UNIVERSIDADE FEDERAL DO ESPIRITO SANTO

CIÊNCIA DA COMPUTAÇÃO

Jéssica Faria dos Santos

RELATÓRIO DO EP2

PROGRAMAÇÃO FUNCIONAL

SÃO MATEUS
2018

Jéssica Faria dos Santos

RELATÓRIO DO EP2

PROGRAMAÇÃO FUNCIONAL

Relatório do exercício de programação 2 sobre Raízes de equações apresentado ao professor Oberlan Romão, da disciplina de Programação Funcional, como requisito para obtenção de nota da Universidade Federal do Espírito Santo – Campus São Mateus.

PROFESSOR: **Oberlan Romão**

SÃO MATEUS
2018

SUMÁRIO

1. Justificativa

Este trabalho possui como intuito analisar três tipos de método, suas vantagens e desvantagens

2. Objetivos

- ✓ Definir três funções lambda;
- ✓ Definir três derivadas das funções escolhidas (lambda);
- ✓ Implementar as funções escolhidas utilizando o Método da Bisseção;
- ✓ Implementar as funções escolhidas utilizando o Método de Newton-Raphson;
- ✓ Implementar as funções escolhidas utilizando o Método da Secante;
- ✓ Calcular as raízes das funções escolhidas, através dos métodos propostos;
- ✓ Calcular o numero de interações das funções escolhidas, através dos métodos propostos;
- ✓ Calcular o erro das funções escolhidas, através dos métodos propostos;

3. Introdução

Com o intuito de encontrar as raízes, os erros e o numero de interações foi projetado um código em python, o mesmo recebe uma função (“lambda”), a derivada desta função (“lambda”) e os limites propostos (“a” e “b”). Através destes parâmetros o código utilizara três métodos para chegar na resposta desejada.

4. Metodologia

4.1 Método da Bisseção

O método da bisseção explora o fato de que uma função contínua f no intervalo fechado [a,b] com f(a) f(b) < 0 tem uma raiz no intervalo (a,b) (Teorema de Bolzano).

```
#Ela recebe como parametros uma função, e o intervalo que se inicia em a e termina no b.
def bissecao(f, a,b,epsilon=10**-15,maxIt = 50,i=1,Lr=[],Li=[]):
    xl=(a+b)/2 #aproxima da raiz da função
    erro=abs(a-b)/2#erro relativo
    print("{0:2d} [{1:.15f}] [{2:.15f}]".format(i, xl, erro))
    "imprime os valores enquanto xl se aproxima da raiz da função"
    if f(xl)==0 or erro<epsilon or i>maxIt:
        return Lr,Li #A função retorna uma lista com as aproximações da raiz e outra com o numero de iterações
    elif f(a)*f(xl)<0:
        return bissecao(f, a,xl,epsilon, maxIt,i+1,Lr+[xl],Li+[i])#Adiciona recursivamente os valores na lista
    else:
        return bissecao(f, xl,b,epsilon, maxIt,i+1,Lr+[xl],Li+[i])#Adiciona recursivamente os valores na lista
```

4.2 Método de Newton-Raphson

O método de Newton-Raphson usa a inclinação (tangente) da função f(x) na solução iterativa atual (xi) para encontrar a solução (xi+1) na próxima iteração.

```
"A função raphson implementa método de Newton-Raphson usa a inclinaçãoda reta tangente dafunção f1 na"
"solução iterativa atual(xi) para encontrar a solução(xi+1) na próxima iteração. "
#Ela recebe como parametro a funçao (f1), sua derivada (f2) e um valor do intervalo [a,b]
def raphson(f1,f2,x1,i=1,epsilon=10**-15,maxIt=50,Lr=[],Li=[]):
    xi=x1-(f1(x1)/f2(x1))#aproxima da raiz da função
    erro=abs((xi-x1)/xi)#erro relativo
    print("{0:2d} [{1:.15f}] [{2:.15f}]".format(i,xi,erro))
    if f1(x1)==0 or f2(x1)==0 or xi==0 or erro<epsilon or i>maxIt:
        return Lr,Li #A função retorna uma lista com as aproximações da raiz e outra com o numero de iterações
    else:
        return raphson(f1,f2,xi,i+1,epsilon,maxIt,Lr+[],Li+[])#Adiciona recursivamente os valores na lista
```

4.3 Método da Secante

O método da Secante é uma variação do método de Newton-Raphson, evitando a necessidade de conhecer-se a derivada analítica de f(x).

```
"O método da Secante é uma variação do método de Newton-Raphson, evitando a necessidade de conhecer a derivada analitica de f1."
#Os parametros são a função f1,e x1 e x2, valores do intervalo [a,b]
def secante(f1,x1,x2,i=1,epsilon=10**-15,maxIt=50,Lr=[],Li=[]):
    xi = x2 - f1(x2)*(x2 - x1)/(f1(x2) - f1(x1)) #aproxima da raiz da função
    erro=abs((xi-x2)/xi) #erro relativo
    print("{0:2d} [{1:.15f}] [{2:.15f}]".format(i, xi, erro))
    if f1(xi)==0 or erro<epsilon or i>maxIt:
        return Lr,Li #A função retorna uma lista com as aproximações da raiz e outra com o numero de iterações
    else:
        return secante(f1,x2,xi,i+1,epsilon, maxIt,Lr+[xi],Li+[i])#Adiciona recursivamente os valores na lista
```

4.4 Jupyter Notebook

Um notebook é um documento produzido pelo Jupyter Notebook App, que contém tanto o código de alguma linguagem de programação quanto os elementos de texto. Os notebooks são documentos legíveis por humanos, contendo uma descrição da análise e os resultados tanto em código como em figuras. Com o Jupyter foi possível criar pasta para o projeto, criar um ambiente virtual e por meio dele executar o programa já feito em python uma de suas grandes vantagens é que ele proporciona o programador a não precisar ficar alterando entre um editor de texto e o interpretador.

5. Resultados e Discussão

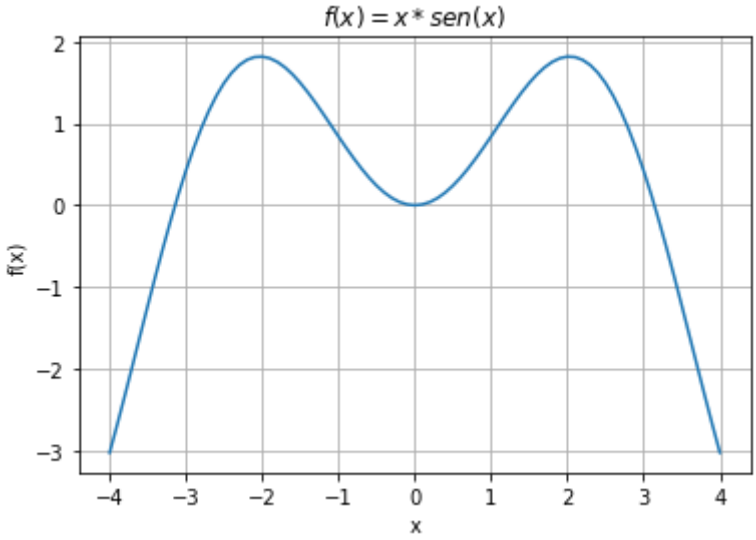
Foram escolhidas três funções (“f(x)”), juntamente de suas derivadas (“f ’(x)”), e escolhido um intervalo de [a,b] para que o gráfico pudesse ser gerado. Através destes parâmetros pode ser calculado através dos métodos de Bisseção, Newton – Raphson e Secante as raízes e com eles podemos analisar o número de iterações, a precisão e a rapidez de cada método.

5.1 Exemplo 1

- Função escolhida:
  f(x) = x.sen(x)

- Derivada da função:
  f'(x) = sen (x)+ x.cos(x)

- Intervalos:
  a = 2
  b = 4



5.1.1 Método da Bisseção – Exemplo 1

Esse método tem a implementação mais simples que os outros, mas a quantidade de iterações é bem maior, ou seja, o tempo gasto para chegar ao resultado é mais extenso, como na função exemplificada. O número de iterações ultrapassa 50.

```
==> METÓDO DA BISSEÇÃO: x1=2 e x2=4
-----------------------------------------------
i            raiz              erro
-----------------------------------------------
 1 [3.000000000000000] [1.000000000000000]
 2 [3.500000000000000] [0.500000000000000]
 3 [3.250000000000000] [0.250000000000000]
 4 [3.125000000000000] [0.125000000000000]
 5 [3.187500000000000] [0.062500000000000]
 6 [3.156250000000000] [0.031250000000000]
 7 [3.140625000000000] [0.015625000000000]
 8 [3.148437500000000] [0.007812500000000]
 9 [3.144531250000000] [0.003906250000000]
10 [3.142578125000000] [0.001953125000000]
11 [3.141601562500000] [0.000976562500000]
12 [3.141113281250000] [0.000488281250000]
13 [3.141357421875000] [0.000244140625000]
14 [3.141479492187500] [0.000122070312500]
15 [3.141540527343750] [0.000061035156250]
16 [3.141571044921875] [0.000030517578125]
17 [3.141586303710938] [0.000015258789062]
18 [3.141593933105469] [0.000007629394531]
19 [3.141590118408203] [0.000003814697266]
20 [3.141592025756836] [0.000001907348633]
21 [3.141592979431152] [0.000000953674316]
22 [3.141592502593994] [0.000000476837158]
23 [3.141592741012573] [0.000000238418579]
24 [3.141592621803284] [0.000000119209290]
25 [3.141592681407928] [0.000000059604645]
26 [3.141592651605606] [0.000000029802322]
27 [3.141592666506767] [0.000000014901161]
28 [3.141592659056187] [0.000000007450581]
29 [3.141592655330896] [0.000000003725290]
30 [3.141592653468251] [0.000000001862645]
31 [3.141592654399574] [0.000000000931323]
32 [3.141592653933913] [0.000000000465661]
33 [3.141592653701082] [0.000000000232831]
34 [3.141592653584667] [0.000000000116415]
35 [3.141592653642874] [0.000000000058208]
36 [3.141592653613770] [0.000000000029104]
37 [3.141592653599218] [0.000000000014552]
   .                .                .
38 [3.141592653591943] [0.000000000007276]
39 [3.141592653588305] [0.000000000003638]
40 [3.141592653590124] [0.000000000001819]
41 [3.141592653589214] [0.000000000000909]
42 [3.141592653589669] [0.000000000000455]
43 [3.141592653589896] [0.000000000000227]
44 [3.141592653589782] [0.000000000000114]
45 [3.141592653589839] [0.000000000000057]
46 [3.141592653589811] [0.000000000000028]
47 [3.141592653589797] [0.000000000000014]
48 [3.141592653589790] [0.000000000000007]
49 [3.141592653589793] [0.000000000000004]
50 [3.141592653589795] [0.000000000000002]
51 [3.141592653589794] [0.000000000000001]
```

5.1.2 Método de Newton-Raphson – Exemplo 1

O método de Raphson utiliza além da função, sua derivada (inclinação da reta tangente). Para a função *3.cos(x)+2* é o método que chega mais rápido ao resultado e com a menor quantidade de erros.

```
==> METÓDO DE NEWTON-RAPHSON: x1=4
-----------------------------------------------
i             raiz                erro
-----------------------------------------------
 1 [3.102084993668375] [0.289455320587394]
 2 [3.142123410035418] [0.012742470979710]
 3 [3.141592743178344] [0.000168916502060]
 4 [3.141592653589796] [0.000000028516921]
 5 [3.141592653589793] [0.000000000000001]
```

5.1.3 Método da Secante – Exemplo 1

O método da secante também obtém um resultado mais rápido. E em algumas funções sua precisão ultrapassa a do método Newton-Raphson. Na função escolhida ela tem o número de iterações maior que o método anterior.

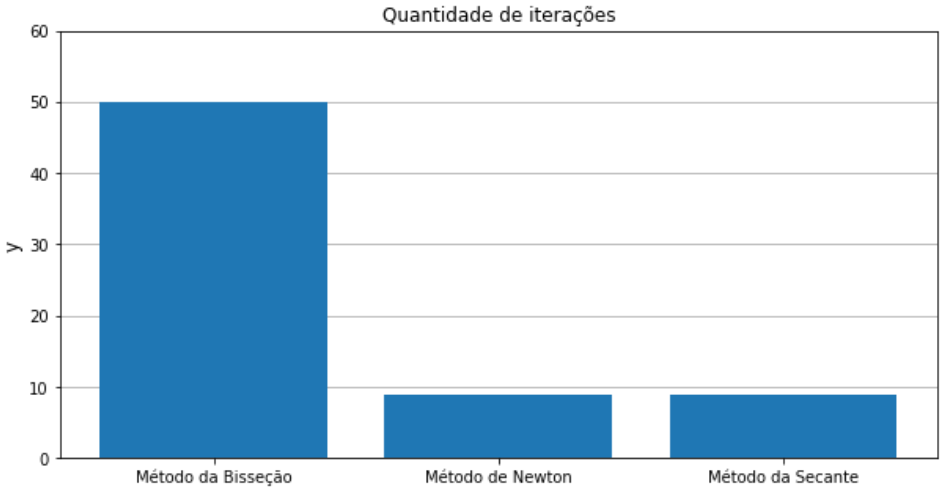
```
==> METÓDO DA SECANTE: x1=2 e x2=4
-----------------------------------------------
i             raiz                erro
-----------------------------------------------
 1 [2.750585265242216] [0.454235958632521]
 2 [3.071959576342204] [0.104615410168466]
 3 [3.154265415144058] [0.026093504499239]
 4 [3.141298041256154] [0.004128030424874]
 5 [3.141591477499772] [0.000093403692275]
 6 [3.141592653700112] [0.000000374396196]
 7 [3.141592653589793] [0.000000000035116]
 8 [3.141592653589793] [0.000000000000000]
```

5.1.4 Gráficos – Exemplo 1

No gráfico podemos observar a rapidez com que cada método se aproxima da raiz da função.

5.1.5 Vantagens e Desvantagens – Exemplo 1

Com a função exemplo número um pode se observar que o melhor método em termo de rapidez foi o método de Newton-Raphson, mas esse método não possui a precisão que o método da bisseção possui. Porem método da Bisseção possui uma desvantagem o tempo de geração do resultado, gerando assim muitas iterações o que o torna mais lento que os demais. Como podemos observar no gráfico a seguir.

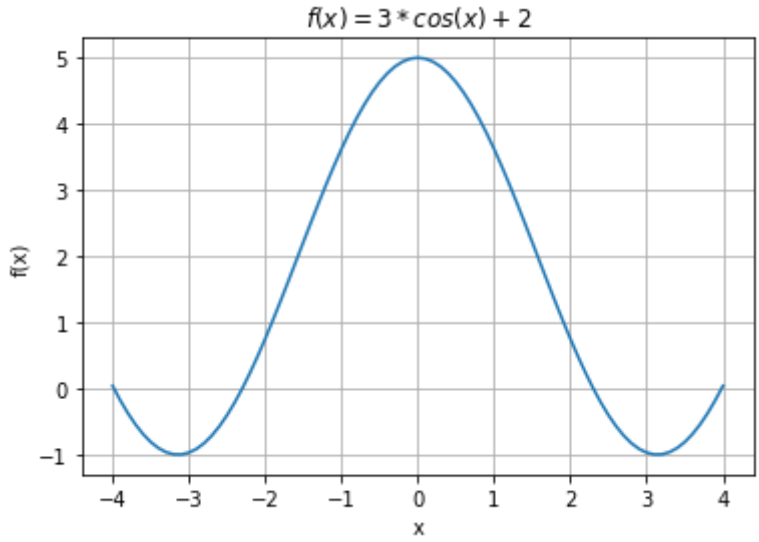


5.2 Exemplo 2

- Função escolhida:
  f(x) = 3.cos(x)+2

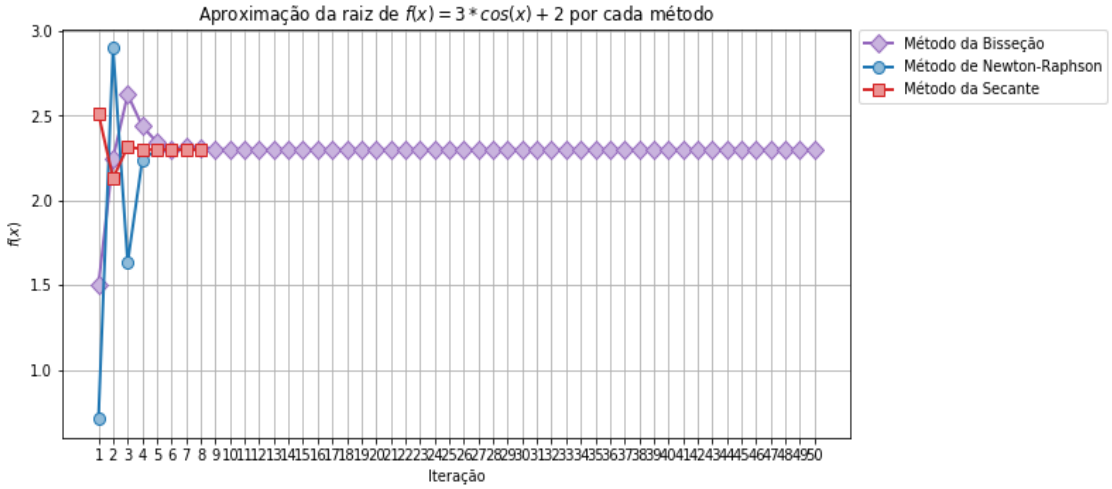
- Derivada da função:
  f'(x) = 3.(-sen(x))

- Intervalos:
- a = 0
- b = 3



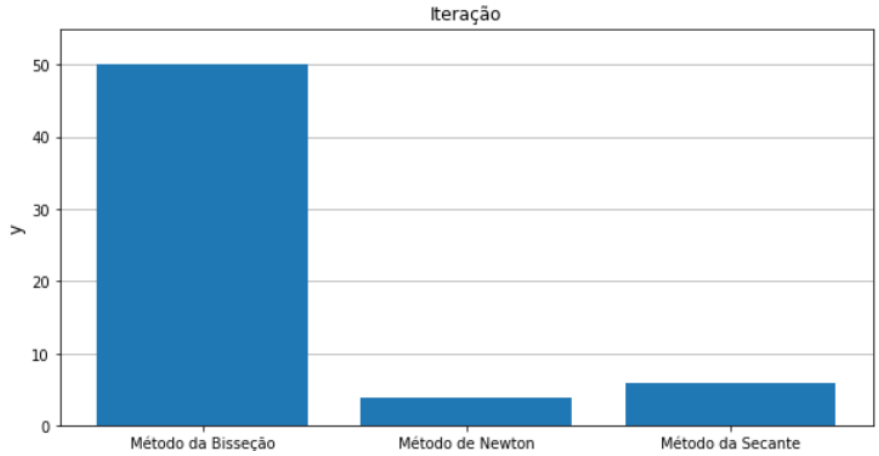
5.2.1 Gráficos – Exemplo 2

O gráfico mostra como no da função anterior a rapidez com que cada método se aproxima da raiz sendo o de Newton e o da Secante os mais rápidos.



5.2.5 Vantagens e Desvantagens – Exemplo 2

Nessa função o método mais rápido é o da Secante o que a difere da anterior. O método da Bisseção tendo mais iterações que os outros. E tendo o método da Secante como o segundo método mais rápido.

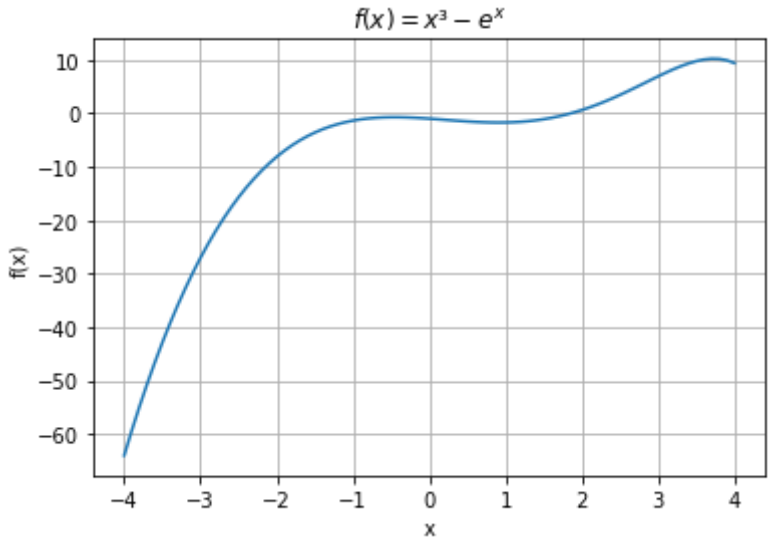


5.3 Exemplo 3

- Função escolhida:
  f(x) = x^3-e^x

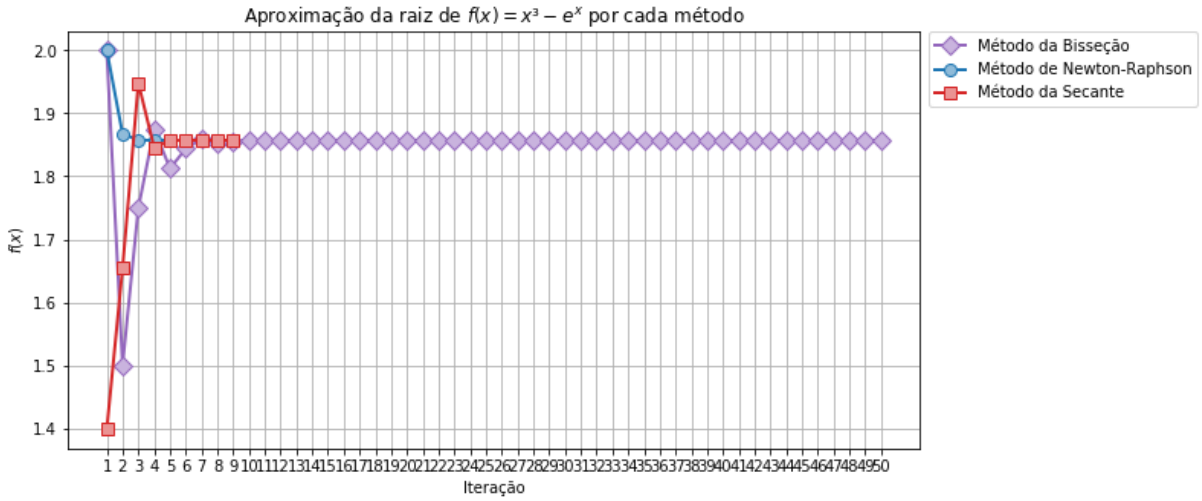
- Derivada da função:
  f'(x) = 3x^2-e^x

- Intervalos:
  a = 1
  b = 3



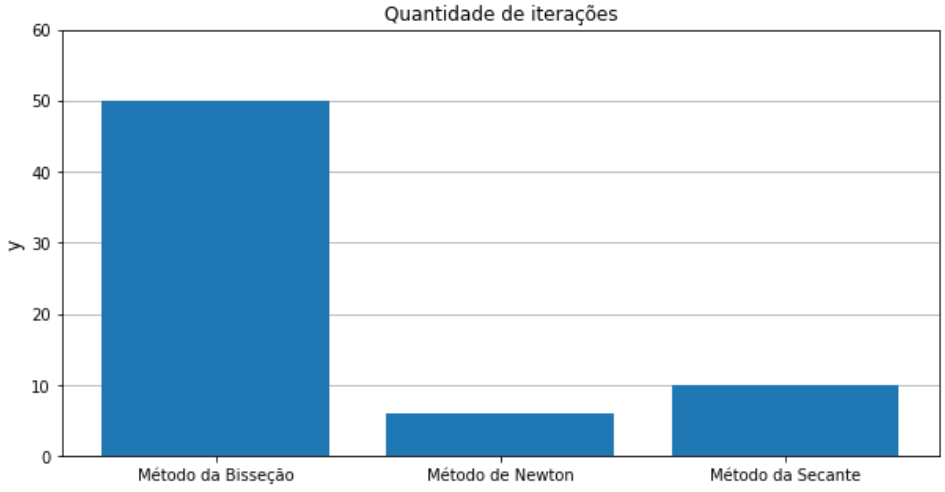
5.3.1 Gráficos – Exemplo 3

O gráfico representa a aproximação da raiz da função através dos métodos:



5.3.5 Vantagens e Desvantagens – Exemplo 3

Com a função exemplo número três pode se observar que o melhor método em termo de rapidez foi o método de Newton-Raphson. O gráfico a seguir ilustra as iterações de cada um.



6. Conclusão

Podemos concluir que a rapidez dos métodos varia de função para função como ocorreu com os exemplos. No primeiro o mais rápido foi o método de Newton e no segundo o da Secante então para escolhermos um dos métodos seria preciso análise e familiaridade com a aplicação. Pois se precisássemos analisar todos os valores próximos a raiz minuciosamente o método mais adequado seria o da bisseção, enquanto se quisermos rapidez os outros dois seriam os indicados.

7.Referências Bibliográficas

file:///C:/Users/iasmi/Documents/UFES%20-%20CIENCIA%20COMPUTAÇÃO/2018_2_(1°periodo)/OBERLAN%20-%20PROG%20FUNC/EP2/EP2%20(1).pdf (Acessado em : 02/12/2018 ás 15:16)

https://www.concrete.com.br/2017/07/21/introducao-ao-python-com-jupyter/ (Acessado em : 02/12/2018 ás 15:30)