

# Universidade de São Paulo

## Instituto de Física de São Carlos

### **Lista 2**

Pedro Calligaris Delbem 5255417

Professor: Attilio Cucchieri

Março de 2025

# Sumário

<b>1</b>	<b>Finding roots</b>	<b>2</b>
1.1	Exercício 1 . . . . .	2
1.2	Exercício 2 . . . . .	3
1.2.1	Raizes de $f(x) = x^2 - 5 = 0$ . . . . .	6
1.2.2	Raizes de $f(x) = 5x^3 - 5x - 24 = 0$ . . . . .	11
<b>2</b>	<b>Eigenvalues of the wave equation</b>	<b>15</b>
2.1	Exercício 3 . . . . .	15
2.2	Exercício 4 . . . . .	16
2.3	Exercício 5 . . . . .	19

# 1 Finding roots

## 1.1 Exercício 1

Tarefa: Demonstrar que no método de Newton-Raphson

$$x_{k+1} = x_k + \frac{f(x_k)}{f'(x_k)} \quad (1)$$

a convergência é quadrática.

Expandimos  $f(x)$  em torno de  $x_n - r$  - onde  $r$  é a raiz de  $f(x)$  - e obtemos:

$$f(x_n) = f(r) + f'(r)(x_n - r) + \frac{1}{2}f''(r)(x_n - r)^2 + O(x_n - r)^3 \quad (2)$$

E como  $f(r) = 0$ , obtemos:

$$f(x_n) = f'(r)(x_n - r) + \frac{1}{2}f''(r)(x_n - r)^2 + O(x_n - r)^3 \quad (3)$$

Expandindo-se, também,  $f'(x_n)$  e obtemos:

$$f'(x_n) = f'(r) + f''(r)(x_n - r) + O(x_n - r)^2 \quad (4)$$

Substituindo em  $x_{n+1} = x_n + \frac{f(x_n)}{f'(x_n)}$  obtemos:

$$x_{n+1} = x_n - \frac{f'(r)(x_n - r) + \frac{1}{2}f''(r)(x_n - r)^2}{f'(r) + f''(r)(x_n - r)} \quad (5)$$

Subtraindo  $r$  de ambos os lados:

$$x_{n+1} - r = x_n - r - \frac{f'(r)(x_n - r) + \frac{1}{2}f''(r)(x_n - r)^2}{f'(r) + f''(r)(x_n - r)} \quad (6)$$

Colocando o termo  $x_n - r$  em evidência:

$$x_{n+1} - r = (x_n - r) \left[ 1 - \frac{f'(r) + \frac{1}{2}f''(r)(x_n - r)}{f'(r) + f''(r)(x_n - r)} \right] \quad (7)$$

Para  $x_n - r$  pequeno,  $f''(r)(x_n - r)$  é desprezível e assim o desprezamos no denominador - obtendo:

$$x_{n+1} - r = (x_n - r) \left[ 1 - \frac{f'(r) + \frac{1}{2}f''(r)(x_n - r)}{f'(r)} \right] \quad (8)$$

Isolando  $x_n - r$ :

$$x_{n+1} - r = -(x_n - r)^2 \left[ \frac{\frac{1}{2}f''(r)}{f'(r)} \right] \quad (9)$$

Rearranjando:

$$r - x_{n+1} = (r - x_n)^2 \left[ \frac{\frac{1}{2}f''(r)}{f'(r)} \right] \quad (10)$$

Como  $r - x_n$  é o erro cometido na  $n$ -ésima iteração e  $r - x_{n+1}$  é o erro cometido na  $n+1$ -ésima iteração, temos que o erro da iteração  $n+1$  é proporcional ao quadrado do erro da iteração  $n$  e portanto a convergência é quadrática.

## 1.2 Exercício 2

Tarefa: Achar as raízes das equações  $f(x) = x^2 - 5 = 0$  e  $f(x) = 5x^3 - 5x - 24 = 0$  usando os métodos de Newton-Raphson e da secante para diferentes chutes iniciais e diferentes condições de convergência.

Código Escrito:

```
1  !-----
2  ! File: L2-5255417-ex-2.f90
3  !
4  ! Description:
5  !
6  !
7  ! Dependencies:
8  !   - None
9  !
10 ! Since:
11 !   - 03/2025
12 !
13 ! Authors:
14 !   - Pedro C. Delbem <pedrodelbem@usp.br>
15 !-----
16 program find_roots
17
18     !deactivate implicit typing
19     implicit none
20
21     !define variables
22     real x_kminus1, x_k, x_kplus1, f_x_kminus1, f_x_k, df_x_k, f_x
23     (5), initial_guess
24     integer iteration
25
26     !request coefficients
27     write(*,*) "Insert x^4 coefficient:"
28     read(*,*) f_x(5)
29     write(*,*) "Insert x^3 coefficient:"
30     read(*,*) f_x(4)
31     write(*,*) "Insert x^2 coefficient:"
32     read(*,*) f_x(3)
33     write(*,*) "Insert x coefficient:"
34     read(*,*) f_x(2)
35     write(*,*) "Insert constant coefficient:"
36     read(*,*) f_x(1)
37
38     !request initial guess
39     write(*,*) "Insert initial guess:"
40     read(*,*) initial_guess
41
42     !initialize variables
43     x_k = initial_guess
44     x_kplus1 = 0.0
45     f_x_k = f(x_k, f_x)
46     df_x_k = df(x_k, f_x)
47     iteration = 1
48
49     !open first output file
```

```

49 open(unit=1, file='output1.txt', action='write')
50
51 !print header
52 write(1,*) 'Newton-Raphson Method'
53 write(1,*) 'Initial guess: ', x_k
54 write(1,*) 'f(x) = ', f_x(5), f_x(4), f_x(3), f_x(2), f_x(1)
55
56 !print first iteration
57 write(1,*) iteration, f(x_kplus1, f_x)
58
59 !update f_x and df_x
60 f_x_k = f(x_k, f_x)
61 df_x_k = df(x_k, f_x)
62
63 !update x_kplus1
64 x_kplus1 = x_k - f_x_k/df_x_k
65
66 !update iteration
67 iteration = iteration + 1
68
69 !print second iteration
70 write(1,*) iteration, f(x_kplus1, f_x)
71
72 !Newton-Raphson method
73 do while (abs(x_kplus1 - x_k) > 1e-6)
74
75     !update x_k
76     x_k = x_kplus1
77
78     !update f_x and df_x
79     f_x_k = f(x_k, f_x)
80     df_x_k = df(x_k, f_x)
81
82     !update x_kplus1
83     x_kplus1 = x_k - f_x_k/df_x_k
84
85     !update iteration
86     iteration = iteration + 1
87
88     !print iteration
89     write(1,*) iteration, f(x_kplus1, f_x)
90
91 end do
92
93 !print root
94 write(*,*) 'Root:', x_kplus1
95
96 !close first output file
97 close(1)
98
99 !reinitialize variables
100 x_kminus1 = initial_guess - 1.0
101 x_k = initial_guess
102 x_kplus1 = 0.0
103 f_x_kminus1 = f(x_kminus1, f_x)
104 f_x_k = f(x_k, f_x)
105 iteration = 1
106

```

```

107 !open second output file
108 open(unit=2, file='output2.txt', action='write')
109
110 !print header
111 write(2,*) 'Secant Method'
112 write(2,*) 'Initial guess: ', x_k
113 write(2,*) 'f(x) = ', f_x(5), f_x(4), f_x(3), f_x(2), f_x(1)
114
115 !print first iteration
116 write(2,*) iteration, f(x_kplus1, f_x)
117
118 !update f_x_k and f_x_k-1
119 f_x_kminus1 = f(x_kminus1, f_x)
120 f_x_k = f(x_k, f_x)
121
122 !update x_kplus1 and x_kminus1
123 x_kplus1 = x_k - f_x_k*(x_k - x_kminus1)/(f_x_k - f_x_kminus1)
124 x_kminus1 = x_k
125
126 !update iteration
127 iteration = iteration + 1
128
129 !print second iteration
130 write(2,*) iteration, f(x_kplus1, f_x)
131
132 !Secant method
133 do while (abs(x_kplus1 - x_k) > 1e-6)
134
135     !update x_k
136     x_k = x_kplus1
137
138     !update f_x_k and f_x_k-1
139     f_x_kminus1 = f(x_kminus1, f_x)
140     f_x_k = f(x_k, f_x)
141
142     !update x_kplus1 and x_kminus1
143     x_kplus1 = x_k - f_x_k*(x_k - x_kminus1)/(f_x_k -
f_x_kminus1)
144     x_kminus1 = x_k
145
146     !update iteration
147     iteration = iteration + 1
148
149     !print iteration
150     write(2,*) iteration, f(x_kplus1, f_x)
151
152 end do
153
154 !print root
155 write(*,*) 'Root:', x_kplus1
156
157 !close second output file
158 close(2)
159
160 contains
161
162 function f(x, f_x) result(result)
163     real, intent(in) :: x

```

```

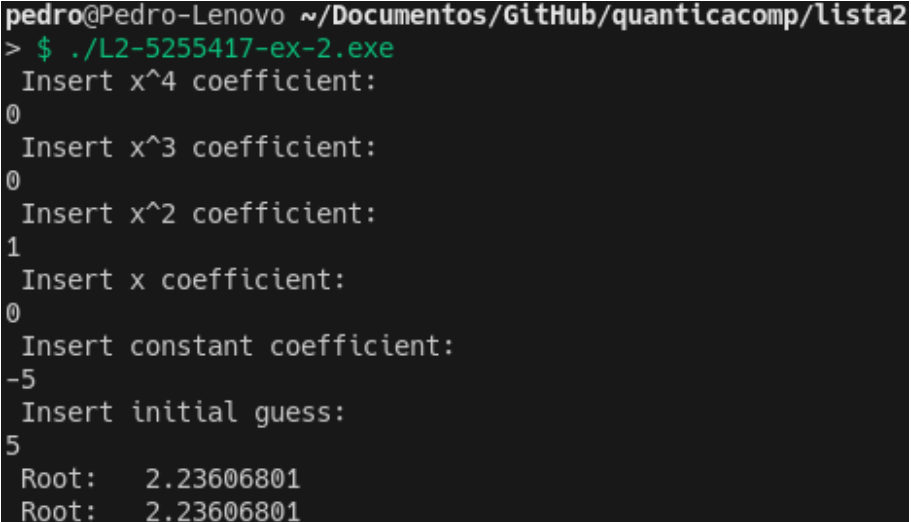
164     real, intent(in) :: f_x(5)
165     real result
166
167     result = f_x(5)*x**4. + f_x(4)*x**3. + f_x(3)*x**2. + f_x
168 (2)*x + f_x(1)
169 end function f
170
171 function df(x,f_x) result(result)
172     real, intent(in) :: x
173     real, intent(in) :: f_x(5)
174     real result
175
176     result = 4.*f_x(5)*x**3. + 3.*f_x(4)*x**2. + 2.*f_x(3)*x +
177 f_x(2)
178 end function df
179
180 end program find_roots

```

Resultados:

### 1.2.1 Raízes de $f(x) = x^2 - 5 = 0$

Para as seguintes informações iniciais:



```

pedro@Pedro-Lenovo ~/Documentos/GitHub/quanticacomp/lista2
> $ ./L2-5255417-ex-2.exe
Insert x^4 coefficient:
0
Insert x^3 coefficient:
0
Insert x^2 coefficient:
1
Insert x coefficient:
0
Insert constant coefficient:
-5
Insert initial guess:
5
Root: 2.23606801
Root: 2.23606801

```

Figura 1: Teste feito para gerar o gráfico onde as duas últimas linhas são as raízes encontradas pelos métodos de Newton-Raphson e da secante, respectivamente

Obteve-se o valor da raiz = 2.23606801 para ambos os métodos.

Com o seguinte gráfico para o método de Newton-Raphson:

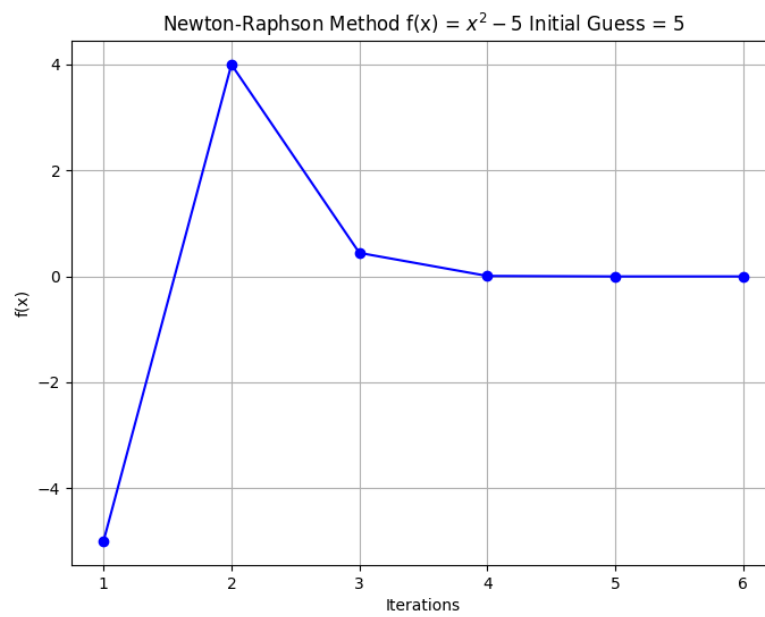


Figura 2:  $f(x)$  ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

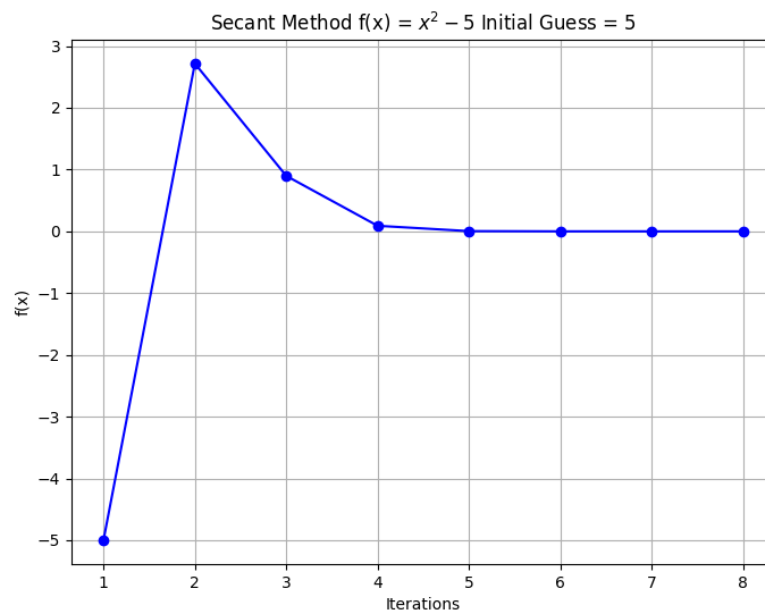


Figura 3:  $f(x)$  ao decorrer das iterações

Para as seguintes informações iniciais:



```

pedro@Pedro-Lenovo ~/Documentos/GitHub/quanticacomp/lista2
> $ ./L2-5255417-ex-2.exe
Insert x^4 coefficient:
0
Insert x^3 coefficient:
0
Insert x^2 coefficient:
1
Insert x coefficient:
0
Insert constant coefficient:
-5
Insert initial guess:
10
Root: 2.23606801
Root: 2.23606801

```

Figura 4: Teste feito para gerar o gráfico onde as duas últimas linhas são as raízes encontradas pelos métodos de Newton-Raphson e da secante, respectivamente

Obteve-se o valor da raiz = 2.23606801 para ambos os métodos.

Com o seguinte gráfico para o método de Newton-Raphson:

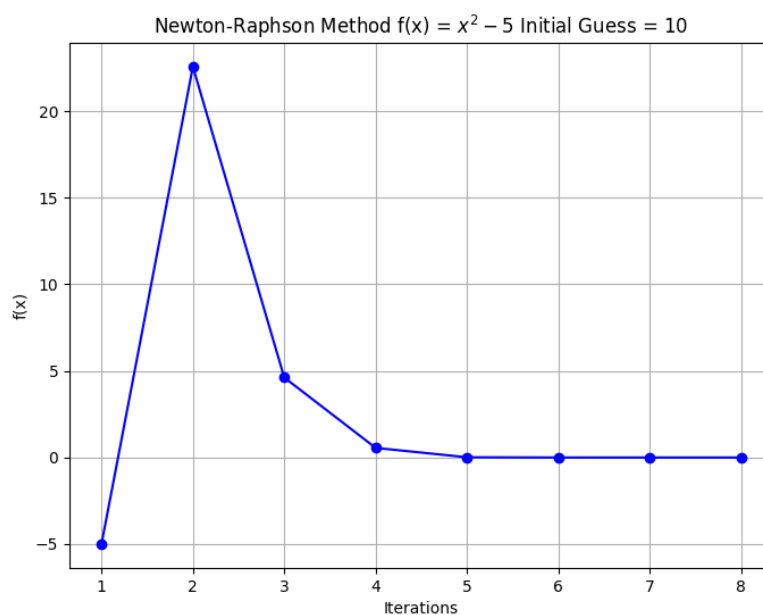


Figura 5:  $f(x)$  ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

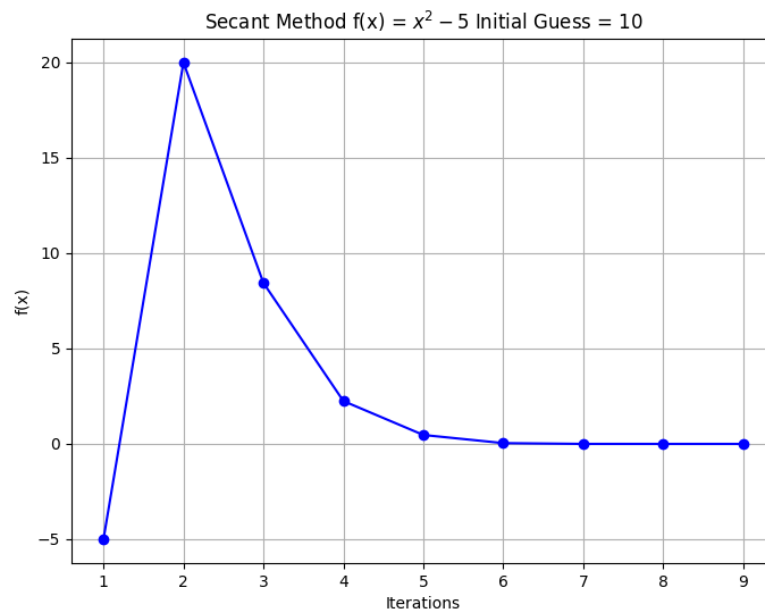


Figura 6:  $f(x)$  ao decorrer das iterações

Para as seguintes informações iniciais:

```

pedro@Pedro-Lenovo ~/Documentos/GitHub/quanticacomp/lista2
> $ ./L2-5255417-ex-2.exe
Insert x^4 coefficient:
0
Insert x^3 coefficient:
0
Insert x^2 coefficient:
1
Insert x coefficient:
0
Insert constant coefficient:
-5
Insert initial guess:
20
Root: 2.23606801
Root: 2.23606801

```

Figura 7: Teste feito para gerar o gráfico onde as duas últimas linhas são as raízes encontradas pelos métodos de Newton-Raphson e da secante, respectivamente

Obteve-se o valor da raiz = 2.23606801 para ambos os métodos.

Com o seguinte gráfico para o método de Newton-Raphson:

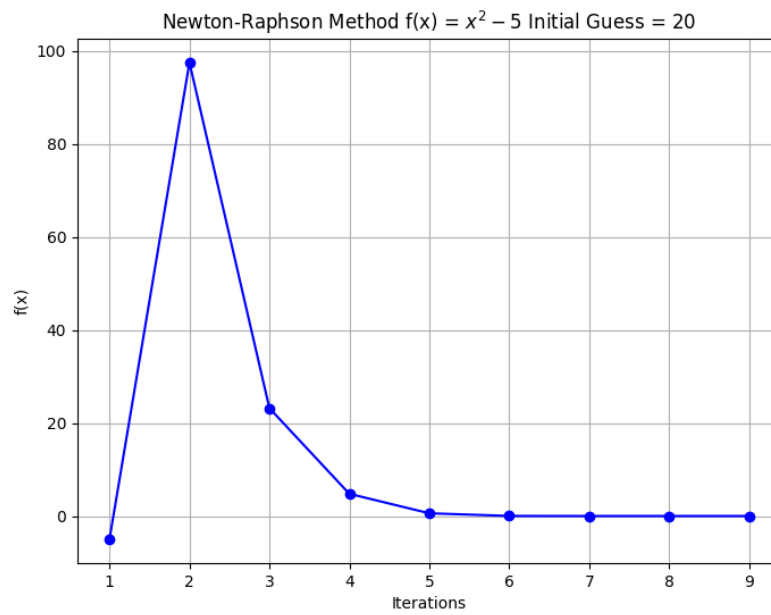


Figura 8:  $f(x)$  ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

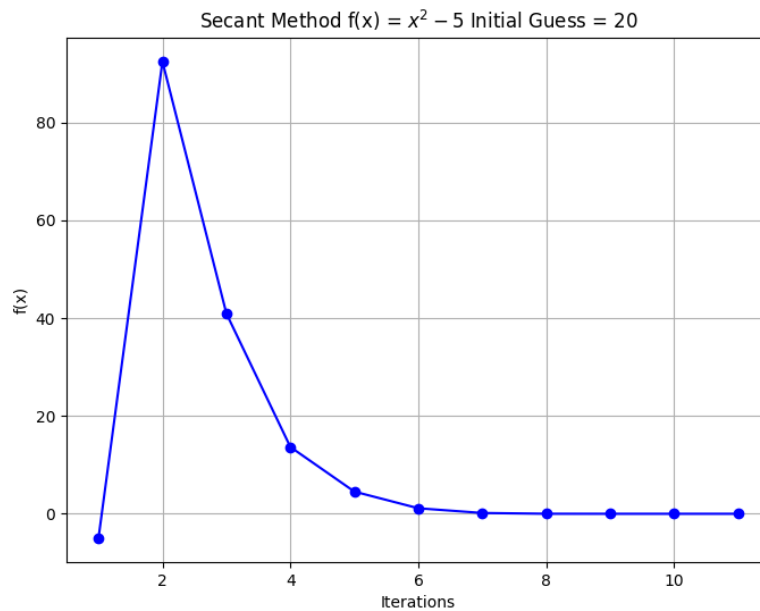


Figura 9:  $f(x)$  ao decorrer das iterações

Percebe-se que ambos os métodos foram suficientes para encontrar as raízes da função. Contudo o método de Newton-Raphson foi mais eficiente em encontrar em menos iterações as raízes da função.

### 1.2.2 Raízes de $f(x) = 5x^3 - 5x - 24 = 0$

Para as seguintes informações iniciais:

```
pedro@Pedro-Lenovo ~/Documentos/GitHub/quanticacomp/lista2
> $ ./L2-5255417-ex-2.exe
Insert x^4 coefficient:
0
Insert x^3 coefficient:
5
Insert x^2 coefficient:
0
Insert x coefficient:
-5
Insert constant coefficient:
-24
Insert initial guess:
5
Root: 1.88367081
Root: 1.88367081
```

Figura 10: Teste feito para gerar o gráfico onde as duas últimas linhas são as raízes encontradas pelos métodos de Newton-Raphson e da secante, respectivamente

Obeve-se o valor da raiz = 1.88367081 para ambos os métodos.

Com o seguinte gráfico para o método de Newton-Raphson:

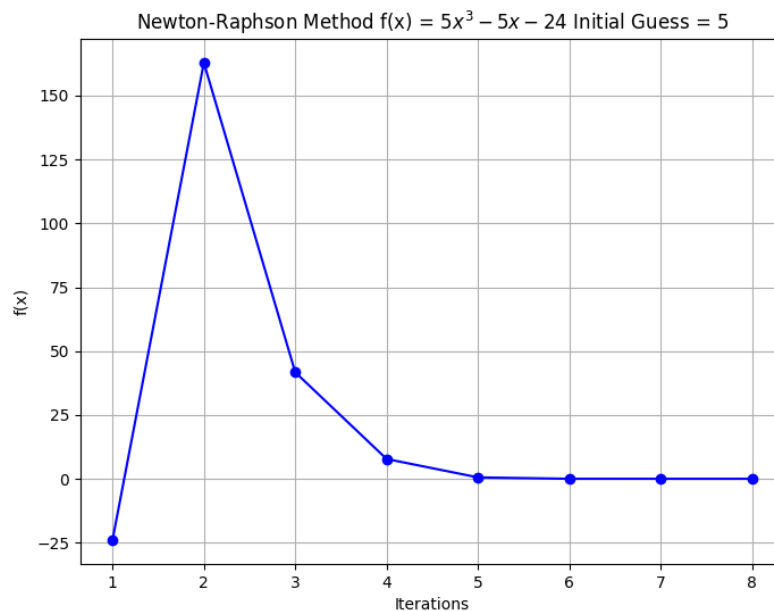


Figura 11:  $f(x)$  ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

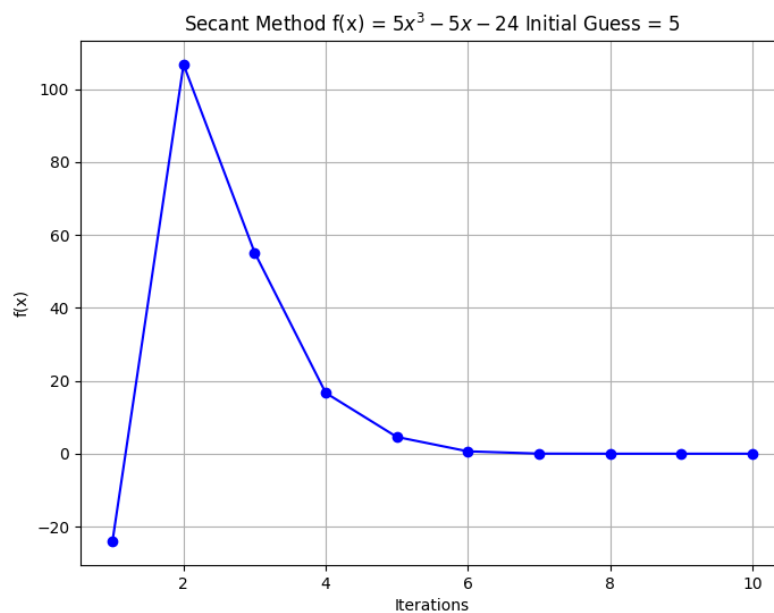


Figura 12:  $f(x)$  ao decorrer das iterações

Para as seguintes informações iniciais:

```

pedro@Pedro-Lenovo ~/Documentos/GitHub/quanticacomp/lista2
> $ ./L2-5255417-ex-2.exe
Insert x^4 coefficient:
0
Insert x^3 coefficient:
5
Insert x^2 coefficient:
0
Insert x coefficient:
-5
Insert constant coefficient:
-24
Insert initial guess:
10
Root: 1.88367081
Root: 1.88367081

```

Figura 13: Teste feito para gerar o gráfico onde as duas últimas linhas são as raízes encontradas pelos métodos de Newton-Raphson e da secante, respectivamente

Obteve-se o valor da raiz = 1.88367081 para ambos os métodos.

Com o seguinte gráfico para o método de Newton-Raphson:

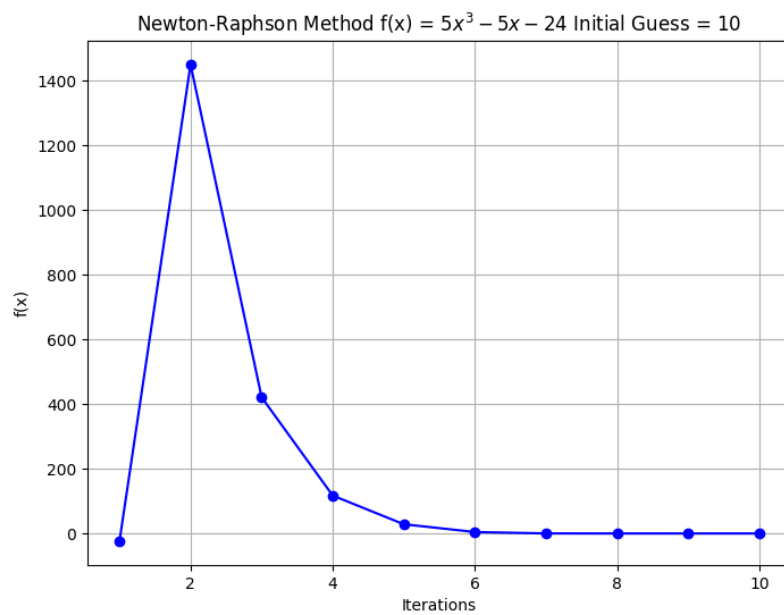


Figura 14:  $f(x)$  ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

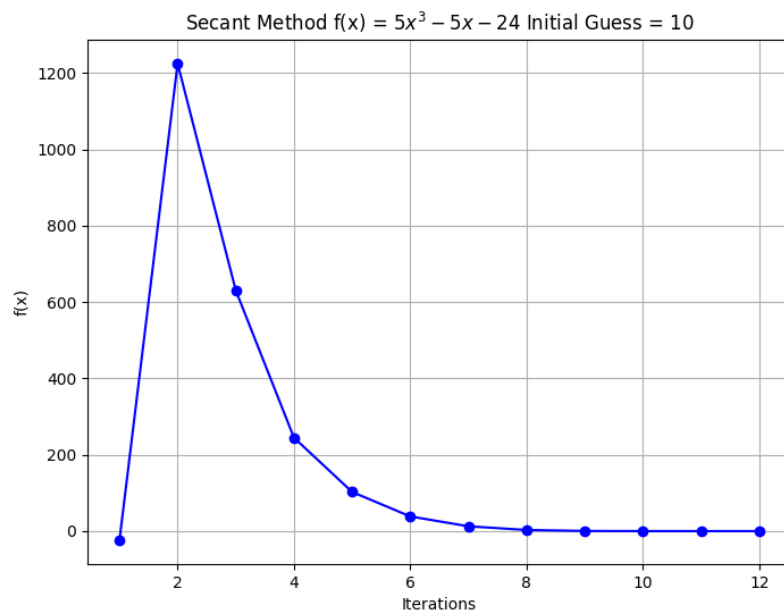


Figura 15:  $f(x)$  ao decorrer das iterações

Para as seguintes informações iniciais:

```

pedro@Pedro-Lenovo ~/Documentos/GitHub/quanticacomp/lista2
> $ ./L2-5255417-ex-2.exe
Insert x^4 coefficient:
0
Insert x^3 coefficient:
5
Insert x^2 coefficient:
0
Insert x coefficient:
-5
Insert constant coefficient:
-24
Insert initial guess:
20
Root: 1.88367081
Root: 1.88367081

```

Figura 16: Teste feito para gerar o gráfico onde as duas últimas linhas são as raízes encontradas pelos métodos de Newton-Raphson e da secante, respectivamente

Obteve-se o valor da raiz = 1.88367081 para ambos os métodos.

Com o seguinte gráfico para o método de Newton-Raphson:

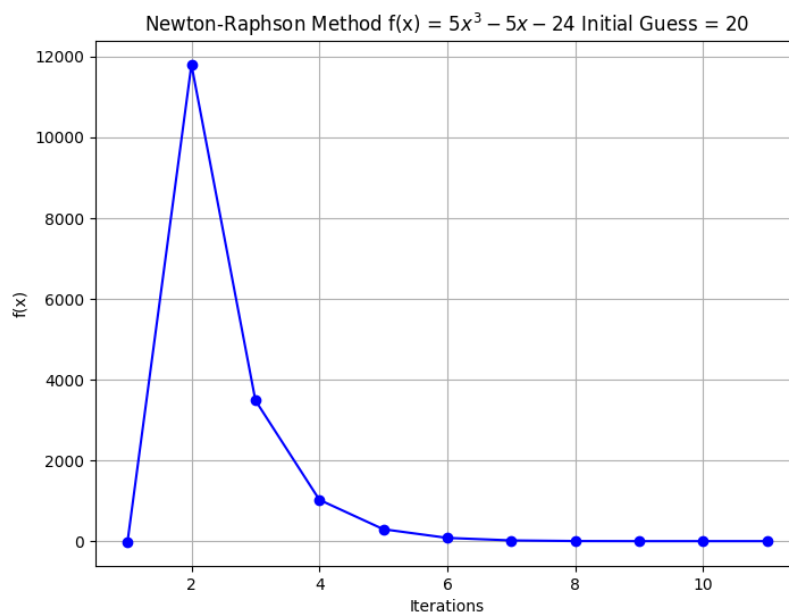


Figura 17:  $f(x)$  ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

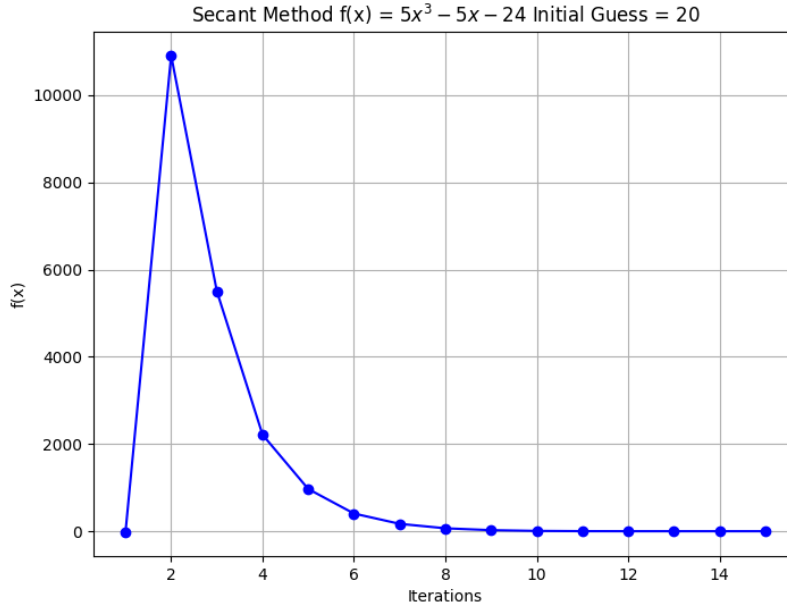


Figura 18:  $f(x)$  ao decorrer das iterações

Percebe-se que ambos os métodos foram suficientes para encontrar as raízes da função. Contudo o método de Newton-Raphson foi mais eficiente em encontrar em menos iterações as raízes da função. Apesar da maior eficiência do método de Newton-Raphson, o método da secante é útil para casos onde a derivada da função é complicada de se calcular - ou até mesmo não existe derivada.

## 2 Eigenvalues of the wave equation

### 2.1 Exercício 3

Tarefa: Escreva a transformação que permitem escrever a equação de Schrödinger para os autoestados de uma partícula em um poço infinito na forma

$$\frac{d^2}{dx^2}\psi(x) = -k^2\psi(x) \quad \text{com} \quad \psi(0) = 0 \text{ e } \psi(\infty) = 0 \quad (11)$$

Seja a equação de Schrödinger:

$$\left( -\frac{\hbar^2}{2m} \frac{d^2}{dx^2} + V(x) \right) \psi(x) = E\psi(x) \quad (12)$$

Para o caso do poço infinito a equação pode ser escrita como:

$$-\frac{\hbar^2}{2m} \frac{d^2}{dx^2} \psi(x) = E\psi(x) \quad \text{com} \quad 0 \leq x \leq L \quad (13)$$



Para tornar adimensional, fazemos a transformação  $x \rightarrow x/L$ :

$$-\frac{\hbar^2}{2mL^2} \frac{d^2}{dx^2} \psi(x) = E\psi(x) \quad \text{com} \quad 0 \leq x \leq 1 \quad (14)$$

Rearranjando:

$$\frac{d^2}{dx^2} \psi(x) = -\frac{2mEL^2}{\hbar^2} \psi(x) \quad \text{com} \quad 0 \leq x \leq 1 \quad (15)$$

Note que  $\frac{2mEL^2}{\hbar^2}$  é adimensional - como desejado. Então, definimos  $k^2 = \frac{2mEL^2}{\hbar^2}$  - obtendo:

$$\frac{d^2}{dx^2} \psi(x) = -k^2 \psi(x) \quad \text{com} \quad \psi(0) = 0 \text{ e } \psi(\infty) = 0 \quad (16)$$

que é a equação adimensional desejada.

## 2.2 Exercício 4

Tarefa: Escreva um código para calcular os primeiros três níveis de energia para o poço de potencial infinito, usando o shooting method e as condições de contorno  $\psi(0) = 0$  e  $\psi'(0) \neq 0$ . Compare o resultado com a solução exata.

Código Escrito:

```

1  ! -----
2  ! File: L2-5255417-ex-4.f90
3  !
4  ! Description:
5  !   Find first three energy levels of quantum 1D infity square well
6  !   using shooting method
7  !
8  ! Dependencies:
9  !   - None
10 !
11 ! Since:
12 !   - 03/2025
13 !
14 ! Authors:
15 !   - Pedro C. Delbem <pedrodelbem@usp.br>
16 ! -----
17 program shooting_method
18
19     !deactivate implicit typing
20     implicit none
21
22     !define variables
23     real deltax, deltak, phi_deltax, dphi_0, k, phi_xminus1, phi_x,
24     phi_xplus1
25     integer i, number_of_iterations
26
27     !define constants
28     real phi_0
29
30     !define phi(0)
31     phi_0 = 0.0

```

```

30
31 write(*,*) "Insert k:"
32 read(*,*) k
33
34 write(*,*) "Insert deltax:"
35 read(*,*) deltax
36
37 write(*,*) "Insert deltak:"
38 read(*,*) deltak
39
40 write(*,*) "Insert phi_deltax (non zero):"
41 read(*,*) phi_deltax
42 do while (phi_deltax == 0.0)
43     write(*,*) "phi_deltax cannot be zero, input again"
44     read(*,*) phi_deltax
45 end do
46
47 write(*,*) "Insert dphi_0 (non zero):"
48 read(*,*) dphi_0
49 do while (dphi_0 == 0.0)
50     write(*,*) "dphi_0 cannot be zero, input again"
51     read(*,*) dphi_0
52 end do
53
54 !define number of iterations
55 number_of_iterations = int(1.0/deltax)
56
57 !update phi
58 phi_xminus1 = phi_0
59 phi_x = phi_deltax
60 phi_xplus1 = 2.0*phi_x - phi_xminus1 - k**2*deltax**2*phi_x
61
62 !do first try
63 !call update_phi(phi_xminus1, phi_x, phi_xplus1, k, deltax,
number_of_iterations)
64
65 !update k until phi(1) >= deltak
66 do while (phi_xplus1 >= deltak)
67
68     !update phi
69     call update_phi(phi_xminus1, phi_x, phi_xplus1, k, deltax,
number_of_iterations)
70     write(*,*) "phi(1) = ", phi_xplus1
71     !update k
72     k = k + deltak
73
74 end do
75
76 write(*,*) "First energy level: ", k
77
78 k = k + 2.0*deltak
79 phi_xminus1 = phi_0
80 phi_x = phi_deltax
81 phi_xplus1 = 2.0*phi_x - phi_xminus1 - k**2*deltax**2*phi_x
82
83 !second level
84 do while (phi_xplus1 >= deltak) !update k until phi(1) >=
deltak

```

```

85
86     !update k
87     k = k + deltak
88
89     !update phi
90     call update_phi(phi_xminus1, phi_x, phi_xplus1, k, deltax,
number_of_iterations)
91
92 end do
93
94 write(*,*) "Second energy level: ", k
95
96 k = k + 2.0*deltak
97 phi_xminus1 = phi_0
98 phi_x = phi_deltax
99 phi_xplus1 = 2.0*phi_x - phi_xminus1 - k**2*deltax**2*phi_x
100
101 !third level
102 do while (phi_xplus1 >= deltak) !update k until phi(1) >=
deltak
103
104     !update k
105     k = k + deltak
106
107     !update phi
108     call update_phi(phi_xminus1, phi_x, phi_xplus1, k, deltax,
number_of_iterations)
109
110 end do
111
112 write(*,*) "Third energy level: ", k
113
114 contains
115
116 subroutine update_phi(phi_xminus1, phi_x, phi_xplus1, k, deltax
, number_of_iterations)
117
118     !deactivate implicit typing
119     implicit none
120
121     !define variables
122     real, intent(inout) :: phi_xminus1, phi_x, phi_xplus1, k,
deltax
123     integer, intent(in) :: number_of_iterations
124
125     do i = 1, number_of_iterations
126         phi_xminus1 = phi_x
127         phi_x = phi_xplus1
128         phi_xplus1 = 2.0*phi_x - phi_xminus1 - k**2*deltax**2*
phi_x
129     end do
130
131 end subroutine update_phi
132
133 end program shooting_method

```

Resultados:

## 2.3 Exercício 5

Tarefa: Escreva um código para calcular os primeiros três níveis de energia para o poço de potencial infinito, usando o método da secante e as condições de contorno  $\psi(0) = 0$  e  $\psi'(0) \neq 0$ . Compare o resultado com a solução exata e com o resultado do exercício 4.

Código Escrito:

```
1 | -----  
2 | File: L2-5255417-ex-5.f90  
3 |  
4 | Description:  
5 |  
6 |  
7 | Dependencies:  
8 |   - None  
9 |  
10 | Since:  
11 |   - 03/2025  
12 |  
13 | Authors:  
14 |   - Pedro C. Delbem <pedrodelbem@usp.br>  
15 | -----
```

Resultados: