7600017

Introdução à Física Computacional

Projeto 2: Cálculo Numérico

Prof: José A. Hoyos

Victor Foscarini Almeida
nUsp: 10728101

São Carlos,2019

1 - Introdução

O cálculo numérico é extremamente utilizado na física para resolver integrais e derivadas que não são possíveis ou não são triviais de serem resolvidas analiticamente. Assim, é essencial para um estudante de física entender a teoria por trás e como implementar essas ferramentas.

2 - Métodos

2.1 - Derivação numérica

A derivação numérica é um método útil para calcular numericamente a derivada de funções complexas que não sejam deriváveis ou não sejam trivialmente deriváveis. O método depende apenas de conhecer a função ou informações relacionadas ao funcionamento dela.

Neste primeiro código serão calculadas, por diversos métodos e precisões diferentes, a derivada da função f (x) = e^2x * cos(x/4). As variáveis ff1 e ff2 se referem à derivada para frente de dois pontos, ft1 e ft2 à derivada para trás de dois pontos, f3s1 e f3s2 à derivada simétrica de três pontos, f5s1 e f5s2 à derivada simétrica de cinco pontos. Nos casos citados anteriormente, as derivadas são de primeira ordem. Já nas variáveis f3s12 e f3s22 é realizada a derivada segunda simétrica de três pontos.

É variado o ‘h’ da fórmula. O ‘h’ tende a zero no limite da derivada, assim quanto mais próximo de zero o ‘h’ está no cálculo, mais precisa é a derivada, porém também é mais demorado o cálculo para o computador. Abaixo encontra-se o código completo.

```
program derivacao

    implicit none
                                                                        real*8
h1,h2,x,ii,ff1,ff2,ft1,ft2,fs1,fs2,f3s1,f3s2,f5s1,f5s2,f3s12,f3s22
    real*8,external :: fa,fb,fc
    integer i

    open(10,file='derivadas.txt')
!cria-se um loop para calcular todas as operacoes de derivacao
numerica e salvar os resultados em um arquivo


    write(10,*) "   h    ff'   ft'   f3s'  f5s' f3s2'' "

    do i=1,8

      ii = real(i,8)
        h1 = 5*10**(-ii)
        h2 = h1/5

        x = 1

        ff1 = fa(x+h1,x,h1)
        ff2 = fa(x+h2,x,h2)

        ft1 = fa(x,x-h1,h1)
        ft2 = fa(x,x-h2,h2)

        f3s1 = fa(x+h1,x-h1,2*h1)
        f3s2 = fa(x+h2,x-h2,2*h2)

        f5s1 = fb(x+2*h1,x+h1,x-h1,x-2*h1,h1)
        f5s2 = fb(x+2*h2,x+h2,x-h2,x-2*h2,h2)

        f3s12 = fc(x+h1,x,x-h1,h1)
        f3s22 = fc(x+h2,x,x-h2,h2)

        write(10,*)h1,ff1,ft1,f3s1,f5s1,f3s12
        write(10,*)h2,ff2,ft2,f3s2,f5s2,f3s22

    enddo
```

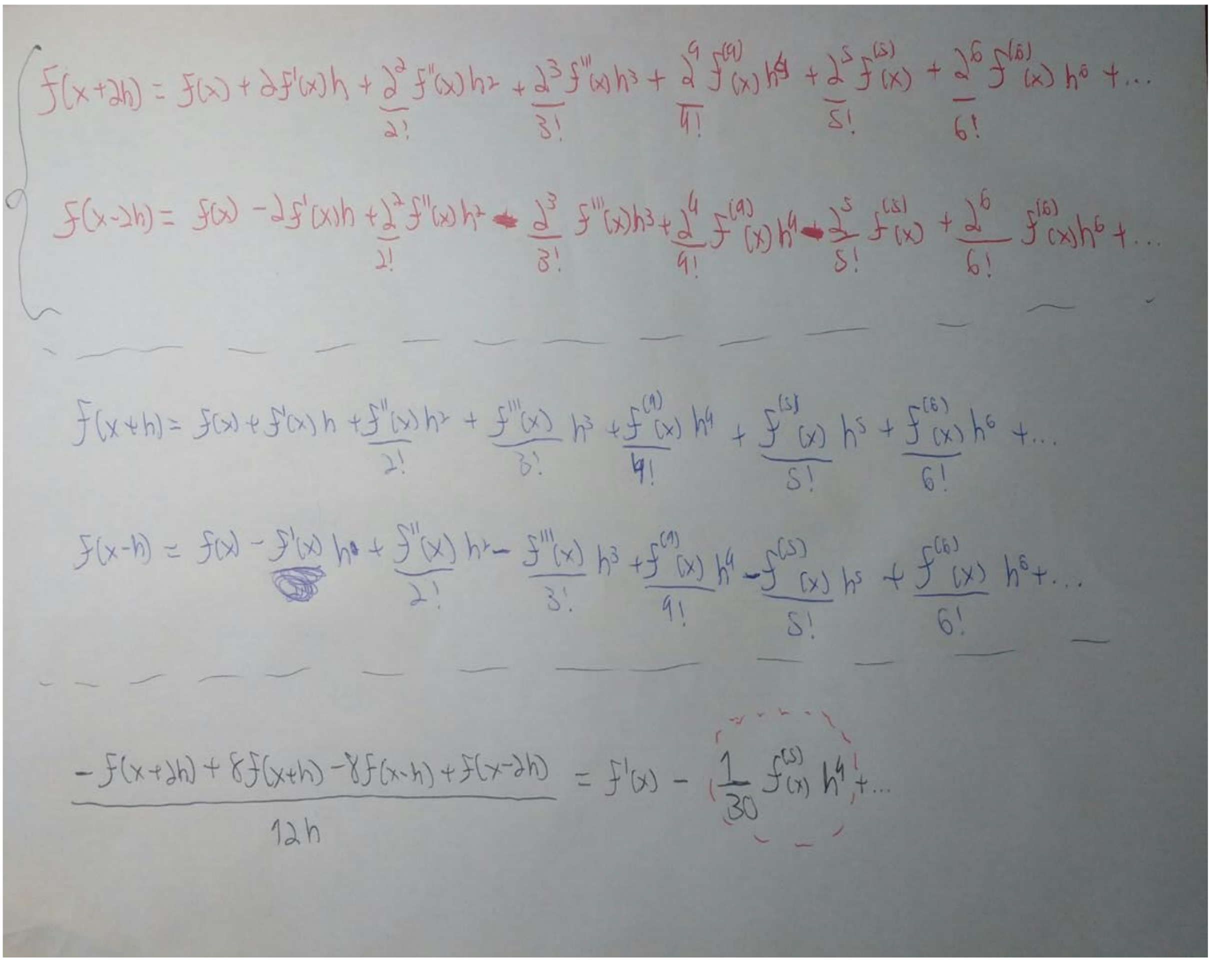
```
end program derivacao


function fa(a1,a2,h)
    implicit none
    real*8 a1,a2,h,fa
      fa = (   ( exp(2.d0*a1) * cos(a1/4.d0) ) - ( exp(2.d0*a2) *
cos(a2/4.d0) )  ) /h
    return
end function

function fb(a1,a2,a3,a4,h)
    implicit none
    real*8 a1,a2,a3,a4,h,fb

fb=(-1.d0*exp(2.d0*a1)*cos(a1/4.d0)+8.d0*exp(2.d0*a2)*cos(a2/4.d0)-8
.d0*exp(2.d0*a3)*cos(a3/4.d0)+exp(2.d0*a4)*cos(a4/4.d0))
    fb = fb/(12*h)
    return
end function

function fc(a1,a2,a3,h)
    implicit none
    real*8 a1,a2,a3,h,fc
     fc = ((exp(2.d0*a1)*cos(a1/4.d0))-2*(exp(2.d0*a2)*cos(a2/4.d0))
+ (exp(2.d0*a3)*cos(a3/4.d0)))/h**2
    return
end function
```

a) Ordem do erro cometido

Será feito o cálculo do erro para a fórmula abaixo da derivada de cinco pontos.

$$f'_{5s}(x) \equiv \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h}.$$

Para tal, toma-se cada função dessa equação, de f(x+2h) até f(x-2h) e abre-se pela série de Taylor. Então, combinando-as de acordo com essa equação, obtém-se o erro. A imagem abaixo ilustra o método.



O erro obtido é então 1/30 * f5(x) * **h^4**, ou seja, a aproximação apresenta erro de quarta ordem.

b) Comparando métodos/precisão

Aqui são realizados os cálculos das derivadas numericamente de forma semelhante à letra (a), porém foi adicionado o cálculo real das derivadas, onde ‘der1’ é de primeira ordem e ‘der2’ de segunda ordem e foi imprimido num arquivo os valores absolutos da diferença entre a derivada real e a numérica.

```
program desvios

    implicit none
    real*8
h1,h2,x,ii,ff1,ff2,ft1,ft2,fs1,fs2,f3s1,f3s2,f5s1,f5s2,f3s12,f3s22,d
er1,der2
    real*8,external :: fa,fb,fc
    integer i

    open(10,file='desvios.txt')

!aqui tem-se a primeira diferença com relaçao ao codigo anterior,
pois sera calculado o valor real da derivada(der1 de primeira ordem
e der 2 de segunda ordem)

    der1 = exp(2.d0) * (2.d0*cos(0.25d0) - 0.25d0*sin(0.25d0))
    der2 = exp(2.d0) * (3.9375d0*cos(0.25d0) - 1.d0*sin(0.25d0))

!cria-se um loop para calcular todas as operacoes de derivacao
numerica e salvar os resultados em um arquivo
!nesse loop, sao chamadas duas funçoes fa e fb, que calculam as
derivadas aproximadas

    write(10,*) "   h    ff'   ft'   f3s'  f5s' f3s'' "

    do i=1,8

     ii = real(i,8)
       h1 = 5*10**(-ii)
       h2 = h1/5
```

```
        x = 1

        ff1 = fa(x+h1,x,h1)
        ff2 = fa(x+h2,x,h2)

        ft1 = fa(x,x-h1,h1)
        ft2 = fa(x,x-h2,h2)

        f3s1 = fa(x+h1,x-h1,2*h1)
        f3s2 = fa(x+h2,x-h2,2*h2)

        f5s1 = fb(x+2*h1,x+h1,x-h1,x-2*h1,h1)
        f5s2 = fb(x+2*h2,x+h2,x-h2,x-2*h2,h2)

        f3s12 = fc(x+h1,x,x-h1,h1)
        f3s22 = fc(x+h2,x,x-h2,h2)

!Aqui ha uma mudança em relaçao ao codigo anterior, pois sera
imprimido no arquivo a diferença absoluta entre o valor real(der) da
derivada e o aproximado


        write(10,*)
h1,abs(der1-ff1),abs(der1-ft1),abs(der1-f3s1),abs(der1-f5s1),abs(der
2-f3s12)
        write(10,*)
h2,abs(der1-ff2),abs(der1-ft2),abs(der1-f3s2),abs(der1-f5s2),abs(der
2-f3s22)

    enddo


end program desvios


function fa(a1,a2,h)
    implicit none
    real*8 a1,a2,h,fa
    fa = (  ( exp(2.d0*a1) * cos(a1/4.d0) ) - ( exp(2.d0*a2) *
cos(a2/4.d0) )  ) /h
    return
end function
```

```
function fb(a1,a2,a3,a4,h)
    implicit none
    real*8 a1,a2,a3,a4,h,fb

fb=(-1.d0*exp(2.d0*a1)*cos(a1/4.d0)+8.d0*exp(2.d0*a2)*cos(a2/4.d0)-8
.d0*exp(2.d0*a3)*cos(a3/4.d0)+exp(2.d0*a4)*cos(a4/4.d0))
    fb = fb/(12*h)
    return
end function

function fc(a1,a2,a3,h)
    implicit none
    real*8 a1,a2,a3,h,fc
    fc = ((exp(2.d0*a1)*cos(a1/4.d0))-2*(exp(2.d0*a2)*cos(a2/4.d0))
+ (exp(2.d0*a3)*cos(a3/4.d0)))/h**2
    return
end function
```

c) Graficando a precisão

Para essa parte, serão computados, para diversos valores de 'h', o erro da diferenciação numérica com relação à diferenciação real. Como serão utilizados valores pequenos de 'h', o real aqui é definido como precisão 16. Novamente, aproveita-se dos códigos anteriores e já imprime-se um arquivo com os dados prontos para serem plotados.

Parte 1: Utiliza-se uma parte do código da 1(c), porém com precisão do real 16 e imprimindo num arquivo os valores já prontos para plotar. Irei analisar o erro da derivada para frente de dois pontos.

```
program grafff

    implicit none
    real*16 h,der,x,ii,ff
    real*16,external :: fa
    integer i

    open(10,file='plot1.dat')
```

```
!vale notar que utilizou-se o real de precisao 16 visto qeu alguns
valores de "h" utilizados aqui serao pequenos
!aqui eh calculado o valor real da derivada

    der = exp(2.d0) * (2.d0*cos(0.25d0) - 0.25d0*sin(0.25))

!cria-se um loop para calcular todas as operacoes de derivacao
numerica e salvar os resultados em um arquivo
!nesse loop, eh chamada fa , que calcula a derivada aproximada

    do i=0,14

          ii = real(i,16)
          ii = 15.d0 - ii
        h = 10**(-ii)

        x = 1

        ff = fa(x+h,x,h)

!Aqui ha uma mudança em relaçao ao codigo anterior
!serao impimidas duas tabelas no arquivo, uma contendo o log na base
de 10 de h e outra da diferenca entre a derivada real e aproximada
desvio
!sera feito assim para utilizar esse arquivo para plotar um grafico
no Xmgrace


        write(10,*) log10(h),log10(abs(der-ff))


    enddo


end program grafff


function fa(a1,a2,h)
    implicit none
    real*16 a1,a2,h,fa
    fa = (  ( exp(2.d0*a1) * cos(a1/4.d0) ) - ( exp(2.d0*a2) *
cos(a2/4.d0) )  ) /h
```

```
    return
end function
```

Parte 2: semelhante à parte 1, porém se referindo à derivada simétrica de três pontos.

```
program graf3s

    implicit none
    real*16 h,der,x,ii,f3s
    real*16,external :: fa
    integer i

    open(10,file='plot2.dat')

!vale notar que utilizou-se o real de precisao 16 visto qeu alguns
valores de "h" utilizados aqui serao pequenos
!aqui eh calculado o valor real da derivada

    der = exp(2.d0) * (2.d0*cos(0.25d0) - 0.25d0*sin(0.25))

!cria-se um loop para calcular todas as operacoes de derivacao
numerica e salvar os resultados em um arquivo
!nesse loop, eh chamada fa , que calcula a derivada aproximada

    do i=0,14

         ii = real(i,16)
         ii = 15.d0 - ii
        h = 10**(-ii)

        x = 1

        f3s = fa(x+h,x-h,2*h)


!Aqui ha uma mudança em relaçao ao codigo anterior
```

```
!serao impimidas duas tabelas no arquivo, uma contendo o log na base
de 10 de h e outra da diferenca entre a derivada real e aproximada
desvio
!sera feito assim para utilizar esse arquivo para plotar um grafico
no Xmgrace


        write(10,*) log10(h),log10(abs(der-f3s))


    enddo


end program graf3s


function fa(a1,a2,h)
    implicit none
    real*16 a1,a2,h,fa
    fa = (  ( exp(2.d0*a1) * cos(a1/4.d0) ) - ( exp(2.d0*a2) *
cos(a2/4.d0) )  ) /h
    return
end function
```

Parte 3: semelhante à parte 1, mas sobre a derivada simétrica de cinco pontos.

```
program graf5s

  implicit none
  real*16 h,der,x,ii,f5s
  real*16,external :: fb
  integer i

  open(10,file='plot3.dat')

!vale notar que utilizou-se o real de precisao 16 visto qeu alguns valores de "h"
utilizados aqui serao pequenos
!aqui eh calculado o valor real da derivada
```

```
   der = exp(2.d0) * (2.d0*cos(0.25d0) - 0.25d0*sin(0.25))

!cria-se um loop para calcular todas as operacoes de derivacao numerica e salvar os
resultados em um arquivo
!nesse loop, eh chamada fb , que calcula a derivada aproximada

   do i=0,14

            ii = real(i,16)
            ii = 15.d0 - ii
      h = 10**(-ii)

      x = 1

      f5s = fb(x+2*h,x+h,x-h,x-2*h,h)

!Aqui ha uma mudança em relaçao ao codigo anterior
!serao impimidas duas tabelas no arquivo, uma contendo o log na base de 10 de h e
outra da diferenca entre a derivada real e aproximada desvio
!sera feito assim para utilizar esse arquivo para plotar um grafico no Xmgrace


      write(10,*) log10(h),log10(abs(der-f5s))


   enddo


end program graf5s


function fb(a1,a2,a3,a4,h)
   implicit none
   real*16 a1,a2,a3,a4,h,fb

fb=(-1.d0*exp(2.d0*a1)*cos(a1/4.d0)+8.d0*exp(2.d0*a2)*cos(a2/4.d0)-8.d0*exp(2.d0*a3)*
cos(a3/4.d0)+exp(2.d0*a4)*cos(a4/4.d0))
   fb = fb/(12*h)
   return
end function
```

2.2 - Integração Numérica

A operação de integração numérica é tão ou mais importante que a operação de derivação numérica.

a) Erro do método de Boole

Para o método de Boole, temos a seguinte equação:

$$I_B = \frac{2h}{45}\left(7f(x_{i-2}) + 32f(x_{i-1}) + 12f(x_i) + 32f(x_{i+1}) + 7f(x_{i+2})\right)$$

Desenvolve-se de forma semelhante à 1a), mas dessa vez aplicando a equação de Lagrange ao invés de Taylor e obtém-se seguinte erro:

$$-\frac{8}{945}h^7 f^{(6)}(c)$$

Percebe-se então que a ordem do erro da aproximação de Boole é **h^7** e o erro é de sétima ordem

b) Comparando métodos/precisão

Para a integração numérica, cria-se um programa para imprimir a diferença entre a integral real e a integral numérica usando os métodos do Trapézio, de Simpsons e de Boole. Ele chamará uma função diferente para cada método que calcula as integrais para um ‘h’ específico.

Assim como na diferenciação numérica, na integral real, o ‘h’ tende a zero e, portanto, quanto menor o ‘h’ mais precisa será a computação, porém mais tempo levará também.

```
program integracao
    implicit none

!sao definidos os reais necessarios para o calculo
!note que intS,fs se referem a Simpson
!intF,ft se referem a regra do Trapezio
```

```
!intB,fb se referem a lei de boole

    real*8 h,ii,intT,intS,intB,int,pi
    real*8,external :: ft,fs,fb
    integer i

    open(10,file='integrais.txt')
!vale notar que aqui utilizou-se o h começando em 0.25 pois esse eh
o menor valor possivel para a aproximacao de Boole
!calcula-se abaxio na variavel 'int' o valor real da integral para
comparacao mais tarde

    pi = 4*atan(1.d0)
    int = -0.2d0*exp(2.d0*1.d0)*(cos(1.d0)-2.d0*sin(1.d0)) + 0.2d0

    do i=2,13

        ii = real(i,8)
        h = 2**(-ii)

        intT = ft(h)
        intS = fs(h)
        intB = fb(h)

        write(10,*)h,abs(int-intT),abs(int-intS),abs(int-intB)
    enddo

end program integracao

function ft(h)
    implicit none
    real*8 x,ft,h
    integer i

    x = h
    ft = 0

    do i=1,int(1.d0/h)-1 !aqui o ultimo deve ficar fora do loop, por
isso o -1
        ft = ft + exp(2*x)*sin(x)
        x = x + h
    enddo
```

```
    ft = ft + exp(2*x)*sin(x)/2
    ft = ft*h

    return
end function

function fs(h)
    implicit none
    real*8 x,fs,h
    integer i

    x = h
    fs = 0

    do i=1,int(1.d0/(2*h)-1)
        fs = fs + 4*exp(2*x)*sin(x)
        x = x+h
        fs = fs + 2*exp(2*x)*sin(x)
        x = x+h
    enddo

    if (x >= 1.d0 - 1.d0*h) then
        fs = fs + 4*exp(2*x)*sin(x)
        x = x+h
    endif

    fs = fs + exp(2*x)*sin(x)
    fs = fs*h/3

    return
end function

function fb(h)
    implicit none
    real*8 x,fb,h
    integer i

    x=0
    fb=0

    do i=1,int(1.0d0/h)
        x = x+h
           if(mod(i,4)==0) then
```

```
                fb = fb + 14.0d0*exp(2*x)*sin(x)
           else if(mod(i,4)==1) then
                fb = fb + 32.0d0*exp(2*x)*sin(x)
           else if(mod(i,4)==2) then
                fb = fb + 12.0d0*exp(2*x)*sin(x)
           else if(mod(i,4)==3) then
                fb = fb + 32.0d0*exp(2*x)*sin(x)
           end if
      end do

    fb = 2.d0*h/45.d0 * ( fb - 7.d0*(exp(2*x)*sin(x) +
exp(2*h)*sin(h)) )

    return
end function
```

c) Graficando a precisão

Para este problema, será aproveitado o código da questão anterior, imprimindo os valores já prontos para serem plotados em gráfico logxlog de ‘h’ pelo erro.

Parte 1: Trapézio

```
program grafT
    implicit none

!sao definidos os reais necessarios para o calculo
!intF,ft se referem a regra do Trapezio


    real*8 h,ii,intT,int,pi
    real*8,external :: ft
    integer i

    open(10,file='trapezio.txt')
!vale notar que aqui utilizou-se o h começando em 0.25 pois esse eh
o menor valor possivel para a aproximacao de Boole
```

```
!calcula-se abaxio na variavel 'int' o valor real da integral para
comparacao mais tarde
!aqui utilizou-se precisao 16 pois foram utilizados valores de h
muito pequenos

    pi = 4*atan(1.d0)
    int = -0.2d0*exp(2.d0*1.d0)*(cos(1.d0)-2.d0*sin(1.d0)) + 0.2d0

    do i=1,10

        ii = real(i,8)
        h = 10**(-ii)

        intT = ft(h)

        write(10,*)log10(h),log10(abs(int-intT))
    enddo

end program grafT

function ft(h)
    implicit none
    real*8 x,ft,h
    integer i

    x = h
    ft = 0

    do i=1,int(1.d0/h)-1 !aqui o ultimo deve ficar fora do loop, por
isso o -1
        ft = ft + exp(2*x)*sin(x)
        x = x + h
    enddo

    ft = ft + exp(2*x)*sin(x)/2
    ft = ft*h

    return
end function
```

Parte 2: Simpsons

```fortran
program grafS
    implicit none

!sao definidos os reais necessarios para o calculo
!note que intS,fs se referem a Simpson


    real*8 h,ii,intS,int,pi
    real*8,external :: fs
    integer i

    open(10,file='simpson.txt')
!vale notar que aqui utilizou-se o h começando em 0.25 pois esse eh
o menor valor possivel para a aproximacao de Boole
!calcula-se abaxio na variavel 'int' o valor real da integral para
comparacao mais tarde
!aqui utilizou-se precisao 16 pois foram uitilizados valores de h
muito pequenos

    pi = 4*atan(1.d0)
    int = -0.2d0*exp(2.d0*1.d0)*(cos(1.d0)-2.d0*sin(1.d0)) + 0.2d0

    do i=1,10

        ii = real(i,8)
        h = 10**(-ii)

        intS = fs(h)

        write(10,*)log10(h),log10(abs(int-intS))
    enddo

end program grafS


function fs(h)
    implicit none
    real*8 x,fs,h
    integer i

    x = h
    fs = 0
```

```
    do i=1,int(1.d0/(2*h)-1)
        fs = fs + 4*exp(2*x)*sin(x)
        x = x+h
        fs = fs + 2*exp(2*x)*sin(x)
        x = x+h
    enddo

    if (x >= 1.d0 - 1.d0*h) then
        fs = fs + 4*exp(2*x)*sin(x)
        x = x+h
    endif

    fs = fs + exp(2*x)*sin(x)
    fs = fs*h/3

    return
end function
```

Parte 3: Boole

```
program grafB
    implicit none

!sao definidos os reais necessarios para o calculo
!intB,fb se referem a lei de Boole

    real*8 h,ii,intB,int,pi
    real*8,external :: fb
    integer i

    open(10,file='boole.txt')
!vale notar que aqui utilizou-se o h começando em 0.25 pois esse eh
o menor valor possivel para a aproximacao de Boole
!calcula-se abaxio na variavel 'int' o valor real da integral para
comparacao mais tarde
!aqui utilizou-se precisao 16 pois foram uitilizados valores de h
muito pequenos
```

```
    pi = 4*atan(1.d0)
    int = -0.2d0*exp(2.d0*1.d0)*(cos(1.d0)-2.d0*sin(1.d0)) + 0.2d0

    do i=1,10

        ii = real(i,8)
        h = 10**(-ii)

        intB = fb(h)

        write(10,*)log(h),log(abs(int-intB))
    enddo

end program grafB

function fb(h)
    implicit none
    real*8 x,fb,h
    integer i

    x=0
    fb=0

    do i=1,int(1.0d0/h)
        x = x+h
           if(mod(i,4)==0) then
                 fb = fb + 14.0d0*exp(2*x)*sin(x)
           else if(mod(i,4)==1) then
                 fb = fb + 32.0d0*exp(2*x)*sin(x)
                 else if(mod(i,4)==2) then
                 fb = fb + 12.0d0*exp(2*x)*sin(x)
           else if(mod(i,4)==3) then
                 fb = fb + 32.0d0*exp(2*x)*sin(x)
           end if
     end do

    fb = 2.d0*h/45.d0 * ( fb - 7.d0*(exp(2*x)*sin(x) +
exp(2*h)*sin(h)) )

    return
end function
```

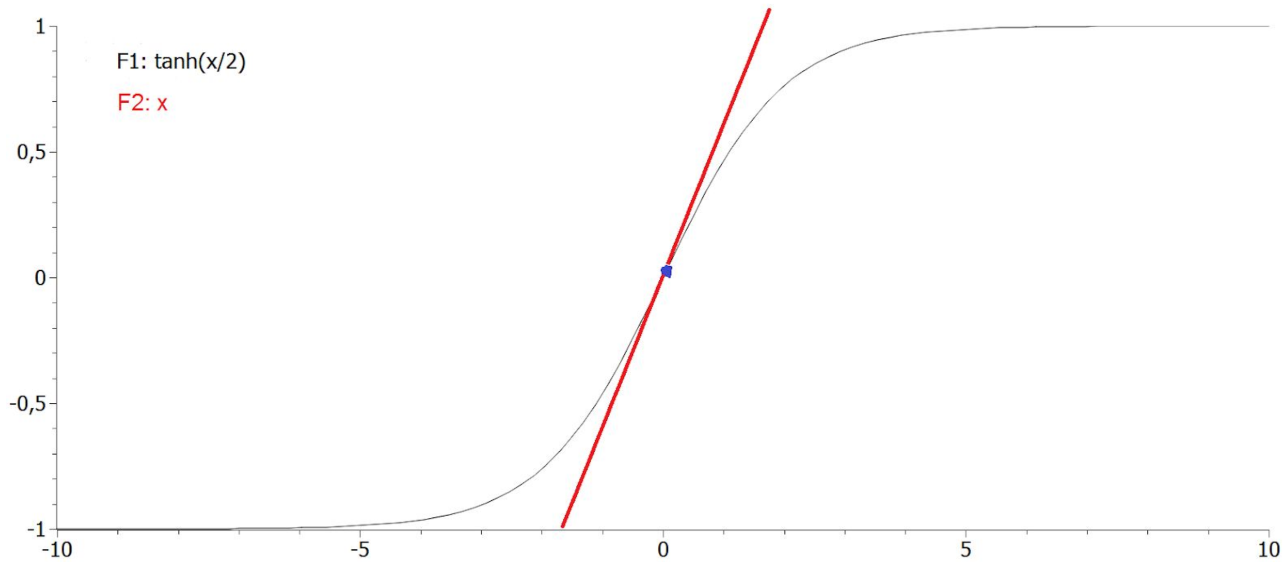
2.3 - Equações algébricas não lineares

Equações algébricas nem sempre tem uma solução fácil obtida algebricamente. Em alguns casos, utiliza-se do método de Newton-Raphson para obter numericamente, com apenas a função e sua derivada, os pontos em que essa função é zero(raízes).

a) Análise gráfica

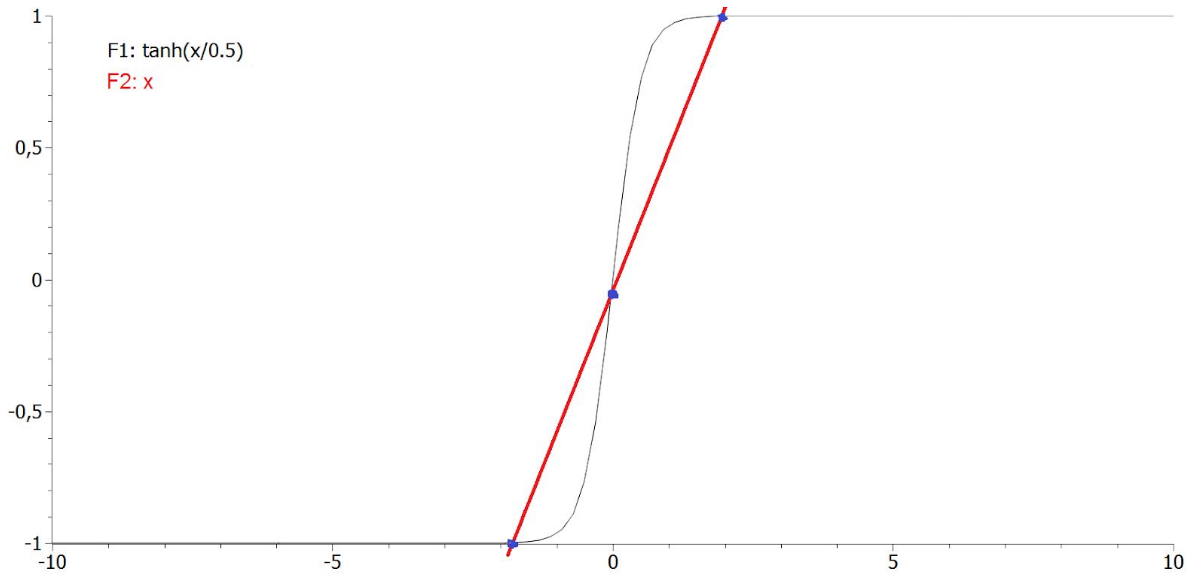
Aqui irei comparar os gráficos de F1=tanh(x/M) com o gráfico de F2=x utilizando valores diferentes de M para localizar o número de raízes existentes.

Primeiro, tomando um M>1:



A partir desse gráfico, é possível notar que a função y=x só se encontra com a função y = tanh(x/2) em um ponto(x=0). Assim, m = tanh(m/T) só tem uma solução para T>1.

Segundo, tomando um M<1:



Já por este outro gráfico, nota-se que a função y=x se encontra com a função y = tanh(x/2) em três pontos. Vendo separadamente, é difícil notar essa diferença, porém com um gráfico próximo ao outro, é evidente que um tem uma curva mais acentuada que o outro. Assim, a função tem três soluções para T<1.

c) Aplicando o método

Aqui, utiliza-se um loop válido enquanto a variação for maior que o erro estipulado para calcular a raiz da reta tangente e depois aplicar essa raiz na fórmula até que chega-se na raiz da função.

Dado o gráfico dessa função, utilizou-se o valor 10 como 'm' inicial, pois se fosse um valor negativo ou próximo de zero, achariam-se outras raízes da função. Assim, é precisa chutar um valor que pareça estar mais próximo da raiz positiva do que do ponto zero ou da raiz negativa.

```
program raizes

real*8 m,T,derivada,variacao,m0

T = 0.5d0
m = 10.d0
```

```
variacao = 1.d0

write(*,*)'m0,variacao,mf'

!cria-se um loop valido enquanto o erro for maior ou igual ao
estipulado
!vale notar que a variacao eh a funcao no ponto dividida pela
derivada

do while (abs(variacao) >= 0.000001)
     derivada = 1 - 1/((cosh(m/T)**2)*T)
     variacao = - (m-tanh(m/T))/derivada
     m0 = m
     m = m + variacao
     write(*,*)m0,variacao,m
enddo

write(*,*) m

end program raizes
```

d)

Já para essa questão, utiliza-se um T igual a 0.9 ao invés de 1 e monta-se um gráfico dos erros com respeito ao número de iterações. Imprime-se num arquivo os valores dos logaritmos.

```
program raizes9
implicit none

    real*8 m,T,derivada,variacao,n,precisao,ii
    integer i

    T = 0.3d0

    open(10,file = 'newton1.dat')

!cria-se um loop para o erro variar
!cria-se um loop valido enquanto o erro for maior ou igual ao
estipulado
```

```
!vale notar que a variacao eh a funcao no ponto dividida pela
derivada

    do i=1,14

        ii = real(i,8)
        precisao = 10 ** (-ii)

        open(10,file = 'newton9.dat')

        n = 0
        m = 10.d0
        variacao = 1.d0

        do while (abs(variacao) >= precisao)
            n = n+1
            derivada = 1 - 1/((cosh(m/T)**2)*T)
            variacao = - (m-tanh(m/T))/derivada
            m = m + variacao
        enddo



        write(10,*) log10(n),log10(precisao)
    enddo

end program raizes9
```

e)

Troca-se o T para 0.9 e novamente imprime-se os valores num arquivo e utiliza-se esse arquivo para montar um gráfico.

```
program raizes3
implicit none

    real*8 m,T,derivada,variacao,n,precisao,ii
    integer i

    T = 0.3d0
```

```
    open(10,file = 'newton1.dat')

!cria-se um loop para o erro variar
!cria-se um loop valido enquanto o erro for maior ou igual ao
estipulado
!vale notar que a variacao eh a funcao no ponto dividida pela
derivada

    do i=1,14

        ii = real(i,8)
        precisao = 10 ** (-ii)

        open(10,file = 'newton3.dat')

        n = 0
        m = 10.d0
        variacao = 1.d0

        do while (abs(variacao) >= precisao)
            n = n+1
            derivada = 1 - 1/((cosh(m/T)**2)*T)
            variacao = - (m-tanh(m/T))/derivada
            m = m + variacao
        enddo



        write(10,*) log10(n),log10(precisao)
    enddo

end program raizes3
```

3-Resultados

3.1 - Derivação Numérica

Para as diversas diferenciações numéricas, obtém-se a seguinte tabela:

h	ff'	ft'	f3s'	f5s'	f3s2''
0.50000000000000000	23.060793916513266	8.9245506998035324	15.992672308158399	13.504499014117590	28.272486433419466
0.10000000000000001	15.265521618196942	12.621875408163907	13.943698513180424	13.861147113923209	26.436462100330345
5.0000000000000003E-002	14.541669190217927	13.222644667217427	13.882156928717677	13.861643067230091	26.380490460010005
1.0000000000000000E-002	13.994307950016793	13.730681976841730	13.862494963429262	13.861675991712012	26.362597317506342
5.0000000000000001E-003	13.927785867450382	13.795975675972372	13.861880771711377	13.861676041138793	26.362038295602019
1.0000000000000000E-003	13.874865163196581	13.848503303787396	13.861684233491989	13.861676044426563	26.361859409185229
5.0000000000000001E-004	13.868268555151886	13.855087628241947	13.861678091696916	13.861676044431745	26.361853819878434
1.0000000000000000E-004	13.862994218927227	13.860358033728204	13.861676126327716	13.861676044439145	26.361851990230889
5.0000000000000002E-005	13.862335111234358	13.861017018612642	13.861676064923500	13.861676044459868	26.361852434320099
1.0000000000000001E-005	13.861807854542716	13.861544235904686	13.861676045223701	13.861676044387334	26.361863803003867
4.9999999999999996E-006	13.861741949483532	13.861610140253335	13.861676044868434	13.861676044735207	26.361846039435484
9.9999999999999995E-007	13.861689224015095	13.861662863767776	13.861676043891435	13.861676043595375	26.360247318280017
4.9999999999999998E-007	13.861682637283934	13.861669451387115	13.861676044335525	13.861676045075672	26.371793637736118
9.9999999999999995E-008	13.861677370385905	13.861674714732430	13.861676042559168	13.861676042559168	26.556534749033748
4.9999999999999998E-008	13.861676677606738	13.861675380866245	13.861676029236492	13.861676018874411	25.934809855243660
1.0000000000000000E-008	13.861676162463255	13.861675984827571	13.861676073645413	13.861676081046898	17.763568394002501

b) Comparando métodos/precisão

Agora relacionando ‘h’ com o erro da aproximação, obtem-se a seguinte tabela:

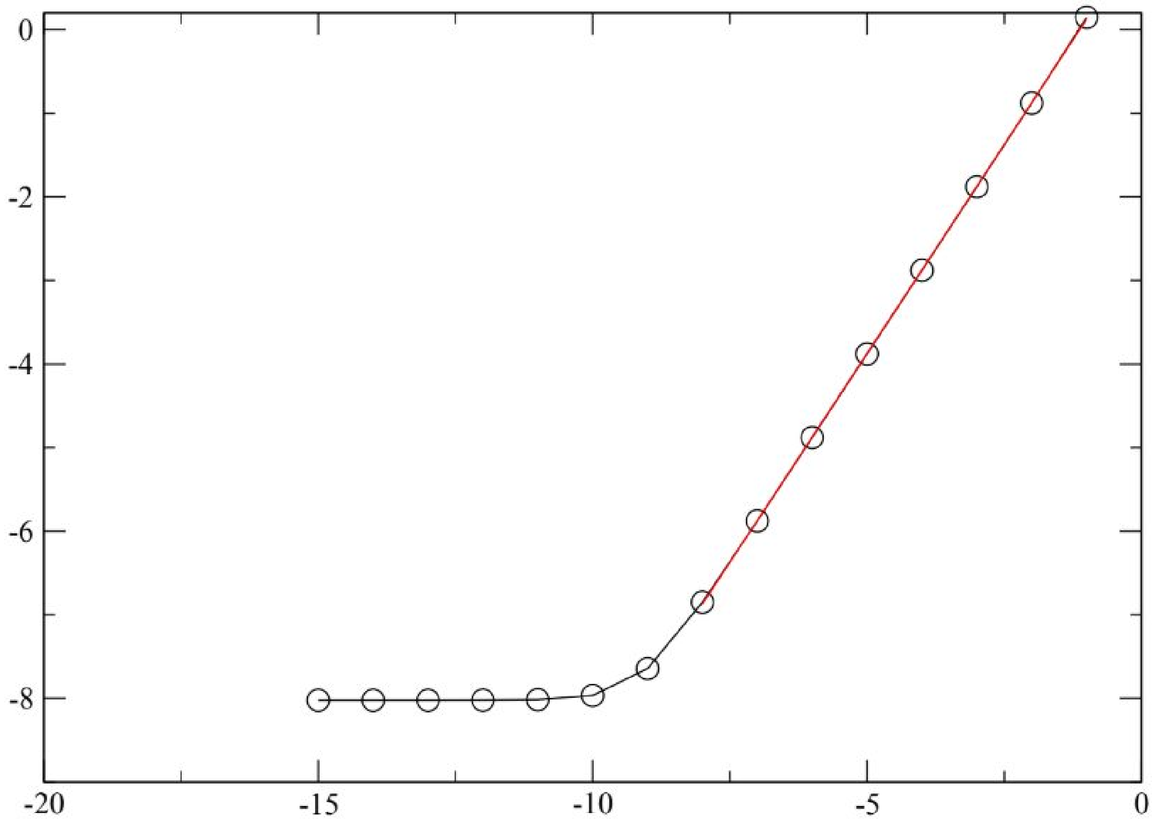
h	ff'	ft'	f3s'	f5s'	f3s''
0.50000000000000000	9.1991178720791602	4.9371253446305730	2.1309962637242936	0.35717703031651560	1.9106344765015351
0.10000000000000001	1.4038455737628368	1.2398006362701981	8.2022468746318467E-002	5.2893051089597520E-004	7.4610143412414232E-002
5.0000000000000003E-002	0.67999314578382197	0.63903137721667846	2.0480884283571754E-002	3.2977204014628114E-005	1.8638503092073933E-002
1.0000000000000000E-002	0.13263190558268789	0.13099406759237553	8.1891899515618150E-004	5.2722093357715494E-008	7.4536058841090380E-004
5.0000000000000001E-003	6.6109823016276437E-002	6.5700368461733660E-002	2.0472727727138817E-004	3.2953124673440470E-009	1.8633868408812759E-004
1.0000000000000000E-003	1.3189118762475971E-002	1.3172740646709258E-002	8.1890578833565542E-006	7.5424111400934635E-012	7.4522672974808302E-006
5.0000000000000001E-004	6.5925107177804421E-003	6.5884161921587747E-003	2.0472628108336721E-006	2.3607782395629329E-012	1.8629605023079421E-006
1.0000000000000000E-004	1.3181744931216599E-003	1.3180107059014290E-003	8.1893610115457705E-008	5.0395243533785106E-012	3.3312957725684100E-008
5.0000000000000002E-005	6.5906680025307196E-004	6.5902582146293298E-004	2.0489395069489547E-008	2.5762503241821832E-011	4.7740216757574672E-007
1.0000000000000001E-005	1.3181010861096354E-004	1.3180852941907517E-004	7.8959594418392953E-010	4.6771475581408595E-011	1.1846085936184636E-005
4.9999999999999996E-006	6.5905049426717710E-005	6.5904180770459675E-005	4.3432812901755824E-010	3.0110136606253946E-010	5.9174824471597276E-006
9.9999999999999995E-007	1.3179580989230999E-005	1.3180666329049018E-005	5.4266990900941892E-010	8.3872997436174046E-010	1.6046386379144906E-003
4.9999999999999998E-007	6.5928498287348702E-006	6.5930469901331890E-006	9.8580699159356300E-011	6.4156679968618846E-010	9.9416808181871374E-003
9.9999999999999995E-008	1.3259517999131276E-006	1.3297016749902468E-006	1.8749375385596068E-009	1.8749375385596068E-009	0.19468279211581674
4.9999999999999998E-008	6.3317263254702993E-007	6.6356786021515290E-007	1.5197613834061485E-008	2.5559694805110666E-008	0.42704210167427092
1.0000000000000000E-008	1.1802914912095730E-007	5.9606534819067747E-008	2.9211307150944776E-008	3.6612792797541260E-008	8.5982835629154302

É possível notar que nas derivadas de dois pontos, o menor ‘h’, 10^-8, foi também o mais preciso. Já nas derivadas de três e cinco pontos e a derivada de segunda ordem foi diferente, visto que o ‘h’ muito pequeno ficou fora da precisão do Fortran e houve uma perda de precisão dessa forma. Foi marcado em vermelho os valores ideais de ‘h’ para casa derivada.

c) Graficando a precisão

Vale notar que foram usados apenas alguns valores, visto que após um ‘h’ muito pequeno, atinge-se o limite da precisão da variável declarada no Fortran e portanto, perde-se a precisão.

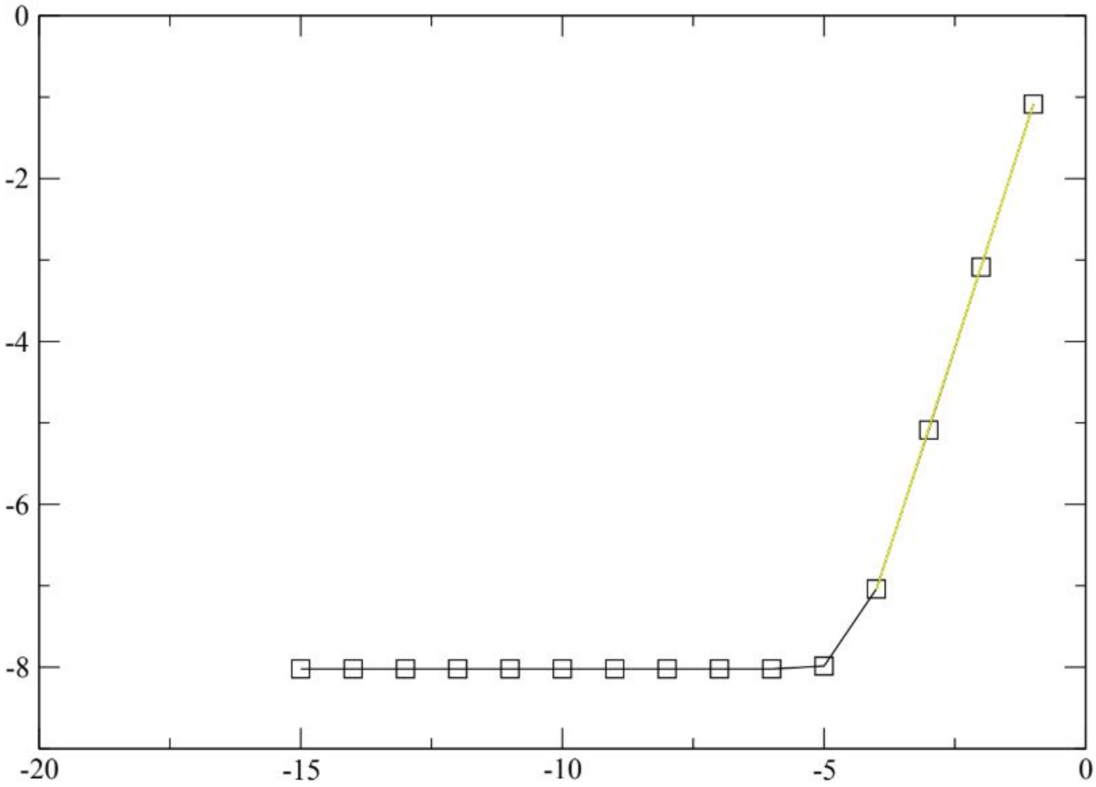
f’f



Coeficiente angular : 0.98463

O erro esperado seria de ordem 1, portanto foi obtido um resultado dentro do esperado teoricamente.

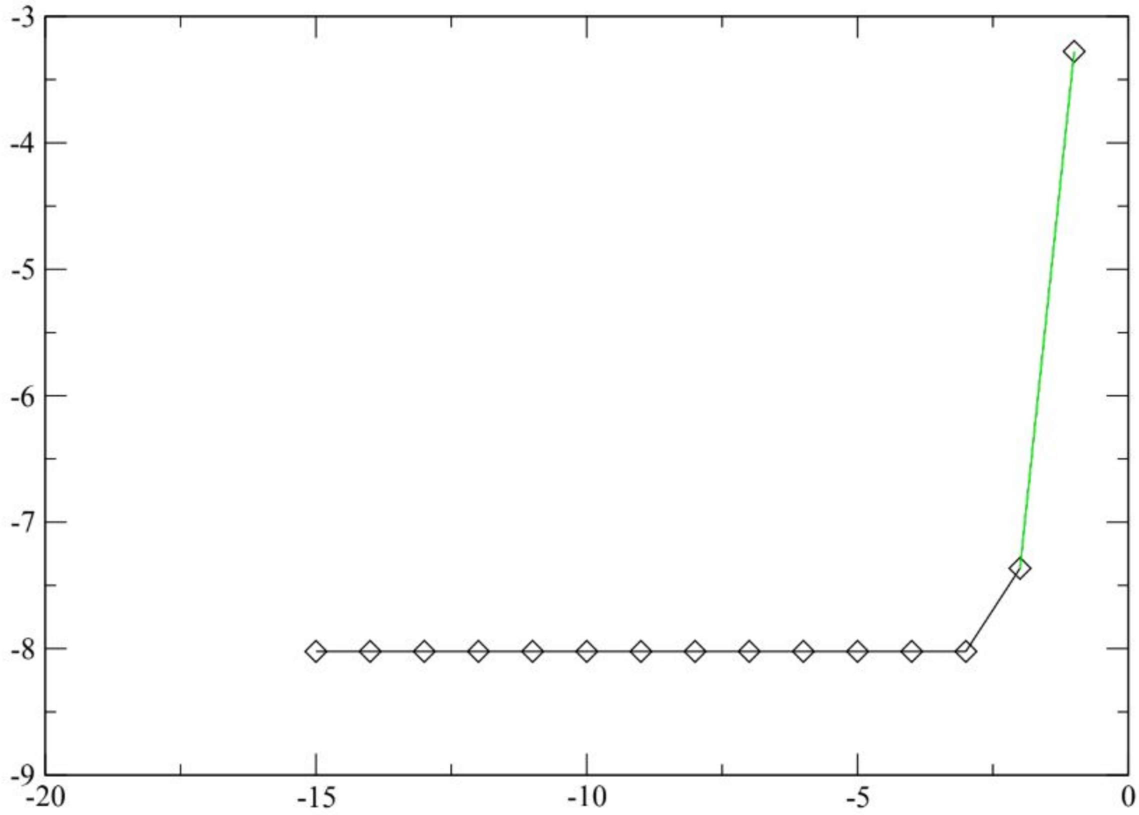
f’3s



Coeficiente angular: 1.9859

O erro esperado era de ordem 2, assim foi obtido um resultado dentro do esperado teoricamente.

f’5s



Coeficiente angular: 4.0877

O erro esperado era de ordem 4, assim foi obtido um erro equivalente ao esperado teoricamente.

3.2 - Integração Numérica

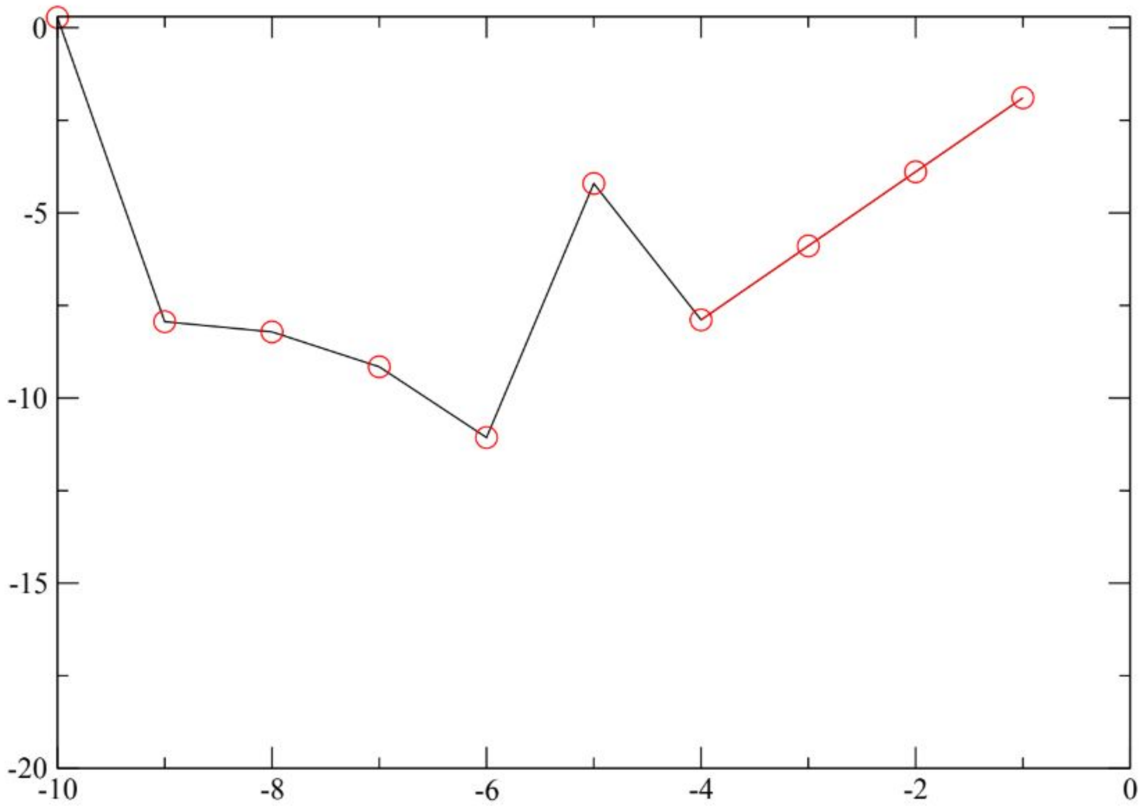
b) Comparando métodos/precisão

0.25000000000000000	8.0105543907243337E-002	1.0006493174723463E-003	3.1757064395927737E-002
0.12500000000000000	2.0072728649336957E-002	6.1790230034830529E-005	6.2263486280045299E-003
6.2500000000000000E-002	5.0210688322940289E-003	3.8488932796454378E-006	1.3762076284731251E-003
3.1250000000000000E-002	1.2554474688142303E-003	2.4034765400138269E-007	3.2336170073077319E-004
1.5625000000000000E-002	3.1387313104525028E-004	1.5018455812310094E-008	7.8362742031323762E-005
7.8125000000000000E-003	7.8468986712820765E-005	9.3860230698794567E-010	1.9287547882607115E-005
3.9062500000000000E-003	1.9617290675011390E-005	5.8661742130539096E-011	4.7843992383445055E-006
1.9531250000000000E-003	4.9043254182201679E-006	3.6664005165221170E-012	1.1914389301814765E-006
9.7656250000000000E-004	1.2260815274167669E-006	2.3026025530725747E-013	2.9727868322559914E-007
4.8828125000000000E-004	3.0652039151313204E-007	1.3988810110276972E-014	7.4247139991712174E-008
2.4414062500000000E-004	7.6630099377084093E-008	1.5543122344752192E-015	1.8552724467824078E-008
1.2207031250000000E-004	1.9157517794354817E-008	6.2172489379008766E-015	4.6370505213388924E-009

c) Graficando a precisão

Vale notar que foram usados apenas alguns valores, visto que após um ‘h’ muito pequeno, atinge-se o limite da precisão da variável declarada no Fortran e portanto, perde-se a precisão. No caso da integração, isso explica o por quê de alguns valores muito estranhos nos gráficos.

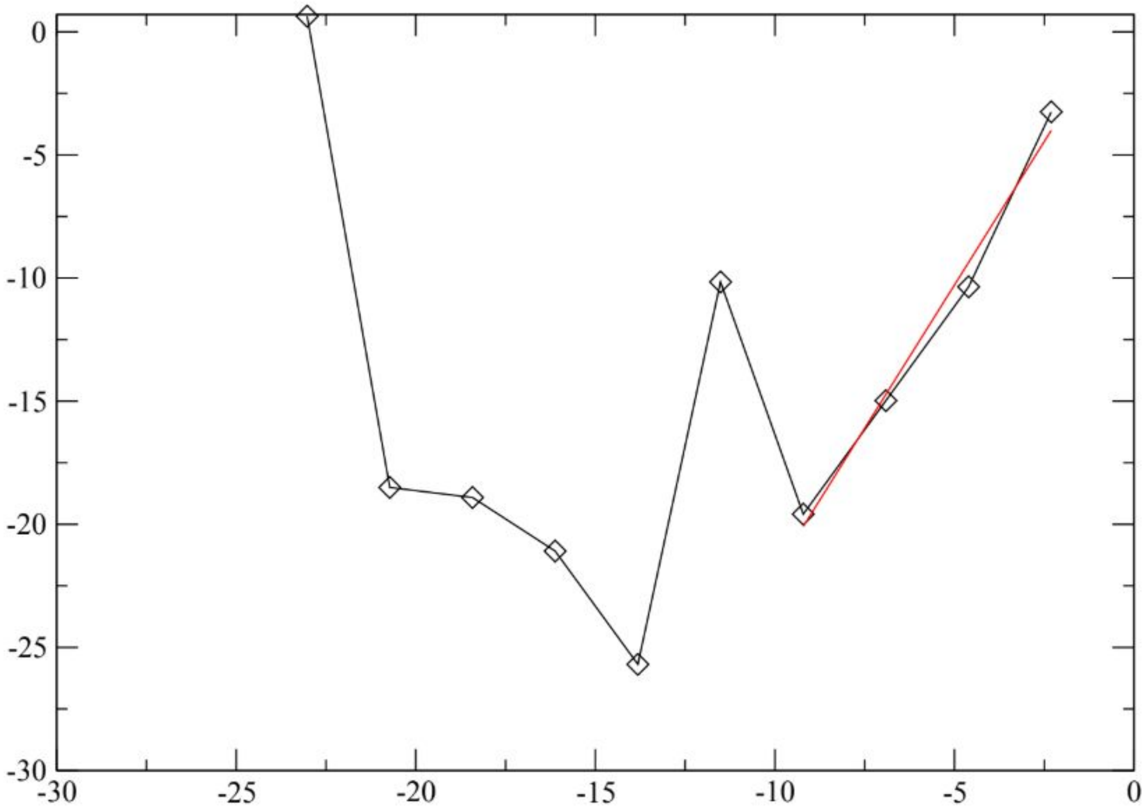
Método dos Trapézios



Coeficiente angular : 1.9999

O erro esperado era 2, portanto foi obtido um erro dentro do esperado teoricamente.

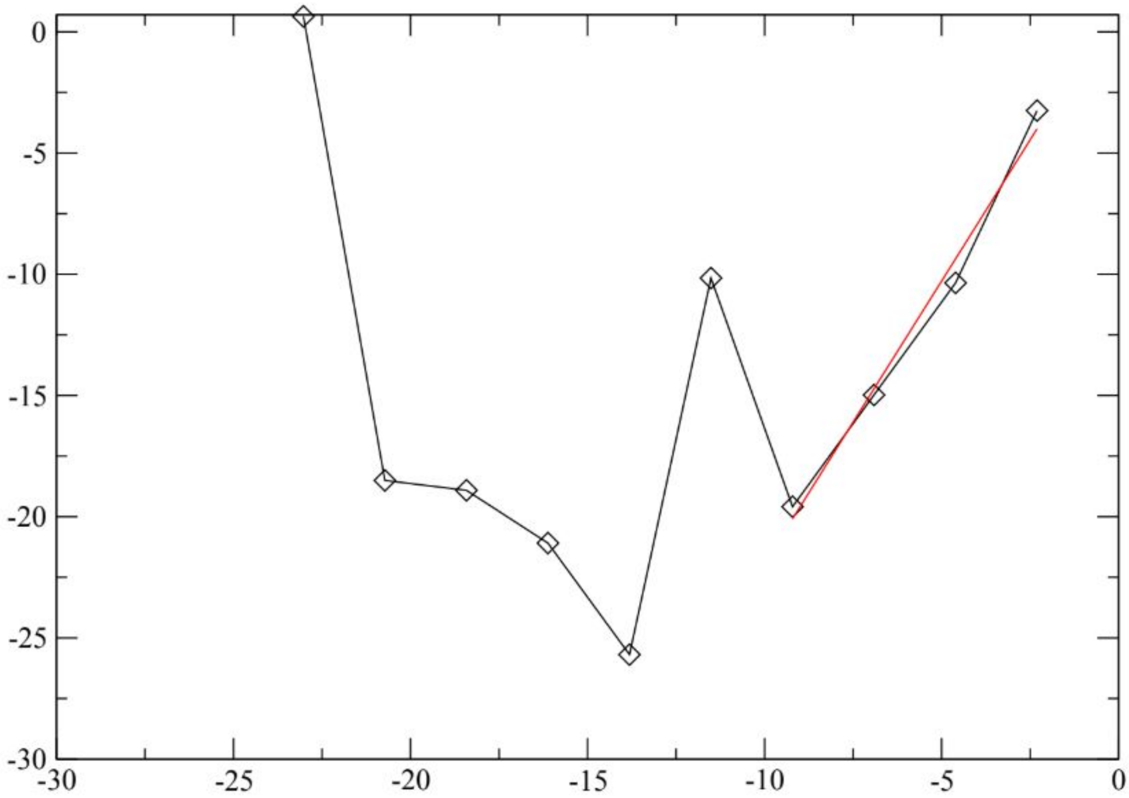
Método de Simpsons



Coeficiente angular: 6.2144

O esperado teoricamente era um erro da ordem 5, portanto foi obtido um erro menor do que o esperado teoricamente e o método de Simpsons obteve um sucesso maior do que o previsto.

Método de Boole



Coeficiente angular: 2.329

O esperado teoricamente era um erro da ordem 7, porém o método de Boole se mostrou ineficiente ao ser aplicado a esta função, talvez devido a uma perda de precisão da variável do Fortran.

3.3 - Equações algébricas não lineares

c) Aplicando o método

d) M=0.9

Obtém-se os seguintes valores:

0.60205999132796240	-1.0000000000000000
0.69897000433601886	-2.0000000000000000
0.77815125038364363	-3.0000000000000000
0.77815125038364363	-4.0000000000000000
0.84509804001425681	-5.0000000000000000
0.84509804001425681	-6.0000000000000000
0.84509804001425681	-7.0000000000000000
0.84509804001425681	-8.0000000000000000
0.84509804001425681	-9.0000000000000000
0.90308998699194354	-10.000000000000000
0.90308998699194354	-11.000000000000000
0.90308998699194354	-12.000000000000000
0.90308998699194354	-13.000000000000000
0.90308998699194354	-14.000000000000000

e) M=0.3

Obtém-se os seguintes valores:

0.30102999566398120	-1.0000000000000000
0.30102999566398120	-2.0000000000000000
0.47712125471966244	-3.0000000000000000
0.47712125471966244	-4.0000000000000000
0.47712125471966244	-5.0000000000000000
0.47712125471966244	-6.0000000000000000
0.60205999132796240	-7.0000000000000000
0.60205999132796240	-8.0000000000000000
0.60205999132796240	-9.0000000000000000
0.60205999132796240	-10.000000000000000

0.60205999132796240	-11.000000000000000
0.60205999132796240	-12.000000000000000
0.60205999132796240	-13.000000000000000
0.60205999132796240	-14.000000000000000

Os resultados das letras d) e e) seguem o mesmo estilo, que mostra que os valores da ‘variacao’ ou diferença entre uma iteração e outro são muito pequenos, assim, apenas para certos valores de log10() assim apenas para alguns valores de precisão máxima epsilon adiciona-se uma iteração ‘n’.

Ambos os gráficos apresentaram funcionamento semelhante, a única diferença foi que, dentro do intervalo, M=0.9 apresentou 4 iterações e M=0.3 apresentou 3 iterações.