ANÁLISE NUMÉRICA I -
Equações Lineares

UAlg FCT
UNIVERSIDADE DO ALGARVE
FACULDADE DE CIÊNCIAS E TECNOLOGIA

Curso:
Licenciatura em Engenharia Informática

Unidade Curricular:
Análise Numérica I

Docente:
Hermenegildo Borges de Oliveira

Realizado por:
Daniel Maryna (64611)
Miguel Silva (80072)
Francisco Nunes (80061)
Brandon Mejia (79261)

10/2024

CONTEÚDO

INTRODUÇÃO

Neste relatório, apresentamos a implementação e análise de três métodos numéricos para o cálculo de raízes de equações não lineares: o **Método da Bissecção**, o **Método de Newton**, e o **Método da Secante**. O problema de encontrar as raízes de equações não lineares é central em várias áreas da ciência e da engenharia, sendo um dos desafios mais comuns em cálculos numéricos.

O objetivo principal deste trabalho é comparar a eficácia e a precisão desses três métodos em diferentes situações. Cada método tem suas vantagens e limitações, que variam de acordo com a função analisada e a natureza da equação. Implementámos cada um dos métodos em Python, com foco na precisão das soluções e na eficiência computacional.

Para o **Método da Bissecção**, estabelecemos uma tolerância relativa inferior a 10^{-5} e permitimos até 20 iterações para garantir a convergência. Já para os **Métodos de Newton e Secante**, aplicámos uma tolerância relativa inferior a 10^{-9} com um máximo de 10 iterações, uma vez que estes métodos têm uma taxa de convergência mais rápida sob condições adequadas.

Ao longo deste relatório, apresentamos os conceitos teóricos que sustentam cada método, a implementação prática e os resultados obtidos a partir de exemplos específicos de equações não lineares.

FUNDAMENTAÇÃO TEÓRICA

A resolução de equações não lineares é um dos problemas fundamentais em análise numérica, envolvendo funções cuja solução não pode ser obtida de forma explícita. Uma equação não linear pode ser descrita genericamente como:

$$f(x) = 0$$

onde $f(x)$ é uma função real de uma variável real. Encontrar as raízes desta equação significa determinar os valores de x que tornam a equação verdadeira. No entanto, para muitas funções não é possível encontrar essas raízes de forma analítica, tornando necessária a aplicação de métodos numéricos.

Definição de Raízes de Equações Não Lineares

Consideramos $f: D \subseteq \mathbb{R} \to \mathbb{R}$, onde D é um subconjunto de $\mathbb{R}$. Diz-se que $x = a \in D$ é uma raiz da equação $f(x) = 0$ se, ao substituirmos x por a, a equação for satisfeita, ou seja, $f(a) = 0$. Alternativamente, $x = a$ também é chamado de zero da função $f(x)$.

Métodos Numéricos

A resolução numérica de equações não lineares envolve quatro etapas principais:

1. **Localização das Raízes:** Determinação de um intervalo $[a, b]$ fora do qual não existem raízes da equação $f(x) = 0$;

2. **Separação das Raízes:** Determinação de intervalos $[a_i, b_i] \subset [a, b]$ de modo que em cada intervalo exista uma única raiz;

3. **Cálculo Aproximado de Cada Raiz:** Aplicação de um algoritmo que possibilite calcular um valor aproximado de cada raiz;

4. **Cálculo do Erro:** Quantificação do erro cometido no cálculo da raiz aproximada.

Métodos Usados

Neste trabalho, três métodos numéricos foram implementados para o cálculo de raízes:

- **Método da Bissecção:** Um método simples e robusto que garante a convergência, mas pode ser relativamente lento. Ele divide iterativamente o intervalo que contém a raiz em dois, escolhendo o SUB intervalo onde ocorre a mudança de sinal da função.

- **Método de Newton:** Um método mais rápido, que usa a derivada da função para encontrar as raízes. Este método tem uma taxa de convergência quadrática, mas requer uma boa estimativa inicial e que a derivada da função não seja nula.

- **Método da Secante:** Uma variação do Método de Newton que não requer o cálculo explícito da derivada. Usa uma aproximação numérica para a derivada, o que torna o método mais eficiente em algumas situações, mas menos estável.

Esses métodos, apesar de diferentes, são amplamente utilizados devido à sua eficiência em situações onde soluções analíticas são difíceis de obter.

FUNCIONAMENTO DOS PROGRAMAS

Nesta secção, apresentamos a implementação dos três métodos numéricos usados para o cálculo de raízes de equações não lineares: **Bissecção**, **Newton** e **Secante**. Cada método foi programado em Python, garantindo os limites de iterações e a precisão conforme solicitado nos requisitos.

Método da Bissecção

O Método da Bissecção é baseado no Teorema do Valor Intermédio, que afirma que, se uma função contínua $f(x)$ assume valores de sinais opostos nos extremos de um intervalo, então existe pelo menos uma raiz nesse intervalo. O algoritmo divide sucessivamente o intervalo ao meio, reduzindo o intervalo até que uma aproximação da raiz seja encontrada.

Critérios de Implementação:

- Tolerância relativa: 10^{-5};
- Número máximo de iterações: 20.

Implementação: O algoritmo começa com dois pontos a e b onde $f(a)$ e $f(b)$ têm sinais opostos. Em cada iteração, o intervalo é dividido ao meio, e a raiz é localizada no SUB intervalo onde ocorre a mudança de sinal.

```
def metodo_da_bissecao(equacao:symbols, intervalo:tuple , iteracoes_erro:tuple):
      if not teorema_bolzano(equacao, intervalo):
      return "Dentro do intervalo não existe um e só um zero, garantidamente.
T.Bolzano não se verifica"
      a, b = intervalo
      numero_de_iteracoes , tolerancia_de_erro = iteracoes_erro
      for i in range(numero_de_iteracoes-1): # -1 pq o range começa em 0.
            print(f"-----------------------------------------------")
            print(f"-> Iteracao numero {i+1}:")
            c = a+((b-a)/2) # Ponto médio -> equivalente a (a+b)/2x
            print(f"\tPonto a: {a}\n\tPonto b: {b}\n\tPonto médio c: {c}")
      if equacao.subs('x',c) == 0 or (abs(b-a)/2) < tolerancia_de_erro
            print(f"        Encontrámos raiz. <- SUCESSO")
            return c
      a, b = get_new_interval(a, b, c, equacao)
            print(f"Nao encontrámos raiz. Definimos novo intervalo: [{a} , {b}]")
            print(f"-----------------------------------------------")
            print(f"")
      return f"Atingimos o número máximo de {numero_de_iteracoes} iterações sem
encontrar a raiz com a tolerância especificada."
```

Código 1 - Método da Bissecção

Este código executa o Método da Bissecção, garantindo a precisão exigida, e interrompe a execução quando a diferença entre os extremos do intervalo é menor que a tolerância ou quando o número máximo de iterações é atingido.

Método de Newton

O Método de Newton, ou método de Newton-Raphson, é um algoritmo de iteração rápida, baseado na aproximação da função $f(x)$ por uma reta tangente. O ponto onde a reta tangente intersecta o eixo x fornece uma nova aproximação para a raiz. A fórmula iterativa usada é:

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

Critérios de Implementação:

- Tolerância relativa: 10^{-9};
- Número máximo de iterações: 10.

Implementação: O código verifica a convergência e a estabilidade da solução para evitar divisões por zero ou oscilações.

```
def metodo_de_newton(equacao:symbols, intervalo:tuple , iteracoes_erro:tuple):
	numero_de_iteracoes, tolerancia_de_erro = iteracoes_erro
	primeira_derivada = diff(equacao, 'x')
	k = 1
#Pedir aproximação inicial
	aprx_ini = escolher_ponto_favoravel(equacao, intervalo)
	r = 0 # raiz ou aproximacao da raiz
	print("----------------------------")
	print(f'Aproximação Dada: {aprx_ini}')
while k <= numero_de_iteracoes:
	r = aprx_ini - equacao.subs('x', aprx_ini) / primeira_derivada.subs('x',
aprx_ini) # Calculo do Rk
	print(f'\t {k}a aproximação: {float(r)}')
	if abs(r - aprx_ini) < tolerancia_de_erro:
		print(f"\t\tEncontrámos raiz! <- SUCESSO")
		return r
		k += 1
		aprx_ini = r
return f"Atingimos o número máximo de {numero_de_iteracoes} iterações sem encontrar a
raiz com a tolerância especificada."
```

Código 2 - Método de Newton

Aqui, o método utiliza a função e sua derivada para calcular as aproximações iterativamente, até atingir a tolerância ou o número máximo de iterações.

Método da Secante

O Método da Secante é uma modificação do Método de Newton que elimina a necessidade de calcular a derivada da função. Em vez disso, aproxima a derivada usando a diferença entre dois pontos. A fórmula iterativa é:

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

Critérios de Implementação:

- Tolerância relativa: 10^{-9};
- Número máximo de iterações: 10.

```
def metodo_da_secante(equacao:symbols, intervalo, iteracoes_erro:tuple):
	if not teorema_bolzano(equacao, intervalo):
		return "Não existe zero no intervalo"
a, b = intervalo #r0 e r1
numero_de_iteracoes, tolerancia_de_erro = iteracoes_erro # No e Tol
	k = 2 # Contador de iterações
	y0 = equacao.subs('x', a)
	y1 = equacao.subs('x', b)
r = 0 # raiz ou aproximacao da raiz
print(f"Intervalo / Aproximações iniciais: {a} , {b}")
	while k <= numero_de_iteracoes:
		r = b - (y1 * (b - a)) / (y1 - y0) # Calculo do Rk
		print(f'-------------------------------------------')
		print(f'\t {k}a aproximação: {float(r)}')
	if abs(r - b) < tolerancia_de_erro:
		print(f"\t\tEncontrámos raiz! <- SUCESSO")
		return r #Deu certo
	k+=1
	a = b
	y0 = y1
	b = r
	y1 = equacao.subs('x', r)
return f"Atingimos o número máximo de {numero_de_iteracoes} iterações sem encontrar a
raiz com a tolerância especificada."
```

Código 3 - Método da Secante

Neste código, o método da secante aproxima a derivada e itera até encontrar a raiz com a tolerância especificada ou até atingir o limite de iterações.

FLUXO DE EXECUÇÃO E RESULTADOS

Método da Secante

O **Método da Secante** foi testado com diversas funções e intervalos iniciais, conforme os prints de execução a seguir. Este método utiliza duas aproximações iniciais para calcular a raiz, sem a necessidade de derivadas, o que o torna mais eficiente em funções onde a derivada é complexa.

Fluxo de Execução:

- O programa solicita ao utilizador a função e o intervalo de entrada;
- As aproximações são calculadas iterativamente até que a tolerância ou o número máximo de iterações seja atingido.

Resultados:

```
Qual é a equação que pretende encontrar o zero? Ex: x^2+x-6
cos(x)-x
Em que intervalo? Ex: -5,5 [Sem parentesis]
0 1
1) Metodo da Bisseção
2) Metodo de Newton
3) Metodo da Secante
Qual o método que deseja aplicar?
3
Intervalo / Aproximações iniciais: 0.0 , 1.0
-----------------------------------------------
        2a aproximação: 0.6850733573260451
-----------------------------------------------
        3a aproximação: 0.736298997613654
-----------------------------------------------
        4a aproximação: 0.7391193619116293
-----------------------------------------------
        5a aproximação: 0.7390851121274639
-----------------------------------------------
        6a aproximação: 0.7390851332150012
-----------------------------------------------
        7a aproximação: 0.7390851332151607
              Encontrámos raiz! <- SUCESSO
O zero da equação--> -x + cos(x) é o ponto x = 0.739085133215161
```

Ilustração 1 - Método da Secante

Na **Ilustração 1 - Método da Secante**, a função testada é $\cos(x) - x$, com um intervalo inicial de $[0.0, 1.0]$. A partir das aproximações iniciais, o método convergiu em 7 iterações para a raiz $x = 0.7390851332151607$. Este valor é a solução da equação $\cos(x) - x = 0$, uma equação clássica para a qual o método da secante é bastante eficiente, pois não requer o cálculo de derivadas, apenas avaliações da função.

Mais Resultados:

```
Qual é a equação que pretende encontrar o zero? Ex: x^2+x-6
x^4-16
Em que intervalo? Ex: -5,5 [Sem parentesis]
-3 1
1) Metodo da Bisseção
2) Metodo de Newton
3) Metodo da Secante
Qual o método que deseja aplicar?
3
Intervalo / Aproximações iniciais: -3.0 , 1.0
------------------------------------------------
         2a aproximação: 0.25
------------------------------------------------
         3a aproximação: 12.294117647058824
------------------------------------------------
         4a aproximação: 0.25843333874548513
------------------------------------------------
         5a aproximação: 0.2668604805780974
------------------------------------------------
         6a aproximação: 220.91164506580043
------------------------------------------------
         7a aproximação: 0.2668619624188011
------------------------------------------------
         8a aproximação: 0.26686344425951986
------------------------------------------------
         9a aproximação: 210.67312508732803
------------------------------------------------
         10a aproximação: 0.26686515271794065
O zero da equação--> x^4 - 16 é o ponto x = Atingimos o número máximo de 10 iterações sem encontrar a raiz com a tolerância especificada.
```

Ilustração 2 - Método da Secante

Na **Ilustração 2**, a função utilizada foi $x^4 - 16$, com um intervalo inicial de $[-3.0, 1.0]$. Durante as iterações, o método apresentou uma oscilação significativa entre os valores das aproximações, o que sugere que o método estava enfrentando dificuldades para encontrar uma raiz estável dentro do número de iterações permitidas. Após 10 iterações, o método atingiu o número máximo de iterações sem encontrar a raiz com a precisão especificada, retornando uma aproximação de $x = 0.26686515271794065$.

Método da Bissecção

O **Método da Bissecção** foi também testado com várias funções e intervalos. Este método é mais robusto, porém, mais lento em comparação com a secante, uma vez que requer mais iterações para convergir para uma raiz.

Fluxo de Execução:

- O programa solicita a função e o intervalo de entrada.
- Em cada iteração, o intervalo é dividido ao meio, e o SUB intervalo contendo a raiz é selecionado com base no Teorema de Bolzano.

Resultados:

```
------------------------------------------------------------------------
Qual é a equação que pretende encontrar o zero? Ex: x^2+x-6
ln(x)-1
Em que intervalo? Ex: -5,5 [Sem parentesis]
2 3
1) Metodo da Bisseção
2) Metodo de Newton
3) Metodo da Secante
Qual o método que deseja aplicar?
1
------------------------------------------------
-> Iteracao numero 1:
        Ponto a: 2.0
        Ponto b: 3.0
        Ponto médio c: 2.5
        Nao encontrámos raiz. Definimos novo intervalo: [2.5 , 3.0]
------------------------------------------------

------------------------------------------------
-> Iteracao numero 2:
        Ponto a: 2.5
        Ponto b: 3.0
        Ponto médio c: 2.75
        Nao encontrámos raiz. Definimos novo intervalo: [2.5 , 2.75]
------------------------------------------------

------------------------------------------------
-> Iteracao numero 3:
        Ponto a: 2.5
        Ponto b: 2.75
        Ponto médio c: 2.625
        Nao encontrámos raiz. Definimos novo intervalo: [2.625 , 2.75]
------------------------------------------------

------------------------------------------------
-> Iteracao numero 4:
        Ponto a: 2.625
        Ponto b: 2.75
        Ponto médio c: 2.6875
        Nao encontrámos raiz. Definimos novo intervalo: [2.6875 , 2.75]
------------------------------------------------

------------------------------------------------
-> Iteracao numero 5:
        Ponto a: 2.6875
        Ponto b: 2.75
        Ponto médio c: 2.71875
        Nao encontrámos raiz. Definimos novo intervalo: [2.6875 , 2.71875]
------------------------------------------------
```

Ilustração 3 - Método da Bissecção

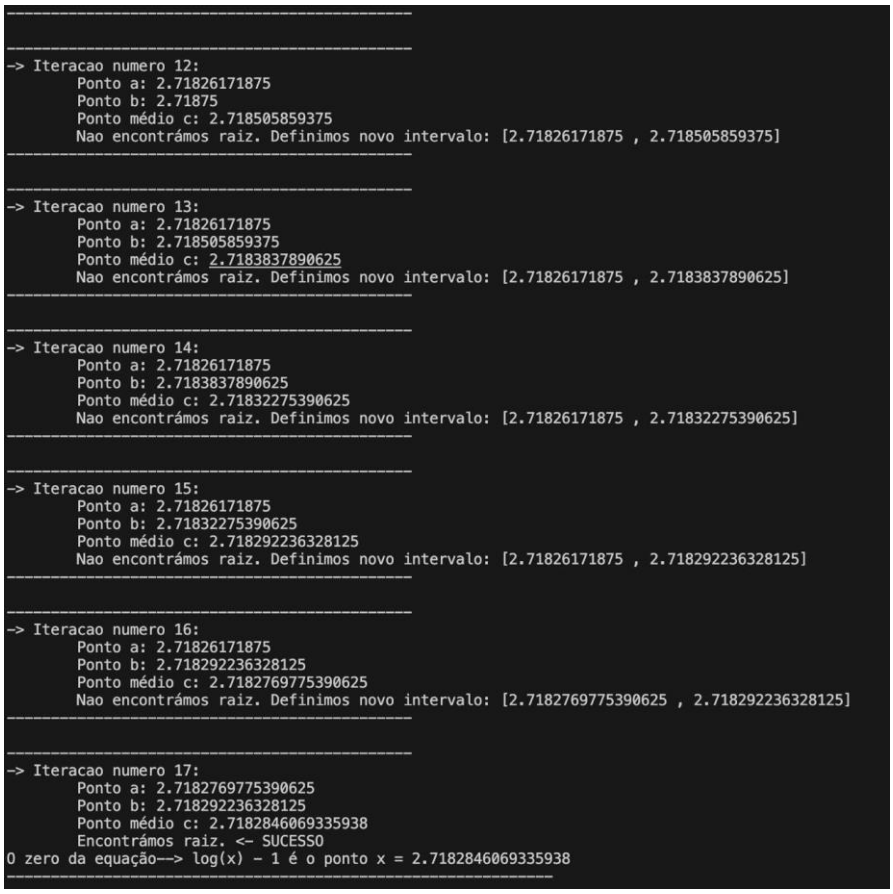

Ilustração 4 - Conclusão da Ilustração 3

Na **Ilustração 3** e **Ilustração 4**, a função usada foi $\ln(x) - 1$, com um intervalo inicial de $[2, 3]$. Durante a execução, o programa realizou várias iterações, ajustando o intervalo progressivamente até encontrar uma aproximação para a raiz. No final da execução (17 iterações), o método convergiu para $x = 2.7182769775390625$, que é a aproximação de e, o número de Euler, solução da equação $\ln(x) = 1$. A precisão final foi atingida, confirmando o sucesso do método.

Mais Resultados:

```
Qual é a equação que pretende encontrar o zero? Ex: x^2+x-6
cos(x)-x
Em que intervalo? Ex: -5,5 [Sem parentesis]
-5 5
1) Metodo da Bisseção
2) Metodo de Newton
3) Metodo da Secante
Qual o método que deseja aplicar?
1
------------------------------------------------
-> Iteracao numero 1:
        Ponto a: -5.0
        Ponto b: 5.0
        Ponto médio c: 0.0
        Nao encontrámos raiz. Definimos novo intervalo: [0.0 , 5.0]
------------------------------------------------

------------------------------------------------
-> Iteracao numero 2:
        Ponto a: 0.0
        Ponto b: 5.0
        Ponto médio c: 2.5
        Nao encontrámos raiz. Definimos novo intervalo: [0.0 , 2.5]
------------------------------------------------

------------------------------------------------
-> Iteracao numero 3:
        Ponto a: 0.0
        Ponto b: 2.5
        Ponto médio c: 1.25
        Nao encontrámos raiz. Definimos novo intervalo: [0.0 , 1.25]
------------------------------------------------

------------------------------------------------
-> Iteracao numero 4:
        Ponto a: 0.0
        Ponto b: 1.25
        Ponto médio c: 0.625
        Nao encontrámos raiz. Definimos novo intervalo: [0.625 , 1.25]
------------------------------------------------
```

Ilustração 5 - Método da Bissecção

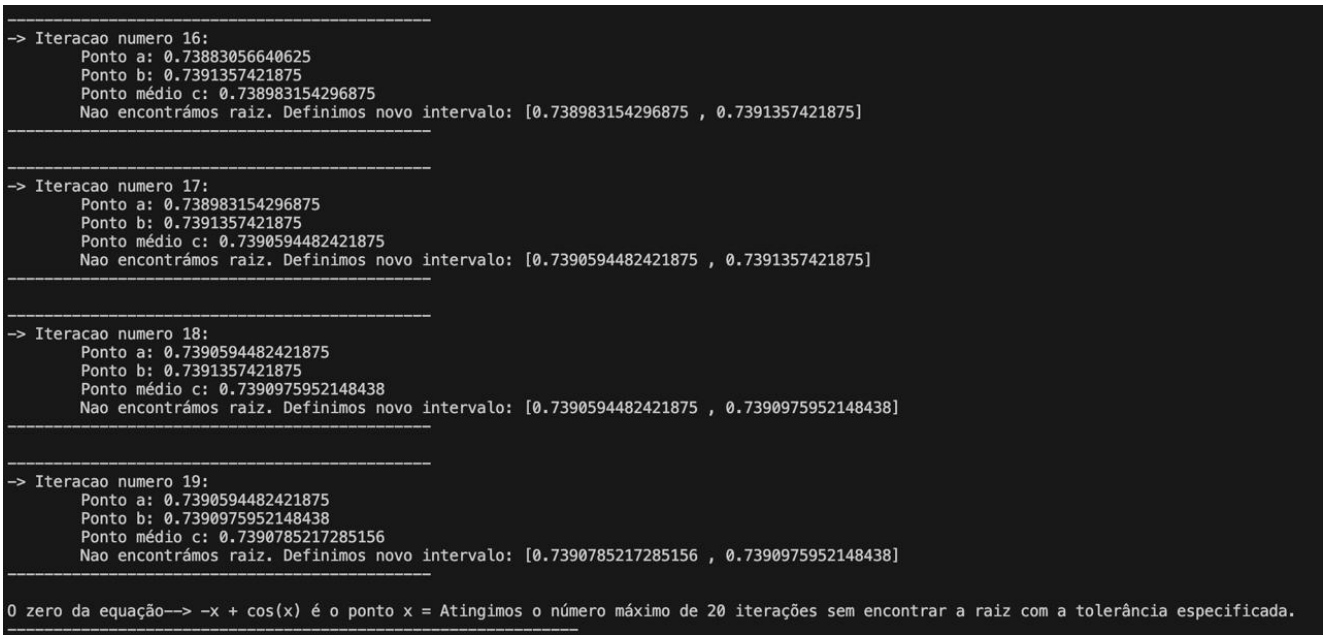

Ilustração 6 - Conclusão da Ilustração 5

Na **Ilustração 5** e **Ilustração 6**, a função testada foi $\cos(x) - x$, com um intervalo inicial de $[-5, 5]$. O método começou a reduzir o intervalo de forma consistente, conforme o esperado. Porém, após 19 iterações, o método não encontrou a raiz com a precisão especificada. A aproximação final foi $x = 0.7390975952148438$, que é uma boa aproximação da solução real, mas o método atingiu o limite de iterações sem alcançar a tolerância desejada, resultando em uma interrupção sem convergência completa.

Método de Newton

O **Método de Newton** é muito eficaz quando a função é suave e a aproximação inicial está próxima da raiz. Ele utiliza a fórmula iterativa, apresentada anteriormente para atualizar as aproximações sucessivas até atingir uma precisão desejada.

Fluxo de Execução:

- O programa solicita a equação e o intervalo.
- As iterações continuam até que a tolerância seja atingida ou até que o número máximo de iterações seja alcançado.

Resultados:

```
francisconunes@unknown424f72a75792 lab02_final % /usr/bin/python3 /Users/francisconunes/Documents/AN/lab02_final/Main.py
------------------------------------------------------------
Qual é a equação que pretende encontrar o zero? Ex: x^2+x-6
x^3-x-2
Em que intervalo? Ex: -5,5 [Sem parentesis]
1 2
1) Metodo da Bisseção
2) Metodo de Newton
3) Metodo da Secante
Qual o método que deseja aplicar?
2
------------------------------
Aproximação Dada: 1.0
         1a aproximação: 2.0
         2a aproximação: 1.6363636363636362
         3a aproximação: 1.5303920521311825
         4a aproximação: 1.5214414651351367
         5a aproximação: 1.521379709733148
         6a aproximação: 1.5213797068045676
                Encontrámos raiz! <- SUCESSO
O zero da equação--> x^3 - x - 2 é o ponto x = 1.52137970680457
------------------------------------------------------------
------------------------------------------------------------
```

Ilustração 7 - Método de Newton

Na Ilustração 7, a equação testada foi $x^3 - x - 2$, e a aproximação inicial foi $x_0 = 1$. O Método de Newton, devido à sua rápida taxa de convergência, encontrou a raiz da equação em apenas 5 iterações e convergiu para a raiz $x = 1.52137970680457$ que é solução da equação supramencionada. O resultado foi obtido com uma alta precisão em poucas iterações, confirmando a eficiência do Método de Newton quando se trata de equações suaves.

CONCLUSÃO

No presente estudo, analisámos três métodos numéricos amplamente utilizados para encontrar as raízes de equações não lineares: **Bissecção**, **Newton** e **Secante**. Cada um dos métodos tem características particulares que os tornam mais adequados para diferentes tipos de problemas.

O **Método da Bissecção** mostrou-se extremamente robusto e confiável, garantindo convergência desde que a função seja contínua e haja uma mudança de sinal no intervalo inicial. No entanto, a sua desvantagem reside na lentidão da convergência, especialmente em comparação com os outros métodos, uma vez que a divisão do intervalo ao meio a cada iteração reduz o erro de forma mais lenta.

Por outro lado, o **Método de Newton** destacou-se pela sua eficiência, apresentando uma taxa de convergência quadrática quando a aproximação inicial está próxima da raiz e a derivada da função é suave. Contudo, este método requer o cálculo da derivada da função, o que pode ser uma limitação em problemas onde a derivada não está disponível ou é difícil de calcular. Além disso, é sensível à escolha da aproximação inicial, podendo divergir em certas situações.

O **Método da Secante** provou ser uma boa alternativa ao Método de Newton, eliminando a necessidade do cálculo da derivada. Embora seja ligeiramente menos eficiente que Newton em termos de convergência, a Secante é mais flexível, especialmente para funções em que a derivada não é facilmente acessível.

Em termos de eficiência geral, o **Método de Newton** apresentou-se como o mais rápido, desde que as condições para a sua aplicação fossem satisfeitas. Já o **Método da Secante** destacou-se pela sua simplicidade e eficiência, tornando-se uma escolha interessante quando o cálculo da derivada não é viável. O **Método da Bissecção** manteve-se uma escolha robusta, porém mais lenta.

Desta forma, a escolha do método mais adequado depende fortemente da natureza do problema e da informação disponível sobre a função. Para funções suaves, o **Método de Newton** é a melhor escolha. Em casos onde o cálculo da derivada é complexo, a **Secante** surge como uma solução prática. Finalmente, a **Bissecção** permanece uma escolha confiável para garantir uma solução quando se conhece pouco sobre a função.

REFERÊNCIAS

1. **Oliveira, H. B.** (2023). Introdução ao Cálculo Infinitesimal. Universidade de Lisboa, 3ª Edição.
2. **Burden, R. L., & Faires, J. D.** (2015). Análise Numérica. Cengage Learning, 9ª Edição.
3. **Chapra, S. C., & Canale, R. P.** (2015). Métodos Numéricos para Engenharia. McGraw-Hill, 7ª Edição.
4. **Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P.** (2007). Numerical Recipes: The Art of Scientific Computing. Cambridge University Press, 3ª Edição.