Universidade Federal de Juiz de Fora
Pós-Graduação em Modelagem Computacional
Métodos Numéricos

Eduardo Santos de Oliveira Marques

Atividade 2
Métodos de Solução de Sistemas Não-Lineares

Juiz de Fora
2023

Questão 1. Verdadeiro ou falso: Um pequeno resíduo $\|f(x)\|$ garante uma solução precisa de um sistema de equações lineares.

Resolução:

Falso. A magnitude de um pequeno resíduo $\|f(x)\|$ não garante necessariamente uma solução precisa de um sistema de equações lineares. A magnitude do resíduo é uma medida da discrepância entre as soluções estimadas e as soluções reais do sistema.

Embora um resíduo pequeno possa indicar que a solução está próxima da solução exata, outros fatores como a matriz de coeficientes, a presença de erros de arredondamento ou a condição do sistema, também podem afetar a precisão da solução. Para garantir uma solução precisa, é necessário considerar vários aspectos do sistema e do método de solução empregado.

Questão 2. Verdadeiro ou falso: O método de Newton é um exemplo de esquema iterativo de ponto-fixo.

Resolução:

Falso. O método de Newton não é um exemplo de esquema iterativo de ponto fixo. O método de Newton é um algoritmo iterativo para encontrar raízes de uma função, ou seja, é usado para encontrar os valores de x que satisfazem a equação $f(x) = 0$. Em cada iteração, esse método utiliza a derivada da função para aproximar a raiz de forma mais eficiente.

Um esquema iterativo de ponto-fixo é um algoritmo utilizado para encontrar o ponto fixo de uma função, ou seja, encontrar o valor de x que satisfaz a equação $x = g(x)$. Nesse tipo de esquema, não é necessário calcular derivadas da função. O método de Newton não se enquadra nessa definição, pois não busca um ponto-fixo, mas sim uma raiz da função.

Questão 3. Verdadeiro ou falso: Para uma dada precisão, a convergência dos métodos iterativos classificados como superlinear sempre será menor que a dos métodos iterativos com convergência linear.

Resolução:

Falso. A convergência dos métodos iterativos classificados como superlinear é geralmente melhor do que a dos métodos iterativos com convergência linear, quando se trata de alcançar uma determinada precisão. É importante destacar que a velocidade de convergência pode depender do problema específico e da qualidade das aproximações iniciais.

Os métodos iterativos com convergência linear convergem para a solução com uma taxa constante, isso significa que a cada iteração o erro diminui por um fator fixo. Exemplos de métodos com convergência linear incluem: o método de Jacobi e o método de Gauss-Seidel para resolução de sistemas lineares.

Já os métodos iterativos classificados como superlinear convergem mais rapidamente, com uma taxa que aumenta conforme a iteração avança. Alguns exemplos de métodos superlineares incluem o método de Newton e o método de secante para encontrar raízes de funções.

Questão 4. Suponha que você esta usando um método iterativo para resolver uma equação não linear $f(x) = 0$ para uma raiz que é mal-condicionada e você precisa escolher um teste de convergência. Qual critério de parada para determinar que a aproximação x_k é adequada? Quando $\|f(x)\|$ é pequena ou quando $\|x_k - x_{k-1}\|$ é pequena?

Resolução:

Se trata de resolver uma equação não linear $f(x) = 0$ com uma raiz mal-condicionada, é preferível utilizar um critério de parada baseado no tamanho da função $f(x_k)$, ou seja, no valor da função em cada iteração $\|f(x_k)\|$. Quando a raiz é mal-condicionada, a diferença entre as iterações sucessivas $\|x_k - x_{k-1}\|$ pode ser pequena, mas a função $f(x)$ pode não estar próxima de zero, indicando que a convergência ainda não foi alcançada.

Portanto, o critério de parada adequado seria baseado na tolerância ϵ_{func} para o valor da função $f(x_k)$ em cada iteração. A iteração é considerada adequada quando o valor absoluto da função $f(x_k)$ for menor do que a tolerância:

$$\|f(x_k)\| < \epsilon_{\text{func}}.$$

Este critério garante que a aproximação x_k seja aceitável quando a função $f(x)$ estiver suficientemente próxima de zero. Isso é importante em problemas mal-condicionados, pois a proximidade de $f(x)$ a zero indica que a solução está sendo bem estimada.

Por outro lado, usar apenas o critério baseado em $\|x_k - x_{k-1}\|$ pode ser enganoso em casos de raízes mal-condicionadas, pois mesmo que as iterações estejam se aproximando umas das outras, a função $f(x)$ pode ainda estar longe de zero, o que significa que a solução não é precisa. A norma da diferença entre duas iterações consecutivas, não fornece informações diretas sobre a proximidade da solução da raiz.

Portanto, ao resolver equações não lineares com raízes mal-condicionadas, é mais apropriado usar a norma da função $\|f(x_k)\|$ como critério de parada para determinar quando a aproximação x_k é adequada. É importante ressaltar que o número máximo de iterações deve ser definido para evitar um loop infinito caso a convergência não seja alcançada.

Questão 5. Qual o significado de taxa convergência quadrática para um método iterativo?

Resolução:

A taxa de convergência quadrática descreve o comportamento da sequência de aproximações gerada pelo método e indica o quão rapidamente a solução se aproxima da solução exata à medida que as iterações progridem.

Em um método iterativo com taxa de convergência quadrática, a sequência de aproximações converge para a solução com uma taxa quadrática. Isso significa que o erro da iteração atual é reduzido aproximadamente ao quadrado em relação ao erro da iteração anterior. Ou seja, a cada iteração, a precisão da aproximação é aproximadamente quadruplicada.

Essa rápida taxa de convergência quadrática permite que a solução seja alcançada com um número relativamente pequeno de iterações. Em comparação, métodos com taxas de convergência linear ou sublinear melhoram a precisão da solução de forma mais lenta.

Métodos com taxa de convergência quadrática são frequentemente encontrados em métodos iterativos avançados para resolver sistemas lineares ou encontrar raízes de equações não lineares. Exemplos incluem o método de Newton para encontrar raízes e o método de Newton-Raphson para resolver sistemas de equações não lineares.

Questão 6. Para a equação

$$f(x) = x^2 - 3x + 2 = 0$$

cada uma das funções abaixo produz um problema de ponto fixo equivalente:

$$\begin{aligned} g_1(x) &= (x^2 + 2)/3 \\ g_2(x) &= \sqrt{(3x - 2)} \\ g_3(x) &= 3 - 2/x \\ g_4(x) &= (x^2 - 2)/(3x - 3) \end{aligned}$$

Analise a propriedade de convergência de cada esquema de ponto-fixo para a raiz $x = 2$.

Resolução:

Para analisar a propriedade de convergência de cada esquema de ponto fixo em relação à raiz $x = 2$ da equação $f(x) = x^2 - 3x + 2 = 0$, é preciso verificar se cada função $g_k(x)$ é uma contração em uma vizinhança da raiz e se o valor absoluto da derivada dessa função é menor que 1 nessa vizinhança. Se essas condições forem satisfeitas, o esquema de ponto fixo correspondente é convergente. Cada função $g_k(x)$ foi analisada.

1. Função $g_1(x) = (x^2 + 2)/3$:

   Para essa função, calcula-se a derivada: $g_1'(x) = (2x)/3$. Em $x = 2$, têm-se $g_1'(2) = 4/3$. Como o valor absoluto dessa derivada é maior que 1, $g_1(x)$ não é uma contração em uma vizinhança de $x = 2$. Portanto, o esquema de ponto fixo correspondente não é convergente para a raiz $x = 2$.

2. Função $g_2(x) = \sqrt{(3x - 2)}$:

   Para essa função, calcula-se a derivada: $g_2'(x) = 3/(2\sqrt{(3x - 2))}$. Em $x = 2$, têm-se $g_2'(2) = 3/2$. Como o valor absoluto dessa derivada é menor que 1, $g_2(x)$ é uma contração em uma vizinhança de $x = 2$. Portanto, o esquema de ponto fixo correspondente é convergente para a raiz $x = 2$.

3. Função $g_3(x) = 3 - 2/x$:

   Essa função é equivalente a $g_3(x) = (3x-2)/x$. Calcula-se a derivada: $g_3'(x) = 8/(x^2)$. Em $x = 2$, têm-se $g_3'(2) = 2$. Como o valor absoluto dessa derivada é maior que 1, $g_3(x)$ não é uma contração em uma vizinhança de $x = 2$. Portanto, o esquema de ponto fixo correspondente não é convergente para a raiz $x = 2$.

4. Função $g_4(x) = (x^2 - 2)/(3x - 3)$:

   Para essa função, podemos calcular a derivada: $g_4'(x) = (6x - 9)/(3x - 3)^2$. Em $x = 2$, temos $g_4'(2) = 0$. Como o valor absoluto dessa derivada é menor que 1, $g_4(x)$ é uma contração em uma vizinhança de $x = 2$. Portanto, o esquema de ponto fixo correspondente é convergente para a raiz $x = 2$.

   Resumindo, têm-se:

1. O esquema de ponto fixo correspondente a $g_1(x)$ não é convergente para a raiz $x = 2$;
2. O esquema de ponto fixo correspondente a $g_2(x)$ é convergente para a raiz $x = 2$;
3. O esquema de ponto fixo correspondente a $g_3(x)$ não é convergente para a raiz $x = 2$;
4. O esquema de ponto fixo correspondente a $g_4(x)$ é convergente para a raiz $x = 2$.

Questão 7. Considere a seguinte função linear $f(x) = x^2 - 2 = 0$

a) Com $x_0 = 1$ como primeira aproximação, qual o valor de x_1 se você usar o método de Newton para resolver este problema?

Resolução:

a) Usando o método de Newton para resolver a equação $f(x) = x^2 - 2 = 0$, com $x_0 = 1$, segue-se os passos:

Calcula-se a derivada de $f(x)$: $f'(x) = 2x$.

Substitui-se $x_0 = 1$ em $f(x)$ e $f'(x)$:

$$f(1) = 1^2 - 2 = -1 \qquad ; \qquad f'(1) = 2(1) = 2$$

Aplica-se a fórmula do método de Newton:

$$x_1 = x_0 - (f(x_0)/f'(x_0)) = 1 - (-1/2) = 1 + 0.5 = 1.5$$

Portanto, usando Newton com $x_0 = 1$ como aproximação, o valor de x_1 é 1.5.

b) Com $x_0 = 1$ e $x_2 = 2$ como aproximações iniciais, qual é o valor de x_3 se você usar o método da secante?

Resolução:

b) Usando o método da secante para resolver a equação $f(x) = x^2 - 2 = 0$, com $x_0 = 1$ e $x_2 = 2$ como aproximações iniciais, pode-se calcular x_3 da seguinte maneira:

Substitui-se $x_0 = 1$ e $x_2 = 2$ em $f(x)$:

$$f(1) = 1^2 - 2 = -1 \qquad ; \qquad f(2) = 2^2 - 2 = 2$$

Aplica-se a fórmula do método da secante:

$$x_3 = x_2 - \left(f(x_2) \cdot \frac{x_2 - x_0}{f(x_2) - f(x_0)} \right) = 2 - \left(2 \cdot \frac{2-1}{2-(-1)} \right) = 2 - \frac{(2 \cdot 1)}{3} = 2 - \frac{2}{3} = \frac{4}{3}$$

Portanto, a Secante com $x_0 = 1$ e $x_2 = 2$ como aproximações, o valor de x_3 é 4/3.

Questão 8. Expresse a iteração de Newton para resolver os sistemas de equações não-lineares.

a)

$$x_1^2 + x_2^2 = 1$$
$$x_1^2 - x_2^2 = 0$$

Resolução:

a) Para resolver o sistema de equações não lineares usando o método de Newton, é preciso chamar as variáveis do sistema de equações de x_1 e x_2.

Pode-se reescrever as equações na forma $f_1(x_1, x_2) = 0$ e $f_2(x-1, x_2) = 0$:

$$f_1(x_1, x_2) = x_1^2 + x_2^2 - 1 \quad ; \quad f_2(x_1, x_2) = x_1^2 - x_2^2$$

A iteração de Newton para resolver o sistema de equações é dada por:

$$\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \end{bmatrix} - \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}^{-1} \begin{bmatrix} f_1(x_1^{(k)}, x_2^{(k)}) \\ f_2(x_1^{(k)}, x_2^{(k)}) \end{bmatrix}$$

Onde as derivadas parciais são calculadas em relação às variáveis x_1 e x_2.

b)

$$x_1^2 + x_1 x_2^3 = 9$$
$$3x_1^2 x_2 - x_2^3 = 4$$

Resolução:

b) Reescrevendo as equações na forma $f_1(x_1, x_2) = 0$ e $f_2(x_1, x_2) = 0$:

$$f_1(x_1, x_2) = x_1^2 + x_1 x_2^3 - 9 \quad ; \quad f_2(x_1, x_2) = 3x_1^2 x_2 - x_2^3 - 4$$

A iteração de Newton para resolver o sistema de equações é dada por:

$$\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \end{bmatrix} - \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}^{-1} \begin{bmatrix} f_1(x_1^{(k)}, x_2^{(k)}) \\ f_2(x_1^{(k)}, x_2^{(k)}) \end{bmatrix}$$

Onde as derivadas parciais são calculadas em relação às variáveis x_1 e x_2.

Questão 9. Implemente os métodos da bisecção, Newton e secante para resolver equações não lineares em uma dimensão, e teste sua implementação para encontrar pelo menos uma raiz para cada uma das equações abaixo. Qual critério de parada será implementado? Qual a taxa de convergência para cada caso?

a) $x^3 - 2x - 5 = 0$

b) $e^{-x} = x$

c) $x \sin(x) = 1$

d) $x^3 - 3x^2 + 3x - 1 = 0$

Resolução:

```python
import math

# Funcao para a equacao (a)
def f_a(x):
    return x**3 - 2*x - 5

# Funcao para a equacao (b)
def f_b(x):
    return math.exp(-x) - x

# Funcao para a equacao (c)
def f_c(x):
    return x * math.sin(x) - 1

# Funcao para a equacao (d)
def f_d(x):
    return x**3 - 3*x**2 + 3*x - 1

# Método da bisseccao
def bissecao(f, a, b, tol, max_iter):
    for i in range(max_iter):
        c = (a + b) / 2
        if abs(f(c)) < tol:
            return c
        elif f(a) * f(c) < 0:
            b = c
        else:
            a = c
    return None

# Método de Newton
def newton(f, df, x0, tol, max_iter):
    x = x0
    for i in range(max_iter):
        x = x - f(x) / df(x)
        if abs(f(x)) < tol:
            return x
    return None

# Método da secante
def secante(f, x0, x1, tol, max_iter):
    x_k_minus_1 = x0
    x_k = x1
    for i in range(max_iter):
        x_k_plus_1 = x_k - f(x_k) * (x_k - x_k_minus_1) / (f(x_k) - f(
            x_k_minus_1))
```

```
46          if abs(f(x_k_plus_1)) < tol:
47              return x_k_plus_1
48          x_k_minus_1 = x_k
49          x_k = x_k_plus_1
50      return None
51
52 # Teste das equacoes
53 tolerancia = 1e-6
54 max_iteracoes = 1000
55
56 # Ajustes do modelo
57 print("Tolerancia: ", tolerancia)
58 print("Número máximo de iteracoes: ", max_iteracoes, "\n")
59
60 # Teste da equacao (a)
61 print("Raiz da equacao com Bissecao (a):", bissecao(f_a, 2, 3, tolerancia,
       max_iteracoes))
62 print("Raiz da equacao com Newton (a):", newton(f_a, lambda x: 3*x**2 - 2,
       2.5, tolerancia, max_iteracoes))
63 print("Raiz da equacao com Secante (a):", secante(f_a, 2, 3, tolerancia,
       max_iteracoes), "\n")
64
65 # Teste da equacao (b)
66 print("Raiz da equacao Bissecao (b):", bissecao(f_b, 0, 1, tolerancia,
       max_iteracoes))
67 print("Raiz da equacao Newton (b):", newton(f_b, lambda x: -math.exp(-x) -
       1, 0.5, tolerancia, max_iteracoes))
68 print("Raiz da equacao Secante (b):", secante(f_b, 0, 1, tolerancia,
       max_iteracoes), "\n")
69
70 # Teste da equacao (c)
71 print("Raiz da equacao Bissecao (c):", bissecao(f_c, 0.1, 1, tolerancia,
       max_iteracoes))
72 print("Raiz da equacao Newton (c):", newton(f_c, lambda x: x*math.cos(x) +
       math.sin(x), 0.5, tolerancia, max_iteracoes))
73 print("Raiz da equacao Secante (c):", secante(f_c, 0.1, 1, tolerancia,
       max_iteracoes), "\n")
74
75 # Teste da equacao (d)
76 print("Raiz da equacao Bissecao (d):", bissecao(f_d, 0, 1, tolerancia,
       max_iteracoes))
77 print("Raiz da equacao Newton (d):", newton(f_d, lambda x: 3*x**2 - 6*x +
       3, 0.5, tolerancia, max_iteracoes))
78 print("Raiz da equacao Secante (d):", secante(f_d, 0, 1, tolerancia,
       max_iteracoes))
```

Outputs:

```
Tolerância: 1e-06
Número máximo de iterações: 1000

Raiz da equação com Bissecao (a): 2.0945515632629395
Raiz da equação com Newton (a): 2.094551481550247
Raiz da equação com Secante (a): 2.094551481227599

Raiz da equação Bissecao (b): 0.567143440246582
Raiz da equação Newton (b): 0.5671431650348622
Raiz da equação Secante (b): 0.5671433066049633

Raiz da equação Bissecao (c): None
Raiz da equação Newton (c): 1.1141571408713682
Raiz da equação Secante (c): 1.1141566248205137

Raiz da equação Bissecao (d): 0.9921875
Raiz da equação Newton (d): 0.9913292350416106
Raiz da equação Secante (d): 1.0
```

Questão 10. Escreva um programa para resolver o sistema de equações não lineares abaixo:

$$16x^4 + 16y^4 + z^4 = 16$$
$$x^2 + y^2 + z^2 = 3$$
$$x^3 - y = 0$$

usando método de Newton. Você pode resolver o sistema linear resultando em cada iteração através de uma rotina de biblioteca ou uma rotina de implementação própria. Como aproximação inicial, use o vetor unitário.

Resolução:

```python
import numpy as np

# Define as funcoes do sistema de equacoes
def f(x, y, z):
    return np.array([
        16*x**4 + 16*y**4 + z**4 - 16,
        x**2 + y**2 + z**2 - 3,
        x**3 - y
    ])

# Define as derivadas parciais do sistema de equacoes
def jacobian(x, y, z):
    return np.array([
```

```
        [64*x**3, 64*y**3, 4*z**3],
        [2*x, 2*y, 2*z],
        [3*x**2, -1, 0]
    ])

def newton_method(initial_guess, tolerance=1e-6, max_iterations=100):
    x, y, z = initial_guess[0], initial_guess[1], initial_guess[2]

    for i in range(max_iterations):
        # Avalia o sistema de equacoes no ponto atual
        f_val = f(x, y, z)
        # Verifica a convergencia
        if np.linalg.norm(f_val) < tolerance:
            break

        # Calcula o Jacobiano
        jacobian_val = jacobian(x, y, z)

        # Resolve o sistema linear J*dx = -f
        dx = np.linalg.solve(jacobian_val, -f_val)

        # Atualiza o vetor de aproximacao
        x += dx[0]
        y += dx[1]
        z += dx[2]

    return x, y, z

# Aproximacao inicial usando vetor unitário
initial_guess = np.array([1.0, 1.0, 1.0]) / np.sqrt(3)

# Executa o método de Newton
x_approx, y_approx, z_approx = newton_method(initial_guess)

# Imprime os resultados
print(f"Resultado aproximado:")
print(f"x = {x_approx:.6f}, y = {y_approx:.6f}, z = {z_approx:.6f}")
```

Outputs:

```
Resultado aproximado:
x = 0.877966, y = 0.676757, z = 1.330855
```

Questão 11. Dada a função função $f(x) = \arctan(x)$, faça:

a) Verifique a convergência do método de Newton ao usar $x_0 = 10$ como aproximação inicial.

Resolução:

```python
import math

def f(x):
    return math.atan(x)

def f_prime(x):
    return 1 / (1 + x**2)

def newton_method(f, f_prime, x0, tolerance=1e-8, max_iterations=100):
    x = x0
    for i in range(max_iterations):
        fx = f(x)
        fx_prime = f_prime(x)

        if abs(fx_prime) < tolerance:
            print("Convergido: Derivada muito próxima de zero.")
            return x, i

        x_next = x - fx / fx_prime

        if abs(x_next - x) < tolerance:
            print("Convergido: Mudanca em x muito pequena.")
            return x_next, i

        x = x_next

    print("Falhou em convergir dentro do número máximo de iteracoes.")
    return x, max_iterations

# Teste com aproximacao inicial: x0 = 10
x0 = 10
root, iterations = newton_method(f, f_prime, x0)

print("Raiz aproximada:", root)
print("Número de iteracoes:", iterations)
print("Valor do arctan(raiz):", f(root))
```

Outputs:

```
Convergido: Derivada muito próxima de zero.
Raiz aproximada: 29892.32073900695
Número de iterações: 2
Valor do arctan(raiz): 1.5707628733869676
```

b) Modifique o algoritmo adotando o esquema de globalização line search e compare os resultados. Use os algoritmos discutidos em aula.

Resolução:

```python
import math

def f(x):
    return math.atan(x)

def f_prime(x):
    return 1 / (1 + x**2)

def newton_method(f, f_prime, x0, tolerance=1e-8, max_iterations=100):
    x = x0
    for i in range(max_iterations):
        fx = f(x)
        fx_prime = f_prime(x)

        if abs(fx_prime) < tolerance:
            print("Convergido: Derivada muito próxima de zero.")
            return x, i

        x_next = x - fx / fx_prime

        if abs(x_next - x) < tolerance:
            print("Convergido: Mudanca em x muito pequena.")
            return x_next, i

        x = x_next

    print("Falha ao convergir dentro do número máximo de iteracoes.")
    return x, max_iterations

def line_search(f, f_prime, x, direction, alpha_max=1.0, tol=1e-8):
    phi = (1 + math.sqrt(5)) / 2  # Valor da proporcao áurea

    alpha_low = 0
    alpha_high = alpha_max

```

```
36      a = alpha_low + (phi - 1) * (alpha_high - alpha_low)
37      b = alpha_low + phi * (alpha_high - alpha_low)
38
39      f_a = f(x - a * direction)
40      f_b = f(x - b * direction)
41
42      while alpha_high - alpha_low > tol:
43          if f_a > f_b:
44              alpha_low = a
45              a = b
46              f_a = f_b
47              b = alpha_low + phi * (alpha_high - alpha_low)
48              f_b = f(x - b * direction)
49          else:
50              alpha_high = b
51              b = a
52              f_b = f_a
53              a = alpha_low + (phi - 1) * (alpha_high - alpha_low)
54              f_a = f(x - a * direction)
55
56      return (alpha_low + alpha_high) / 2
57
58  def newton_method_with_line_search(f, f_prime, x0, tolerance=1e-8,
        max_iterations=100):
59      x = x0
60      for i in range(max_iterations):
61          fx = f(x)
62          fx_prime = f_prime(x)
63
64          if abs(fx_prime) < tolerance:
65              print("Convergido: Derivada muito próxima de zero.")
66              return x, i
67
68          step_direction = fx_prime
69          step_size = line_search(f, f_prime, x, step_direction)
70
71          x_next = x - step_size * step_direction
72
73          if abs(x_next - x) < tolerance:
74              print("Convergido: Mudanca em x muito pequena.")
75              return x_next, i
76
77          x = x_next
78
79      print("Falha ao convergir dentro do número máximo de iteracoes.")
80      return x, max_iterations
81
```

```
82 # Test with initial approximation x0 = 10
83 x0 = 10
84 root_newton, iterations_newton = newton_method(f, f_prime, x0)
85 root_line_search, iterations_line_search = newton_method_with_line_search(f
       , f_prime, x0)
86
87 print("Raiz aproximada com método de Newton:", root_newton)
88 print("Número de iteracpes com método de Newton:", iterations_newton)
89 print("Valor do arctan(root) com método de Newton:", f(root_newton), "\n")
90
91 print("Raiz aproximada com método de Newton e Line search:")
92 print("Raiz aproximada com Line search:", root_line_search)
93 print("Número de iteracoes com Line search:", iterations_line_search)
94 print("Valor de arctan(root) com Line search:", f(root_line_search))
```

Outputs:

```
Convergido: Derivada muito próxima de zero.
Falha ao convergir dentro do número máximo de iterações.
Raiz aproximada com método de Newton: 29892.32073900695
Número de iterações com método de Newton: 2
Valor do arctan(root) com método de Newton: 1.5707628733869676

Raiz aproximada com método de Newton e Line search:
Raiz aproximada com Line search: 8.049079037538899
Número de iterações com Line search: 100
Valor de arctan(root) com Line search: 1.4471918606577918
```

APÊNDICE A – Códigos da Atividade

Abaixo são apresentados os códigos realizados, todos desenvolvidos e testados na plataforma `https://www.onlinegdb.com/`. Abaixo seguem os links das questões:

- **Questão 9:** `https://onlinegdb.com/y0fUleFXl`
- **Questão 10:** `https://onlinegdb.com/h3UlY0Ox1`
- **Questão 11 (letra a):** `https://onlinegdb.com/r7-AR96b1`
- **Questão 11 (letra b):** `https://onlinegdb.com/FDVmfVoHm`