

Cálculo Numérico

Metódo de Newton-Raphson e suas aplicações



Igor Ferris Eduardo



Gabriel Pires Cruz

Sumário

Pag 02

- Definição

- Equação de Newton-Raphson

- Aplicando o método Newton-Raphson

- Exercicio 1

- Exercicio 2

- Exercicio 3

- Exercicio 4

- Considerações Finais

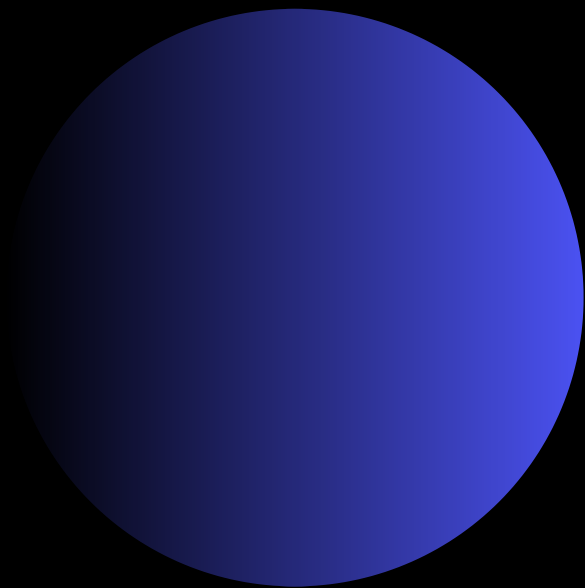
DEFINIÇÃO

Pag 03

Método de Newton-Raphson

O Método de Newton-Raphson é uma técnica numérica amplamente utilizada para encontrar raízes de funções reais. Este método é especialmente eficaz em problemas de otimização e análise de dados, onde a identificação de pontos críticos é essencial. A abordagem se baseia na ideia de que uma função pode ser aproximada por sua tangente em um ponto inicial, permitindo que se faça uma iteração sucessiva até que a raiz desejada seja encontrada com um nível de precisão aceitável.

Equação de Newton-Raphson



$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Imagem 1 - Equação de Newton-Raphson

Onde:

x_n = É a n-ésima aproximação da raiz.

x_{n+1} = É a próxima aproximação da raiz.

$f(x_n)$ = É o valor da função em x_n , ou seja, quanto a função vale nesse ponto.

$f'(x_n)$ = É a derivada da função em x_n , ou seja, a inclinação da reta tangente à curva $F(x_n)$ naquele ponto.

Aplicando o método Newton-Raphson

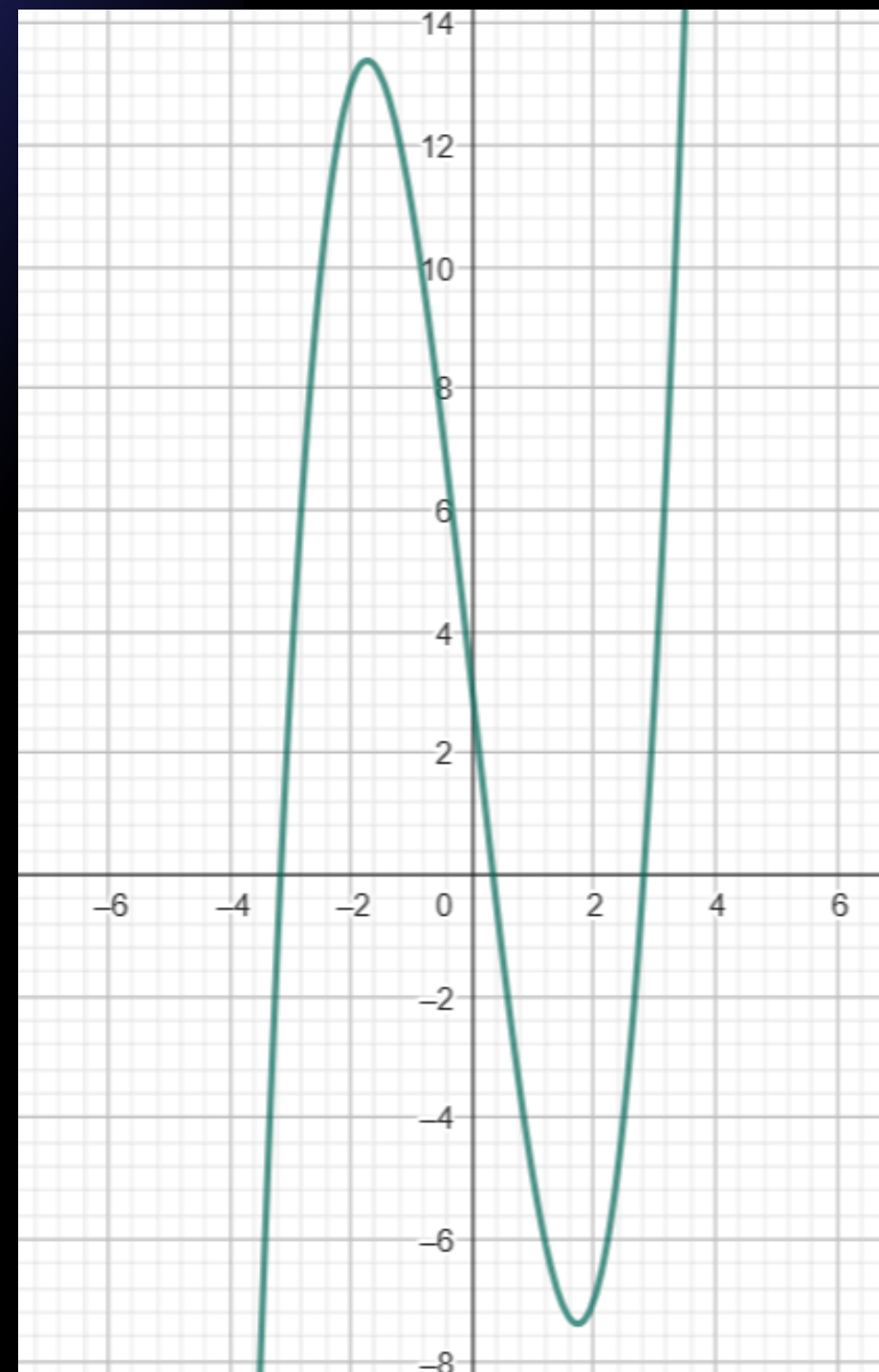
1. Escolha um valor inicial para x .
 - Este valor é uma estimativa onde se espera que haja uma raiz.
2. Encontrar o valor da função utilizando o valor de x do passo 1.
3. Encontrar o valor da derivada da função utilizando o valor de x do passo 1.
4. Use os valores encontrados para aprimorar a estimativa de raiz usando a seguinte fórmula:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

5. Repetir os passos 2-4 até a estimativa da raiz convergir para um valor preciso.

EXERCÍCIO 1

Determine a raiz da função $f(x) = x^3 - 9x + 3$ dentro dos intervalos de $[0,1]$ e com uma tolerância de $\varepsilon = 1,0 \times 10^{-6}$.



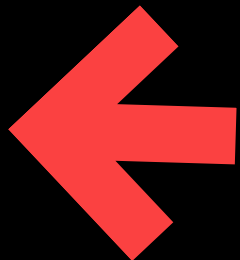
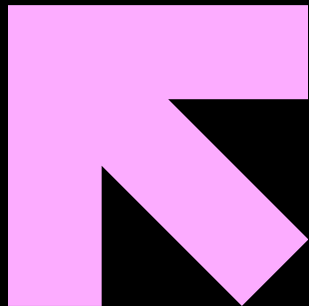
RESOLUÇÃO EM PLANILHA

Com os dados mostrados , podemos que :

$F(x) = x^3 - 9x + 3$
 $F'(x) = 3x^2 - 9$
 $x = \bar{x} = (1+0)/ 2 = 0,5$
 $\varepsilon = 1,0 \times 10^{-6}.$
R : $x = 0,337609$

Iteração (n)	x_n	f(x_n)	f'(x_n)	x_{n+1}
1	0,5	-1,375	-8,25	0,333333
2	0,333333	0,037037	-8,666667	0,337607
3	0,337607	0,000018	-8,658065	0,337609
4	0,337609	0	-8,658061	0,337609
5	0,337609	0	-8,658061	0,337609

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$



A partir na interação 4 , podemos ver que o resultado obtido da interação 4 e substitiundo-se na interação 5 , deu o mesmo resultado

RESOLUÇÃO EM PYTHON

Nesta imagem(2) , é a construção dos comandos em python

(2)

```
import numpy as np
import math as mt
import sympy as sp

# --- Algoritmo de Newton-Raphson ---

x = sp.symbols('x')
print("Método de Newton-Raphson\Digite a função f(x): ")
f = input('f(x): ')
fx = sp.sympify(f)
df = sp.diff(f,x)
print("d/dx = " , df)
print("digite o chute inicial: ")
x0 = float(input('x0: '))
print("Digite a tolerância: ")
tol = float(input("Tol: "))
print("Número máximo de iterações: ")
max_iter = int(input('max_iter = '))
i = 0
err = 100

# Exibe primeira iteração
print('Iteração: 0, f(x0) = ' + str(fx.subs(x, x0)) + ', d/dx = ' + str(df.subs(x, x0)))

# Loop de iteração
while (tol < err ) and (i < max_iter):
    i = i + 1
    xi = x0 - (fx.subs(x, x0) / df.subs(x, x0)) # Fórmula de Newton-Raphson
    print("Iteração: " + str(i) + ', x = ' + str(xi) + ', f(x) = ' + str(fx.subs(x, xi)))
    if (xi != 0):
        err = abs((xi - x0) / xi) # Erro relativo
        x0 = xi #

# Resultado final
print(f"\nA raiz aproximada é: {xi}, com erro relativo de {err} após {i} iterações.")
```


RESOLUÇÃO EM PYTHON

Pag 09

```
Método de Newton-Raphson\Digite a função f(x):  
f(x): x**3 -9*x +3  
d/dx = 3*x**2 - 9  
digite o chute inicial:  
x0: 0.5  
Digite a tolerância:  
Tol: 0.000001  
Número máximo de iterações:  
max_iter = 10  
Iteração: 0, f(x0) = -1.375000000000000, d/dx = -8.250000000000000  
Iteração: 1, x = 0.3333333333333333, f(x) = 0.0370370370370368  
Iteração: 2, x = 0.337606837606838, f(x) = 1.83408850951139e-5  
Iteração: 3, x = 0.337608955965313, f(x) = 4.54480897360554e-12  
Iteração: 4, x = 0.337608955965838, f(x) = 4.44089209850063e-16  
  
A raiz aproximada é: 0.337608955965838, com erro relativo de 1.55479716034515E-12 após 4 iterações.
```

(3)

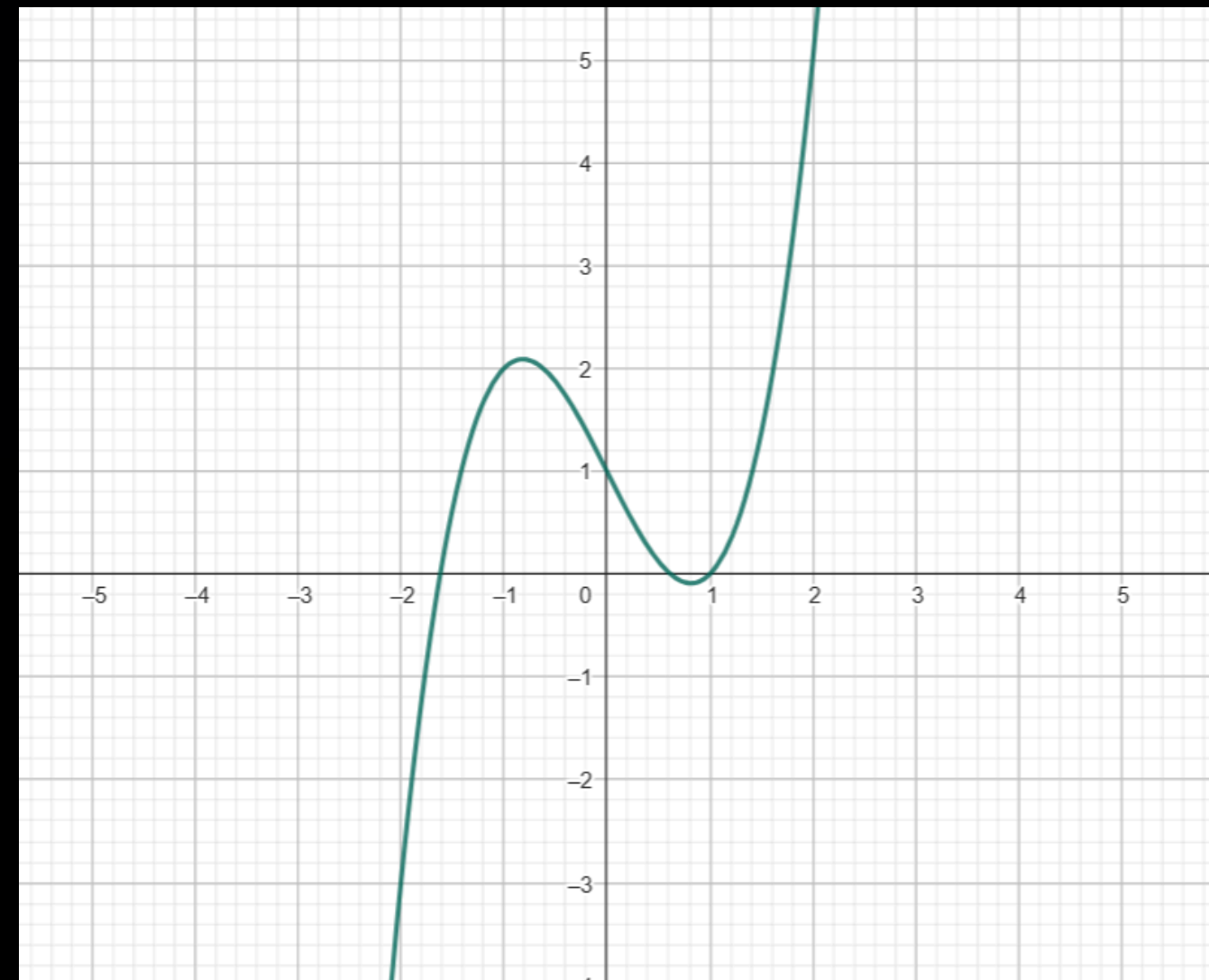
Na imagem (3) o resultado com as interações e a raiz de x.

EXERCÍCIO 2

Usar o método Newton-Raphson para encontrar uma raiz negativa do seguinte polinômio cúbico:

$$f(x) = x^3 - 2x + 1$$

Utilizar como ponto de partida o valor inicial de $x = -1,5$ para encontrar por aproximações sucessivas o valor da raiz com três casas decimais de precisão.



RESOLUÇÃO EM PLANILHA

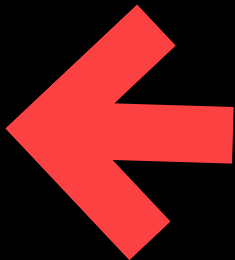
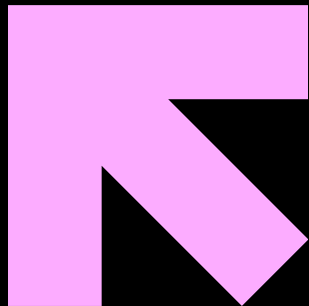
Com os dados mostrados , podemos que :

$F(x) = x^3 - 2x + 1$
 $F'(x) = 3x^2 - 2$
 $x = -1,5$
 $\varepsilon = 1,0 \times 10^{-3}.$

R : $x = -1,618$

Iteração (n)	x_n	f(x_n)	f'(x_n)	x_{n+1}
1	-1,5	0,625	4,75	-1,632
2	-1,632	-0,08	5,986	-1,618
3	-1,618	-0,001	5,856	-1,618
4	-1,618	0	5,854	-1,618

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$



A partir na interação 3 , podemos ver que o resultado obtido da interação 3 e substitiundo-se na interação 4 , deu o mesmo resultado

RESOLUÇÃO EM PYTHON

Na imagem(4) , é a construção dos comandos em python

(4)

Pag 12

```
import numpy as np
import math as mt
import sympy as sp

# --- Algoritmo de Newton-Raphson ---

x = sp.symbols('x')
print("Método de Newton-Raphson\Digite a função f(x): ")
f = input('f(x): ')
fx = sp.sympify(f)
df = sp.diff(f,x)
print("d/dx = " , df)
print("digite o chute inicial: ")
x0 = float(input('x0: '))
print("Digite a tolerância: ")
tol = float(input("Tol: "))
print("Número máximo de iterações: ")
max_iter = int(input('max_iter = '))
i = 0
err = 100

# Exibe primeira iteração
print('Iteração: 0, f(x0) = ' + str(fx.subs(x, x0)) + ', d/dx = ' + str(df.subs(x, x0)))

# Loop de iteração
while (tol < err ) and (i < max_iter):
    i = i + 1
    xi = x0 - (fx.subs(x, x0) / df.subs(x, x0)) # Fórmula de Newton-Raphson
    print("Iteração: " + str(i) + ', x = ' + str(xi) + ', f(x) = ' + str(fx.subs(x, xi)))
    if (xi != 0):
        err = abs((xi - x0) / xi) # Erro relativo
        x0 = xi

# Resultado final
print(f"\nA raiz aproximada é: {xi}, com erro relativo de {err} após {i} iterações.")
```

RESOLUÇÃO EM PYTHON

```
Método de Newton-Raphson\Digite a função f(x):  
f(x): x**3 -2*x +1  
d/dx = 3*x**2 - 2  
digite o chute inicial:  
x0: -1.5  
Digite a tolerância:  
Tol: 0.001  
Número máximo de iterações:  
max_iter = 10  
Iteração: 0, f(x0) = 0.6250000000000000, d/dx = 4.750000000000000  
Iteração: 1, x = -1.63157894736842, f(x) = -0.0801866161247990  
Iteração: 2, x = -1.61818358946881, f(x) = -0.000875886502743395  
Iteração: 3, x = -1.61803400730379, f(x) = -1.08616372340720e-7  
  
A raiz aproximada é: -1.61803400730379, com erro relativo de 0.0000924468610375548 após 3 iterações.
```

(5)

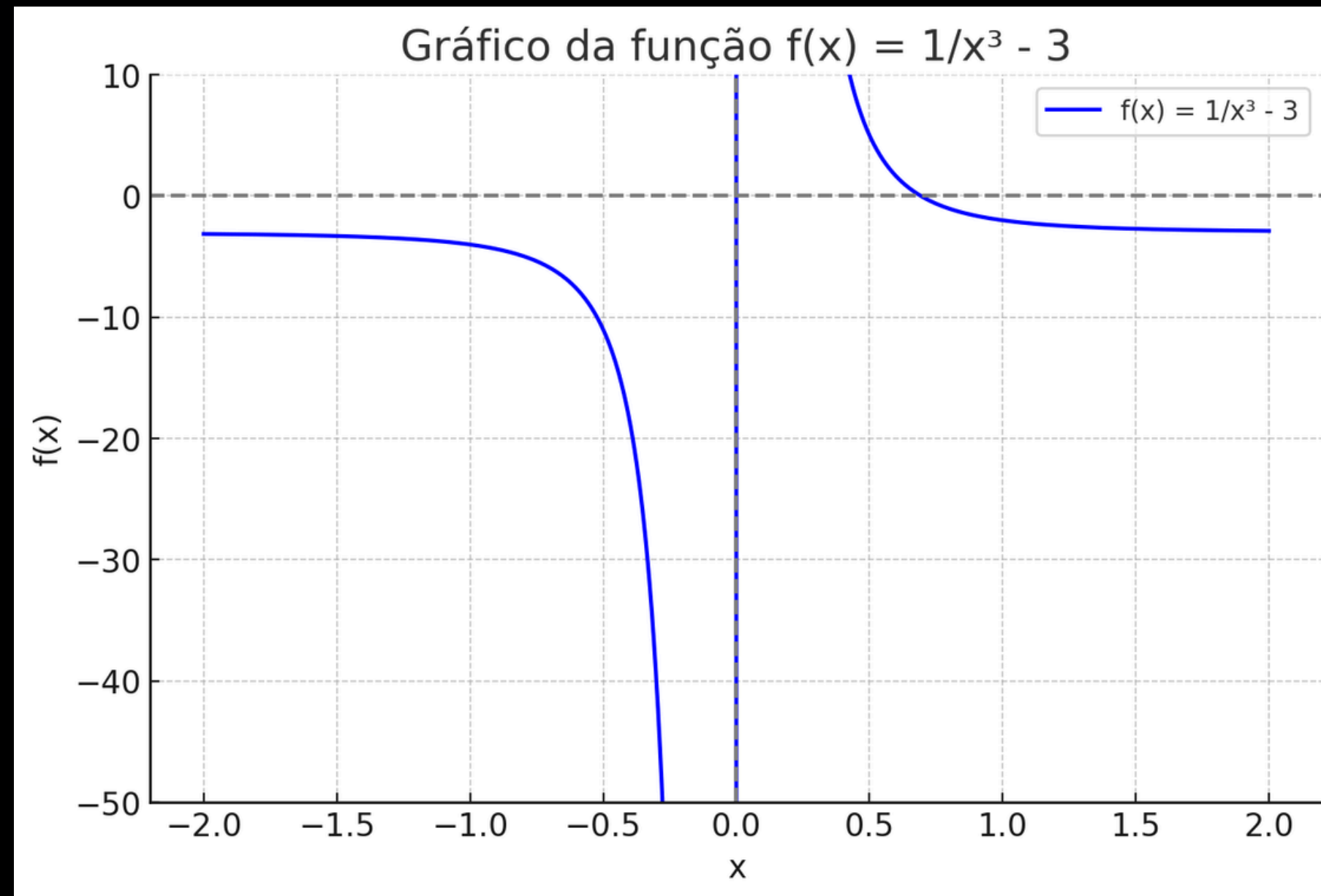
Na imagem (5) o resultado com as interações e a raiz de x.

EXERCÍCIO 3

Pag 14

Descubra o valor para x , que faz $F(x)$ ser igual a zero:

$$1/x^3 - 3 = 0$$



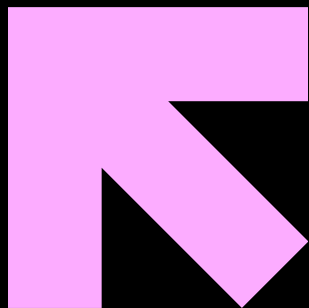
RESOLUÇÃO EM PLANILHA

Com os dados mostrados , podemos que :

$F(x) = 1/x^3 - 3$
 $F'(x) = 3/x^4$
x é aproximadamente 0.5
O erro deve ser igual a 0

R : x = 0,693361

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$



3. Qual o valor em que a função $1/x^3 - 3$ é igual a 0			
$f(x)= 1/x^3 - 3$	$f'(x)= -3/x^4$		
Xn	F (Xn)	F'(Xn)	Xn+1
0,5	5	-48	0,60416666666667
0,60416666666667	1,53450326	-22,51615412	0,67231788164304
0,67231788164304	0,2906069678	-14,68326393	0,69210959614707
0,69210959614707	0,01630592768	-13,07440011	0,69335676064694
0,69335676064694	0,00005858974776	-12,98058414	0,69336127429187
0,69336127429187	0,0000000007628120358	-12,98024614	0,69336127435064
0,69336127435064	0	-12,98024613	0,69336127435064
0,69336127435064	0	-12,98024613	0,69336127435064

A partir na interação 7 , podemos ver que o resultado obtido da interaç

RESOLUÇÃO EM PYTHON

Na imagem(6) , é a construção do comando em python

(6)

```

1 import numpy as np
2 import math as mt
3 import sympy as sp
4
5 x = sp.symbols('x')
6 print("Método de Newton-Raphson\Digite sua função f(x): ")
7 f = input('f(x): ')
8 fx = sp.sympify(f)
9 df = sp.diff(f,x)
10 print("d/dx = " , df)
11 print("digite o valor que você estima que é a raiz: ")
12 x0 = float(input('x0: '))
13 print("Digite a tolerância do erro: ")
14 tol_input = input("Tol: ")
15 if '%' in tol_input:
16     tol = float(tol_input.strip('%')) / 100
17 else:
18     tol = float(tol_input)
19 tol = float(tol)
20 print("Número máximo de iterações: ")
21 max_iter = int(input('max_iter = '))
22 i = 0
23 err = 100
24 # Exibe primeira iteração
25 print('Iteração: 0, f(x0) = ' + str(fx.subs(x, x0)) + ', d/dx = ' + str(df.subs(x, x0)))
26 # Loop de iteração
27 while (tol < err ) or (i < max_iter):
28     i = i + 1
29     xi = x0 - (fx.subs(x, x0) / df.subs(x, x0)) # Fórmula de Newton-Raphson
30     print("Iteração: " + str(i) + ', x = ' + str(xi) + ', f(x) = ' + str(fx.subs(x, xi)))
31     if (xi != 0):
32         err = abs(float(xi - x0)) / abs(float(xi)) # Erro relativo
33     x0 = xi # Atualiza chute
34 # Resultado final
35 print(f"\nA raiz aproximada é: {xi}, com erro relativo de {err} após {i} iterações.")

```

```
Método de Newton-Raphson\Digite sua função f(x):  
f(x): 1/x^3 - 3  
d/dx = -3/x**4  
digite o valor que você estima que é a raiz:  
x0: 0.5  
Digite a tolerância do erro:  
Tol: 0.01%  
Número máximo de iterações:  
max_iter = 10  
Iteração: 0, f(x0) = 5.000000000000000, d/dx = -48.00000000000000  
Iteração: 1, x = 0.6041666666666667, f(x) = 1.53450325966624  
Iteração: 2, x = 0.672317881643036, f(x) = 0.290606967755961  
Iteração: 3, x = 0.692109596147066, f(x) = 0.0163059276798054  
Iteração: 4, x = 0.693356760646939, f(x) = 5.85897477631825e-5  
Iteração: 5, x = 0.693361274291868, f(x) = 7.62812035759453e-10  
Iteração: 6, x = 0.693361274350635, f(x) = 4.44089209850063e-16  
Iteração: 7, x = 0.693361274350635, f(x) = 4.44089209850063e-16  
Iteração: 8, x = 0.693361274350635, f(x) = 4.44089209850063e-16  
Iteração: 9, x = 0.693361274350635, f(x) = 4.44089209850063e-16  
Iteração: 10, x = 0.693361274350635, f(x) = 4.44089209850063e-16
```

A raiz aproximada é: 0.693361274350635, com erro relativo de 0.0 após 10 iterações.

Na imagem (7)
o resultado
com as
iterações e a
raiz da
equação.

EXERCÍCIO 4

Pag 18

Descubra o resultado da seguinte operação através do método Newton-Raphson:

$$2\sqrt{2}$$

RESOLUÇÃO EM PLANILHA

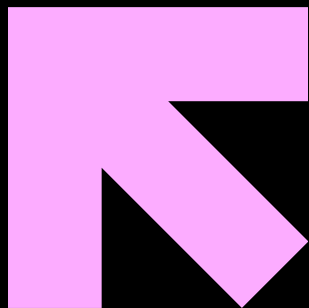
Com os dados mostrados , temos que $2 \cdot 2^{1/2}$ é igual a x, e manipulando a equação obtemos:

$F(x) = x^2/4 - 2$

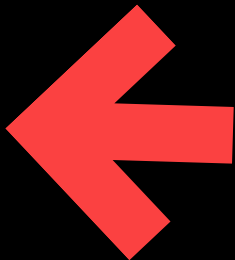
$F'(x)=x/2$

e para obtermos uma bom resultado utilizamos 10 interações

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$



4. Descubra o valor de duas vezes a raiz quadrada de dois			
$f(x)= x^2/4 - 2$	$f'(x)= x/2$		
X_n	$F(X_n)$	$F'(X_n)$	X_{n+1}
3	0,25	1,5	2,8333333333333330
2,8333333333333330	0,00694444444444	1,416666667	2,828431372549020
2,828431372549020	0,000006007304883	1,414215686	2,828427124749380
2,828427124749380	0	1,414213562	2,828427124746190
2,828427124746190	0	1,414213562	2,828427124746190
2,828427124746190	0	1,414213562	2,828427124746190
2,828427124746190	0	1,414213562	2,828427124746190
2,828427124746190	0	1,414213562	2,828427124746190



A partir na interação 2 , podemos ver que o resultado obtido da interação 2 e substitiundo-se na interação 3 , deu o mesmo resultado

RESOLUÇÃO EM PYTHON

Na imagem (8) o resultado com as interações e a raiz da equação.

(8)

```

1 import numpy as np
2 import math as mt
3 import sympy as sp
4
5 x = sp.symbols('x')
6 print("Método de Newton-Raphson\Digite sua função f(x): ")
7 f = input('f(x): ')
8 fx = sp.sympify(f)
9 df = sp.diff(f,x)
10 print("d/dx = " , df)
11 print("digite o valor que você estima que é a raiz: ")
12 x0 = float(input('x0: '))
13 print("Digite a tolerância do erro: ")
14 tol_input = input("Tol: ")
15 if '%' in tol_input:
16     tol = float(tol_input.strip('%')) / 100
17 else:
18     tol = float(tol_input)
19 tol = float(tol)
20 print("Número máximo de iterações: ")
21 max_iter = int(input('max_iter = '))
22 i = 0
23 err = 100
24 # Exibe primeira iteração
25 print('Iteração: 0, f(x0) = ' + str(fx.subs(x, x0)) + ', d/dx = ' + str(df.subs(x, x0)))
26 # Loop de iteração
27 while (tol < err ) or (i < max_iter):
28     i = i + 1
29     xi = x0 - (fx.subs(x, x0) / df.subs(x, x0)) # Fórmula de Newton-Raphson
30     print("Iteração: " + str(i) + ', x = ' + str(xi) + ', f(x) = ' + str(fx.subs(x, xi)))
31     if (xi != 0):
32         err = abs(float(xi - x0)) / abs(float(xi)) # Erro relativo
33     x0 = xi # Atualiza chute
34 # Resultado final
35 print(f"\nA raiz aproximada é: {xi}, com erro relativo de {err} após {i} iterações.")

```


RESOLUÇÃO EM PYTHON

Pag 21

```
Método de Newton-Raphson\Digite sua função f(x):
f(x): x**2/4 - 2
d/dx = x/2
digite o valor que você estima que é a raiz:
x0: 3
Digite a tolerância do erro:
Tol: 0.5%
Número máximo de iterações:
max_iter = 10
Iteração: 0, f(x0) = 0.2500000000000000, d/dx = 1.5000000000000000
Iteração: 1, x = 2.833333333333333, f(x) = 0.006944444444444464
Iteração: 2, x = 2.82843137254902, f(x) = 6.00730488287127e-6
Iteração: 3, x = 2.82842712474938, f(x) = 4.51061410444709e-12
Iteração: 4, x = 2.82842712474619, f(x) = 4.44089209850063e-16
Iteração: 5, x = 2.82842712474619, f(x) = -4.44089209850063e-16
Iteração: 6, x = 2.82842712474619, f(x) = 4.44089209850063e-16
Iteração: 7, x = 2.82842712474619, f(x) = -4.44089209850063e-16
Iteração: 8, x = 2.82842712474619, f(x) = 4.44089209850063e-16
Iteração: 9, x = 2.82842712474619, f(x) = -4.44089209850063e-16
Iteração: 10, x = 2.82842712474619, f(x) = 4.44089209850063e-16

A raiz aproximada é: 2.82842712474619, com erro relativo de 1.570092458683775e-16 após 10 iterações.
```

(9)

Na imagem (9) o resultado com as interações e a raiz da equação.

CONSIDERAÇÕES FINAIS

- Com base no conteúdo apresentado no trabalho, é possível concluir que o método de Newton-Raphson é uma ferramenta poderosa e eficiente na resolução de equações não lineares. A aplicação prática por meio de planilhas e códigos em Python evidenciou como esse método é útil para obter aproximações precisas das raízes de funções, mesmo com poucos passos iterativos, desde que se escolha um bom ponto inicial. A compreensão da fórmula e da lógica por trás das iterações é essencial para seu uso correto.
- Através dos exercícios desenvolvidos, foi possível observar que a convergência do método depende diretamente da função analisada e do valor inicial escolhido. Em todos os casos, o uso do critério de parada baseado no erro permitiu verificar com clareza a precisão dos resultados obtidos, o que reforça a confiabilidade do método quando bem aplicado. Além disso, a comparação entre os resultados obtidos manualmente e aqueles gerados por código confirmou a consistência do processo.
- Por fim, este trabalho proporcionou não apenas um entendimento teórico do método de Newton-Raphson, mas também uma experiência prática em sua aplicação, o que fortalece o aprendizado em Cálculo Numérico. O uso de ferramentas computacionais mostrou-se extremamente útil para agilizar os cálculos e validar os resultados, demonstrando a importância da programação na solução de problemas matemáticos complexos. O estudo realizado serve como base para aprofundamentos futuros em métodos numéricos mais avançados.

Obrigado