

Universidade de São Paulo
Instituto de Física de São Carlos

Relatório 3 - IntroFisComp

Alexandre de Taunay Voloch

1 Tarefa 1

Aqui o programa é bem simples, sendo apenas muito extenso por causa das diversas funções diferentes que precisamos programar. O programa imprime os valores corretos (analíticos) das derivadas e também os valores a serem colocados em cada tabela respectiva. Segue o programa:

```
1      implicit real*8 (a-h,o-z)
2
3      real*8 valores_h(14)
4      real*8 valores_tabela(7,14)
5      data valores_h / 5d-1, 1d-1, 5d-2, 1d-2, 5.d-3, 1.d-3, 5.d-4,
6      &1.d-4, 5.d-5, 1.d-5, 5.d-6, 1.d-6, 1.d-7, 1.d-8/
7
8      x = 0.5d0
9
10     do i=1,7
11         do j=1,14
12             valores_tabela(i,j) = 0.d0
13         end do
14     end do
15
16     f1_correto = anal_f1(x)
17     f2_correto = anal_f2(x)
18     f3_correto = anal_f3(x)
19
20     !write(*,*)f1_correto, f2_correto, f3_correto
21
22     do i=1,14
23         h = valores_h(i)
24         valores_tabela(1,i) = f1_2f(x,h) - f1_correto
25         valores_tabela(2,i) = f1_2t(x,h) - f1_correto
26         valores_tabela(3,i) = f1_3s(x,h) - f1_correto
27         valores_tabela(4,i) = f1_5s(x,h) - f1_correto
28         valores_tabela(5,i) = f2_3s(x,h) - f2_correto
29         valores_tabela(6,i) = f2_5s(x,h) - f2_correto
30         valores_tabela(7,i) = f3_5a(x,h) - f3_correto
31     end do
32
33     ! fazer valores da tabela serem absolutos
34     do j=1,7
35         do i=1,14
36             valores_tabela(j,i) = abs(valores_tabela(j,i))
37         end do
```

```

38     end do
39
40     write(*,*) "Valores da tabela da primeira derivada:"
41     ! printar valores da primeira tabela, de f'
42     do i=1,14
43         write(*,*) (valores_tabela(j,i), j=1,4)
44     end do
45     write(*,*) "Valor exato:", f1_correto
46
47     !printar valores da segunda tabela, de f''
48     write(*,*) "Valores da tabela da segunda derivada:"
49     do i=1,14
50         write(*,*) (valores_tabela(j,i), j=5,6)
51     end do
52     write(*,*) "Valor exato:", f2_correto
53
54     write(*,*) "Valores da tabela da terceira derivada:"
55     ! printar valores da terceira tabela, de f'''
56     do i=1,14
57         write(*,*) valores_tabela(7,i)
58     end do
59     write(*,*) "Valor exato:", f3_correto
60     end
61
62     function cf(x)
63         real*8 x
64         real*8 cf
65
66         cf = dexp(x/2.d0)*dtan(2.d0*x)
67     return
68 end function cf
69
70     function dsec2(x)
71         ! sec2x
72         real*8 x
73         dsec2 = (1.d0/dcos(x))*2.d0
74         return
75     end function
76
77     function anal_f1(x)
78         ! f' analitico
79         real*8 x
80         real*8 anal_f1
81

```

```

82         ds = dsec2(2.d0*x)
83         dt = dtan(2.d0*x)
84
85         anal_f1 = dexp(x/2.d0)*(2.d0*ds + 0.5d0*dt)
86     return
87 end function
88
89 function anal_f2(x)
90     ! f'' analitico
91     real*8 x
92     real*8 anal_f2
93
94     ds = dsec2(2.d0*x)
95     dt = dtan(2.d0*x)
96
97     anal_f2 = dexp(x/2.d0)*(2.d0*ds + 0.25d0*dt + 8.d0*ds*dt)
98 end function
99
100 function anal_f3(x)
101     ! f'' analitico
102     real*8 x
103     real*8 anal_f3
104
105     ds = dsec2(2.d0*x)
106     dt = dtan(2.d0*x)
107
108     anal_f3 = dexp(x/2.d0)*(1.5d0*ds + (1.d0/8.d0)*dt +
109     &      ↪ 12.d0*ds*dt
110     + 16.d0*(ds**2.d0) + 32.d0*ds*(dt**2.d0))
111 end function
112
113 function f1_2f(x,h)
114     real*8 cf
115     external cf
116     ! f'_{2f}
117     real*8 x,h,f1_2f
118
119     f1_2f = (cf(x+h)-cf(x))/h
120
121 end function
122
123 function f1_2t(x,h)
124     ! f'_{2t}
125     real*8 cf

```

```

125     external cf
126     real*8 x,h,f1_2t
127     f1_2t = (cf(x)-cf(x-h))/h
128
129 end function
130
131 function f1_3s(x,h)
132     real*8 cf
133     external cf
134     real*8 x,h,f1_3s
135     f1_3s = (cf(x+h)-cf(x-h))/(2.d0*h)
136
137 end function
138
139 function f1_5s(x,h)
140     real*8 cf
141     external cf
142     real*8 x,h, f1_5s
143     f1_5s = (cf(x-2.d0*h) - 8.d0*cf(x-h)
144 & + 8.d0*cf(x+h) - cf(x+2.d0*h))/(12.d0 * h)
145 end function
146
147 function f2_3s(x,h)
148     real*8 cf
149     external cf
150     real*8 x,h, f2_3s
151     f2_3s = (cf(x+h) -2.d0*cf(x) + cf(x-h))/(h**2.d0)
152 end function
153
154 function f2_5s(x,h)
155     real*8 cf
156     external cf
157     real*8 x,h, f2_5s
158     f2_5s = (-1.d0*cf(x-2.d0*h) + 16.d0*cf(x-h) -30.d0*cf(x)
159 & +16.d0*cf(x+h) - cf(x+2.d0*h)) / (12.d0*(h**2.d0))
160 end function
161
162 function f3_5a(x,h)
163     real*8 cf
164     external cf
165     real*8 x,h, f3_5a
166     f3_5a = (-1.d0*cf(x- 2.d0*h) + 2.d0*cf(x-h) - 2.d0*cf(x+h)
167 & + cf(x+2.d0*h))/(2.d0*(h**3.d0))
168 end function

```

Executando-o e tabelando os resultados, obtemos as seguintes tabelas:

h	f'_{2f}	f'_{2t}	f'_{3s}	f'_{5s}
5×10^{-1}	$2.10013274476838 \times 10^1$	5.79727963749901	$1.33993035425914 \times 10^1$	$1.47520006627981 \times 10^1$
1×10^{-1}	4.92612234674952	2.37530455157099	1.27540889758926	1.22780673941021
5×10^{-2}	1.94152325544609	1.36425502733287	$2.88634114056608 \times 10^{-1}$	$4.02908137876121 \times 10^{-2}$
1×10^{-2}	$3.32088970649423 \times 10^{-1}$	$3.09677366242413 \times 10^{-1}$	$1.12058022035040 \times 10^{-2}$	$5.49830810037122 \times 10^{-5}$
5×10^{-3}	$1.63093714957750 \times 10^{-1}$	$1.57495704052170 \times 10^{-1}$	$2.79900545278977 \times 10^{-3}$	$3.26013078932874 \times 10^{-6}$
1×10^{-3}	$3.21616449508326 \times 10^{-2}$	$3.19374617383090 \times 10^{-2}$	$1.12091606261799 \times 10^{-4}$	$1.65651719896687 \times 10^{-7}$
5×10^{-4}	$1.60527810142845 \times 10^{-2}$	$1.59964790257945 \times 10^{-2}$	$2.81509942450242 \times 10^{-5}$	$1.70790166009738 \times 10^{-7}$
1×10^{-4}	$3.20620694131790 \times 10^{-3}$	$3.20362629620163 \times 10^{-3}$	$1.29032255813399 \times 10^{-6}$	$1.71131150139558 \times 10^{-7}$
5×10^{-5}	$1.60290909571970 \times 10^{-3}$	$1.60200723216164 \times 10^{-3}$	$4.50931779027997 \times 10^{-7}$	$1.71134852067212 \times 10^{-7}$
1×10^{-5}	$3.20673902141522 \times 10^{-4}$	$3.20309297709542 \times 10^{-4}$	$1.82302215989694 \times 10^{-7}$	$1.71100063894869 \times 10^{-7}$
5×10^{-6}	$1.60419847670568 \times 10^{-4}$	$1.60071918788418 \times 10^{-4}$	$1.73964441074759 \times 10^{-7}$	$1.71177779506593 \times 10^{-7}$
1×10^{-6}	$3.22203966156565 \times 10^{-5}$	$3.18781076664720 \times 10^{-5}$	$1.71144474592211 \times 10^{-7}$	$1.71088963440980 \times 10^{-7}$
1×10^{-7}	$3.37147336537669 \times 10^{-6}$	$3.03451348671047 \times 10^{-6}$	$1.68479939333110 \times 10^{-7}$	$1.66814604796173 \times 10^{-7}$
1×10^{-8}	$5.64829559124291 \times 10^{-7}$	$1.90122097620815 \times 10^{-7}$	$1.87353730751738 \times 10^{-7}$	$1.91054473575036 \times 10^{-7}$
Exato	9.79678184270445			

Tabela 1: Erros absolutos para a primeira derivada

Como podemos ver, para a primeira derivada um maior valor de h equivale, em geral, a uma maior precisão. No caso das últimas duas técnicas (f'_{3s} e f'_{5s}) a precisão é maximizada com $h = 10^{-7}$ e depois torna a diminuir levemente.

h	f''_{3s}	f''_{5s}
5×10^{-1}	$9.45064193068224 \times 10^1$	$1.02804389820175 \times 10^2$
1×10^{-1}	8.91594529675227	8.60379834714472
5×10^{-2}	2.01724196912639	$2.82325806748908 \times 10^{-1}$
1×10^{-2}	$7.83100027307881 \times 10^{-2}$	$3.85608147027483 \times 10^{-4}$
5×10^{-3}	$1.95601155312346 \times 10^{-2}$	$2.31802004719839 \times 10^{-5}$
1×10^{-3}	$7.83002688862666 \times 10^{-4}$	$8.24995879611379 \times 10^{-7}$
5×10^{-4}	$1.96393705252262 \times 10^{-4}$	$8.58117530810887 \times 10^{-7}$
1×10^{-4}	$8.68874256809704 \times 10^{-6}$	$8.72772460525084 \times 10^{-7}$
5×10^{-5}	$2.87117391906122 \times 10^{-6}$	$9.91196259292337 \times 10^{-7}$
1×10^{-5}	$3.70134640093056 \times 10^{-6}$	$4.44149507927705 \times 10^{-6}$
5×10^{-6}	$2.96053443378241 \times 10^{-5}$	$4.81090614243840 \times 10^{-5}$
1×10^{-6}	$1.80595675701056 \times 10^{-4}$	$4.95158866002043 \times 10^{-4}$
1×10^{-7}	$3.84551655812402 \times 10^{-2}$	$4.40062807043802 \times 10^{-2}$
1×10^{-8}	$1.13968419880578 \times 10^1$	$1.54676597450168 \times 10^1$
Exato	64.0983236864528	

Tabela 2: Erros absolutos para a segunda derivada

Para a segunda derivada, podemos ver que no caso de f''_{3s} , a maior precisão é atingida em $h = 5 \cdot 10^{-5}$, enquanto que para f''_{5s} , a maior precisão é atingida muito mais rapidamente com $h = 1 \cdot 10^{-3}$.

h	f'''_{5a}
5×10^{-1}	$6.39049869974093 \times 10^2$
1×10^{-1}	$8.30414781340628 \times 10^2$
5×10^{-2}	$1.17905225967068 \times 10^2$
1×10^{-2}	4.13251621152244
5×10^{-3}	1.02913919815319
1×10^{-3}	$4.11268346746283 \times 10^{-2}$
5×10^{-4}	$1.02952751469729 \times 10^{-2}$
1×10^{-4}	$6.86961002543285 \times 10^{-4}$
5×10^{-5}	$1.97757425655709 \times 10^{-3}$
1×10^{-5}	$7.25440551477732 \times 10^{-1}$
5×10^{-6}	4.49260766426050
1×10^{-6}	5.38078608396029
1×10^{-7}	$5.54439997711719 \times 10^5$
1×10^{-8}	$4.44089881364663 \times 10^8$
Exato	671.514600859054

Tabela 3: Erros absolutos para a terceira derivada

Aqui na terceira derivada, vemos que a precisão é menor, e que atinge seu máximo em $h = 10^{-4}$, depois passando a diminuir muito e praticamente divergir para h muito pequeno.

2 Tarefa 2

Aqui o programa imprime primeiro o valor analítico da integral, que vale $\frac{e-1}{4e\pi^2}$ (fonte: Wolfram), e depois imprime as diferenças observadas utilizando cada método de aproximação da integral. O programa é o seguinte:

```

1      implicit real*8  (a-h,o-z)
2
3      pi = 4.d0*datan(1.d0)
4      e = dexp(1.d0)
5
6      valor_analitico = (e - 1.d0)/(e + (4.d0*e*(pi**2.d0)))
7
8      write(*,*)valor_analitico
9
10     do i=2,12

```



```

11      h = 1.d0/(3.d0*(2.d0**i))
12      xt = trapezio(h)
13      xs = simpson(h)
14      xb = boole(h)
15
16      et = abs(valor_analitico-xt)
17      es = abs(valor_analitico-xs)
18      eb = abs(valor_analitico-xb)
19
20      write(*,*)et, es, eb
21  end do
22
23  end
24
25  function cf(x)
26      real*8 x
27      real*8 cf
28      pi = 4.d0*datan(1.d0)
29      cf = dexp(-x)*dcos(2.d0*pi*x)
30  end function
31
32  function trapezio(h)
33      real*8 h, cf, trapezio, a, b, soma
34      integer*16 nparticoes
35      external cf
36
37      a = 0.d0
38      b = 1.d0
39      nparticoes = dint((b-a)/h)
40
41      soma = 0.5d0 * (cf(a) + cf(b))
42      do i=1,(nparticoes-1)
43          soma = soma + cf(a + dble(i)*h)
44      end do
45
46      trapezio = h * soma
47  end function
48
49  function simpson(h)
50      real*8 h, cf, simpson, a, b, soma
51      integer*16 i, nparticoes
52      external cf
53
54      a = 0.d0

```

```

55      b = 1.d0
56      nparticoes = dint((b-a)/h)
57
58      soma = cf(a) + cf(b)
59
60      do i=1,nparticoes-1,2
61          soma = soma + 4.d0*cf(a + dble(i)*h)
62      end do
63
64      do i=2,nparticoes-1,2
65          soma = soma + 2.d0*cf(a + dble(i)*h)
66      end do
67
68      simpson = (h/3.d0) * soma
69  end function
70
71  function boole(h)
72      real*8 h, cf, boole, a, b, soma
73      integer*16 nparticoes, i
74      external cf
75
76      a = 0.d0
77      b = 1.d0
78      nparticoes = dint((b-a)/h)
79
80      soma = 0.d0
81      do i=0,nparticoes-1,4
82          soma = soma + (2.d0*h/45.d0) * (7.d0*cf(a + dble(i)*h) +
83      &          32.d0*cf(a + dble(i+1)*h) + 12.d0*cf(a + dble(i+2)*h)
84      &          ↪ +
85      &          32.d0*cf(a + dble(i+3)*h) + 7.d0*cf(a + dble(i+4)*h))
86      end do
87
88      boole = soma
89  end function

```

Tabelando esses resultados, temos

h^{-1}	Regra do Trapézio	Regra de Simpson	Regra de Boole
3×2^2	3.7084×10^{-4}	2.0955×10^{-5}	4.0327×10^{-6}
3×2^3	9.1773×10^{-5}	1.2502×10^{-6}	6.3421×10^{-8}
3×2^4	2.2892×10^{-5}	6.8806×10^{-8}	9.9554×10^{-9}
3×2^5	5.7261×10^{-6}	4.2771×10^{-9}	9.1493×10^{-9}
3×2^6	1.4382×10^{-6}	8.8331×10^{-9}	9.1368×10^{-9}
3×2^7	3.6638×10^{-7}	9.1177×10^{-9}	9.1366×10^{-9}
3×2^8	9.8446×10^{-8}	9.1354×10^{-9}	9.1366×10^{-9}
3×2^9	3.1464×10^{-8}	9.1365×10^{-9}	9.1366×10^{-9}
3×2^{10}	1.4718×10^{-8}	9.1366×10^{-9}	9.1366×10^{-9}
3×2^{11}	1.0532×10^{-8}	9.1366×10^{-9}	9.1366×10^{-9}
3×2^{12}	9.4855×10^{-9}	9.1366×10^{-9}	9.1366×10^{-9}
Exato	$1.5616236904490828 \times 10^{-2}$		

Tabela 4: Diferenças entre os métodos e o valor analítico

Aqui podemos ver que a regra do trapézio vai aumentando de precisão conforme h diminui, e não chegamos num limite de precisão. Portanto, para este método o valor de h ótimo depende da precisão desejada. Já a regra de Simpson chega rapidamente numa precisão boa, com $h^{-1} = 3 \cdot 2^6$ e praticamente não aumenta a partir dali. Já para a regra de Boole, chegamos numa precisão boa muito rapidamente, com $h^{-1} = 3 \cdot 2^4$ e também não aumenta quase nada a partir dali. Portanto, estas últimas duas técnicas permitem utilizar um valor maior de h , que significa menos cálculos realizados no total e, portanto, maior rapidez na execução do programa (lembrando que h representa as subdivisões da "malha" do eixo x no qual estamos integrando).

3 Tarefa 3

Aqui utilizamos os métodos de acordo com a forma que são explicados no projeto. Primeiro o código procura os pontos onde há mudança de sinal (o início do método de bisseção) e depois utiliza estes como ponto de partida para cada um dos métodos de busca de raízes. Os métodos são executados até que atinjam a tolerância desejada, que neste caso é de $\epsilon = 10^{-6}$. O programa é o seguinte:

```

1  implicit real*8 (a-h,o-z)
2
3  real*8 espac,busca_direta, xm, tolerancia, dif

```

```

4      real*8 a(3), b(3)
5      real*8 xnewt(3), asec(3), bsec(3), difs(3)
6      real*8 resultados(3,3,100)  ! tabela de resultados
7      logical terminamos
8
9      !espac = 1d-1*dsqrt(5.d0)
10     espac = 0.1d0
11     data a / -10d0, -10d0, -10d0 /
12     data b / -10d0, -10d0, -10d0 /
13
14     tolerancia = 1d-6
15
16     do i=1,3
17         dif = 1.d0
18         do while(dif.gt.0d0)
19             a(i) = a(i) + espac
20             b(i) = a(i) + espac
21             dif = cf(b(i))*cf(a(i))
22             !      write(1,*)a(i),b(i),dif
23         end do
24         write(*,*)"encontramos ponto de bisseção:",a(i),b(i)
25         if(i.lt.3) then
26             a(i+1)=b(i)
27             b(i+1)=b(i)
28         end if
29     end do
30
31     write(*,*)"Terminamos de pesquisar"
32     xnewt = a
33     asec = a
34     bsec = b
35
36     ! calcular usando o método da bisseção
37
38     write(*,*)"Método da bisseção:"
39
40     do i=1,3
41     write(*,*)"tentando i=", i
42         niter = 0
43         dif = 1.d0
44         do while(dif.ge.tolerancia)
45             xm = (b(i) + a(i))/2d0
46             if( (cf(b(i))*cf(xm)).gt.0d0) then
47                 b(i) = xm

```

```

48         else
49             a(i) = xm
50         end if
51
52         dif = abs(b(i)-a(i))
53         if (dif.lt.tolerancia) then
54             !calcular xm novamente para imprimir o valor correto
55             xm = (b(i) + a(i))/2d0
56         end if
57         write(*,*)xm
58     end do
59 end do
60
61 write(*,*) "Método de Newton:"
62 do i=1,3
63     write(*,*) "tentando i=", i
64     niter = 0
65     dif = 1.d0
66     do while(dif.ge.tolerancia)
67         x_newt_antigo = xnewt(i) ! variável temporária p/
68         ↪ calcular precisão
69         xnewt(i) = xnewt(i) - cf(xnewt(i))/df(xnewt(i))
70
71         dif = abs(xnewt(i) - x_newt_antigo)
72         write(*,*)xnewt(i)
73     end do
74 end do
75
76 write(*,*) "Método da Secante:"
77 do i=1,3
78     write(*,*) "tentando i=", i
79     niter = 0
80     dif = 1.d0
81     do while(dif.ge.tolerancia)
82         xtemp = bsec(i) ! variável temporária pra guardar o valor
83         ↪ antigo de x_n
84
85         bsec(i) = bsec(i) - (cf(bsec(i)) * (bsec(i)-asec(i))
86         & / (cf(bsec(i))-cf(asec(i))))
87
88         asec(i) = xtemp
89
90         dif = abs(bsec(i)-asec(i))
91         write(*,*)bsec(i)

```

```

90         end do
91     end do
92
93     end
94
95     function cf(x)
96         real*8 cf,x
97         cf = x**3d0 - 4d0*(x**2d0) - 59d0*x + 126d0
98     end function
99
100    function df(x)
101        real*8 df,x
102        df = 3d0*(x**2d0) - 8d0*x - 59d0
103    end function
104

```

Executando-o, para o intervalo de espaçamento requisitado de 0.1, obtemos os seguintes resultados:

iteração	r_1	r_2	r_3
0	entre -7.01 e -6.9	entre 1.99999 e 2.1	entre 8.999 e 9.1
1	-6.9500000000000108	2.049999999999821	9.049999999999652
2	-6.9750000000000103	2.024999999999821	9.024999999999666
3	-6.9875000000000105	2.012499999999820	9.012499999999673
4	-6.9937500000000110	2.006249999999819	9.006249999999659
5	-6.9968750000000108	2.003124999999821	9.003124999999652
6	-6.9984375000000103	2.001562499999821	9.001562499999666
7	-6.9992187500000105	2.000781249999820	9.000781249999673
8	-6.9996093750000110	2.000390624999819	9.000390624999659
9	-6.9998046875000108	2.000195312499821	9.000195312499652
10	-6.9999023437500103	2.000097656249821	9.000097656249666
11	-6.9999511718750105	2.0000488281249820	9.0000488281249673
12	-6.9999755859375110	2.0000244140624819	9.0000244140624659
13	-6.9999877929687608	2.0000122070312321	9.0000122070312152
14	-6.9999938964843853	2.0000061035156071	9.0000061035155916
15	-6.9999969482421980	2.0000030517577945	9.0000030517577798
16	-6.9999984741211048	2.0000015258788881	9.0000015258788721
17	-6.9999996185302837	2.0000003814697087	9.0000003814696932
Exato	-7.0000000000000000	2.0000000000000000	9.0000000000000000

Tabela 5: Método da Bisseção

iteração	r_1	r_2	r_3
0	entre -7.01 e -6.9	entre 1.99999 e 2.1	entre 8.999 e 9.1
1	-7.0000000000000000	2.0000000000000000	9.0000000000000000
Exato	-7.0000000000000000	2.0000000000000000	9.0000000000000000

Tabela 6: Método de Newton

iteração	r_1	r_2	r_3
0	entre -7.01 e -6.9	entre 1.99999 e 2.1	entre 8.999 e 9.1
1	-7.0000000000000000	2.0000000000000000	8.999999999999982
2	-7.0000000000000000	2.0000000000000000	9.0000000000000000
Exato	-7.0000000000000000	2.0000000000000000	9.0000000000000000

Tabela 7: Método da Secante

Podemos variar um pouco o valor do espaçamento para dar um pouco mais de trabalho para os métodos de Newton e Secante. No caso, mudando o intervalo para $\frac{1}{10}\sqrt{5} \approx 0.223$, temos

iteração	r_1	r_2	r_3
0	entre -7.09 e 6.86	entre 1.85 e 2.07	entre 8.78 e 9.007
1	-6.9813082303752889	1.9629636796238739	8.8947744098732251
2	-7.0372099298127839	2.0188653790613689	8.9506761093107201
3	-7.0092590800940364	1.9909145293426214	8.9786269590294658
4	-6.9952836552346627	2.0048899542019951	8.9926023838888405
5	-7.0022713676643491	1.9979022417723082	8.9995900963185278
6	-6.9987775114495054	2.0013960979871515	9.0030839525333697
7	-7.0005244395569273	1.9996491698797298	9.0013370244259487
8	-6.9996509755032168	2.0005226339334405	9.0004635603722392
9	-7.0000877075300725	2.0000859019065853	9.0000268283453835
10	-6.9998693415166446	1.9998675358931575	8.9998084623319556
11	-6.9999785245233586	1.9999767188998714	8.9999176453386696
12	-7.0000331160267155	2.0000313104032283	8.9999722368420265
13	-7.0000058202750370	2.0000040146515499	8.9999995325937050
14	-6.9999921723991978	1.9999903667757106	9.0000131804695442
15	-6.9999989963371174	1.9999971907136302	9.0000063565316246
16	-7.0000024083060772	2.0000006026825901	9.0000029445626648
17	-7.0000007023215973	1.9999988966981102	9.0000012385781858
18	-7.0000002758254780	2.0000001761864699	8.9999999590898252
Exato	-7.0000000000000000	2.0000000000000000	9.0000000000000000

Tabela 8: Método da Bisseção com intervalo modificado

iteração	r_1	r_2	r_3
0	entre -7.09 e 6.86	entre 1.85 e 2.07	entre 8.78 e 9.007
1	-7.0014686344168542	1.9994063811274549	9.0104043919142427
2	-7.0000003743126573	1.999999888202884	9.0000221555644782
3	-7.00000000000000240	2.0000000000000000	9.0000000001008029
4	-	-	9.0000000000000000
Exato	-7.0000000000000000	2.0000000000000000	9.0000000000000000

Tabela 9: Método de Newton com intervalo modificado

iteração	r_1	r_2	r_3
0	entre -7.09 e 6.86	entre 1.85 e 2.07	entre 8.78 e 9.007
1	-6.9978801131115222	2.0003394867587136	8.9996965282300696
2	-7.0000488919289863	1.9999991618752768	8.9999995904863042
3	-6.9999999820010332	2.0000000000090346	9.0000000000255227
4	-6.9999999999998472	-	9.0000000000000000
Exato	-7.0000000000000000	2.0000000000000000	9.0000000000000000

Tabela 10: Método da Secante com intervalo modificado