Instituto de Física de São Carlos
Universidade de São Paulo

[7600017] - Introdução à Física Computacional

Projeto 3

Docente:
Francisco Castilho Alcaraz

Aluno:
Breno Henrique Pelegrin da Silva (13687303)

26 de outubro de 2023

Sumário

Lista de Figuras

1 Contexto

Nesse projeto, foram explorados diversos métodos numéricos de derivação e integração, que podem auxiliar em diversos processos relacionados à tópicos de Física Computacional. Além disso, também foram explorados vários métodos para encontrar raízes de polinômios, que também podem ser úteis na resolução de problemas de física. Nos códigos, o formato CSV (valores separados por vírgula) foi usado para expressar os resultados, devido à sua grande facilidade de manipulação em outros softwares de processamento de dados. Por fim, as tabelas de resultados expostas neste relatório tiveram seu tamanho reduzido, para que fosse possível inseri-las no espaço disponível das páginas.

2 Tarefa A

Nessa tarefa, foi criado um programa para calcular numericamente as derivadas de 1ª ordem, 2ª ordem e 3ª ordem da função $f(x)$ expressa na equação 1, utilizando métodos numéricos diferentes. Os métodos utilizados para calcular as derivadas foram: derivada simétrica de 3 pontos (d1s3p), derivada para frente de 2 pontos (d1f2p), derivada para trás de 2 pontos (d1t2p), derivada simétrica de 5 pontos (d1s5p), derivada segunda simétrica de 5 pontos (d2s5p), derivada terceira anti-simétrica de 5 pontos (d3s5p).

As relações para calcular cada uma das derivadas citadas estão expostas nas equações 3, 4, 5, 6, 7 e 8, respectivamente. Nas relações, utiliza-se uma notação f_n, cujo significado é dado pela equação 2, onde h é um passo arbitrário no eixo x e x_0 é um ponto arbitrário do eixo x onde deseja-se calcular o valor da derivada. Para essa tarefa, escolheu-se $x_0 = \frac{1}{2}$. Além disso, o símbolo $O(h^n)$ significa que é esperado um erro da ordem de h^n.

$$f(x) = e^{\frac{x}{2}} tan(2x) \tag{1}$$

$$f_n = f(x_0 + nh), \text{ com } n \in \mathbb{Z} \tag{2}$$

$$f' = \frac{f_1 - f_{-1}}{2h} + O(h^2) \tag{3}$$

$$f' = \frac{f_1 - f_0}{h} + O(h) \tag{4}$$

$$f' = \frac{f_0 - f_{-1}}{h} + O(h) \tag{5}$$

$$f' = \frac{f_{-2} - 8f_{-1} + 8f_1 - f_2}{12h} + O(h^4) \tag{6}$$

$$f'' = \frac{-f_{-2} + 16f_{-1} - 30f_0 + 16f_1 - f_2}{12h^2} + O(h^4) \tag{7}$$

$$f'' = \frac{-f_{-2} + 2f_{-1} - 2f_1 + f_2}{2h^3} + O(h^2) \tag{8}$$

As expressões analíticas para as derivadas de 1ª, 2ª e 3ª ordem da função $f(x)$ estão expostas nas equações 9, 10 e 11.

$$f' = \left(2\tan^2(2x) + 2\right) e^{\frac{x}{2}} + \frac{e^{\frac{x}{2}} \tan(2x)}{2} \tag{9}$$

$$f'' = \left(2\tan^2(2x) + 2\right) e^{\frac{x}{2}} + 2 \cdot \left(4\tan^2(2x) + 4\right) e^{\frac{x}{2}} \tan(2x) + \frac{e^{\frac{x}{2}} \tan(2x)}{4} \tag{10}$$

$$\begin{aligned} f''' = {} & \left(2\tan^2(2x) + 2\right)\left(8\tan^2(2x) + 8\right) e^{\frac{x}{2}} + \frac{3 \cdot \left(2\tan^2(2x) + 2\right) e^{\frac{x}{2}}}{4} \\ & + 8 \cdot \left(4\tan^2(2x) + 4\right) e^{\frac{x}{2}} \tan^2(2x) + 2 \cdot \left(4\tan^2(2x) + 4\right) e^{\frac{x}{2}} \tan(2x) \\ & + \frac{\left(8\tan^2(2x) + 8\right) e^{\frac{x}{2}} \tan(2x)}{2} + \frac{e^{\frac{x}{2}} \tan(2x)}{8} \end{aligned} \tag{11}$$

$$f'(x_0) = 9.79678201384$$

$$f''(x_0) = 64.09832454947$$

$$f'''(x_0) = 671.51461345787$$

Utilizando as relações numéricas apresentadas, calculou-se as derivadas citadas para diferentes passos h, utilizando $x_0 = \frac{1}{2}$, estando os valores de h e os resultados obtidos para cada método expostos na tabela 1, bem como os valores exatos com 11 casas decimais de precisão para cada derivada, para fins de comparação.

lém disso, na tabela 2, estão expostas as diferenças absolutas ($|calculado - analitico|$) entre o cálculo numérico e o resultado analítico, a fim de observar o comportamento do erro associado ao cálculo. O código elaborado para essa tarefa está exposto na figura 1, e sua explicação é dada logo em seguida no verbete 1.

Figura 1: Código-fonte da tarefa A

```fortran
      program Derivadas
         implicit real*8 (a-h,o-z)
         dimension h(14)
         parameter(d1=9.79678201384)
         parameter(d2=64.09832454947)
         parameter(d3=671.51461345787)
         h = [5.0E-1, 2.0E-1, 1.0E-1, 5.0E-2, 1.0E-2, 5.0E-3, 1.0E-3,
     &5.0E-4, 1.0E-4, 5.0E-5, 1.0E-5, 1.0E-6, 1.0E-7, 1.0E-8]
         x0 = 0.5d0
         open(unit=50, file='saida-1-diff-13687303.csv')
         open(unit=51, file='saida-2-comp-13687303.csv')
         write(50,100) "h", "d1s3p", "d1f2p", "d1t2p", "d1s5p", "d2s5p"
     &,"d3a5p"
         write(51,100) "h", "d1s3p", "d1f2p", "d1t2p", "d1s5p", "d2s5p"
     &,"d3a5p"
 100     format(a, ",", a, ",", a, ",", a, ",", a, ",", a, ",", a)
         do i=1, 14, 1
            d1s3p = (fn(x0, 1, h(i)) - fn(x0, -1, h(i)))/(2*h(i))
            d1f2p = (fn(x0, 1, h(i)) - fn(x0, 0, h(i)))/(h(i))
            d1t2p = (fn(x0, 0, h(i)) - fn(x0, -1, h(i)))/(h(i))
            d1s5p = (fn(x0, -2, h(i)) - 8*fn(x0, -1, h(i))
     & + 8*fn(x0, 1, h(i)) - fn(x0, 2, h(i))) / (12*h(i))
            d2s5p = (-fn(x0, -2, h(i)) + 16*fn(x0, -1, h(i))
     & - 30*fn(x0, 0, h(i)) + 16*fn(x0, 1, h(i)) - fn(x0, 2, h(i)))
     & / (12*(h(i)**2))
            d3a5p = (-fn(x0, -2, h(i)) + 2*fn(x0, -1, h(i))
     & - 2*fn(x0, 1, h(i)) + fn(x0, 2, h(i)))/(2*(h(i)**3))
         write(50,101) h(i), d1s3p, d1f2p, d1t2p, d1s5p, d2s5p, d3a5p
         write(51,101) h(i), abs(d1s3p-d1), abs(d1f2p-d1), abs(d1t2p-d1
     &), abs(d1s5p-d1), abs(d2s5p-d2), abs(d3a5p-d3)
 101     format(E16.2, ",", F24.11, ",", F24.11, ",", F24.11, ",",
     & F24.11, ",", F24.11, ",", F24.11)
         end do
      write(50,102) d1, d1, d1, d1, d2, d3
 102  format("exatos", ",", F24.11, ",", F24.11, ",", F24.11, ",",
     & F24.11, ",", F24.11, ",", F24.11)
      close(50)
      close(51)
      end program

      function func(x)
         implicit real*8 (a-h,o-z)
         func = exp(x/2.0d0) * tan(2.0d0*x)
         return
      end function func

      function fn(x0, n, h)
         implicit real*8 (a-h,o-z)
         fn = func(x0 + n*h)
         return
      end function fn
```

Verbete 1: Explicação do código da tarefa A

Inicialmente, inicializa-se o vetor de passos, onde serão armazenados os passos a serem utilizados nos cálculos e define-se como parâmetros os valores exatos das derivadas de 1ª, 2ª e 3ª ordem, a fim de calcular a diferença entre os valores calculados e os valores exatos. Depois, calcula-se as derivadas utilizando as expressões analíticas, com o apoio de duas funções que foram definidas no código, uma para a expressão f_n e outra para a função $f(x)$.
Os resultados das derivadas são escritos em formato CSV (valores serparados por vírgula) em um arquivo *saida-1-diff-13687303.csv* e os resultados da diferença absoluta entre os valores calculados e os valores exatos são escritos em outro arquivo CSV, chamado *saida-2-comp-13687303.csv*.

Tabela 1: Valores calculados e exatos para cada método de derivação

h	**d1s3p**	**d1f2p**	**d1t2p**	**d1s5p**	**d2s5p**	**d3a5p**
0.50E+00	-3.60252169989	-11.20454560498	3.99950220521	-4.95521882009	-38.70606629556	32.46473088496
0.20E+00	18.58183816540	31.13918870401	6.02448762678	27.66549220246	189.32424380495	-1362.54806495201
0.10E+00	11.07219078361	14.72290430232	7.42147726491	8.56897498968	55.49452454316	1501.92943159643
0.50E-01	10.08541596563	11.73830513320	8.43252679806	9.75649102631	63.81599786143	789.41983085453
0.10E-01	9.80798764441	10.12887080566	9.48710448315	9.79672685963	64.09793807834	675.64711688441
0.50E-02	9.79958084803	9.95987555395	9.63928614211	9.79677858257	64.09830050628	672.54374001045
0.10E-02	9.79689393432	9.82894348919	9.76484437945	9.79678200836	64.09832451234	671.55572725930
0.50E-03	9.79680999370	9.81283462448	9.78078536292	9.79678201350	64.09832454831	671.52489281759
0.10E-03	9.79678313303	9.79998804956	9.79357821649	9.79678201384	64.09832449668	671.51533871177
0.50E-04	9.79678229364	9.79838475176	9.79517983551	9.79678201384	64.09832468912	671.51445053328
0.10E-04	9.79678202504	9.79710251663	9.79646153344	9.79678201385	64.09832100318	671.46293618103
0.10E-05	9.79678201405	9.79681406319	9.79674996491	9.79678201390	64.09846759836	555.11151651714
0.10E-06	9.79678201216	9.79678521738	9.79677880695	9.79678201216	64.13018114690	222044.59714052684
0.10E-07	9.79678200078	9.79678233385	9.79678166771	9.79678199893	68.64879119041	222044608.97344028950
EXATOS	9.79678153992	9.79678153992	9.79678153992	9.79678153992	64.09832763672	671.51458740234

Fonte: Autoria própria

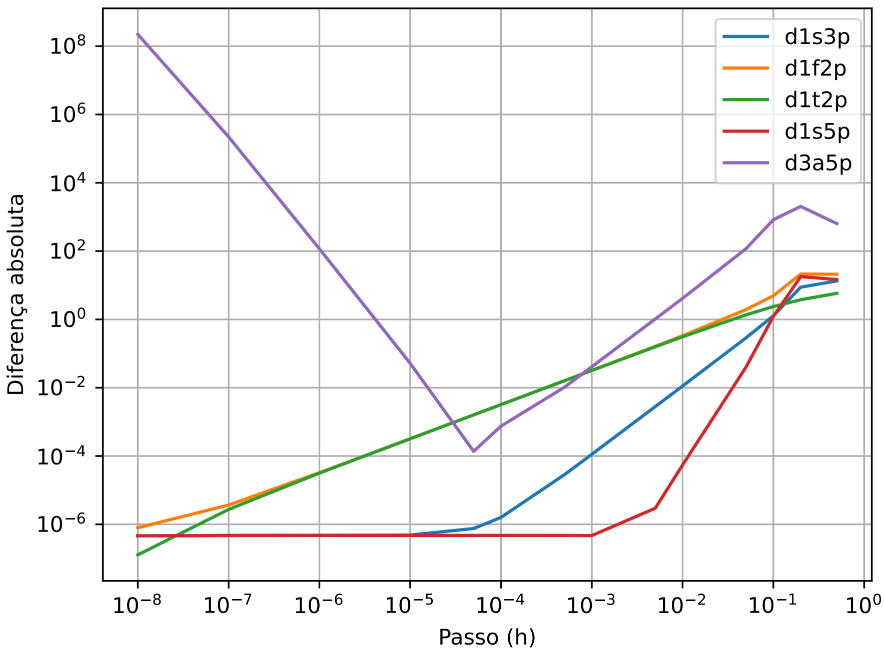
Tabela 2: Diferenças absolutas para cada método de derivação

h	**d1s3p**	**d1f2p**	**d1t2p**	**d1s5p**	**d2s5p**	**d3a5p**
0.50E+00	13.39930323980	21.00132714490	5.79727933471	14.75200036001	102.80439393228	639.04985651738
0.20E+00	8.78505662548	21.34240716409	3.77229391313	17.86871066254	125.22591616823	2034.06265235436
0.10E+00	1.27540924369	4.92612276240	2.37530427501	1.22780655023	8.60380309355	830.41484419408
0.50E-01	0.28863442572	1.94152359329	1.36425474185	0.04029051361	0.28232977529	117.90524345218
0.10E-01	0.01120610449	0.33208926575	0.30967705677	0.00005468029	0.00038955837	4.13252948207
0.50E-02	0.00279930812	0.16309401404	0.15749539780	0.00000295734	0.00002713044	1.02915260810
0.10E-02	0.00011239440	0.03216194927	0.03193716046	0.00000046844	0.00000312438	0.04113985696
0.50E-03	0.00002845378	0.01605308457	0.01599617700	0.00000047358	0.00000308841	0.01030541525
0.10E-03	0.00000159311	0.00320650965	0.00320332342	0.00000047392	0.00000314004	0.00075130942
0.50E-04	0.00000075372	0.00160321184	0.00160170440	0.00000047392	0.00000294760	0.00013686906
0.10E-04	0.00000048512	0.00032097672	0.00032000648	0.00000047393	0.00000663354	0.05165122131
0.10E-05	0.00000047413	0.00003252327	0.00003157501	0.00000047398	0.00013996164	116.40307088520
0.10E-06	0.00000047224	0.00000367746	0.00000273297	0.00000047224	0.03185351018	221373.08255312449
0.10E-07	0.00000046086	0.00000079393	0.00000012780	0.00000045901	4.55046355369	222043937.45885288715

Fonte: Autoria própria

A partir das diferenças absolutas entre os cálculos numéricos e os resultados analíticos, expostas na tabela 2, foi criado um gráfico das das diferenças em função do passo h para cada método de derivada, exposto na figura 2, na qual ambos os eixos cartesianos estão em escala log-log. A partir da análise do gráfico, pode-se verificar que o passo h ótimo é $0,5 \times 10^{-4}$, já que a maioria dos métodos de derivação apresenta erro mínimo para esse passo.

Figura 2: Diferença absoluta em função do passo



Fonte: Autoria própria

3 Tarefa B

Nessa tarefa, criou-se um programa para calcular a integral da função $f(x)$ dada na equação 12, no intervalo de 0 a 1, utilizando diversos métodos numéricos diferentes, sendo eles: regra do trapézio (tr), regra de Simpson (sp) e regra de Boole (bl), para diferentes divisões N do intervalo de 0 a 1. A partir do número de divisões N do intervalo de integração, pode-se definir o tamanho de cada divisão como $h = (b-a)/N$, onde b é o limite superior e a o limite inferior.

Assim, no caso da integral citada, como $b = 1$ e $a = 0$, o tamanho h é $1/N$. A integral analítica está expressa na equação 13, e as expressões para a integração de cada método estão expostas, respectivamente, nas equações 15, 16 e 17. Nas expressões das integrais numéricas, utiliza-se a expressão f_n, cujo significado é dado pela equação 14, onde x_0 é o ponto atual onde a integral está sendo calculada. A expressão $O(h^n)$ denota que a integral numérica pode ter um erro na ordem de h^n.

$$f(x) = e^{-x} cos(2\pi x) \tag{12}$$

$$I = \int_0^1 f(x)dx = \left(\frac{2\pi \sin(2\pi x)}{e^x + 4\pi^2 e^x} - \frac{\cos(2\pi x)}{e^x + 4\pi^2 e^x} \right) \Bigg|_0^1 = -\frac{1}{e + 4e\pi^2} + \frac{1}{1 + 4\pi^2} \tag{13}$$

$$\int_0^1 f(x)dx = 0.015616236904$$

$$f_n = f(x_0 + nh), \text{ com } n \in \mathbb{Z} \tag{14}$$

$$\int_{-h}^{h} f(x)dx = \frac{h}{2}(f_{-1} + 2f_0 + f_1) + O(h^3) \tag{15}$$

$$\int_{-h}^{h} f(x)dx = \frac{h}{3}(f_1 + 4f_0 + f_{-1}) + O(h^5) \tag{16}$$

$$\int_{x_0}^{x_4} f(x)dx = \frac{2h}{45}(7f_0 + 32f_1 + 12f_2 + 32f_3 + 7f_4) + O(h^7) \tag{17}$$

Para calcular a integral no método do trapézio, itera-se para cada x_0 do eixo x, começando em $x_0 = a + h$ e indo até $x_0 = b - h$ em passos de $2h$, já que o método do trapézio calcula a área formada pelo trapézio anterior e posterior ao ponto x_0 de interesse. A mesma relação vale para o método de Simpson. Já no método de Boole, inicia-se a iteração em $x_0 = a$, indo até $x_0 = b - 4h$ em passos de $4h$, uma vez que esse método considera a área de quatro seções após o ponto x_0 de interesse.

O código-fonte do programa desenvolvido para essa tarefa está exposto em 3 partes, nas figuras 3, 4 e 5, respectivamente. A explicação para o código é dada logo em seguida, no verbete 2, e os resultados para a diferença absoluta entre os valores calculados e os valores analíticos em cada método são exibidos na tabela 3.

Figura 3: Código-fonte da tarefa B (parte 1)

```fortran
      program Integrais
         implicit real*8 (a-h, o-z)
         parameter (anlint = 0.015616236904)
         dimension n(10)

         n(1) = 12

         do i=2, 10, 1
            n(i) = n(i-1)*2
         end do
         x0 = 0.0d0

         open(unit=50, file='saida-1-int-13687303.csv')
         open(unit=51, file='saida-2-erro-13687303.csv')

         write(50,100)
         write(51,100)
 100     format("n", ",", "h", ",", "tr", ",", "sp", ",", "bl")
         tr=0
         sp=0
         bl=0
         do i=1, 10, 1
            h = 1.0d0/n(i)
            tr = trint(0.0d0, 1.0d0, n(i))
            sp = spint(0.0d0, 1.0d0, n(i))
            bl = blint(0.0d0, 1.0d0, n(i))
            write(50,102) n(i), h, tr, sp, bl
            write(51,102) n(i), h, abs(tr-anlint), abs(sp-anlint), abs(
     &bl-anlint)
 102        format(I0, ",", F24.11, ",", F24.11, ",", F24.11, "," F24.1
     &1)
         end do
         write(50,*) "exatos", ",", "exatos", ",", anlint, ",", anlint,
     &",", anlint
         close(50)
         close(51)
      end program
```

Figura 4: Código-fonte da tarefa B (parte 2)

```
      function func(x)
         implicit real*8 (a-h,o-z)
         pi = 4.0d0*atan(1.0d0)
         func = exp(-x) * cos(2.0d0*pi*x)
         return
      end function func

      function fn(x0, n, h)
         implicit real*8 (a-h,o-z)
         fn = func(x0 + n*h)
         return
      end function fn

      function trint(a,b,n)
         implicit real*8 (a-h,o-z)
         trint=0
         h=(b-a)/n
         do i=1, n-1, 2
            x0=a+i*h
            trint=trint+(h/2.0d0)*(fn(x0, -1, h) + 2.0d0*fn(x0, 0, h)+f
     &n(x0,1, h))
         end do
         return
      end function

      function spint(a,b,n)
         implicit real*8 (a-h,o-z)
         spint=0
         h=(b-a)/n
         do i=1, n-1, 2
            x0=a+i*h
            spint=spint+(h/3.0d0)*(fn(x0, 1, h) + 4.0d0*fn(x0, 0, h) +
     &fn(x0,-1, h))
         end do
         return
      end function
```

Figura 5: Código-fonte da tarefa B (parte 3)

```
      function blint(a,b,n)
         implicit real*8 (a-h,o-z)
         blint=0
         h=(b-a)/n
         do i=0, n-4, 4
            x0=a+i*h
            blint=blint+(2.0d0*h/45.0d0)*(7.0d0*fn(x0, 0, h) + 32.0d0*f
     &n(x0,1, h) + 12.0d0*fn(x0, 2, h) + 32.0d0*fn(x0, 3, h) + 7.0d0*fn
     &(x0,4, h))
         end do
         return
      end function

      function anlint2(x)
         implicit real*8 (a-h,o-z)
         anlint2=0
         pi = 4.0d0*atan(1.0d0)
         anlint2=(exp(-x)/(1-4*(pi**2)))*(cos(2*pi*x) - 2*pi*sin(2
     &*pi*x))
         return
      end function
```

Verbete 2: Explicação do código da tarefa B

Inicialmente, define-se o vetor de divisões N e o valor analítico da integral. Em seguida, o vetor N é preenchido, e inicia-se um laço para calcular as integrais dos três métodos para cada N do vetor. Após calculadas as integrais, os resultados brutos são escritos no arquivo *saida-1-int-13687303.csv* e as diferenças absolutas entre os resultados calculados e o valor analítico são escritas no arquivo *saida-1-comp-13687303.csv*. Ambos os arquivos estão em formato CSV (valores separados por vírgula).
O cálculo das integrais é implementado em suas respectivas funções: a integral do trapézio na função *trint*, de Simpson na função *spint* e de Boole na função *blint*, utilizando as equações e limites de iteração citadas anteriormente no texto. Além disso, a fim de auxiliar o usuário, a integral analítica foi implementada na função *anlint2*.

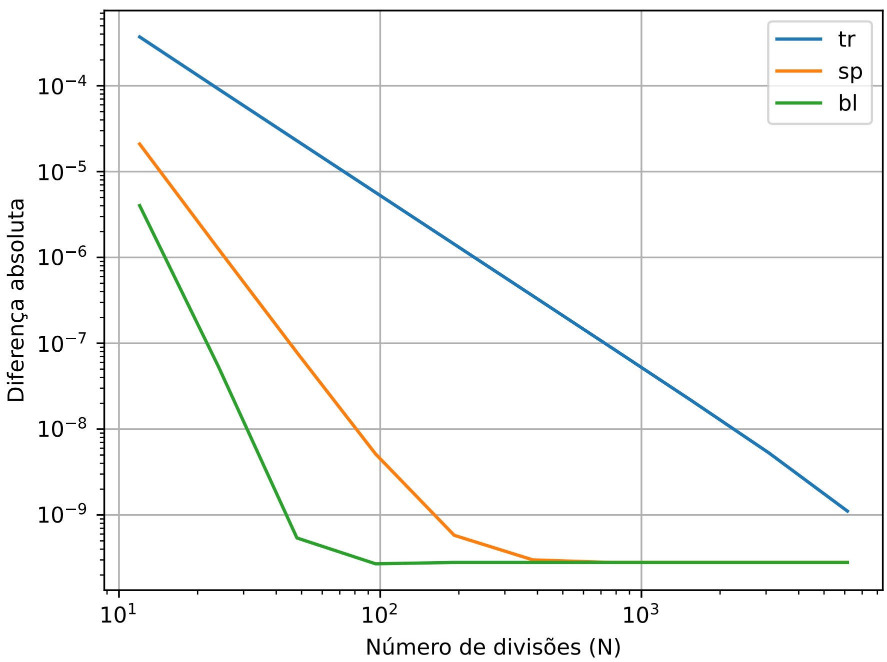
Tabela 3: Diferenças absolutas para cada método de integração

N	h	Trapézio (tr)	Simpson (sp)	Boole (bl)
12	0.08333333333	0.00037083460	0.00002096441	0.00000402326
24	0.04166666667	0.00009176391	0.00000125965	0.00000005400
48	0.02083333333	0.00002288231	0.00000007822	0.00000000054
96	0.01041666667	0.00000571672	0.00000000514	0.00000000027
192	0.00520833333	0.00000142874	0.00000000058	0.00000000028
384	0.00260416667	0.00000035696	0.00000000030	0.00000000028
768	0.00130208333	0.00000008903	0.00000000028	0.00000000028
1536	0.00065104167	0.00000002205	0.00000000028	0.00000000028
3072	0.00032552083	0.00000000530	0.00000000028	0.00000000028
6144	0.00016276042	0.00000000111	0.00000000028	0.00000000028

Fonte: Autoria própria

A partir das diferenças absolutas entre os métodos numéricos e o resultado analítico expostas na tabela 3, foi criado um gráfico das diferenças em função do número de divisões N, exposto na figura 6 em escala log-log, a fim de observar o desvio do resultado numérico em função de N. Ao analisar a figura, observa-se que o número de divisões ideal é 768, pois a maioria dos métodos de integração apresentam erro mínimo a partir desse N.

Figura 6: Desvios dos métodos de integração em função de N



Fonte: Autoria própria

4 Tarefa C

Nessa tarefa, foi criado um programa para calcular as raízes de um polinômio $p(x)$, dado pela equação 18, utilizando três métodos diferentes: método da busca direta ou bisseção, método de Newton-Raphson e método da secante. Analiticamente, as raízes do polinômio são $r_1 = -7$, $r_2 = 2$ e $r_3 = 9$ e a derivada do polinômio é dada pela equação 19.

$$p(x) = x^3 - 4x^2 - 59x + 126 \tag{18}$$

$$p'(x) = 3x^2 - 8x - 59 \tag{19}$$

No método da bisseção, inicia-se com um intervalo $[a, b]$ e, em seguida, define-se um ponto x_1 na metade desse intervalo. Se $p(a) \cdot p(x_1) < 0$, então a função inverte seu sinal no espaço entre a e x_1. Assim, define-se um novo ponto médio entre a e x_1, e verifica-se a condição de mudança de sinal novamente, repetindo o processo até que $|p(x_1)| = 0 + \epsilon$, quando pode-se afirmar que uma raíz foi encontrada com erro ϵ.

No método de Newton-Raphson, avalia-se a equação da reta tangente ao polinômio no ponto x_0:

$$y = p(x_0) + p'(x_0)(x - x_0)$$

Quando a $y = 0$, temos uma raíz e, então, pode ser obtida a seguinte relação:

$$x_1 = x_0 - \frac{p(x_0)}{p'(x_0)}$$

Esse processo pode ser repetido iterativamente para vários valores de x_i, e assim, tem-se a expressão iterativa para o método de Newton-Raphsn, dada pela equação 20.

$$x_{i+1} = x_i - \frac{p(x_i)}{p'(x_i)} \tag{20}$$

O método da secante é análogo ao método de Newton-Raphson. Pode-se avaliar o zero da equação da reta secante à função nos pontos $f(x_i)$ e $f(x_{i-1})$ iterativamente, conforme exposto na equação 21.

$$x_{i+1} = x_i - \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i) \tag{21}$$

O código elaborado para essa tarefa está exposto em quatro partes, nas figuras 7, 8, 9 e 10, respectivamente. A explicação do código é dada logo em seguida, no verbete 3.

Figura 7: Código-fonte da tarefa C (parte 1)

```
      program Raizes
         implicit real*8 (a-h, o-z)
         x0 = -10.0d0
         e = 1e-6
         open(20, file='saida-1-raizes-13687303.csv')
         write(20,*) 'i,', 'r1n,', 'r2n,', 'r3n,', 'r1d,', 'r2d,', 'r3d
     &,','r1s,', 'r2s,', 'r3s'
         do i=1, 7, 1
            r1n = xnewton(x0, i, e)
            r2n = xnewton(0.0d0, i, e)
            r3n = xnewton(10.0d0, i, e)

            r1d = xdireta(-10.0d0, 0.0d0, i, e)
            r2d = xdireta(0.0d0, 5.0d0, i, e)
            r3d = xdireta(5.0d0, 10.0d0, i, e)

            r1s = xsecante(-10.0d0, -5.0d0, i, e)
            r2s = xsecante(0.0d0, 5.0d0, i, e)
            r3s = xsecante(5.0d0, 10.0d0, i, e)

            write(20,50) i-1, r1n, r2n, r3n, r1d, r2d, r3d, r1s, r2s, r3
     &s
 50         format(I0, ",", F24.11, ",", F24.11, ",", F24.11, "," F24.1
     &1, ',', F24.11, ",", F24.11, ",", F24.11, ",", F24.11, ",", F24.1
     &1)
         end do
         close(20)
      end program

      function f(x)
         implicit real*8 (a-h, o-z)
         f=x**3 - 4*(x**2) - 59*x + 126
         return
      end function

      function diff(x)
         implicit real*8 (a-h, o-z)
         diff = 3*x**2 - 8*x - 59
         return
      end function
```

Figura 8: Código-fonte da tarefa C (parte 2)

```fortran
      function f(x)
         implicit real*8 (a-h, o-z)
         f=x**3 - 4*(x**2) - 59*x + 126
         return
      end function

      function diff(x)
         implicit real*8 (a-h, o-z)
         diff = 3*x**2 - 8*x - 59
         return
      end function

      function xnewton(x0, n, e)
         implicit real*8 (a-h,o-z)
         x = x0
         do iter=1, n, 1
            xnew = x - (f(x)/diff(x))
            if(abs(x - xnew) .lt. e) goto 100
            x = xnew
         end do
100      xnewton = x
         return
      end function
```

Figura 9: Código-fonte da tarefa C (parte 3)

```
      function xdireta(a,b,n,e)
         implicit real*8 (a-h, o-z)
         a_new = a
         b_new = b
         f_a = f(a_new)
         f_b = f(b_new)
         if(f_a*f_b .gt. 0.0d0) then
            write(*,*) "Erro: a função não muda de sinal em [a,b]."
            xdireta = 0.0d0
            return
         end if

         xinterv = abs(b_new-a_new)
         x0 = (a_new+b_new)/2.0d0
         f_x = f(x0)
         iter=1
99       if (iter .lt. n) then
            if(xinterv .le. e) then
               xdireta = x0
               return
            end if

            if(f_a * f_x > 0.0d0) then
               a_new = x0
               f_a = f_x
            else
               b_new = x0
               f_b = f_x
            end if

            xinterv = abs(b_new-a_new)
            x0 = (a_new+b_new)/2.0d0
            f_x = f(x0)
            iter = iter + 1
            goto 99
         end if
         xdireta = x0
         return
      end function
```

Figura 10: Código-fonte da tarefa C (parte 4)

```

      function xsecante(x0, x1, n, e)
         implicit real*8 (a-h, o-z)
         x0_new = x0
         x1_new = x1
         if(abs(f(x0)) .le. e) then
            xsecante = x0_new
            return
         end if
         if(abs(x1) .le. e) then
            xsecante = x1_new
            return
         end if
         do i=1, n+1, 1
            x2_new = (x0_new*f(x1_new) - x1_new*f(x0_new))/(f(x1_new)-f
     &(x0_new))
            if(abs(f(x2)) .le. e) then
               xsecante = x2_new
               return
            end if
            x0_new = x1_new
            x1_new = x2_new
         end do
         xsecante = x2_new
         return
      end function
```

Verbete 3: Explicação do código da tarefa C

Inicialmente, define-se o $x_0 = -10$ e a precisão $\epsilon = 10^{-6}$. Em seguida, é feito um laço de $i = 1$ a $i = 7$, para calcular as raízes para cada método com i iterações. Após calculadas as raízes dos intervalos de interesse, o resultado é escrito em um arquivo *saida-1-raizes-13687303.csv*, em formato CSV (valores separados por vírgula).
O cálculo das raízes é feito utilizando funções separadas para cada método: o polinômio é implementado na função f, sua derivada é $diff$, o método da busca direta é $xdireta$, o método da secante é $xsecante$ e o método de Newton-Raphson é $xnewton$.

Na tabela 4, são exibidos os resultados para as três raízes obtidas para cada método, para cada número de iterações. As raízes r1n, r2n e r3n são as 3 raízes obtidas pelo método de Newton-Raphson, as raízes r1d, r2d e r3d são as 3 raízes obtidas pelo método da busca direta e as raízes r1s, r2s e r3s são as raízes obtidas pelo método da secante. O índice i indica o número de iterações necessárias para gerar tal raíz.

Tabela 4: Raízes para cada método e número de iterações

i	r1n	r2n	r3n	r1d	r2d	r3d	r1s	r2s	r3s
0	-7.86915887850	2.13559322034	9.15527950311	-5.00000000000	2.50000000000	7.50000000000	-7.50002305954	1.88461538462	8.68922108576
1	-7.10646566346	1.99933084465	9.00471464883	-7.50000000000	1.25000000000	8.75000000000	-6.91915429979	2.00136536908	9.12656952039
2	-7.00191344530	1.99999998580	9.00000455770	-6.25000000000	1.87500000000	9.37500000000	-6.99335751505	2.00000470020	8.99167756119
3	-7.00000063531	1.99999998580	9.00000000000	-6.87500000000	2.18750000000	9.06250000000	-7.00009433390	1.99999999980	8.99978775386
4	-7.00000063531	1.99999998580	9.00000000000	-7.18750000000	2.03125000000	8.90625000000	-6.99999989112	2.00000000000	9.00000036325
5	-7.00000063531	1.99999998580	9.00000000000	-7.03125000000	1.95312500000	8.98437500000	-7.00000000000	2.00000000000	8.99999999998
6	-7.00000063531	1.99999998580	9.00000000000	-6.95312500000	1.99218750000	9.02343750000	-7.00000000000	2.00000000000	9.00000000000

Fonte: Autoria própria

A partir da análise dos dados expostos na tabela 4, é possível perceber que os dois métodos que oferecem maior precisão com o menor número de iterações é, em primeiro lugar, o método da secante, e em segundo lugar, o método de Newton-Raphson. Assim, é mais eficiente utilizar esses métodos para encontrar as raízes do polinômio.