UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI - UFSJ
DEPARTAMENTO DE ENGENHARIA ELÉTRICA - DEPEL
COORDENAÇÃO DE ENGENHARIA ELÉTRICA - COELE

GABRIEL AUGUSTO SILVA BATISTA

TRABALHO 4

São João del-Rei
Outubro de 2023

```python
import numpy as np

ci = [1.0,2.0]
ci2 = [-2.0,3.0]
new1 = 1.5
new2 = -1.5
sec1 = ci[0]
sec2= ci[1]
secb1 = ci2[0]
secb2 = ci2[1]
max_it = 1000
prec_x = 1E-5
prec_f = 1E-7

```

Aqui foi declarado as variaveis que serão utilizadas em cada metodo, tambem foi criado os parametros de precisão

```python
#Resultado
def calc_res1(x):
    return (5 * np.log(x)) - 2 + (0.4 * x)

def calc_res2(x):
    return (x - (((x**5) - 26) / (5 * x**4)))

#Derivada
def calc_der1(x):
    return (5/x) + 0.4

def calc_der2(x):
    return ((4/5) - (104/(5*(x**5))))
```

Nessa parte criei uma função para calcular os resultados das funções e das derivadas dessas funções que foram solicitadas na letra a) e b)

```python
#Bisseção a)
for i in range(max_it):
    bis = (ci[0] + ci[1])/2
    res1 = calc_res1(bis)
    if abs(ci[1]-ci[0]) < prec_x and abs(res1) < prec_f:
        break
    if res1 < 0:
        ci[0] = bis
    elif res1 >= 0:
        ci[1] = bis
print(f"Bisseção a), iteração {i+1}: {bis}\n")

#Bisseção b)
for i in range(max_it):
    bis = (ci2[0] + ci2[1])/2
    res1 = calc_res2(bis)
    if abs(ci2[1]-ci2[0]) < prec_x and abs(res1) < prec_f:
        break
    if res1 < 0:
        ci2[0] = bis
    elif res1 >= 0:
        ci2[1] = bis
print(f"Bisseção b), iteração {i+1}: {bis}\n")
```

A partir da linha 30 temos a implementação do método de bisseção, um algoritmo de busca de raízes que funciona dividindo o intervalo ao meio e selecionando o subintervalo onde a raiz está.

Para a função a) o código realiza o método de bisseção no intervalo definido por “ci” Em cada iteração, ele calcula o ponto médio “bis” e avalia a função ‘calc_res1” nesse ponto. Se a função é negativa nesse ponto, ele atualiza o limite inferior do intervalo para ser o ponto médio. Se a função é positiva ou zero, ele atualiza o limite superior do intervalo para ser o ponto médio. O loop continua até que a diferença entre os limites superior e inferior seja menor que “prec_x” (precisão no x) e o valor absoluto da função no ponto médio seja menor que “prec_f” (precisão na função).

Já para a função b) o código faz praticamente a mesma coisa que a primeira parte, mas usa um intervalo diferente “ci2” e uma função diferente “calc_res2”.
Em ambas as partes, o resultado da bisseção é impresso após todas as iterações.

```
53  #Newton a)
54  for i in range(max_it):
55      fx1 = calc_res1(new1)
56      fxl1 = calc_der1(new1)
57      new1 -= fx1/fxl1
58      if abs(fx1) < prec_f:
59          break
60  print(f"Newton-Raphson a), iteração {i+1}: {new1}\n")
61
62  #Newton b)
63  for i in range(max_it):
64      fx2 = calc_res2(new2)
65      fxl2 = calc_der2(new2)
66      new2 -= fx2/fxl2
67      if abs(fx2) < prec_f:
68          break
69  print(f"Newton-Raphson b), iteração {i+1}: {new2}\n")
70
```

Esta parte do código é uma implementação do método de Newton-Raphson, um algoritmo de busca de raízes que usa a derivada da função para encontrar a raiz.

O código é dividido em duas partes, cada uma realizando o método de Newton-Raphson para uma função diferente “calc_res1” e “calc_res2”

Para a função a) o código realiza o método de Newton-Raphson para a função “calc_res1”. Em cada iteração, ele calcula o valor da função “fx1” e sua derivada “fxl1” no ponto “new1”. Em seguida, atualiza “new1” subtraindo o quociente de “fx1” e “fxl1” dele. O loop continua até que o valor absoluto da função no ponto “new1” seja menor que “prec_f” (precisão na função).

Já para a função b) o código faz praticamente a mesma coisa que a primeira parte, mas usa um ponto inicial diferente “new2” e uma função diferente “calc_res2”.

Em ambas as partes, o resultado do método de Newton-Raphson é impresso após todas as iterações.

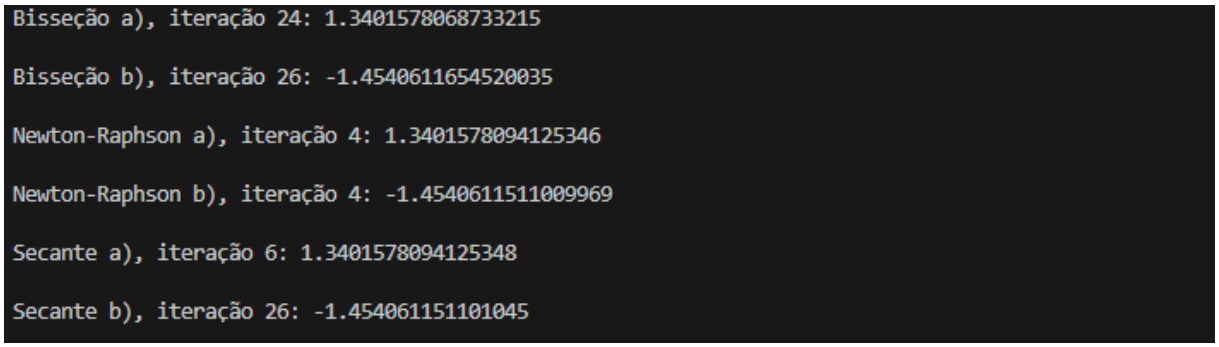
```
71  #Secante a)
72  for i in range(max_it):
73      xk1 = calc_res1(sec1)
74      xk2 = calc_res1(sec2)
75      aux = sec2
76      if xk2 != xk1:
77          sec2 -= (xk2 * (sec2 - sec1)) / (xk2 - xk1)
78          if abs(sec2-sec1) < prec_x and abs(xk2) < prec_f:
79              break
80      else:
81          print("xk2 - xk1 resultou em 0")
82          break
83      sec1 = aux
84  print(f"Secante a), iteração {i+1}: {sec2}\n")
85
86  #Secante b)
87  sec1 = 1
88  sec2= 2
89  for i in range(max_it):
90      xk3 = calc_res2(sec1)
91      xk4 = calc_res2(sec2)
92      aux = sec2
93      if xk4 != xk3:
94          sec2 -= (xk4 * (sec2 - sec1)) / (xk4 - xk3)
95          if abs(sec2-sec1) < prec_x and abs(xk4) < prec_f:
96              break
97      else:
98          print("xk4 - xk3 resultou em 0")
99          break
100     sec1 = aux
101 print(f"Secante b), iteração {i+1}: {sec2}\n")
```

Esta parte do código é uma implementação do método da secante, um algoritmo de busca de raízes que usa uma aproximação da derivada da função para encontrar a raiz.

O código é dividido em duas partes, cada uma realizando o método da secante para uma função diferente “calc_res1” e “calc_res2”.

Para a função a) o código realiza o método da secante para a função “calc_res1”. Em cada iteração, ele calcula o valor da função nos pontos sec1 e sec2 (xk1 e xk2, respectivamente). Em seguida, atualiza “sec2” subtraindo o produto de “xk2” e a diferença entre “sec2” e “sec1”, dividido pela diferença entre “xk2” e “xk1”. O loop continua até que a diferença entre “sec2” e “sec1” seja menor que “prec_x” (precisão no x) e o valor absoluto de “xk2” seja menor que “prec_f” (precisão na função). Se “xk2” e “xk1” forem iguais, o loop é interrompido e uma mensagem é impressa.

Para a função b) código faz praticamente a mesma coisa que a primeira parte, mas usa pontos iniciais diferentes (sec1 e sec2) e uma função diferente (calc_res2). Em ambas as partes, o resultado do método da secante é impresso após todas as iterações.



Essa é a saida do programa após a execução, é mostrado o numeros de iterações necessarias para até atingir alguma as condições de parada e também é mostrado a raiz encontrada.