Projeto 3

Juliana Naomi Yamauti Costa, nºUSP 10260434

29 Abril 2020

A descrição de fenômenos físicos tem como seu coração o Cálculo. Temos como importantes ferramentas a integração e diferenciação quando o assunto é movimento, entretanto, frequentemente encontramos funções difíceis de lidar por não se comportarem de forma regular e previsível.

É neste campo - onde há dificuldade nas abordagens analíticas - em que entram os Métodos Númericos, assim como a Integração e Diferenciação Numérica.

1 Tarefa A

No código abaixo foram obtidas a 1ª, 2ª e 3ª derivadas da seguinte função:

$$f(x) = sinh(2x) \cdot sin\left(\frac{x}{4}\right) \tag{1}$$

As derivadas foram calculadas para comparação com seus respectivos valores ao redor do ponto x=1 e as seguintes equações foram obtidas:

$$f'(x) = 2 \cdot cosh(2x) \cdot sin\left(\frac{x}{4}\right) + \frac{1}{4} \cdot cos\left(\frac{x}{4}\right) \cdot sinh(2x) \tag{2}$$

$$f''(x) = cosh(2x) \cdot cos\left(\frac{x}{4}\right) + \frac{63}{16} \cdot sin\left(\frac{x}{4}\right) \cdot sinh(2x) \tag{3}$$

$$f'''(x) = \frac{61}{8} \cdot sin\left(\frac{x}{4}\right) \cdot cosh\left(\frac{x}{4}\right) + \frac{191}{64} \cdot sinh(2x) \cdot cos\left(\frac{x}{4}\right) \tag{4}$$

Diferentes método foram utilizados para o cálculo das derivadas. A acurácia do resultado pode ser refinada de duas formas: incluindo mais pontos ou diminuindo o intervalo h. Porém, cada uma delas apresentam problemas devido a propagação de erros de arredondamento ao se calcular as derivadas numéricas. Quando h é muito pequeno, o erro de arredondamento cresce e, quando mais pontos são utilizados, os erros se acumulam nas derivadas.

1.1 Algorítmo

Todas as funções utilizadas foram definidas previamente e os valores h foram armazenados como vetores (vh(i)). Os desvios foram armazenados no arquivo de saída com a formatação necessária para que apenas os dígitos mais relevantes a análise fossem retornados.

Para resultados mais acurados, foi utilizada dupla precisão.

tarefaA-10260434.f90

```fortran
Program Tarefa_A

IMPLICIT REAL*8 (a-h,o-z) !dupla precisão
dimension vh(1:14)        !vetor que armazena os valores de h

f(x) = dsinh(2.d0*x)*dsin(x/4.d0)!função
df(x) = 2.d0*dcosh(2.d0*x)*dsin(x/4.d0)+(1.d0/4.d0)*dcos(x/4.d0)*dsinh(2.d0*x)!derivada 1
df2(x) = dcosh(2.d0*x)*dcos(x/4.d0)+(63.d0/16.d0)*dsin(x/4.d0)*dsinh(2.d0*x) !derivada 2
df3(x) = (61.d0/8.d0)*dsin(x/4.d0)*dcosh(2.d0*x)+(191.d0/64.d0)*dsinh(2.d0*x)*dcos(x/4.d0) !derivada 3
ds3f(y,h) = ( f(y+h) - f(y-h) ) /(2*h)!derivada simétrica 3 pts
ds5f(y,h) = ( f(y-2.d0*h)-8.d0*f(y-h)+8.d0*f(y+h)-f(y+2.d0*h) )/(12.d0*h) !derivada simétrica 5 pts
d2sf(y,h) = ( -f(y-2.d0*h)+16.d0*f(y-h)-30.d0*f(y)+16.d0*f(y+h)-f(y+2.d0*h) )/(12.d0*h**2.d0)  !derivada segunda simétrica 5 pts
d3asf(y,h) = ( -f(y-2.d0*h)+2.d0*f(y-h)-2.d0*f(y+h)+f(y+2.d0*h) )/(2.d0*h**3) !derivada terceira a-simétrica 5 pts
df2f(y,h) = ( f(y+h) - f(y) )/h !derivada p/ frente 2 pts
dt2f(y,h) = ( f(y)- f(y-h) )/h !derivada p/ trás 2 pontos

vh = (/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/)

xp=1.d0 !definir o ponto da derivada

open(20, file = 'saida-tarefaA-10260434.txt')
do i=1, 14
     erro1 = abs(df(xp) - ds3f(xp,vh(i)))        !diferença entre o valor exato
     erro2 = abs(df(xp) - df2f(xp,vh(i)))
     erro3 = abs(df(xp) - dt2f(xp,vh(i)))
     erro4 = abs(df(xp) - ds5f(xp,vh(i)))
     erro5 = abs(df2(xp) - d2sf(xp,vh(i)))
     erro6 = abs(df3(xp) - d3asf(xp,vh(i)))

     3 format(f15.8,3f18.11,f18.13,2f18.11)
     write(20,3) vh(i), erro1, erro2, erro3, erro4, erro5, erro6
end do
close(20)
End Program Tarefa_A
```

1.2 Resultados

Na tabela a seguir foram colocados os resultados do programa para cada fórmula de derivação proposta.

h	Derivada simétrica (3 pontos)	Derivada p/ frente 2 pts	Derivada p/ trás 2 pts	Derivada simétrica 5 pts	Derivada segunda simétrica 5 pts	Derivada terceira anti-simétrica 5 pts
0.5	0.78266274607	2.80385377816	1.23852828603	0.2236662989702	0.16163123909	6.56726116362
0.2	0.11847998780	0.85031608232	0.61335610671	0.0050876718210	0.00379654952	0.95051247417
0.1	0.02938552958	0.39004458399	0.33127352482	0.0003126231557	0.00023438100	0.23425519573
0.05	0.00733179035	0.18700806427	0.17234448356	0.0000194560580	0.00001460379	0.05835494233
0.01	0.00029308502	0.03618659966	0.03560042961	0.0000000310873	0.00000002335	0.00233153056
0.005	0.00007326980	0.01801937522	0.01787283562	0.0000000019429	0.00000000147	0.00058286179
0.001	0.00000293077	0.00359211039	0.00358624884	0.0000000000031	0.00000000022	0.00002324829
0.0005	0.00000073269	0.00179532185	0.00179385646	0.0000000000005	0.00000000153	0.00000731660
0.0001	0.00000002931	0.00035894707	0.00035888845	0.0000000000005	0.00000002368	0.00002136900
0.00005	0.00000000733	0.00017946621	0.00017945155	0.0000000000008	0.00000014396	0.00092232064
0.00001	0.00000000029	0.00003589206	0.00003589147	0.0000000000014	0.00000157984	0.04311079877
0.000001	0.00000000003	0.00000358905	0.00000358899	0.0000000000314	0.00031943006	73.09578756897
0.0000001	0.00000000049	0.00000035700	0.00000035799	0.0000000005864	0.02944448407	55493.56464921445
0.00000001	0.00000000178	0.00000002598	0.00000002954	0.0000000009954	0.79457294053	17.58463591726
EXATO		2.7400917441721711			7.17835540972	17.58463591726

Tabela 1: Desvios obtidos pelo programa para cada derivada e seu respectivo tamanho de intervalo (h). Na última linha estão os valores exatos de cada uma.

Os primeiros valores a serem notados na tabela aparecem na última coluna (derivada terceira anti-simétrica 5 pts), quando os valores de h se tornam muito pequenos. Os erros aparentam estar excessivamente grandes, portanto incorretos. Assim, como hipótese, foram atribuídos ao ponto flutuante - decorrendo de um problema computacional, e não numérico. Isto acontece quando é fornecido ao computador um número menor do que ele é capaz gravar, passando o mesmo a retornar valores aleatórios. Podemos observar no gráfico abaixo que para a terceira derivada, quando valores muito pequenos de h são atingidos não é mais possível prever o erro.

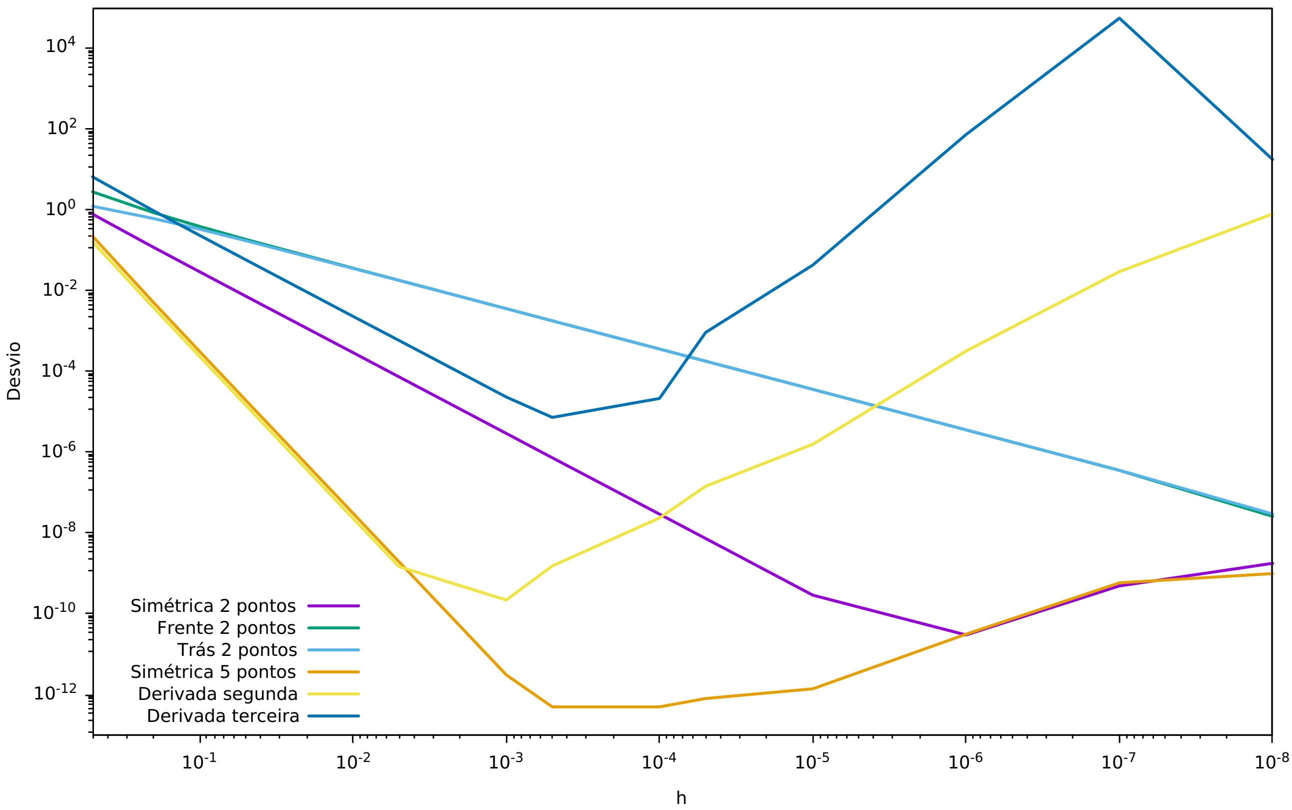

Figura 1: Comportamento do desvio calculado em cada derivada. Escala logarítmica.

Para a primeira derivada, vemos que ao aumentar o número de pontos a precisão aumenta, no entanto, para h muito pequeno essa precisão decai. Podemos ver que este é um dos problemas da derivação numérica, uma vez que geralmente é necessário verificar o melhor parâmetro para se obter o resultado mais preciso, apresentando-se como um método instável. A tabela abaixo indica o melhor h para cada método.

	Derivada simétrica 3 pts	Derivada p/ frente 2 pts	Derivada p/ traz 2 pts	Derivada simétrica 5pts	Derivada segunda simétrica 5pts	Derivada terceira anti-simétrica 5 pts
h ≈	10^{-7}	10^{-8}	10^{-8}	$5 \cdot 10^{-4}$	10^{-3}	$5 \cdot 10^{-4}$

Tabela 2: Melhores valores de h para cada método determinados pela análise direta dos resultados.

Quanto aos métodos utilizados para a derivada de primeira ordem, o erro decai da mesma forma para as derivadas para frente e para trás. Decai mais rapidamente para a derivada simétrica e ainda mais rápido para a simétrica com cinco pontos. Isso ocorre pois o erro de truncamento é proporcional a h, ou seja, ele vai a 0 de acordo com h. As derivadas para frente e para trás tem erro proporcional a h, enquanto as derivadas simétricas de três e cinco pontos tem erros proporcionais a h^2 e h^4, respectivamente. Por isto seus erros decaíram a zero mais rapidamente, como mostrado na Tabela 1, e também são as mais recomendadas.

Também observa-se que para derivadas de ordem maior, mais problemática se torna a aproximação devido aos erros gerados pelo arredondamento. Assim, o erro não depende apenas de h, mas também da função.

2 Tarefa B

Nesta tarefa o valor aproximado da seguinte integral foi calculado:

$$\int_0^1 e^{\frac{x}{2}} \cdot cos(\pi x)dx \tag{5}$$

O método básico de aproximação de uma dada integral definida é chamado de **quadratura numérica**. As técnicas utilizadas estão dentro de uma grupo de fórmulas conhecidas como **Regras de Newton-Cotes**, baseadas na aproximação da função em pontos **igualmente espaçados** por polinômios de diferentes graus. As regras utilizadas foram: Regra do Trapézio, Regra de Simpson e Regra de Bode.

Através do seguinte código, o valor exato foi comparado com o valor obtido por cada método junto ao número de divisões do intervalo.

2.1 Algorítmo

Os intervalos para cada método foram definidos a fim de evitar que não houvesse sobreposição dos mesmos após cada iteração, também respeitando as exigências de cada método. Também foi trabalhado em dupla precisão e os valores de N foram armazenados como vetores.

tarefaB-10260434.f90

```
Program Tarefa_B

IMPLICIT real*8 (a-h,o-z)
dimension N(1:11)        !armazena valores de N

f(x) = dexp(x/2)*dcos(pi*x)        !função
!integral (p/ comparar com o valor exato)
fi(x) = (dexp(x/2)/(pi**2 + 1.d0/4))*(pi*dsin(pi*x) + dcos(pi*x)/2)

a= 0.d0                  !limites de integração
b= 1.d0
pi = 4.d0*datan(1.d0)

rfi = fi(b)-fi(a) !valor exato

N = (/4,8,16,32,64,128,256,512,1024,2048,4096/)

open(10, file='saida-tarefaB-10260434.txt')
do j=1, 11
        soma = 0.d0
        soma1 = 0.d0
        soma2 = 0.d0
        h = (b-a)/N(j)        !valores de h p/ cada N

        !regra do trapézio
        do i=0, N(j)-1
        xi = a + i*h
        soma = soma + (f(xi)+f(xi+h))*(h/2)
        end do

```

```
        !regra de Simpson
        do i=1, N(j)-1, 2
        xi1 = a + i*h
        soma1 = soma1 + (h/3)*(f(xi1+h)+4*f(xi1)+f(xi1-h))
        end do

        !regra de Bode
        do i=0, N(j)-1, 4
        xi2 = a + i*h
        soma2 = soma2 +  ((2*h)/45)*(7*f(xi2)+32*f(xi2+h)+12*f(xi2+2*h)+32*f(xi2+3*h)+7*f(xi2+4*h))
        end do

        soma= abs(soma-rfi)
        soma1 = abs(soma1-rfi)
        soma2 = abs(soma2 -rfi)


        write(10,*) N(j),soma, soma1, soma2
end do
close(10)
End Program Tarefa_B
```

2.2 Resultados

Na tabela a seguir se encontram os desvios (diferença entre o valor exato e o obtido pelo algoritmo) de cada método.

N	h	Regra do Trapézio	Regra de Simpson	Regra de Bode
4	0.2500000	7.1137027110855511E-003	9.5157185224165053E-004	5.0169532364960356E-004
8	0.1250000	1.7376897085270571E-003	5.4314625659135363E-005	5.5025227797544929E-006
16	0.0625000	4.3193219600226596E-004	3.3203081727106731E-006	7.9312993089830641E-008
32	0.0312500	1.0782826388766598E-004	2.0638015049700975E-007	1.2150510320108054E-009
64	0.01562500	2.6947405186023010E-005	1.2881047978252624E-008	1.8892193365260823E-011
128	0.0078125	6.7362477045862512E-006	8.0478904096459303E-010	2.9479196861359469E-013
256	0.00390625	1.6840242049731557E-006	5.0295212439266379E-011	4.6074255521943996E-015
512	0.00195312	4.2100369390674075E-007	3.1434854719236682E-012	2.4980018054066022E-016
1024	0.00097656	1.0525077576151176E-007	1.9614865287564953E-013	5.5511151231257827E-017
2048	0.00048828	2.6312685447171802E-008	1.2517764602648640E-014	3.3306690738754696E-016
4096	0.00024414	6.5781706193313028E-009	1.2212453270876722E-015	3.3306690738754696E-016
EXATO			-0.13087079127	

Tabela 3: Desvios obtidos do cálculo numérico da função sendo integrada em relação ao valor exato, para cada método. Na última linha encontra-se seu valor exato.

É notável a diferença entre as precisões obtidas em cada método. As Regras de Simpson e Bode foram as mais precisas, tendo apenas como contrapartida o maior esforço computacional que requerem. Além disso, diferentemente da diferenciação numérica, temos resultados mais estavéis no sentido de obtermos um limite bem definido quando N passa a ser grande e h pequeno. Isso porque todos os valores da função entram nas fórmulas de quadratura com o mesmo sinal.

O que os resultados indicam é que métodos que utilizam polinômios de maior grau para aproximação do valor da função apresentam maior precisão, assim como utilizar valores pequenos de h. No entanto, utilizar polinôminios de grau muito alto podem tornar o método inapropriado, já que essas funções tendem a oscilar violentamente gerando interpolaçoes inadequadas. Já é possível observar uma pequena oscilação na Regra de Bode, em que os últimos dois resultados diminuíram a precisão.

3 Tarefa C

Nesta tarefa tentamos encontrar as raízes da seguinte equação:

$$x^3 - 21x - 20 = 0 \tag{6}$$

Obter a solução para uma dada equação é equivalente a encontrar o zero de uma função real apropriada. Foram utilizados três métodos diferentes: Método direto (método da bisseção), Método de Newton-Raphsen e Método da Secante.

3.1 Métodos

O **método da bisseção** explora o fato de que uma função contínua $f : [a,b] \to \mathbb{R}$ com $f(a) \cdot f(b) < 0$ tem um zero no intervalo (a,b), garantido pelo Teorema de Bolzano. Assim, sua aproximação inicial é o ponto médio do intervalo [a,b]. O algoritmo consiste nessa verificação e repetição do processo em intervalos cada vez menores até que se obtenha a aproximação desejada.

Já o **método de Newton-Raphson** toma como auxiliar a reta tangente da função no ponto. Um ponto inicial é escolhido e, após isso, calcula-se a reta tangente por meio de sua derivada da função no ponto e a sua intersecção no eixo das abcissas, aproximando-se cada vez mais do valor da raíz.

O **método da secante** é uma variação do método de Newton-Raphson, porém ao invés de utilizar linhas tangentes, é auxiliada por linhas secantes.

3.2 Algorítmo

O código abaixo foi utilizado para a verificação dos três métodos, feito em dupla precisão. Como há mais de uma raíz, foi definido um intervalo de procura e, para cada intervalo, foi analisado se havia alguma raíz e em seguida os métodos foram aplicados **até** que fosse obtida uma raíz com a tolerância exigida (10^{-6}).

O espaçamento utilizado foi de $h = 0.55$, pois não foi possível encontrar as raízes quando elas estão nas extremidades dos intervalos analisados. Inspecionando o código, verificou-se que isso ocorre porque na etapa inicial de verificação de raízes no espaço, o produto $f(a) \cdot f(b)$ resultará em zero (pois é uma raíz!), ocasionando como resposta que não há raíz. Apenas com a mudança no espaçamento foi possível fazer o código funcionar corretamente.

Como não há desvio inicial, o valor utilizado foi colocado como duas vezes o desvio solicitado (escrito como 'erro' no programa).

tarefaC-10260434.f90

```
Program Tarefa_C

IMPLICIT REAL*8 (a-h,o-z) !dupla precisão

f(x) = x**3 - 21.d0*x - 20.d0        !função
df(x)= 3.d0*x**2 - 21.d0        !derivada da função
a = -10.d0        !xinicial
b = 10.d0        !xfinal
h = 0.55        !espaçamento
erro = 10d-6        !tolerância

print*,'-------------------Método Direto-------------------'
```

```fortran
do d = a, b-h, h          !procura raíz em cada intervalo
        xleft = d
        xright = xleft + h
        desvio = 2.d0*erro        !desvio inicial
        if (f(xright)*f(xleft) .lt. 0) then        !verifica se há raíz neste intervalo
                i = 0
                do while (desvio .gt. erro)
                        c = (xright+xleft)/2.d0        !ponto médio do intervalo
                        if (f(xright)*f(c) .gt. 0) then                !se a raíz não está a direita
                                xright = c        !então está a esquerda (redefine intervalo)
                        else
                                xleft = c
                        end if
                        desvio = abs(f(xleft)-f(xright))
                        i = i + 1
                        print'(f16.11, i5)', c, i
                end do
        end if
end do

print*,'-----------------Método de Newton------------------'
do d = a, b-h, h
        xleft = d
        xright = xleft + h
        desvio = 2.d0*erro
        if (f(xright)*f(xleft) .lt. 0) then        !verifica se há raíz, então aplica o método
                j = 0
                do while (desvio .gt. erro)
                        xleft = xright - f(xright)/df(xright)        !aplicação do método
                        desvio = abs(f(xleft)-f(xright))
                        xright= xleft
                        j = j + 1
                        print'(f16.11, i5)', xleft, j
                end do
        end if
end do

print*,'----------------Método das Secantes----------------'
do d = a, (b-2.d0*h), h        !raciocínio análogo
        xleft = d
        xright = xleft + h
        desvio = 2.d0*erro
        k = 0
```

```
        if (f(xright)*f(xleft) .lt. 0) then
                do while (desvio .gt. erro)
                        xleft = xright - f(xright)*((xright-(xright-h))/(f(xright)-f(xright-h)))
                        desvio = abs(f(xleft)-f(xright))
                        xright= xleft
                        k = k + 1
                        print'(f16.11, i5)', xleft, k
                end do
        end if
end do
End Program Tarefa_C
```

3.3 Resultados

Cada tabela abaixo apresenta as raízes obtidas pelos método a cada iteração feita, com espaçamento h = 0.55 e tolerância = 10^{-6}.

3.3.1 Procura Direta

Para o método direto foram necessária entre 20 e 22 iterações para que fosse alcançada a precisão solicitada. Observa-se que na sexta iteração a precisão obtida ainda está entre as segunda e quarta casa decimal, indicando ser um processo relativamente lento quando comparado as outras técnicas. Isso ocorre porque a taxa de convergência deste método é linear, entretanto, pode ser utilizado para encontrar raízes em um intervalo e, em seguida, métodos que convergem mais rapidamente são aplicados.

Iteração	Procura Direta		
	r_1	r_2	r_3
0	-4.22499987483	-0.92499980330	5.12500032783
1	-4.08749987185	-1.06249980628	4.98750032485
2	-4.01874987036	-0.99374980479	5.05625032634
3	-3.98437486961	-1.02812480554	5.02187532559
4	-4.00156236999	-1.01093730517	5.00468782522
5	-3.99296861980	-1.00234355498	4.99609407503
6	-3.99726549489	-0.99804667989	5.00039095012
EXATOS	-4.00000000000	-1.00000000000	5.00000000000

Tabela 4: Raízes obtidas pela Procura Direta a cada iteração. Na última linha encontra-se o valor exato de cada uma.

3.3.2 Método de Newton-Raphson e Método da Secante

Observa-se que são métodos que convergem mais rapidamente para as soluções, uma vez que foram necessárias 7 a 9 iterações para o Método da Secante e apenas 4 para o Método de Newton.

Iteração	Método da Secante		
	r_1	r_2	r_3
0	-3.99046047763	-1.01086748663	4.98418940464
1	-3.99808051741	-0.99863627550	5.00282828451
2	-3.99961011305	-1.00016527846	4.99951507381
3	-3.99992065738	-0.99997988106	5.00008377920
4	-3.99998384752	-1.00000244773	4.99998554460
5	-3.99999671144	-0.99999970218	5.00000249472
6	-3.99999933046	-1.00000003624	4.99999956948
EXATOS	-4.00000000000	-1.00000000000	5.00000000000

Tabela 5: Raízes obtidas pelo Método da Secante a cada iteração. Na última linha encontra-se o valor exato de cada uma.

Apesar de se apresentar como um método menos eficiente que o de Newton, ainda assim tem taxa de convergência superlinear (converge rapidamente em sua parte final). Assim, neste caso converge mais rapidamente que o método da bisseção (não estritamente verdade em todos os casos).

Iteração	Newton-Raphson		
	r_1	r_2	r_3
0	-4.00115277183	-0.98572151196	5.03802653301
1	-4.00000059012	-0.99996650254	5.00039532468
2	-4.00000000000	-0.99999999981	5.00000004340
3	-4.00000000000	-1.00000000000	5.00000000000
4	-	-	-
5	-	-	-
6	-	-	-
EXATOS	-4.00000000000	-1.00000000000	5.00000000000

Tabela 6: Raízes obtidas pelo Método de Newton-Raphsen a cada iteração. Na última linha encontra-se o valor exato de cada uma.

O método de Newton tem convergência quadrática, por isto menos iterações foram necessárias. Entretanto, é fundamental notar suas limitações, pois tanto a função, quanto a sua derivada devem ser analíticas - exigências não essenciais ao método da Secante.

Referências

JUDICE, Joaquim J; FERNANDES, Luıs M. EQUACOES NAO LINEARES E OPTIMIZACAO UNIDIMENSIONAL.

KOONIN, Steven E. **Computational physics: Fortran version**. [S.l.]: CRC Press, 2018.

MØRKEN, Knut. Numerical algorithms and digital representation. **Lecture Notes for course MATINF1100 Modelling and Computations,(University of Oslo, Ch. 11, 2010)**, 2013.

PANG, Tao. **An introduction to computational physics**. [S.l.]: American Association of Physics Teachers, 1999.

WIKIPÉDIA. **Fórmulas de Newton-Cotes — Wikipédia, a enciclopédia livre**. [S.l.: s.n.], 2019. [Online; accessed 18-abril-2019]. Disponível em: ¡https://pt.wikipedia.org/w/index.php?title=F%C3%B3rmulas_de_Newton-Cotes&oldid=54861770¿.

________. **Método da bisseção — Wikipédia, a enciclopédia livre**. [S.l.: s.n.], 2019. [Online; accessed 19-setembro-2019]. Disponível em: ¡https://pt.wikipedia.org/w/index.php?title=M%C3%A9todo_da_bisse%C3%A7%C3%A3o&oldid=56282283¿.

________. **Método das secantes — Wikipédia, a enciclopédia livre**. [S.l.: s.n.], 2020. [Online; accessed 16-abril-2020]. Disponível em: ¡https://pt.wikipedia.org/w/index.php?title=M%C3%A9todo_das_secantes&oldid=58054196¿.

________. **Método de Newton–Raphson — Wikipédia, a enciclopédia livre**. [S.l.: s.n.], 2020. [Online; accessed 2-abril-2020]. Disponível em: ¡https://pt.wikipedia.org/w/index.php?title=M%C3%A9todo_de_Newton%E2%80%93Raphson&oldid=57943435¿.