

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	4
2	Eigenvalues of the wave equation	7
2.1	Exercício 1	7
2.2	Exercício 2	7
2.3	Exercício 3	8

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(unit=1, file='output.txt', action='write')
```

```

44
45     f_x_k = f(x_k,f_x)
46     df_x_k = df(x_k,f_x)
47     x_kplus1 = x_k - f_x_k/df_x_k
48     iteration = iteration + 1
49     write(1,*) iteration, f(x_kplus1,f_x)
50
51     !Newton-Raphson method
52     do while (abs(x_kplus1 - x_k) > 1e-6)
53         x_k = x_kplus1
54         f_x_k = f(x_k,f_x)
55         df_x_k = df(x_k,f_x)
56         x_kplus1 = x_k - f_x_k/df_x_k
57         iteration = iteration + 1
58         write(1,*) iteration, f(x_kplus1,f_x)
59     end do
60
61     close(1)
62
63 contains
64
65     function f(x,f_x) result(result)
66         real, intent(in) :: x
67         real, intent(in) :: f_x(5)
68         real result
69
70         result = f_x(5)*x**4. + f_x(4)*x**3. + f_x(3)*x**2. + f_x
71         (2)*x + f_x(1)
72     end function f
73
74     function df(x,f_x) result(result)
75         real, intent(in) :: x
76         real, intent(in) :: f_x(5)
77         real result
78
79         result = 4.*f_x(5)*x**3. + 3.*f_x(4)*x**2. + 2.*f_x(3)*x +
80         f_x(2)
81     end function df
82
83 end program newton_raphson

```

Resultados:

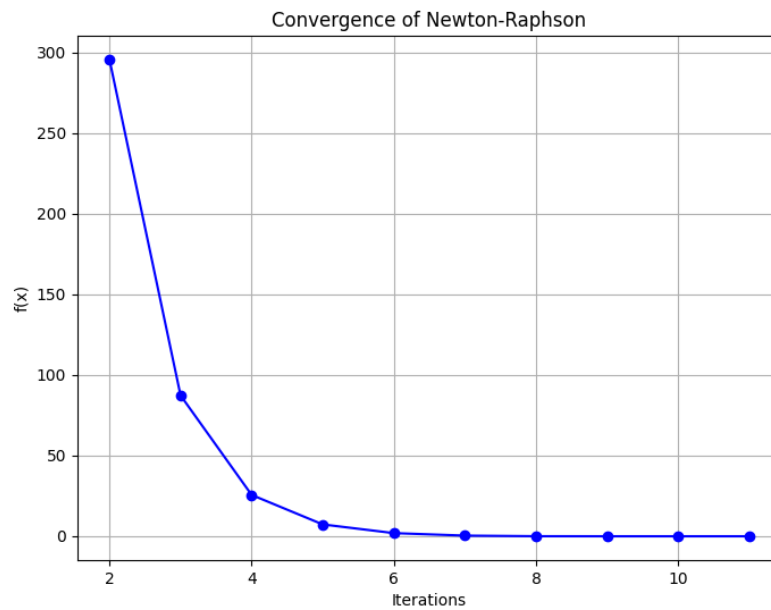


Figura 1: $f(x)$ ao decorrer das iterações

Percebe-se 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
(5), initial_guess
23 integer iteration
24
25 !request coefficients
26 write(*,*) "Insert x^4 coefficient:"
27 read(*,*) f_x(5)
28 write(*,*) "Insert x^3 coefficient:"
29 read(*,*) f_x(4)
30 write(*,*) "Insert x^2 coefficient:"
31 read(*,*) f_x(3)
32 write(*,*) "Insert x coefficient:"
33 read(*,*) f_x(2)
34 write(*,*) "Insert constant coefficient:"
35 read(*,*) f_x(1)
36
37 !request initial guess
38 write(*,*) "Insert initial guess:"
39 read(*,*) initial_guess
40
41 !initialize variables
42 x_k = initial_guess
43 x_kplus1 = 0.0
44 f_x_k = f(x_k,f_x)
45 df_x_k = df(x_k,f_x)
46 iteration = 1
47
48 open(unit=1, file='output1.txt', action='write')
49
50 f_x_k = f(x_k,f_x)
51 df_x_k = df(x_k,f_x)
52 x_kplus1 = x_k - f_x_k/df_x_k
53 iteration = iteration + 1
54 write(1,*) iteration, f(x_kplus1,f_x)
55
56 !Newton-Raphson method
57 do while (abs(x_kplus1 - x_k) > 1e-6)
58     x_k = x_kplus1
59     f_x_k = f(x_k,f_x)
60     df_x_k = df(x_k,f_x)
61     x_kplus1 = x_k - f_x_k/df_x_k
62     iteration = iteration + 1
63     write(1,*) iteration, f(x_kplus1,f_x)
64 end do
65
66 close(1)
67
68 !reinitialize variables
69 x_kminus1 = initial_guess - 1.0
70 x_k = initial_guess
71 x_kplus1 = 0.0
72 f_x_kminus1 = f(x_kminus1,f_x)
73 f_x_k = f(x_k,f_x)
74 iteration = 1
75
76 open(unit=2, file='output2.txt', action='write')

```

```

77
78     f_x_kminus1 = f(x_kminus1,f_x)
79     f_x_k = f(x_k,f_x)
80     x_kminus1 = x_k
81     x_kplus1 = x_k - f_x_k*(x_k - x_kminus1)/(f_x_k - f_x_kminus1)
82     iteration = iteration + 1
83     write(2,*) iteration, f(x_kplus1,f_x)
84
85     !Secant method
86     do while (abs(x_kplus1 - x_k) > 1e-6)
87         x_k = x_kplus1
88         f_x_kminus1 = f(x_kminus1,f_x)
89         f_x_k = f(x_k,f_x)
90         x_kminus1 = x_k
91         x_kplus1 = x_k - f_x_k*(x_k - x_kminus1)/(f_x_k -
f_x_kminus1)
92         iteration = iteration + 1
93         write(2,*) iteration, f(x_kplus1,f_x)
94     end do
95
96     close(2)
97
98 contains
99
100     function f(x,f_x) result(result)
101         real, intent(in) :: x
102         real, intent(in) :: f_x(5)
103         real result
104
105         result = f_x(5)*x**4. + f_x(4)*x**3. + f_x(3)*x**2. + f_x
(2)*x + f_x(1)
106     end function f
107
108     function df(x,f_x) result(result)
109         real, intent(in) :: x
110         real, intent(in) :: f_x(5)
111         real result
112
113         result = 4.*f_x(5)*x**3. + 3.*f_x(4)*x**2. + 2.*f_x(3)*x +
f_x(2)
114     end function df
115
116 end program find_roots

```

Resultados:

Nota-se que os valores obtidos são coerentes com o esperado.

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>
15 !-----
```

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 execução 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 precisão 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.