INSTITUTO FEDERAL DE EDUCAÇÃO DO CEARÁ

IFCE CAMPUS TIANGUÁ

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ANNA ZENAIDE DIAS ALVES

ANTONIA ESTEFANE RIBEIRO VERAS

WANDERSON GOMES DA COSTA

RELATÓRIO DO SEMINÁRIO DE ANÁLISE DE MÉTODOS NUMÉRICOS PARA ENCONTRAR RAIZES DE FUNÇÕES

TIANGUÁ – CE

2021

ANNA ZENAIDE DIAS ALVES

ANTONIA ESTEFANE RIBEIRO VERAS

WANDERSON GOMES DA COSTA

RELATÓRIO DO SEMINÁRIO DE ANÁLISE DE MÉTODOS NUMÉRICOS PARA ENCONTRAR RAIZES DE FUNÇÕES

Relatório sobre o seminário da disciplina de Cálculo Numérico, voltado para a análise dos métodos numéricos da Bissecção, Falsa Posição e Newton-Raphson, visando obtenção da primeira nota da primeira fase do semestre 2021.1.

Professor: Lucas Campos Freitas.

TIANGUÁ – CE

2021

RESUMO

Este relatório tem como objetivo realizar uma analise sobre o desempenho dos três métodos, apresentados na disciplina de Cálculo Numérico do Semestre 2021.1, que são usados para obtenção de raizes de funções. Os métodos analisados são o da Bissecção, Falsa Posição e o método de Newton-Raphson. Os métodos foram aplicados na função: $f(x) = sen(x) - x^3 - 2$.

Palavras-chaves: Cálculo Numérico, Bissecção, Falsa Posição, Newton-Raphson.

SUMÁRIO

1 INTRODUÇÃO……05
2 GRÁFICO DA FUNÇÃO……...06
3 DETERMINAÇÃO DO INTERVALO……..…07
4 IMPLEMENTAÇÃO DOS ALGORITMOS……..08
4.1 BISSECÇÃO……08
4.2 FALSA POSIÇÃO……...12
4.3 NEWTON-RAPHSON……17
5 COMPARAÇÃO DO DESEMPENHO DOS MÉTODOS……..21
5.1 TABELA……21
5.2 GRAFICOS DE CONVERGÊNCIA ……22
5.3 CONCLUSÃO……..23

1 – INTRODUÇÃO

Foi solicitado a nossa equipe a análise do desempenho dos métodos da Bissecção, Falsa Posição e o método de Newton-Raphson sobre as seguintes condições:

- **Funcão:** $f(x) = sen(x) – x^3 – 2$.
- **Erro:** 0.0001.

Ficando o trabalho dividido da seguinte maneira:

- Anna Zenaide Dias Alves: Newton-Raphson.
- Antonia Estefane Ribeiro Veras: Bissecção.
- Wanderson Gomes da Costa: Falsa Posição.

Além desse relatório escrito, também foi disponibilizado os códigos de implementação em um repositório no GitHub, que poderá ser acessado atraves do *link (https://github.com/WandersonGomes/SEMINARIO_CALCULO_NUMERICO)*.

Alem do códigos fontes também foram realizados vídeos explicando tanto a implementação como também a execução dos respectivos programas, que poderão ser acessados através dos seguintes links:

- Newton-Rapshon: https://www.youtube.com/watch?v=vIG8WmgCmY4
- Bissecção: https://youtu.be/vscqDhdfijo
- Falsa Posição: https://youtu.be/0ZZZOKvcbPQ
  https://youtu.be/3imZuTJj9_Q

2 – GRÁFICO DA FUNÇÃO

Para o desenho do gráfico da função foi utilizado o *software* GeoGebra na sua versão 5.0. Obtendo assim o seguinte resultado:

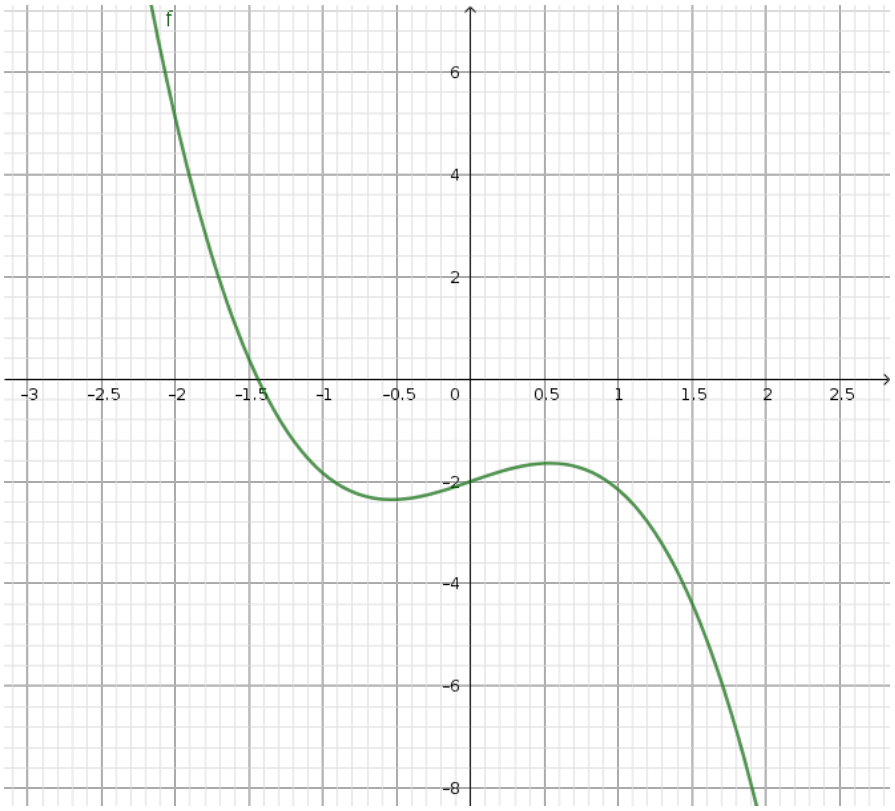


Figura 01 – Gráfico da Função

03 – DETERMINAÇÃO DO INTERVALO

O intervalo escolhido para fazer a análise foi o intervalo [-2,-1], pois como visto no gráfico da figura 01 apresenta no seu interior uma raiz.

Outra forma de se identificar a existência de uma raiz no intervalo é utilizar o teorema de Bolzano que afirma que se em um intervalo [a, b] tivermos f(a)f(b) < 0 teremos pelo menos uma raiz da função neste intervalo.

$$f(-2) = \sin(-2) - (-2)^3 - 2 = 5.965100503$$

$$f(-1) = \sin(-1) - (-1)^3 - 2 = -1.017452406$$

$$f(a)f(b) < 0$$

$$(5.965100503)*(-1.017452406) = -6.069205861$$

$$-6.069205861 < 0$$

04 – IMPLEMENTAÇÃO DOS ALGORITMOS

A seguir teremos a implementação dos algoritmos de cada método para que seja possível computar os dados que serão apresentados na seção 5 e comparados na seção 6.

4.1 BISSECÇÃO

A implementação desse método ficou sob a responsabilidade da aluna Antonia Estefane Ribeiro Veras que escolheu a implementação na linguagem de programação C.

Algoritmo:

1) Dados iniciais:

a) Intervalo inicial [a, b];

2) Aplicar a formula:

$$x = \frac{a+b}{2}$$

3) Separar os intervalos:

[a, x]

[x, b]

4) Se f(a) * f(x) < 0, então a raiz está entre o intervalo de [a, x]. Se não, está entre o intervalo de [x, b].

5) Criterio de parada:

a) c = b – a < l (amplitude final);

b) b – a <= precisao (erro);

c) contador >= maximo de tentativas.

6) Se algum dos critérios de paradas for satisfeito, então finalize. Se não, começa novamente do passo 2 utilizando o intervalo reduzido.

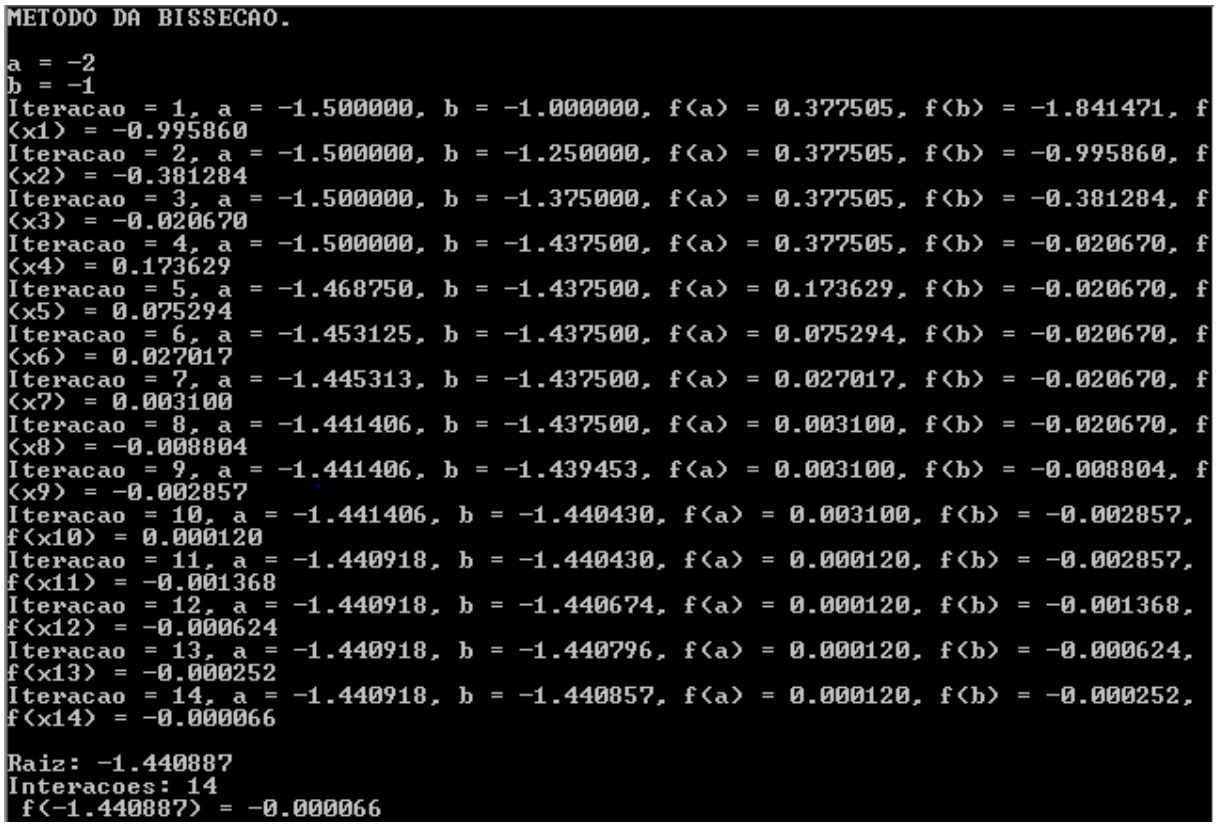
Código-Fonte:

```
/*
Aluna: Antonia Estefane Ribeiro Veras
Instituicao: IFCE – Campus Tiangua
Disciplina: Calculo Numerico
Professor: Lucas Campos Freitas
Semestre: 2021.1

Descricao: Implementacao do algoritmo do metodo da bisseccao.
*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double f(double x) {
    double y;
    y = sin(x) – pow(x,3) – 2;

    return y;
}

int main() {
    double a = 0.00, b = 0.00, c = 0.00, x = 0.00;
    int cont = 0;
    //Numero maximo de interacoes
    const int max = 100;
    //amplitude final, serve para criterio de parada
    const double I = 0.001;
```

```
    //precisao
    const double erro = 0.0001;

    printf("METODO DA BISSECAO.\n\n");

    printf("a = ");
    scanf("%lf", &a);

    printf("b = ");
    scanf("%lf", &b);

    c = b – a;
    x = (a + b)/2.0;

    while (fabs(f(x)) > erro || c > l) {
        if (f(a)*f(x) < 0.0) {
            b = x;
        }
        else{
            a = x;
        }
        c = b – a;
        x = (a + b)/2.0;
        cont++;

        if (cont >= max) {
            break;
        }
        printf("Iteracao = %d, a = %f, b = %f, f(a) = %f, f(b) = %f, f(x%d) = %f\n", cont,
a, b, f(a), f(b), cont, f(x));
    }
    printf("\nRaiz: %f\nInteracoes: %d\n f(%f) = %f\n", x, cont, x, f(x));
```

```
    return 0;
}
```

Execução do programa:



4.2 FALSA POSIÇÃO

A implementação desse método ficou de responsabilidade do aluno Wanderson Gomes da Costa que escolheu a implementação em linguagem de programação C.

Algoritmo:

1) Dados iniciais

a) intervalo inicial [a, b]

b) precisões **precisao1** e **precisao2**

2) Se (b – a) < precisao1, então escolha para raiz qualquer x pertencente ao intervalo [a, b]. FIM

Se |f(a)| < precisao2 ou se |f(b)| < precisao2 escolha a ou b como raiz. FIM.

3) k = 1

4) x = $\frac{af(b)-bf(a)}{f(b)-f(a)}$

5) Se f(a)*f(x) > 0, faça a = x. Vá para o passo 7.

6) b = x

7) Se |f(x)| < precisao2, escolha raiz = x. FIM.

8) Se (b – a) < precisao1, então escolha para raiz qualquer x pertencente ao intervalo (a, b). FIM

9) k = k + 1. Volte ao passo 4.

Código-Fonte:

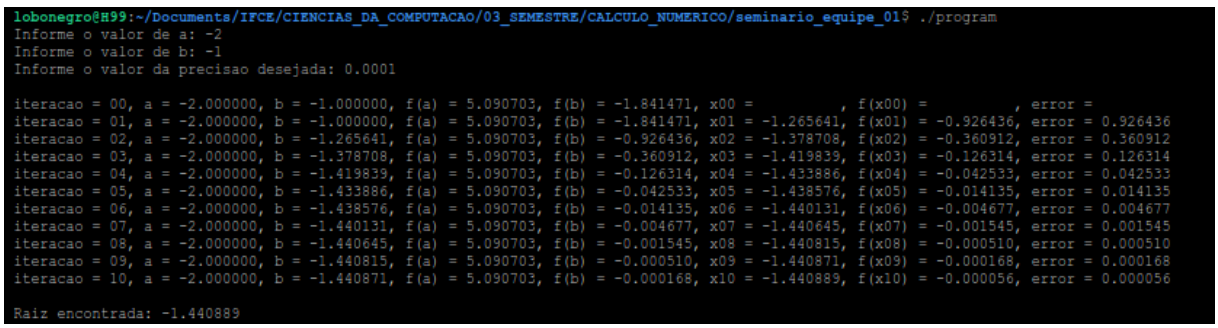
```
/*
Aluno: Wanderson Gomes da Costa
Instituicao: IFCE – Campus Tiangua
Disciplina: Calculo Numerico
Professor: Lucas Campos Freitas
Semestre: 2021.1

Descricao:
Implementacao do algoritmo do metodo da Falsa Posicao
*/
#include <stdio.h>
#include <math.h>

#define MAX_ITERACOES 100

//FUNCAO QUE RETORNA O MODULO DE UM NUMERO
double modulo(double numero) {
    return (numero < 0) ? (-1)*numero : numero;
}

//FUNCAO ANALISADA
double funcao(double x) {
    return sin(x) – (x*x*x) – 2;
}

//FUNCAO QUE IMPRIME A LINHA COM OS DADOS NECESSARIOS
void imprimeLinha(int iteracao, double a, double b, double x) {
    double erro = modulo(funcao(x));
    printf(“iteracao = %02d, a = %lf, b = %lf, f(a) = %lf, f(b) = %lf, x%02d = %lf, f(x
%02d) = %lf, error = %lf\n”, iteracao, a, b, funcao(a), funcao(b), iteracao, x, iteracao,
funcao(x), erro);
```

```
}

//FUNCAO QUE IMPLEMENTA O METODO DA FALSA POSICAO
double metodoFalsaPosicao(double a, double b, double precisao1, double precisao2, int max_iteracoes) {
    //variaveis auxiliares para o calculo
    int iteracao = 0;
    double M = 0.00, x = 0.00;

    //apresenta uma linha com os dados iniciais
    printf("iteracao = 00, a = %lf, b = %lf, f(a) = %lf, f(b) = %lf, x00 =          ,
f(x00) =          , error = \n", a, b, funcao(a), funcao(b));

    //passo 02
    if ((b – a) < precisao1) {
        return (a+b)/2;
    }

    if (modulo(funcao(a)) < precisao2) {
        return a;
    } else if (modulo(funcao(b)) < precisao2) {
        return b;
    }

    //passo 03
    iteracao = 1;

    while (iteracao <= max_iteracoes) {
        //passo 04
        x = (a*funcao(b) – b*funcao(a))/(funcao(b) – funcao(a));

        //apresenta os dados
```

```
            imprimeLinha(iteracao, a, b, x);

            //passo 05
            if ((funcao(a) * funcao(x) > 0) {
                a = x;
            } else {
                b = x;
            }

            //passo 07
            if (modulo(funcao(x)) < precisao2) {
                return x;
            }

            //passo08
            if ((b – a) < precisao1) {
                return (a + b)/2;
            }

            //passo09
            iteracao = iteracao + 1;
        }

        //caso ultrapasse o maximo de iteracoes
        return (a + b)/2;
}

//PROGRAMA PRINCIPAL
int main() {
        double a = 0.00, b = 0.00, precisao = 0.00;
        double raiz = 0.00;
```

```
    //solicita os dados para o calculo
    printf("Informe o valor de a: ");
    scanf("%lf", &a);
    printf("Informe o valor de b: ");
    scanf("%lf", &b);

    //solicita uma precisao
    do {
        printf("Informe o valor da precisao desejada: ");
        scanf("%lf", &precisao);
        if (precisao <= 0)
            printf("Error: valor invalido! Tente novamente.\n");
    } while (precisao <= 0);

    //chama o metodo para realizar os calculos
    printf("\n");
    raiz = metodoFalsaPosicao(a, b, precisao, precisao, MAX_ITERACOES);

    //apresenta a raiz encontrada
    printf("\nRaiz encontrada: %lf\n", raiz);

    return 0;
}
```

Execução do programa:



4.3 NEWTON-RAPHSON

Algoritmo:

Dada $f(x)=\sin(x)-x^3-2$, temos:

$\varphi(x)=x-\frac{f(x)}{f'(x)}$, isto é:

$\varphi(x)=x-\frac{\sin(x)-x^3-2}{\cos(x)-3x^2}$

1) Dados iniciais:

a) Aproximação inicial: x0

b) precisões: ε_1 e ε_2

2) Se $|f(x_0)| < \varepsilon_1$, faça $\bar{x} = x_0$. FIM.

3) k = 1

4) $\varphi(x_1)=x_0-\frac{f(x_0)}{f'(x_0)}$

5) Se $|f(x_1)| < \varepsilon_1$, faça $\bar{x} = x_1$. FIM.

Ou se $|x_1 - x_0| < \varepsilon_2$, faça $\bar{x} = x_1$. FIM.

6) $x_0 = x_1$

7) k = k+1. Volte ao passo 4.

Código-Fonte:

```
/*
Aluno: Anna Zenaide Dias Alves
Instituicao: IFCE – Campus Tiangua
Disciplina: Calculo Numerico
```

Professor: Lucas Campos Freitas
Semestre: 2021.1

Descricao:
Implementacao do algoritmo do metodo de Newton-Raphson
*/

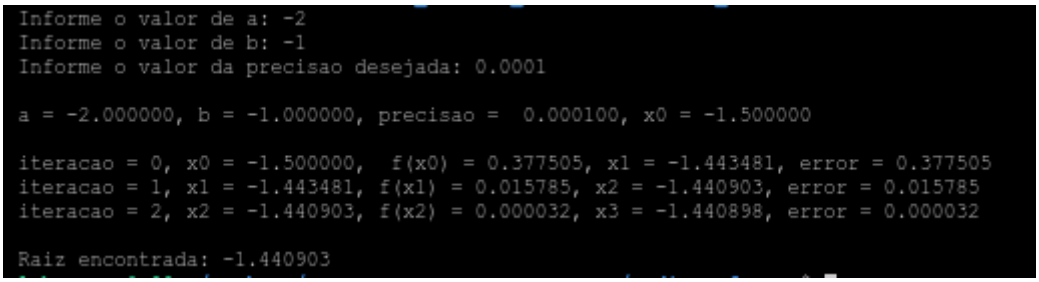
```
#include <stdio.h>
#include <math.h>

#define MAX_ITERACOES 100

double funcao(double);
double derivada(double);
double Newton(double, double, double, double, int);
void imprimeLinha(int, double, double, double);

int main() {
    //Programa principal
    double a, b, precisao;
    double raiz;

    //solicita os dados para o calculo
    printf("Informe o valor de a: ");
    scanf("%lf", &a);
    printf("Informe o valor de b: ");
    scanf("%lf", &b);

    //solicita uma precisao
    do {
        printf("Informe o valor da precisao desejada: ");
        scanf("%lf", &precisao);
        if (precisao <= 0)
```

```
            printf("Error: valor invalido! Tente novamente.\n");
    } while (precisao <= 0);

    //chama o metodo para realizar os calculos
    printf("\n");
    raiz = Newton(a, b, precisao, precisao, MAX_ITERACOES);

    //apresenta a raiz encontrada
    printf("\nRaiz encontrada: %lf\n", raiz);
    return 0;
}

double funcao(double x) {
    //Funcao Analisada;
    return sin(x) – (x*x*x) – 2;
}

double derivada(double x) {
    //Derivada da funcao analisada
    return cos(x) – (3*x*x);
}

void imprimeLinha(int iteracao, double x0, double x1, double erro) {
    //Imprime uma linha com dados em cada iteracao
    printf("iteracao = %d, x%d = %lf, f(x%d) = %lf, x%d = %lf, error = %lf\n", iteracao,
iteracao, x0, iteracao, funcao(x0), iteracao + 1, x1, erro);
 }

double Newton(double a, double b, double precisao1, double precisao2, int
max_iteracacoes) {
    //Implementacao do metodo de Newton-Raphson
    //variaveis auxiliares para o calculo
```

```
    int iteracao = 0;
    double x0, x1;
    x0 = (double)(a + b)/2.0;
    printf("a = %lf, b = %lf, precisao = %lf, x0 = %lf\n\n", a, b, precisao1, x0);
    //apresenta uma linha com os dados iniciais
    printf("iteracao = %d, x0 = %lf, f(x0) = %lf, x%d = %lf, error = %lf\n", iteracao, x0,
funcao(x0), iteracao+1, x0 – funcao(x0)/derivada(x0), fabs(funcao(x0)));

    //passo 02
    if (fabs(funcao(x0)) < precisao1) {
        return x0;
    }

    //passo 03
    iteracao = 1;

    while (iteracao <= max_iteracoes) {
        //passo 04
        x1 = x0 – (funcao(x0)/derivada(x0));

        //apresenta os dados
        imprimeLinha(iteracao, x1, x1 – funcao(x1)/derivada(x1), fabs(funcao(x1)));

        //passo 05
        if ((fabs(funcao(x1)) < precisao1) || (fabs(x1 – x0) < precisao)) {
            return x1;
        }
        //passo 6
        x0 = x1;
        //passo 07
        ++iteracao;
    }
```

```
    //caso ultrapassse o maximo de iteracoes
    return x1;
}
```

Execução do Programa:



5 COMPARAÇÃO DO DESEMPENHO DOS MÉTODOS

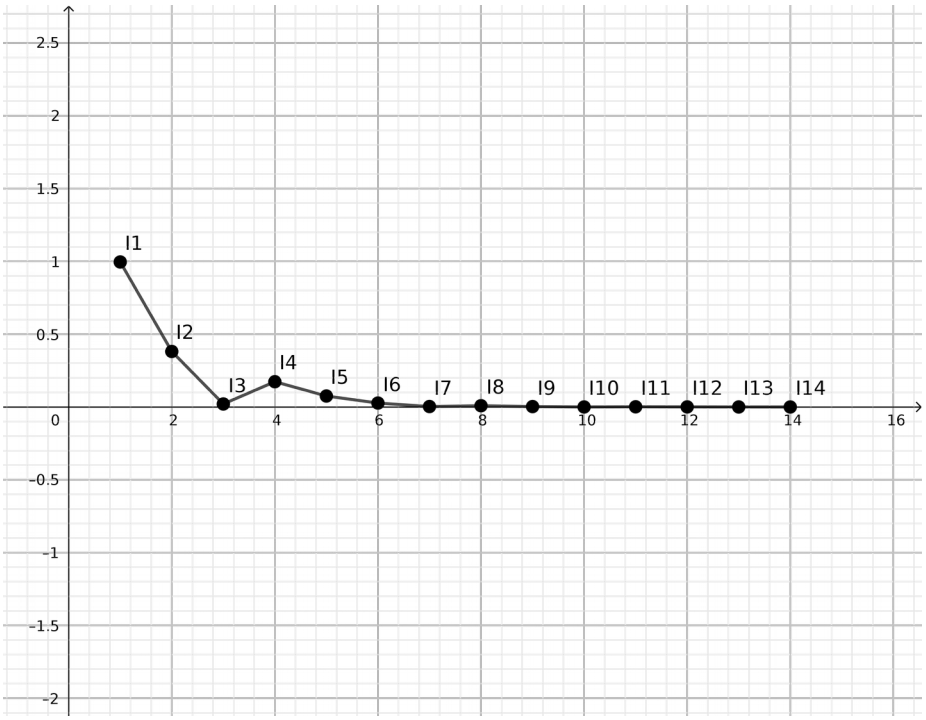
Nesta secção vamos fazer uma avaliação sobre os dados que obtemos com os métodos numéricos e dar um parecer sobre qual método foi mais eficiente no caso estudado.

5.1 TABELA

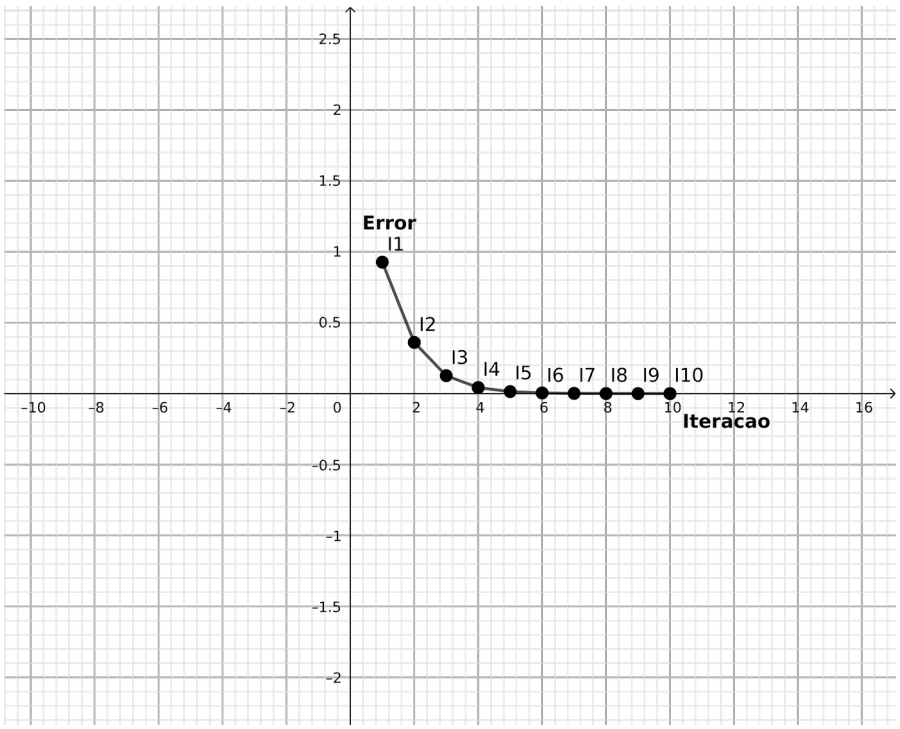
MÉTODO	INTERAÇÕES	ERRO	RAIZ	VALOR DA RAIZ
BISSECÇÃO	14	0.000066	-1.440889	-0.000066
FALSA POSIÇÃO	10	0.000056	-1.440889	-0.000056
NEWTON-RAPHSON	03	0.000032	-1.440903	0.000032

5.2 GRAFICOS DE CONVERGÊNCIA

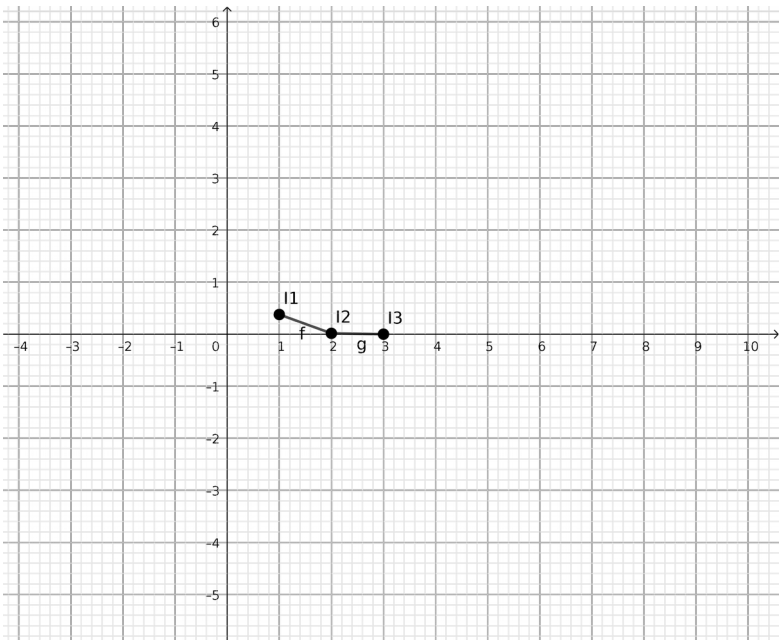
BISSSECÇÃO:



FALSA POSICÃO:



NEWTON-RAPHSON:



5.3 CONCLUSÃO

Pelos dados obtidos verificamos que o algoritmo mais eficiente para o caso estudado foi o de Newton que teve apenas 3 iteracoes para encontrar a raiz da função. E também podemos observar que para o caso estudado o método da Falsa Posição foi mais eficiente do que o método da Bissecção.