Universidade de São Paulo

Relatório de Cálculo Numérico: Projeto Prático 1

Zeros de Funções de Uma Variável

Leandro A. Silva 9805341
Marianna Karenina de A. Flôres 10821144
Marilene Andrade Garcia 10276974

São Carlos
2020

1 Objetivos

Esse trabalho foi elaborado o objetivo de fazer demonstrações teóricas de duas afirmações sobre o método de Householder. Além da implementação eficiente de quatro métodos de aproximação de raízes, sendo eles, bissecção, secante, Newton e Halley. Os quais foram testados em determinadas equações, para em seguida, estimar a ordem de convergência desses métodos e analisar e os resultados.

2 Introdução

2.1 Ordem de Convergência

Ao considerarmos uma sequência x_k com limite x^* é natural perguntarmos quão rápido esta sequência converge, especialmente se ela tiver sido gerada por um algoritmo e o seu limite for solução de algum problema que desejamos resolver.

A ferramenta que utilizaremos para avaliar essa velocidade é a **ordem de convergência**.

Dizemos que uma sequência de convergente $x_k \to x^*$ tem razão de convergência $p \geq 1$ quando temos o seguinte limite:

$$lim \frac{|x_{k+1} - x^*|}{|x_k - x^*|p} = c \tag{1}$$

Se definirmos $\epsilon_k := ||x_k - x^*||$ e trabalharmos na expressão, encontraremos:

$$\frac{log\frac{\epsilon_{k+1}}{\epsilon_k}}{log\frac{\epsilon_k}{\epsilon_{k-1}}} \approx p \tag{2}$$

2.2 Métodos de Ordem Elevada Para Raízes de Funções

D-ésimo Método de Householder:

$$x_{k+1} = x_k + d\frac{(\frac{1}{f(x_k)})^{(d-1)}}{(\frac{1}{f(x_k)})^{(d)}} \tag{3}$$

Onde $g^{(d)}$ é a derivada de ordem d da função g. Isto é, $(\frac{1}{f(x_k)})^{(d)}$ é a d-ésima derivada de $\frac{1}{f(x_k)}$. O d-ésimo método de Householder tem, sob condições favoráveis, ordem de convergência $p = d + 1$.

3 Demonstrações Teóricas

3.1 Método de Newton

Questão proposta: Mostre que o método de Householder com $d = 1$ é o método de Newton.

Resolução:

Sabendo que o método de Newton pode ser representado pela equação:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{4}$$

Começamos igualando a equação 3, já substituindo $d = 1$, com a equação 4;

$$x_{k+1} = x_k + 1\frac{(\frac{1}{f(x_k)})^{(1-1)}}{(\frac{1}{f(x_k)})^{(1)}} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{5}$$

Reescrevendo a equação 5 de maneira mais simplificada teremos:

$$x_{k+1} = x_k + \frac{\frac{1}{f(x_k)}}{(\frac{1}{f(x_k)})'} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{6}$$

Resolvendo, agora, a derivada da parte correspondente ao **método de Householder**, teremos:

$$x_{k+1} = x_k + \frac{\frac{1}{f(x_k)}}{-\frac{f'(x_k)}{f(x_k)^2}} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Realizando a divisão entre as frasões da parte correspondente ao **método de Householder** :

$$x_{k+1} = x_k + \frac{1}{f(x_k)} \cdot -\frac{f(x_k)^2}{f'(x_k)} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Simplificando a equação:
$$x_{k+1} = x_k + \frac{1}{1} \cdot -\frac{f(x_k)}{f'(x_k)} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Chegamos então em:
$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Finalmente, podemos concluir, através da demonstração acima, que o **método de Householder para** $d = 1$ é equivalente ao **método de Newton**.

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{7}$$

3.2 Método de Halley

Questão proposta: Mostre que o método de Householder com $d = 2$ é dado pela fórmula:

$$x_{k+1} = x_k - \frac{f(x_k)f'(x_k)}{f'(x_k)^2 - \frac{1}{2}f(x_k)f''(x_k)} \tag{8}$$

Resolução:

Começamos igualando a equação 3, já substituindo $d = 2$, com a equação 8;

$$x_{k+1} = x_k + 2\frac{(\frac{1}{f(x_k)})^{(2-1)}}{(\frac{1}{f(x_k)})^{(2)}} = x_k - \frac{f(x_k)f'(x_k)}{f'(x_k)^2 - \frac{1}{2}f(x_k)f''(x_k)} \tag{9}$$

Reescrevendo a equação 9 de maneira mais simplificada teremos:

$$x_{k+1} = x_k + 2\frac{(\frac{1}{f(x_k)})'}{(\frac{1}{f(x_k)})''} = x_k - \frac{f(x_k)f'(x_k)}{f'(x_k)^2 - \frac{1}{2}f(x_k)f''(x_k)} \tag{10}$$

Resolvendo a derivada do numerador da parte correspondente ao **método de Householder**, teremos:

$$x_{k+1} = x_k + 2\frac{(-\frac{f'(x_k)}{f(x_k)^2})}{(\frac{1}{f(x_k)})''} = x_k - \frac{f(x_k)f'(x_k)}{f'(x_k)^2 - \frac{1}{2}f(x_k)f''(x_k)}$$

Agora, resolveremos a derivada do denominador da parte correspondente ao **método de Householder**:

$$x_{k+1} = x_k + 2\frac{(-\frac{f'(x_k)}{f(x_k)^2})}{2\frac{f'(x_k)^2 - f(x_k)f''(x_k)}{f(x_k)^3}} = x_k - \frac{f(x_k)f'(x_k)}{f'(x_k)^2 - \frac{1}{2}f(x_k)f''(x_k)}$$

Realizando a divisão entre as frasões da parte correspondente ao **método de Householder** :

$$x_{k+1} = x_k + 2 \cdot -\frac{f'(x_k)}{f(x_k)^2} \cdot \frac{f(x_k)^3}{2f'(x_k)^2 - f(x_k)f''(x_k)} = x_k - \frac{f(x_k)f'(x_k)}{f'(x_k)^2 - \frac{1}{2}f(x_k)f''(x_k)}$$

Simplificando a equação:

$$x_{k+1} = x_k - \frac{f'(x_k)}{1} \cdot \frac{f(x_k)}{f'(x_k)^2 - \frac{f(x_k)f''(x_k)}{2}} = x_k - \frac{f(x_k)f'(x_k)}{f'(x_k)^2 - \frac{1}{2}f(x_k)f''(x_k)}$$

Assim, chegamos então em:

$$x_{k+1} = x_k - \frac{f(x_k)f'(x_k)}{f'(x_k)^2 - \frac{1}{2}f(x_k)f''(x_k)} = x_k - \frac{f(x_k)f'(x_k)}{f'(x_k)^2 - \frac{1}{2}f(x_k)f''(x_k)}$$

Finalmente, podemos concluir, através da demonstração acima, que o **método de Householder para** $d = 2$ resulta na equação 8, proposta inicialmente.

$$x_{k+1} = x_k - \frac{f(x_k)f'(x_k)}{f'(x_k)^2 - \frac{1}{2}f(x_k)f''(x_k)} = x_k - \frac{f(x_k)f'(x_k)}{f'(x_k)^2 - \frac{1}{2}f(x_k)f''(x_k)} \tag{11}$$

4 Implementações

A linguagem de programação escolhida foi Python e os códigos das implementações dos métodos e da ordem de conversão estão em arquivos diferentes nomeados de acordo com método codificado. No arquivo "main.py" estão todas as equações e as chamadas dos métodos, sendo que os resultados são salvos no arquivo texto "resultados.txt".

O sistema operacional no qual foi feita a execução dos códigos foi o Ubuntu Linux. A compilação e execução é feita no terminal, bastando acessar o diretório do projeto e executar o comando:

```
python main.py
```

Toda a implementação com os resultados podem ser encontradas no seguinte repositório do GitHub: https://github.com/leandroS08/SME0602$_{P}rojeto1.git$

4.1 Método da Bissecção

O código desenvolvido para esse método segue o algoritmo apresentado em aula.

```
import math
import numpy as np

def bisseccao( f, a, b, e ):
    x = []
```

```
6
7       while abs(b-a)/2 > e:
8           xi = (a + b)/2
9           x.append(xi)
10
11          if f(xi)*f(a) < 0:
12              a_next = a
13              b_next = xi
14          elif f(xi)*f(b) < 0:
15              a_next = xi
16              b_next = b
17          else:
18              break
19
20          a = a_next
21          b = b_next
22
23      return x
```

4.2 Método da Secante

O código desenvolvido para esse método segue o algoritmo apresentado em aula.

```
1  import math
2  import numpy as np
3
4  def secante( f, x0, x1, e ):
5    x = []
6    f0 = f(x0)
7    f1 = f(x1)
8
9    x2 = x1 - f1 * (x1 -x0) / (f1 - f0)
10   x.append(x2)
11
12   while abs(x2 - x1) > e:
13     x0 = x1
14     f0 = f1
15
16     x1 = x2
17     f1 = f(x1)
18
19     x2 = x1 - f1 * (x1 -x0) / (f1 - f0)
20     x.append(x2)
21
22   return x
```

4.3 Método de Newton

O código desenvolvido para esse método segue o algoritmo apresentado em aula.

```
1  import math
2  import numpy as np
3
4  def newton( f, x0, e ):
5      x = []
6      x.append(x0)
7      x_now = x0 - f(x0)/f_der(f,x0)
```

```
    x_previous = x0

    while abs(x_now - x_previous) > e:
        if(f_der(f,x_now) != 0):
            x_next = x_now - f(x_now)/f_der(f,x_now)

        x.append(x_now)
        x_previous = x_now
        x_now = x_next

    return x

def f_der( f, x ):
    h = 1e-5
    return (f(x+h)-f(x-h))/(2*h)
```

4.4 Método de Halley

O código desenvolvido para esse método foi baseado no método de Newton.

```
import math
import numpy as np

def halley( f, x0, e ):
    x = []
    x.append(x0)

    der_f = f_der(f,x0)
    der2_f = f_der2(f,x0)
    x_now = x0 - (f(x0)*der_f)/(pow(der_f,2)-0.5*f(x0)*der2_f)
    x_previous = x0

    while abs(x_now - x_previous) > e:
        if(f_der(f,x_now) != 0):
            x_next = x_now - (f(x_now)*der_f)/(pow(der_f,2)-0.5*f(x_now)*
   der2_f)

        x.append(x_now)
        x_previous = x_now
        x_now = x_next
        der_f = f_der(f,x_now)
        der2_f = f_der2(f,x_now)

    return x

def f_der( f, x ):
    h = 1e-5
    return (f(x+h)-f(x-h))/(2*h)

def f_der2( f, x ):
    h = 1e-5
    return (f_der(f,x+h)-f_der(f,x-h))/(2*h)
```

4.5 Ordem de convergência

Para analisar a ordem de convergência dos métodos foi desenvolvido o seguinte algoritmo, baseado no documento de especificação desse trabalho.

```
import math
import numpy as np

def ordem_convergencia( x, x_right ):
    tam = len(x)
    e = []
    p = 0.0

    for i in range(0, tam):
        e.append( abs(x[i] - x_right) )

    for j in range(1, tam-1):
        n = math.log( (e[j+1]/e[j]) ,10)
        d = math.log( (e[j]/e[j-1]) ,10)
        if(d != 0):
            p =  n / d

    c = e[tam-1] / pow(e[tam-2],p)

    return c
```

5 Uso das implementações

Todos os métodos implementados foram testados para três equações:

1. x - cos(x) = 0
2. $x^3 - 9x^2 + 27x - 27 = 0$
3. $e^x - cos(x) = 0$ (maior raiz)

Em seguida os resultados foram analisados e organizados na tabela 1, tabela 2 e tabela 3. Nas quais é possível notar a aproximação inicial (x_0) na primeira linha, além de todos os outros pontos por onde o método passou nas seguintes, com 16 casas decimais de aproximação. Os pontos iniciais dos métodos secante e bissecção foram ajustados com o intuito de que todos os métodos se iniciassem na mesma aproximação inicial.Também foi analisada a ordem de convergência de cada método para todas as equações e os resultados estão na tabela 4.

5.1 Equação 1

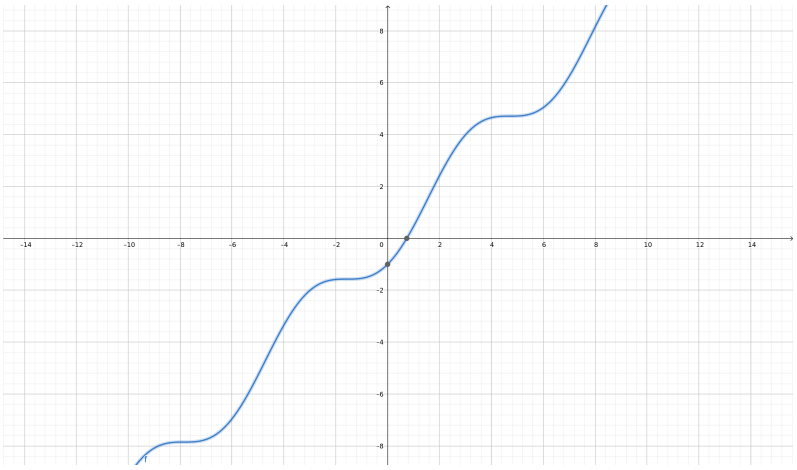

Fig. 1: Gráfico da equação 1.

Os pontos iniciais usados foram:

- Bissecção: $a_0 = 0$; $b_0 = \pi$; sendo a aproximação inicial $x_0 = 1{,}570796326794890$
- Secante: $x_0 = 22.0$; $x_1 = 6.0$; sendo a aproximação inicial $x_2 = 1{,}510206751496100$
  - Vale ressaltar que os pontos iniciais da secante não estão suficientemente próximos da raiz, logo a ordem de convergência não será $\alpha = \frac{1+\sqrt{5}}{2}$. E também não é possível garantir que o método irá convergir, apesar de que empiricamente foi demonstrado que para esses valores ele converge após algumas iterações.
- Newton: $x_0 = 1{,}570796326794890$
- Halley: $x_0 = 1{,}570796326794890$

Tab. 1: Resultados da equação 1

Iterações	**Bissecção**	**Secante**	**Newton**	**Halley**
1	1,570796326794890	1,510206751496100	1,570796326794890	1,570796326794890
2	0,785398163397448	-0,302698460131341	0,785398163398233	0,785398163398233
3	0,392699081698724	0,539318605102993	0,739536133514740	0,746252472292152
4	0,589048622548086	0,825292459128243	0,739085178106004	0,739085175691060
5	0,687223392972767	0,735052369685492	0,739085133215161	0,739085133215160
6	0,736310778185107	0,739011628383364	0,739085133215160	
7	0,760854470791278	0,739085198809083		
8	0,748582624488192	0,739085133214096		
9	0,742446701336650	0,739085133215160		
10	0,739378739760879	0,739085133215160		
11	0,737844758972993	0,739085133215160		
12	0,738611749366936			
13	0,738995244563907			
14	0,739186992162393			
15	0,739091118363150			
16	0,739043181463529			
17	0,739067149913339			
18	0,739079134138245			
19	0,739085126250697			
20	0,739088122306924			
21	0,739086624278810			
22	0,739085875264754			
23	0,739085500757725			
24	0,739085313504211			
25	0,739085219877454			
26	0,739085173064076			
27	0,739085149657386			
28	0,739085137954042			
29	0,739085132102369			
30	0,739085135028206			
31	0,739085133565288			

Continua na próxima página

Table 1 – continuação da página anterior

Iterações	Bissecção	Secante	Newton	Halley
32	0,739085132833829			
33	0,739085133199558			
34	0,739085133382423			
35	0,739085133290990			
36	0,739085133245274			
37	0,739085133222416			
38	0,739085133210987			
39	0,739085133216702			
40	0,739085133213844			
41	0,739085133215273			
42	0,739085133214559			
43	0,739085133214916			
44	0,739085133215095			
45	0,739085133215184			
46	0,739085133215139			
47	0,739085133215161			
48	0,739085133215150			
49	0,739085133215156			
50	0,739085133215159			
51	0,739085133215160			
52	0,739085133215161			
53	0,739085133215160			
54	0,739085133215160			

5.2 Equação 2

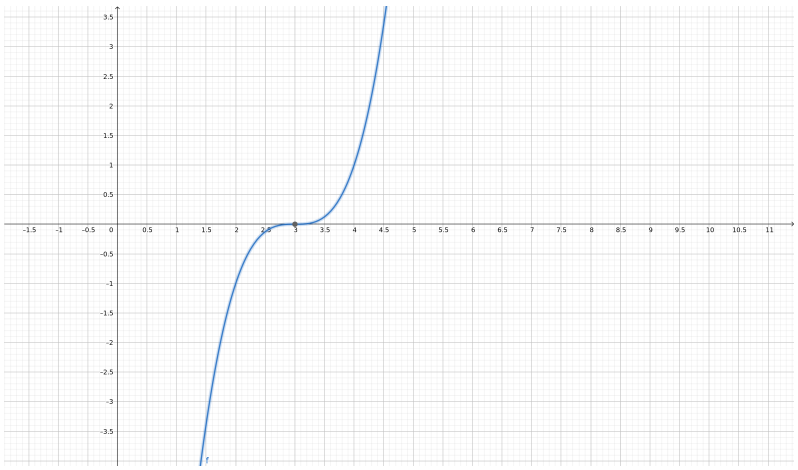

Fig. 2: Gráfico da equação 2.

Os pontos iniciais usados foram:

- Bissecção: $a_0 = 0$; $b_0 = 15.0$; sendo a aproximação inicial $x_0 = 7{,}50000000000000$

- Secante: $x_0 = 9.0$; $x_1 = 11.0$; sendo a aproximação inicial $x_2 = 7{,}54054054054054$
  - As mesmas observações para os pontos iniciais desse método, na equação anterior, ainda são válidas para esta.
- Newton: $x_0 = 7{,}50000000000000$
- Halley: $x_0 = 7{,}50000000000000$

Tab. 2: Resultados da equação 2

Iterações	**Bissecção**	**Secante**	**Newton**	**Halley**
1	7,50000000000000	7,54054054054054	7,50000000000000	7,50000000000000
2	3,75000000000000	6,76652493235407	5,99999999993224	5,25000360745145
3	1,87500000000000	5,73706127040383	4,99999999989635	5,05435047881582
4	2,81250000000000	5,09603726805613	4,33333333330809	4,02717368804155
5	3,28125000000000	4,57346630116852	3,88888888883258	3,51359012127916
6	3,04687500000000	4,19031488039005	3,59259259259452	3,25679751820556
7	2,92968750000000	3,89780291528159	3,39506172826311	3,12840208193433
8	2,98828125000000	3,67794594743857	3,26337448550874	3,06419909492423
9	3,01757812500000	3,51170390799235	3,17558299032244	3,03209919999160
10	3,00292968750000	3,38629196046171	3,11705532680689	3,01604674235244
11	2,99560546875000	3,29159791323392	3,07803688528764	3,00802568687212
12	2,99926757812500	3,22012228022930	3,05202459103491	3,00401051473241
13	3,00109863281250	3,16616494933238	3,03468306136235	3,00200876057015
14	3,00018310546875	3,12543433809472	3,02312204204629	3,00100329476161
15	2,99972534179687	3,09468754295642	3,01541469621704	3,00049795949278
16	2,99995422363281	3,07147752232844	3,01027646358667	3,00025061878286
17	3,00006866455078	3,05395678208402	3,00685097612983	3,00012348149014
18	3,00001144409179	3,04073077065715	3,00456731622338	3,00006688574546
19		3,03074674882957	3,00304487534913	3,00003622881115
20		3,02321003408139	3,00202993142270	3,00002069939939
21		3,01752073632927	3,00135329617727	3,00001242353732
22		3,01322601255201	3,00090228893652	
23		3,00998402150638	3,00060159085624	
24		3,00753671480780	3,00040117882223	
25		3,00568929766664	3,00026753176340	
26		3,00429472379029	3,00017862087231	
27		3,00324199109193	3,00011869549918	
28		3,00244730645444	3,00007969549918	
29		3,00184741681308	3,00005290304635	
30		3,00139457173962	3,00003623637968	
31		3,00105273489283	3,00002854407199	
32		3,00079468548506	3,00000854407199	
33		3,00059988458975	3,00000354407199	
34		3,00045284258226		
35		3,00034186043176		
36		3,00025801255234		

Continua na próxima página

Table 2 – continuação da página anterior

Iterações	Bissecção	Secante	Newton	Halley
37		3,00019486527157		
38		3,00014713767565		
39		3,00011101949495		
40		3,00008393085942		
41		3,00006373751294		
42		3,00004713409472		
43		3,00003702766624		
44		3,00002355242826		
45		3,00001546728548		
46		3,00001546728548		

5.3 Equação 3

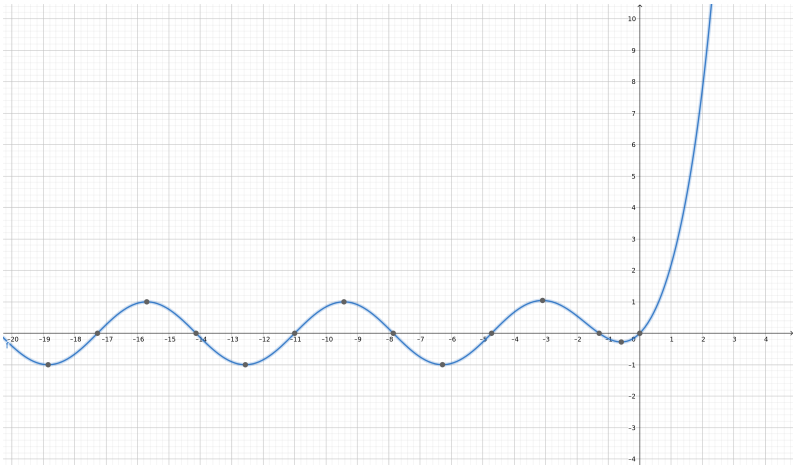

Fig. 3: Gráfico da equação 3.

Os pontos iniciais usados foram:

- Bissecção: $a_0 = -0.5$; $b_0 = 2.5$; sendo a aproximação inicial $x_0 = 1{,}000000000000000000$
- Secante: $x_0 = 1.75$; $x_1 = 2.15$; sendo a aproximação inicial $x_2 = 1{,}008247257304410000$
  - Esta equação possui raiz múltipla, logo não é garantido que o método irá convergir, apesar de que empiricamente foi demonstrado que para esses valores iniciais ele converge.
- Newton: $x_0 = 1{,}000000000000000000$
- Halley: $x_0 = 1{,}000000000000000000$

Tab. 3: Resultados da equação 3

It	Bissecção	Secante	Newton	Halley
1	1,000000000000000000	1,008247257304410000	1,000000000000000000	1,000000000000000000
2	0,250000000000000000	0,644284036672756000	0,388165516874897000	0,150187749985688000
3	-0,125000000000000000	0,279417115137927000	0,092030345952365200	0,100392343849721000
4	0,062500000000000000	0,102296540232614000	0,007346329327991320	0,000668474838912525
5	-0,031250000000000000	0,021701430386016400	0,000053315921033724	0,000000000248471730
6	0,015625000000000000	0,002011989564316990	0,000000002842335777	0,000000000000000042
7	-0,007812500000000000	0,000042816593134421	0,000000000000000093	
8	0,003906250000000000	0,000000085999269942		
9	-0,001953125000000000	0,000000000003682036		
10	0,000976562500000000	0,000000000000000093		
11	-0,000488281250000000	0,000000000000000093		
12	0,000244140625000000			
13	-0,000122070312500000			
14	0,000061035156250000			
15	-0,000030517578125000			
16	0,000015258789062500			
17	-0,000007629394531250			
18	0,000003814697265625			
19	-0,000001907348632813			
20	0,000000953674316406			
21	-0,000000476837158203			
22	0,000000238418579102			
23	-0,000000119209289551			
24	0,000000059604644775			
25	-0,000000029802322388			
26	0,000000014901161194			
27	-0,000000007450580597			
28	0,000000003725290298			
29	-0,000000001862645149			
30	0,000000000931322575			
31	-0,000000000465661287			
32	0,000000000232830644			
33	-0,000000000116415322			
34	0,000000000058207661			
35	-0,000000000029103830			
36	0,000000000014551915			
37	-0,000000000007275958			
38	0,000000000003637979			
39	-0,000000000001818989			
40	0,000000000000909495			
41	-0,000000000000454747			
42	0,000000000000227374			
43	-0,000000000000113687			
44	0,000000000000056843			

Continua na próxima página

Table 3 – continuação da página anterior

It	Bissecção	Secante	Newton	Halley
45	-0,000000000000028422			
46	0,000000000000014211			
47	-0,000000000000007105			
48	0,000000000000003553			
49	-0,000000000000001776			
50	0,000000000000000888			
51	-0,000000000000000444			
52	0,000000000000000222			
53	-0,000000000000000111			
54	0,000000000000000056			

5.4 Ordem de convergência de cada método para as equações

Usando a fórmula 7, foi estimada a taxa de convergência dos métodos para cada equação proposta. Os resultados estão na tabela 4.

Tab. 4: Resultados da ordem de convergência

Ordem de convergência	Bissecção	Secante	Newton	Halley
Equação 1	0.333793791648	0.0	0.000220176920926	1.03731466744
Equação 2	-4.4190225827	0.0	0.729524888594	0.912035435947
Equação 3	1.0	0.0	1.75190937755	1.05283766532

6 Resultados e conclusão

A demonstração teórica e a implementação em Python de diferentes métodos numéricos permitiram uma análise do funcionamento e da eficiência de cada um para encontrar a raiz de funções de uma variável.

Para a equação 1 é possível ver nitidamente por meio da tabela de mesmo índice que os métodos mais "certeiros" são, em ordem: Halley (5 iterações), Newton (6 iterações), Secante (11 iterações) e Bissecção (54 iterações). O mesmo vale para a tabela 2: Halley (6 iterações), Newton (7 iterações), Secante (11 iterações) e Bissecção (54 iterações). Vale ressaltar nesses dois casos que os métodos de Newton e Halley apresentam desempenho muito parecido, isso porque a derivada segunda dessas duas equações para as raízes são valores relativamente pequenos - algo em torno de 0,75 e 2 respectivamente.

Para equação 2, a ordem dos métodos e o número de iterações necessárias em geral para todos os métodos muda: Bissecção (18 iterações), Halley (21 iterações), Newton (33 iterações) e Secante (46 iterações). Isso se dá porque... Além disso, é possível notar que existe uma diferença bem grande entre as iterações de Newton e Halley, pois, neste caso, a derivada segunda é bem maior para a raíz 3: -6.

Portanto, os resultados obtidos foram bem conclusivos em ressaltar que os métodos de bissecção e secante geralmente necessitam de mais iterações do que os métodos de Newton e Halley

para encontrar a raiz na precisão desejada. Entretanto, os dois últimos métodos visivelmente necessitam de maior poder computacional para serem calculados, pois utilizam cálculos de derivadas. Uma análise de tempo de execução versus número de iterações seria necessária para uma conclusão mais correta sobre a eficiência de cada método.