

**Universidade Estadual de Santa Cruz – UESC**

**Relatórios de Implementações de Métodos da Disciplina Análise Numérica**

**Relatório de implementações realizadas por Thiago de Freitas Gonçalves**

**Disciplina Análise Numérica.**

**Curso Ciência da Computação**

**Semestre 202X.Y**

**Professor Gesil Sampaio Amarante II**

**Ilhéus – BA**

**2023**

ÍNDICE

Lista de Figuras

Linguagem Escolhida e justificativas

A justificativa para a escolha da linguagem Python se deve à sua facilidade de acesso e implementação de soluções matemáticas. Além disso, devido à grande quantidade de bibliotecas, métodos e materiais disponíveis, o Python na versão 3.10 ou superior é a escolha óbvia para a construção deste trabalho.

Método 1: Bissecção

# Estratégia de Implementação:

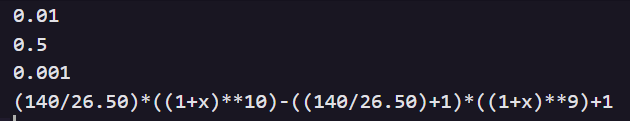
O programa começa coma leitura do arquivo de entrada “Entrada.txt” e instanciação de métodos para calcular seno, cosseno e tangente, assim como a abertura do arquivo de saída “Saida.txt”. O arquivo de entrada é divido em primeira linha a, segunda linha b, terceira linha erro, quarta linha função, logo apos a leitura f(a) e f(b) são encontrados junto com o valor de c, e há verificação se f(c) é raiz ou se 20 execuções foram concluídas. Se não um novo a ou b são setados com base na mudança de sinal de f(c).

# Estrutura dos Arquivos de Entrada/Saída

O arquivo de entrada esta dividido em 4 linhas, primeira linha a, segunda linha b, terceira linha erro, quarta linha função sendo um arquivo no txt denominado “Entrada.txt”, assim como o arquivo de saida esta dividido por linha onde cada linha representa uma execução do algoritimo mostrando respectivamento o valor de a, b, c, f(a), f(b) e f(c) sendo um arquivo de formato txt denominado “Saida.txt”.

# **Problema teste 1:** 3.8

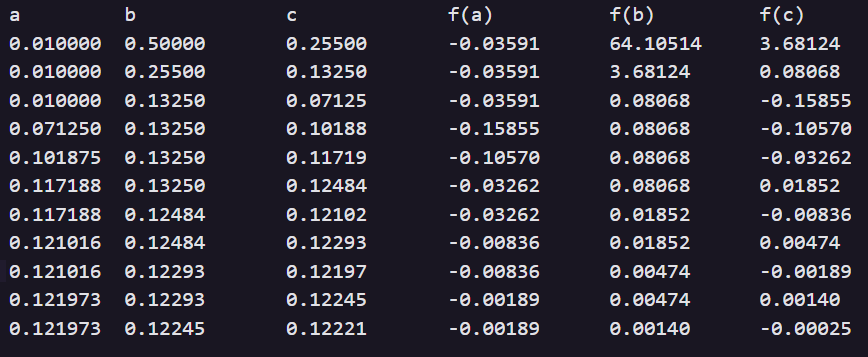
figura 1.0



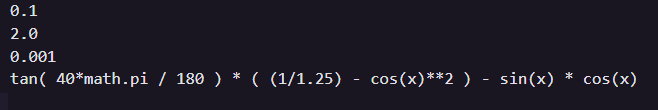
Fonte: Compilação do autor.

Saída:

figura 1.1

Fonte: Compilação do autor.

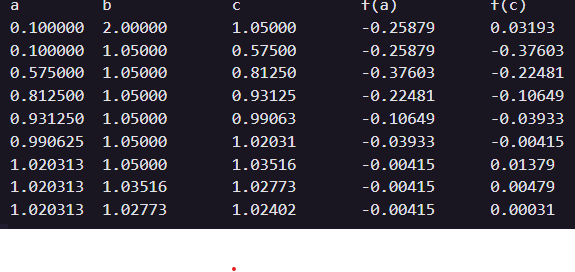
# **Problema teste 2:** 3.6

figura 1.2

Fonte: Compilação do autor.

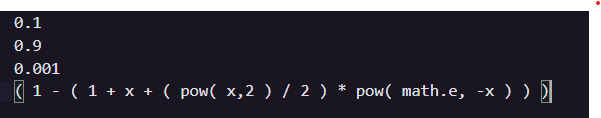
Saída:

figura 1.3

Fonte: Compilação do autor.

# **Problema teste 3:** 3.3

figura 1.4

Fonte: Compilação do autor.

Saída:

figura 1.5

# 

Fonte: Compilação do autor.

# Dificuldades enfrentadas:

Entender os problemas e tranformalos em codigo.

Método 2: Secante

# Estratégia de Implementação:

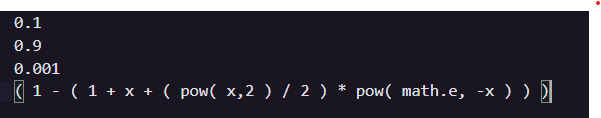
O programa começa coma leitura do arquivo de entrada “Entrada.txt” e instanciação de métodos para calcular seno, cosseno e tangente, assim como a abertura do arquivo de saída “Saida.txt”. O arquivo de entrada é divido em primeira linha a, segunda linha b, terceira linha erro, quarta linha função, logo apos a leitura f(a) e f(b) são encontrados e há verificação se aux1 é menor que o erro ou se 4 execuções foram feitas. Se não um novo a ou b são setados com base nos resultados.

# Estrutura dos Arquivos de Entrada/Saída

O arquivo de entrada esta dividido em 4 linhas, primeira linha a, segunda linha b, terceira linha erro, quarta linha função sendo um arquivo no txt denominado “Entrada.txt”, assim como o arquivo de saida esta dividido por linha onde cada linha representa uma execução do algoritimo mostrando respectivamento o valor de a, b, erro, f(a), f(b) sendo um arquivo de formato txt denominado “Saida.txt”.

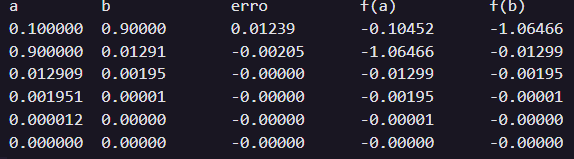
# Problema teste 3.3:

Figura 2.0

Fonte: Compilação do autor.

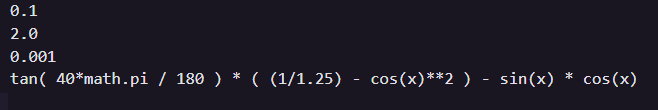
Saída:

Figura 2.1

Fonte: Compilação do autor.

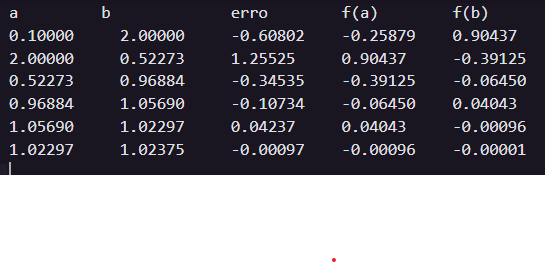
# Problema teste 3.6:

Figura 2.2

Fonte: Compilação do autor.

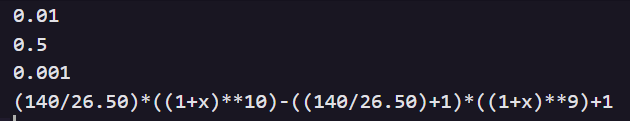
Saída:

Figura 2.3

Fonte: Compilação do autor.

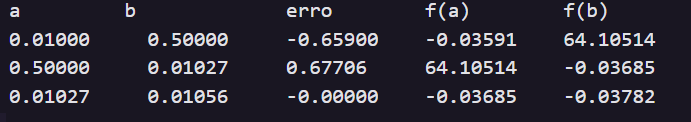
# Problema teste 3.8:

Figura 2.4

Fonte: Compilação do autor.

Saída:

Figura 2.5

Fonte: Compilação do autor.

# Dificuldades enfrentadas:

Entender os problemas e tranformalos em codigo.

Método 3: Fatoração LU

# Estratégia de Implementação:

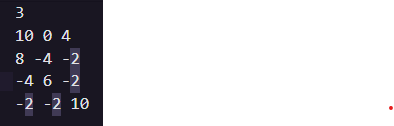
A implementação utiliza a estratégia de fatoração LU com pivoteamento parcial para resolver sistemas lineares. Ele lê os dados de entrada, inicializa as matrizes L e U, pivoteia se necessário, elimina abaixo da diagonal e resolve o sistema linear usando substituição progressiva e regressiva.

# Estrutura dos Arquivos de Entrada/Saída

O arquivo de entrada esta em formato txt denominado "Entrada.txt" conténdo as informações necessárias para resolver o sistema linear, incluindo o tamanho da matriz, os elementos do vetor b e os elementos da matriz A. O arquivo de saída está em formato txt denominao "Saida.txt" contém a solução do sistema linear em uma única linha

# Problema 1: 4.1

Figura 3.0



Saída:

Figura 3.1



# Problema 2: 4.3

# Problema 3: 4.6

# Dificuldades enfrentadas:

Entender os problemas e tranformalos em codigo.

Método 4: Eliminação de Gauss

# Estratégia de Implementação:

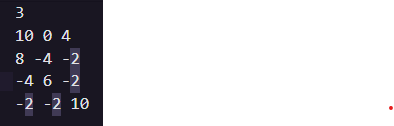
A primeira parte do código lê os dados de entrada de um arquivo de texto contendo o número de equações (N), o vetor b e a matriz A. Os dados são armazenados em variáveis. A segunda parte do código executa o método da eliminação de Gauss para resolver o sistema de equações. Ele começa anexando o vetor b à matriz A como uma coluna adicional e, em seguida, aplica as etapas do método da eliminação de Gauss iterativamente para obter a solução do sistema de equações

# Estrutura dos Arquivos de Entrada/Saída

O arquivo de entrada para o código em Python contém um número inteiro N, seguido por um vetor b e uma matriz A. O arquivo de saída contém as soluções do sistema linear Ax=b encontradas pela biblioteca NumPy e pelo método da eliminação de Gauss implementado manualmente, ambas como uma lista de números separados por espaço.

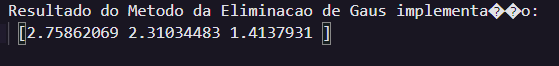
# Problema 1: 4.1

Figura 4.0



Saída:

Figura 4.1



# Problema 2: 4.3

# Problema 3: 4.6

# Dificuldades enfrentadas:

Entender os problemas e tranformalos em codigo.

Método 5: Jacobi

# Estratégia de Implementação:

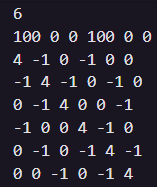
A implementação do método de Jacobi para resolver sistemas lineares Ax=b. Ele lê os dados do arquivo de entrada e cria a matriz A e o vetor b. Em seguida, separa a diagonal da matriz A e calcula sua inversa. Define os valores iniciais para o vetor x e os critérios de parada para a iteração do método. Dentro do loop principal, o código executa o método de Jacobi interativamente, atualizando o vetor x até que a diferença entre a solução atual e a anterior seja menor do que a tolerância especificada ou até que se atinja o número máximo de iterações permitido.

# Estrutura dos Arquivos de Entrada/Saída

O arquivo de entrada "input.txt" deve conter os dados do sistema linear a ser resolvido pelo método de Jacobi, incluindo a matriz A e o vetor b. O arquivo de saída "output.txt" conterá a solução do sistema encontrada pelo método de Jacobi, que será escrita em uma única linha com a mensagem "Solução encontrada pelo método de Jacobi seguida pelo vetor com as soluções do sistema

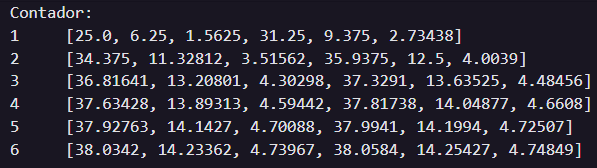
# Problema 1: 5.1

Figure 5.0



# Saída:

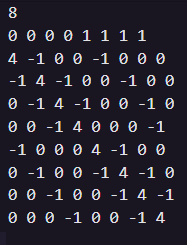
Figure 5.1



# Problema 2: 5.2

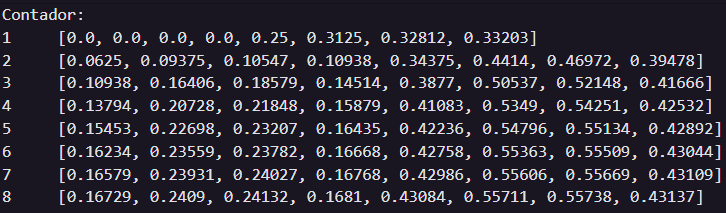
# Problema 3: 5.5

Figure 5.2



Saída:

Figure 5.3



# Dificuldades enfrentadas:

Entender os problemas e tranformalos em codigo.

Método 6: Gauss-Seidel

# Estratégia de Implementação:

Descreva as escolhas que teve que fazer para poder superar eventuais dificuldades com limitações da linguagem, do ambiente de programação, etc.

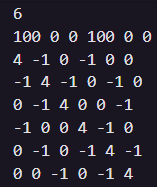
De que forma resolveu tratar os erros, critérios de parada, formato de saída e quaisquer outros fatores importantes para a solução dos problemas.

# Estrutura dos Arquivos de Entrada/Saída

Descreva o formato dos arquivos de entrada e saída e justifique as escolhas.

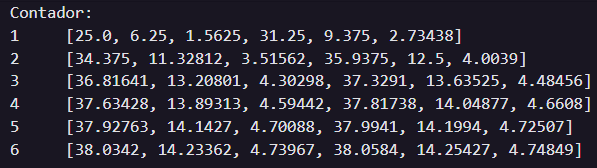
# Problema 1: 5.1

Figure 6.0



Saída:

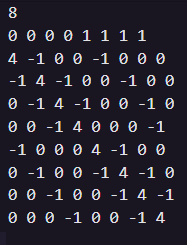
Figure 6.1



# Problema 2: 5.2

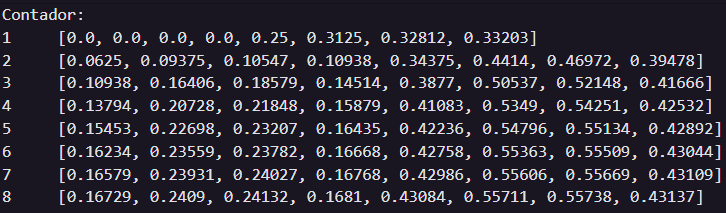
# Problema 3: 5.5

Figure 6.2



Saída:

Figure 6.3



# Dificuldades enfrentadas:

Entender os problemas e tranformalos em codigo.

Considerações Finais

A criação dos métodos demandou um esforço considerável, mas foi uma experiência empolgante e enriquecedora, permitindo-nos aprender profundamente e aplicar nossos conhecimentos de forma significativa.