

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	3
2	Eigenvalues of the wave equation	11
2.1	Exercício 3	11
2.2	Exercício 4	12
2.3	Exercício 5	15

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,
23     initial_guess, pre_initial_guess
24     integer iteration
25
26     !request initial guess
27     write(*,*) "Insert initial guess:"
28     read(*,*) initial_guess
29
30     !open first output file
31     open(unit=1, file='newtonraphson1.txt', action='write')
32
33     !initialize variables
34     x_k = initial_guess
35     x_kplus1 = 0.0
36     f_x_k = f1(x_k)
37     df_x_k = df1(x_k)
38     iteration = 1
39
40     !print header
41     write(1,*) 'Newton-Raphson Method'
42     write(1,*) 'Initial guess: ', x_k
43     write(1,*) 'f(x) = x^2 - 5'
44
45     !print first iteration
46     write(1,*) iteration, f1(x_kplus1)
47
48     !update f_x and df_x
49     f_x_k = f1(x_k)
```

```

49     df_x_k = df1(x_k)
50
51     !update x_kplus1
52     x_kplus1 = x_k - f_x_k/df_x_k
53
54     !update iteration
55     iteration = iteration + 1
56
57     !print second iteration
58     write(1,*) iteration, f1(x_kplus1)
59
60     !Newton-Raphson method
61     do while (abs(x_kplus1 - x_k) > 1e-6)
62
63         !update x_k
64         x_k = x_kplus1
65
66         !update f_x and df_x
67         f_x_k = f1(x_k)
68         df_x_k = df1(x_k)
69
70         !update x_kplus1
71         x_kplus1 = x_k - f_x_k/df_x_k
72
73         !update iteration
74         iteration = iteration + 1
75
76         !print iteration
77         write(1,*) iteration, f1(x_kplus1)
78
79     end do
80
81     !print root
82     write(1,*) 'Root:', x_kplus1
83
84     !close first output file
85     close(1)
86
87     !open first output file
88     open(unit=2, file='newtonraphson2.txt', action='write')
89
90     !initialize variables
91     x_k = initial_guess
92     x_kplus1 = 0.0
93     f_x_k = f2(x_k)
94     df_x_k = df2(x_k)
95     iteration = 1
96
97     !print header
98     write(2,*) 'Newton-Raphson Method'
99     write(2,*) 'Initial guess: ', x_k
100    write(2,*) 'f(x) = 5x^3 - 5x - 24'
101
102    !print first iteration
103    write(2,*) iteration, f2(x_kplus1)
104
105    !update f_x and df_x
106    f_x_k = f2(x_k)

```

```

107     df_x_k = df2(x_k)
108
109     !update x_kplus1
110     x_kplus1 = x_k - f_x_k/df_x_k
111
112     !update iteration
113     iteration = iteration + 1
114
115     !print second iteration
116     write(2,*) iteration, f2(x_kplus1)
117
118     !Newton-Raphson method
119     do while (abs(x_kplus1 - x_k) > 1e-6)
120
121         !update x_k
122         x_k = x_kplus1
123
124         !update f_x and df_x
125         f_x_k = f2(x_k)
126         df_x_k = df2(x_k)
127
128         !update x_kplus1
129         x_kplus1 = x_k - f_x_k/df_x_k
130
131         !update iteration
132         iteration = iteration + 1
133
134         !print iteration
135         write(2,*) iteration, f2(x_kplus1)
136
137     end do
138
139     !print root
140     write(2,*) 'Root:', x_kplus1
141
142     !close first output file
143     close(2)
144
145     !request initial guess
146     write(*,*) "Insert pre-initial guess:"
147     read(*,*) pre_initial_guess
148
149     !open second output file
150     open(unit=3, file='secant1.txt', action='write')
151
152     !reinitialize variables
153     x_kminus1 = pre_initial_guess
154     x_k = initial_guess
155     x_kplus1 = 0.0
156     f_x_kminus1 = f1(x_kminus1)
157     f_x_k = f1(x_k)
158     iteration = 1
159
160     !print header
161     write(3,*) 'Secant Method'
162     write(3,*) 'Initial guess: ', x_k
163     write(3,*) 'Pre-initial guess: ', x_kminus1
164     write(3,*) 'f(x) = x^2 - 5'

```

```

165
166     !print first iteration
167     write(3,*) iteration, f1(x_kplus1)
168
169     !update f_x_k and f_x_k-1
170     f_x_kminus1 = f1(x_kminus1)
171     f_x_k = f1(x_k)
172
173     !update x_kplus1 and x_kminus1
174     x_kplus1 = x_k - f_x_k*(x_k - x_kminus1)/(f_x_k - f_x_kminus1)
175     x_kminus1 = x_k
176
177     !update iteration
178     iteration = iteration + 1
179
180     !print second iteration
181     write(3,*) iteration, f1(x_kplus1)
182
183     !Secant method
184     do while (abs(x_kplus1 - x_k) > 1e-6)
185
186         !update x_k
187         x_k = x_kplus1
188
189         !update f_x_k and f_x_k-1
190         f_x_kminus1 = f1(x_kminus1)
191         f_x_k = f1(x_k)
192
193         !update x_kplus1 and x_kminus1
194         x_kplus1 = x_k - f_x_k*(x_k - x_kminus1)/(f_x_k -
f_x_kminus1)
195         x_kminus1 = x_k
196
197         !update iteration
198         iteration = iteration + 1
199
200         !print iteration
201         write(3,*) iteration, f1(x_kplus1)
202
203     end do
204
205     !print root
206     write(3,*) 'Root:', x_kplus1
207
208     !close second output file
209     close(3)
210
211     !open second output file
212     open(unit=4, file='secant2.txt', action='write')
213
214     !reinitialize variables
215     x_kminus1 = pre_initial_guess
216     x_k = initial_guess
217     x_kplus1 = 0.0
218     f_x_kminus1 = f2(x_kminus1)
219     f_x_k = f2(x_k)
220     iteration = 1
221

```

```

222 !print header
223 write(4,*) 'Secant Method'
224 write(4,*) 'Initial guess: ', x_k
225 write(4,*) 'Pre-initial guess: ', x_kminus1
226 write(4,*) 'f(x) = 5x^3 - 5x - 24'
227
228 !print first iteration
229 write(4,*) iteration, f2(x_kplus1)
230
231 !update f_x_k and f_x_k-1
232 f_x_kminus1 = f2(x_kminus1)
233 f_x_k = f2(x_k)
234
235 !update x_kplus1 and x_kminus1
236 x_kplus1 = x_k - f_x_k*(x_k - x_kminus1)/(f_x_k - f_x_kminus1)
237 x_kminus1 = x_k
238
239 !update iteration
240 iteration = iteration + 1
241
242 !print second iteration
243 write(4,*) iteration, f2(x_kplus1)
244
245 !Secant method
246 do while (abs(x_kplus1 - x_k) > 1e-6)
247
248     !update x_k
249     x_k = x_kplus1
250
251     !update f_x_k and f_x_k-1
252     f_x_kminus1 = f2(x_kminus1)
253     f_x_k = f2(x_k)
254
255     !update x_kplus1 and x_kminus1
256     x_kplus1 = x_k - f_x_k*(x_k - x_kminus1)/(f_x_k -
f_x_kminus1)
257     x_kminus1 = x_k
258
259     !update iteration
260     iteration = iteration + 1
261
262     !print iteration
263     write(4,*) iteration, f2(x_kplus1)
264
265 end do
266
267 !print root
268 write(4,*) 'Root:', x_kplus1
269
270 !close second output file
271 close(4)
272
273 contains
274
275 function f1(x) result(result)
276     real, intent(in) :: x
277     real result
278

```



```

279     result = x**2. - 5.
280 end function f1
281
282 function f2(x) result(result)
283     real, intent(in) :: x
284     real result
285
286     result = 5.*x**3. - 5.*x - 24.
287 end function f2
288
289 function df1(x) result(result)
290     real, intent(in) :: x
291     real result
292
293     result = 2.*x
294 end function df1
295
296 function df2(x) result(result)
297     real, intent(in) :: x
298     real result
299
300     result = 15.*x**2. - 5.
301 end function df2
302
303 end program find_roots

```

O código foi compilado com o comando:

```
gfortran L2-5255417-ex-2.f90 -o L2-5255417-ex-2.exe
```

Resultados:

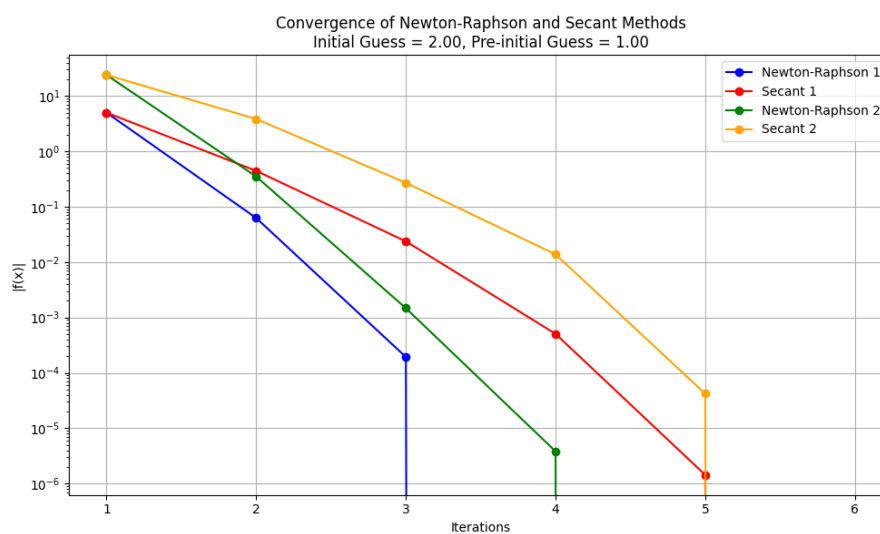


Figura 1

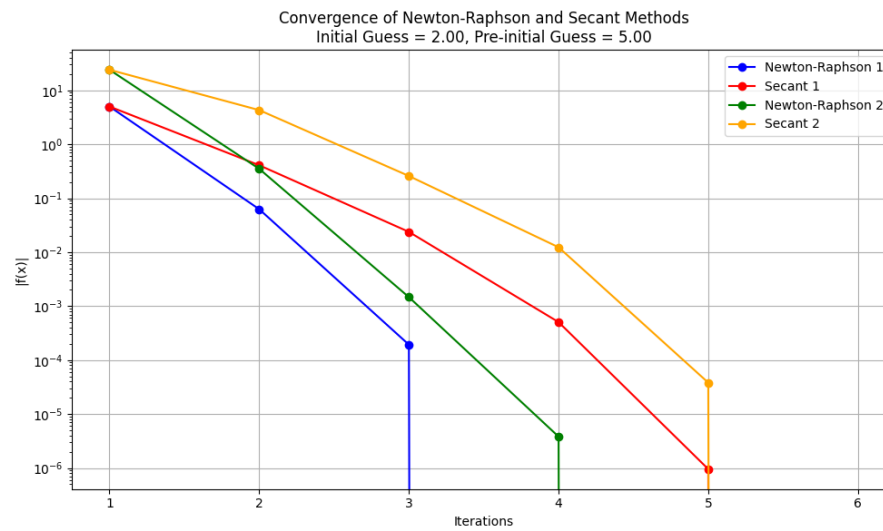


Figura 2

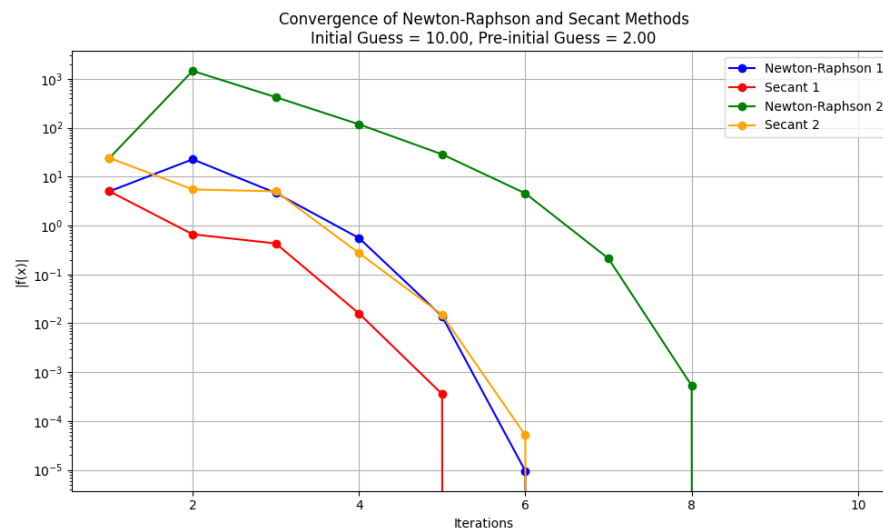


Figura 3

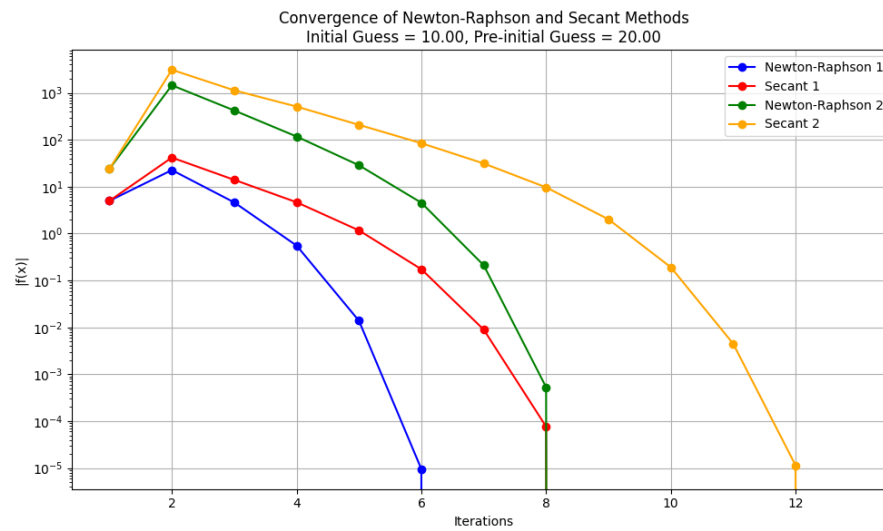


Figura 4

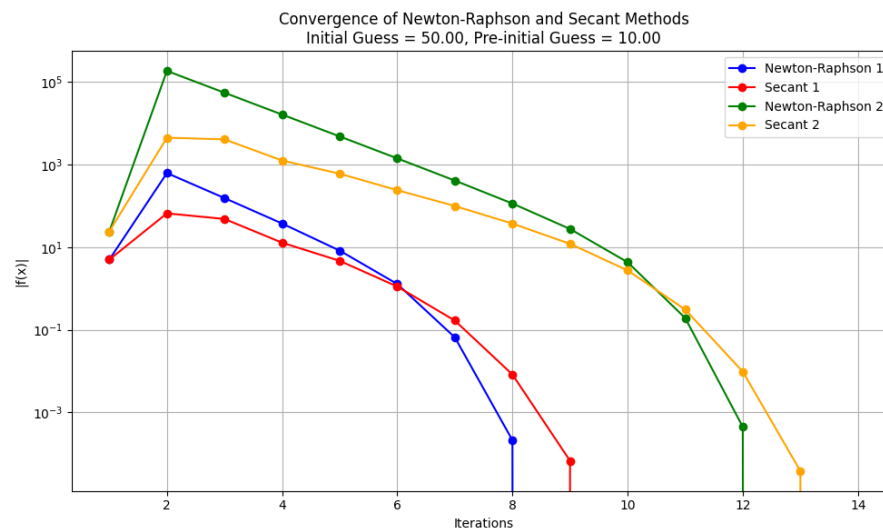


Figura 5

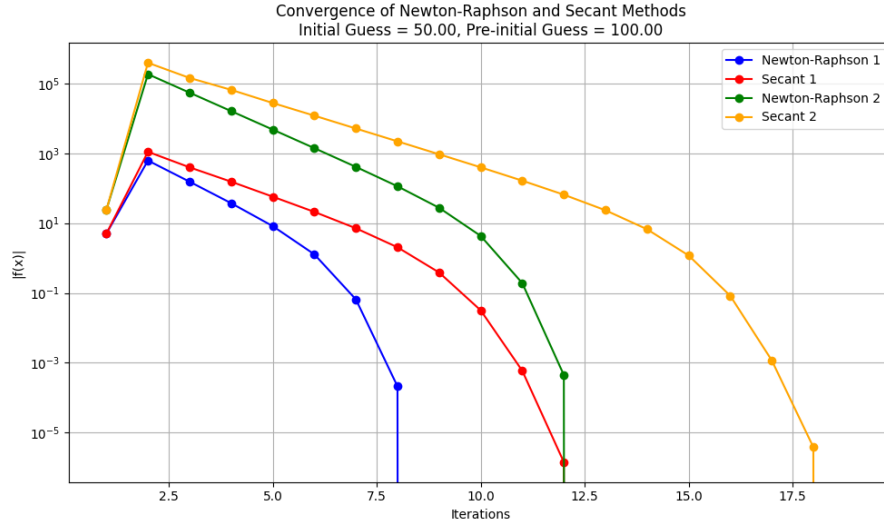


Figura 6

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, x
25     integer i, number_of_iterations
26
27     !define constants
28     real phi_0
29
30     !define phi(0)
31     phi_0 = 0.0
32
33     !initialize x
34     x = 0.0
35
36     write(*,*) "Insert the number of iterations:"
37     read(*,*) number_of_iterations
38
39     write(*,*) "Insert k:"
40     read(*,*) k
41
42     write(*,*) "Insert deltak:"
43     read(*,*) deltak
44
45     write(*,*) "Insert phi_deltax (non zero):"
46     read(*,*) phi_deltax
47     do while (phi_deltax == 0.0)
48         write(*,*) "phi_deltax cannot be zero, input again"
49         read(*,*) phi_deltax
50     end do
```

```

48     end do
49
50     !define deltax
51     deltax = 1.0/number_of_iterations
52     write(*,*) "deltax: ", deltax
53
54     !initialize phi
55     phi_x = 1.0
56
57     !update k until phi(1) >= deltak
58     do while (phi_x >= deltak)
59
60         !do the first iteration
61         phi_xminus1 = phi_0
62         phi_x = phi_deltax
63         phi_xplus1 = 2.0*phi_x - phi_xminus1 - (k**2.0)*(deltax
**2.0)*phi_x
64         x = deltax
65
66         !update phi
67         call update_phi(phi_xminus1, phi_x, phi_xplus1, k, deltax,
number_of_iterations, x)
68
69         !update k
70         k = k + deltak
71
72     end do
73
74     write(*,*) "First energy level: ", k
75
76     !update k
77     k = k + 2.0*deltak
78
79     !second level
80     do while (phi_x >= deltak) !update k until phi(1) >= deltak
81
82         !do the first iteration
83         phi_xminus1 = phi_0
84         phi_x = phi_deltax
85         phi_xplus1 = 2.0*phi_x - phi_xminus1 - (k**2.0)*(deltax
**2.0)*phi_x
86         x = deltax
87
88         !update phi
89         call update_phi(phi_xminus1, phi_x, phi_xplus1, k, deltax,
number_of_iterations, x)
90
91         !update k
92         k = k + deltak
93
94     end do
95
96     write(*,*) "Second energy level: ", k
97
98     !update k
99     k = k + 2.0*deltak
100
101     !third level

```

```

102     do while (phi_x >= deltak) !update k until phi(1) >= deltak
103
104         !do the first iteration
105         phi_xminus1 = phi_0
106         phi_x = phi_deltax
107         phi_xplus1 = 2.0*phi_x - phi_xminus1 - (k**2.0)*(deltax
**2.0)*phi_x
108         x = deltax
109
110         !update phi
111         call update_phi(phi_xminus1, phi_x, phi_xplus1, k, deltax,
number_of_iterations, x)
112
113         !update k
114         k = k + deltak
115
116     end do
117
118     write(*,*) "Third energy level: ", k
119
120     write(*,*) deltax*number_of_iterations
121
122 contains
123
124     subroutine update_phi(phi_xminus1, phi_x, phi_xplus1, k, deltax
, number_of_iterations, x)
125
126         !deactivate implicit typing
127         implicit none
128
129         !define variables
130         real, intent(inout) :: phi_xminus1, phi_x, phi_xplus1, k,
deltax, x
131         integer, intent(in) :: number_of_iterations
132
133         do i = 2, number_of_iterations
134             phi_xminus1 = phi_x
135             phi_x = phi_xplus1
136             phi_xplus1 = 2.0*phi_x - phi_xminus1 - (k**2.0)*(deltax
**2.0)*phi_x
137             x = x + deltax
138             write(*,*) "x: ", x, "i", i
139         end do
140
141     end subroutine update_phi
142
143 end program shooting_method

```

O código foi compilado com o comando:

```
gfortran L2-5255417-ex-4.f90 -o L2-5255417-ex-4.exe
```

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 ! -----
```

O código foi compilado com o comando:

```
gfortran L2-5255417-ex-5.f90 -o L2-5255417-ex-5.exe
```

Resultados: