

**UNIVERSIDADE ESTADUAL DE SANTA CRUZ – UESC**

**ISRAEL SANTANA DOS ANJOS**

# RELATÓRIO DE IMPLEMENTAÇÃO DE MÉTODOS PARA A DISCIPLINA DE ANÁLISE NÚMERICA

**ILHÉUS – BAHIA**

**2024**

**ISRAEL SANTANA DOS ANJOS**

**RELATÓRIO DE IMPLEMENTAÇÃO DE MÉTODOS PARA A DISCIPLINA DE ANÁLISE NÚMERICA**

Relatório da análise de implementação de métodos apresentado ao Curso de graduação em Ciência da computação da Universidade Estadual de Santa Cruz, como parte de um dos créditos da disciplina de análise numérica.

Orientador: Gesil Sampaio Amarantes

**ILHÉUS – BAHIA**

**2024**

Sumário

[**RELATÓRIO DE IMPLEMENTAÇÃO DE MÉTODOS PARA A DISCIPLINA DE ANÁLISE NÚMERICA** 1](#_Toc163165344)

[Linguagem escolhida e justificativa 4](#_Toc163165345)

[Teste e funcionamento do código 4](#_Toc163165346)

[Método da Bissecção 4](#_Toc163165347)

[Estratégias da implementação 4](#_Toc163165348)

[Estrutura dos arquivos de entrada/saída 5](#_Toc163165349)

[Problema teste 3.3: 6](#_Toc163165350)

[Problema teste 3.6: 7](#_Toc163165351)

[Problema teste 3.8: 7](#_Toc163165352)

[Dificuldades enfrentadas: 8](#_Toc163165353)

# Linguagem escolhida e justificativa

Para um melhor desenvolvimento foi escolhido o python como linguagem de programação, pois, essa linguagem é bastante simples implementar métodos numéricos, como também, possui uma vasta biblioteca e comunidade grande. A versão foi a 3.11.7, o interpretador pode ser baixado através do link: [www.python.org](http://www.python.org).

A instalação das bibliotecas utilizadas pode ser feita pelo prompt de comando, ou terminal da IDE, pelo seguinte comando:

* pip install sympy
* pip install numpy

Uma vez instalado não se faz necessário chamar os comandos novamente.

Como ambiente de desenvolvimento foi escolhido o *visual studio code,* por facilitar a escrita e depuração de códigos, sendo também, uma opção de fácil identificação de erros de sintaxe e formatação concisa. A versão utilizada foi a 1.87.2 e o download poder ser feito pelo link: [code.visualstudio.com](https://code.visualstudio.com/).

Outra ferramenta utilizada na realização dos cálculos e plotar os gráficos das funções foi o geoGebra. Essa ferramenta, foi de grande auxílio na escolha dos intervalos para a resolução dos problemas. O geoGebra pode ser acessado pelo link: [www.geogebra.org](https://www.geogebra.org).

## Teste e funcionamento do código

Para executar o código de cada método que vai ser apresentado, é necessário certificar-se que o interpretador python e as bibliotecas está devidamente instalado na máquina.

Abra a pasta de métodos e clique na pasta do método desejado, em seguida, edite o arquivo “entrada.txt” com a entrada desejada e salve – o. Após, abra o arquivo com a terminação “.py” do método escolhido, e ao abrir com o visual studio code, clique no menu superior em “executar” e depois em “iniciar depuração” (ou pressione F5). Por conseguinte, o arquivo “resultado.txt” será atualizado, esses passos valem para todos os métodos implementados.

Na pasta onde estão os métodos há um arquivo “entradasDeTeste.txt”, nele, contém algumas entradas teste para os problemas desenvolvidos nesse relatório.

# Método da Bissecção

## Estratégias da implementação

A iteração dos dados foi feita através de arquivos externos (.txt). Foi necessário o uso da biblioteca sympy para a conversão das strings (x e função) em variáveis simbólicas e função.

Após a leitura do arquivo e atribuições necessárias, é feita a verificação inicial (figura 1), o código apura se o valor da função tem o mesmo sinal.

Texto

Descrição gerada automaticamente

Figura 1

Se tiverem, uma mensagem de erro é exibida e o programa é encerrado ‘*SystemExit()*’. Isso ocorre porque o método da bissecção depende da mudança de sinal dentro do intervalo.

O loop *while* é executado enquanto a diferença entre ‘a’ e ‘b’ dividida por 2.0 for maior que a tolerância especificada (figura 2).

Texto

Descrição gerada automaticamente

Figura 2

Dentro do loop ‘c’ é calculado como ponto médio do intervalo. A variável ‘*cont’* é um contador para quantificar o número de iterações.

Três condições são verificadas, mostrado na figura 2:

* **Caso base:** Se ‘*f.subs*(x, c) for menor que a tolerância.
* **Mudança de sinal:** Se ‘*f.subs*(x, c) e *f.subs*(x, a)’ tem sinais opostos, indica que a raiz esta no subintervalo [a, c], então b é atualizado para diminuir a busca.
* **Mesmo sinal:** Se ‘*f.subs*(x, c)’ tem o mesmo sinal que ‘*f.subs*(x, a)’ então, a raiz está no subintervalo [c, b]. Portanto, a é atualizado.

Após o loop terminar o resultado é calculado como a média de ‘a’ e ‘b’, aproximando a raiz da função ‘f’ (raiz estimada) dentro da tolerância especificada (figura 3).



Figura 3

## Estrutura dos arquivos de entrada/saída

O arquivo de entrada é organizado de maneira que em cada linha esteja armazenado um dos dados necessários para o cálculo do método. Neste sentido, a primeira linha contém o valor do intervalo inicial ‘a’, em seguida, na segunda linha, o valor do final ‘b’, na terceira, o erro tolerado e por último a função (figura 4).

Essa estrutura foi pensada no intuito de criar uma padronização, de forma que, o algoritmo não encontre dificuldades para encontrar as informações necessárias.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Figura 4

A saída esta estruturada da seguinte forma; a primeira linha contém o título com o número do problema, na segunda linha o intervalo, e na terceira e quarta linha, respectivamente, a função aplicando a raiz e o número de iterações (figura 5).

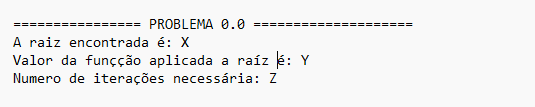


Figura 5

## Problema teste 3.3:

Para g = 0.1 e intervalo inicial e final, mostrados na figura 6, foi encontrada a seguinte raiz (figura 8):

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Figura 6 - entrada

Texto

Descrição gerada automaticamente

Figura 7 – saída

Para g = 0.9 e dados de entrada e saída mostrados respectivamente na figura 8 e 9, foi encontrada a seguinte raiz:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Figura 8 - entrada

Texto

Descrição gerada automaticamente

Figura 9 – saída

## Problema teste 3.6:

Ao plotar o gráfico para a função verificou – se que não é possível encontrar raiz para o intervalo inicialmente proposto, então, foi definido um novo intervalo para que seja possível o teste. Nesse sentido, ficou – se então definido os seguintes intervalos (figura 10):

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Figura 10 – entrada

Esses foram os resultados:

Texto

Descrição gerada automaticamente

Figura 11 – saída

## Problema teste 3.8:

Neste problema ao plotar novamente o gráfico, assim como no problema anterior, foi verificado que existe uma raiz entre os pontos 1.01 e 1.2.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Figura 12 - entrada

Texto

Descrição gerada automaticamente

Figura 13 – saída

Nesta função o problema se repete quanto a raiz, foi verificado ao projetar o gráfico no *geogebra* que existe uma raiz entre 1.01 e 1.2, o resultado gerado pelo algoritmo está demostrado na figura 15.

Texto

Descrição gerada automaticamente

Figura 14 - entrada

Texto

Descrição gerada automaticamente

Figura 15 – saída

## Dificuldades enfrentadas:

Não houve maiores dificuldades, com a utilização da ferramenta geogebra, foi bastante simples encontrar a raiz que desejamos encontrar com a saída do algoritmo.