

Universidade de São Paulo

Instituto de Física de São Carlos

Lista 2

Pedro Calligaris Delbem 5255417

Professor: Attilio Cucchieri

Março de 2025

Sumário

1	Finding roots	2
1.1	Exercício 1	2
1.2	Exercício 2	5
1.2.1	Raizes de $f(x) = x^2 - 5 = 0$	8
1.2.2	Raizes de $f(x) = 5x^3 - 5x - 24 = 0$	13
2	Eigenvalues of the wave equation	17
2.1	Exercício 1	17
2.2	Exercício 2	18
2.3	Exercício 3	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.

Código Escrito:

```
1 ! -----
2 ! File: L2-5255417-ex-1.f90
3 !
4 ! Description:
5 !   Computes Newton-Raphson method convergence
6 !
7 ! Dependencies:
8 !   - None
9 !
10 ! Since:
11 !   - 03/2025
12 !
13 ! Authors:
14 !   - Pedro C. Delbem <pedrodelbem@usp.br>
15 ! -----
16 program newton_raphson
17
18     !deactivate implicit typing
19     implicit none
20
21     !define variables
22     real x_k, x_kplus1, f_x_k, df_x_k, f_x(5)
23     integer iteration
24
25     write(*,*) "Insert x^4 coefficient:"
26     read(*,*) f_x(5)
27     write(*,*) "Insert x^3 coefficient:"
28     read(*,*) f_x(4)
29     write(*,*) "Insert x^2 coefficient:"
30     read(*,*) f_x(3)
31     write(*,*) "Insert x coefficient:"
32     read(*,*) f_x(2)
33     write(*,*) "Insert constant coefficient:"
34     read(*,*) f_x(1)
35
36     !initialize variables
37     x_k = 10.0
38     x_kplus1 = 0.0
39     f_x_k = f(x_k,f_x)
40     df_x_k = df(x_k,f_x)
41     iteration = 1
42
43     !open output file
```

```

44  open(unit=1, file='output.txt', action='write')
45
46  !print first iteration
47  write(1,*) iteration, f(x_k,f_x)
48
49  !update f_x_k and df_x_k
50  f_x_k = f(x_k,f_x)
51  df_x_k = df(x_k,f_x)
52
53  !update x_kplus1
54  x_kplus1 = x_k - f_x_k/df_x_k
55
56  !update iteration
57  iteration = iteration + 1
58
59  !print second iteration
60  write(1,*) iteration, f(x_kplus1,f_x)
61
62  !Newton-Raphson method
63  do while (abs(x_kplus1 - x_k) > 1e-6)
64
65      !update x_k
66      x_k = x_kplus1
67
68      !update f_x_k and df_x_k
69      f_x_k = f(x_k,f_x)
70      df_x_k = df(x_k,f_x)
71
72      !update x_kplus1
73      x_kplus1 = x_k - f_x_k/df_x_k
74
75      !update iteration
76      iteration = iteration + 1
77
78      !print iteration
79      write(1,*) iteration, f(x_kplus1,f_x)
80
81  end do
82
83  !print root
84  write(*,*) 'Root:', x_kplus1
85
86  !close output file
87  close(1)
88
89  contains
90
91  function f(x,f_x) result(result)
92      real, intent(in) :: x
93      real, intent(in) :: f_x(5)
94      real result
95
96      result = f_x(5)*x**4. + f_x(4)*x**3. + f_x(3)*x**2. + f_x
(2)*x + f_x(1)
97  end function f
98
99  function df(x,f_x) result(result)
100     real, intent(in) :: x

```

```

pedro@Pedro-Lenovo ~/Documentos/GitHub/quanticacomp/lista2
> $ ./L2-5255417-ex-1.exe
Insert x^4 coefficient:
0
Insert x^3 coefficient:
0
Insert x^2 coefficient:
3
Insert x coefficient:
0
Insert constant coefficient:
-10
Root: 1.82574189

```

Figura 1: Teste feito para gerar o gráfico

```

101     real, intent(in) :: f_x(5)
102     real result
103
104     result = 4.*f_x(5)*x**3. + 3.*f_x(4)*x**2. + 2.*f_x(3)*x +
f_x(2)
105     end function df
106
107 end program newton_raphson

```

Resultados:

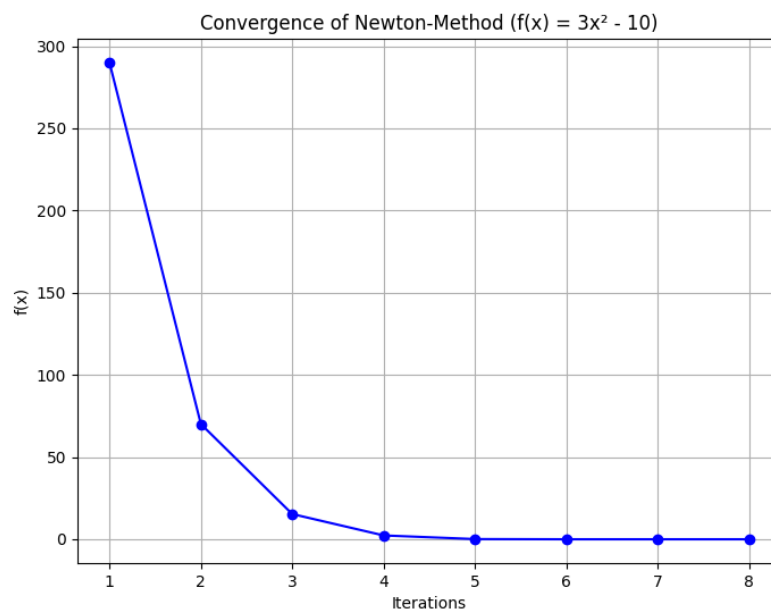


Figura 2: $f(x)$ ao decorrer das iterações

Testando para o caso $f(x) = 3x^2 - 10 = 0$ percebe-se - pelo gráfico obtido - que o método Newton-Raphson converge quadraticamente.

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 3: 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:

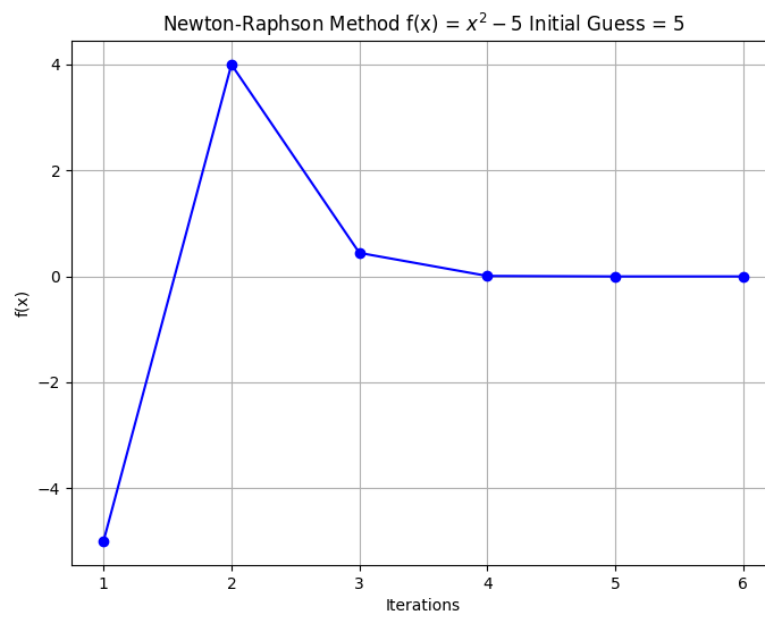


Figura 4: $f(x)$ ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

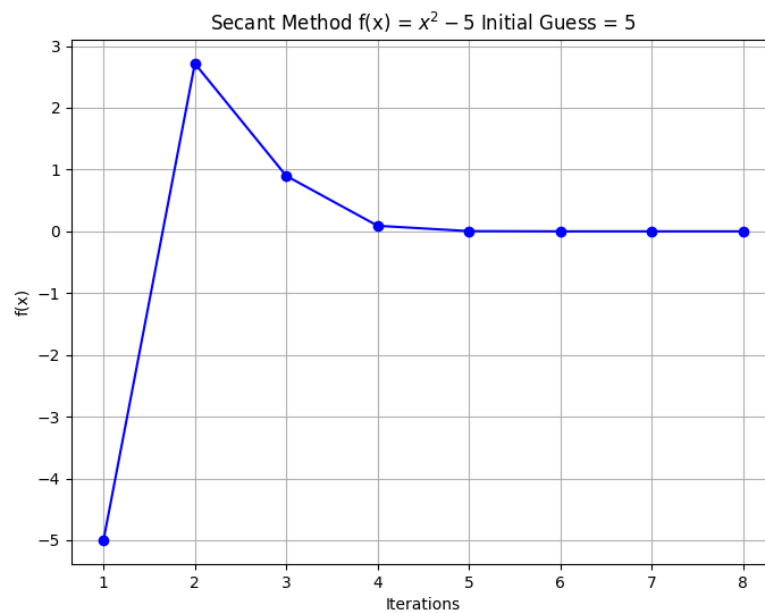


Figura 5: $f(x)$ ao decorrer das iterações

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 6: 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:

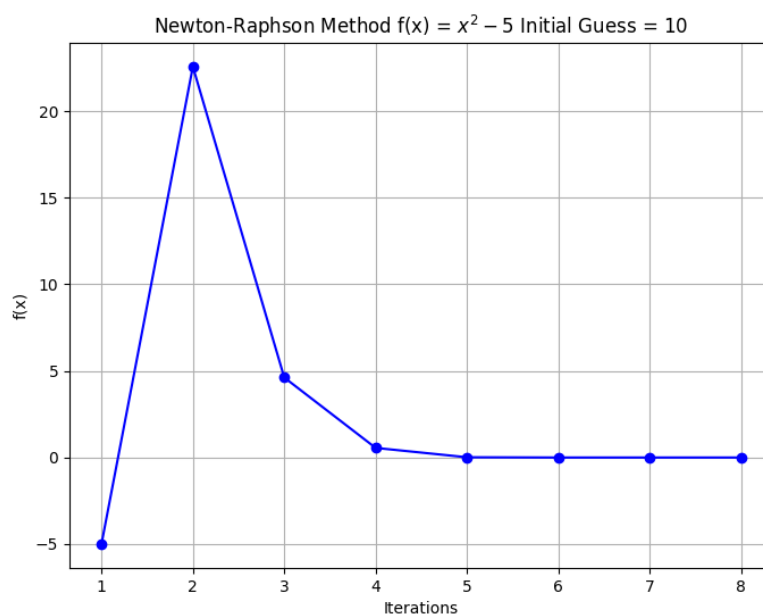


Figura 7: $f(x)$ ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

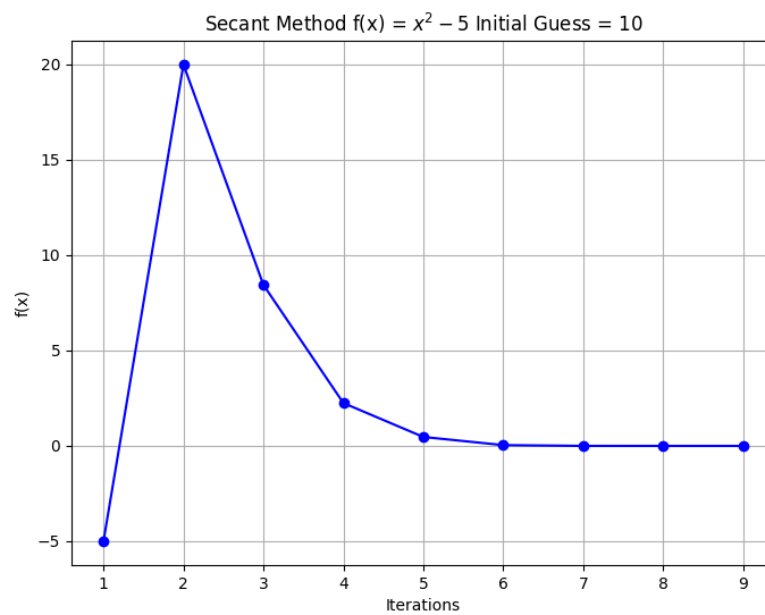


Figura 8: $f(x)$ ao decorrer das iterações

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 9: 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:

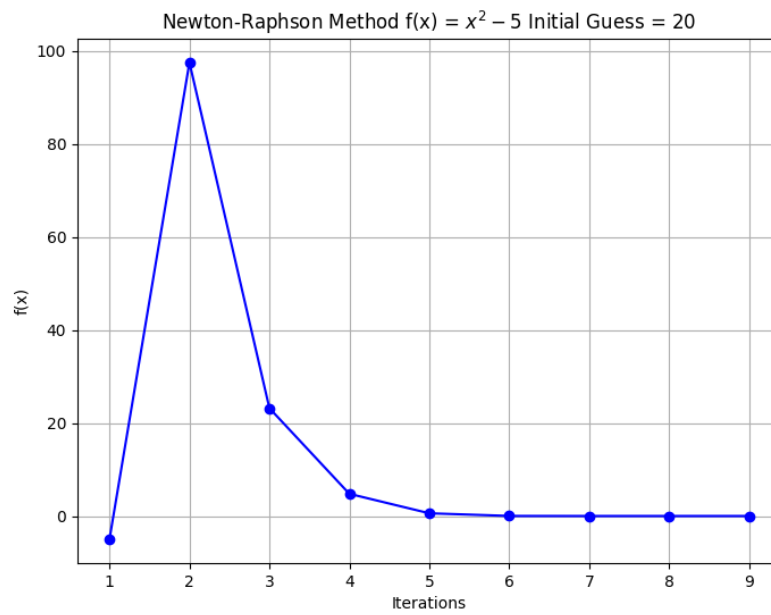


Figura 10: $f(x)$ ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

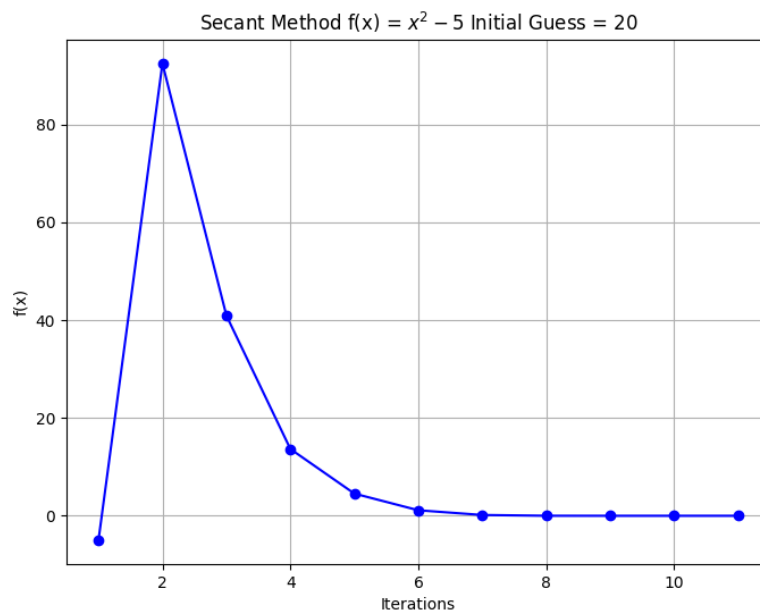


Figura 11: $f(x)$ ao decorrer das iterações

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 12: 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:

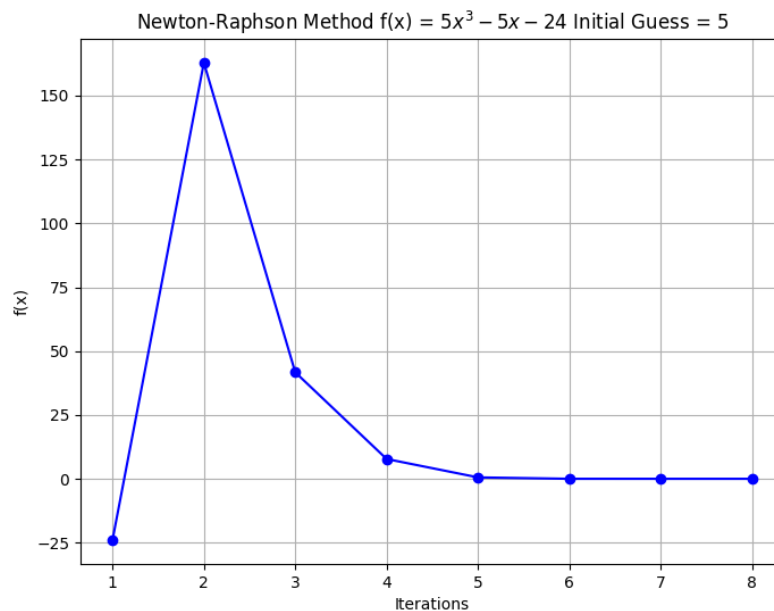


Figura 13: $f(x)$ ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

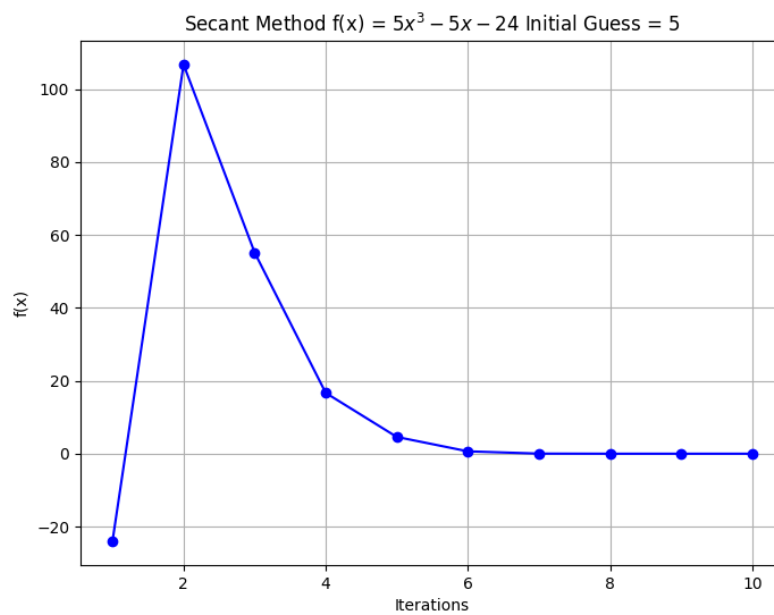


Figura 14: $f(x)$ ao decorrer das iterações

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 15: 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:

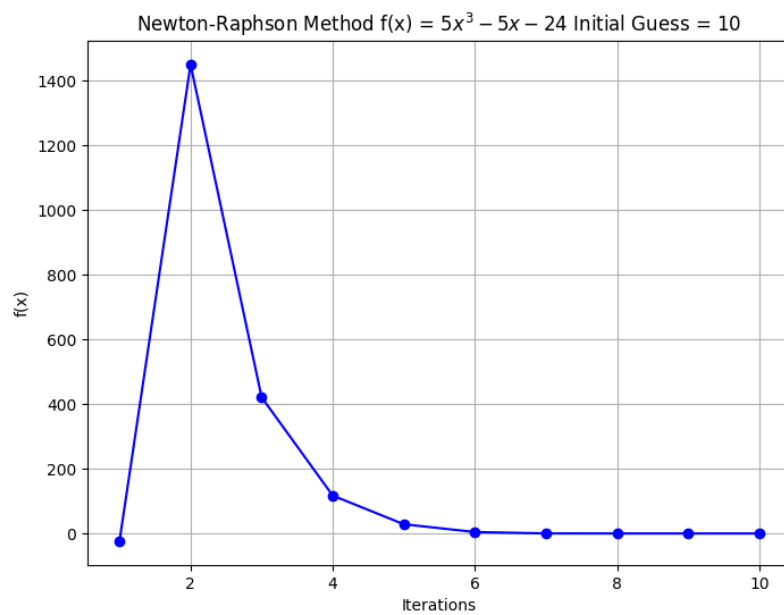


Figura 16: $f(x)$ ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

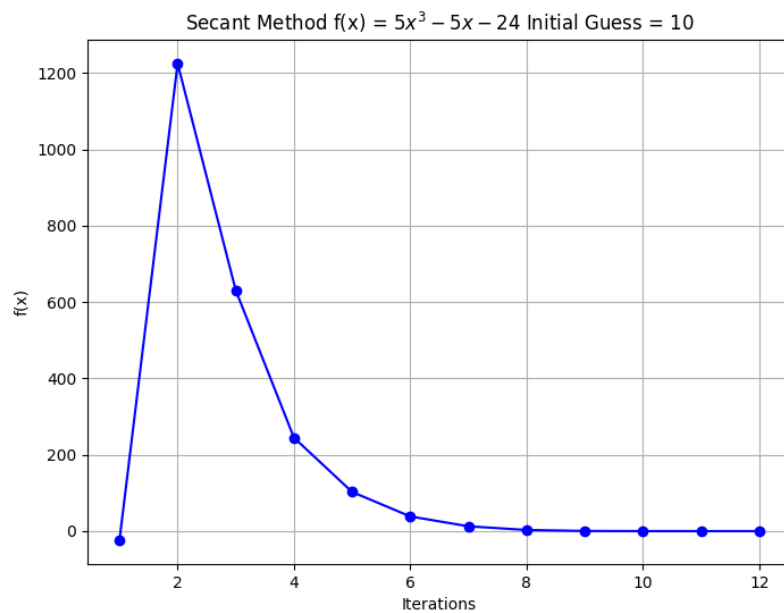


Figura 17: $f(x)$ ao decorrer das iterações

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 18: 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:

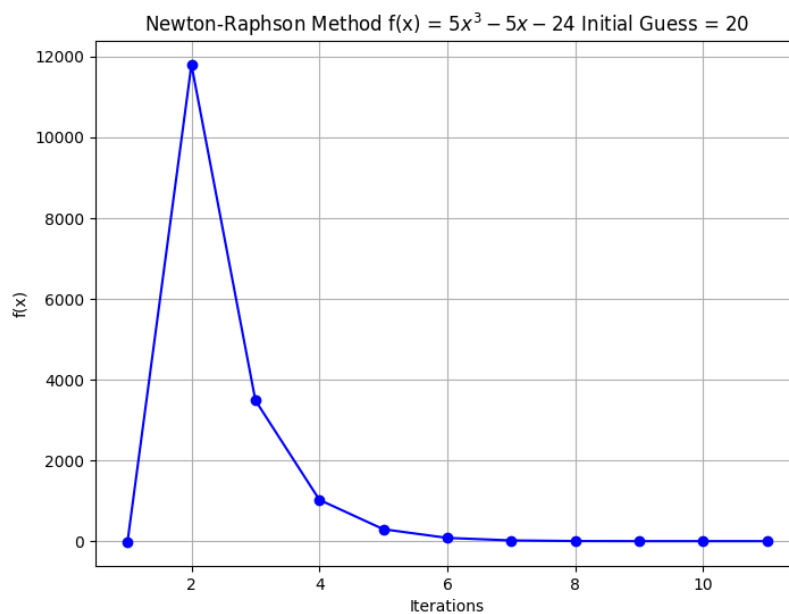


Figura 19: $f(x)$ ao decorrer das iterações

Com o seguinte gráfico para o método da secante:

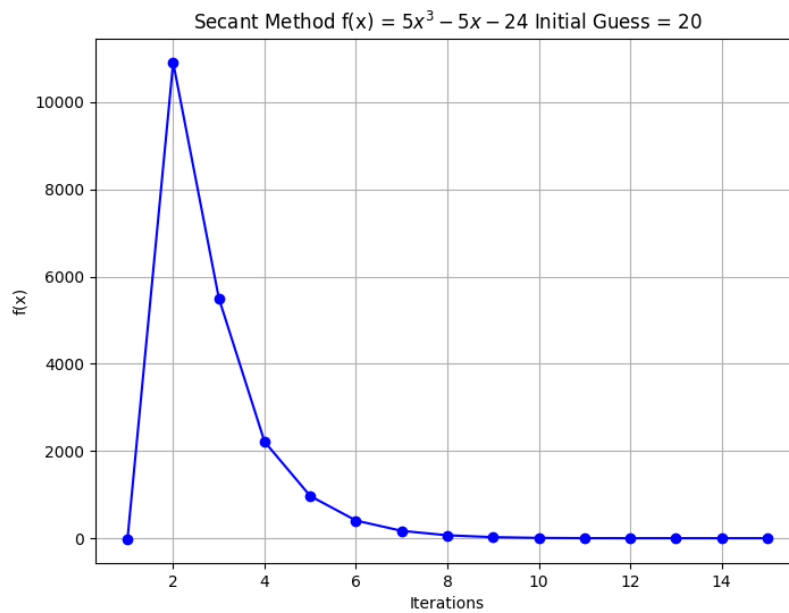


Figura 20: $f(x)$ ao decorrer das iterações

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.

2 Eigenvalues of the wave equation

2.1 Exercício 1

Tarefa: Achar a precisão do computador, i.e., o maior número positivo tal que $1 + \epsilon = 1$, usando precisão simples e dupla.

Código Escrito:

```

1 ! -----
2 ! File: L2-5255417-ex-3.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>

```

Resultados:

Nota-se que os valores obtidos correspondem aos valores esperados.

2.2 Exercício 2

Tarefa: Calcular

$$e^{-x} = 1 - x + x^2/2! - x^3/3! + \dots \quad (2)$$

para $x = 0.1, 1, 10, 100$ e 1000 com um erro menor do que 10^{-8} . Problema: quando truncar a série? É preciso calcular o fatorial explicitamente? Comparar o valor obtido usando a série com o resultado exato.

Código Escrito:

```

1  ! -----
2  ! File: L2-5255417-ex-4.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:

Descrição: O código faz um loop onde, em cada iteração, calcula o valor da série para um valor de x diferente. Na subrotina "compute-exponencial" calcula-se a série definindo o próximo termo como a multiplicação do termo anterior por $-x/n$, pois - deste modo - não se faz necessário calcular o fatorial explicitamente. Ademais, interrompe-se a soma quando o termo atual for menor que 10^{-8} garantindo a precisão desejada, uma vez que cada termo da série é menor - em módulo - que o anterior.

Por fim, percebe-se que o resultado esperado foi obtido para todos os casos testados - com exceção dos casos onde o resultado é menor que a precisão.

2.3 Exercício 3

Tarefa: Considerar a somatória

$$\Sigma(N) = \sum_{n=1}^{2N} (-1)^n \frac{n}{n+1} = - \sum_{n=1}^N \frac{2n-1}{2n} + \sum_{n=1}^N \frac{2n}{2n+1} = \sum_{n=1}^N \frac{1}{2n(2n+1)} \quad (3)$$

e calcular $\Sigma(N)$ para $N = 1, 2, \dots, 10^6$ usando as três fórmulas acima. Comparar os resultados usando precisões simples.

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:

Nota-se que a primeira série demora é mais instável do que as demais. Além disso, a terceira série se mostra mais estável que a segunda - sendo então a melhor versão.