

**Universidade Estadual de Santa Cruz – UESC**

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

### Relatório de implementações realizadas por Arthur Marques Azevedo

### Disciplina Análise Numérica. Curso Ciência da Computação Semestre 2023.1

### Professor Gesil Sampaio Amarante II

**Ilhéus – BA 2023**

ÍNDICE

[Lista de Figuras 4](#_TOC_250052)

[Linguagem(ns) Escolhida(s) e justificativas 7](#_TOC_250051)

[Método da Bissecção 8](#_TOC_250050)

[Estratégia de Implementação: 8](#_TOC_250049)

[Estrutura dos Arquivos de Entrada/Saída 8](#_TOC_250048)

[Problema teste 1, 2, 3. 8](#_TOC_250047)

[Dificuldades enfrentadas 10](#_TOC_250046)

[Método da Posição Falsa 11](#_TOC_250045)

[Estratégia de Implementação: 11](#_TOC_250044)

[Estrutura dos Arquivos de Entrada/Saída 11](#_TOC_250043)

[Problema teste 1, 2, 3. 11](#_TOC_250042)

[Dificuldades enfrentadas 13](#_TOC_250041)

[Método de Newton-Raphson 14](#_TOC_250040)

[Estratégia de Implementação: 14](#_TOC_250039)

[Estrutura dos Arquivos de Entrada/Saída 14](#_TOC_250038)

[Problema teste 1, 2, 3. 14](#_TOC_250037)

[Dificuldades enfrentadas 16](#_TOC_250036)

[Método da Secante 17](#_TOC_250035)

[Estratégia de Implementação: 17](#_TOC_250034)

[Estrutura dos Arquivos de Entrada/Saída 17](#_TOC_250033)

[Problema teste 1, 2, 3. 17](#_TOC_250032)

[Dificuldades enfrentadas 19](#_TOC_250031)

[Método de Eliminação de Gauss 20](#_TOC_250030)

[Estratégia de Implementação: 20](#_TOC_250029)

[Estrutura dos Arquivos de Entrada/Saída 20](#_TOC_250028)

[Problema teste 1, 2, 3. 20](#_TOC_250027)

[Dificuldades enfrentadas 22](#_TOC_250026)

[Método da Fatoração LU 23](#_TOC_250025)

[Estratégia de Implementação: 23](#_TOC_250024)

[Estrutura dos Arquivos de Entrada/Saída 23](#_TOC_250023)

[Problema teste 1, 2, 3. 23](#_TOC_250022)

[Dificuldades enfrentadas 25](#_TOC_250021)

[Método de Jacobi 26](#_TOC_250020)

[Estratégia de Implementação: 26](#_TOC_250019)

[Estrutura dos Arquivos de Entrada/Saída 26](#_TOC_250018)

[Problema teste 1, 2, 3. 26](#_TOC_250017)

[Dificuldades enfrentadas 28](#_TOC_250016)

[Método de Gauss-Seidel 29](#_TOC_250015)

[Estratégia de Implementação: 29](#_TOC_250014)

[Estrutura dos Arquivos de Entrada/Saída 29](#_TOC_250013)

[Problema teste 1, 2, 3. 29](#_TOC_250012)

[Dificuldades enfrentadas 31](#_TOC_250011)

[Método da Inversão 32](#_TOC_250010)

[Estratégia de Implementação: 32](#_TOC_250009)

[Estrutura dos Arquivos de Entrada/Saída 32](#_TOC_250008)

[Problema teste 1, 2, 3… 32](#_TOC_250007)

[Dificuldades enfrentadas 34](#_TOC_250006)

[Método do Número de Condição da Matriz 35](#_TOC_250005)

[Estratégia de Implementação: 35](#_TOC_250004)

[Estrutura dos Arquivos de Entrada/Saída 35](#_TOC_250003)

[Problema teste 1, 2, 3… 35](#_TOC_250002)

[Dificuldades enfrentadas 37](#_TOC_250001)

[Considerações Finais 38](#_TOC_250000)

# Lista de Figuras

### Método da Bissecção

Imagem 1.1 - Arquivo de Entrada do Problema 3.3 Imagem 1.2 - Arquivo de Saída do Problema 3.3 Imagem 1.3 - Arquivo de Entrada do Problema 3.6 Imagem 1.4 - Arquivo de Saída do Problema 3.6 Imagem 1.5 - Arquivo de Entrada do Problema 3.8 Imagem 1.6 - Arquivo de Saída do Problema 3.6

### Método da Posição Falsa

Imagem 2.1 - Arquivo de Entrada do Problema 3.3 Imagem 2.2 - Arquivo de Saída do Problema 3.3 Imagem 2.3 - Arquivo de Entrada do Problema 3.6 Imagem 2.4 - Arquivo de Saída do Problema 3.6 Imagem 2.5 - Arquivo de Entrada do Problema 3.8 Imagem 2.6 - Arquivo de Saída do Problema 3.8

### Método de Newton-Raphson

Imagem 3.1 - Arquivo de Entrada do Problema 3.3 Imagem 3.2 - Mensagem de Erro do Terminal Imagem 3.3 - Arquivo de Entrada do Problema 3.6 Imagem 3.4 - Mensagem de Erro do Terminal Imagem 3.5 - Arquivo de Entrada do Problema 3.8 Imagem 3.6 - Arquivo de Saída do Problema 3.8

### Método da Secante

Imagem 4.1 - Arquivo de Entrada do Problema 3.3 Imagem 4.2 - Arquivo de Saída do Problema 3.3 Imagem 4.3 - Arquivo de Entrada do Problema 3.6

Imagem 4.4 - Arquivo de Saída do Problema 3.6 Imagem 4.5 - Arquivo de Entrada do Problema 3.8 Imagem 4.6 - Arquivo de Saída do Problema 3.8

### Método da Eliminação de Gauss

Imagem 5.1 - Arquivo de Entrada do Problema 4.1 Imagem 5.2 - Arquivo de Saída do Problema 4.1 Imagem 5.3 - Arquivo de Entrada do Problema 4.3 Imagem 5.4 - Arquivo de Saída do Problema 4.3 Imagem 5.5 - Arquivo de Entrada do Problema 4.6 Imagem 5.6 - Arquivo de Saída do Problema 4.6

### Método da Fatoração LU

Imagem 6.1 - Arquivo de Entrada do Problema 4.1 Imagem 6.2 - Arquivo de Saída do Problema 4.1 Imagem 6.3 - Arquivo de Entrada do Problema 4.3 Imagem 6.4 - Arquivo de Saída do Problema 4.3 Imagem 6.5 - Arquivo de Entrada do Problema 4.6 Imagem 6.6 - Arquivo de Saída do Problema 4.6

### Método de Jacobi

Imagem 7.1 - Arquivo de Entrada do Problema 5.1 Imagem 7.2 - Arquivo de Saída do Problema 5.1 Imagem 7.3 - Arquivo de Entrada do Problema 5.2 Imagem 7.4 - Arquivo de Saída do Problema 5.2 Imagem 7.5 - Arquivo de Entrada do Problema 5.5 Imagem 7.6 - Arquivo de Saída do Problema 5.5

### Método de Gauss-Seidel

Imagem 8.1 - Arquivo de Entrada do Problema 5.1 Imagem 8.2 - Arquivo de Saída do Problema 5.1

Imagem 8.3 - Arquivo de Entrada do Problema 5.2 Imagem 8.4 - Arquivo de Saída do Problema 5.2 Imagem 8.5 - Arquivo de Entrada do Problema 5.5 Imagem 8.6 - Arquivo de Saída do Problema 5.5

### Método da Inversão

Imagem 9.1 - Arquivo de Entrada do Problema 5.1 Imagem 9.2 - Arquivo de Saída do Problema 5.1 Imagem 9.3 - Arquivo de Entrada do Problema 5.2 Imagem 9.4 - Arquivo de Saída do Problema 5.2 Imagem 9.5 - Arquivo de Entrada do Problema 5.5 Imagem 9.6 - Arquivo de Saída do Problema 5.5

### Método do Número de Condição da Matriz

Imagem 10.1 - Arquivo de Entrada do Problema 5.1 Imagem 10.2 - Arquivo de Saída do Problema 5.1 Imagem 10.3 - Arquivo de Entrada do Problema 5.2 Imagem 10.4 - Arquivo de Saída do Problema 5.2 Imagem 10.5 - Arquivo de Entrada do Problema 5.5 Imagem 10.6 - Arquivo de Saída do Problema 5.5

# Linguagem(ns) Escolhida(s) e justificativas

Python é uma linguagem de programação amplamente utilizada na análise numérica e científica, graças à sua sintaxe clara e legível e à grande variedade de bibliotecas disponíveis, como NumPy e SciPy, que foram as utilizadas durante as implementações dos métodos. Visto a facilidade em implementar os métodos utilizando essas bibliotecas, isso faz com que a própria linguagem e suas bibliotecas sejam uma vantagem durante as implementações, e não resultou em grandes empecilhos durante suas criações e execuções.

No entanto, é importante levar em conta as limitações de desempenho do Python em comparação com linguagens de baixo nível, como C ou Fortran, especialmente quando se trabalha com grandes conjuntos de dados ou se busca resultados em tempo real. Bibliotecas especializadas, como Numba ou Cython, podem ajudar a superar essas limitações, permitindo a compilação do código Python para obter um desempenho mais próximo ao de linguagens de baixo nível. Entretanto, para a finalidade de precisão, o uso dessas bibliotecas não foi feito durante a implementação dos métodos.

# Método da Bissecção

## Estratégia de Implementação:

O código implementa o método da bissecção para encontrar uma raiz de uma função dada em um intervalo especificado pelo usuário. Ele lê os dados de entrada de um arquivo de texto, que contém a expressão da função, os limites inferior e superior do intervalo e a tolerância desejada para a solução. Em seguida, usa a biblioteca SymPy para converter a expressão em uma função lambdify do Python e, em seguida, usa o método da bissecção para iterativamente reduzir o intervalo até que a diferença entre a solução atual e a solução anterior seja menor que a tolerância especificada. Finalmente, ele escreve a solução em um arquivo de texto.

## Estrutura dos Arquivos de Entrada/Saída

O programa lê os dados de entrada de um arquivo chamado "entrada.txt”. A primeira linha do arquivo contém a definição da função matemática a ser resolvida, que é uma expressão matemática no formato padrão. Na segunda linha, são indicados os pontos iniciais para os métodos numéricos. A terceira linha indica a tolerância, que é a precisão desejada na solução. Quanto menor a tolerância, mais precisa será a solução.

Por exemplo, um arquivo com as informações da seguinte forma pode ser utilizado:

(0.25 \* (x-2) + 0.1 \* sin(x)) 1, 2

0.00001

Neste caso, a primeira linha define a função matemática a ser resolvida. A segunda linha indica os pontos iniciais para os métodos numéricos, enquanto a terceira linha indica a tolerância desejada para a solução.

O arquivo de saída , o programa escreve a raiz encontrada utilizando o método em questão, o nome de arquivo de saída possui a seguinte configuração: "resultado\_<Nome do Metodo>.txt”.

## Problema teste 1, 2, 3...

Foi feito testes com utilizando utilizando os conceitos e técnicas apresentados no livro Cálculo Numérico, de Neide Franco. A implementação foi testada em 3

problemas apresentados no livro, e os resultados obtidos foram coerentes com outros métodos conhecidos, mostrando que a implementação funciona de forma eficiente.

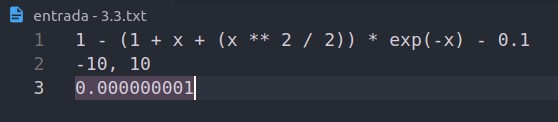


Imagem 1.1 - Arquivo de Entrada do Problema 3.3

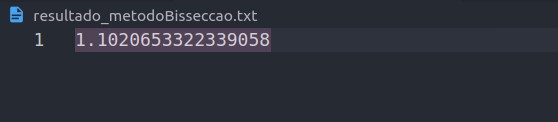


Imagem 1.2 - Arquivo de Saída do Problema 3.3

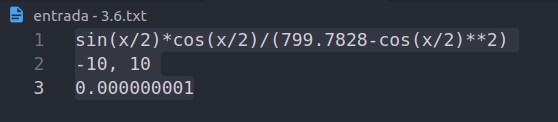


Imagem 1.3 - Arquivo de Entrada do Problema 3.6

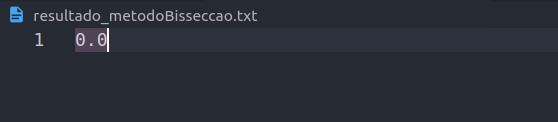


Imagem 1.4 - Arquivo de Saída do Problema 3.6

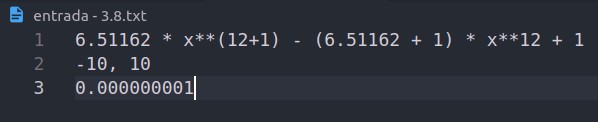


Imagem 1.5 - Arquivo de Entrada do Problema 3.8

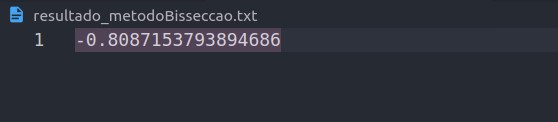


Imagem 1.6 - Arquivo de Saída do Problema 3.6

## Dificuldades enfrentadas

Não houveram dificuldades.

# Método da Posição Falsa

## Estratégia de Implementação:

O código implementa o método da posição falsa para encontrar a raiz de uma função dada em um intervalo dado. A função posicaoFalsa recebe como entrada a função função a ser avaliada, os extremos a e b do intervalo e uma tolerância tol. A função converge quando o valor da função for menor que a tolerância tol. Os extremos, entram em um loop que calcula o novo valor de c na linha 11, usando a fórmula da posição falsa, e verifica se a nova raiz c está suficientemente próxima da raiz verdadeira. Se for, retorna o valor de c. Caso contrário, atualiza o valor de a ou b baseado no sinal da função no ponto c. O loop é repetido até que o valor da função seja menor que a tolerância tol

## Estrutura dos Arquivos de Entrada/Saída

O programa lê os dados de entrada de um arquivo chamado "entrada.txt”. A primeira linha do arquivo contém a definição da função matemática a ser resolvida, que é uma expressão matemática no formato padrão. Na segunda linha, são indicados os pontos iniciais para os métodos numéricos. A terceira linha indica a tolerância, que é a precisão desejada na solução. Quanto menor a tolerância, mais precisa será a solução.

Por exemplo, um arquivo com as informações da seguinte forma pode ser utilizado:

(0.25 \* (x-2) + 0.1 \* sin(x)) 1, 2

0.00001

Neste caso, a primeira linha define a função matemática a ser resolvida. A segunda linha indica os pontos iniciais para os métodos numéricos, enquanto a terceira linha indica a tolerância desejada para a solução.

O arquivo de saída , o programa escreve a raiz encontrada utilizando o método em questão, o nome de arquivo de saída possui a seguinte configuração: "resultado\_<Nome do Metodo>.txt”.

## Problema teste 1, 2, 3...

Foi feito testes com utilizando utilizando os conceitos e técnicas apresentados no livro Cálculo Numérico, de Neide Franco. A implementação foi testada em 3

problemas apresentados no livro, e os resultados obtidos foram coerentes com outros métodos conhecidos, mostrando que a implementação funciona de forma eficiente.

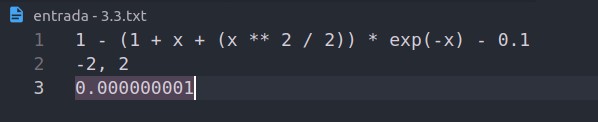


Imagem 2.1 - Arquivo de Entrada do Problema 3.3

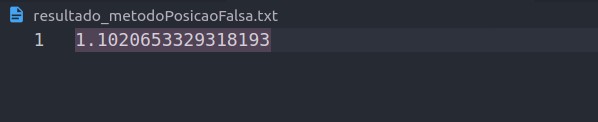


Imagem 2.2 - Arquivo de Saída do Problema 3.3

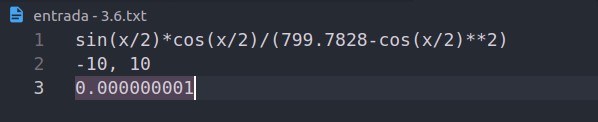


Imagem 2.3 - Arquivo de Entrada do Problema 3.6

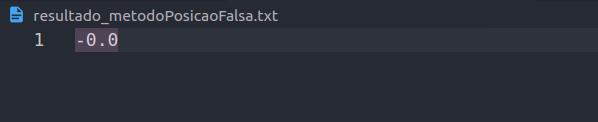


Imagem 2.4 - Arquivo de Saída do Problema 3.6

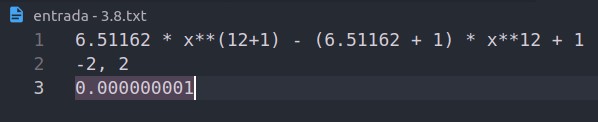


Imagem 2.5 - Arquivo de Entrada do Problema 3.8

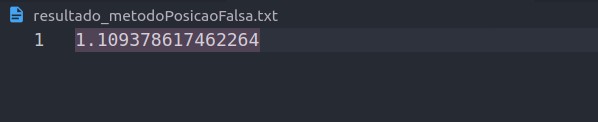


Imagem 2.6 - Arquivo de Saída do Problema 3.8

## Dificuldades enfrentadas

Não houveram dificuldades.

# Método de Newton-Raphson

## Estratégia de Implementação:

A função NewtonRaphson implementa o método de Newton-Raphson para encontrar a raiz de uma função. A estratégia de implementação consiste em usar a biblioteca SymPy para obter a derivada da função a ser resolvida, avaliar a derivada e a função no ponto de início a e aplicar a fórmula do método de Newton-Raphson para encontrar um novo ponto a em cada iteração até que a condição de parada seja satisfeita (ou seja, a diferença entre f(a) e zero é menor que a tolerância especificada).

## Estrutura dos Arquivos de Entrada/Saída

O programa lê os dados de entrada de um arquivo chamado "entrada.txt”. A primeira linha do arquivo contém a definição da função matemática a ser resolvida, que é uma expressão matemática no formato padrão. Na segunda linha, são indicados os pontos iniciais para os métodos numéricos. A terceira linha indica a tolerância, que é a precisão desejada na solução. Quanto menor a tolerância, mais precisa será a solução.

Por exemplo, um arquivo com as informações da seguinte forma pode ser utilizado:

(x\*\*2) + x - 6 1

0.00001

Neste caso, a primeira linha define a função matemática a ser resolvida. A segunda linha indica os pontos iniciais para os métodos numéricos, enquanto a terceira linha indica a tolerância desejada para a solução.

O arquivo de saída , o programa escreve a raiz encontrada utilizando o método em questão, o nome de arquivo de saída possui a seguinte configuração: "resultado\_<Nome do Metodo>.txt”.

## Problema teste 1, 2, 3...

Foi feito testes com utilizando utilizando os conceitos e técnicas apresentados no livro Cálculo Numérico, de Neide Franco. A implementação foi testada em 3 problemas apresentados no livro, e apenas um resultado foi obtido da maneira correta, visto a dificuldade enfrentada em utilizar derivadas na implementação.

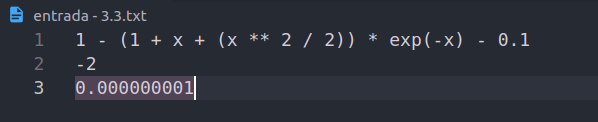


Imagem 3.1 - Arquivo de Entrada do Problema 3.3

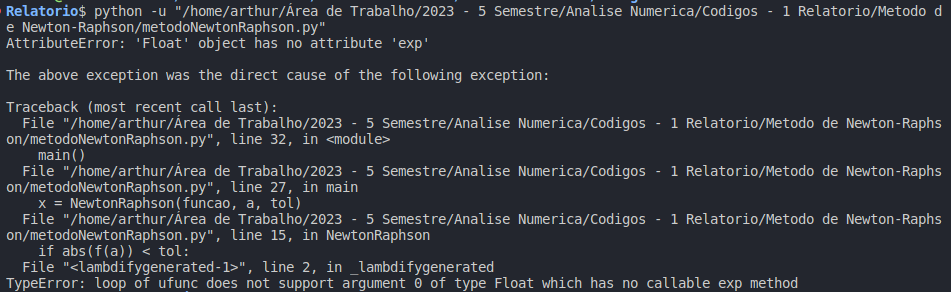


Imagem 3.2 - Mensagem de Erro do Terminal

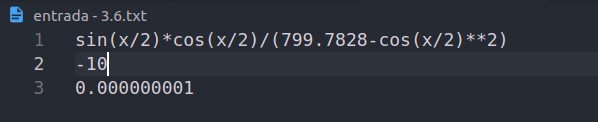


Imagem 3.3 - Arquivo de Entrada do Problema 3.6

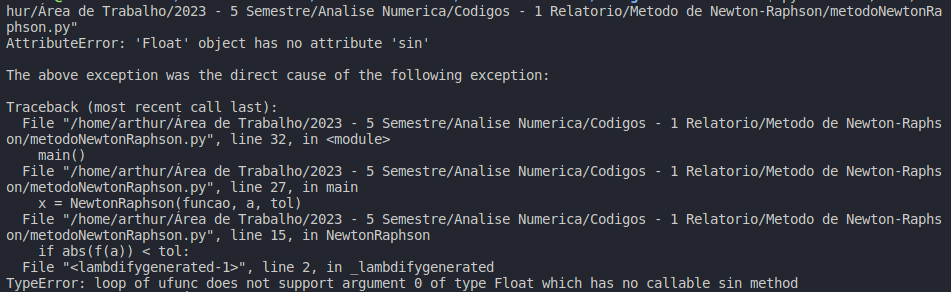


Imagem 3.4 - Mensagem de Erro do Terminal

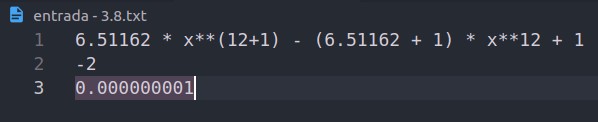


Imagem 3.5 - Arquivo de Entrada do Problema 3.8

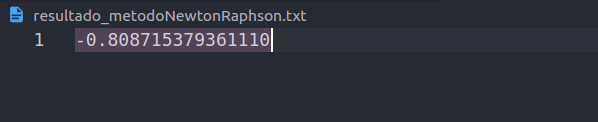


Imagem 3.6 - Arquivo de Saída do Problema 3.8

## Dificuldades enfrentadas

Não obtive a solução desejada durante o processo, soluções para as questões

3.3 e 3.6, ocorreram erros com o processo de derivar a função.

# Método da Secante

## Estratégia de Implementação:

O código implementa o método da secante para encontrar as raízes de uma função. O método da secante é uma técnica iterativa de aproximação que utiliza uma reta secante para estimar a raiz da função, que é obtida pela interseção da reta com o eixo x. O algoritmo recebe como entrada a função, os valores iniciais de a e b, que correspondem a dois pontos próximos à raiz, e a tolerância, que é o critério de parada da iteração. O método continua iterando até que o valor absoluto da função em x2, ponto de interseção da reta secante com o eixo x, seja menor que a tolerância.

## Estrutura dos Arquivos de Entrada/Saída

O programa lê os dados de entrada de um arquivo chamado "entrada.txt”. A primeira linha do arquivo contém a definição da função matemática a ser resolvida, que é uma expressão matemática no formato padrão. Na segunda linha, são indicados os pontos iniciais para os métodos numéricos. A terceira linha indica a tolerância, que é a precisão desejada na solução. Quanto menor a tolerância, mais precisa será a solução.

Por exemplo, um arquivo com as informações da seguinte forma pode ser utilizado:

(0.25 \* (x-2) + 0.1 \* sin(x)) 1, 2

0.00001

Neste caso, a primeira linha define a função matemática a ser resolvida. A segunda linha indica os pontos iniciais para os métodos numéricos, enquanto a terceira linha indica a tolerância desejada para a solução.

O arquivo de saída , o programa escreve a raiz encontrada utilizando o método em questão, o nome de arquivo de saída possui a seguinte configuração: "resultado\_<Nome do Metodo>.txt”.

## Problema teste 1, 2, 3...

Foi feito testes com utilizando utilizando os conceitos e técnicas apresentados no livro Cálculo Numérico, de Neide Franco. A implementação foi testada em 3 problemas apresentados no livro, e os resultados obtidos foram coerentes com

outros métodos conhecidos, mostrando que a implementação funciona de forma eficiente.

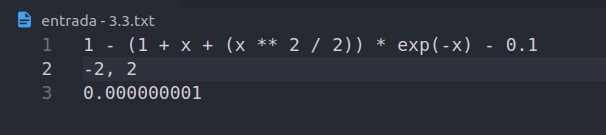


Imagem 4.1 - Arquivo de Entrada do Problema 3.3

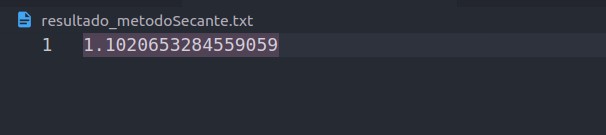


Imagem 4.2 - Arquivo de Saída do Problema 3.3

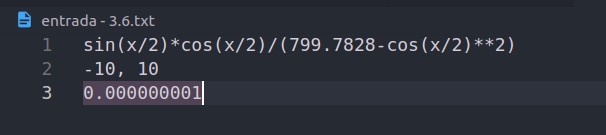


Imagem 4.3 - Arquivo de Entrada do Problema 3.6

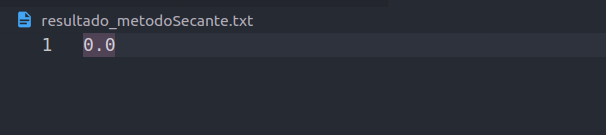


Imagem 4.4 - Arquivo de Saída do Problema 3.6

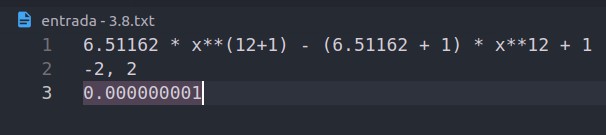


Imagem 4.5 - Arquivo de Entrada do Problema 3.8

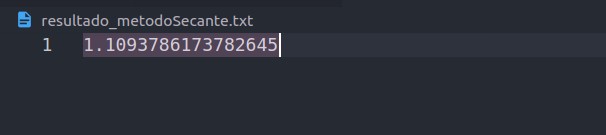


Imagem 4.6 - Arquivo de Saída do Problema 3.8

## Dificuldades enfrentadas

Não houveram dificuldades.

# Método de Eliminação de Gauss

## Estratégia de Implementação:

O código implementa o método de eliminação de Gauss para resolver sistemas lineares representados por uma matriz de coeficientes (A) e um vetor de termos independentes (b). O algoritmo de eliminação de Gauss é implementado usando dois loops aninhados para percorrer as colunas e linhas da matriz, respectivamente. Em cada iteração, é calculado um fator de eliminação que é usado para eliminar as entradas abaixo da diagonal principal na coluna atual. Em seguida, o vetor de termos independentes é atualizado com o mesmo fator de eliminação. Após a fase de eliminação, o método realiza a substituição regressiva para calcular a solução do sistema. O código usa a biblioteca NumPy para trabalhar com matrizes e vetores.

## Estrutura dos Arquivos de Entrada/Saída

O programa lê os dados de entrada de um arquivo chamado "entrada.txt”. A matriz é definida pelas primeiras n-1 linhas e n-1 colunas do arquivo, em que cada linha representa uma equação e cada coluna representa um coeficiente. O vetor de resultados é definido pela última coluna do arquivo, em que cada elemento corresponde ao resultado da equação correspondente na matriz.

Por exemplo, um arquivo com as informações da seguinte forma pode ser utilizado:

8 -4 -2 5

-4 6 -2 0

-2 -2 10 0

O arquivo de saída , o programa escreve a solução encontrada utilizando o método em questão, o nome de arquivo de saída possui a seguinte configuração: "resultado\_<Nome do Metodo>.txt”.

## Problema teste 1, 2, 3...

Foi feito testes com utilizando utilizando os conceitos e técnicas apresentados no livro Cálculo Numérico, de Neide Franco. A implementação foi testada em 3 problemas apresentados no livro, e os resultados obtidos foram coerentes com outros métodos conhecidos, mostrando que a implementação funciona de forma eficiente.

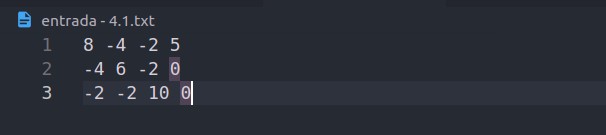


Imagem 5.1 - Arquivo de Entrada do Problema 4.1

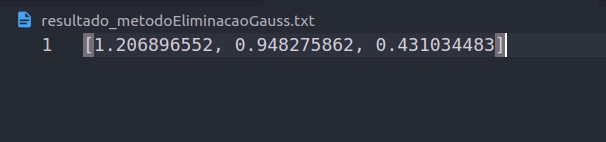


Imagem 5.2 - Arquivo de Saída do Problema 4.1

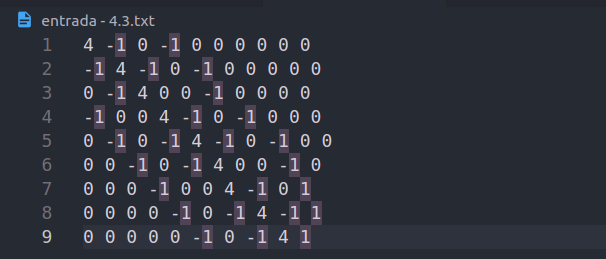


Imagem 5.3 - Arquivo de Entrada do Problema 4.3

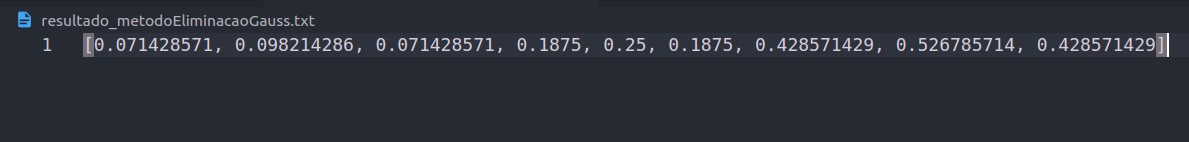


Imagem 5.4 - Arquivo de Saída do Problema 4.3

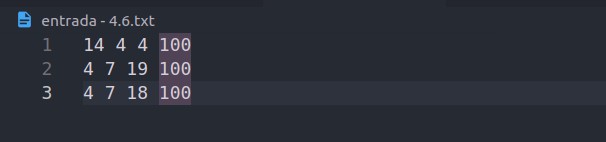


Imagem 5.5 - Arquivo de Entrada do Problema 4.6

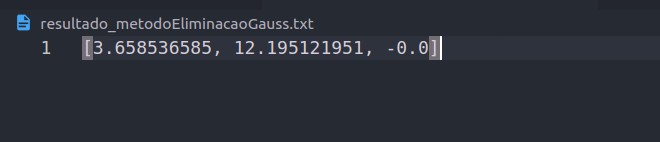


Imagem 5.6 - Arquivo de Saída do Problema 4.6

## Dificuldades enfrentadas

Não houveram dificuldades.

# Método da Fatoração LU

## Estratégia de Implementação:

Este código implementa o método da fatoração LU para resolver sistemas lineares. Primeiramente, a função fatoracao\_LU recebe como entrada uma matriz de coeficientes A e um vetor de termos independentes b. Em seguida, a função realiza a fatoração LU da matriz A utilizando a função scipy.linalg.lu\_factor. Essa função retorna a matriz LU e o vetor piv de permutações que são utilizados para resolver os sistemas triangulares separados na função scipy.linalg.lu\_solve. Por fim, a função retorna o vetor solução x do sistema linear.

## Estrutura dos Arquivos de Entrada/Saída

O programa lê os dados de entrada de um arquivo chamado "entrada.txt”. A matriz é definida pelas primeiras n-1 linhas e n-1 colunas do arquivo, em que cada linha representa uma equação e cada coluna representa um coeficiente. O vetor de resultados é definido pela última coluna do arquivo, em que cada elemento corresponde ao resultado da equação correspondente na matriz.

Por exemplo, um arquivo com as informações da seguinte forma pode ser utilizado:

8 -4 -2 5

-4 6 -2 0

-2 -2 10 0

O arquivo de saída , o programa escreve a solução encontrada utilizando o método em questão, o nome de arquivo de saída possui a seguinte configuração: "resultado\_<Nome do Metodo>.txt”.

## Problema teste 1, 2, 3...

Foi feito testes com utilizando utilizando os conceitos e técnicas apresentados no livro Cálculo Numérico, de Neide Franco. A implementação foi testada em 3 problemas apresentados no livro, e os resultados obtidos foram coerentes com outros métodos conhecidos, mostrando que a implementação funciona de forma eficiente.

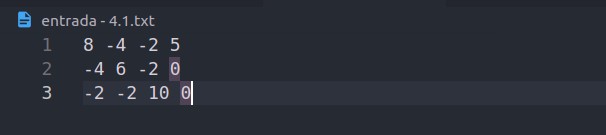


Imagem 6.1 - Arquivo de Entrada do Problema 4.1

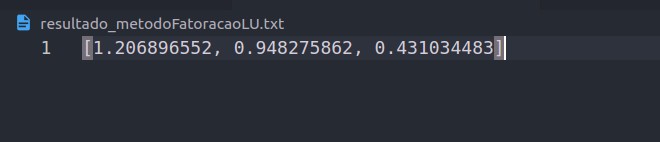


Imagem 6.2 - Arquivo de Saída do Problema 4.1

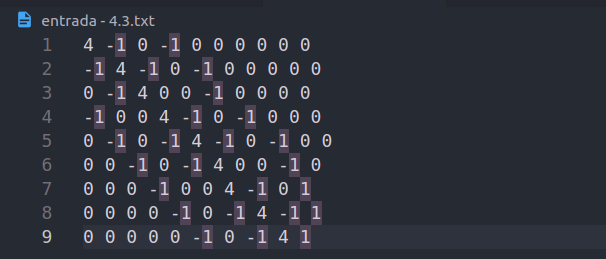


Imagem 6.3 - Arquivo de Entrada do Problema 4.3

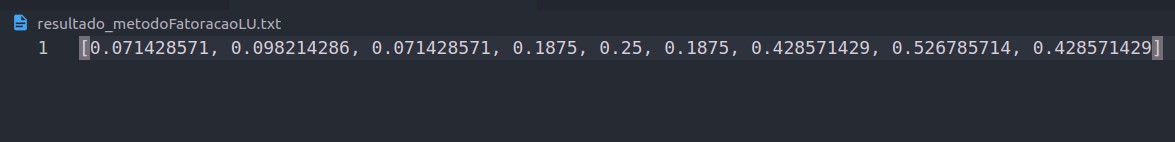


Imagem 6.4 - Arquivo de Saída do Problema 4.3

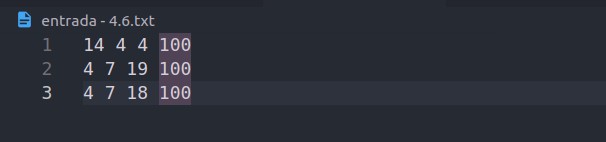


Imagem 6.5 - Arquivo de Entrada do Problema 4.6

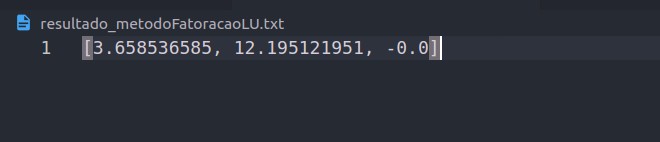


Imagem 6.6 - Arquivo de Saída do Problema 4.6

## Dificuldades enfrentadas

Não houveram dificuldades.

# Método de Jacobi

## Estratégia de Implementação:

O código implementa o método iterativo de Jacobi para resolver sistemas lineares. Na função jacobi, os parâmetros de entrada são a matriz de coeficientes A, o vetor de termos independentes b, um vetor de aproximação inicial x0, a tolerância tol e o número máximo de iterações max\_iter. A função retorna o vetor solução x.

Dentro da função jacobi, é criada uma cópia do vetor x0, chamada x, que será atualizada a cada iteração. É então executado um loop for que itera até o número máximo de iterações max\_iter. Dentro do loop, é criada uma cópia de x chamada x\_old, que é utilizada para verificar se a diferença entre x e x\_old é menor que a tolerância tol para o critério de parada.

## Estrutura dos Arquivos de Entrada/Saída

O programa lê os dados de entrada de um arquivo chamado "entrada.txt”. A matriz é definida pelas primeiras n-1 linhas e n-1 colunas do arquivo, em que cada linha representa uma equação e cada coluna representa um coeficiente. O vetor de resultados é definido pela última coluna do arquivo, em que cada elemento corresponde ao resultado da equação correspondente na matriz.

Por exemplo, um arquivo com as informações da seguinte forma pode ser utilizado:

8 -4 -2 5

-4 6 -2 0

-2 -2 10 0

O arquivo de saída , o programa escreve a solução encontrada utilizando o método em questão, o nome de arquivo de saída possui a seguinte configuração: "resultado\_<Nome do Metodo>.txt”.

## Problema teste 1, 2, 3...

Foi feito testes com utilizando utilizando os conceitos e técnicas apresentados no livro Cálculo Numérico, de Neide Franco. A implementação foi testada em 3 problemas apresentados no livro, e os resultados obtidos foram coerentes com outros métodos conhecidos, mostrando que a implementação funciona de forma eficiente.

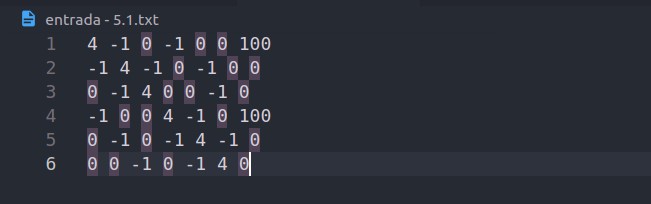


Imagem 7.1 - Arquivo de Entrada do Problema 5.1

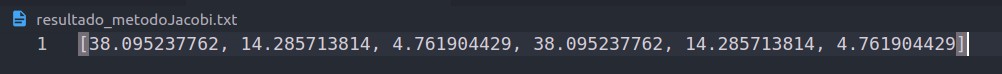


Imagem 7.2 - Arquivo de Saída do Problema 5.1

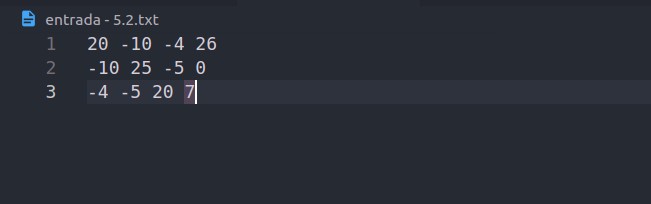


Imagem 7.3 - Arquivo de Entrada do Problema 5.2

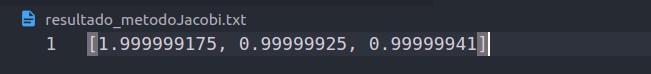


Imagem 7.4 - Arquivo de Saída do Problema 5.2

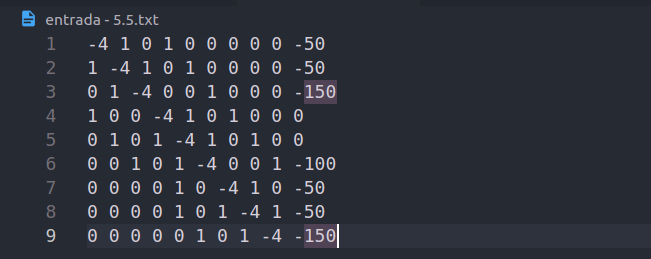


Imagem 7.5 - Arquivo de Entrada do Problema 5.5

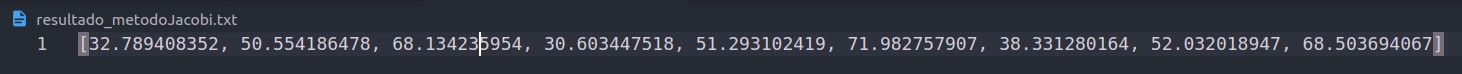


Imagem 7.6 - Arquivo de Saída do Problema 5.5

## Dificuldades enfrentadas

Não houveram dificuldades.

# Método de Gauss-Seidel

## Estratégia de Implementação:

A estratégia de implementação do código é importar a biblioteca NumPy, definir a função gauss\_seidel para resolver sistemas lineares usando o método iterativo de Gauss-Seidel, calcular o tamanho do vetor de termos independentes e executar um loop que itera até um número máximo de iterações, utilizando os elementos do vetor solução em cada iteração.

## Estrutura dos Arquivos de Entrada/Saída

O programa lê os dados de entrada de um arquivo chamado "entrada.txt”. A matriz é definida pelas primeiras n-1 linhas e n-1 colunas do arquivo, em que cada linha representa uma equação e cada coluna representa um coeficiente. O vetor de resultados é definido pela última coluna do arquivo, em que cada elemento corresponde ao resultado da equação correspondente na matriz.

Por exemplo, um arquivo com as informações da seguinte forma pode ser utilizado:

8 -4 -2 5

-4 6 -2 0

-2 -2 10 0

O arquivo de saída , o programa escreve a solução encontrada utilizando o método em questão, o nome de arquivo de saída possui a seguinte configuração: "resultado\_<Nome do Metodo>.txt”.

## Problema teste 1, 2, 3...

Foi feito testes com utilizando utilizando os conceitos e técnicas apresentados no livro Cálculo Numérico, de Neide Franco. A implementação foi testada em 3 problemas apresentados no livro, e os resultados obtidos foram coerentes com outros métodos conhecidos, mostrando que a implementação funciona de forma eficiente.

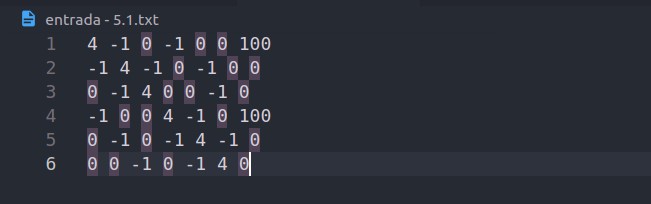


Imagem 8.1 - Arquivo de Entrada do Problema 5.1

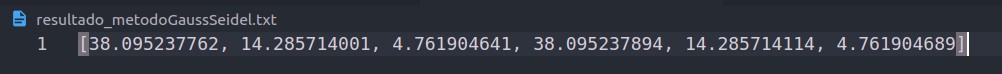


Imagem 8.2 - Arquivo de Saída do Problema 5.1

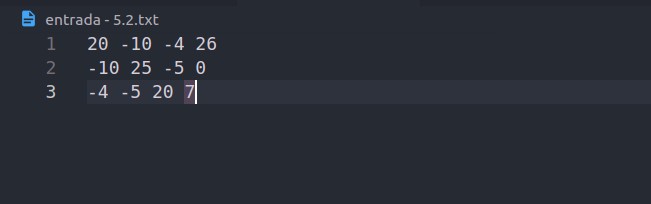


Imagem 8.3 - Arquivo de Entrada do Problema 5.2

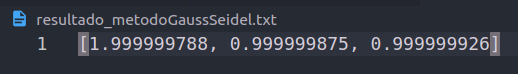


Imagem 8.4 - Arquivo de Saída do Problema 5.2

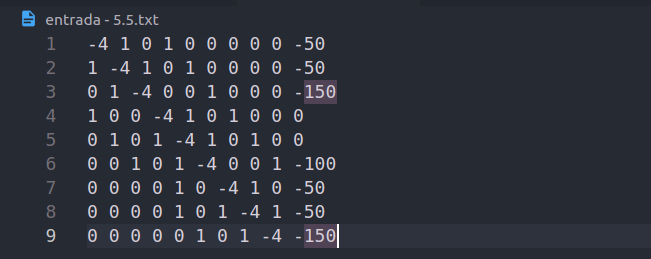


Imagem 8.5 - Arquivo de Entrada do Problema 5.5

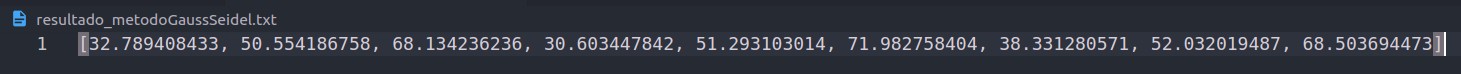


Imagem 8.6 - Arquivo de Saída do Problema 5.5

## Dificuldades enfrentadas

Não houveram dificuldades.

# Método da Inversão

## Estratégia de Implementação:

A estratégia de implementação do código é importar a biblioteca NumPy, definir a função inversao para resolver sistemas lineares usando o método da inversão, calcular a inversa da matriz de coeficientes usando a função np.linalg.inv e obter o vetor solução multiplicando a inversa pela matriz de termos independentes.

## Estrutura dos Arquivos de Entrada/Saída

O programa lê os dados de entrada de um arquivo chamado "entrada.txt”. A matriz é definida pelas primeiras n-1 linhas e n-1 colunas do arquivo, em que cada linha representa uma equação e cada coluna representa um coeficiente. O vetor de resultados é definido pela última coluna do arquivo, em que cada elemento corresponde ao resultado da equação correspondente na matriz.

Por exemplo, um arquivo com as informações da seguinte forma pode ser utilizado:

8 -4 -2 5

-4 6 -2 0

-2 -2 10 0

O arquivo de saída , o programa escreve a solução encontrada utilizando o método em questão, o nome de arquivo de saída possui a seguinte configuração: "resultado\_<Nome do Metodo>.txt”.

## Problema teste 1, 2, 3…

Foi feito testes com utilizando utilizando os conceitos e técnicas apresentados no livro Cálculo Numérico, de Neide Franco. A implementação foi testada em 3 problemas apresentados no livro, e os resultados obtidos foram coerentes com outros métodos conhecidos, mostrando que a implementação funciona de forma eficiente.

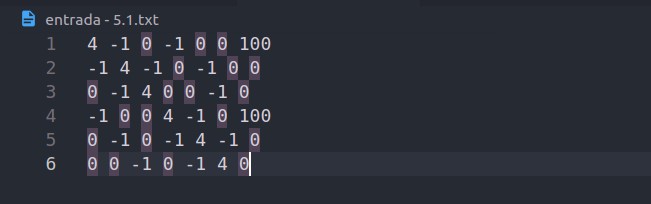


Imagem 9.1 - Arquivo de Entrada do Problema 5.1

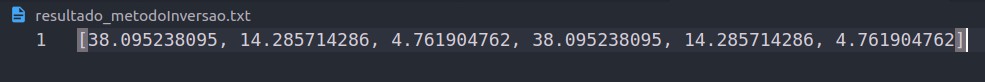


Imagem 9.2 - Arquivo de Saída do Problema 5.1

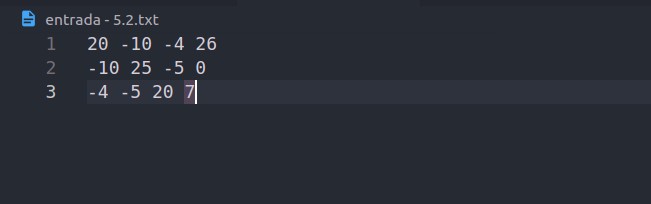


Imagem 9.3 - Arquivo de Entrada do Problema 5.2

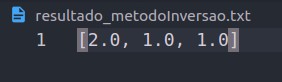


Imagem 9.4 - Arquivo de Saída do Problema 5.2

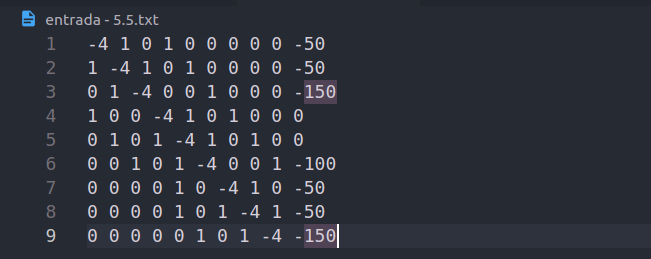


Imagem 9.5 - Arquivo de Entrada do Problema 5.5

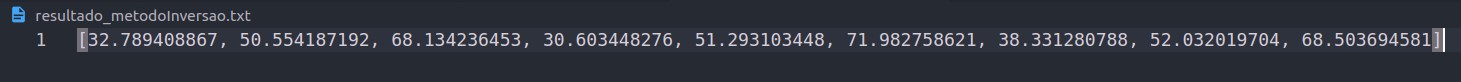


Imagem 9.6 - Arquivo de Saída do Problema 5.5

## Dificuldades enfrentadas

Não houveram dificuldades.

# Método do Número de Condição da Matriz

## Estratégia de Implementação:

A estratégia de implementação do código é importar a biblioteca NumPy, definir a função condicao para calcular o número de condição de uma matriz de coeficientes de um sistema linear, utilizando a função np.linalg.cond.

## Estrutura dos Arquivos de Entrada/Saída

O programa lê os dados de entrada de um arquivo chamado "entrada.txt”. A matriz é definida pelas primeiras n-1 linhas e n-1 colunas do arquivo, em que cada linha representa uma equação e cada coluna representa um coeficiente. O vetor de resultados é definido pela última coluna do arquivo, em que cada elemento corresponde ao resultado da equação correspondente na matriz.

Por exemplo, um arquivo com as informações da seguinte forma pode ser utilizado:

8 -4 -2 5

-4 6 -2 0

-2 -2 10 0

O arquivo de saída , o programa escreve a solução encontrada utilizando o método em questão, o nome de arquivo de saída possui a seguinte configuração: "resultado\_<Nome do Metodo>.txt”.

## Problema teste 1, 2, 3…

Foi feito testes com utilizando utilizando os conceitos e técnicas apresentados no livro Cálculo Numérico, de Neide Franco. A implementação foi testada em 3 problemas apresentados no livro, e os resultados obtidos foram coerentes com outros métodos conhecidos, mostrando que a implementação funciona de forma eficiente.

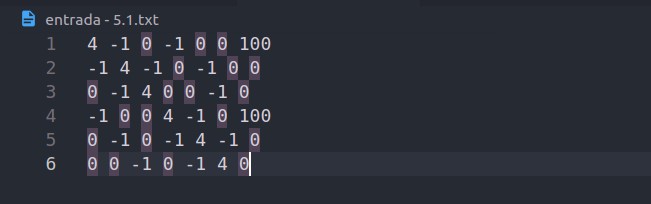


Imagem 10.1 - Arquivo de Entrada do Problema 5.1

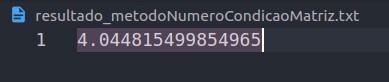


Imagem 10.2 - Arquivo de Saída do Problema 5.1

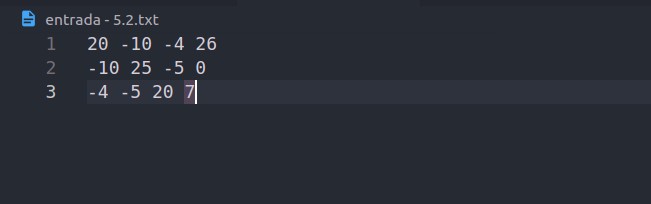


Imagem 10.3 - Arquivo de Entrada do Problema 5.2

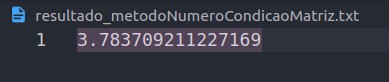


Imagem 10.4 - Arquivo de Saída do Problema 5.2

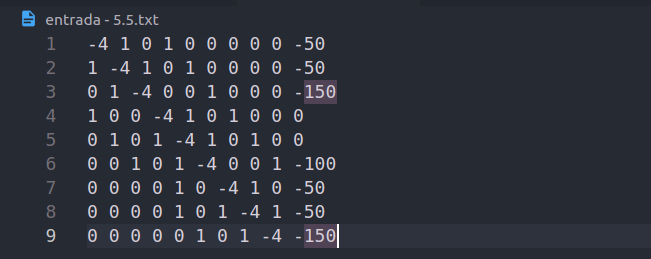


Imagem 10.5 - Arquivo de Entrada do Problema 5.5

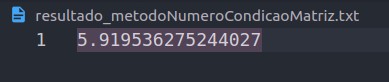


Imagem 10.6 - Arquivo de Saída do Problema 5.5

## Dificuldades enfrentadas

Não houveram dificuldades.

# Considerações Finais

Podemos concluir que a implementação dos métodos matemáticos estudados foi fundamental para compreender a resolução de problemas numéricos. Cada um dos métodos apresentados possui características específicas que os tornam adequados para diferentes tipos de problemas.

A bissecção e a posição falsa são métodos de busca de raízes que permitem encontrar soluções aproximadas para equações não-lineares. Já o método de Newton-Raphson e a secante são métodos mais sofisticados que fornecem soluções mais precisas em menos iterações.

Os métodos de eliminação de Gauss e fatoração LU são técnicas para a solução de sistemas de equações lineares. A eliminação de Gauss permite a resolução de sistemas lineares de forma direta, enquanto a fatoração LU é uma técnica mais geral que permite a resolução de sistemas lineares com matrizes não-simétricas.

Os métodos iterativos de Jacobi e Gauss-Seidel são técnicas para a solução de sistemas de equações lineares que requerem menos cálculos e são mais adequados para sistemas grandes. A inversão de matrizes é um método mais direto para a resolução de sistemas lineares, mas que pode ser muito custoso computacionalmente.

Finalmente, o número de condição da matriz é um importante conceito da análise numérica que nos ajuda a avaliar a estabilidade e a precisão de soluções obtidas pelos diferentes métodos. É fundamental entender esse conceito para garantir a confiabilidade dos resultados obtidos.