UESC

Universidade Estadual de Santa Cruz – UESC

Relatórios de Implementações de Métodos da Disciplina Análise Numérica

Relatório de implementações realizadas por Gabriel Rosa Galdino

Disciplina Análise Numérica.

Curso Ciência da Computação

Semestre 2025.2

Professor Gesil Sampaio Amarante II

Ilhéus – BA
2025

ÍNDICE

Lista de Figuras

Linguagem Escolhida e justificativas

A linguagem Python foi a selecionada para este trabalho devido à sua simplicidade e ao seu poderoso ecossistema para computação científica, o que permitiu focar mais na lógica dos métodos do que em complexidades de programação.

A escolha é justificada principalmente pelo uso de bibliotecas essenciais como o NumPy, que oferece uma base sólida para operações matemáticas de alta performance. Para a visualização e análise dos dados, foi fundamental o uso do Matplotlib. Além disso, a biblioteca SciPy foi de grande ajuda, pois já contém implementações de diversos algoritmos numéricos, o que facilitou a verificação e a aplicação prática dos métodos abordados na disciplina.

Dessa forma, o Python se mostrou a escolha ideal para os objetivos deste relatório, aliando um código legível a um ambiente computacional robusto, o que tornou a implementação dos métodos uma tarefa prática e didática.

Método da Bisseção

Estratégia de Implementação:

Para resolver numericamente equações do tipo f(x)=0, implementei o método da bissecção em Python. A estratégia adotada no meu código começa com uma validação essencial: a verificação de que os valores da função nos extremos do intervalo [a,b] possuem sinais opostos, através da condição **if self.f(self.a) * self.f(self.b) >= 0:.** Se esta premissa fundamental do método não for atendida, o programa exibe um aviso e interrompe a execução, garantindo que o algoritmo não prossiga em um cenário onde a convergência não é garantida.

A função matemática é lida como uma *string* de um arquivo de entrada, e para interpretá-la de forma flexível, utilizei a biblioteca SymPy. Através da função **lambdify**, a expressão simbólica é convertida em uma função numérica de alta performance, que pode ser avaliada eficientemente dentro do laço de iterações.

O processo iterativo consiste em calcular o ponto médio c=(a+b)/2 e, em seguida, reduzir o intervalo de busca pela metade, atualizando **a** ou **b** com base no sinal do produto **f(a) * f(c).** O critério de parada que implementei é duplo, para assegurar tanto precisão quanto eficiência: o laço é interrompido se o erro, medido pela metade da amplitude do intervalo, for menor que a tolerância definida **(error / 2 < self.tolerance)**, ou se o valor da função no ponto c já for residual (abs(f_c) < 1e-15). Adicionalmente, incluí um limite de 1000 iterações como uma salvaguarda para evitar loops infinitos.

Estrutura dos Arquivos de Entrada/Saída

A estrutura dos arquivos de entrada e saída, foi projetada para ser intuitiva, legível e de fácil automação para testes.

- Arquivo de Entrada (ex: **entrada.txt**)
  O programa espera um arquivo de texto localizado em um diretório chamado input/. Este arquivo deve conter quatro linhas, que são lidas e processadas pelo construtor da classe CalculadorDeRaizes:
  - **Expressão da função**: A função a ser analisada, em formato de *string*.
  - **Extremo inferior do intervalo (a)**: O valor inicial do intervalo.
  - **Extremo superior do intervalo (b)**: O valor final do intervalo.
  - **Tolerância (opcional)**: A precisão desejada. Implementei a lógica para que, se esta linha estiver vazia, uma tolerância padrão de 1e-8 seja automaticamente adotada.

- Arquivo de Saída (**bissecao_saida.txt**)
  Após a execução, um relatório detalhado é gerado no diretório **output/**. A saída é formatada para oferecer uma análise clara do processo de convergência:
  - Um cabeçalho identifica o método e a função analisada.
  - Uma tabela exibe o progresso de cada iteração com as colunas: **Iter, a, b, c (raiz), f(c) e Erro |b-a|**.
  - Ao final do arquivo, um bloco de resumo apresenta de forma concisa a raiz encontrada, o erro final estimado, o número total de iterações e o tempo de execução, permitindo uma rápida verificação dos resultados.

Dificuldades enfrentadas

A principal dificuldade na implementação do método da bissecção foi garantir sua robustez. O desafio inicial foi como lidar com um intervalo de entrada inválido. Para solucionar isso, implementei a verificação explícita de $f(a) \cdot f(b) < 0$ no início da função. Dessa forma, em vez de falhar ou entrar em um loop incorreto, o programa informa o usuário sobre a inadequação do intervalo, o que torna a ferramenta mais confiável.

Outro ponto de atenção foi o critério de parada. Confiar apenas na tolerância do intervalo poderia ser insuficiente se a função se aproximasse de zero muito rapidamente. Por isso, adotei um critério duplo, que verifica tanto a amplitude do intervalo quanto o valor de $|f(c)|$, garantindo que o método pare assim que uma das condições de sucesso for atingida.

Finalmente, para evitar que o programa ficasse preso em um loop caso a convergência fosse muito lenta ou impossível, estabeleci um limite máximo de iterações. Essa medida simples funciona como um mecanismo de segurança essencial, garantindo que o programa sempre termine, mesmo que a solução não seja encontrada dentro do limite estipulado.

Método da Posição Falsa

Estratégia de Implementação:

Para a implementação do método da Posição Falsa, segui uma abordagem que combina a segurança de um método de intervalo com uma convergência potencialmente mais rápida que a da bissecção. Assim como no método anterior, a primeira etapa é a validação do intervalo inicial [a,b], garantindo que a condição **self.f(self.a) * self.f(self.b) < 0** seja satisfeita. Se os valores da função nos extremos não tiverem sinais opostos, o programa emite um aviso e não prossegue.

O diferencial deste método, e o ponto central da minha implementação, está no cálculo da nova aproximação da raiz. Em vez de simplesmente dividir o intervalo ao meio, utilizei a fórmula da interpolação linear: **c = (a * f_b - b * f_a) / (f_b - f_a)**. Essa fórmula determina o ponto c onde a reta que conecta (a,f(a)) e (b,f(b)) cruza o eixo x, o que geralmente aproxima a raiz de forma mais eficiente. Para o critério de parada, decidi utilizar uma medida de erro baseada na diferença entre a aproximação atual e a anterior **(error = abs(c - c_old))**. Essa abordagem monitora o quanto a solução está mudando a cada passo. O processo iterativo termina quando este erro se torna menor que a tolerância especificada (error < self.tolerance) ou quando o valor da função em c é suficientemente próximo de zero **(abs(f_c) < 1e-15).** A variável **c_old** é inicializada com um valor infinito para garantir que o cálculo do erro na primeira iteração seja válido. Após o cálculo de c, o intervalo é atualizado de forma análoga à bissecção, mantendo a raiz sempre isolada.

Estrutura dos Arquivos de Entrada/Saída

A estrutura dos arquivos foi mantida consistente com a implementação anterior para garantir a modularidade e a facilidade de uso do programa.

- **Arquivo de Entrada (ex: entrada.txt)** O programa lê os dados de um arquivo de texto localizado no diretório **input/**, que deve seguir o mesmo formato de quatro linhas:
    - **Expressão da função**: A função em formato de *string*.
    - **Extremo inferior do intervalo (a)**.
    - **Extremo superior do intervalo (b)**.
    - **Tolerância (opcional)**: Com um valor padrão de 1e-8 caso a linha esteja vazia.

- **Arquivo de Saída (posicao_falsa_saida.txt)** O relatório de execução é salvo em um arquivo específico no diretório **output/**. A estrutura da saída foi projetada para ser clara e informativa:
    - Um cabeçalho que identifica o método e a função.
    - Uma tabela detalhada do processo iterativo, com as colunas: **Iter, a, b, c (raiz), f(c)** e **Erro |c-c_old|**. A coluna de erro reflete a métrica de convergência que implementei.
    - Um bloco de resumo ao final do arquivo, com a raiz final, o erro, o número de iterações e o tempo de execução.

Dificuldades enfrentadas

A principal dificuldade na implementação do método da Posição Falsa foi a definição e o controle do critério de convergência. Ao contrário da bissecção, onde o erro pode ser medido pela amplitude do intervalo, aqui optei por medir a variação entre aproximações sucessivas ($|C - C_{antigo}|$). Isso exigiu a criação de uma variável adicional, **c_old**, para armazenar o valor da iteração anterior e o cuidado de inicializá-la de forma que não afetasse a primeira iteração.

Método de Newton-Raphson

Estratégia de Implementação:

Para a implementação do método de Newton-Raphson, adotei uma estratégia que aproveita a rápida convergência do método, ao mesmo tempo em que automatiza uma de suas etapas mais complexas: o cálculo da derivada. A essência do algoritmo, presente no meu código, está na fórmula iterativa $X_{i+1} = X_i - \frac{f(x_i)}{f'(x_i)}$, que utiliza a reta tangente à função em um ponto para encontrar a próxima aproximação da raiz.

A característica mais significativa desta implementação é o cálculo automático da derivada. Utilizando a biblioteca **SymPy**, o programa lê a função como uma *string*, calcula sua derivada simbólica com a função **diff**, e então converte tanto a função original quanto sua derivada em funções numericamente executáveis através de **lambdify**. Isso elimina a necessidade de o usuário fornecer a derivada manualmente, tornando a ferramenta mais prática e menos suscetível a erros.

O chute inicial, **x**, é determinado de forma pragmática, utilizando o ponto médio do intervalo **[a, b]** fornecido no arquivo de entrada. O critério de parada é duplo, interrompendo as iterações quando a diferença absoluta entre as aproximações sucessivas **(error = abs(x_new - x))** for menor que a tolerância definida, ou quando o valor da função no ponto atual, |f(xi)|, for praticamente nulo.

Para garantir a robustez do método, implementei uma verificação crucial para evitar a divisão por zero ou valores muito pequenos, que é um ponto de falha comum. A condição **if abs(f_prime_x) < 1e-12:** interrompe o método e informa o usuário caso a derivada se aproxime de zero, prevenindo instabilidade numérica.

Estrutura dos Arquivos de Entrada/Saída

A estrutura de arquivos foi mantida consistente com os outros métodos, facilitando a interoperabilidade e a realização de testes comparativos.

- **Arquivo de Entrada (ex: entrada.txt)** O programa utiliza o mesmo formato de arquivo de entrada, localizado no diretório **input/**:
    - **Expressão da função**: A função em formato de *string*.
    - **Extremo inferior do intervalo (a)**: Usado para calcular o chute inicial.

    - **Extremo superior do intervalo (b)**: Usado para calcular o chute inicial.
    - **Tolerância (opcional)**: Com o valor padrão 1e-8 se não for especificado.
- **Arquivo de Saída (newton_raphson_saida.txt)** O relatório gerado no diretório **output/** foi customizado para refletir as particularidades do método:
    - O cabeçalho informa o método, a função e o chute inicial utilizado.
    - A tabela de iterações exibe informações detalhadas, incluindo as colunas: **Iter, x_i, f(x_i), f'(x_i)** (o valor da derivada) e **Erro |x_i-x_i-1|**. A inclusão do valor da derivada é útil para analisar o comportamento da convergência.
    - O arquivo é finalizado com o mesmo bloco de resumo, apresentando a raiz encontrada, o erro final, o número de iterações e o tempo de execução.

Dificuldades enfrentadas

A principal dificuldade teórica na aplicação do método de Newton-Raphson é a necessidade de se obter a derivada da função. Para contornar esse desafio e tornar a implementação mais eficiente, optei por automatizar esse processo. A integração com a biblioteca **SymPy** para realizar a diferenciação simbólica foi a solução encontrada, o que exigiu um tratamento cuidadoso das expressões matemáticas lidas do arquivo.

O segundo desafio significativo foi lidar com a instabilidade numérica que ocorre quando a derivada da função se aproxima de zero, o que corresponde a um ponto de inclinação horizontal na reta tangente. Para garantir que o programa não falhasse com um erro de divisão por zero, implementei uma verificação explícita que interrompe a execução de forma controlada caso **|f'(x)|** seja menor que um limiar (1e-12), informando ao usuário a causa da falha.

Método da secante

Estratégia de Implementação:

Desenvolvi o método da Secante como uma alternativa eficiente ao método de Newton-Raphson, com a vantagem de não requerer o cálculo explícito da derivada da função. A estratégia se baseia em aproximar a derivada utilizando a inclinação da reta que passa pelos dois últimos pontos calculados. A fórmula iterativa que implementei é

$$x_{i+1} = x_i - f(x_i)\,\frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})}.$$

Figura 1 - Fórmula Interativa.

Para dar início ao processo, o algoritmo utiliza os valores a e b do arquivo de entrada como os dois pontos iniciais, x0 e x1. Em cada iteração, esses dois pontos são usados para calcular uma nova aproximação, x2. O erro é então medido pela diferença absoluta entre as duas últimas aproximações, **error = abs(x2 - x1).**

O critério de parada é similar ao dos outros métodos, combinando a verificação da tolerância com a análise do resíduo da função. O laço termina se **error < self.tolerance** ou se o valor absoluto da função na nova aproximação, **abs(self.f(x2))**, for suficientemente próximo de zero.

Uma preocupação central na implementação foi a estabilidade numérica. O denominador da fórmula, f(x1) - f(x0), pode se aproximar de zero se os valores da função nos dois pontos forem muito similares. Para evitar uma divisão instável, adicionei a condição if abs(f_x1 - f_x0) < 1e-12:, que interrompe o método de forma segura caso isso ocorra. Ao final de cada passo, os pontos são atualizados (x0, x1 = x1, x2) para preparar a próxima iteração.

Estrutura dos Arquivos de Entrada/Saída

A estrutura de arquivos foi mantida consistente para garantir uma experiência de uso uniforme em todo o código.

- **Arquivo de Entrada (ex: entrada.txt)** O método utiliza o mesmo formato de arquivo de entrada, localizado no diretório **input/**:
  - **Expressão da função**: A função em formato de *string*.
  - **Ponto inicial (a)**: Utilizado como o primeiro ponto, x0.

    - **Ponto inicial (b)**: Utilizado como o segundo ponto, x1.
    - **Tolerância (opcional)**: Com o valor padrão 1e-8 se não for especificado.
- **Arquivo de Saída (secante_saida.txt)** O relatório gerado no diretório **output/** foi adaptado para refletir as informações relevantes para o método da Secante:
    - O cabeçalho informa o método, a função e os pontos iniciais x0 e x1.
    - A tabela de progresso contém as colunas: Iter, x_i+1 (a nova aproximação), f(x_i+1) e Erro |x_i+1 - x_i|.
    - O arquivo é concluído com o bloco de resumo padrão, que informa a raiz encontrada, o erro final, o total de iterações e o tempo de execução.

Dificuldades enfrentadas

A principal dificuldade na implementação do método da Secante foi garantir a estabilidade numérica, especificamente em relação à divisão na fórmula iterativa. Se os valores da função nos dois pontos de referência, f(x1) e f(x0), forem quase idênticos, o denominador se aproxima de zero, o que pode levar a um resultado incorreto ou a um erro de execução. Para mitigar esse risco, implementei uma verificação explícita que interrompe o algoritmo e notifica o usuário se o denominador for perigosamente pequeno, garantindo que o programa não falhe de forma inesperada.

Problemas Estabelecidos

Problema 1 (Exercício 3.3)

Um amplificador eletrônico com acoplamento R - C com três estágios em cascata tem uma resposta a um degrau unitário de tensão dada pela expressão: $g(T) = 1 – (1 + T + T^2/2)e^{-T}$; onde $T = t/RC$ é uma unidade de tempo normalizada. O tempo de subida de um amplificador é definido como o tempo necessário para sua resposta ir de 10% a 90% de seu valor final. No caso, como $g(\infty) = 1$ é necessário calcular os valores de T para os quais $g(T) = 0.1$ e $g(T) = 0.9$, ou seja, resolver as equações:

- $0.1 = 1 – (1 + T + T^2/2)e^{-T}$
- $0.9 = 1 – (1 + T + T^2/2)e^{-T}$

Entrada 1:

```
1 - (1 + x + x**2 / 2) * e**(-x) - 0.1
0.1
2
1e-6
```

Figura 2 – Entrada para o Problema 1 (Exercício 3.3).

Entrada 2:

```
1 - (1 + x + x**2 / 2) * e**(-x) - 0.9
3
6
1e-6
```

Figura 3 – Entrada para o Problema 2 (Exercício 3.3).

Bisseção

Saída 1:

```
--- Relatorio do Metodo da Bissecao ---
Funcao: f(x) = 1 - (1 + x + x**2 / 2) * e**(-x) - 0.1

Iter |          a |          b |   c (raiz) |          f(c) |     Erro |b-a|
---------------------------------------------------------------------------
   1 |   0.100000 |   2.000000 |   1.050000 |  -1.027557e-02 |  1.900000e+00
   2 |   1.050000 |   2.000000 |   1.525000 |   9.745435e-02 |  9.500000e-01
   3 |   1.050000 |   1.525000 |   1.287500 |   4.001865e-02 |  4.750000e-01
   4 |   1.050000 |   1.287500 |   1.168750 |   1.380766e-02 |  2.375000e-01
   5 |   1.050000 |   1.168750 |   1.109375 |   1.478935e-03 |  1.187500e-01
   6 |   1.050000 |   1.109375 |   1.079687 |  -4.472687e-03 |  5.937500e-02
   7 |   1.079687 |   1.109375 |   1.094531 |  -1.515147e-03 |  2.968750e-02
   8 |   1.094531 |   1.109375 |   1.101953 |  -2.263338e-05 |  1.484375e-02
   9 |   1.101953 |   1.109375 |   1.105664 |   7.270242e-04 |  7.421875e-03
  10 |   1.101953 |   1.105664 |   1.103809 |   3.519131e-04 |  3.710937e-03
  11 |   1.101953 |   1.103809 |   1.102881 |   1.645692e-04 |  1.855469e-03
  12 |   1.101953 |   1.102881 |   1.102417 |   7.095023e-05 |  9.277344e-04
  13 |   1.101953 |   1.102417 |   1.102185 |   2.415401e-05 |  4.638672e-04
  14 |   1.101953 |   1.102185 |   1.102069 |   7.592099e-07 |  2.319336e-04
  15 |   1.101953 |   1.102069 |   1.102011 |  -1.093736e-05 |  1.159668e-04
  16 |   1.102011 |   1.102069 |   1.102040 |  -5.089144e-06 |  5.798340e-05
  17 |   1.102040 |   1.102069 |   1.102055 |  -2.164984e-06 |  2.899170e-05
  18 |   1.102055 |   1.102069 |   1.102062 |  -7.028916e-07 |  1.449585e-05
  19 |   1.102062 |   1.102069 |   1.102065 |   2.815808e-08 |  7.247925e-06
  20 |   1.102062 |   1.102065 |   1.102064 |  -3.373670e-07 |  3.623962e-06
  21 |   1.102064 |   1.102065 |   1.102065 |  -1.546045e-07 |  1.811981e-06


---------------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:     1.10206456
Erro final estimado: 9.06e-07
Numero de iteracoes: 21
Tempo de execucao:   0.007855 segundos
```

Figura 4 - Saída do Método da Bissecção para o Problema 1, Equação 1.

Saída 2:

```
--- Relatorio do Metodo da Bissecao ---
Funcao: f(x) = 1 - (1 + x + x**2 / 2) * e**(-x) - 0.9

Iter |          a |          b |   c (raiz) |          f(c) |     Erro |b-a|
---------------------------------------------------------------------------
   1 |   3.000000 |   6.000000 |   4.500000 |  -7.357807e-02 |  3.000000e+00
   2 |   4.500000 |   6.000000 |   5.250000 |  -5.114353e-03 |  1.500000e+00
   3 |   5.250000 |   6.000000 |   5.625000 |   1.904956e-02 |  7.500000e-01
   4 |   5.250000 |   5.625000 |   5.437500 |   7.682609e-03 |  3.750000e-01
   5 |   5.250000 |   5.437500 |   5.343750 |   1.471733e-03 |  1.875000e-01
   6 |   5.250000 |   5.343750 |   5.296875 |  -1.773275e-03 |  9.375000e-02
   7 |   5.296875 |   5.343750 |   5.320312 |  -1.389052e-04 |  4.687500e-02
   8 |   5.320312 |   5.343750 |   5.332031 |   6.693628e-04 |  2.343750e-02
   9 |   5.320312 |   5.332031 |   5.326172 |   2.659682e-04 |  1.171875e-02
  10 |   5.320312 |   5.326172 |   5.323242 |   6.371663e-05 |  5.859375e-03
  11 |   5.320312 |   5.323242 |   5.321777 |  -3.754796e-05 |  2.929688e-03
  12 |   5.321777 |   5.323242 |   5.322510 |   1.309591e-05 |  1.464844e-03


---------------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:     5.32250977
Erro final estimado: 7.32e-04
Numero de iteracoes: 12
Tempo de execucao:   0.000062 segundos
```

Figura 5 - Saída do Método da Bissecção para o Problema 1, Equação 2.

Posição Falsa

Saída 1:

```
--- Relatorio do Metodo da Posicao Falsa ---
Funcao: f(x) = 1 - (1 + x + x**2 / 2) * e**(-x) - 0.1

Iter |          a |          b |   c (raiz) |         f(c) |     Erro |c-c_old|
------------------------------------------------------------------------------
   1 |   0.100000 |   2.000000 |   0.687019 | -6.741872e-02 |           inf
   2 |   0.687019 |   2.000000 |   0.991479 | -2.125926e-02 |  3.044604e-01
   3 |   0.991479 |   2.000000 |   1.079140 | -4.581043e-03 |  8.766114e-02
   4 |   1.079140 |   2.000000 |   1.097650 | -8.890757e-04 |  1.850993e-02
   5 |   1.097650 |   2.000000 |   1.101228 | -1.688222e-04 |  3.578110e-03
   6 |   1.101228 |   2.000000 |   1.101907 | -3.192207e-05 |  6.789161e-04
   7 |   1.101907 |   2.000000 |   1.102035 | -6.031230e-06 |  1.283559e-04
   8 |   1.102035 |   2.000000 |   1.102060 | -1.139345e-06 |  2.425039e-05
   9 |   1.102060 |   2.000000 |   1.102064 | -2.152246e-07 |  4.581057e-06
  10 |   1.102064 |   2.000000 |   1.102065 | -4.065616e-08 |  8.653706e-07

---------------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:     1.10206513
Erro final estimado: 8.65e-07
Numero de iteracoes: 10
Tempo de execucao:   0.003028 segundos
```

Figura 6 - Saída do Método da Posição Falsa para o Problema 1, Equação 1.

Saída 2:

```
--- Relatorio do Metodo da Posicao Falsa ---
Funcao: f(x) = 1 - (1 + x + x**2 / 2) * e**(-x) - 0.9

Iter |          a |          b |   c (raiz) |         f(c) |     Erro |c-c_old|
------------------------------------------------------------------------------
   1 |   3.000000 |   6.000000 |   5.684145 |  2.236057e-02 |           inf
   2 |   3.000000 |   5.684145 |   5.510454 |  1.226761e-02 |  1.736909e-01
   3 |   3.000000 |   5.510454 |   5.418647 |  6.462903e-03 |  9.180671e-02
   4 |   3.000000 |   5.418647 |   5.371229 |  3.330284e-03 |  4.741799e-02
   5 |   3.000000 |   5.371229 |   5.347044 |  1.696239e-03 |  2.418492e-02
   6 |   3.000000 |   5.347044 |   5.334790 |  8.588088e-04 |  1.225397e-02
   7 |   3.000000 |   5.334790 |   5.328603 |  4.334959e-04 |  6.187765e-03
   8 |   3.000000 |   5.328603 |   5.325483 |  2.184767e-04 |  3.119178e-03
   9 |   3.000000 |   5.325483 |   5.323912 |  1.100241e-04 |  1.570966e-03
  10 |   3.000000 |   5.323912 |   5.323122 |  5.538608e-05 |  7.908639e-04

---------------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:     5.32312163
Erro final estimado: 7.91e-04
Numero de iteracoes: 10
Tempo de execucao:   0.000099 segundos
```

Figura 7 Saída do Método da Posição Falsa para o Problema 1, Equação 2.

Newton-Raphson

Saída 1:

```
--- Relatorio do Metodo de Newton-Raphson ---
Funcao: f(x) = 1 - (1 + x + x**2 / 2) * e**(-x) - 0.1
Chute inicial: x0 = 1.050000

Iter |          x_i |         f(x_i) |        f'(x_i) |     Erro |x_i-x_i-1|
------------------------------------------------------------------------------
   1 |   1.05000000 |  -1.027557e-02 |   1.929032e-01 |          5.326802e-02
   2 |   1.10326802 |   2.427332e-04 |   2.019244e-01 |          1.202099e-03
   3 |   1.10206592 |   1.186401e-07 |   2.017270e-01 |          5.881222e-07

---------------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:     1.10206533
Erro final estimado: 5.88e-07
Numero de iteracoes: 3
Tempo de execucao:   0.000849 segundos
```

Figura 8 – Saída do Método de Newton-Raphson para o Problema 1, Equação 1.

Saída 2:

```
--- Relatorio do Metodo de Newton-Raphson ---
Funcao: f(x) = 1 - (1 + x + x**2 / 2) * e**(-x) - 0.9
Chute inicial: x0 = 4.500000

Iter |          x_i |         f(x_i) |        f'(x_i) |     Erro |x_i-x_i-1|
------------------------------------------------------------------------------
   1 |   4.50000000 |  -7.357807e-02 |   1.124786e-01 |          6.541518e-01
   2 |   5.15415179 |  -1.225480e-02 |   7.671225e-02 |          1.597502e-01
   3 |   5.31390198 |  -5.835609e-04 |   6.950224e-02 |          8.396290e-03
   4 |   5.32229827 |  -1.525632e-06 |   6.913909e-02 |          2.206612e-05

---------------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:     5.32232034
Erro final estimado: 2.21e-05
Numero de iteracoes: 4
Tempo de execucao:   0.001422 segundos
```

Figura 9 – Saída do Método de Newton-Raphson para o Problema 1, Equação 2.

Secante

Saída 1:

```
--- Relatorio do Metodo da Secante ---
Funcao: f(x) = 1 - (1 + x + x**2 / 2) * e**(-x) - 0.1
Pontos iniciais: x0 = 0.1, x1 = 2.0

Iter |          x_i+1 |       f(x_i+1) |     Erro |x_i+1 - x_i|
-----------------------------------------------------------------
   1 |     0.68701856 |   -6.741872e-02 |          1.312981e+00
   2 |     0.99147898 |   -2.125926e-02 |          3.044604e-01
   3 |     1.13170167 |    6.049765e-03 |          1.402227e-01
   4 |     1.10063816 |   -2.877313e-04 |          3.106352e-02
   5 |     1.10204848 |   -3.397767e-06 |          1.410328e-03
   6 |     1.10206534 |    1.977824e-09 |          1.685332e-05
   7 |     1.10206533 |   -1.365574e-14 |          9.804533e-09

---------------------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:     1.10206533
Erro final estimado: 9.80e-09
Numero de iteracoes: 7
Tempo de execucao:   0.000111 segundos
```

Figura 10 – Saída do Método da Secante para o Problema 1, Equação 1.

Saída 2:

```
--- Relatorio do Metodo da Secante ---
Funcao: f(x) = 1 - (1 + x + x**2 / 2) * e**(-x) - 0.9
Pontos iniciais: x0 = 3.0, x1 = 6.0

Iter |          x_i+1 |       f(x_i+1) |     Erro |x_i+1 - x_i|
-----------------------------------------------------------------
   1 |     5.68414489 |    2.236057e-02 |          3.158551e-01
   2 |     5.23344827 |   -6.317479e-03 |          4.506966e-01
   3 |     5.33273211 |    7.175158e-04 |          9.928384e-02
   4 |     5.32260592 |    1.974298e-05 |          1.012619e-02
   5 |     5.32231941 |   -6.426063e-08 |          2.865133e-04

---------------------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:     5.32231941
Erro final estimado: 2.87e-04
Numero de iteracoes: 5
Tempo de execucao:   0.000110 segundos
```

Figura 11 – Saída do Método da Secante para o Problema 1, Equação 2.

Problema 2 (Exercício 3.6)

A equação: tg(θ/2) = (senα cosα)/(gR/v² – cos²α), permite calcular o ângulo de inclinação, α, em que o lançamento do míssil deve ser feito para atingir um determinado alvo. Na equação acima,

• α - ângulo de inclinação com a superfície da Terra com a qual é feita o lançamento do míssil ,

• g - aceleração da gravidade ≈ 9.81 m/s2 ,

• R - raio da Terra ≈ 6371000 m,

• v - velocidade de lançamento do míssil, m/s ,

• θ - ângulo (medido do centro da Terra) entre o ponto de lançamento e o ponto de impacto desejado

Resolva o problema considerando: θ = 80º e v tal que v²/gR = 1.25, ou seja, aproximadamente 8.840 m/s.

Entrada:

```
0.8391 - (sin(x)*cos(x)) / (0.8 - cos(x)**2)
2
50
1e-6
```

Figura 12 – Entrada para o Problema 2 (Exercício 3.6).

Bisseção

Saída:

```
--- Relatorio do Metodo da Bissecao ---
Funcao: f(x) = 0.8391 - (sin(x)*cos(x)) / (0.8 - cos(x)**2)

Iter |          a |          b |   c (raiz) |           f(c) |     Erro |b-a|
---------------------------------------------------------------------------
   1 |   1.000000 |   2.000000 |   1.500000 |   7.503449e-01 |   1.000000e+00
   2 |   1.000000 |   1.500000 |   1.250000 |   4.119688e-01 |   5.000000e-01
   3 |   1.000000 |   1.250000 |   1.125000 |   2.055795e-01 |   2.500000e-01
   4 |   1.000000 |   1.125000 |   1.062500 |   8.411003e-02 |   1.250000e-01
   5 |   1.000000 |   1.062500 |   1.031250 |   1.687593e-02 |   6.250000e-02
   6 |   1.000000 |   1.031250 |   1.015625 |  -1.870349e-02 |   3.125000e-02
   7 |   1.015625 |   1.031250 |   1.023438 |  -7.384560e-04 |   1.562500e-02
   8 |   1.023438 |   1.031250 |   1.027344 |   8.111630e-03 |   7.812500e-03
   9 |   1.023438 |   1.027344 |   1.025391 |   3.697426e-03 |   3.906250e-03
  10 |   1.023438 |   1.025391 |   1.024414 |   1.482209e-03 |   1.953125e-03
  11 |   1.023438 |   1.024414 |   1.023926 |   3.725594e-04 |   9.765625e-04
  12 |   1.023438 |   1.023926 |   1.023682 |  -1.827774e-04 |   4.882812e-04
  13 |   1.023682 |   1.023926 |   1.023804 |   9.493370e-05 |   2.441406e-04
  14 |   1.023682 |   1.023804 |   1.023743 |  -4.391116e-05 |   1.220703e-04
  15 |   1.023743 |   1.023804 |   1.023773 |   2.551394e-05 |   6.103516e-05
  16 |   1.023743 |   1.023773 |   1.023758 |  -9.197942e-06 |   3.051758e-05
  17 |   1.023758 |   1.023773 |   1.023766 |   8.158165e-06 |   1.525879e-05
  18 |   1.023758 |   1.023766 |   1.023762 |  -5.198469e-07 |   7.629395e-06
  19 |   1.023762 |   1.023766 |   1.023764 |   3.819170e-06 |   3.814697e-06
  20 |   1.023762 |   1.023764 |   1.023763 |   1.649664e-06 |   1.907349e-06

---------------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:     1.02376270
Erro final estimado: 9.54e-07
Numero de iteracoes: 20
Tempo de execucao:   0.000153 segundos
```

Figura 13 – Saída do Método da Bissecção para o Problema 2.

Posição Falsa

Saída:

```
--- Relatorio do Metodo da Posicao Falsa ---
Funcao: f(x) = 0.8391 - (sin(x)*cos(x)) / (0.8 - cos(x)**2)

Iter |           a |           b |    c (raiz) |          f(c) |    Erro |c-c_old|
--------------------------------------------------------------------------------
   1 |    2.000000 |   50.000000 |   29.329453 |  6.866979e-02 |              inf
   2 |   29.329453 |   50.000000 |   30.553201 |  2.149464e+00 |     1.223748e+00
   3 |   30.553201 |   50.000000 |   43.451677 |  8.618796e+00 |     1.289848e+01
   4 |   43.451677 |   50.000000 |   49.264079 |  1.730559e+00 |     5.812402e+00
   5 |   49.264079 |   50.000000 |   49.715406 |  6.921218e+00 |     4.513263e-01
   6 |   49.715406 |   50.000000 |   49.961240 | -1.753124e+00 |     2.458344e-01
   7 |   49.715406 |   49.961240 |   49.911556 | -3.230791e+00 |     4.968424e-02
   8 |   49.715406 |   49.911556 |   49.849133 | -9.310960e+00 |     6.242313e-02
   9 |   49.715406 |   49.849133 |   49.772425 |  1.818809e+01 |     7.670732e-02
  10 |   49.772425 |   49.849133 |   49.823160 | -2.221157e+01 |     5.073483e-02
  11 |   49.772425 |   49.823160 |   49.795267 |  7.733061e+01 |     2.789380e-02
  12 |   49.795267 |   49.823160 |   49.816936 | -3.188115e+01 |     2.166966e-02
  13 |   49.795267 |   49.816936 |   49.810610 | -5.575437e+01 |     6.325817e-03
  14 |   49.795267 |   49.810610 |   49.804182 | -2.117873e+02 |     6.428119e-03
  15 |   49.795267 |   49.804182 |   49.797651 |  1.207231e+02 |     6.531025e-03
  16 |   49.797651 |   49.804182 |   49.800022 |  2.770809e+02 |     2.371190e-03
  17 |   49.800022 |   49.804182 |   49.802380 | -9.157873e+02 |     2.357713e-03
  18 |   49.800022 |   49.802380 |   49.800570 |  3.965307e+02 |     1.810060e-03
  19 |   49.800570 |   49.802380 |   49.801117 |  6.977122e+02 |     5.469288e-04
  20 |   49.801117 |   49.802380 |   49.801663 |  2.913764e+03 |     5.462055e-04
  21 |   49.801663 |   49.802380 |   49.802209 | -1.336358e+03 |     5.454825e-04
  22 |   49.801663 |   49.802209 |   49.802037 | -2.470404e+03 |     1.715150e-04
  23 |   49.801663 |   49.802037 |   49.801866 | -1.628008e+04 |     1.715865e-04
  24 |   49.801663 |   49.801866 |   49.801694 |  3.548622e+03 |     1.716581e-04
  25 |   49.801694 |   49.801866 |   49.801725 |  4.537292e+03 |     3.072059e-05
  26 |   49.801725 |   49.801866 |   49.801755 |  6.289804e+03 |     3.071830e-05
  27 |   49.801755 |   49.801866 |   49.801786 |  1.024857e+04 |     3.071601e-05
  28 |   49.801786 |   49.801866 |   49.801817 |  2.765708e+04 |     3.071372e-05
  29 |   49.801817 |   49.801866 |   49.801847 | -3.958054e+04 |     3.071143e-05
  30 |   49.801817 |   49.801847 |   49.801829 |  9.179976e+04 |     1.807879e-05
  31 |   49.801829 |   49.801847 |   49.801842 | -6.958296e+04 |     1.263225e-05
  32 |   49.801829 |   49.801842 |   49.801837 | -2.875324e+05 |     5.446614e-06
  33 |   49.801829 |   49.801837 |   49.801831 |  1.348536e+05 |     5.446686e-06
  34 |   49.801831 |   49.801837 |   49.801833 |  2.539609e+05 |     1.738943e-06
  35 |   49.801833 |   49.801837 |   49.801835 |  2.175042e+06 |     1.738936e-06
  36 |   49.801835 |   49.801837 |   49.801836 | -3.313336e+05 |     1.738928e-06
  37 |   49.801835 |   49.801836 |   49.801836 | -3.908780e+05 |     2.298799e-07

---------------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:     49.80183613
Erro final estimado: 2.30e-07
Numero de iteracoes: 37
Tempo de execucao:   0.000390 segundos
```

Figura 14 – Saída do Método da Posição Falsa para o Problema 2.

Newton-Raphson

Saída:

```
--- Relatorio do Metodo de Newton-Raphson ---
Funcao: f(x) = 0.8391 - (sin(x)*cos(x)) / (0.8 - cos(x)**2)
Chute inicial: x0 = 26.000000

Iter |           x_i |          f(x_i) |         f'(x_i) |     Erro |x_i-x_i-1|
------------------------------------------------------------------------------
   1 |    26.00000000 |   -4.540055e-01 |    3.771486e+00 |        1.203784e-01
   2 |    26.12037843 |   -8.618618e-02 |    2.504505e+00 |        3.441246e-02
   3 |    26.15479089 |   -3.903782e-03 |    2.284763e+00 |        1.708616e-03
   4 |    26.15649950 |   -8.422186e-06 |    2.274920e+00 |        3.702190e-06
   5 |    26.15650321 |   -3.929190e-11 |    2.274898e+00 |        1.727329e-11

---------------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:     26.15650321
Erro final estimado: 1.73e-11
Numero de iteracoes: 5
Tempo de execucao:   0.000420 segundos
```

Figura 15 – Saída do Método de Newton-Raphson para o Problema 2.

Secante

Saída:

```
--- Relatorio do Metodo da Secante ---
Funcao: f(x) = 0.8391 - (sin(x)*cos(x)) / (0.8 - cos(x)**2)
Pontos iniciais: x0 = 2.0, x1 = 50.0

Iter |          x_i+1 |        f(x_i+1) |     Erro |x_i+1 - x_i|
------------------------------------------------------------
   1 |    29.32945313 |    6.866979e-02 |          2.067055e+01
   2 |    30.55320122 |    2.149464e+00 |          1.223748e+00
   3 |    29.28906733 |   -2.077660e-02 |          1.264134e+00
   4 |    29.30116941 |    6.965076e-03 |          1.210207e-02
   5 |    29.29813095 |    7.982450e-05 |          3.038455e-03
   6 |    29.29809572 |   -3.086397e-07 |          3.522648e-05
   7 |    29.29809586 |    1.364575e-11 |          1.356778e-07

---------------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:     29.29809586
Erro final estimado: 1.36e-07
Numero de iteracoes: 7
Tempo de execucao:   0.000147 segundos
```

Figura 16 – Saída do Método da Secante para o Problema 2.

Análise dos Resultados Divergentes

Ao aplicar os métodos numéricos para encontrar a raiz da função **f(x) = 0.8391 - (sin(x)*cos(x)) / (0.8 - cos(x)**2)**, foram obtidos resultados distintos para cada algoritmo (aproximadamente 1.02, 49.80, 26.15 e 29.29). Essa divergência não indica uma falha nos métodos, mas sim uma consequência direta da natureza complexa da função e da estratégia de busca de cada algoritmo.

As principais razões para os diferentes resultados são:

1. Múltiplas Raízes: A função é periódica devido aos termos sin(x) e cos(x), o que significa que ela cruza o eixo x em múltiplos pontos. Portanto, existem várias raízes corretas, e cada método convergiu para uma delas.

2. Sensibilidade às Condições Iniciais: A escolha do intervalo ou do ponto inicial foi o fator determinante para a raiz encontrada:

    - Bissecção: Ao ser executado em um intervalo menor e bem definido (provavelmente [1, 2]), o método isolou e convergiu para a primeira raiz positiva (~1.02).

    - Posição Falsa e Secante: Utilizando um intervalo muito amplo como [2, 50], a reta secante inicial traçada entre os extremos apontou para regiões distantes do gráfico, fazendo com que os métodos "saltassem" para outras raízes válidas dentro desse vasto domínio.

    - Newton-Raphson: O método foi iniciado com um chute em x0 = 26 (o ponto médio do intervalo [2, 50]). Por sua natureza de rápida convergência local, ele encontrou a raiz mais próxima a esse ponto de partida (~26.15).

3. Comportamento Complexo da Função: A função também possui descontinuidades (assíntotas verticais) onde o denominador se anula. Esse comportamento errático, combinado com a periodicidade, faz com que a direção da busca de cada método seja altamente imprevisível em intervalos grandes.

Em suma, os métodos funcionaram corretamente, mas o comportamento oscilatório e as múltiplas soluções da função, aliados à sensibilidade dos algoritmos às condições iniciais, levaram cada um a convergir para uma raiz diferente e válida.

Problema 3 (Exercício 3.8)

Uma loja de eletrodomésticos oferece dois planos de financiamento para um produto cujo preço à vista é de R$ 162,00:

- Plano A: entrada de R$ 22,00 + 9 prestações iguais de R$ 26,50
- Plano B: entrada de R$ 22,00 + 12 prestações iguais de R$ 21,50

Qual dos dois planos apresenta a menor taxa de juros, sendo portanto melhor para o consumidor? Sabe-se que a equação que relaciona os juros mensais (J) com o prazo (P), o valor financiado (VF) e a prestação mensal (PM) é dada por:

$$(1 - (1 + J)^{(-P)}) / J = VF / PM$$

a) Fazendo as substituições $x = 1 + J$ e $k = VF / PM$, verifique que a equação acima pode ser reescrita da seguinte forma:

$$f(x) = k \cdot x^{(P+1)} - (k + 1) \cdot x^{P} + 1 = 0$$

b) Escreva a equação do item (a) para cada um dos planos e encontre um intervalo que contenha a raiz positiva x != 1.

Entrada Plano A:

```
5.28301887 * x**10 - 6.28301887 * x**9 + 1
1.01
1.2
1e-7
```

Figura 17 – Entrada Plano A para o Problema 3 (Exercício 3.8).

Entrada Plano B:

```
6.51162791 * x**13 - 7.51162791 * x**12 + 1
1.01
1.2
1e-7
```

Figura 18 – Entrada Plano B para o Problema 3 (Exercício 3.8)

Bisseção

Saída Plano A:

```
--- Relatorio do Metodo da Bissecao ---
Funcao: f(x) = 5.28301887 * x**10 - 6.28301887 * x**9 + 1

Iter |          a |          b |   c (raiz) |          f(c) |     Erro |b-a|
---------------------------------------------------------------------------
   1 |   1.010000 |   1.200000 |   1.105000 | -9.369603e-02 |  1.900000e-01
   2 |   1.105000 |   1.200000 |   1.152500 |  3.028443e-01 |  9.500000e-02
   3 |   1.105000 |   1.152500 |   1.128750 |  4.879592e-02 |  4.750000e-02
   4 |   1.105000 |   1.128750 |   1.116875 | -3.448958e-02 |  2.375000e-02
   5 |   1.116875 |   1.128750 |   1.122812 |  3.920526e-03 |  1.187500e-02
   6 |   1.116875 |   1.122812 |   1.119844 | -1.606393e-02 |  5.937500e-03
   7 |   1.119844 |   1.122812 |   1.121328 | -6.270087e-03 |  2.968750e-03
   8 |   1.121328 |   1.122812 |   1.122070 | -1.224824e-03 |  1.484375e-03
   9 |   1.122070 |   1.122812 |   1.122441 |  1.335284e-03 |  7.421875e-04
  10 |   1.122070 |   1.122441 |   1.122256 |  5.209504e-05 |  3.710937e-04
  11 |   1.122070 |   1.122256 |   1.122163 | -5.871474e-04 |  1.855469e-04
  12 |   1.122163 |   1.122256 |   1.122209 | -2.677220e-04 |  9.277344e-05
  13 |   1.122209 |   1.122256 |   1.122233 | -1.078624e-04 |  4.638672e-05
  14 |   1.122233 |   1.122256 |   1.122244 | -2.789594e-05 |  2.319336e-05
  15 |   1.122244 |   1.122256 |   1.122250 |  1.209649e-05 |  1.159668e-05
  16 |   1.122244 |   1.122250 |   1.122247 | -7.900492e-06 |  5.798340e-06
  17 |   1.122247 |   1.122250 |   1.122249 |  2.097806e-06 |  2.899170e-06
  18 |   1.122247 |   1.122249 |   1.122248 | -2.901391e-06 |  1.449585e-06
  19 |   1.122248 |   1.122249 |   1.122248 | -4.018041e-07 |  7.247925e-07
  20 |   1.122248 |   1.122249 |   1.122248 |  8.479982e-07 |  3.623962e-07
  21 |   1.122248 |   1.122248 |   1.122248 |  2.230963e-07 |  1.811981e-07

---------------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:     1.12224834
Erro final estimado: 9.06e-08
Numero de iteracoes: 21
Tempo de execucao:   0.000027 segundos
```

Figura 19 – Saída do Método da Bissecção para o Plano A.

Saída Plano B:

```
--- Relatorio do Metodo da Bissecao ---
Funcao: f(x) = 6.51162791 * x**13 - 7.51162791 * x**12 + 1

Iter |          a |          b |   c (raiz) |          f(c) |     Erro |b-a|
---------------------------------------------------------------------------
   1 |   1.010000 |   1.200000 |   1.105000 | -4.813636e-02 |  1.900000e-01
   2 |   1.105000 |   1.200000 |   1.152500 |  9.616872e-01 |  9.500000e-02
   3 |   1.105000 |   1.152500 |   1.128750 |  3.086635e-01 |  4.750000e-02
   4 |   1.105000 |   1.128750 |   1.116875 |  9.973946e-02 |  2.375000e-02
   5 |   1.105000 |   1.116875 |   1.110937 |  1.888379e-02 |  1.187500e-02
   6 |   1.105000 |   1.110937 |   1.107969 | -1.627224e-02 |  5.937500e-03
   7 |   1.107969 |   1.110937 |   1.109453 |  8.840935e-04 |  2.968750e-03
   8 |   1.107969 |   1.109453 |   1.108711 | -7.798207e-03 |  1.484375e-03
   9 |   1.108711 |   1.109453 |   1.109082 | -3.483250e-03 |  7.421875e-04
  10 |   1.109082 |   1.109453 |   1.109268 | -1.306147e-03 |  3.710938e-04
  11 |   1.109268 |   1.109453 |   1.109360 | -2.126711e-04 |  1.855469e-04
  12 |   1.109360 |   1.109453 |   1.109407 |  3.352997e-04 |  9.277344e-05
  13 |   1.109360 |   1.109407 |   1.109384 |  6.121145e-05 |  4.638672e-05
  14 |   1.109360 |   1.109384 |   1.109372 | -7.575554e-05 |  2.319336e-05
  15 |   1.109372 |   1.109384 |   1.109378 | -7.278471e-06 |  1.159668e-05
  16 |   1.109378 |   1.109384 |   1.109381 |  2.696488e-05 |  5.798340e-06
  17 |   1.109378 |   1.109381 |   1.109379 |  9.842805e-06 |  2.899170e-06
  18 |   1.109378 |   1.109379 |   1.109378 |  1.282066e-06 |  1.449585e-06
  19 |   1.109378 |   1.109378 |   1.109378 | -2.998227e-06 |  7.247925e-07
  20 |   1.109378 |   1.109378 |   1.109378 | -8.580868e-07 |  3.623962e-07
  21 |   1.109378 |   1.109378 |   1.109378 |  2.119882e-07 |  1.811981e-07

---------------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:     1.10937838
Erro final estimado: 9.06e-08
Numero de iteracoes: 21
Tempo de execucao:   0.000051 segundos
```

Figura 20 – Saída do Método da Bissecção para o Plano B

Posição Falsa

Saída Plano A:

```
--- Relatorio do Metodo da Posicao Falsa ---      You, yesterday • adicionando novas entr
Funcao: f(x) = 5.28301887 * x**10 - 6.28301887 * x**9 + 1

Iter |          a |          b |   c (raiz) |           f(c) |    Erro |c-c_old|
--------------------------------------------------------------------------------
   1 |   1.010000 |   1.200000 |   1.015137 |  -5.323349e-02 |              inf
   2 |   1.015137 |   1.200000 |   1.022452 |  -7.634788e-02 |     7.315034e-03
   3 |   1.022452 |   1.200000 |   1.032358 |  -1.042184e-01 |     9.905938e-03
   4 |   1.032358 |   1.200000 |   1.044871 |  -1.325507e-01 |     1.251277e-02
   5 |   1.044871 |   1.200000 |   1.059305 |  -1.533209e-01 |     1.443371e-02
   6 |   1.059305 |   1.200000 |   1.074229 |  -1.578864e-01 |     1.492443e-02
   7 |   1.074229 |   1.200000 |   1.087924 |  -1.432543e-01 |     1.369532e-02
   8 |   1.087924 |   1.200000 |   1.099110 |  -1.151718e-01 |     1.118589e-02
   9 |   1.099110 |   1.200000 |   1.107367 |  -8.364935e-02 |     8.257076e-03
  10 |   1.107367 |   1.200000 |   1.113000 |  -5.628751e-02 |     5.632468e-03
  11 |   1.113000 |   1.200000 |   1.116632 |  -3.593207e-02 |     3.631860e-03
  12 |   1.116632 |   1.200000 |   1.118887 |  -2.216668e-02 |     2.255727e-03
  13 |   1.118887 |   1.200000 |   1.120256 |  -1.338621e-02 |     1.368098e-03
  14 |   1.120256 |   1.200000 |   1.121073 |  -7.979671e-03 |     8.177080e-04
  15 |   1.121073 |   1.200000 |   1.121558 |  -4.720006e-03 |     4.844528e-04
  16 |   1.121558 |   1.200000 |   1.121843 |  -2.779087e-03 |     2.855127e-04
  17 |   1.121843 |   1.200000 |   1.122011 |  -1.631863e-03 |     1.677459e-04
  18 |   1.122011 |   1.200000 |   1.122109 |  -9.566941e-04 |     9.837515e-05
  19 |   1.122109 |   1.200000 |   1.122167 |  -5.603460e-04 |     5.763062e-05
  20 |   1.122167 |   1.200000 |   1.122201 |  -3.280210e-04 |     3.374024e-05
  21 |   1.122201 |   1.200000 |   1.122220 |  -1.919586e-04 |     1.974619e-05
  22 |   1.122220 |   1.200000 |   1.122232 |  -1.123135e-04 |     1.155380e-05
  23 |   1.122232 |   1.200000 |   1.122239 |  -6.570658e-05 |     6.759451e-06
  24 |   1.122239 |   1.200000 |   1.122243 |  -3.843773e-05 |     3.954268e-06
  25 |   1.122243 |   1.200000 |   1.122245 |  -2.248486e-05 |     2.313141e-06
  26 |   1.122245 |   1.200000 |   1.122246 |  -1.315265e-05 |     1.353091e-06
  27 |   1.122246 |   1.200000 |   1.122247 |  -7.693619e-06 |     7.914904e-07
  28 |   1.122247 |   1.200000 |   1.122248 |  -4.500335e-06 |     4.629781e-07
  29 |   1.122248 |   1.200000 |   1.122248 |  -2.632431e-06 |     2.708152e-07
  30 |   1.122248 |   1.200000 |   1.122248 |  -1.539814e-06 |     1.584107e-07
  31 |   1.122248 |   1.200000 |   1.122248 |  -9.006968e-07 |     9.266058e-08


----------------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:      1.12224818
Erro final estimado: 9.27e-08
Numero de iteracoes: 31
Tempo de execucao:    0.000041 segundos
```

Figura 19 – Saída do Método da Posição Falsa para o Plano A.

Saída Plano B:

```
--- Relatorio do Metodo da Posicao Falsa ---
Funcao: f(x) = 6.51162791 * x**13 - 7.51162791 * x**12 + 1

Iter |          a |          b |   c (raiz) |          f(c) |     Erro |c-c_old|
--------------------------------------------------------------------------------
   1 |   1.010000 |   1.200000 |   1.012709 |  -6.733439e-02 |            inf
   2 |   1.012709 |   1.200000 |   1.016060 |  -8.407973e-02 |   3.351440e-03
   3 |   1.016060 |   1.200000 |   1.020152 |  -1.037926e-01 |   4.091813e-03
   4 |   1.020152 |   1.200000 |   1.025065 |  -1.262422e-01 |   4.913169e-03
   5 |   1.025065 |   1.200000 |   1.030844 |  -1.506578e-01 |   5.778453e-03
   6 |   1.030844 |   1.200000 |   1.037470 |  -1.755287e-01 |   6.625905e-03
   7 |   1.037470 |   1.200000 |   1.044839 |  -1.985095e-01 |   7.369686e-03
   8 |   1.044839 |   1.200000 |   1.052749 |  -2.165915e-01 |   7.909676e-03
   9 |   1.052749 |   1.200000 |   1.060901 |  -2.266739e-01 |   8.152362e-03
  10 |   1.060901 |   1.200000 |   1.068940 |  -2.264819e-01 |   8.038784e-03
  11 |   1.068940 |   1.200000 |   1.076508 |  -2.154715e-01 |   7.568162e-03
  12 |   1.076508 |   1.200000 |   1.083312 |  -1.951902e-01 |   6.803552e-03
  13 |   1.083312 |   1.200000 |   1.089166 |  -1.688029e-01 |   5.853974e-03
  14 |   1.089166 |   1.200000 |   1.094007 |  -1.400320e-01 |   4.841445e-03
  15 |   1.094007 |   1.200000 |   1.097877 |  -1.121050e-01 |   3.869638e-03
  16 |   1.097877 |   1.200000 |   1.100884 |  -8.716793e-02 |   3.006697e-03
  17 |   1.100884 |   1.200000 |   1.103168 |  -6.623024e-02 |   2.284002e-03
  18 |   1.103168 |   1.200000 |   1.104872 |  -4.943404e-02 |   1.704833e-03
  19 |   1.104872 |   1.200000 |   1.106128 |  -3.640538e-02 |   1.255685e-03
  20 |   1.106128 |   1.200000 |   1.107044 |  -2.654471e-02 |   9.157201e-04
  21 |   1.107044 |   1.200000 |   1.107707 |  -1.921399e-02 |   6.629288e-04
  22 |   1.107707 |   1.200000 |   1.108184 |  -1.383411e-02 |   4.773691e-04
  23 |   1.108184 |   1.200000 |   1.108527 |  -9.922461e-03 |   3.424248e-04
  24 |   1.108527 |   1.200000 |   1.108772 |  -7.097261e-03 |   2.449452e-04
  25 |   1.108772 |   1.200000 |   1.108946 |  -5.066461e-03 |   1.748667e-04
  26 |   1.108946 |   1.200000 |   1.109071 |  -3.611652e-03 |   1.246597e-04
  27 |   1.109071 |   1.200000 |   1.109160 |  -2.571994e-03 |   8.877750e-05
  28 |   1.109160 |   1.200000 |   1.109223 |  -1.830301e-03 |   6.317784e-05
  29 |   1.109223 |   1.200000 |   1.109268 |  -1.301827e-03 |   4.493682e-05
  30 |   1.109268 |   1.200000 |   1.109300 |  -9.256063e-04 |   3.195068e-05
  31 |   1.109300 |   1.200000 |   1.109323 |  -6.579413e-04 |   2.271143e-05
  32 |   1.109323 |   1.200000 |   1.109339 |  -4.675933e-04 |   1.614091e-05


----------------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:     1.10937813
Erro final estimado: 9.57e-08
Numero de iteracoes: 32       You, 1 second ago • Uncommitted changes
Tempo de execucao:   0.000051 segundos
```

Figura 20 – Saída do Método da Posição Falsa para o Plano B.

Newton-Raphson

Saída Plano A:

```
--- Relatorio do Metodo de Newton-Raphson ---      You, yesterday • adicionando nova
Funcao: f(x) = 5.28301887 * x**10 - 6.28301887 * x**9 + 1
Chute inicial: x0 = 1.105000

Iter |            x_i |         f(x_i) |         f'(x_i) |     Erro |x_i-x_i-1|
--------------------------------------------------------------------------------
   1 |     1.10500000 |  -9.369603e-02 |    4.068123e+00 |         2.303176e-02
   2 |     1.12803176 |   4.300814e-02 |    7.987990e+00 |         5.384100e-03
   3 |     1.12264766 |   2.769036e-03 |    6.970323e+00 |         3.972608e-04
   4 |     1.12225040 |   1.441478e-05 |    6.897810e+00 |         2.089763e-06
   5 |     1.12224831 |   3.976091e-10 |    6.897429e+00 |         5.764589e-11

--------------------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:     1.12224831
Erro final estimado: 5.76e-11
Numero de iteracoes: 5
Tempo de execucao:   0.000012 segundos
```

Figura 21 – Saída do Método de Newton-Raphson para o Plano A.

Saída Plano B:

```
--- Relatorio do Metodo de Newton-Raphson ---      You, yesterday • adicionando nov
Funcao: f(x) = 6.51162791 * x**13 - 7.51162791 * x**12 + 1
Chute inicial: x0 = 1.105000

Iter |            x_i |         f(x_i) |         f'(x_i) |     Erro |x_i-x_i-1|
--------------------------------------------------------------------------------
   1 |     1.10500000 |  -4.813636e-02 |    1.019680e+01 |         4.720732e-03
   2 |     1.10972073 |   4.066203e-03 |    1.194236e+01 |         3.404856e-04
   3 |     1.10938025 |   2.224248e-05 |    1.181184e+01 |         1.883067e-06
   4 |     1.10937836 |   6.777761e-10 |    1.181112e+01 |         5.738454e-11

--------------------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:     1.10937836
Erro final estimado: 5.74e-11
Numero de iteracoes: 4
Tempo de execucao:   0.000008 segundos
```

Figura 22 – Saída do Método de Newton-Raphson para o Plano B.

Secante

Saída Plano A:

```
--- Relatorio do Metodo da Secante ---        You, yesterday • adicion
Funcao: f(x) = 5.28301887 * x**10 - 6.28301887 * x**9 + 1
Pontos iniciais: x0 = 1.01, x1 = 1.2

Iter |           x_i+1 |         f(x_i+1) |     Erro |x_i+1 - x_i|
-----------------------------------------------------------------
   1 |      1.01513723 |   -5.323349e-02 |          1.848628e-01
   2 |      1.02245226 |   -7.634788e-02 |          7.315034e-03
   3 |      0.99829037 |    6.387877e-03 |          2.416189e-02
   4 |      1.00015587 |   -5.790761e-04 |          1.865495e-03
   5 |      1.00000081 |   -3.018950e-06 |          1.550554e-04
   6 |      1.00000000 |    1.464655e-09 |          8.126008e-07
   7 |      1.00000000 |   -3.552714e-15 |          3.940452e-10

----------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:      1.00000000
Erro final estimado: 3.94e-10
Numero de iteracoes: 7
Tempo de execucao:   0.000016 segundos
```

Figura 23 – Saída do Método de Secante para o Plano A.

Saída Plano B:

```
--- Relatorio do Metodo da Secante ---        You, yesterday • adiciona
Funcao: f(x) = 6.51162791 * x**13 - 7.51162791 * x**12 + 1
Pontos iniciais: x0 = 1.01, x1 = 1.2

Iter |           x_i+1 |         f(x_i+1) |     Erro |x_i+1 - x_i|
-----------------------------------------------------------------
   1 |      1.01270886 |   -6.733439e-02 |          1.872911e-01
   2 |      1.01606030 |   -8.407973e-02 |          3.351440e-03
   3 |      0.99923245 |    4.219671e-03 |          1.682786e-02
   4 |      1.00003662 |   -2.009723e-04 |          8.041734e-04
   5 |      1.00000006 |   -3.363912e-07 |          3.655952e-05
   6 |      1.00000000 |    2.726797e-11 |          6.129659e-08

----------------------------------------------------------------------
--- Resumo Final ---
Raiz encontrada:      1.00000000
Erro final estimado: 6.13e-08
Numero de iteracoes: 6
Tempo de execucao:   0.000022 segundos
```

Figura 24 – Saída do Método de Secante para o Plano B

Método da Eliminação de Gauss

Estratégia de Implementação:

A estratégia adotada no método da Eliminação de Gauss para resolver sistemas lineares do tipo **Ax=b** se divide em duas etapas principais: a fase de triangularização e a substituição regressiva. Uma decisão crucial na implementação para garantir maior estabilidade numérica foi a incorporação do pivoteamento parcial. Antes de cada etapa de eliminação, um laço percorre a coluna atual para encontrar o elemento de maior valor absoluto usando np.argmax, e a linha correspondente é trocada com a linha do pivô. Essa abordagem minimiza erros de arredondamento e evita falhas na divisão por zero. Após o pivoteamento, uma verificação explícita **if A[i, i] == 0** garante que, se ainda assim o pivô for nulo, o método identifique a matriz como singular e encerre a execução.

Uma vez que a matriz está em formato triangular superior, inicia-se a fase de substituição regressiva. Esta etapa foi implementada de forma eficiente utilizando a função np.dot do **NumPy** para calcular os somatórios, resolvendo o sistema de "trás para frente" a partir da última variável. Para refinar a solução, valores computados muito próximos de zero (menores que 1e-12) são explicitamente convertidos para 0.0, melhorando a clareza do resultado final.

Estrutura dos Arquivos de Entrada/Saída

A estrutura de arquivos foi mantida consistente para garantir uma experiência de uso uniforme em todo o código.

- **Arquivo de Entrada (ex: entrada_sistemas.txt)** O método utiliza um sistema de leitura flexível, localizado no diretório **input/**, que aceita múltiplos formatos para definir a Matriz A e o Vetor b:
    - **Formato de Dimensão:** A primeira linha contém a ordem n do sistema, seguida por n linhas da matriz aumentada.
    - **Formato de Listas:** Um arquivo de duas linhas contendo a Matriz A (ex: [[...]] ou [...],[...]) na primeira e o Vetor b (ex: [...]) na segunda.
- **Arquivo de Saída (gauss_saida.txt)** O relatório gerado no diretório **output/** foi projetado para ser claro e direto:
    - O cabeçalho informa o método executado.

- O corpo do relatório exibe a Matriz A e o Vetor b originais para referência da entrada.
- Ao final, são apresentados a Solução (x) encontrada, formatada para fácil leitura, e o Tempo de execução.

Dificuldades enfrentadas

Um dos principais desafios na implementação foi o tratamento de pivôs nulos ou próximos de zero. Sem uma estratégia de pivoteamento, o algoritmo poderia falhar ou gerar grande instabilidade numérica devido a divisões por valores muito pequenos.

Método de Fatoração LU

Estratégia de Implementação:

A estratégia adotada na Fatoração LU para resolver sistemas lineares consiste em decompor a matriz de coeficientes A em um produto de duas matrizes: uma triangular inferior (L) e uma triangular superior (U). O processo inicia com uma verificação de singularidade, utilizando **np.linalg.det(A) == 0**, para encerrar o método caso a decomposição não seja possível. Em seguida, a matriz L é inicializada como uma matriz identidade com **np.eye(n)** e a matriz U como uma cópia de A. O processo de decomposição ocorre em um laço aninhado que calcula os multiplicadores e os armazena em L, enquanto atualiza U para sua forma triangular superior.

Após a decomposição, o sistema original Ax=b é transformado em LUx=b. Este sistema é então resolvido em duas etapas mais simples: primeiro, resolve-se Ly=b por meio de substituição progressiva para encontrar o vetor auxiliar y; em seguida, resolve-se Ux=y através de substituição regressiva para encontrar a solução final x. Ambas as etapas de substituição foram implementadas de forma otimizada usando np.dot, aproveitando a eficiência do **NumPy** para operações vetoriais.

Estrutura dos Arquivos de Entrada/Saída

A estrutura de arquivos foi mantida consistente para garantir uma experiência de uso uniforme em todo o código.

- **Arquivo de Entrada (ex: entrada_sistemas.txt)** O método utiliza o mesmo sistema de leitura flexível dos outros métodos de sistemas lineares, localizado no diretório **input/**, que aceita os formatos de "dimensão + matriz aumentada" e os baseados em "listas Python".
- **Arquivo de Saída (lu_saida.txt)** O relatório gerado no diretório **output/** foi enriquecido para fornecer uma visão completa do processo:
    - O cabeçalho informa o método e exibe a Matriz A e o Vetor b originais.
    - Como um diferencial, as matrizes Matriz L e Matriz U resultantes da decomposição são apresentadas, permitindo a verificação da corretude da fatoração.
    - O arquivo é concluído com a Solução (x) final e o Tempo de execução total.

Dificuldades enfrentadas

O principal desafio na implementação da Fatoração LU (especificamente, a decomposição de Doolittle) foi garantir o preenchimento correto e simultâneo das matrizes L e U dentro do laço de repetição. A lógica para calcular cada fator da matriz L e, com base nele, atualizar a linha correspondente da matriz U exigiu um controle rigoroso dos índices. Outra dificuldade foi a implementação correta das duas fases de resolução: a substituição progressiva para encontrar y e a regressiva para encontrar x. Coordenar a passagem do vetor auxiliar y entre essas duas etapas foi fundamental para que o resultado final fosse correto.

Método de Jacobi

Estratégia de Implementação:

O método de Jacobi foi implementado como uma abordagem iterativa para a solução de sistemas lineares. A estratégia central consiste em calcular uma nova aproximação para cada variável do vetor solução x utilizando apenas os valores da iteração anterior. Uma decisão de implementação fundamental foi a inclusão da função auxiliar **_make_diagonally_dominant**, que é chamada no início do método. Essa função tenta reorganizar as linhas da matriz A e do vetor b para que a matriz se torne diagonalmente dominante, uma condição que garante a convergência do método.

O processo iterativo continua até que um de dois critérios de parada seja atingido: ou a norma Euclidiana da diferença entre os vetores de solução de duas iterações consecutivas **(np.linalg.norm(x_new - x))** se torna menor que uma tolerância definida, ou um número máximo de iterações é alcançado, o que evita laços infinitos caso o método não convirja. Para fins de análise, a implementação armazena o histórico de cada iteração (vetor x e erro) em uma lista chamada passos, que é posteriormente utilizada pelo módulo de relatórios.

Estrutura dos Arquivos de Entrada/Saída

A estrutura de arquivos foi mantida consistente para garantir uma experiência de uso uniforme em todo o código.

- **Arquivo de Entrada (ex: entrada_sistemas.txt)** O método utiliza o mesmo sistema de leitura flexível dos outros métodos de sistemas lineares, localizado no diretório **input/**, que aceita os formatos de "dimensão + matriz aumentada" e os baseados em "listas Python".
- **Arquivo de Saída (jacobi_saida.txt)** O relatório gerado no diretório **output/** foi adaptado para refletir a natureza iterativa do método de Jacobi:
    - O cabeçalho informa o método e exibe as matrizes de entrada.
    - Uma tabela de progresso detalhada exibe a evolução do cálculo, com as colunas: **Iter** (iteração), **Vetor x** (a aproximação atual) e **Erro** (a norma da diferença para a iteração anterior).
    - O arquivo é concluído com um bloco de resumo final, que informa a Solução encontrada, o Erro final estimado, o Número de iterações e o Tempo de execução.

Dificuldades enfrentadas

A principal dificuldade foi lidar com a convergência do método, que não é garantida para todos os sistemas. A implementação da função **_make_diagonally_dominant** foi uma resposta direta a esse desafio, na tentativa de pré-condicionar o sistema para o sucesso do algoritmo. Outro desafio de implementação foi garantir que o vetor da iteração anterior (x) fosse corretamente preservado para o cálculo de todas as componentes do novo vetor **(x_new)**, sem utilizar valores recém-calculados dentro da mesma iteração, que é a característica que define o método de Jacobi.

Método de Gauss-Seidel

Estratégia de Implementação:

O método de Gauss-Seidel foi implementado como um aprimoramento do método de Jacobi. A estratégia iterativa é semelhante, mas com a diferença crucial de que o método utiliza os valores das variáveis recém-calculados dentro da mesma iteração. Para implementar essa lógica, o algoritmo mantém uma cópia do vetor da iteração anterior (x_old), enquanto o vetor principal x é atualizado "no lugar" (in-place). O cálculo do somatório para cada variável x[i] foi dividido em duas partes usando np.dot: uma que utiliza as componentes já atualizadas de x (índices < i) e outra que utiliza as componentes de x_old (índices > i). Assim como em Jacobi, a implementação inicia com uma chamada à função **_make_diagonally_dominant** para melhorar as condições de convergência. O critério de parada também se baseia na norma Euclidiana da diferença entre os vetores x e x_old (**np.linalg.norm(x - x_old)**), comparada a uma tolerância, ou no atingimento do número máximo de iterações.

Estrutura dos Arquivos de Entrada/Saída

A estrutura de arquivos foi mantida consistente para garantir uma experiência de uso uniforme em todo o código.

- **Arquivo de Entrada (ex: entrada_sistemas.txt)** O método utiliza o mesmo sistema de leitura flexível dos outros métodos de sistemas lineares, localizado no diretório **input/**, que aceita os formatos de "dimensão + matriz aumentada" e os baseados em "listas Python".
- **Arquivo de Saída (gauss-seidel_saida.txt)** O relatório gerado no diretório **output/** segue o mesmo formato detalhado do método de Jacobi para permitir uma comparação direta:
  - O cabeçalho informa o método e exibe as matrizes de entrada.
  - A tabela de progresso contém as colunas: **Iter** (iteração), **Vetor x** (a aproximação atual) e **Erro** (a norma da diferença para a iteração anterior).
  - O arquivo é concluído com o bloco de resumo final, informando a Solução encontrada, o Erro final estimado, o Número de iterações e o Tempo de execução.

Dificuldades enfrentadas

A maior dificuldade na implementação de Gauss-Seidel foi a correta manipulação dos vetores para garantir que os valores recém-calculados de x fossem usados imediatamente no cálculo das componentes seguintes na mesma iteração. Diferente de Jacobi, onde se pode calcular todo o novo vetor x_new de uma vez, aqui o vetor x é atualizado "no lugar" (in-place). Isso exigiu uma implementação cuidadosa do cálculo do somatório, dividindo-o em dois produtos escalares distintos que usam partes do vetor x atualizado e partes do x_old.

Problemas Estabelecidos

Problema 1 (Exercício 4.1)

Considere o circuito a seguir com resistências e baterias tal como indicado; escolhemos arbitrariamente as correntes e os valores da malha:

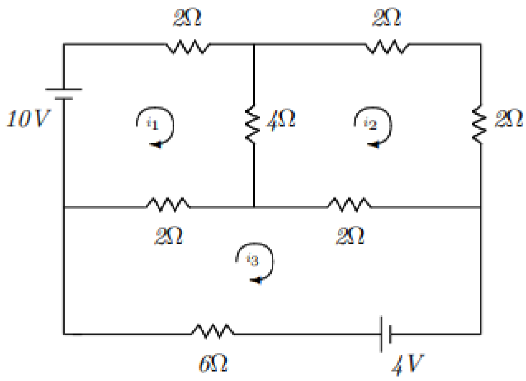


Figura 25 – Circuito Elétrico do Problema 1 (Exercício 4.1).

Aplicando a Lei de Kirchoff que diz que a soma algébrica da diferenças de potencial em qualquer circuito fechado é zero, obtemos para as correntes i1, i2, i3, o seguinte sistema linear:

$$\begin{cases} 2\,i_1 + 4\,(i_1 - i_2) + 2\,(i_1 - i_3) - 10 = 0 \\ 2\,i_2 - 2\,i_2 + 2\,(i_2 - i_3) + 4\,(i_2 - i_1) = 0 \\ 6\,i_3 + 2\,(i_3 - i_1) + 2\,(i_3 - i_2) - 4 = 0 \end{cases}$$

Figura 26 – Sistema de Equações Lineares do Problema 1.

Entrada:

```
[8, -4, -2],[-6, 8, -2],[-2, -2, 10]
[10, 0 ,4]
```

Figura 27 – Entrada para o Problema 1 (Exercício 4.1).

Saída Eliminação de Gauss:

```
--- Relatorio do Metodo de Eliminacao de Gauss ---
Matriz A:
[[ 8. -4. -2.]
 [-6.  8. -2.]
 [-2. -2. 10.]]

Vetor b:
[10.  0.  4.]

Solucao (x): [2.89189 2.54054 1.48649]
Tempo de execucao: 0.020034 segundos
```

Figura 28 – Saída do Método da Eliminação de Gauss para o Problema 1 .

Saída Fatoração de LU:

```
--- Relatorio do Metodo de Fatoracao LU ---
Matriz A:
[[ 8. -4. -2.]
 [-6.  8. -2.]
 [-2. -2. 10.]]

Vetor b:
[10.  0.  4.]

Matriz L:
[[ 1.    0.    0.  ]
 [-0.75  1.    0.  ]
 [-0.25 -0.6   1.  ]]

Matriz U:
[[ 8.  -4.  -2. ]
 [ 0.   5.  -3.5]
 [ 0.   0.   7.4]]

Solucao (x): [2.89189 2.54054 1.48649]
Tempo de execucao: 0.027830 segundos
```

Figura 29 – Saída do Método da Fatoração de LU para o Problema 1.

Problema 2 (Exercício 4.3)

Um cachorro está perdido em um labirinto quadrado de corredores (Figura 30). Em cada interseção ao escolhe uma direção ao acaso e segue até a interseção seguinte onde escolhe novamente ao acaso nova direção e assim por diante. Qual a probabilidade do cachorro, estando na interseção i, sair eventualmente pelo lado sul?

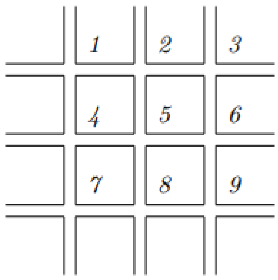


Figura 30 – Labirinto do Problema 2 (Exercício 4.3) .

Esclarecimentos: Suponhamos que haja exatamente às 9 interseções mostradas na figura. Seja P1 a probabilidade de o cachorro, que está na interseção 1, sair pelo lado sul. Sejam P2,P3,...,P9 definidas de modo similar. Supondo que, em cada interseção a que chegue, o cachorro tenha a mesma possibilidade de escolher uma direção ou outra e que, tendo chegado a uma saída, sua caminhada termine, a teoria das probabilidades oferece as seguintes equações para Pi:

$$\left\{\begin{array}{lcl} P_1 & = & (0+0+P_2+P_4)/4 \\ P_2 & = & (0+P_1+P_3+P_5)/4 \\ P_3 & = & (0+P_2+0+P_6)/4 \\ P_4 & = & (P_1+0+P_5+P_7)/4 \\ P_5 & = & (P_2+P_4+P_6+P_8)/4 \\ P_6 & = & (P_3+P_5+0+P_9)/4 \\ P_7 & = & (P_4+0+P_8+1)/4 \\ P_8 & = & (P_5+P_7+P_9+1)/4 \\ P_9 & = & (P_6+P_8+0+1)/4 \end{array}\right.$$

Figura 31 – Sistema de Equações Lineares do Problema 2.

Para saber a resposta resolva o sistema linear obtido.

Entrada:

```
9
4 -1 0 -1 0 0 0 0 0 0
-1 4 -1 0 -1 0 0 0 0 0
0 -1 4 0 0 -1 0 0 0 0
-1 0 0 4 -1 0 -1 0 0 0
0 -1 0 -1 4 -1 0 -1 0 0
0 0 -1 0 -1 4 0 0 -1 0
0 0 0 -1 0 0 4 -1 0 1
0 0 0 0 -1 0 -1 4 -1 1
0 0 0 0 0 -1 0 -1 4 1
```

Figura 32 – Entrada para o Problema 2 (Exercício 4.3)

Saída Eliminação de Gauss:

```
--- Relatorio do Metodo de Eliminacao de Gauss ---
Matriz A:
[[ 4. -1.  0. -1.  0.  0.  0.  0.  0.]
 [-1.  4. -1.  0. -1.  0.  0.  0.  0.]
 [ 0. -1.  4.  0.  0. -1.  0.  0.  0.]
 [-1.  0.  0.  4. -1.  0. -1.  0.  0.]
 [ 0. -1.  0. -1.  4. -1.  0. -1.  0.]
 [ 0.  0. -1.  0. -1.  4.  0.  0. -1.]
 [ 0.  0.  0. -1.  0.  0.  4. -1.  0.]
 [ 0.  0.  0.  0. -1.  0. -1.  4. -1.]
 [ 0.  0.  0.  0.  0. -1.  0. -1.  4.]]

Vetor b:
[0. 0. 0. 0. 0. 0. 1. 1. 1.]

Solucao (x): [0.07143 0.09821 0.07143 0.18750 0.25000 0.18750 0.42857 0.52679 0.42857]
Tempo de execucao: 0.009501 segundos
```

Figura 33 – Saída do Método da Eliminação de Gauss para o Problema 2.

Saída Fatoração de LU:

```
--- Relatorio do Metodo de Fatoracao LU ---
Matriz A:
[[ 4. -1.  0. -1.  0.  0.  0.  0.  0.]
 [-1.  4. -1.  0. -1.  0.  0.  0.  0.]
 [ 0. -1.  4.  0.  0. -1.  0.  0.  0.]
 [-1.  0.  0.  4. -1.  0. -1.  0.  0.]
 [ 0. -1.  0. -1.  4. -1.  0. -1.  0.]
 [ 0.  0. -1.  0. -1.  4.  0.  0. -1.]
 [ 0.  0.  0. -1.  0.  0.  4. -1.  0.]
 [ 0.  0.  0.  0. -1.  0. -1.  4. -1.]
 [ 0.  0.  0.  0.  0. -1.  0. -1.  4.]]

Vetor b:
[0. 0. 0. 0. 0. 0. 1. 1. 1.]

Matriz L:
[[ 1.          0.          0.          0.          0.          0.
   0.          0.          0.        ]
 [-0.25        1.          0.          0.          0.          0.
   0.          0.          0.        ]        You, 31 seconds ago • Uncommitted changes
 [ 0.         -0.26666667  1.          0.          0.          0.
   0.          0.          0.        ]
 [-0.25       -0.06666667 -0.01785714  1.          0.          0.
   0.          0.          0.        ]
 [ 0.         -0.26666667 -0.07142857 -0.28708134  1.          0.
   0.          0.          0.        ]
 [ 0.          0.         -0.26785714 -0.00478469 -0.31601124  1.
   0.          0.          0.        ]
 [ 0.          0.          0.         -0.26794258 -0.08426966 -0.02815735
   1.          0.          0.        ]
 [ 0.          0.          0.          0.         -0.29353933 -0.0931677
  -0.295038    1.          0.        ]
 [ 0.          0.          0.          0.          0.         -0.29482402
  -0.00759946 -0.32835821  1.        ]]

Matriz U:
[[ 4.         -1.          0.         -1.          0.          0.
   0.          0.          0.        ]
 [ 0.          3.75       -1.         -0.25       -1.          0.
   0.          0.          0.        ]
 [ 0.          0.          3.73333333 -0.06666667 -0.26666667 -1.
   0.          0.          0.        ]
 [ 0.          0.          0.          3.73214286 -1.07142857 -0.01785714
  -1.          0.          0.        ]
 [ 0.          0.          0.          0.          3.40669856 -1.07655502
  -0.28708134 -1.          0.        ]
 [ 0.          0.          0.          0.          0.          3.39185393
  -0.09550562 -0.31601124 -1.        ]
 [ 0.          0.          0.          0.          0.          0.
   3.70517598 -1.0931677  -0.02815735]
 [ 0.          0.          0.          0.          0.          0.
   0.          3.35449262 -1.10147519]
 [ 0.          0.          0.          0.          0.          0.
   0.          0.          3.34328358]]

Solucao (x): [0.07143 0.09821 0.07143 0.18750 0.25000 0.18750 0.42857 0.52679 0.42857]
Tempo de execucao: 0.009000 segundos
```

Figura 34 – Saída do Método da Fatoração de LU para o Problema 2

Problema 3 (Exercício 4.6)

Considere o circuito em escada, mostrado na Figura 4.7, composto de resistores concentrados e fontes de tensão ideais, uma independente e outra controlada.

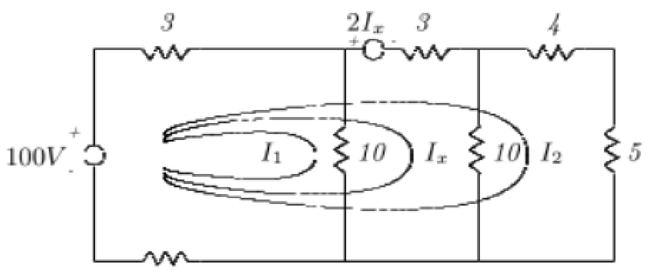


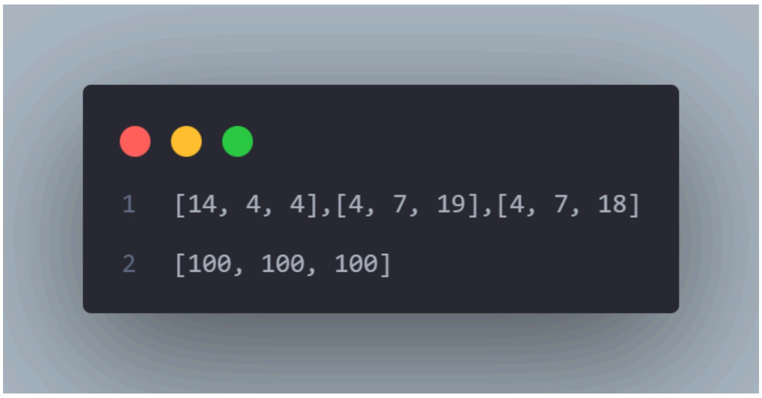
Figura 35 – Circuito em Escada do Problema 3 (Exercício 4.6).

O sistema de equações que se obtém ao solucionar o circuito pelo método dos laços é o seguinte:

$$\begin{cases} 4\ (I_1 + I_2 + \ I_x) + 10\ I_1 = 100 \\ 4\ (I_1 + I_2 + I_x) + 3\ (I_2 + I_x) + 10\ I_x = 100 - 2\ I_x \\ 4\ (I_1 + I_2 + I_x) + 3\ (I_2 + I_x) + 9\ I_x = 100 - 2\ I_x \end{cases}$$

Figura 36 – Sistema de Equações Lineares do Problema 3.

Entrada:



```
[14, 4, 4],[4, 7, 19],[4, 7, 18]
[100, 100, 100]
```

Figura 37 – Entrada para o Problema 3 (Exercício 4.6).

Saída Eliminação de Gauss:

```
--- Relatorio do Metodo de Eliminacao de Gauss ---
Matriz A:
[[14.   4.   4.]
 [ 4.   7. 19.]
 [ 4.   7. 18.]]

Vetor b:
[100. 100. 100.]

Solucao (x): [3.65854 12.19512 0.00000]
Tempo de execucao: 0.007031 segundos
```

Figura 38 – Saída do Método da Eliminação de Gauss para o Problema 3.

Saída Fatoração de LU:

```
--- Relatorio do Metodo de Fatoracao LU ---
Matriz A:
[[14.   4.   4.]
 [ 4.   7. 19.]
 [ 4.   7. 18.]]

Vetor b:
[100. 100. 100.]

Matriz L:
[[1.         0.         0.        ]
 [0.28571429 1.         0.        ]
 [0.28571429 1.         1.        ]]

Matriz U:
[[14.          4.          4.        ]
 [ 0.          5.85714286 17.85714286]
 [ 0.          0.         -1.        ]]

Solucao (x): [3.65854 12.19512 0.00000]
Tempo de execucao: 0.003349 segundos
```

Figura 39 – Saída do Método da Fatoração de LU para o Problema 3.

Problema 4 (Exercício 5.1)

Uma maneira de se obter a solução da equação de Laplace:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \ ,$$

Figura 40 – Sistema Linear da Equação de Laplace (Exercício 5.1).

em uma região retangular consiste em se fazer uma discretização que transforma a equação em um problema aproximado consistindo em uma equação de diferenças cuja solução, em um caso particular, exige a solução do seguinte sistema linear:

$$\begin{pmatrix} 4 & -1 & 0 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 \\ -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 0 & -1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} 100 \\ 0 \\ 0 \\ 100 \\ 0 \\ 0 \end{pmatrix} .$$

Figura 41 - Matriz do (Exercício 5.1).

Entrada:

```
[4, -1, 0, -1, 0, 0][-1, 4, -1, 0, -1, 0][0, -1, 4, 0, 0, -1][-1, 0, 0, 4, -1, 0][0, -1, 0, -1, 4, -1][0, 0, -1, 0, -1, 4]
[100, 0, 0, 100, 0, 0]
```

Figura 42 – Entrada para o Problema 4 (Exercício 5.1).

Saída de Jacobi:

```
--- Relatorio do Metodo de Jacobi ---
Matriz A:
[[ 4. -1.  0. -1.  0.  0.]
 [-1.  4. -1.  0. -1.  0.]
 [ 0. -1.  4.  0.  0. -1.]
 [-1.  0.  0.  4. -1.  0.]
 [ 0. -1.  0. -1.  4. -1.]
 [ 0.  0. -1.  0. -1.  4.]]

Vetor b:
[100.   0.   0. 100.   0.   0.]

Iter | Vetor x                                         |            Erro
-----------------------------------------------------------------------
   1 | [25.000000 0.000000 0.000000 25.000000 0.000000 0.000000] |    3.535534e+01
   2 | [31.250000 6.250000 0.000000 31.250000 6.250000 0.000000] |    1.250000e+01
   3 | [34.375000 9.375000 1.562500 34.375000 9.375000 1.562500] |    6.629126e+00
   4 | [35.937500 11.328125 2.734375 35.937500 11.328125 2.734375] |    3.906250e+00
   5 | [36.816406 12.500000 3.515625 36.816406 12.500000 3.515625] |    2.347815e+00
   6 | [37.329102 13.208008 4.003906 37.329102 13.208008 4.003906] |    1.416016e+00
   7 | [37.634277 13.635254 4.302979 37.634277 13.635254 4.302979] |    8.545358e-01
   8 | [37.817383 13.893127 4.484558 37.817383 13.893127 4.484558] |    5.157471e-01
   9 | [37.927628 14.048767 4.594421 37.927628 14.048767 4.594421] |    3.112798e-01
  10 | [37.994099 14.142704 4.660797 37.994099 14.142704 4.660797] |    1.878738e-01
  11 | [38.034201 14.199400 4.700875 38.034201 14.199400 4.700875] |    1.133919e-01
  12 | [38.058400 14.233619 4.725069 38.058400 14.233619 4.725069] |    6.843805e-02
  13 | [38.073005 14.254272 4.739672 38.073005 14.254272 4.739672] |    4.130602e-02
  14 | [38.081819 14.266737 4.748486 38.081819 14.266737 4.748486] |    2.493039e-02
  15 | [38.087139 14.274261 4.753806 38.087139 14.274261 4.753806] |    1.504682e-02
  16 | [38.090350 14.278801 4.757017 38.090350 14.278801 4.757017] |    9.081559e-03
  17 | [38.092288 14.281542 4.758954 38.092288 14.281542 4.758954] |    5.481206e-03
  18 | [38.093457 14.283196 4.760124 38.093457 14.283196 4.760124] |    3.308200e-03
  19 | [38.094163 14.284194 4.760830 38.094163 14.284194 4.760830] |    1.996676e-03
  20 | [38.094589 14.284797 4.761256 38.094589 14.284797 4.761256] |    1.205100e-03
  21 | [38.094847 14.285161 4.761513 38.094847 14.285161 4.761513] |    7.273424e-04
  22 | [38.095002 14.285380 4.761668 38.095002 14.285380 4.761668] |    4.389900e-04
  23 | [38.095095 14.285513 4.761762 38.095095 14.285513 4.761762] |    2.649539e-04
  24 | [38.095152 14.285593 4.761819 38.095152 14.285593 4.761819] |    1.599138e-04
  25 | [38.095186 14.285641 4.761853 38.095186 14.285641 4.761853] |    9.651652e-05
  26 | [38.095207 14.285670 4.761873 38.095207 14.285670 4.761873] |    5.825287e-05
  27 | [38.095219 14.285688 4.761886 38.095219 14.285688 4.761886] |    3.515872e-05
  28 | [38.095227 14.285698 4.761893 38.095227 14.285698 4.761893] |    2.122016e-05
  29 | [38.095231 14.285705 4.761898 38.095231 14.285705 4.761898] |    1.280750e-05
  30 | [38.095234 14.285708 4.761901 38.095234 14.285708 4.761901] |    7.730011e-06
  31 | [38.095236 14.285711 4.761902 38.095236 14.285711 4.761902] |    4.665475e-06
  32 | [38.095237 14.285712 4.761903 38.095237 14.285712 4.761903] |    2.815863e-06
  33 | [38.095237 14.285713 4.761904 38.095237 14.285713 4.761904] |    1.699524e-06
  34 | [38.095238 14.285714 4.761904 38.095238 14.285714 4.761904] |    1.025753e-06
  35 | [38.095238 14.285714 4.761904 38.095238 14.285714 4.761904] |    6.190969e-07

---------------------------------------------------------------------------
--- Resumo Final ---
Solucao encontrada:  [38.09524 14.28571 4.76190 38.09524 14.28571 4.76190]
Erro final estimado: 6.19e-07
Numero de iteracoes: 35
Tempo de execucao:   0.004665 segundos
```

Figura 43 – Saída do Método de Jacobi para o Problema 4.

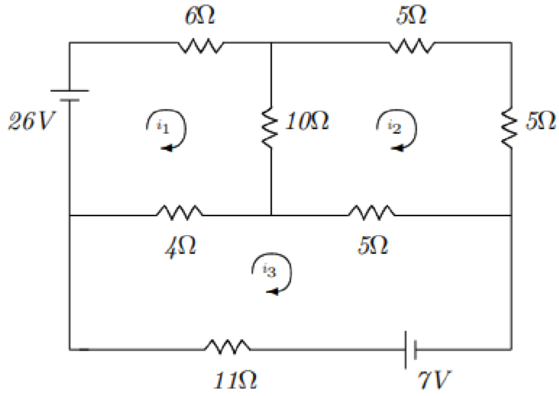
Saída de Gauss-Seidel:

```
--- Relatorio do Metodo de Gauss-Seidel ---
Matriz A:
[[ 4. -1.  0. -1.  0.  0.]
 [-1.  4. -1.  0. -1.  0.]
 [ 0. -1.  4.  0.  0. -1.]
 [-1.  0.  0.  4. -1.  0.]
 [ 0. -1.  0. -1.  4. -1.]
 [ 0.  0. -1.  0. -1.  4.]]

Vetor b:
[100.   0.   0. 100.   0.   0.]

Iter | Vetor x                                     |            Erro
-----------------------------------------------------------------
   1 | [25.000000 6.250000 1.562500 31.250000 9.375000 2.734375] |    4.169453e+01
   2 | [34.375000 11.328125 3.515625 35.937500 12.500000 4.003906] |    1.228180e+01
   3 | [36.816406 13.208008 4.302979 37.329102 13.635254 4.484558] |    3.683847e+00
   4 | [37.634277 13.893127 4.594421 37.817383 14.048767 4.660797] |    1.289850e+00
   5 | [37.927628 14.142704 4.700875 37.994099 14.199400 4.725069] |    4.666100e-01
   6 | [38.034201 14.233619 4.739672 38.058400 14.254272 4.748486] |    1.697720e-01
   7 | [38.073005 14.266737 4.753806 38.081819 14.274261 4.757017] |    6.183128e-02
   8 | [38.087139 14.278801 4.758954 38.090350 14.281542 4.760124] |    2.252290e-02
   9 | [38.092288 14.283196 4.760830 38.093457 14.284194 4.761256] |    8.204518e-03
  10 | [38.094163 14.284797 4.761513 38.094589 14.285161 4.761668] |    2.988712e-03
  11 | [38.094847 14.285380 4.761762 38.095002 14.285513 4.761819] |    1.088718e-03
  12 | [38.095095 14.285593 4.761853 38.095152 14.285641 4.761873] |    3.965945e-04
  13 | [38.095186 14.285670 4.761886 38.095207 14.285688 4.761893] |    1.444701e-04
  14 | [38.095219 14.285698 4.761898 38.095227 14.285705 4.761901] |    5.262711e-05
  15 | [38.095231 14.285708 4.761902 38.095234 14.285711 4.761903] |    1.917083e-05
  16 | [38.095236 14.285712 4.761904 38.095237 14.285713 4.761904] |    6.983486e-06
  17 | [38.095237 14.285714 4.761904 38.095238 14.285714 4.761905] |    2.543921e-06
  18 | [38.095238 14.285714 4.761905 38.095238 14.285714 4.761905] |    9.266912e-07

---------------------------------------------------------------------------
--- Resumo Final ---
Solucao encontrada:  [38.09524 14.28571 4.76190 38.09524 14.28571 4.76190]
Erro final estimado: 9.27e-07
Numero de iteracoes: 18
Tempo de execucao:   0.000536 segundos
```

Figura 44 – Saída do Método de Gauss-Seidel para o Problema 4.

Problema 5 (Exercício 5.2)

Considere o circuito da figura a seguir, com resistências e baterias tal como indicado; escolhemos arbitrariamente as orientações das correntes.

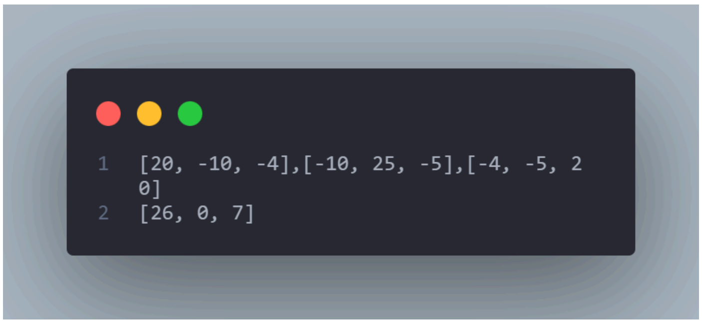


Aplicando a lei de Kirchoff, que diz que a soma algébrica das diferenças de potencial em qualquer circuito fechado é zero, achamos para as correntes i1, i2, i3:

$$\begin{cases} 6i_1 + 10(i_1 - i_2) + 4(i_1 - i_3) - 26 = 0 \\ 5i_2 + 5i_2 + 5(i_2 - i_3) + 10(i_2 - i_1) = 0 \\ 11i_3 + 4(i_3 - i_1) + 5(i_3 - i_2) - 7 = 0 \end{cases}$$

Figura 45 – Sistema de Equações Lineares do Problema 5.

Entrada:



```
[20, -10, -4],[-10, 25, -5],[-4, -5, 20]
[26, 0, 7]
```

Figura 46 – Entrada para o Problema 5 (Exercício 5.2).

Saída de Jacobi:

```
--- Relatorio do Metodo de Jacobi ---
Matriz A:
[[ 20. -10.  -4.]
 [-10.  25.  -5.]
 [ -4.  -5.  20.]]

Vetor b:
[26.  0.  7.]

Iter | Vetor x                              |            Erro
-----------------------------------------------------------------
   1 | [1.300000 0.000000 0.350000]         |    1.346291e+00
   2 | [1.370000 0.590000 0.610000]         |    6.485368e-01
   3 | [1.717000 0.670000 0.771500]         |    3.910131e-01
   4 | [1.789300 0.841100 0.860900]         |    2.061428e-01
   5 | [1.892730 0.887900 0.918135]         |    1.271371e-01
   6 | [1.927577 0.940719 0.950521]         |    7.108455e-02
   7 | [1.960464 0.961135 0.970695]         |    4.365025e-02
   8 | [1.974707 0.978325 0.982376]         |    2.519506e-02
   9 | [1.985638 0.986358 0.989522]         |    1.533256e-02
  10 | [1.991083 0.992160 0.993717]         |    8.995028e-03
  11 | [1.994823 0.995177 0.996257]         |    5.434959e-03
  12 | [1.996840 0.997181 0.997759]         |    3.215348e-03
  13 | [1.998142 0.998288 0.998663]         |    1.933756e-03
  14 | [1.998876 0.998989 0.999200]         |    1.149114e-03
  15 | [1.999335 0.999391 0.999523]         |    6.891563e-04
  16 | [1.999600 0.999638 0.999715]         |    4.105102e-04
  17 | [1.999762 0.999783 0.999830]         |    2.457920e-04
  18 | [1.999857 0.999871 0.999898]         |    1.466048e-04
  19 | [1.999915 0.999923 0.999939]         |    8.769695e-05
  20 | [1.999949 0.999954 0.999964]         |    5.234603e-05
  21 | [1.999970 0.999972 0.999978]         |    3.129595e-05
  22 | [1.999982 0.999984 0.999987]         |    1.868811e-05
  23 | [1.999989 0.999990 0.999992]         |    1.116963e-05
  24 | [1.999994 0.999994 0.999995]         |    6.671376e-06
  25 | [1.999996 0.999996 0.999997]         |    3.986712e-06
  26 | [1.999998 0.999998 0.999998]         |    2.381483e-06
  27 | [1.999999 0.999999 0.999999]         |    1.423001e-06
  28 | [1.999999 0.999999 0.999999]         |    8.500982e-07

------------------------------------------------------------------
--- Resumo Final ---
Solucao encontrada:  [2.00000 1.00000 1.00000]
Erro final estimado: 8.50e-07
Numero de iteracoes: 28
Tempo de execucao:   0.011413 segundos
```

Figura 47 – Saída do Método de Jacobi para o Problema 5.

Saída de Gauss-Seidel:

```
--- Relatorio do Metodo de Gauss-Seidel ---
Matriz A:
[[ 20. -10.  -4.]
 [-10.  25.  -5.]
 [ -4.  -5.  20.]]

Vetor b:
[26.  0.  7.]

Iter | Vetor x                                    |          Erro
-----------------------------------------------------------------
   1 | [1.300000 0.520000 0.740000]               |  1.583667e+00
   2 | [1.708000 0.831200 0.899400]               |  5.373247e-01
   3 | [1.895480 0.938072 0.963614]               |  2.251529e-01
   4 | [1.961759 0.977426 0.986708]               |  8.046732e-02
   5 | [1.986055 0.991764 0.995152]               |  2.944737e-02
   6 | [1.994912 0.996995 0.998231]               |  1.073803e-02
   7 | [1.998144 0.998904 0.999355]               |  3.917736e-03
   8 | [1.999323 0.999600 0.999765]               |  1.429259e-03
   9 | [1.999753 0.999854 0.999914]               |  5.214249e-04
  10 | [1.999910 0.999947 0.999969]               |  1.902269e-04
  11 | [1.999967 0.999981 0.999989]               |  6.939883e-05
  12 | [1.999988 0.999993 0.999996]               |  2.531818e-05
  13 | [1.999996 0.999997 0.999998]               |  9.236611e-06
  14 | [1.999998 0.999999 0.999999]               |  3.369713e-06
  15 | [1.999999 1.000000 1.000000]               |  1.229343e-06
  16 | [2.000000 1.000000 1.000000]               |  4.484907e-07


----------------------------------------------------------------------
--- Resumo Final ---
Solucao encontrada:  [2.00000 1.00000 1.00000]
Erro final estimado: 4.48e-07
Numero de iteracoes: 16
Tempo de execucao:   0.001181 segundos
```

Figura 48 – Saída do Método de Gauss-Seidel para o Problema 5.

Problema 6 (Exercício 5.5)

Suponha que uma membrana com dimensões 80cm × 80cm tenha cada um dos seus lados mantidos a uma temperatura constante. Usando a teoria de equações diferenciais parciais pode-se formular uma equação que determina o valor da temperatura no interior dessa membrana. Essa equação diferencial pode ser simplificada colocando-se uma malha com 9 pontos sobre essa membrana e calculando-se a temperatura nos pontos da malha, como mostra a figura:

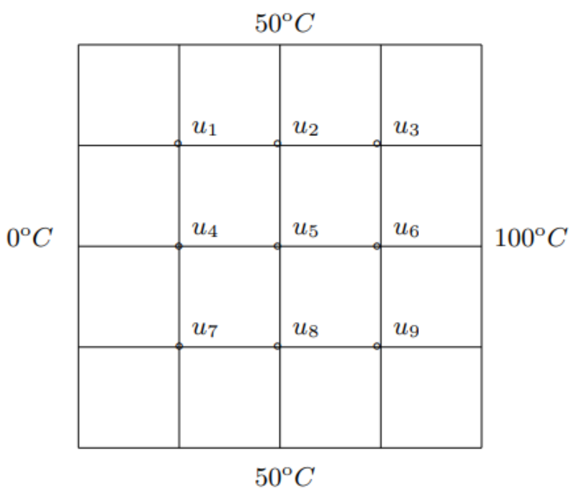


Figura 49 – Malha de Temperatura da Membrana (Exercício 5.5).

onde u1 , u2 , ..., u9 são os valores da temperatura em cada ponto da malha. O sistema de equações resultante é dado por:

Entrada:

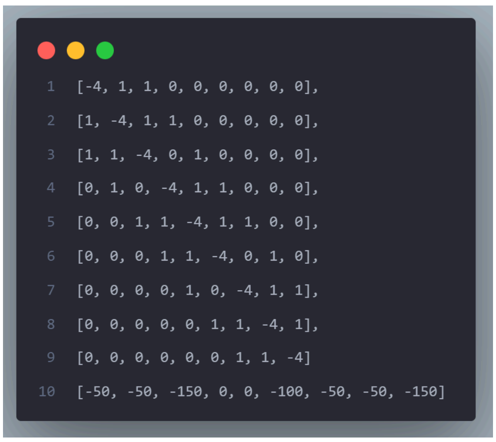


Figura 50 – Entrada para o Problema 6 (Exercício 5.5).

Saída de Jacobi:

```
--- Resumo Final ---
Solucao encontrada:  [46.03676 55.38839 78.75865 46.75815 63.60947 68.03475 60.88631 61.77137
 68.16442]
Erro final estimado: 8.63e-07
Numero de iteracoes: 64
Tempo de execucao:   0.003849 segundos
```

Figura 51 – Saída do Método de Jacobi para o Problema 6.

Saída de Gauss-Seidel:

```
--- Resumo Final ---
Solucao encontrada:  [46.03676 55.38839 78.75866 46.75815 63.60947 68.03475 60.88631 61.77137
 68.16442]
Erro final estimado: 6.44e-07
Numero de iteracoes: 35
Tempo de execucao:   0.013529 segundos
```

Figura 52 – Saída do Método de Gauss-Seidel para o Problema 6.

Condição da Matriz

Estratégia de Implementação:

A estratégia central adotada foi basear o cálculo na definição $Cond(A)=||A|| \cdot ||A^{-1}||$. Uma decisão fundamental no desenvolvimento foi implementar o cálculo da matriz inversa a partir de seus princípios matemáticos, em vez de utilizar uma função pronta como np.linalg.inv(). Para isso, o método da **Fatoração LU**, já desenvolvido para o módulo de sistemas lineares, foi reutilizado como um passo intermediário.

O processo se inicia com a decomposição da matriz de entrada A nas matrizes L (triangular inferior) e U (triangular superior). Em seguida, a matriz inversa é construída coluna por coluna, resolvendo-se **n** sistemas lineares do tipo Ax = e, onde e é cada um dos vetores da base canônica (colunas da matriz identidade). Com A e sua inversa A^{-1} em mãos, o número de condição é finalmente calculado pelo produto de suas normas. Foi escolhida a **Norma-2 (ou Espectral)**, computada eficientemente pela função np.linalg.norm(A, ord=2), por ser a definição mais robusta e comum em contextos teóricos. Toda a lógica de cálculo foi encapsulada em um bloco **try...except** para lidar de forma segura com matrizes singulares, que não admitem inversa.

Estrutura dos Arquivos de Entrada/Saída

A estrutura de arquivos foi mantida consistente para garantir uma experiência de uso uniforme em todo o código.

- **Arquivo de Entrada (ex: entrada_matriz.txt)** O método se destaca por um sistema de leitura de arquivos (_parse_matrix) excepcionalmente flexível, localizado no diretório **input/**. Ele é capaz de interpretar automaticamente múltiplos formatos de matriz, tais como:
  - **Formato de Dimensão:** A primeira linha contém a ordem n da matriz, seguida por n linhas com os valores da matriz separados por espaço.
  - **Formato de Linha Única:** Um arquivo de uma única linha contendo a matriz inteira, seja como lista aninhada ([[...]]) ou como sequência de listas ([...],[...]).
  - **Formato Linha por Linha:** Um arquivo onde cada linha representa uma linha da matriz, com valores em formato de lista ([...]) ou separados por espaço.

- **Arquivo de Saída (condicionamento_saida.txt)** O relatório gerado no diretório **output/** foi projetado para ser completo e informativo:
    - O cabeçalho informa o tipo de análise e exibe a **Matriz Original A** para referência.
    - O corpo do relatório apresenta os resultados principais: a **Norma da Matriz A**, o **Número de Condição K(A)** e uma conclusão automática se a matriz é considerada "bem condicionada" ou "mal condicionada" (baseado em um limiar de 100).
    - Em seguida, a **Matriz Inversa A^{-1}** completa é exibida, juntamente com o valor de sua norma.
    - O arquivo é concluído com o **Tempo de execução** total do processo.

Dificuldades enfrentadas

O principal obstáculo na implementação foi o cálculo da matriz inversa a partir da fatoração LU. Diferente de usar uma função pronta, essa abordagem exigiu a construção de um laço para resolver n sistemas lineares distintos (**Ax = e_i**), o que demandou um controle cuidadoso da criação dos vetores da base canônica e da montagem da matriz inversa, que foi preenchida coluna por coluna no array A_inv. Outro ponto crítico foi garantir a estabilidade numérica do processo de fatoração LU. A presença de um pivô nulo na diagonal da matriz U poderia causar uma falha por divisão por zero. Para contornar isso, uma verificação explícita **if U[i,i] == 0** foi adicionada para lançar uma exceção (**np.linalg.LinAlgError**), que é capturada pela função principal, informando ao usuário que a matriz é provavelmente singular e o cálculo não pode prosseguir.

Problemas Estabelecidos

Problema 1 (Exercício 4.1)

Entrada:

```
8 -4 -2
-6 8 -2
-2 -2 10
```

Figura 53 – Entrada para a Matriz do Problema 4.1.

Saída:

```
--- Relatório de Análise de Condicionamento de Matriz ---

Matriz Original A:
[[8.0000 -4.0000 -2.0000]
 [-6.0000 8.0000 -2.0000]
 [-2.0000 -2.0000 10.0000]]

------------------------------------------------------------
Norma da Matriz A (norma-inf): 13.06363
Número de Condição K(A):        6.34471
-> A matriz é considerada bem condicionada.

------------------------------------------------------------
Matriz Inversa A⁻¹:
[[0.25676 0.14865 0.08108]
 [0.21622 0.25676 0.09459]
 [0.09459 0.08108 0.13514]]

Norma da Matriz Inversa (norma-inf): 0.48568

------------------------------------------------------------
Tempo de execução: 0.003768 segundos
```

Figura 54 – Saída da Análise de Condicionamento para o Problema 4.1.

Problema 2 (Exercício 4.3)

Entrada:

```
4 -1 0 -1 0 0 0 0 0 0
-1 4 -1 0 -1 0 0 0 0 0
0 -1 4 0 0 -1 0 0 0 0
-1 0 0 4 -1 0 -1 0 0 0
0 -1 0 -1 4 -1 0 -1 0 0
0 0 -1 0 -1 4 0 0 -1 0
0 0 0 -1 0 0 4 -1 0 1
0 0 0 0 -1 0 -1 4 -1 1
0 0 0 0 0 -1 0 -1 4 1
```

Figura 55 – Entrada para a Matriz do Problema 4.3.

Saída:

```
--- Relatório de Análise de Condicionamento de Matriz ---

Matriz Original A:
[[4.0000 -1.0000 0.0000 -1.0000 0.0000 0.0000 0.0000 0.0000 0.0000]
 [-1.0000 4.0000 -1.0000 0.0000 -1.0000 0.0000 0.0000 0.0000 0.0000]
 [0.0000 -1.0000 4.0000 0.0000 0.0000 -1.0000 0.0000 0.0000 0.0000]
 [-1.0000 0.0000 0.0000 4.0000 -1.0000 0.0000 -1.0000 0.0000 0.0000]
 [0.0000 -1.0000 0.0000 -1.0000 4.0000 -1.0000 0.0000 -1.0000 0.0000]
 [0.0000 0.0000 -1.0000 0.0000 -1.0000 4.0000 0.0000 0.0000 -1.0000]
 [0.0000 0.0000 0.0000 -1.0000 0.0000 0.0000 4.0000 -1.0000 0.0000]
 [0.0000 0.0000 0.0000 0.0000 -1.0000 0.0000 -1.0000 4.0000 -1.0000]
 [0.0000 0.0000 0.0000 0.0000 0.0000 -1.0000 0.0000 -1.0000 4.0000]]

------------------------------------------------------------
Norma da Matriz A (norma-inf): 6.82843
Número de Condição K(A):         5.82843
-> A matriz é considerada bem condicionada.

------------------------------------------------------------
Matriz Inversa A⁻¹:
[[0.29911 0.09821 0.03125 0.09821 0.06250 0.02679 0.03125 0.02679 0.01339]
 [0.09821 0.33036 0.09821 0.06250 0.12500 0.06250 0.02679 0.04464 0.02679]
 [0.03125 0.09821 0.29911 0.02679 0.06250 0.09821 0.01339 0.02679 0.03125]
 [0.09821 0.06250 0.02679 0.33036 0.12500 0.04464 0.09821 0.06250 0.02679]
 [0.06250 0.12500 0.06250 0.12500 0.37500 0.12500 0.06250 0.12500 0.06250]
 [0.02679 0.06250 0.09821 0.04464 0.12500 0.33036 0.02679 0.06250 0.09821]
 [0.03125 0.02679 0.01339 0.09821 0.06250 0.02679 0.29911 0.09821 0.03125]
 [0.02679 0.04464 0.02679 0.06250 0.12500 0.06250 0.09821 0.33036 0.09821]
 [0.01339 0.02679 0.03125 0.02679 0.06250 0.09821 0.03125 0.09821 0.29911]]

Norma da Matriz Inversa (norma-inf): 0.85355

------------------------------------------------------------
Tempo de execução: 0.011676 segundos
```

Figura 56 – Saída da Análise de Condicionamento para o Problema 4.3.

Problema 3 (Exercício 4.6)

Entrada:

```
14 4 4
4 7 19
4 7 18
```

Figura 57 – Entrada para a Matriz do Problema 4.6.

Saída:

```
--- Relatório de Análise de Condicionamento de Matriz ---

Matriz Original A:
[[14.0000 4.0000 4.0000]
 [4.0000 7.0000 19.0000]
 [4.0000 7.0000 18.0000]]

------------------------------------------------------------
Norma da Matriz A (norma-inf): 29.83806
Número de Condição K(A):       134.14125
-> AVISO: A matriz é considerada mal condicionada.

------------------------------------------------------------
Matriz Inversa A⁻¹:
[[0.08537 0.53659 -0.58537]
 [-0.04878 -2.87805 3.04878]
 [-0.00000 1.00000 -1.00000]]

Norma da Matriz Inversa (norma-inf): 4.49564

------------------------------------------------------------
Tempo de execução: 0.000305 segundos
```

Figura 58 – Saída da Análise de Condicionamento para o Problema 4.6.

Problema 4 (Exercício 5.1)

Entrada:

```
4 -1 0 -1 0 0
-1 4 -1 0 -1 0
0 -1 4 0 0 -1
-1 0 0 4 -1 0
0 -1 0 -1 4 -1
0 0 -1 0 -1 4
```

Figura 59 – Entrada para a Matriz do Problema 5.1.

Saída:

```
--- Relatório de Análise de Condicionamento de Matriz ---

Matriz Original A:
[[4.0000 -1.0000 0.0000 -1.0000 0.0000 0.0000]
 [-1.0000 4.0000 -1.0000 0.0000 -1.0000 0.0000]
 [0.0000 -1.0000 4.0000 0.0000 0.0000 -1.0000]
 [-1.0000 0.0000 0.0000 4.0000 -1.0000 0.0000]
 [0.0000 -1.0000 0.0000 -1.0000 4.0000 -1.0000]
 [0.0000 0.0000 -1.0000 0.0000 -1.0000 4.0000]]


-----------------------------------------------------------
Norma da Matriz A (norma-inf): 6.41421
Número de Condição K(A):        4.04482
-> A matriz é considerada bem condicionada.


-----------------------------------------------------------
Matriz Inversa A⁻¹:
[[0.29482 0.09317 0.02816 0.08613 0.04969 0.01946]
 [0.09317 0.32298 0.09317 0.04969 0.10559 0.04969]
 [0.02816 0.09317 0.29482 0.01946 0.04969 0.08613]
 [0.08613 0.04969 0.01946 0.29482 0.09317 0.02816]
 [0.04969 0.10559 0.04969 0.09317 0.32298 0.09317]
 [0.01946 0.04969 0.08613 0.02816 0.09317 0.29482]]

Norma da Matriz Inversa (norma-inf): 0.63060


-----------------------------------------------------------
Tempo de execução: 0.002795 segundos
```

Figura 60 – Saída da Análise de Condicionamento para o Problema 5.1.

Problema 5 (Exercício 5.2)

Entrada:

```
20 -10 -4
-10 25 -5
-4 -5 20
```

Figura 61 – Entrada para a Matriz do Problema 5.2.

Saída:

```
--- Relatório de Análise de Condicionamento de Matriz ---

Matriz Original A:
[[20.0000 -10.0000 -4.0000]
 [-10.0000 25.0000 -5.0000]
 [-4.0000 -5.0000 20.0000]]


------------------------------------------------------------
Norma da Matriz A (norma-inf): 33.00423
Número de Condição K(A):        3.78371
-> A matriz é considerada bem condicionada.


------------------------------------------------------------
Matriz Inversa A⁻¹:
[[0.07090 0.03284 0.02239]
 [0.03284 0.05731 0.02090]
 [0.02239 0.02090 0.05970]]

Norma da Matriz Inversa (norma-inf): 0.11464


------------------------------------------------------------
Tempo de execução: 0.019614 segundos
```

Figura 62 – Saída da Análise de Condicionamento para o Problema 5.2.

Problema 6 (Exercício 5.5)

Entrada:

```
-4 1 1 0 0 0 0 0 0
1 -4 1 1 0 0 0 0 0
1 1 -4 0 1 0 0 0 0
0 1 0 -4 1 1 0 0 0
0 0 1 1 -4 1 1 0 0
0 0 0 1 1 -4 0 1 0
0 0 0 0 1 0 -4 1 1
0 0 0 0 0 1 1 -4 1
0 0 0 0 0 0 1 1 -4
```

Figura 63 – Entrada para a Matriz do Problema 5.5.

Saída:

```
------------------------------------------------------------
Norma da Matriz A (norma-inf): 6.26566
Número de Condição K(A):        6.38096
-> A matriz é considerada bem condicionada.

------------------------------------------------------------
Matriz Inversa A⁻¹:
[[-0.31083 -0.12124 -0.12207 -0.05206 -0.05621 -0.03078 -0.01994 -0.01486
  -0.00870]
 [-0.12124 -0.34597 -0.13899 -0.12363 -0.08876 -0.05980 -0.03261 -0.02682
  -0.01486]
 [-0.12207 -0.13899 -0.34929 -0.08460 -0.13609 -0.06333 -0.04716 -0.03261
  -0.01994]
 [-0.05206 -0.12363 -0.08460 -0.35786 -0.16272 -0.14510 -0.06333 -0.05980
  -0.03078]
 [-0.05621 -0.08876 -0.13609 -0.16272 -0.39941 -0.16272 -0.13609 -0.08876
  -0.05621]
 [-0.03078 -0.05980 -0.06333 -0.14510 -0.16272 -0.35786 -0.08460 -0.12363
  -0.05206]
 [-0.01994 -0.03261 -0.04716 -0.06333 -0.13609 -0.08460 -0.34929 -0.13899
  -0.12207]
 [-0.01486 -0.02682 -0.03261 -0.05980 -0.08876 -0.12363 -0.13899 -0.34597
  -0.12124]
 [-0.00870 -0.01486 -0.01994 -0.03078 -0.05621 -0.05206 -0.12207 -0.12124
  -0.31083]]

Norma da Matriz Inversa (norma-inf): 1.01840

------------------------------------------------------------
Tempo de execução: 0.002757 segundos
```

Figura 64 – Saída da Análise de Condicionamento para o Problema 5.5.

Considerações Finais

A escolha de um método numérico é sempre um balanço entre segurança, velocidade e complexidade, e a minha experiência neste projeto ajudou a clarificar quando cada ferramenta se mostra mais adequada.

Para a tarefa de encontrar raízes de funções, onde se busca o valor de para o qual f(x)=0, o **Método da Bissecção** revelou-se a escolha mais segura. A sua maior força é a robustez, com convergência praticamente infalível para um intervalo válido, funcionando como uma "rede de segurança" ideal para garantir uma solução, não importa o tempo. Contudo, a sua convergência linear é lenta, pois avança sobre a raiz de forma metódica, sem acelerar o passo. Como uma melhoria inteligente, o **Método da Posição Falsa** é geralmente mais rápido, pois usa a informação dos valores de f(x) para traçar uma reta secante mais promissora, sendo uma boa opção para acelerar a busca mantendo a segurança de isolar a raiz num intervalo. A sua principal fraqueza, no entanto, é a tendência a estagnar em funções com curvaturas acentuadas. No extremo da velocidade, o **Método de Newton-Raphson** é incomparável, com a sua convergência quadrática, tornando-o a ferramenta de eleição para o "refinamento final" de uma solução quando se exige alta precisão. No entanto, é um método exigente: precisa da derivada, é extremamente sensível a um bom chute inicial e pode divergir espetacularmente. Como alternativa pragmática, o **Método da Secante** oferece uma velocidade quase tão boa, mas com a grande vantagem de não precisar da derivada, representando o melhor equilíbrio entre velocidade e simplicidade quando a derivada é um obstáculo.

No que diz respeito à resolução de sistemas lineares, onde se procura o vetor x que satisfaz a equação Ax=b, os métodos dividem-se em duas categorias principais. Os **Métodos Diretos**, como a **Eliminação de Gauss** e a **Fatoração LU**, são a minha escolha padrão para sistemas de pequeno a médio porte onde a exatidão é fundamental. Eles fornecem a solução exata (dentro da precisão da máquina) num número finito de passos. A Fatoração LU, em particular, brilha em cenários onde o mesmo sistema precisa de ser resolvido repetidamente para múltiplos vetores de resultado, pois o maior custo computacional da decomposição é pago uma única vez. Por outro lado, para problemas massivos, como os que surgem na discretização de equações diferenciais, os **Métodos Iterativos** como **Jacobi** e **Gauss-Seidel** são a solução. São muito mais económicos em termos de memória e, muitas vezes, mais rápidos para sistemas de grande escala e esparsos. A sua desvantagem é que a convergência não é garantida e a solução é sempre uma aproximação. Entre os dois, a minha experiência mostrou que o **Gauss-Seidel** é quase sempre preferível, pois a sua convergência é tipicamente mais rápida ao aproveitar a informação mais recente disponível a cada passo da iteração, acelerando a aproximação da solução.

A aplicação destes algoritmos em problemas práticos, desde circuitos elétricos a sistemas de equações diferenciais, tornou claro que a escolha do método ideal é uma decisão estratégica. Fatores como a natureza da função, as propriedades da matriz – seja a sua esparsidade, dominância diagonal ou o seu número de condição – e a precisão exigida são cruciais para o sucesso computacional.

Ao longo deste projeto, a jornada de transformar algoritmos numéricos em código funcional foi uma experiência de profundo aprendizado. Mais do que apenas implementar fórmulas, este trabalho permitiu-me vivenciar na prática as vantagens, limitações e os contextos ideais para cada método. Transformar a teoria matemática em código Python funcional revelou os desafios reais da análise numérica, como a importância da estabilidade, o impacto da velocidade de convergência e a notável sensibilidade dos métodos às condições iniciais. Este projeto reforçou a minha percepção de que a análise numérica é uma ferramenta indispensável na engenharia e na ciência, atuando como a ponte essencial entre modelos matemáticos abstratos e soluções quantitativas e concretas.