrido: 0.0 segundos
```

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"
    - x^3 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.