Universidade de São Paulo
Instituto de Física de São Carlos

Projeto 1

Pedro Calligaris Delbem 5255417

Professor: Francisco Alcaraz

Outubro de 2023

Sumário

1 Tarefa A

Tarefa: Escreva um codigo FORTRAN77 que operando em precisão dupla forneça os dados da tabela abaixo para as derivadas da função f(x) = $e^{x/2}tan(2x)$ para x = 1/2. Escreva apenas os desvios em relação aos resultados exatos. Na última linha escreva os valores numéricos com precisão 10^{-11} obtidos mediante a expressão analítica que você deve derivar.

h	derivada simétrica 3 pontos	derivada p/frente 2 pontos	derivada p/traz 2 pontos	derivada simétrica 5 pontos	derivada segunda simétrica 5 pontos	derivada terceira anti-simétrica 5 pontos
0.5						
0.2						
0.1						
0.05						
0.01						
0.005						
0.001						
0.0005						
0.0001						
0.00005						
0.00001						
0.000001						
0.0000001						
0.00000001						
EXATOS						

Código Escrito:

```
      program derivada

      implicit real*8 (a-h,o-z)
      dimension val(14)

c     define os valores de h que serao utilizados
      val(1) = 0.5d0
      val(2) = 0.2d0
      val(3) = 0.1d0
      val(4) = 0.05d0
      val(5) = 0.01d0
      val(6) = 5*10d-4
      val(7) = 10d-4
      val(8) = 5*10d-5
      val(9) = 10d-5
      val(10) = 5*10d-6
      val(11) = 10d-6
      val(12) = 10d-7
      val(13) = 10d-8
      val(14) = 10d-9

```

```
c     abre o arquivo de saida
      open(unit=1,file='saida-5255417')

c     escreve o cabecalho da tabela
      write(1,*)'|          h                fs3                   ff2
     1                ft2                   fs5                 f2s5
     2           f3as5           |'

c     loop que imprime os valores da tabela das derivadas para cada h
      do i=1,14

c        define como v o valor de h para o loop atual
         v = val(i)

c        defiene o valor real das derivadas
         d1 = 9.796782013838
         d2 = 64.098324549472
         d3 = 671.514613457866

c        escreve os valores de cada derivada para o valor corrente de h
         write(1,1) v,abs(fs3(v)-d1),abs(ff2(v)-d1),abs(ft2(v)-d1),abs(fs
     1   5(v)-d1),abs(f2s5(v)-d2),abs(f3as5(v)-d3)

      end do

c     escreve o fim da tabela
      write(1,*)'EXATO: 9.796782013838 | 9.796782013838 | 9.796782013838 | 9.7
     296782013838 | 64.098324549472 | 64.098324549472 | 671.514613457866
     3'

c     fecha o arquivo de saida
      close(1)

c     formata as escritas
1     format(7('|',f20.10),'|')

      end program

c     define a funcao de 1/2 + h
      real*8 function f(h)
      implicit real*8 (a-h,o-z)

         f = dexp( (0.5d0 + h)/2.0d0 )*dtan( 2.0d0*(0.5d0 + h) )

      end function

c     define a derivada para traz de 2 pontos
      real*8 function ft2(h)
      implicit real*8 (a-h,o-z)

         ft2 = ( f(0.0d0*h)-f(-1.0d0*h) )/h

      end function

c     define a derivada para frente de 2 pontos
      real*8 function ff2 (h)
      implicit real*8 (a-h,o-z)

```

```
        ff2 = ( f(1.0d0*h)-f(0.0d0*h) )/h

      end function

c     define a derivada simetrica de 3 pontos
      real*8 function fs3 (h)
      implicit real*8 (a-h,o-z)

        fs3 = ( f(1.0d0*h)-f(-1.0d0*h) )/(2.0d0*h)

      end function

c     define a segunda derivada simetrica de 5 pontos
      real*8 function fs5 (h)
      implicit real*8 (a-h,o-z)

        fs5 = (f(-2.0d0*h)-8.0d0*f(-1.0d0*h)+8.0d0*f(1.0d0*h)-f(2.0d0*h)
     1  )/(12.0d0*h)

      end function

c     define a segunda derivada simetrica de 5 pontos
      real*8 function f2s5 (h)
      implicit real*8 (a-h,o-z)

        f2s5 = ( -f(-2.0d0*h)+16.0d0*f(-1.0d0*h)-30.0d0*f(0.0d0*h )+16.
     1  0d0*f(1.0d0*h)-f(2.0d0*h))/(12.0d0*(h**2.0d0))

      end function

c     define a terceira derivada anti-simetrica de 5 pontos
      real*8 function f3as5 (h)
      implicit real*8 (a-h,o-z)

        f3as5 = (-f(-2.0d0*h)+2.0d0*f(-1.0d0*h)-2.0d0*f(1.0d0*h)+f(2.0d
     1  0*h) )/(2.0d0*(h**3.0d0) )

      end function
```

Descrição:

Primeiramente defini-se todas as variáveis reais como dupla precisão. Em seguida defini-se uma lista, chamada val, com os valores de h a serem utilizados. Após isso, aloca-se memória para um arquivo chamado "saida-5255417"onde será salva a tabela com os erros de cada derivada para cada h. Por fim, escreve-se o cabeçalho da tabela.

A seguir, inicia-se um do, de i=1 até 14, para calcular as derivadas. Salva-se o valor do vetor val(i) - que corresponde ao h da linha atual da tabela. Define os valores das derivadas (1ª,2ª e 3ª) no ponto 1/2 - demonstrados posteriormente. Por fim, escreve-se - utilizando o format na linha nomeada como 1 - as diferenças entre os métodos para calcular as derivadas e seus valores reais.

Ao fim do programa fecha-se a memória do arquivo.

Após isso, defini-se os métodos para calcular as derivadas - de acordo com o h recebido -, de modo que os mesmos chamam a função definida como f(h), que calcula f(1/2+h). Assim, se um método utiliza - por exemplo - f(1/2+2h), ele chama f(2h).

Cálculo das derivadas:

Tomemos a derivada da função abaixo

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

pela regra da multiplição, teremos

$$\frac{1}{2} e^{\frac{x}{2}} (tan(2x) + 4sec^2(2x)) \tag{2}$$

pela mesma regra, derivamos outra vez para obter

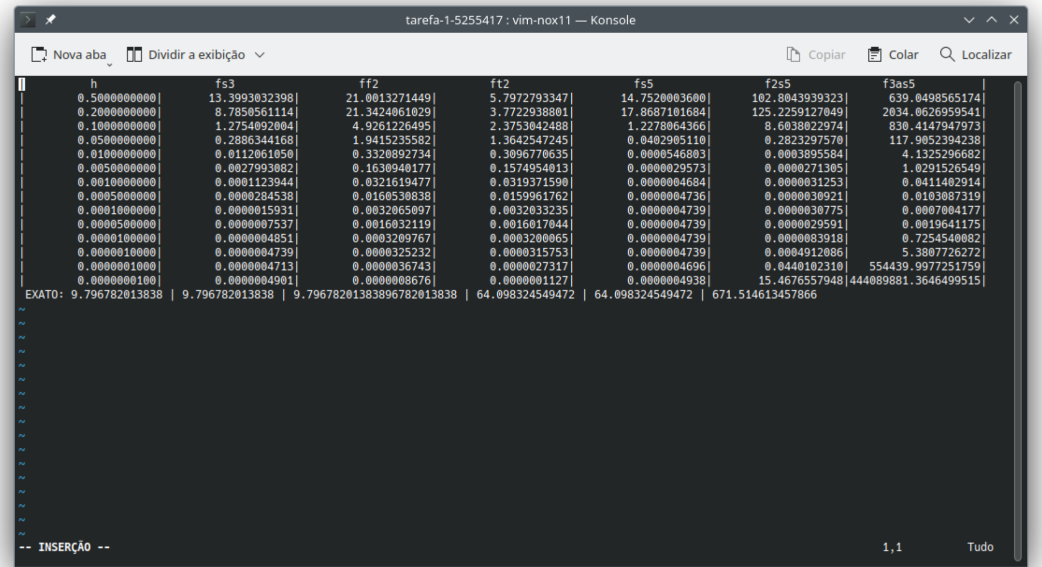
$$\frac{1}{4} e^{\frac{x}{2}} (tan(2x) + 8(4tan(2x) + 1)sec^2(2x)) \tag{3}$$

e por fim, derivando a terceira vez

$$\frac{1}{8} e^{\frac{x}{2}} (tan(2x) + 128sec^4(2x) + 4(64tan^2(2x) + 24tan(2x) + 3)sec^2(2x)) \tag{4}$$

Calculando em x=1/2, obteremos - aproximadamente - os valores inseridos no programa.

Resultados:


tarefa-1-5255417 : vim-nox11 — Konsole

h	fs3	ff2	ft2	fs5	f2s5	f3as5
0.5000000000	13.3993032398	21.0013271449	5.7972793347	14.7520003600	102.8043939323	639.0498565174
0.2000000000	8.7850561114	21.3424061029	3.7722938801	17.8687101684	125.2259127049	2034.0626959541
0.1000000000	1.2754092004	4.9261226495	2.3753042488	1.2278064366	8.6038022974	830.4147947973
0.0500000000	0.2886344168	1.9415235582	1.3642547245	0.0402905110	0.2823297570	117.9052394238
0.0100000000	0.0112061050	0.3320892734	0.3096770635	0.0000546803	0.0003895584	4.1325296682
0.0050000000	0.0027993082	0.1630940177	0.1574954013	0.0000029573	0.0000271305	1.0291526549
0.0010000000	0.0001123944	0.0321619477	0.0319371590	0.0000004684	0.0000031253	0.0411402914
0.0005000000	0.0000284538	0.0160530838	0.0159961762	0.0000004736	0.0000030921	0.0103087319
0.0001000000	0.0000015931	0.0032065097	0.0032033235	0.0000004739	0.0000030775	0.0007004177
0.0000500000	0.0000007537	0.0016032119	0.0016017044	0.0000004739	0.0000029591	0.0019641175
0.0000100000	0.0000004851	0.0003209767	0.0003200065	0.0000004739	0.0000083918	0.7254540082
0.0000010000	0.0000004739	0.0000325232	0.0000315753	0.0000004739	0.0004912086	5.3807726272
0.0000001000	0.0000004713	0.0000036743	0.0000027317	0.0000004696	0.0440102310	554439.9977251759
0.0000000100	0.0000004901	0.0000008676	0.0000001127	0.0000004938	15.4676557948	444089881.3646499515

EXATO: 9.796782013838 | 9.796782013838 | 9.79678201383896782013838 | 64.098324549472 | 64.098324549472 | 671.514613457866

-- INSERÇÃO -- 1,1 Tudo

Percebe-se que o melhor h, o que apresenta menor erro, é: fs3 - 10^{-7}

ff2 - 10^{-8}

ft2 - 10^{-8}

fs5 - 10^{-7}

f2s5 - 5.10^{-5}

f3as5 - 10^{-4}

2 Tarefa B

Tarefa: Escreva um código em FORTRAN77 que calcule

$$\int_0^1 e^{-x} cos(2\pi x)\, dx$$

usando diversos métodos, para diferentes números de divisões do inervalo 0 a 1. Estime apenas os devios em relação ao valor exato. Na última linha da tabela escreva o valor numérico exato com precisão 10^{-11} obtido pela expressão analitica.

N	$h = (b - a)/N$	Regra do Trapézio	Regra de Simpson	Regra de Boole
12	1/12			
24	1/24			
48	1/48			
96	1/96			
192	1/192			
384	1/384			
768	1/768			
1536	1/1536			
3072	1/3072			
6144	1/6144			
EXATOS	-			

Código Escrito:

```
      program integral

      implicit real*8 (a-h,o-z)
      dimension val(10)

c     define os valores de h que serao utilizados
      val(1) = 1.0d0/12.0d0
      val(2) = 1.0d0/24.0d0
      val(3) = 1.0d0/48.0d0
      val(4) = 1.0d0/96.0d0
      val(5) = 1.0d0/192.0d0
      val(6) = 1.0d0/384.0d0
```

```fortran
      val(7) = 1.0d0/768.0d0
      val(8) = 1.0d0/1536.0d0
      val(9) = 1.0d0/3072.0d0
      val(10) = 1.0d0/6144.0d0

c     abre o arquivo de saida
      open(unit=1,file='saida-5255417')

c     escreve o cabecalho da tabela
      write(1,*)'|          h               trapezio             Simpso
     1n              Boole          |'

c     loop que imprime os valores da tabela das integrais para cada h
      do i=1,10

c       define como v o valor de h para o loop atual
        v = val(i)

c       define o valor real da integral
        ri = 0.01561624581

c       define o valor inicial das integrais para cada metodo
        trap = 0.0d0
        simp = 0.0d0
        bool = 0.0d0

c       define o valor inicial de h
        h = val(i)

c       define o do pra somar os valores da integral de -h ate h para o
c       metodo do trapezio e de simpson
        do while(h.lt.1.0d0)

          trap = trap + t(h,v)
          simp = simp + s(h,v)

          h = h + 2*val(i)

        end do

c       define o valor inicial de h
        h = val(i)

c       define o do pra somar os valores da integral de -2h ate 2h para o
c       metodo de boole
        do while(h.lt.1.0d0)

          bool = bool + b(h-v,v)

          h = h + 4*val(i)

        end do

c       escreve os valores de cada integral para o valor corrente de h
        write(1,1)v,abs(trap-ri),abs(simp-ri),abs(bool-ri)

      end do

```

```
c       escreve o fim da tabela
        write(1,*) 'EXATO: -0.01561624581'

c       fecha o arquivo de saida
        close(1)

c       formata as escritas
1       format(4('|',f20.10),'|')

        end program

c       define a integral de x
        real*8 function f(x,h)
        implicit real*8 (a-h,o-z)

        pi = 4.0d0*datan(1.0d0)

        f = dexp( -(x+h) )*dcos( 2.0d0*pi*(x+h) )

        end function

c       define a regra do trapezio
        real*8 function t(x,h)
        implicit real*8 (a-h,o-z)

        t = h/2.0d0*(f(x,-h)+2.0d0*f(x,0.0d0)+f(x,h))

        end function

c       define a regra de Simpson
        real*8 function s(x,h)
        implicit real*8 (a-h,o-z)

        s = h/3.0d0*(f(x,-h)+4.0d0*f(x,0.0d0)+f(x,h))

        end function

c       define a regra de Boole
        real*8 function b(x,h)
        implicit real*8 (a-h,o-z)

        b = 2.0d0*h/45.0d0*(7*f(x,0.0d0)+32.0d0*f(x,h*1.0d0)+12.0d0*f(x,h
       1*2.0d0)+32.0d0*f(x,3.0d0*h)+7*f(x,4.0d0*h))

        end function
```

Descrição:

Primeiramente defini-se todas as variáveis reais como dupla precisão. Em seguida defini-se uma lista, chamada val, com os valores de h a serem utilizados. Após isso, aloca-se memória para um arquivo chamado "saida-5255417"onde será salva a tabela com os erros de cada derivada para cada h. Por fim, escreve-se o cabeçalho da tabela.

A seguir, inicia-se um do, de i=1 até 10, para calcular as derivadas. Salva-se o valor do vetor val(i) - que corresponde ao h da linha atual da tabela - em v. Define o valor real da integral - demonstrado posteriormente - e também define-se cada variável que corresponderá ao valor da integral obtido por cada método. Defini-se então h da mesma forma, para que some-se val(i) em h até obter 1 (integral de 0 a 1, dividida em espaços de tamanho h)

Então, inicaliza-se um loop - até que h seja igual a 1, somando os valores que as funções que definem os métodos do trapézio e de sympson além de somar 2val(i) ao h.

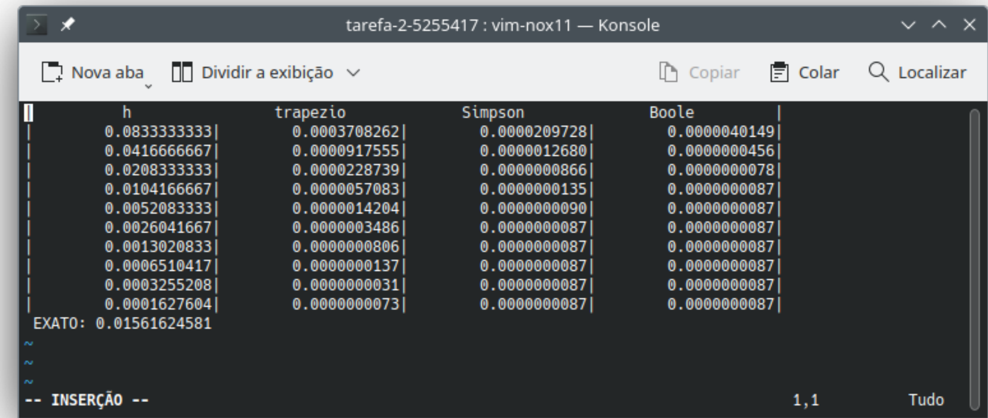
O análogo é feito para o método de boole - mas somando 4val(i) -, uma vez que o método calcula uma integral de x-2h até x+2h.

Após isso, escreve-se a diferença entre os métodos e os valores reais - utilizando o format da linha nomeada como 1.

Ao fim do programa fecha-se a memória do arquivo.

Após isso, defini-se os métodos para calcular a integral - de acordo com o x e o h recebido -, de modo que os mesmos chamam a função definida como f(x,h), que calcula f(x+h). Assim, se um método utiliza - por exemplo - f(x+2h), ele chama f(x,2h).

Resultados:



h	trapezio	Simpson	Boole
0.0833333333	0.0003708262	0.0000209728	0.0000040149
0.0416666667	0.0000917555	0.0000012680	0.0000000456
0.0208333333	0.0000228739	0.0000000866	0.0000000078
0.0104166667	0.0000057083	0.0000000135	0.0000000087
0.0052083333	0.0000014204	0.0000000090	0.0000000087
0.0026041667	0.0000003486	0.0000000087	0.0000000087
0.0013020833	0.0000000806	0.0000000087	0.0000000087
0.0006510417	0.0000000137	0.0000000087	0.0000000087
0.0003255208	0.0000000031	0.0000000087	0.0000000087
0.0001627604	0.0000000073	0.0000000087	0.0000000087

EXATO: 0.01561624581

-- INSERÇÃO --   1,1   Tudo

Nota-se que o método do trapézio é o que necessita de mais partições para obter o reultado real da integral, em seguida temos o método de Simpson como o segundo que necessita de mais partições e por fim, o método de Boole encontra a integral de maneira exata a partir de h = 0.02083 - que corresponde a 48 partições do intervalo.

Os valores obtidos como 87.10^{-}10 se devem ao fato do programa trabalhar com dupla precisão que corresponde a 8 casas decimais, deste modo - tais valores podem ser considerados como 0.

3 Tarefa C

Tarefa; Escreva um código em FORTRAN77 qye calcule as raízes positivas e nagativas de f(x) = $x^3 - 4x^2 - 59x + 126$, preenchendo a tabela a seguir. Eleja uma tolerância de 10^{-6}, Inicie sua procura em x = -10 na busca direta ou como ponto inicial nos outros métodos. Eleja, na busca direta, um espaçamento inicial de 0,1. Na última linha da tabela coloque os valores exatos.

	Procura Direta			Newton-Raphson			Método da Secante		
Iteração	r_1	r_2	r_3	r_1	r_2	r_3	r_1	r_2	r_3
0									
1									
2									
3									
4									
5									
6									
EXATOS									

Código Escrito:

```
      program raiz

      implicit real*8 (a-h,o-z)
      dimension vetbd(3,6), vetrn(3,6), vetsec(3,6), auxiliar(6)

c     zera os vetores
      do i=1,3
            do j=1,6
                  vetbd(i,j)=0.0d0
                  vetrn(i,j)=0.0d0
                  vetsec(i,j)=0.0d0
            end do
      end do

c     tolerancia para considerar a raiz exata
      tol = 10d-6

c     passo incementado na busca pela raiz
      ap = 0.1d0

c     define o valor inicial como -10
      raizbd = -10.0000005d0

c     abre o arquivo de saida
      open(unit=1,file='saida-5255417')

c     busca direta:
      do i=1,3
```

```
c             passo incementado na busca pela raiz
              ap = 0.1d0

c             inicializa o valor de raizbd para procurar a proxima raiz
              raizbd = raizbd + ap
              j = 1
              do while(j.eq.1)

c                   verifica se a raiz esta dentro do intervalo
                    aux = raizbd + ap
                    if(f(raizbd)*f(aux).le.0.0d0) then

c                         inicia o loop ate que a raiz seja menor ou igual a tolerancia
                          icount = 1
                          do while(abs(f(raizbd)).ge.tol)

                                ap = ap/2.0d0
                                aux = raizbd + ap

c                               verifica se a tabela suporta mais um valor
                                if(icount.lt.6)then
c                                     divide o intervalo pela metade e imprimi
c                                     a raiz correspondente
                                      vetbd(i,icount) = aux
                                      icount = icount + 1

c                               se a tabela nao suportar mais um valor
c                               faz a mudanca para ficar com os 6 ultimos
                                else

c                                     salva a nova ordem de valores em uma lsita auxili
                                      auxiliar(1) = vetbd(i,2)
                                      auxiliar(2) = vetbd(i,3)
                                      auxiliar(3) = vetbd(i,4)
                                      auxiliar(4) = vetbd(i,5)
                                      auxiliar(5) = vetbd(i,6)
                                      auxiliar(6) = aux

c                                     passa os valores da lista auxiliar para lista
                                      vetbd(i,1) = auxiliar(1)
                                      vetbd(i,2) = auxiliar(2)
                                      vetbd(i,3) = auxiliar(3)
                                      vetbd(i,4) = auxiliar(4)
                                      vetbd(i,5) = auxiliar(5)
                                      vetbd(i,6) = auxiliar(6)

                                end if

c                               muda o intervalo analisado ate que haja
c                               uma raiz no intervalo
                                do while(f(raizbd)*f(aux).gt.0.0d0)

                                      raizbd = raizbd + ap
                                      aux = raizbd + ap

                                end do

```

```

                  end do
                  j = 0

              end if

              raizbd = raizbd + ap

          end do

      end do

c     Newton-Raphson:
      do i=1,3

c         define o valor do passo
          ap = (i-1)*10.0d0

c         define o valor inicial
          raizrn = -10.0d0 + ap

c         define o contador de interacoes
          icount = 1

          do while(abs(f(raizrn)).ge.tol)

c             atualiza o valor
              x = rn(raizrn)
              raizrn = x
              vetrn(i,icount) = raizrn

c             incrementa o contador
              icount = icount + 1

          end do

      end do

c     Secante:
      do i=1,3

c         define o valor do passo
          ap = (i-1)*10.0d0

c         define o valor inicial
          raizs = -10.0d0 + ap

c         define o valor anterior
          aux = -10.0d0 + (ap-1.0d0)

c         define o contador de interacoes
          icount = 1

          do while(abs(f(raizs)).ge.tol)

c             passa o valor atual para x
              x = s(raizs,aux)

```

```fortran
c                   atualiza o valor antigo
                    aux = raizs

c                   atualiza o valor atual
                    raizs = x
                    vetsec(i,icount) = raizs

c                   incrementa o contador
                    icount = icount + 1

                end do

            end do

c      imprimi o cabe alho da tabela
       write(1,*)'|                Busca Direta                |
     1      Newton-Raphson                |                    Secante
     4               |'
       write(1,*)'|      r1      |      r2      |      r3      |      r1
     3     |      r2      |      r3      |      r1      |      r2      |
     4      r3      |'

c      imprimi os resultados no arquivo
       do i=1,6

c             salva qual e a interacao corrente
              a = i
              write(1,1)i,vetbd(1,i),vetbd(2,i),vetbd(3,i),vetrn(1,i),vetr
     4n(2,i),vetrn(3,i),vetsec(1,i),vetsec(2,i),vetsec(3,i)

       end do

c      imprimi os valores exatos
       write(1,*)'Valores Exatos: -7, 2, 9'

c      formata as escritas
1      format(i1,9('|',f14.8),'|')

c      fecha o arquivo de saida
       close(1)

       end program

c      define a funcao
       real*8 function f(x)
       implicit real*8 (a-h,o-z)

       f = x**3.0d0 - 4.0d0*(x**2.0d0) - 59.0d0*x + 126.0d0

       end function

c      define o metodo de newton-raphson
       real*8 function rn(x)
       implicit real*8 (a-h,o-z)

       rn = x - f(x)/( 3.0d0*(x**2.0d0) - 8.0d0*x - 59.0d0)

       end function
```

```
c       define o metodo da secante
        real*8 function s(x,xa)
        implicit real*8 (a-h,o-z)

        s = x - (f(x)*(x-xa))/(f(x)-f(xa))

        end function
```

Descrição:

Primeiramente defini-se todas as variáveis reais como dupla precisão. Em seguida, são declarados 3 vetores 3 por 6 onde armazer-se-a os valores, das raízes, obtidos e um vetor de tamanho 6, auxiliar. É feito um loop para zerar os vetores. Declara-se uma tolerância ”tol”de 10^{-6}, um passo ”ap”igual a 0,1 e o valor inicial ”raizbd”como -10.0000005 (para evitar que - caso a raizseja multipla direta do intervalo, não se obtenha a raiz na primeira iteração por ”sorte”). Por fim, aloca-se memória para um arquivo, onde será escrita a tabela, de nome ”saida-5255417”.

Inicia-se o método da busca direta:

Inicia-se um do e i=1 até 3 - onde cada interação buscará uma raiz, No começo de cada interação, (re)defini-se o passo como 0,1 e (re)defini-se a raizbd como a soma dela mesma com ap, além de iniciar j como 1.

Inicia-se um do que se repete até j ser 0, assim defini-se uma variavél auxiliar ”aux”com ”raizbd”+ ”ap”que serve para tertar se haverá mudança de sinal no próximo passo.

Enquanto não ocorre a mudança de sinal, apenas incrementa-se ”raizbd”por ”ap”.

Se houver mudança, ou seja, se a multiplicação do valor da função no ponto atual pelo próximo ponto for negativo, então inicia-se um subloop até que a função no ponto atual seja menor que a tolerância. Onde dividi-se o tamanho do passo pela metade - e salva-se (caso os vetores estejam lotados - ou seja, não caibam mais valores na tabela - utiliza-se uma lista auxiliar para salvar os valores que correspondem aos últimos 6), no vetor correspondente, o valor da função no ponto atual+”ap”(já dividido pela metade) - a cada vez que é detectada uma mudança de sinal (igual feito anteriormente, mas desta vez o passo é incrementado enquanto não há mudança, ao haver o subsubloop é encerrado). Ao sair do loop maior, muda-se j para 0.

Em seguida inicia-se o método de Newton-Raphson:

Iniciando um do de i=1 até 3, defini-se o passo ”ap”como (i-1) vezes 10, defini-se o ponto inicial ”raizrn”como -10+”ap”[deste modo, a cada interação o programa inicia 10 pontos a frente (começando do 0)] e um contador de innterações ”icount”como 1.

Em seguida, inicia-se um do que ocorre até que a função no ponto "raizrn"seja menor que a "tol", assim atualiza-se o ponto utilizando o método de newton e salva-o no vetor correspondente, além de incrementar o contador de interações (que serve para indexar o vetor de raízes).

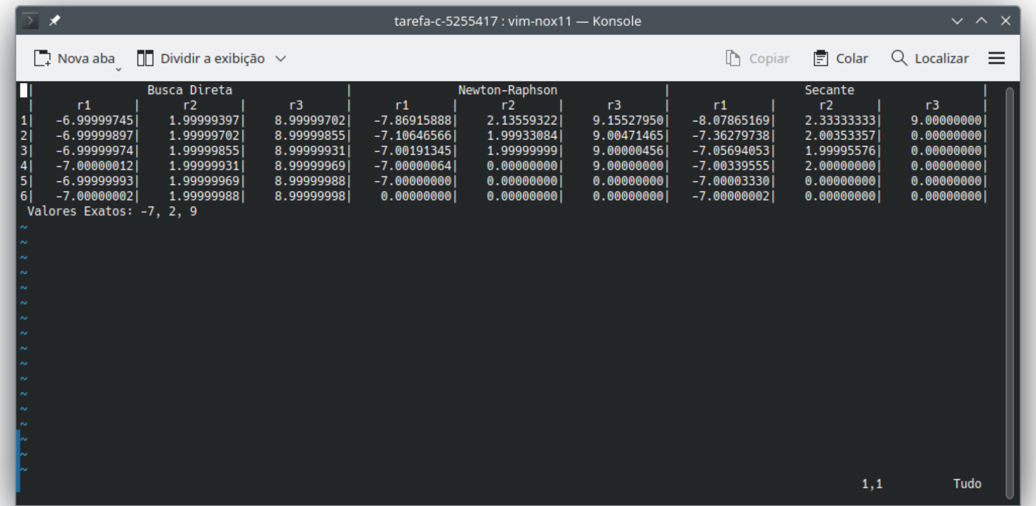
Em seguida inicia-se o método da Secante que é muito similar ao de Newton-Raphson, mas também possui uma variável auxiliar "aux"inicialmente definida da mesma forma que "raizs", a variável principal - mas subtraindo 1 - que é atualizada no decorrer do programa para equivaler ao ponto anterior ao atual. Pois o método da secante exige que se saiba qual era o ponto anterior.

Por fim, imprimi-se o cabeçalho da tabela. Após isso, imprimi-se as raízes obtidas e a interação correspondente. Por fim, imprimi-se os valores exatos das raízes.

No final, é definido numa linha chamada "1"o formato de impressão e fecha-se o arquivo.

Após o fim do programa defini-se uma f(x) que corresponde a função da qual o programa encontra as raízes. Também defini-se função rn(x) que corresponde ao método de Newton-Raphson e a função s(x,xa) que corresponde ao método da secante.

Resultados:



	Busca Direta			Newton-Raphson			Secante		
	r1	r2	r3	r1	r2	r3	r1	r2	r3
1	-6.99999745	1.99999397	8.99999702	-7.86915888	2.13559322	9.15527950	-8.07865169	2.33333333	9.00000000
2	-6.99999897	1.99999702	8.99999855	-7.10646566	1.99933084	9.00471465	-7.36279738	2.00353357	0.00000000
3	-6.99999974	1.99999855	8.99999931	-7.00191345	1.99999999	9.00000456	-7.05694053	1.99995576	0.00000000
4	-7.00000012	1.99999931	8.99999969	-7.00000064	0.00000000	9.00000000	-7.00339555	2.00000000	0.00000000
5	-6.99999993	1.99999969	8.99999988	-7.00000000	0.00000000	0.00000000	-7.00003330	0.00000000	0.00000000
6	-7.00000002	1.99999988	8.99999998	0.00000000	0.00000000	0.00000000	-7.00000002	0.00000000	0.00000000

Valores Exatos: -7, 2, 9

É notório que o método da busca direta é o menos eficiente, uma vez que suas raízes demoram mais interações para chegar ao valor real - com a precisão desejada - (ou sequer chegam na quantidade de interações utilizada).

Já o método de Newton-Raphson sempre obteve a precisão desejada para a quantidade de interações utilizada, e foi mais rápido que o da busca direta.

Por fim, o método da secante foi o que mais rapidamente obteve as raízes com a precisão desejada, necessitando de 11 interações, no total, para encontrar as 3 raízes.