MS211 - Cálculo Numérico

Turma K - Prof João Batista Florindo

Projeto 1

Artur Lima (166916)

João Pedro Martins (176117)

Introdução

O objetivo deste projeto é analisar e comparar a performance computacional de dois métodos numéricos iterativos de refinamento de raiz de funções, isto é, métodos que recebem como *input* uma estimativa inicial (podendo ser um ponto ou um intervalo) de raiz para uma determinada função *f* e um conjunto de precisões e, após um determinado número de iterações, produzem como *output* uma raiz aproximada de acordo com as precisões fornecidas.

Os métodos estudados neste projeto são o método da bissecção e o método de Newton-Raphson, e a função aplicada a eles é a Equação de Butler-Volmer com $\alpha = 0.2$ e $\beta = 2$.

.

Método da Bissecção

O método da bissecção consiste na busca pela raiz no intervalo de 2 pontos, x_0 e x_1, tais que dada uma função f, $f(x_0) * f(x_1) < 0$ para encontrar a raiz desejada. Nesse caso, temos uma divisão sucessiva do intervalo $[x_0, x_1]$ ao meio, até que consigamos que $x_1 - x_0 < \varepsilon$, onde ε é a precisão desejada.

Esse método é um dos mais simples de se desenvolver, já que seu algoritmo se baseia na busca binária. Outra vantagem é a garantia de que o método irá convergir, porém ele não pode ser usado para encontrar múltiplas raízes, o que o faz perder importância com relação a outros algoritmos que conseguem tal feito.

Como dito antes, o algoritmo do método é bem simples. Ele recebe como entrada dois pontos (x_0 e x_1) e uma precisão E. A cada iteração, fazemos a divisão $x = (x_0 + x_1)/2$ e comparamos esse valor com a precisão E. Se for menor, paramos e retornamos x. Caso contrário, verificamos se $f(x_0) * f(x) < 0$. Se for, fazemos $x_1 = x$ ou, caso contrário, $x_0 = x$. Logo após isso, retornaremos para o início da iteração com os valores atuais de x_0 e x_1 para uma nova divisão, até que a divisão seja menor que o valor E da precisão.

O tempo de execução total do código ocorre em tempo O(n), já que a iteração do código consegue fazer em tempo constante as operações que ocorrem dentro dela, portanto podemos considerar que esse código roda em tempo linear.

Método de Newton-Raphson

O método de Newton é um método numérico iterativo muito eficiente para refinamento de raiz. É derivado de outro método chamado método do ponto fixo que faz o refinamento de uma raiz x de uma função f por meio de aplicações sucessivas de uma função g chamada "função de iteração", que recebe esse nome porque a cada iteração a aproximação x_{k+1} é obtida da iteração anterior tomando-se $x_{k+1} = g(x_k)$.

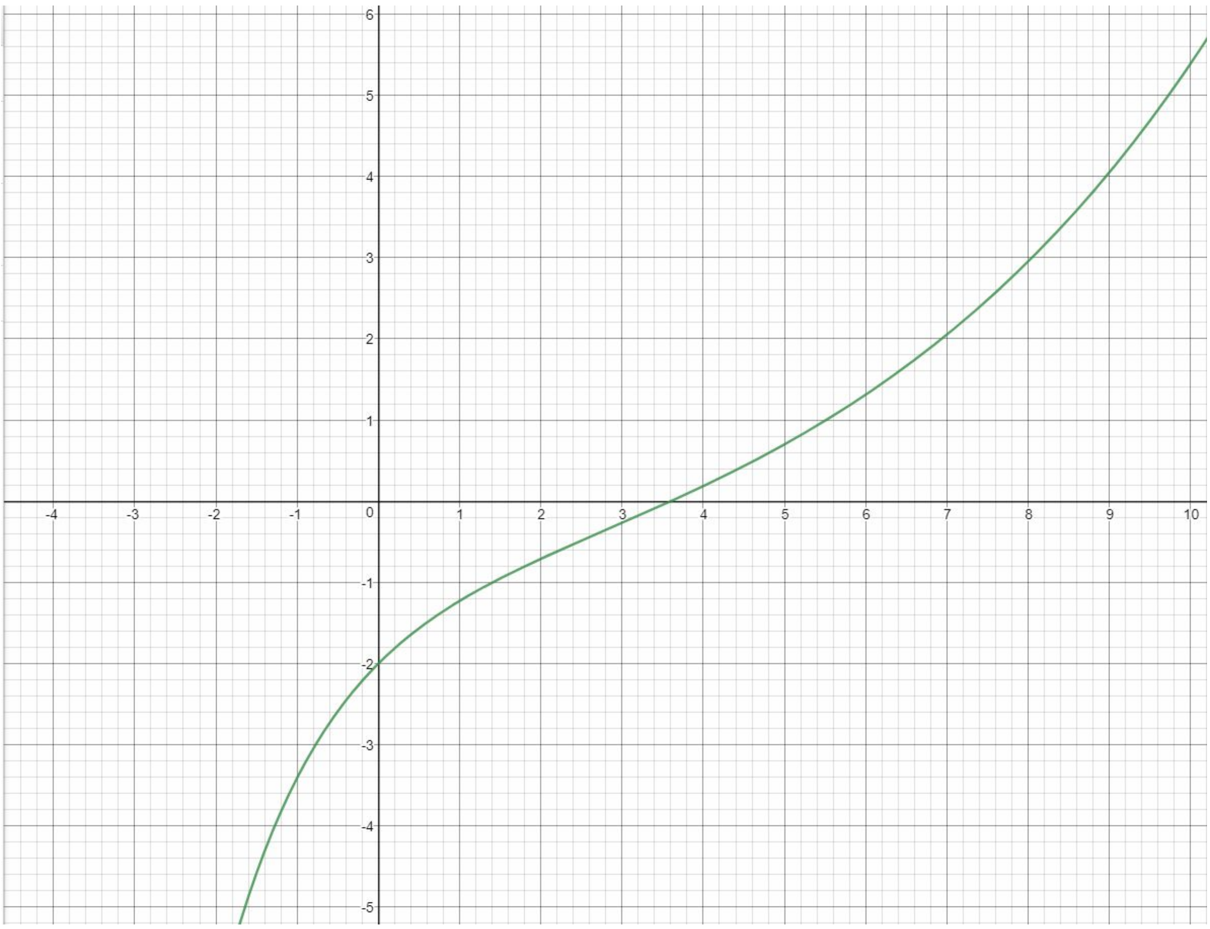
Porém, o que diferencia o método de Newton com o método do ponto fixo é a maneira de como o método de newton escolhe a função de iteração g: g é escolhida de tal forma que $g(\xi) = 0$, onde ξ é a raiz de f. Essa escolha transforma a convergência apenas linear ($p = 1$) do método do ponto fixo em uma convergência pelo menos quadrática ($p = 2$).

Como $g(x)$ é da forma $x + A(x)f(x)$, tomando $g(x) = 0$ obtemos que $g(x) = x - f(x) / f`(x)$ e portanto a relação $x_{k+1} = x_k - f(x_k) / f'(x_k)$ representa a sequência obtida pelo método.

O algoritmo do método de Newton é bem simples. Ele recebe como entrada uma aproximação inicial x_0 e precisões ε_1 e ε_2. A cada iteração, a aproximação x_{k+1} é obtida da aproximação anterior pela fórmula $x_{k+1} = x_k - f(x_k) / f'(x_k)$. No fim de cada iteração, é verificado se pelo menos uma das precisões desejadas foi atingida: é verificado se $|f(x_{k+1})| < \varepsilon_1$ ou $|x_1 - x_0| < \varepsilon_2$. Em caso positivo, o algoritmo encerra devolvendo x_{k+1} como a melhor aproximação encontrada; em caso negativo, passa-se para a próxima iteração. Cada iteração do algoritmo executa em tempo constante, isto é, em $O(1)$, quando se desconsidera o tempo para calcular os valores de $f(x)$ e $f'(x)$ assumindo que esses valores estejam tabelados de alguma forma, obtendo-se então um tempo de execução da ordem $O(n)$ para n iterações; se os valores de $f(x)$ e $f'(x)$ precisarem ser calculados a cada iteração, o tempo de execução passa então a depender do cálculo de $f(x)$ e $f'(x)$.

Item a

Antes de aplicar os métodos de refinamento de raiz, precisamos isolar a raiz da Equação de Butler-Volmer, obtendo assim uma estimativa inicial em forma de intervalo que contém a raiz. Utilizando uma ferramenta online (*desmos.com*) para plotar o gráfico da equação de Butler-Volmer com α = 0.2 e β = 2, obtivemos o seguinte gráfico:



Equação de Butler-Volmer com α = 0.2 e β = 2, com eixo f(x) como eixo vertical e eixo x como eixo horizontal .

Pelo gráfico é possível observar que a função plotada tem uma raiz no intervalo *[3, 4]*, mais precisamente próximo ao ponto $x = 3.6$, onde $f(x) = 0$.

Item b

Implementação dos métodos da Bissecção e de Newton-Raphson na linguagem de programação C, seguindo os algoritmos presentes no material didático:

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

#define MAX_ITERATION 3 // limite de iteracoes para o metodo

// calcula Butler-Volmer em x com a = 0.2 e b = 2
double f(double x){
    double a = 0.2;
    double b = 2;

    return (exp(a*x) - exp((a-1)*x) - b);
}

// metodo da bisseccao no intervalo [a, b] e precisao e na funcao f
double bisection(double a, double b, double e){
    int k = 1;
    double x = a;
    while((b-a) >= e && k++ <= MAX_ITERATION){
        x = (a+b)/2;
        if(f(a)*f(x) < 0)
            b = x;
        else
            a = x;
    }
    return x;
}

// testador
int main (){
    double a, b, e, x;
    scanf("%lf %lf %lf", &a, &b, &e);
    x = bisection(a, b, e);
    printf("%lf\n", x);
    return 0;
}
```

Método da Bissecção em C.

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

#define MAX_ITERATION 3 // limite de iteracoes para o metodo

// calcula Butler-Volmer em x com a = 0.2 e b = 2
double f(double x){
    double b = 2;
    double a = 0.2;

    return (exp(a*x) - exp((a-1)*x) - b);
}

// calcula derivada de Butler-Volmer em x com a = 0.2
double df(double x){
    double a = 0.2;

    return (exp((a-1)*x)*(a*(exp(x)-1)+1));
}

// metodo de newton-raphson no ponto inicial x e precisoes e1 e e2
double newton(double x, double e1, double e2){
    int k = 1;
    double x0, x1;

    x0 = x;

    if(fabs(f(x0)) < e1)
        return x0;

    while(k <= MAX_ITERATION){
        x1 = x0 - f(x0)/df(x0);
        if(fabs(f(x0)) < e1 || fabs(x1-x0) < e2)
            return x1;
        x0 = x1;
        k++;
    }
    return x0;
}

// testador
int main (){
    double x0, x, e1, e2;
    scanf("%lf %lf %lf", &x0, &e1, &e2);
    x = newton(x0, e1, e2);
    printf("%lf\n", x);
    return 0;
}
```

Método de Newton-Raphson em C.

Item c

Utilizando o tipo *double* em C, sendo impresso com dez casas decimais de precisão, após um número *k* de iterações os resultados de ambos os algoritmos convergem para o número $\bar{x}$ = *3.6037324492*, que representa a melhor aproximação com dez casas decimais de precisão da raiz da função estudada.

Utilizando esse valor como referência, executamos ambos os métodos estudados com valores iniciais e número de iterações variados e precisões de entrada setadas como 0. Com isso pudemos comparar seus resultados e evidenciar suas diferentes taxas de convergências.

Os resultados obtidos são apresentados nas tabelas abaixo:

Método da Bissecção		
[a, b]	*x*	iterações
[3, 4]	3.5	1
[3, 4]	3.75	2
[3, 4]	3.625	3
[3, 4]	3.5625	4
[3, 4]	3.6037324492	37
[2, 5]	3.5	1
[2, 5]	4.25	2
[2, 5]	3.875	3
[2, 5]	3.6875	4
[2, 5]	3.6037324492	37
[1, 6]	3.5	1
[1, 6]	4.75	2
[1, 6]	4.125	3
[1, 6]	3.8125	4
[1, 6]	3.6037324492	37
[0, 7]	3.5	1
[0, 7]	5.25	2
[0, 7]	4.375	3
[0, 7]	3.9375	4
[0, 7]	3.6037324492	37

Legenda:

[a,b] = intervalo inicial;

x = *output;*

iterações = número de iterações executadas;

precisão ε de entrada setada como 0.

Método de Newton		
x_0	x	iterações
3	3.6146460109	1
3	3.6037385479	2
3	3.6037324492	3
3	3.6037324492	4
3	3.6037324492	5
2	3.5440299527	1
2	3.6039078334	2
2	3.6037324508	3
2	3.6037324492	4
2	3.6037324492	5
1	3.0338533708	1
1	3.6139133582	2
1	3.6037377545	3
1	3.6037324492	4
1	3.6037324492	5
0	2	1
0	3.5440299527	2
0	3.6039078334	3
0	3.6037324508	4
0	3.6037324492	5

Legenda:

x_0 = ponto inicial;

x = *output;*

iterações = número de iterações executadas;

precisões ε_1 e ε_2 de entrada ambas setadas como 0.

Conclusão

É possível observar pelas tabelas que o método de Newton atinge o resultado $\bar{x}$ esperado para dez casas decimais em muito menos iterações do que o método da bissecção.

Como sua taxa de convergência é pelo menos quadrática, podemos notar que a cada iteração a quantidade de dígitos corretos produzidos pelo método de Newton praticamente duplica.

Enquanto o método da bissecção precisou de exatas 37 iterações para atingir o valor esperado $\bar{x}$ para o maior intervalo inicial testado, o método de Newton precisou de apenas 5 iterações para atingir esse valor para o ponto inicial mais distante da raiz testado.

Isso evidencia as diferentes taxas de convergências entre esses dois métodos. O método da bissecção possui uma taxa de convergência apenas linear, mas é mais simples de ser implementado e sempre converge para a raiz, se esta existir. O método de Newton, por outro lado, possui taxa de convergência pelo menos quadrática, mas requer uma boa estimativa inicial e requer o cálculo de derivadas, o que pode ser custoso.

Bibliografia

RUGGIERO, M.; LOPES, V. Cálculo Numérico, Aspectos Teóricos e Computacionais: 2. ed.