UNIVERSIDADE DE SÃO PAULO - USP

INTRODUÇÃO À FÍSICA COMPUTACIONAL – 7600017 – 2025/2
PROF. FRANCISCO C. ALCARAZ

RELATÓRIO DO 3° PROJETO
JOÃO VITOR LIMA DE OLIVEIRA - 12694394

São Carlos

2025

Parte I

Introdução Geral

MOTIVAÇÃO

A análise de funções por meio de métodos numéricos é uma das ferramentas centrais da Física Computacional. Derivadas, integrais e equações algébricas estão no coração da formulação matemática de quase todos os fenômenos físicos, mas nem sempre as soluções exatas são acessíveis. Nesse cenário, técnicas aproximadas permitem transformar problemas complexos em algoritmos implementáveis em computador, abrindo caminho para simulações que seriam impossíveis de resolver de forma puramente analítica.

A derivação numérica, por exemplo, fornece meios de obter informações sobre taxas de variação mesmo quando só conhecemos valores discretos de uma função. Já a quadratura numérica possibilita calcular áreas, probabilidades e fluxos em situações onde não há primitiva conhecida. O estudo de raízes, por sua vez, conecta-se diretamente à busca por estados estacionários e soluções de equações transcendentais que emergem em mecânica, termodinâmica e teoria de campos.

Além da aplicação direta, a implementação desses métodos exige compreender limitações de precisão, erros de truncamento e estabilidade numérica. Esse exercício não apenas fortalece a intuição sobre os algoritmos, mas também ilustra a importância da expansão em série de Taylor como base conceitual unificadora.

Assim, este trabalho busca integrar teoria e prática computacional, explorando como procedimentos aparentemente simples, diferenças finitas, regras de quadratura e métodos iterativos de raízes, constituem a base para simulações mais sofisticadas. A motivação central está em compreender que dominar esses métodos é um passo fundamental para investigar problemas reais da física, onde a complexidade frequentemente supera as ferramentas analíticas tradicionais.

Parte II

Desenvolvimento

TAREFA - 1

ENUNCIADO

Figura 1 – Enunciado da Tarefa 1.

1. **Derivação numérica:** Escreva um código FORTRAN que forneça os dados da tabela I para as derivadas da função

$$f(x) = e^{2x^2} \tanh(2x) \tag{2}$$

para $x = \frac{1}{2}$. Na última linha escreva os valores numéricos exatos com precisão 10^{-11} obtidos mediante a expressão analítica que você deve derivar. Diga em cada caso qual o valor de h mais apropriado para uso. Explique seus resultados.

h	$f'_{2f}(x)$	$f'_{2t}(x)$	$f'_{3s}(x)$	$f'_{5s}(x)$	$f''_{3s}(x)$	$f''_{5s}(x)$
5^{-1}						
5^{-2}						
5^{-3}						
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
5^{-11}						
5^{-12}						
Exato						

Tabela I. Derivadas numéricas de $f(x)$ em (2) no ponto $x = \frac{1}{2}$ por meio de diferentes aproximações em função do passo h.

Fonte: Compilado pelo Autor.

CÓDIGO

Figura 2 – Função principal do código.

```fortran
        program main
        implicit real*8 (a-h,o-z)

        open(unit=1,file='saida-1-12694394.txt')
        write(1,3)

        x = 0.5d0
        do i = 1,12
            h = 5d0**(-i)
        write(1,7) i,f1(x,h),f2(x,h),f3(x,h),f4(x,h),f5(x,h),f6(x,h)
        end do
        write(1,3)

7       format('|',I3,6('|',F14.12),'|')
3       format(95('-'))
        close(1)
        end program main
```

Fonte: Compilado pelo Autor.

Figura 3 – Função estudada.

```
        function f(x)
        real*8 f,x

        f = exp(2d0*x*x)*tanh(2d0*x)
        return
        end function f
```

Fonte: Compilado pelo Autor.

Figura 4 – Derivadas númericas.

```fortran

        function f1(x,h)
        implicit real*8 (a-h,o-z)

        f1 = (f(x+h)-f(x))/h

        return
        end function f1

        function f2(x,h)
        real*8 f2,f,x,h

        f2 = (f(x)-f(x-h))/h

        return
        end function f2

        function f3(x,h)
        implicit real*8 (a-h,o-z)

        f3 = (f(x + h) - f(x - h))/(2d0*h)

        return
        end function f3

        function f4(x,h)
        implicit real*8 (a-h,o-z)

        f4 = (f(x-2d0*h)-8d0*f(x-h)+8d0*f(x+h)-f(x+2d0*h))/(12d0*h)
        return
        end function f4

        function f5(x,h)
        implicit real*8 (a-h,o-z)

        f5 = (f(x+h) -2*f(x) + f(x-h))/(h*h)

        return
        end function f5

        function f6(x,h)
        implicit real*8 (a-h,o-z)

        f6 =-f(x-2*h)+16*f(x-h)-30*f(x)+16*f(x+h)-f(x+2*h)
        f6=f6/(12*h*h)
        return
        end function f6
```

Fonte: Compilado pelo Autor.

Figura 5 – Segundas derivadas númericas.

```
function f5(x,h)
implicit real*8 (a-h,o-z)

f5 = (f(x+h) -2*f(x) + f(x-h))/(h*h)

return
end function f5

function f6(x,h)
implicit real*8 (a-h,o-z)

f6 =-f(x-2*h)+16*f(x-h)-30*f(x)+16*f(x+h)-f(x+2*h)
f6=f6/(12*h*h)
return
end function f6
```

Fonte: Compilado pelo Autor.

DESCRIÇÃO DO CÓDIGO

O programa `main` tem como objetivo calcular numericamente derivadas de primeira e segunda ordem da função

$$f(x) = e^{2x^2}\tanh(2x), \tag{1}$$

utilizando diferentes fórmulas de derivadas númericas. Essas aproximações são avaliadas para um ponto fixo $x = 0.5$, variando o passo h em potências decrescentes de 5, isto é:

$$h = 5^{-i}, \quad i = 1, 2, \ldots, 12. \tag{2}$$

Os resultados são escritos em um arquivo de saída denominado `saida-1-12694394.txt`. Ademais no início do código, é utilizada o comando:

```
implicit real*8 (a-h,o-z)
```

que define todas as variáveis cujos nomes começam com as letras de **a** a **h** e **o** a **z** como números reais de dupla precisão. Em seguida, o programa abre o arquivo de saída com o comando:

```
open(unit=1,file='saida-1-12694394.txt')
```

e escreve uma linha de separação utilizando o formato definido pelo rótulo 3, que imprime 95 hifens (`format(95('-'))`).

No bloco principal, é fixado o valor $x = 0.5$. O programa então entra em um laço de 12 iterações (`do i = 1,12`), no qual o passo h é definido conforme a Equação (2). Para cada valor de h, o programa calcula seis aproximações numéricas das derivadas da função $f(x)$ por meio das funções `f1` a `f6`, gravando os resultados no arquivo de saída com a instrução:

```
write(1,7) i,f1(x,h),f2(x,h),f3(x,h),f4(x,h),f5(x,h),f6(x,h)
```

O formato identificado pelo rótulo 7:

```
7   format('|',I3,6('|',F16.12),'|')
```

organiza os resultados em colunas, contendo o número da iteração seguido das seis aproximações, todas com 12 casas decimais. Após o laço, o programa imprime novamente a linha de separação e fecha o arquivo com o comando:

```
close(1)
```

Funções auxiliares

As funções `f1` a `f6` implementam diferentes fórmulas de diferenças finitas para a primeira e segunda derivadas de $f(x)$.

1. **Função** `f(x)`
   Define a função principal:
$$f(x) = e^{2x^2}\tanh(2x). \tag{3}$$

2. **Função** `f1(x,h)` — Derivada para frente de 2 pontos (1ª derivada)
$$f_1'(x) \approx \frac{f(x+h) - f(x)}{h}. \tag{4}$$
   Essa é a aproximação de primeira ordem para a derivada.

3. **Função** `f2(x,h)` — Derivada para trás de 2 pontos (1ª derivada)
$$f_2'(x) \approx \frac{f(x) - f(x-h)}{h}. \tag{5}$$
   Também é de primeira ordem, mas utiliza pontos à esquerda de x.

4. **Função** `f3(x,h)` — Derivada simétrica de 3 pontos (1ª derivada)
$$f_3'(x) \approx \frac{f(x+h) - f(x-h)}{2h}. \tag{6}$$
   É uma aproximação de segunda ordem, mais precisa que as anteriores.

5. **Função** `f4(x,h)` — Derivada simétrica de 5 pontos de quarta ordem (1ª derivada)
$$f_4'(x) \approx \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h}. \tag{7}$$
   Fornece uma aproximação de quarta ordem, mais precisa para a derivada primeira.

6. **Função** `f5(x,h)` — Derivada segunda simétrica de três pontos (2ª derivada)
$$f_5''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \tag{8}$$
   Essa é uma aproximação de segunda ordem para a derivada segunda.

7. **Função** `f6(x,h)` — Derivada segunda simétrica de 5 pontos de quarta ordem (2ª derivada)

$$f_6''(x) \approx \frac{-f(x+2h) + 16f(x+h) - 30f(x) + 16f(x-h) - f(x-2h)}{12h^2}. \tag{9}$$

Essa é uma aproximação de quarta ordem, ainda mais precisa.

RESULTADOS

Tabela 1 – Valores de $f'(x)$ obtidos por diferentes métodos numéricos.

$h = 5^{-i}$	$f'_{2f}(x)$	$f'_{2t}(x)$	$f'_{3s}(x)$	$f'_{5s}(x)$
5^{-1}	5.51662120	3.06345709	4.29003915	3.81049648
5^{-2}	4.13920591	3.68318855	3.91119723	3.89603900
5^{-3}	3.94222385	3.85128585	3.89675485	3.89615405
5^{-4}	3.90527099	3.88708551	3.89617825	3.89615423
5^{-5}	3.89797373	3.89433665	3.89615519	3.89615423
5^{-6}	3.89651798	3.89579056	3.89615427	3.89615423
5^{-7}	3.89622697	3.89608149	3.89615423	3.89615423
5^{-8}	3.89616878	3.89613968	3.89615423	3.89615423
5^{-9}	3.89615714	3.89615132	3.89615423	3.89615423
5^{-10}	3.89615481	3.89615365	3.89615423	3.89615423
5^{-11}	3.89615435	3.89615411	3.89615423	3.89615424
5^{-12}	3.89615418	3.89615423	3.89615421	3.89615421
Exato	3.89615422946			

Fonte: Compilado pelo Autor

Tabela 2 – Valores de $f''(x)$ obtidos por diferentes métodos numéricos.

$h = 5^{-i}$	$f''_{3s}(x)$	$f''_{5s}(x)$
5^{-1}	12.26582057	11.19954015
5^{-2}	11.40043402	11.36563532
5^{-3}	11.36724925	11.36586850
5^{-4}	11.36592409	11.36586887
5^{-5}	11.36587108	11.36586887
5^{-6}	11.36586891	11.36586877
5^{-7}	11.36587027	11.36586936
5^{-8}	11.36582690	11.36579019
5^{-9}	11.36379402	11.36301757
5^{-10}	11.39259314	11.40141640
5^{-11}	11.64670302	11.42612153
5^{-12}	-13.23488980	-23.16105715
Exato	11.36586887467	

Fonte: Compilado pelo Autor

Tabela 3 – Valores de $f'(x)$ e $f''(x)$ obtidos por diferentes métodos numéricos.

$h = 5^{-i}$	$f'_{2f}(x)$	$f'_{2t}(x)$	$f'_{3s}(x)$	$f'_{5s}(x)$	$f''_{3s}(x)$	$f''_{5s}(x)$
5^{-1}	5.51662120	3.06345709	4.29003915	3.81049648	12.26582057	11.19954015
5^{-2}	4.13920591	3.68318855	3.91119723	3.89603900	11.40043402	11.36563532
5^{-3}	3.94222385	3.85128585	3.89675485	3.89615405	11.36724925	11.36586850
5^{-4}	3.90527099	3.88708551	3.89617825	3.89615423	11.36592409	11.36586887
5^{-5}	3.89797373	3.89433665	3.89615519	3.89615423	11.36587108	11.36586887
5^{-6}	3.89651798	3.89579056	3.89615427	3.89615423	11.36586891	11.36586877
5^{-7}	3.89622697	3.89608149	3.89615423	3.89615423	11.36587027	11.36586936
5^{-8}	3.89616878	3.89613968	3.89615423	3.89615423	11.36582690	11.36579019
5^{-9}	3.89615714	3.89615132	3.89615423	3.89615423	11.36379402	11.36301757
5^{-10}	3.89615481	3.89615365	3.89615423	3.89615423	11.39259314	11.40141640
5^{-11}	3.89615435	3.89615411	3.89615423	3.89615424	11.64670302	11.42612153
5^{-12}	3.89615418	3.89615423	3.89615421	3.89615421	-13.23488980	-23.16105715
Exato	3.89615422946				11.36586887467	

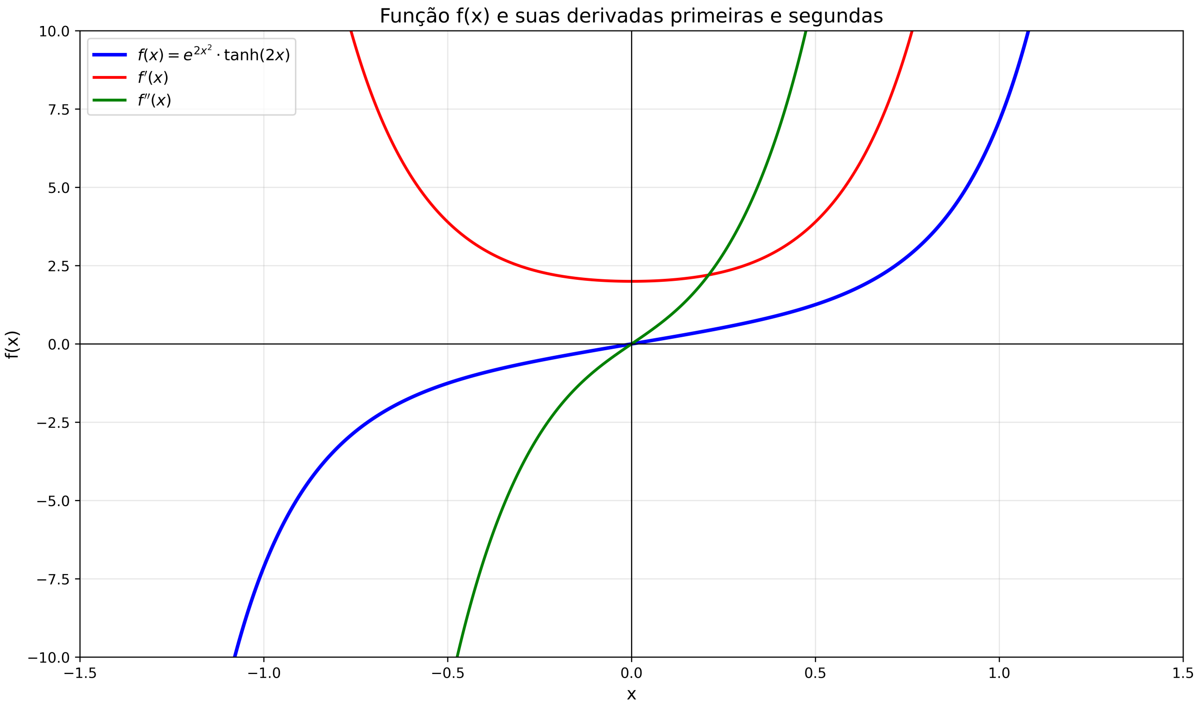
Fonte: Compilado pelo Autor

Figura 6 – Dados salvos no arquivo de saida para a Tarefa - 1.

1	5.516621203949	3.063457090490	4.290039147220	3.810496475878	12.265820567294	11.199540147581
2	4.139205906346	3.683188545531	3.911197225938	3.896039003775	11.400434020386	11.365635319811
3	3.942223846331	3.851285852349	3.896754849340	3.896154046352	11.367249247673	11.365868502851
4	3.905270991771	3.887085513233	3.896178252502	3.896154229172	11.365924086192	11.365868874216
5	3.897973729756	3.894336651011	3.896155190384	3.896154229465	11.365871080676	11.365868869084
6	3.896517975707	3.895790560097	3.896154267902	3.896154229465	11.365868912271	11.365868772229
7	3.896226972566	3.896081489426	3.896154230996	3.896154229463	11.365870267524	11.365869364022
8	3.896168777797	3.896139681280	3.896154229539	3.896154229474	11.365826899437	11.365790194676
9	3.896157138670	3.896151320407	3.896154229539	3.896154229575	11.363794020364	11.363017573495
10	3.896154813707	3.896153647105	3.896154230406	3.896154230587	11.392593140570	11.401416400438
11	3.896154351837	3.896154113312	3.896154232574	3.896154235285	11.646703024747	11.426121528066
12	3.896154178364	3.896154232574	3.896154205469	3.896154209987	-13.234889800848	-23.161057151485

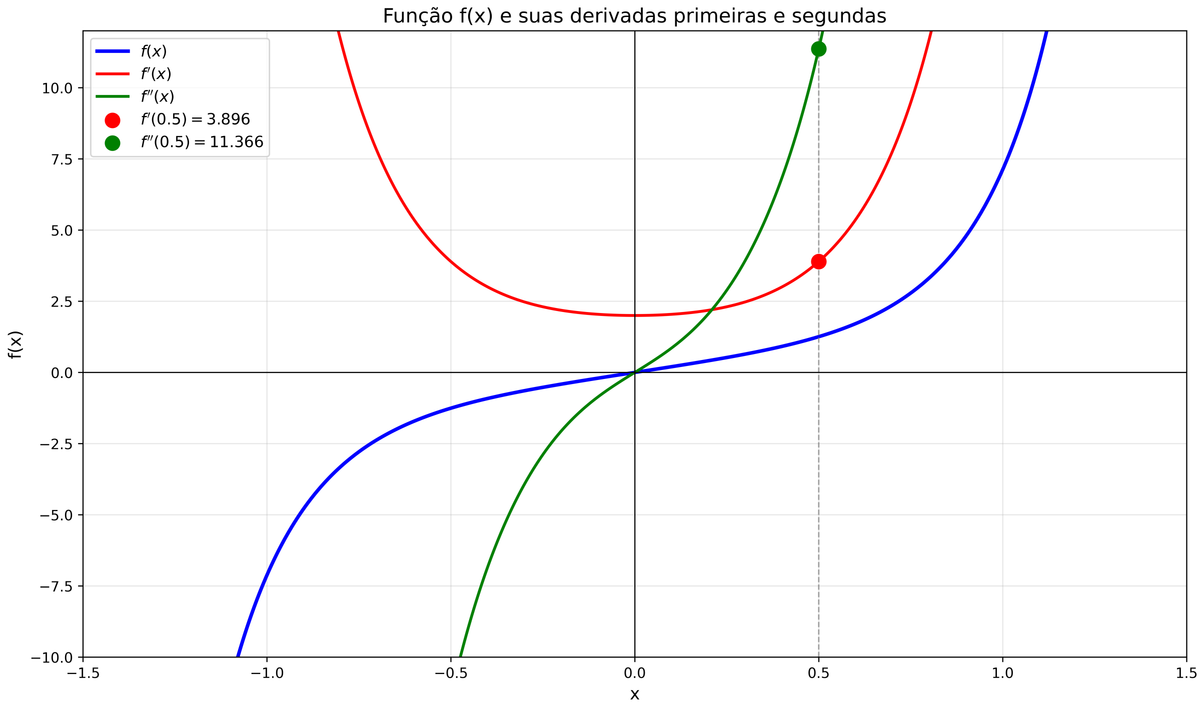
Fonte: Compilado pelo Autor.

Figura 7 – Gráfico das funções estudadas.



Fonte: Compilado pelo Autor.

Figura 8 – Gráfico das funções estudadas marcando o ponto $x = 0.50$.



Fonte: Compilado pelo Autor.

TAREFA - 2

ENUNCIADO

Figura 9 – Enunciado da Tarefa 2.

2. **Quadratura numérica:** Escreva um código FORTRAN que calcule uma aproximação da integral

$$I = \int_0^{2\pi} e^{-x} \sin(x)\, dx \qquad (3)$$

usando os métodos e intervalos para preencher a tabela II. Na última linha da tabela escreva o valor numérico exato com precisão 10^{-11} obtido pela expressão analítica que você deve calcular. Aponte o valor ótimo de h em cada um dos casos e discuta seus resultados.

Fonte: Compilado pelo Autor.

Figura 10 – Tarefa enunciado da Tarefa 2.

$2\pi \times h^{-1}$	Regra do trapézio	Regra de Simpson	Regra de Boole
2^2			
2^3			
2^4			
$\vdots$	$\vdots$	$\vdots$	$\vdots$
2^{12}			
2^{13}			
Exato			

Tabela II. Integral numérica de $f(x)$ em (3) no intervalo $[0, 1]$ por meio de diferentes métodos e partições do intervalo h.

Fonte: Compilado pelo Autor.

CÓDIGO

Figura 11 – Função principal do código.

```
      program main
      implicit real*8 (a-h,o-z)

C     Constantes

      pi = acos(-1d0)

      a = 0d0
      b = 2d0*pi

C     Variáveis

      rval = 0.5d0*(1d0-exp(-2d0*pi))

      open(unit=1,file='saida-1-12694394.txt')

      do i = 2,13
      n = 2**i
      write(1,7) i,trap(a,b,n),simp(a,b,n),boole(a,b,n)
      end do
      write(1,3) rval
3     format(F16.11)
7     format(I3,3(',',F16.12))

      close(1)
      end program main
```

Fonte: Compilado pelo Autor.

Figura 12 – Função principal do código.

```
      function f(x)
      implicit real*8 (a-h,o-z)

      f = exp(-x)*sin(x)
      return
      end function f
```

Fonte: Compilado pelo Autor.

Figura 13 – Função principal do código.

```
function trap(a,b,n)
implicit real*8 (a-h,o-z)

h = (b-a)/n
trap = 0d0
do i = 1,n
    x = a + (i-1)*h
    rr = 0.5d0*h*(f(x+h)+f(x))
    trap = trap + rr
end do

return
end function trap
```

Fonte: Compilado pelo Autor.

Figura 14 – Função principal do código.

```
function boole(a,b,n)
implicit real*8 (a-h,o-z)


h = (b-a)/n
boole = 0d0
do i = 1,n
x = a+(i-1)*h
r1 = (2d0/45d0)*h
r2 =(7d0*f(x-2d0*h)+7d0*f(x+2d0*h))
r3 = (32d0*f(x-h)+12*f(x+h)+32d0*f(x+h))

boole = boole + r1*(r2 + r3)/4d0
end do

return
end function boole
```

Fonte: Compilado pelo Autor.

Figura 15 – Função principal do código.

```
        function simp(a,b,n)
        implicit real*8 (a-h,o-z)


        h = (b-a)/n
        simp = 0d0

        do i = 1,n
        x = a+(i-1)*h
        rr = (3d0*h/8d0)*(f(x)+3d0*f(x+h)+3d0*f(x+2d0*h)+f(x+3d0*h))
        simp = simp + rr/3d0
        end do

        return
        end function simp
```

Fonte: Compilado pelo Autor.

DESCRIÇÃO DO CÓDIGO

O programa `main` tem como objetivo calcular numericamente a integral da função

$$I = \int_0^{2\pi} e^{-x} \sin(x) dx \tag{10}$$

utilizando diferentes métodos de integração numérica: regra do trapézio, regra de Simpson 3/8 e regra de Boole. As aproximações são avaliadas para valores de $n = 2^i$, com $i = 2, 3, \ldots, 13$, permitindo analisar a convergência dos métodos à medida que o número de subdivisões aumenta.

No início do código, é utilizada a diretiva:

```
implicit real*8 (a-h,o-z)
```

que define todas as variáveis cujos nomes começam com as letras de **a** a **h** e **o** a **z** como números reais de dupla precisão. Em seguida, o programa define constantes:

```
pi = acos(-1d0)
a = 0d0
b = 2d0*pi
```

e o valor exato para a integral estudada:

```
rval = 0.5d0*(1d0-exp(-2d0*pi))
```

O arquivo de saída é aberto com o comando:

```
open(unit=1,file='saida-1-12694394.txt')
```

No bloco principal, o programa entra em um laço:

```
do i = 2,13
    n = 2**i
    write(1,7) i,trap(a,b,n),simp(a,b,n),boole(a,b,n)
end do
```

Em cada iteração, $n = 2^i$ define o número de subdivisões, e as funções `trap`, `simp` e `boole` calculam a integral usando as respectivas regras. Os resultados são escritos no arquivo de saída com o formato:

```
7 format(I3,3(',',F16.12))
```

Após o laço, o valor de referência `rval` é escrito no arquivo com o formato:

```
3 format(F16.11)
```

Por fim, o arquivo é fechado com:

```
close(1)
```

Funções auxiliares

As funções implementam diferentes métodos de integração numérica sobre $[a, b]$.

1. **Função** `f(x)`
   Define a função a ser integrada:
$$f(x) = e^{-x}\sin(x) \tag{11}$$

2. **Função** `trap(a,b,n)` — Regra do Trapézio
   Aproxima a integral por:
$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} \frac{h}{2}\Big(f(x_i) + f(x_{i+1})\Big), \tag{12}$$
   onde $h = (b-a)/n$ e $x_i = a + (i-1)h$.

3. **Função** `simp(a,b,n)` — Regra de Simpson 3/8
   Aproximação de quarta ordem que utiliza quatro pontos por subintervalo:
$$\int_{x_0}^{x_0+3h} f(x)\,dx \approx \sum_{i=1}^{n} \frac{3h}{8}\Big(f(x_i) + 3f(x_{i+1}) + 3f(x_{i+2}) + f(x_{i+3})\Big). \tag{13}$$
4. **Função** `boole(a,b,n)` — Regra de Boole
   Fórmula de quinta ordem que utiliza cinco pontos igualmente espaçados:
$$\int_{x_0}^{x_0+4h} f(x)\,dx \approx \sum_{i=1}^{n} \frac{2h}{45}\Big(7f(x_i) + 32f(x_{i+1}) + 12f(x_{i+2}) + 32f(x_{i+3}) + 7f(x_{i+4})\Big). \tag{14}$$

Dessa forma, o programa permite calcular a integral de $f(x)$ sobre $[0, 2\pi]$ usando diferentes métodos, comparando a precisão e convergência dos esquemas numéricos conforme aumenta o número de subdivisões.

RESULTADOS

Tabela 4 – Raízes de $f(x)$ em (4) por meio de diferentes métodos e números de iterações.

Iteração 2^i	**Trapézio**	**Simpson 3/8**	**Boole**
2	0.31242555	0.14946221	-2.98119765
3	0.44884307	0.30210410	-0.39096151
4	0.48630565	0.41986759	0.33488498
5	0.49586365	0.47382433	0.46461834
6	0.49826485	0.49194786	0.49118674
7	0.49886587	0.49717723	0.49718160
8	0.49901617	0.49857980	0.49860535
9	0.49905375	0.49894285	0.49895230
10	0.49906315	0.49903519	0.49903794
11	0.49906550	0.49905848	0.49905921
12	0.49906608	0.49906432	0.49906451
13	0.49906623	0.49906579	0.49906584
Exato	0.49906627863		

Fonte: Compilado pelo Autor

Figura 16 – Dados salvos no arquivo de saida para a Tarefa - 2.

```
 2,  0.312425554409,  0.149462211019, -2.981197653506
 3,  0.448843070178,  0.302104096766, -0.390961511237
 4,  0.486305646676,  0.419867591325,  0.334884981923
 5,  0.495863645010,  0.473824332391,  0.464618339284
 6,  0.498264845766,  0.491947864702,  0.491186741076
 7,  0.498865872096,  0.497177231472,  0.497181597353
 8,  0.499016173981,  0.498579799314,  0.498605353674
 9,  0.499053752282,  0.498942847897,  0.498952302634
10,  0.499063147034,  0.499035192477,  0.499037939948
11,  0.499065495733,  0.499058478404,  0.499059213277
12,  0.499066082909,  0.499064324982,  0.499064514703
13,  0.499066229703,  0.499065789771,  0.499065837952
 0.49906627863
```

Fonte: Compilado pelo Autor.

TAREFA - 3

ENUNCIADO

Figura 17 – Enunciado da Tarefa 3.

3. **Raízes de funções:** Faça um programa que calcule as raízes reais de

$$f(x) = 0.042 - 0.13x - 0.6x^2 + x^3 \tag{4}$$

no intervalo $x \in [-5, 5]$, preenchendo a tabela III abaixo. Eleja uma tolerância de 10^{-6}. Inicie fazendo uma **busca direta** usando como ponto inicial $x = -5$ e um espaçamento inicial de 0,01. Quando verificar a mudança de sinal em $f(x)$, use o intervalo correspondente $]x_-, x_+[$ para iniciar o método da bisseção e conte as iterações a partir daí. Para o método da secante, use os extremos desse intervalo como pontos iniciais x_{-1} e x_0. Para o método de Newton-Raphson, use um dos extremos como ponto inicial x_0. Finalmente, na última linha da tabela coloque os valores exatos. (Note que o número total de tabelas é igual ao número de raízes dentro do intervalo $x \in [-10, 10]$.)

Fonte: Compilado pelo Autor.

Figura 18 – Tarefa enunciado da Tarefa 3.

Iteração	Bisseção		Newton-Raphson	Método da Secante	
	$x_{i,-}$	$x_{i,+}$	x_i	x_{i-1}	x_i
$i = 0$					
1					
2					
3					
4					
5					
$\vdots$					
Exato					

Tabela III. Raízes de $f(x)$ em (4) por meio de diferentes métodos e números de iterações.

Fonte: Compilado pelo Autor.

CÓDIGO

DESCRIÇÃO DO CÓDIGO

O programa `main` tem como objetivo determinar numericamente as raízes da função

$$f(x) = 0.042 - 0.13x - 0.6x^2 + x^3 \tag{15}$$

utilizando três métodos iterativos de busca de raízes: **bisseção**, **Newton-Raphson** e **secante**. Cada método é implementado em uma função separada, e seus resultados são exibidos na tela e gravados em arquivos distintos. O código também avalia a convergência dos métodos a partir do número de iterações até atingir a tolerância estabelecida.

No início do programa, é utilizada a diretiva:

```
implicit real*8 (a-h,o-z)
```

que define todas as variáveis cujos nomes começam com as letras de **a** a **h** e **o** a **z** como números reais de dupla precisão. Em seguida, são definidas as constantes principais:

```
a = 0.5d0
b = 5d0
tol = 1d-6
```

onde a e b representam os limites iniciais do intervalo de busca e `tol` define a tolerância do erro desejado.

O programa principal chama as três funções numéricas responsáveis por calcular as raízes da função `f(x)`:

```
write(*,1) 'Bisseção: ', bissec(a,b,tol)
write(*,1) 'Newton-Raphson: ', raphson(a,tol)
write(*,1) 'Secante: ', secante(a,tol)
```

Cada chamada executa o método correspondente e imprime o resultado com o formato:

```
1 format(A15,F14.12)
```

permitindo visualizar o valor da raiz com 12 casas decimais de precisão.

Funções auxiliares

As funções implementam tanto a função principal e sua derivada, quanto os três métodos iterativos de determinação de raízes.

1. **Função** `f(x)`
   Define a função cúbica a ser analisada:

$$f(x) = 0.042 - 0.13x - 0.6x^2 + x^3 \tag{16}$$

2. **Função** `df(x)`
   Define a derivada de $f(x)$, necessária para o método de Newton-Raphson:

$$f'(x) = -0.13 - 1.2x + 3x^2 \tag{17}$$

3. **Função** `bissec(a,b,tol)` — Método da Bisseção
   O método da bisseção busca um intervalo $[a, b]$ onde há troca de sinal em $f(x)$. Inicialmente, o código realiza uma varredura incremental com passo `stp = 0.01`, até encontrar um subintervalo onde $f(a)f(a + stp) < 0$. Em seguida, aplica-se o procedimento iterativo:

$$c = \frac{a+b}{2}, \qquad \text{se } f(a)f(c) < 0,\ b = c,\ \text{senão } a = c \tag{18}$$

   até que $|b - a| <$ `tol`. Os valores intermediários de c são gravados no arquivo:

```
        open(unit=13,file='saida-1-12694394.txt')
```

   com o formato:

```
        14 format(I3,F16.12)
```

4. **Função** `raphson(x,tol)` — Método de Newton-Raphson
   Este método utiliza a fórmula iterativa:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{19}$$

   As iterações continuam enquanto $|f(x)/f'(x)| >$ `tol`. A cada passo, o par (`iter`, x_k) é escrito no arquivo:

```
        open(unit=15,file='saida-2-12694394.txt')
```

   usando o formato:

```
        16 format(I3,F16.12)
```

5. **Função** `secante(x,tol)` — Método da Secante
   Este método aproxima a derivada por diferenças finitas entre dois pontos consecutivos, conforme:
   $$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \tag{20}$$
   Sendo `stp = 0.01` o deslocamento inicial. O processo é repetido até que $|x_{k+1} - x_k| <$ `tol`. Os valores iterativos são gravados em:

```
        open(unit=17,file='saida-3-12694394.txt')
```

   com o formato:

```
        18 format(I3,F16.12)
```

Assim, o programa permite determinar numericamente as raízes de uma função cúbica, comparando a eficiência e a convergência dos três métodos clássicos de busca de raízes: bisseção, Newton-Raphson e secante. Cada método gera um arquivo contendo as iterações realizadas, permitindo visualizar o comportamento da convergência.

RESULTADOS

As tabelas abaixo apresentam os resultados obtidos para as três raízes da função $f(x) = 0.042 - 0.13x - 0.6x^2 + x^3$ utilizando os métodos de Bissecção, Newton-Raphson e Secante.

Tabela 5 – Raízes de $f(x)$ por meio de diferentes métodos e números de iterações.

Iteração	Bissecção	Newton	Secante
0	-0.29500000	-0.3000000	-0.3000000
1	-0.29750000		
2	-0.29875000		
3	-0.29937500		
4	-0.29968750		
5	-0.29984375		
6	-0.29992188		
7	-0.29996094		
8	-0.29998047		
9	-0.29999023		
10	-0.29999512		
11	-0.29999756		
12	-0.29999878		
13	-0.29999939		
Raiz Aproximada	-0.3		

Fonte: Compilado pelo Autor.

Tabela 6 – Raízes de $f(x)$ por meio de diferentes métodos e números de iterações.

Iteração	Bissecção	Newton	Secante
0	0.1950000000	0.19999938960	0.200000000
1	0.1975000000		
2	0.1987500000		
3	0.1993750000		
4	0.1996875000		
5	0.1998437500		
6	0.1999218750		
7	0.1999609375		
8	0.1999804688		
9	0.1999902344		
10	0.1999951172		
11	0.1999975586		
12	0.1999987793		
13	0.1999993896		
Raiz Aproximada	0.2		

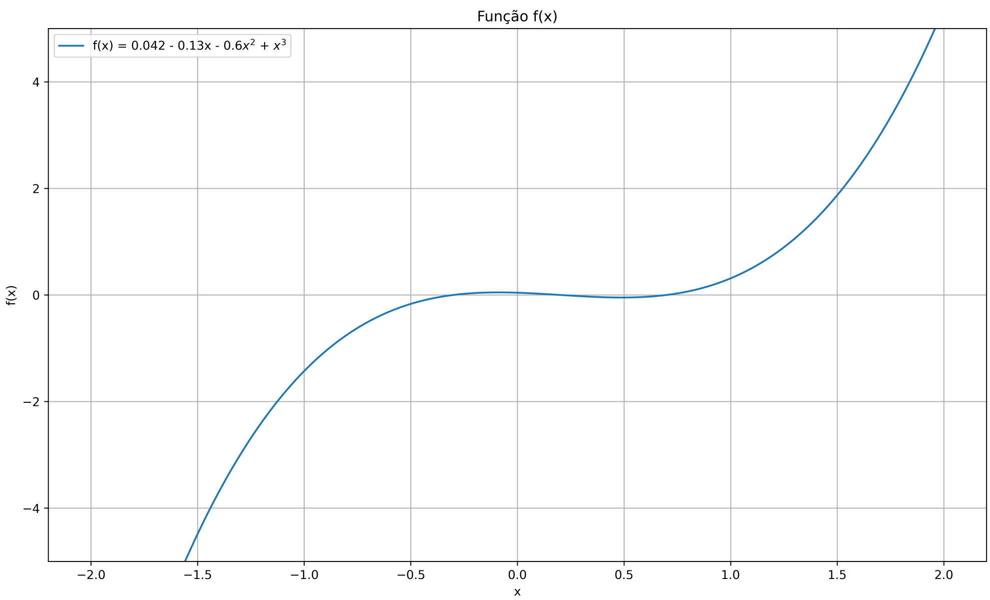
Fonte: Compilado pelo Autor.

Tabela 7 – Raízes de $f(x)$ por meio de diferentes métodos e números de iterações.

Iteração	Bissecção	Newton	Secante
0	0.695000000000	0.69999939	0.70000000
1	0.697500000000		
2	0.698750000000		
3	0.699375000000		
4	0.699687500000		
5	0.699843750000		
6	0.699921875000		
7	0.699960937500		
8	0.699980468750		
9	0.699990234375		
10	0.699995117188		
11	0.699997558594		
12	0.699998779297		
13	0.699999389648		
Raiz Aproximada	0.7		

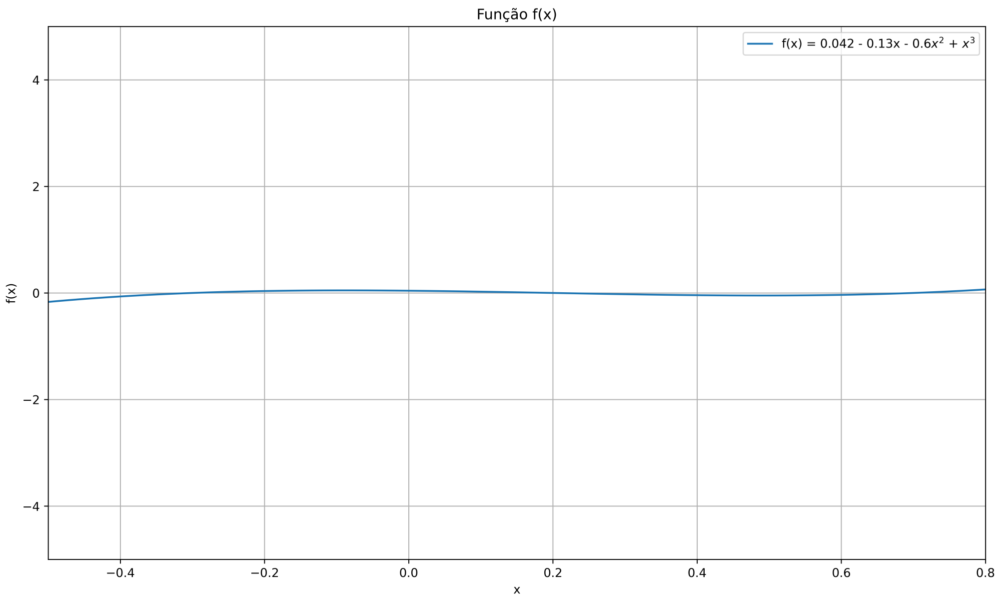
Fonte: Compilado pelo Autor.

Figura 19 – Gráfico da função f.



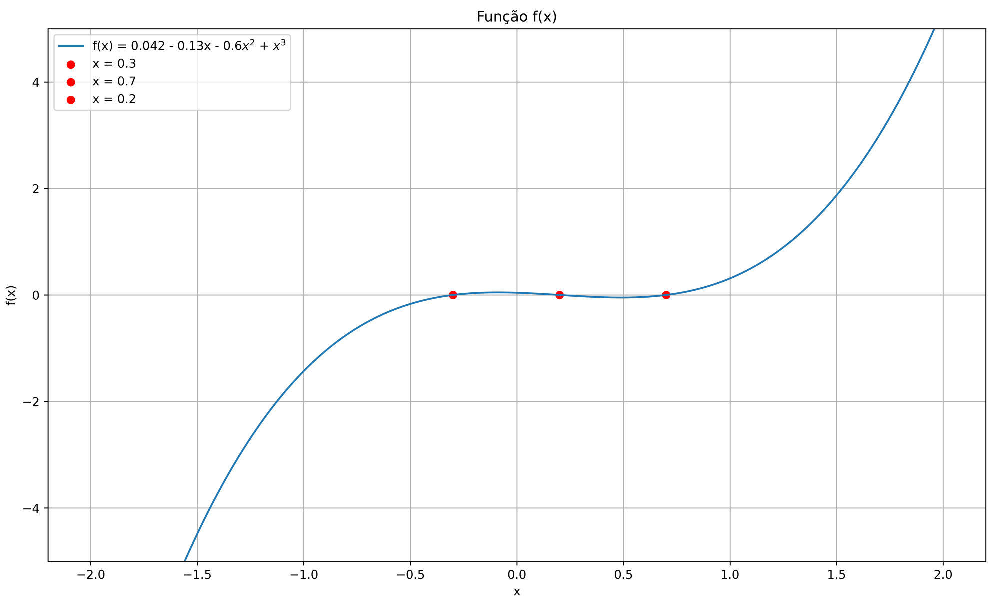
Fonte: Compilado pelo Autor.

Figura 20 – Gráfico da função f.



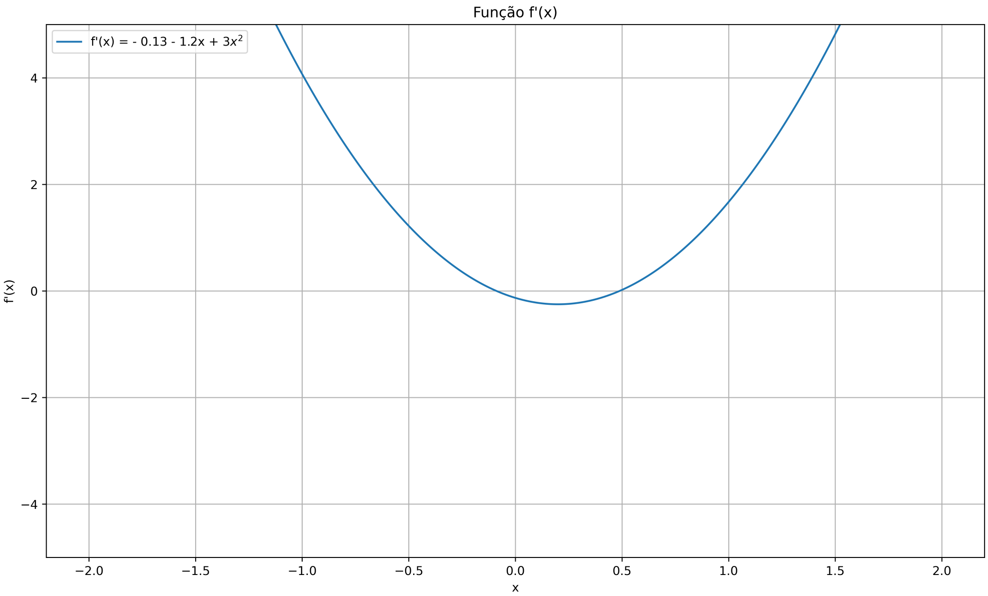
Fonte: Compilado pelo Autor.

Figura 21 – Gráfico da função f.



Fonte: Compilado pelo Autor.

Figura 22 – Gráfico da função f.



Fonte: Compilado pelo Autor.

Ademais, é visto nos terminais os seguintes valores.

Figura 23 – Terminal a=-5.

```
~/root/Graduação/Semestre 2 - 2025/IntroFiscomp/Projeto-3/tarefa-3  ./tarefa-3-12694394.exe
   Bisseção: -.299999694824
Newton-Raphson:-.300000000000
      Secante: -.300000000000
~/root/Graduação/Semestre 2 - 2025/IntroFiscomp/Projeto-3/tarefa-3
```

Fonte: Compilado pelo Autor.

Figura 24 – Terminal a=0.

```
~/root/Graduação/Semestre 2 - 2025/IntroFiscomp/Projeto-3/tarefa-3  ./tarefa-3-12694394.exe
   Bisseção: 0.199999694824
Newton-Raphson:0.200000000000
      Secante: 0.200000000000
~/root/Graduação/Semestre 2 - 2025/IntroFiscomp/Projeto-3/tarefa-3
```

Fonte: Compilado pelo Autor.

Figura 25 – Terminal a=0.50.

```
 ~/root/Graduação/Semestre 2 - 2025/IntroFiscomp/Projeto-3/tarefa-3  ./tarefa-3-12694394.exe
   Bisseção: 0.699999694824
Newton-Raphson:0.700000000001
      Secante: 0.700000000000
 ~/root/Graduação/Semestre 2 - 2025/IntroFiscomp/Projeto-3/tarefa-3 
```

Fonte: Compilado pelo Autor.