UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE ESTATÍSTICA E INVESTIGAÇÃO OPERACIONAL

Métodos numéricos de resolução de equações não lineares

Rodrigo de Oliveira Correia

Mestrado em Matemática Aplicada à Economia e Gestão

Trabalho de Projeto orientado por:
Cristian Angel Barbarosie

2024

Agradecimentos

Desejo exprimir os meus agradecimentos a todos aqueles que, de alguma forma, permitiram que esta tese se concretizasse.

Em primeiro lugar quero agradecer à minha família que me apoiou sempre e que me deu as ferramentas necessárias para concretizar este meu sonho, especialmente a minha mãe que se sentou durante horas comigo a ajudar-me nos meus estudos.

A todos os professores que participaram na minha formação, principalmente:

- Explicadora Carla Dias, acompanhou-me durante os meus 3 anos de secundário e ajudou-me a perceber que a matemática era a minha paixão e que era algo que queria desenvolver ainda mais no meu futuro;
- Explicador Manuel Pedro, um dos homens mais sábios que alguma vez tive o prazer de conhecer e a única pessoa cujo nível de paixão pela matemática se igualava ao meu, não foi apenas um explicador, mas sim um amigo;
- Professora Teresa Alpuim, foi alguém que me impactou bastante porque sempre me apoiou em todas as minhas dificuldades e sempre se interessou pelos meus objetivos, foi uma professora que considero um exemplo a seguir para todos os professores;
- Professor e orientador Cristian Barbarosie, por ter tido a paciência necessária para me ajudar em todo o processo.

Obrigado a todos.

Resumo

Este trabalho foi realizado no âmbito da dissertação para o Mestrado de Matemática Aplicada à Economia e Gestão, tem como principal objetivo mensurar o desempenho de cinco métodos numéricos iterativos de resolução de equações não lineares implementados no Python. Os métodos utilizados são:

- Bissecção;
- Falsa Posição;
- Ponto Fixo;
- Newton-Raphson;
- Secante.

Foram solucionadas diferentes equações não lineares, buscando, ao longo das análises, comparar a eficiência de cada método, no que concerne ao número de iterações e tempo de processamento necessário para se atingir uma solução aceitável. No fim, é apresentada uma classificação dos métodos de acordo com as métricas utilizadas.

Palavras-chave: Métodos numéricos, Convergência, Python, Métodos iterativos, Equações não lineares.

Abstract

This work was carried out as part of the dissertation for the Master of Mathematics Applied to Economics and Management. Its main objective is to measure the performance of five iterative numerical methods for solving non-linear equations implemented in Python. The methods used are:

- Bisection;
- False Position;
- Fixed point;
- Newton-Raphson;
- Secant.

Different nonlinear equations were solved, seeking, throughout the analyses, to compare the efficiency of each method, with regard to the number of iterations and processing time required to reach an acceptable solution. At the end, a classification of the methods is presented according to the metrics used.

Keywords: Numerical methods, Convergence, Python, Iterative methods, Nonlinear equations.

Índice

Capítulo 1: Introdução..... 1
Capítulo 2: Conceitos preliminares..... 2
2.1 Teoremas e conceitos básicos da Análise Matemática..... 2
2.2 Conceitos básicos da teoria dos erros..... 3
2.2.1 Erros na Fase de Modelação..... 4
2.2.2 Erros na Fase de Resolução..... 4
2.2.3 Erros Absolutos e Relativos..... 4
2.2.4 Erro de Arredondamento..... 5
2.2.5 Erro de Truncamento..... 6
2.2.6 Propagação de erros..... 7
2.2.7 Avaliação prática do efeito dos erros de arredondamento..... 8
Capítulo 3: Determinação de raízes - equações não-lineares..... 9
3.1 Métodos iterativos..... 10
3.1.1 Convergência de sucessões..... 10
3.1.2 Ordem de convergência..... 13
3.1.3 Tempo de cálculo..... 17
3.1.4 Composição de um método iterativo..... 19
3.1.5 Classificação dos métodos iterativos..... 21
3.2 Método de Bissecção..... 21
3.2.1 Descrição do método..... 21
3.2.2 Erro e Convergência..... 23
3.2.3 Estimativa do número de iterações..... 24
3.2.4 Tempo de cálculo do método..... 24
3.2.5 Algoritmo do método..... 25
3.2.6 Aplicação..... 26
3.2.7 Considerações finais..... 28
3.3 Método da Falsa Posição..... 29
3.3.1 Descrição do método..... 29
3.3.2 Erro e Convergência..... 30
3.3.3 Estimativa do número de iterações..... 32
3.3.4 Tempo de cálculo do método..... 32
3.3.5 Algoritmo do método..... 33
3.3.6 Aplicação..... 33
3.3.7 Considerações finais..... 34
3.3.8 Método da falsa posição modificada..... 36
3.4 Método do Ponto Fixo..... 37
3.4.1 Descrição do método..... 37

3.4.2 Visualização do método iterativo....38
3.4.3 Erro e Convergência.... 39
3.4.4 Tempo de cálculo do método....42
3.4.5 Algoritmo do método....43
3.4.6 Aplicação.... 43
3.4.7 Considerações finais.... 45
3.5 Método de Newton.... 45
3.5.1 Descrição do método.... 45
3.5.2 Erro e Convergência.... 46
3.5.3 Tempo de cálculo do método....47
3.5.4 Método de otimização....48
3.5.5 Método de Newton no caso de zeros múltiplos....48
3.5.6 Algoritmo do método....49
3.5.7 Aplicação.... 50
3.5.8 Considerações finais.... 51
3.6 Método da Secante.... 51
3.6.1 Descrição do método.... 51
3.6.2 Erro e Convergência.... 53
3.6.3 Tempo de cálculo do método....56
3.6.4 Algoritmo do método....56
3.6.5 Aplicação.... 57
3.6.6 Considerações finais.... 59
Capítulo 4: Comparação de métodos iterativos.... 60
Capítulo 5: Conclusão....64
Capítulo 6: Bibliografia....66
Capítulo 7: Anexos....68
7.1 Método da Bissecção em Python (versão completa)....68
7.2 Método do Ponto Fixo em Python (versão completa)....69
7.3 Método da Falsa Posição em Python (versão completa)....70
7.4 Método de Newton-Raphson em Python (versão completa)....71
7.5 Método da Secante em Python (versão completa).... 72

Índice de Figuras

Figura 1.1: Fases de um método numérico..... 1
Figura 2.2.1: Processo de solução de um problema físico..... 3
Figura 3.1: Uma função com dois zeros..... 9
Figura 3.1.1.1: Estudo de convergência..... 12
Figura 3.1.4.1: Fluxograma de um método iterativo..... 20
Figura 3.2.1.1: Método da Bissecção..... 22
Figura 3.2.6.1: Evolução dos valores centrais (m) para o método da Bissecção..... 27
Figura 3.2.6.2: Evolução dos valores centrais (m) para o método da Bissecção sem limite..... 28
Figura 3.3.1.1: Método da Falsa Posição..... 29
Figura 3.3.6.1: Evolução dos valores centrais para o método da Falsa Posição sem limite..... 34
Figura 3.3.8.1: Método da Falsa Posição modificado..... 36
Figura 3.4.1.1: Método do ponto fixo - caso de convergência..... 37
Figura 3.4.1.2: Método do ponto fixo - caso de divergência..... 38
Figura 3.4.2.1: Visualização do método iterativo..... 38
Figura 3.4.2.2: Procedimento do método do ponto fixo..... 39
Figura 3.4.3.1: Ponto de repulsão..... 41
Figura 3.4.6.1: Evolução de estimativas no método do Ponto Fixo..... 44
Figura 3.5.1.1: Método de Newton..... 46
Figura 3.5.7.1: Evolução dos valores da estimativa no método de Newton..... 51
Figura 3.6.1.1: Método da Secante..... 53
Figura 3.6.5.1: Evolução dos valores da estimativa no método da Secante..... 58
Figura 4.1: Comparação de métodos..... 63

Índice de Tabelas

Tabela 2.2.4.1: Aproximação de epsilon....6
Tabela 3.2.5.1: Versão simplificada do método da Bissecção em Python....25
Tabela 3.2.6.1: Aplicação do método da Bissecção em Python....26
Tabela 3.2.6.2: Aplicação do método da Bissecção sem limite em Python....27
Tabela 3.3.5.1: Versão simplificada do método da Falsa posição....33
Tabela 3.3.6.1: Aplicação do método da Falsa Posição em Python....34
Tabela 3.4.5.1: Versão simplificada do método do Ponto Fixo....43
Tabela 3.4.6.1: Aplicação do método da Falsa posição....44
Tabela 3.5.6.1: Versão simplificada do método de Newton....49
Tabela 3.5.7.1: Aplicação do método de Newton....50
Tabela 3.6.4.1: Versão simplificada do método da Secante....57
Tabela 3.6.5.1: Aplicação do método da Secante....57
Tabela 4.1: Comparação dos métodos iterativos....60
Tabela 4.2: Aplicação dos métodos estudados na função 1....61
Tabela 4.3: Aplicação dos métodos estudados na função 2....61
Tabela 4.4: Aplicação dos métodos estudados na função 3....61
Tabela 4.5: Aplicação dos métodos estudados na função 4....62
Tabela 4.6: Aplicação dos métodos estudados na função 5....62
Tabela 4.7: Aplicação dos métodos estudados na função 6....62

Capítulo 1: Introdução

Este projeto enquadra-se no âmbito da obtenção do grau de Mestre em Matemática Aplicada à Economia e Gestão.

O estudo das raízes de equações tem atraído ao longo do tempo a atenção de vários estudiosos, visto que sistemas de equações não-lineares são problemas que podem surgir nas mais diversas áreas da Matemática Aplicada. Sabendo que na maior parte dos problemas matemáticos não é possível encontrar soluções por métodos diretos, ou seja, soluções exatas, opta-se pela resolução usando métodos numéricos.

Num método numérico é possível distinguir três fases:

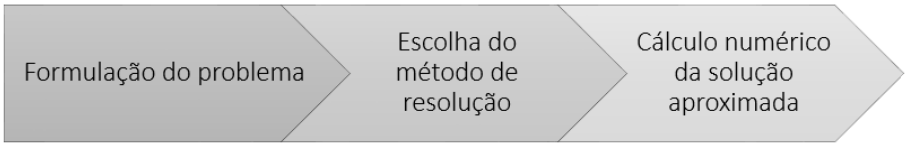


Figura 1.1: Fases de um método numérico

O método numérico é essencialmente um conjunto organizado de operações lógicas e aritméticas baseadas em teoremas da Análise Matemática, que levam a uma solução aproximada de um problema. Um algoritmo sempre acompanha um método numérico.

O surgimento dos computadores tornou possíveis os cálculos numéricos que antes eram humanamente impossíveis. Isso levou ao desenvolvimento de novos métodos numéricos e ao aprimoramento dos métodos já existentes.

A **Teoria Dos Erros** é uma teoria exclusiva da análise numérica. Um método numérico deve sempre ser acompanhado de um estudo de convergência e majorações de erros. Assim, para resolver equações não lineares, é necessário um bom conhecimento de análise matemática, principalmente em teoremas básicos.

Como resultado, a primeira parte deste trabalho está reservada aos conceitos básicos da Análise Matemática, seguidos de conceitos básicos da teoria dos erros.

Na segunda parte são apresentadas as bases teóricas para os principais métodos de resolução de equações não lineares, juntamente com os algoritmos correspondentes que permitem a tradução desses métodos num programa inteligível para o computador.

As versões finais de cada algoritmo serão apresentadas no final em forma de anexo, devolvendo tanto o resultado final quanto os cálculos intermédios. Por outro lado, os códigos apresentados serão apenas versões simplificadas, que devolverão apenas o resultado final.

Capítulo 2: Conceitos preliminares

2.1 Teoremas e conceitos básicos da Análise Matemática

Teorema 2.1.1 (do Valor Intermédio) Seja f: $[a, b]\rightarrow \mathbb{R}$**,** uma função contínua. Dado um número γ entre os valores $f(a)$ e $f(b)$ existe um ponto intermédio $c \in \ ]a, b[$ tal que $f(c) = \gamma$.

Corolário do teorema do valor intermédio

Se f é uma função contínua em $[a, b]$, $\subset \mathbb{R}$ e se $f(a)$ e $f(b)$ tiverem sinais contrários, então existe pelo menos um número real c entre a e b tal que $f(c) = 0$

Teorema 2.1.2 (do Valor Intermédio de Bolzano também conhecido como Teorema de Lagrange): Dada uma função contínua f definida num intervalo fechado $I = [a, b]$ de $\mathbb{R}$ e diferenciável em $]a, b[$, existe um ponto c em $]a, b[$ tal que

$$f'(c) = \frac{f(b)-f(a)}{b-a} \qquad (2.1.1)$$

Corolários do Teorema de Lagrange:

1. Se $f'(x) = 0\ \forall x \in I$, f é $constante\ em\ I$.
2. Se $f'(x) \geq 0$ $(f'(x) \leq 0)$, $\forall x \in I$, $f(x)$ é $crescente$ $(decrescente)$ $em\ I$.
3. Se $f'(x) > 0$ $(f'(x) < 0)$, $\forall x \in I$, $f(x)$ é $estritamente\ crescente$ $(decrescente)$ $em\ I$.

Definição 2.1.1 (extremos relativos): Seja X uma parte não vazia de $\mathbb{R}$. Diz-se que f: $X \rightarrow \mathbb{R}$ tem um máximo local ou relativo no ponto a, quando existe $\varepsilon > 0$ tal que, para qualquer, se tem $f(x) \geq f(a)$. Do mesmo modo f tem um mínimo local ou relativo no ponto a, quando existe $\varepsilon > 0$ tal que, para qualquer $x \in \ ]a - \varepsilon, a + \varepsilon[\cap X$, se tem $f(x) \geq f(a)$. Dizemos também que f tem um extremo local ou relativo no ponto a.

Teorema 2.1.3 (Teorema de Fermat) Se I é um intervalo de $\mathbb{R}$ com mais de um ponto, f: $I \rightarrow \mathbb{R}$ é diferenciável no ponto a interior a I e f tem um extremo local em a, então $f'(a) = 0$.

Teorema 2.1.4: (Teorema de Rolle) Se f é uma função contínua em $[a,\ b]$ derivável em $[a,\ b]$ e $f(a) = f(b)$ então existe pelo menos um ponto $c \in \ ]a,\ b[$, tal que $f'(c) = 0$.

Corolário do Teorema de Rolle: se $c_1\ e\ c_2$ são dois zeros consecutivos de f' então entre $c_1\ e\ c_2$ existe quanto muito um zero de f.

Teorema 2.1.5 (Teorema de Taylor) Seja f uma função derivável até ordem $n + 1$ no intervalo $[a,\ b]$, e supondo que $f^{(n+1)}$ é contínua no intervalo $[a,\ b]$. Se x e x_0 são dois pontos de $[a,\ b]$, com $x \neq x_0$, então existe um ponto c entre x e x_0 tal que:

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k + \frac{f^{(n+1)}(c)}{(n+1)!}(x - x_0)^{n+1} \qquad (2.1.2)$$

A igualdade acima é conhecida por fórmula de Taylor com o resto de Lagrange, onde

$$R(x) = \frac{f^{(n+1)}(c)}{(n+1)!}(x - x_0)^{n+1} \qquad (2.1.3)$$

$R(x)$ é chamado de Resto de Lagrange, ou mais precisamente, **resto na forma de Lagrange**.

Teorema 2.1.6 (estimativa elementar de erros de raízes). Seja x uma aproximação da raiz z da função $f \in C^1([x; z])$, então

$$\left|e_x\right| = |z - x| \leq \frac{|f(x)|}{min_{\xi \in [x;z]}|f'(\xi)|}. \qquad (2.1.4)$$

Teorema 2.1.7 (teorema de erro da interpolação) sejam $x_0, \ldots, x_n, n + 1$ pontos distintos reais, e seja $p_n(x)$ o polinómio interpolador a $f\colon D \to \mathbb{R}$ de grau até n nos x_k então para cada $x \in D$ o erro de interpolação é

$$f(x) - p_n(x) = f[x_0, x_1, \ldots, x_n, x] \prod_{j=0}^{n}(x - x_j) \qquad (2.1.5)$$

com

$$f[x_0, x_1, \ldots, x_n, x] = \frac{f[x_1, \ldots, x_n, x] - f[x_0, x_1, \ldots, x_n]}{x - x_0} \qquad (2.1.6)$$

e

$$f[x_0] = f(x_0) \qquad (2.1.7)$$

2.2 Conceitos básicos da teoria dos erros

Todos os campos do cálculo numérico têm a noção de erro. Por um lado, os dados em si não são sempre precisos; por outro lado, as operações sobre valores inexatos propagam esses erros aos seus resultados. Finalmente, a fim de minimizar os erros, os próprios métodos numéricos, que geralmente são aproximados, procuram os resultados mais próximos possível do que seriam valores exatos.

Erro é a diferença entre o valor exato e o valor aproximado.

Para facilitar a apresentação das fontes de erros, o processo de solução de um problema físico, por meio de aplicação de métodos numéricos, é representado abaixo de uma forma geral.

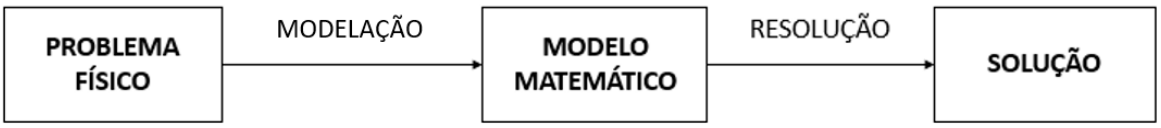


Figura 2.2.1: Processo de solução de um problema físico

Duas fases podem ser identificadas no diagrama anterior:

a) **Modelação** – é a fase de obtenção de um modelo matemático que descreve o comportamento do sistema físico em questão.
b) **Resolução** – é a fase de obtenção da solução do modelo matemático através da aplicação de métodos numéricos

2.2.1 Erros na Fase de Modelação

Raramente se obtém uma descrição precisa de um fenômeno do mundo físico quando se usa um método matemático para representá-lo. Um modelo geralmente requer várias simplificações do mundo físico.

Exemplo:

Estudo do movimento de um corpo sujeito a uma aceleração constante. Tem-se a seguinte equação:

$$d = d_0 + v_0 t + \frac{at^2}{2} \qquad (2.2.1.1)$$

Onde:

d: distância percorrida

d_0: distância inicial

v_0: velocidade inicial

t: tempo

a: aceleração

Determinar a altura de um edifício com uma bola de metal e um cronómetro: 3s

$$d = 0 + 0 \times 3 + \frac{9.8 \times 3^2}{2} = 44.1 \qquad (2.2.1.2)$$

Este resultado é confiável?

1. Fatores não considerados:
   - resistência do ar
   - velocidade do vento, etc.
2. Precisão dos dados de entrada:
   - Se o tempo fosse 3,5s → d = 60.025m
   - Variação de 16,7% no cronómetro → 36% na altura.

2.2.2 Erros na Fase de Resolução

Muitas vezes, a resolução de modelos matemáticos requer o uso de instrumentos de cálculo que requerem certas aproximações para funcionar. Tais aproximações podem gerar erros, tais como: conversão de bases, erros de arredondamento e erros de truncamento.

2.2.3 Erros Absolutos e Relativos

Erro absoluto (EA) é a diferença entre o valor exato de um número N e o seu valor aproximado N':

$$EA_N = N - N' \qquad (2.2.3.1)$$

Por exemplo, sabendo-se que $\pi \in (3.14, 3.15)$ tomaremos para π um valor dentro deste intervalo e teremos, então, $|EA_N| = |\pi - \pi'| < 0.01$.

O erro absoluto pode não ser suficiente para informar sobre a qualidade de uma aproximação.

Erro Relativo é definido como o erro absoluto dividido pelo valor aproximado:

$$ER_N = \frac{EA_N}{N'} = \frac{N - N'}{N'} \qquad (2.2.3.2)$$

É claro que EA_N só poderá ser determinado se N for exatamente conhecido; como isso é raro, em cálculos numéricos é habitual trabalhar com uma limitação máxima para o erro (**estimativa do erro**), ao invés do próprio (indicando-se, então, $|E| < \varepsilon$, onde ε é a estimativa).

Por exemplo, se $\alpha = 3876.373$ e só desejamos a parte inteira α', o erro absoluto será:

$$\Delta_a = \left|a - a'\right| = 0.373 \qquad (2.2.3.3)$$

Se fizermos o mesmo com o número $\beta = 1.373$, teremos:

$$\Delta_\beta = \left|\beta - \beta'\right| = 0.373 \qquad (2.2.3.4)$$

Obviamente, o efeito de aproximação de β é muito maior do que em α, mas o erro absoluto é o mesmo nos dois casos. O erro relativo, no entanto, pode traduzir perfeitamente este fato, pois:

$$\delta_a = \frac{0.373}{3876} \cong 0.000096 < 10^{-4} \qquad (2.2.3.5)$$

$$\delta_\beta = \frac{0.373}{1} \cong 0.373 < 5 * 10^{0} \qquad (2.2.3.6)$$

2.2.4 Erro de Arredondamento

Quer os cálculos sejam efectuados manualmente, quer obtidos por computador, somos conduzidos a utilizar uma aritmética de precisão finita, ou seja, apenas podemos ter em consideração um número finito de dígitos. O erro devido a desprezar os outros e arredondar o número é designado por **erro de arredondamento**, os erros de arredondamento podem ser: iniciais, intermédios ou finais.

Os erros iniciais ocorrem quando os dados iniciais são arredondados. Os erros intermediários e finais são causados pela apresentação final dos resultados e pelos erros cometidos durante o uso do método numérico. Os tipos de arredondamentos mais conhecidos são:

- Arredondamento para baixo;
- Arredondamento para cima;
- Arredondamento para o número de máquina mais próximo

Critério de Arredondamento: no cálculo manual, ao registar um valor aproximado, é habitual usar a seguinte regra:

1. somar meia unidade após a última casa decimal a conservar;

2. desprezar as demais casas.

Assim, tem-se:

$$\sqrt{2} = 1.414... \cong 1.41 \qquad (1.414... + 0.005 = 1.419... \rightarrow 1.41) \qquad (2.2.4.1)$$

$$\sqrt[3]{2} = 1.259\ ... \cong 1.26 \qquad (1.259... + 0.005 = 1.264... \rightarrow 1.26) \qquad (2.2.4.2)$$

O uso deste critério limita o erro a meia unidade da última casa conservada:

$$E = \sqrt{2} - 1.41 = 1.41421... - 1.41 = 0.00421 < 0.005 \qquad (2.2.4.3)$$

Os valores aproximados obtidos podem ser inferiores (valor aproximado por falta) ou superiores (valor aproximado por excesso) aos exatos;

- 1.41 é o valor aproximado de $\sqrt{2}$ (arredondamento para baixo);
- 1.26 é o valor aproximado de $\sqrt[3]{2}$ (arredondamento para cima).

Para concluir este tópico de erro de arredondamento, é fundamental saber o número de dígitos significativos do sistema de representação da máquina que será utilizado para se ter uma ideia da precisão do resultado que será obtido.

Além da precisão decimal, o cálculo do chamado Épsilon da máquina dá-nos uma ideia da exatidão da máquina.

O ε da máquina é o menor número de ponto flutuante, tal que: $1 + \varepsilon > 1$. Alguns métodos para cálculo de ε não dão o seu valor exato, mas isto nem sempre é necessário, pois o que importa é a sua ordem de grandeza.

O programa abaixo, escrito na linguagem Python, calcula uma aproximação do ε da máquina:

Tabela 2.2.4.1: Aproximação de epsilon

```
Eps = 1;
while (Eps + 1 > 1):
   Eps = Eps/2;
print("A máquina acha que", Eps, "somado ao 1, dá 1.")
```

O programa acima, executado no Python, obteve a seguinte resposta:

A máquina acha que 1.1102230246251565e-16 somado ao 1, dá 1.

Logo, o número de dígitos significativos é 16.

2.2.5 Erro de Truncamento

São erros causados por processos que devem ser infinitos ou muito grandes para determinar um valor e, por motivos práticos, são truncados.

Estes processos infinitos são amplamente utilizados para avaliar funções matemáticas como logaritmos, exponencial, trigonométricas e muitas outras que uma máquina pode ter.

Exemplo: uma máquina poderia calcular a função seno e exponencial utilizando os seguintes métodos:

$$sen(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots \qquad (2.2.5.1)$$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots \qquad (2.2.5.2)$$

Fazendo truncamento:

$$sen(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots + (-1)^n \frac{x^n}{n!} \qquad (2.2.5.3)$$

$$e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} \qquad (2.2.5.4)$$

A solução é interromper os cálculos quando uma determinada precisão é atingida.

De uma maneira geral, pode-se dizer que o erro de truncamento pode diminuir até ficar na ordem do erro de arredondamento. A partir desse ponto, não faz sentido diminuir o erro mais, pois o erro de arredondamento será dominante.

2.2.6 Propagação de erros

A maneira como um erro que ocorre em algum passo dos cálculos se propaga é uma questão importante na Análise Numérica. É importante saber se o impacto desse erro será maior ou menor à medida que as operações restantes são executadas.

Supondo que as operações abaixo sejam processadas em uma máquina com 4 dígitos significativos fazendo-se

$$x_1 = 0.3491 * 10^4 \qquad (2.2.6.1)$$

$$x_2 = 0.2345 * 10^0 \qquad (2.2.6.2)$$

Tem-se

$$\begin{aligned}(x_1 + x_2) - x_1 &= \left(0.3491 * 10^4 + 0.2345 * 10^0\right) - 0.3491 * 10^4 \\ &= 0.3491 * 10^4 - 0.3491 * 10^4 \\ &= 0.0000\end{aligned} \qquad (2.2.6.3)$$

$$\begin{aligned}x_2 + (x_1 - x_1) &= 0.2345 * 10^0 + \left(0.3491 * 10^4 - 0.3491 * 10^4\right) \\ &= 0.2345 + 0.0000 \\ &= 0.2345\end{aligned} \qquad (2.2.6.4)$$

Como a adição é uma operação associativa e comutativa, os dois resultados são diferentes, o que não deveriam ser. A diferença foi causada por um arredondamento feito ao adicionar $(x_1 + x_2)$, cujo resultado tem 8 dígitos. Como a máquina só armazena quatro dígitos, os dígitos menos importantes foram desprezados.

Ao usar máquinas de calcular, deve-se estar atento a essas particularidades causadas pelo erro de arredondamento, não apenas na adição, mas também em outras operações.

2.2.7 Avaliação prática do efeito dos erros de arredondamento

Em algoritmos de uma certa complexidade, qualquer método de análise de erros dificilmente poderá ser aplicado. Nesta circunstância, que podemos fazer para verificar a influência dos erros de arredondamento nos resultados? Existem basicamente duas técnicas empíricas que, embora não totalmente seguras, contribuem para esse objetivo.

1. A primeira, dirigida principalmente à verificação da influência da precisão finita, consiste em resolver o problema em causa com precisão aumentada, adoptando, por exemplo, precisão dupla para todas as variáveis. Se os resultados vierem substancialmente alterados, então podemos concluir que o nosso algoritmo é muito sensível aos erros de arredondamento, e neste caso há que examinar o problema a fim de averiguar se é apenas o algoritmo que é instável ou se é o próprio problema que é mal condicionado e tomar as medidas adequadas;
2. A segunda técnica consiste em produzir pequenas perturbações nos dados e analisar a sua influência nos resultados. Uma grande variação destes aponta para a instabilidade do algoritmo e /ou mau condicionamento do problema.

Deve-se sublinhar que qualquer uma destas técnicas requer alguma experiência, quer na sua aplicação quer na correta interpretação dos efeitos que produzem.

Capítulo 3: Determinação de raízes - equações não-lineares

Este capítulo examinará várias abordagens para a resolução numérica de equações algébricas não lineares, isto é, equações que podem ser representadas como $f(x) = 0$, onde f é uma função real de variável real. Um valor s que anula f, isto é, tal que $f(s) = 0$, chamamos de **zero da função** f ou solução da equação $f(x) = 0$.

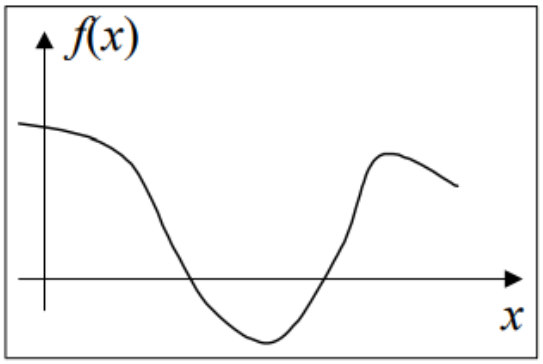


Figura 3.1: Uma função com dois zeros

Antes de tentar aplicar qualquer método de resolução a uma equação do tipo $f(x) = 0$, é fundamental verificar se a equação realmente possui uma solução, ou seja, se existe um número real s tal que $f(s) = 0$.

Muitas vezes também é importante descobrir se a solução é única ou se existem diferentes soluções. Nesses casos, é importante saber qual ou quais das soluções escolher.

Os métodos de resolução de uma equação do tipo $f(x) = 0$ podem dividir-se em dois grandes grupos:

- **métodos diretos**
- **métodos iterativos**.

Nos métodos diretos, a equação é resolvida por intermédio de expressões que envolvem a função f. As soluções de uma equação são encontradas de forma precisa após um número finito de operações supondo que a aritmética exacta é usada). Estes métodos são limitados a determinados tipos de problemas. Por exemplo, resolver uma equação linear $ax + b = 0$ reduz-se a efetuar a divisão $-b/a$, enquanto para resolver uma equação de segundo grau $ax^2 + bx + c = 0$, podemos usar uma fórmula resolvente que nos dá $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$, valores que são trivialmente calculáveis.

A determinação de uma fórmula resolvente para equações de terceiro e quarto grau constitui um problema algébrico que perdurou por dois milénios e foi finalmente resolvido por matemáticos italianos do séc. XVI (Tartaglia, Cardan, Ferrari). No entanto, a determinação da fórmula resolvente para uma equação de grau 5 voltou a causar grandes dificuldades aos matemáticos seguintes, durante quase três séculos.

No seguimento de trabalhos de Lagrange, Vandermonde e Ruffini, demonstrou-se no séc. XIX (Galois, Abel), que é impossível obter uma fórmula resolvente geral para equações algébricas de grau maior ou

igual a cinco. Este resultado surpreendente apresenta logo grandes restrições à possibilidade de resolução algébrica de equações. Com efeito, se existe dificuldade em determinar zeros de polinómios, essa dificuldade assume ainda maiores proporções quando analisamos funções f não polinomiais.

Por exemplo, se é simples dizer que os zeros da função $f(x) = sin(x)$ são valores $x = k\pi$ para $k \in \mathbb{Z}$, já nos é impossível determinar de forma geral as raízes de equação da forma $sin(ax^2 + b) = cx + d$, etc. Para este tipo de equações não lineares temos, no entanto, a possibilidade de encontrar soluções usando métodos iterativos.

3.1 Métodos iterativos

Os métodos iterativos caracterizam-se por gerarem sucessões convergentes para as soluções da equação a resolver. Estes métodos distinguem-se entre si pela forma como são geradas as sucessões de soluções aproximadas. Os métodos iterativos são aplicáveis a uma variedade de problemas e podem ser classificados da seguinte forma:

$$x_{k+1} = \phi(x_k); \quad k = 0, 1, 2, 3,... \qquad (3.1.1)$$

Usando uma estimativa inicial da solução, x_0, é possível gerar uma sequência de valores a partir da fórmula iterativa:

$$\begin{aligned} x_1 &= \phi(x_0) \\ x_2 &= \phi(x_1) \\ x_3 &= \phi(x_2) \\ &\dots \end{aligned} \qquad (3.1.2)$$

A aplicação de métodos iterativos exige apenas a satisfação de condições sobre propriedades mais gerais da função f, como **continuidade**, **monotonia**, **diferenciabilidade** ou **limites inferiores** ou **superiores de derivadas**. Isso difere dos métodos diretos, que exigem formas bem específicas da função f, como funções quadráticas ou funções afins, por exemplo.

3.1.1 Convergência de sucessões

Um método iterativo consiste, de um modo geral, numa aproximação inicial x_0, também designada por ***iteração inicial***, e num processo de obter sucessivamente novas iterações x_{n+1} a partir das anteriores x_0, x_{n-1}, ...

Ao executar este procedimento criamos uma sucessão (x_n) que idealmente converge para o valor pretendido z.

Será crucial estabelecer de que forma um elemento da sucessão está mais ou menos próximo de z, e para isso definimos em cada iteração o erro

$$e_n = z - x_n \qquad (3.1.1.1)$$

É claro que, havendo convergência, o erro cometido em cada iteração $e_n = z - x_n$ vai tender para zero, permitindo assim considerar os valores x_n como aproximações da raiz z a menos de um valor $\varepsilon > 0$ suficientemente pequeno, majorante do erro absoluto, $|e_n| < \varepsilon$.

Devemos distinguir conceptualmente entre dois tipos de estimativas de erros que podemos estabelecer, as estimativas ***a priori*** e as estimativas ***a posteriori***.

Numa estimativa *a priori* não é necessário calcular a iterada x_n para podermos apresentar uma majoração do erro $|e_n|$, ou seja, a partir da função conseguimos prever (sem efetuar iterações) que x_n é um valor suficientemente próximo do resultado. Ao contrário, numa *a posteriori* apenas podemos estabelecer qual a majoração do erro $|e_n|$ se calcularmos o valor x_n, através do cálculo efetivo das iterações.

Como é suposto o erro e_n convergir para zero, começamos por relembrar as notações, que permitem comparar a convergência de sucessões que tendem para 0,

$$e_n = O(a_n), \quad se\ \exists C > 0: \ \left|\frac{e_n}{a_n}\right| < C \qquad (3.1.1.2)$$

$$e_n = o(a_n), \quad se\ \left|\frac{e_n}{a_n}\right| \to 0 \qquad (3.1.1.3)$$

Assim quando escrevemos $e_n = O\left(\frac{1}{n^2}\right)$, significa que a sucessão e_n converge para zero de forma semelhante a $\frac{1}{n^2}$. Iremos abordar métodos iterativos para os quais a convergência do erro para zero é muito superior a $\frac{1}{n^2}$. Na realidade, podemos obter, em alguns casos, convergências do erro para zero do tipo $O\left(2^{-n}\right)$ ou mesmo $O\left(2^{-2^n}\right)$.

Um erro frequente é supor que se verificarmos que $x_{n+1} - x_n$ converge para zero, isso significa que há convergência, isso deve-se ao facto de o critério $|x_{n+1} - x_n| < \varepsilon$ ser muitas vezes utilizado como critério de paragem de um método.

Um exemplo seria considerar a série

$$s_n = \sum_{k=0}^{n} \frac{1}{k} \qquad (3.1.1.4)$$

Que não é convergente e no entanto $s_{n+1} - s_n = \frac{1}{n+1}$ tende para zero. Porém, uma análise mais detalhada permite concluir o seguinte lema.

Lema 3.1.1.1 Seja $d_n = x_{n+1} - x_n$. A sucessão x_n é convergente se e só se a série

$$s_n = \sum_{k=0}^{n} d_k \qquad (3.1.1.5)$$

for convergente. Em particular, se existir p tal que

$$\left|x_{n+1} - x_n\right| n^p \to c, \qquad (3.1.1.6)$$

então x_n converge se $p > 1$ e $c < +\infty$, e diverge se $p \leq 1$.

Demonstração: basta reparar que

$$x_n - x_m = \sum_{k=m}^{n-1} x_{n+1} - x_n = \sum_{k=m}^{n-1} d_n = s_{n-1} - s_m \qquad (3.1.1.7)$$

Assim s_n é uma sucessão de Cauchy se e só se x_n também o for. O resultado particular surge de aplicar o critério de comparação com a série $\sum \frac{1}{n^p}$, que é convergente para $p > 1$ e divergente para $p \leq 1$.

Aplicando o lema anterior, basta que $x_{n+1} - x_n$ convirja para zero mais rapidamente que $\frac{1}{n^p}$, com $p > 1$, para podermos concluir a convergência. Quando $x_{n+1} - x_n$ converge para zero tão ou mais lentamente que $\frac{1}{n}$, é impossível garantir a convergência. Este facto permite extrair informações importantes em circunstâncias em que não se sabe se uma sucessão converge ou não.

Exemplo Consideremos a sucessão x_n, definida recursivamente por

$$x_1 = 1, \; x_{n+1} = 1 + \frac{x_n^2}{x_n + 1} \qquad (3.1.1.8)$$

Calculando os termos da sucessão, obtemos por exemplo

$$x_{998} = 43.7122, \; x_{999} = 43.7345, x_{1000} = 43.7569, \; x_{1001} = 43.7792. \qquad (3.1.1.9)$$

E reparamos que os termos se estão a aproximar, embora muito lentamente. Uma observação descuidada poderia levar-nos a atribuir uma aproximação 43.7… para o limite da sucessão.

Apesar disso, a sucessão x_n não converge. Com base na comparação da sucessão $|x_{n+1} - x_n|$ com $\frac{1}{\sqrt{n}}$ obtemos o gráfico seguinte:

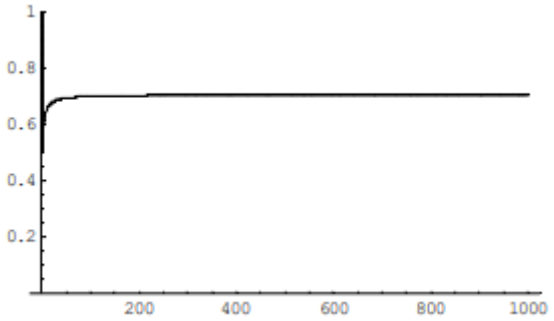


Figura 3.1.1.1: Estudo de convergência

Sendo assim, $\left|x_{n+1} - x_n\right|\sqrt{n} \sim 0.706...$, e portanto segundo o critério apresentado x_n não deverá convergir.

Ainda com base nestes valores e reparando que $x_{n+1} - x_n = \frac{1}{x_n+1}$, podemos escrever $\frac{1}{x_n+1} \approx \frac{0.706}{\sqrt{n}}$, o que nos dá $x_n \approx \frac{\sqrt{n}}{0.706} - 1$. Assim podemos efetuar uma estimativa para o valor x_{5000} com o simples cálculo da raiz e sem fazer as 4000 iterações restantes. O valor obtido pela estimativa é 99.02 (erro relativo de 0.1%). Se o mesmo fosse efetuado para x_{10000} obtemos pela estimativa 140.6 ao invés de 140.4, com erro relativo de 0.14%, mas poupando 9000 iterações. Este exemplo ilustra a utilidade da previsão do comportamento assimptótico.

Poderá pesar-se que é bem conhecida uma justificação teórica global que permite concluir a estimativa $x_n = O(\sqrt{n})$, mas não… Neste caso, o problema "mais simples" consiste em mostrar que para sucessões definidas recursivamente por $x_{n+1} = x_n + \frac{1}{x_n^p+1}$ o comportamento assimptótico de x_n é um $O(n^{\frac{1}{p+1}})$. Repare-se que isto é consistente nos casos limite, p = 0, quando se reduz ao somatório, e quando $p = \infty$, em que se reduz à constante inicial.

3.1.2 Ordem de convergência

Consideremos diversos tipos de sucessões que convergem para 1,

$$u_n = 1 + \frac{1}{n^3},\ v_n = 1 + \frac{1}{3^n},\ x_n = 1 + \frac{1}{3^{2n}},\ y_n = 1 + \frac{1}{3^{3n}},\ z_n = 1 + \frac{1}{3^{2^n}} \qquad (3.1.2.1)$$

Qualquer uma destas sucessões é uma aproximação do número 1, enquanto sucessões de Cauchy de racionais que convergem para 1. No entanto, reparamos que podem ser distinguidas pela rapidez de convergência, verificando que a mais lenta é a primeira, u_n, e a mais rápida é a última z_n.

Assim começamos a estabelecer essa diferença definindo ordem de convergência.

Definição 3.1.2.1 (ordem de convergência linear) Seja x_n uma sucessão convergente para z, e seja $e_n = z - x_n \neq 0$. Consideremos a sucessão

$$K_n = \frac{|e_{n+1}|}{|e_n|} \qquad (3.1.2.2)$$

Se $K_n \leq K^+ < 1$, para n suficientemente grande, dizemos que a sucessão tem pelo menos ordem de convergência 1 ou linear.

Se, para além disso, $K_n \geq K^- > 0$ dizemos que a sucessão tem ordem de convergência linear.

- Quando existe limite finito

$$K_\infty = lim\, K_n = lim \frac{|e_{n+1}|}{|e_n|} \qquad (3.1.2.3)$$

este valor é designado **coeficiente assimptótico de convergência**.

- Se $K_\infty < 1$, é fácil verificar que estamos numa situação em que há pelo menos convergência linear.
- Se $0 < K_\infty < 1$, então podemos concluir que estamos numa situação em que a convergência é linear.
- Se $K_\infty = 0$, dizemos que a sucessão tem ordem de convergência supralinear
- Falta analisar se a situação $K_\infty \geq 1$ ocorre. Suponhamos que $K_\infty > 1$, então, a partir de certa ordem, teríamos $K_n > 1$, ou seja

$$\left|e_{n+1}\right| > |e_n| \qquad (3.1.2.4)$$

e isto contraria o facto de a sucessão convergir. Portanto, apenas podemos ter $K_\infty = 1$.

Quando a sucessão converge e $K_\infty = 1$, então dizemos que a ordem de convergência é logarítmica.

Exemplo. Vejamos os exemplos apresentados de sucessões que convergem para 1.

- A sucessão u_n tem convergência logarítmica, porque

$$\frac{|e_{n+1}|}{|e_n|} = \frac{|1-u_{n+1}|}{|1-u_n|} = \frac{\frac{1}{(n+1)^3}}{\frac{1}{n^3}} = \frac{n^3}{(n+1)^3} \to 1, \qquad (3.1.2.5)$$

concluímos que $K_\infty = 1$.

- A sucessão v_n tem convergência linear, porque

$$\frac{|e_{n+1}|}{|e_n|} = \frac{|1-v_{n+1}|}{|1-v_n|} = \frac{3^{-(n+1)}}{3^{-n}} = \frac{1}{3}, \qquad (3.1.2.6)$$

concluímos que $K_\infty = \frac{1}{3} \in]0,\ 1[$.

- A sucessão x_n tem convergência supralinear, porque

$$\frac{|e_{n+1}|}{|e_n|} = \frac{|1-x_{n+1}|}{|1-x_n|} = \frac{3^{-2^{(n+1)}}}{3^{-2^n}} = \frac{3^{-2^n}}{3^{-2^n}} 3^{-2^n} = 3^{-2^n} \to 0 \qquad (3.1.2.7)$$

concluímos que $K_\infty = 0$.

Por cálculos semelhantes concluímos que y_n e z_n também têm convergência supralinear.

Observação: a definição não é enunciada diretamente no caso em que o limite K_∞ existe, porque pode haver situações em que tal limite não exista, e até se verifique em algumas iteradas $\left|e_{n+1}\right| > |e_n|$. Por exemplo, consideramos a sucessão

$$w_n = 1 + \frac{2+(-1)^n}{4^n} \qquad (3.1.2.8)$$

que converge para 1, com rapidez semelhante à de v_n. Neste caso

$$K_n = \frac{|1-w_{n+1}|}{|1-w_n|} = \frac{|2+(-1)^{n+1}|}{|4^{n+1}|}\frac{|4^n|}{|2+(-1)^n|} = \frac{2+(-1)^{n+1}}{2+(-1)^n}\frac{1}{4} = \quad (3.1.2.9)$$

$$= \frac{1}{12}\ caso\ n\ seja\ par$$

ou

$$= \frac{3}{4}\ caso\ n\ seja\ ímpar$$

e, portanto, não há limite. No entanto, como enquadramos

$$0 < \frac{1}{12} < K_n < \frac{3}{4} < 1 \quad (3.1.2.10)$$

podemos assim concluir que se trata de uma convergência linear.

Definição 3.1.2.2 (ordem de convergência superior) Seja x_n uma sucessão convergente para z. Para $p > 1$ consideramos a sucessão

$$K_n^{[p]} = \frac{|e_{n+1}|}{|e_n|^p} \quad (3.1.2.11)$$

Se $K_n^{[p]} \leq K^+ <+\infty$, para n suficientemente grande, dizemos que a sucessão tem pelo menos ordem de convergência p.

Se, para além disso, $K_n^{[p]} \geq K^- > 0$ dizemos que a sucessão tem ordem de convergência p.

- Mais uma vez, quando existe o limite finito,

$$K_\infty^{[p]} = lim\, K_n^{[p]} = lim\, \frac{|e_{n+1}|}{|e_n|^p} \quad (3.1.2.12)$$

designamos este valor coeficiente assimptótico de convergência de ordem p (quando for clara qual é a ordem, não colocamos o índice p).

- Se $0 < K_\infty^{[p]} <+\infty$, então podemos concluir a convergência tem ordem p.
- Se $p = 2$ a convergência diz-se quadrática e se $p = 3$, cúbica. Se $K_\infty^{[p]} = 0$ para qualquer p, então diremos que se trata de uma convergência exponencial.

Observação: note-se que se a sucessão tiver ordem de convergência p, então terá pelo menos ordem de convergência $q < p$, o que o torna a designação consistente. Com efeito, como $lim|e_n|^{p-q} = 0$,

$$\frac{|e_{n+1}|}{|e_n|^q} = \frac{|e_{n+1}|}{|e_n|^p}|e_n|^{p-q} \to K_\infty^{[p]}\, lim|e_n|^{p-q} = 0 \quad (3.1.2.13)$$

Se, por outro lado, considerarmos $q > p$, temos $lim|e_n|^{p-q} =+\infty$, e como $K_{p,\infty} \neq 0$,

$$\frac{|e_{n+1}|}{|e_n|^q} = \frac{|e_{n+1}|}{|e_n|^p}|e_n|^{p-q} \to K_\infty^{[p]}\, lim|e_n|^{p-q} =+\infty \quad (3.1.2.14)$$

donde se conclui que não poderia ter ordem de convergência superior.

Exemplo Analisamos agora os exemplos das sucessões x_n, y_n, z_n que tínhamos verificado terem convergência supralinear. No caso da sucessão x_n,

$$K_n^{[p]} = \frac{|e_{n+1}|}{|e_n|^p} = \frac{3^{-2^{(n+1)}}}{(3^{-2^n})^p} = 3^{-2^{(n+1)} + 2^n p} = 3^{-2^n(2-p)} \qquad (3.1.2.15)$$

e reparamos que se $p > 2$, temos $K_\infty^{[p]} = 0$. Portanto apenas no caso $p = 2$, obtemos $K_\infty^{[2]} = 3$, e concluímos que se trata de uma convergência quadrática.

No caso da sucessão y_n é fácil ver que a sua convergência é cúbica e que temos $K_\infty^{[3]} = 3$.

Finalmente, no caso da sucessão z_n a convergência é considerada exponencial, pois $K_n^{[p]} = 0$, qualquer que seja p, já que

$$K_n^{[p]} = \frac{|e_{n+1}|}{|e_n|^p} = \frac{3^{3^{n^2}p}}{3^{3^{n^2}+2n+1}} = 3^{3^{n^2}(p-3^{2n+1})} \to 0, \ \forall p \qquad (3.1.2.16)$$

Observação: Um dos processos muito utilizados para o cálculo de valores de funções baseia-se no desenvolvimento em série de Taylor. Reparamos que se tivermos um desenvolvimento em série de potências em torno de z_0,

$$f(z) = \sum_{k=0}^{\infty} a_k (z - z_0)^k, \qquad (3.1.2.17)$$

o cálculo dos termos de série pode ser considerado um processo iterativo da forma

$$como\ s_0 = a_0$$

$$e\ s_{n+1} = s_0 + a_{n+1}(z - z_0)^{n+1} \qquad (3.1.2.18)$$

$$portanto,\ s_n = \sum_{k=0}^{n} a_k \left(z - z_0\right)^k$$

No caso da série de Taylor é bem conhecido que $a_n = \frac{f^{(n)}(z_0)}{n!}$. O caso mais simples ocorre quando os termos a_n são constantes. Por exemplo, se $a_n = 1$, $z_0 = 0$, obtemos para $|z| < 1$,

$$f(z) = \sum_{k=0}^{\infty} z^n = \frac{1}{1-z} \qquad (3.1.2.19)$$

ou seja, a bem conhecida fórmula da série geométrica.

Para avaliarmos a rapidez de convergência da série de Taylor, reparamos que

$$\left|\frac{e_{n+1}}{e_n}\right| = \frac{f(z) - s_n + s_n - s_{n-1}}{f(z) - s_n} = 1 - \frac{s_{n+1} - s_n}{f(z) - s_n} \qquad (3.1.2.20)$$

Como

$$s_{n+1} - s_n = \frac{f^{(n+1)}(z_0)}{(n+1)!}(z - z_0)^{n+1}, \qquad (3.1.2.21)$$

$$f(z) - s_n = \frac{f^{(n+1)}(\xi_{n+1})}{(n+1)!}(z - z_0)^{n+1} \qquad (3.1.2.22)$$

onde $\xi_{n+1} \in]z_0; z[$, pelo resto de Lagrange da série de Taylor.

Efetuando a razão entre termos, ficamos com

$$\left|\frac{e_{n+1}}{e_n}\right| = \left|1 - \frac{f^{(n+1)}(z_0)}{f^{(n+1)}(\xi_{n+1})}\right|, \qquad (3.1.2.23)$$

e como admitimos a continuidade de todas as derivadas, para valores de z próximos de z_0 temos $\frac{f^{(n+1)}(z_0)}{f^{(n+1)}(\xi_{n+1})}$ próximo de 1, o que significa que será possível majorar o valor $\left|\frac{e_{n+1}}{e_n}\right|$ por uma constante inferior a 1, concluindo-se a convergência linear (apenas poderia ser supralinear se $\xi_n \to z_0$).

3.1.3 Tempo de cálculo

Notamos que nem sempre uma maior "rapidez" de convergência é profícua.

Assim, suponhamos que x_n é uma sucessão que converge para z mais rapidamente do que y_n. A priori, podemos pensar que é melhor utilizar a sucessão x_n, mas se o cálculo de cada y_n envolver um tempo médio t_y menor que t_x (o tempo médio de cálculo de cada x_n), isso poderá não acontecer.

Existe também a situação em que os tempos de cálculo aumentam com n, por exemplo, no caso de métodos iterativos em que o cálculo de cada iteração necessita do valor de uma ou mais iterações anteriores. Com efeito, podemos assumir que os tempos totais são T_x e T_y, necessários para o cálculo $x_n \, e \, y_n$ (respetivamente), serão $T_x = nt_x$, $T_y = nt_y$. Contudo, não iremos considerar esta situação.

Suponhamos que pretendemos obter um erro absoluto inferior a ε. Examinemos o número de iterações necessário para garantir essa estimativa, considerando várias majorações de erro, de acordo com a ordem de convergência.

- caso de convergência logarítmica

Suponhamos que $|e_n| \leq an^{-s}$, com $s > 0$. Para que $|e_n| \leq \varepsilon$, basta que $an^{-s} \leq \varepsilon$, donde obtemos a relação

$$log\, n \geq \frac{log(a) - log(\varepsilon)}{s} \qquad (3.1.3.1)$$

- caso de convergência linear

Suponhamos que $|e_n| \leq ar^{-n}$, com $r > 1$. Desta mesma forma, para que $|e_n| \leq \varepsilon$, basta que $ar^{-n} \leq \varepsilon$, donde obtemos a relação

$$n \geq \frac{log(a) - log(\varepsilon)}{log(r)} \qquad (3.1.3.2)$$

- caso de convergência supralinear, de ordem $p > 1$

Suponhamos que $\left|e_n\right| \leq ar^{-p^n}$, com $r > 1$. Desta mesma forma, para que $\left|e_n\right| \leq \varepsilon$, basta que $ar^{-p^n} \leq \varepsilon$, donde obtemos a relação

$$p^n \geq \frac{log(a) - log(\varepsilon)}{log(r)} \qquad (3.1.3.3)$$

Estabelecendo a relação $a = M\varepsilon$ em que M será um valor elevado (pois ε pretende-se pequeno) e assumindo que escrevemos $s = log\, r$, obtemos:

- $n \geq M^{\frac{1}{s}}$, no caso de **convergência logarítmica**
- $n \geq log(M^{\frac{1}{s}}) = \frac{log(M)}{log(r)}$, no caso de **convergência linear**
- $n \geq log_p(log(M^{\frac{1}{s}})) = log_p(\frac{log(M)}{log(r)})$, no caso de **ordem supralinear** (de ordem p)

Desta forma, é bem visível a diferença no número de iterações a calcular para obter uma certa precisão. Nestas três classes o número de iterações cresce de forma diferenciada e podemos ver que:

1. se x_n converge linearmente e y_n logaritmicamente, para atingir um erro absoluto inferior a ε, a relação entre os tempos totais será

$$\frac{T_x}{T_y} = \frac{log(M)t_x}{log(r)M^{\frac{1}{s}}t_y} \to 0 \ (M \to \infty) \qquad (3.1.3.4)$$

ou seja, como a é fixo, assumimos $M \to \infty$ corresponde a considerar $\varepsilon \to 0$. Nesse caso, qualquer que seja a relação entre t_x e t_y (considerados constantes), o tempo total de cálculo T_x será inferior a T_y, quando se pretendem erros cada vez mais pequenos. Isto não invalida, como é óbvio, que para alcançar um certo erro, não possa ser inferior T_y. Por exemplo, sendo $a = 1$, para alcançar $\varepsilon = 10^{-N}$, teremos $T_x = T_y$ se

$$\frac{t_y}{t_x} = \frac{log_{10}(10^N)}{log_{10}(r)10^{\frac{N}{s}}} = \frac{10^{-\frac{N}{s}}N}{log_{10}(r)}. \qquad (3.1.3.5)$$

Num caso concreto, $r = 10$, $s = 4$, $N = 8$, obtemos

$$\frac{t_y}{t_x} = \frac{log_{10}(10^8)}{log_{10}(10)10^{\frac{8}{4}}} = \frac{8}{10^2} = 0.08, \qquad (3.1.3.6)$$

e portanto se $t_y < 0.08t_x$, o cálculo de 100 iterações para obter y_{100} com a estimativa $\left|e_n\right| \leq n^{-4}$ (portanto $\left|e_{100}\right| \leq 10^{-8}$), irá demorar menos tempo que o cálculo de 8 iterações para obter x_8 com estimativa $\left|e_n\right| \leq 10^{-n}$. É também claro que se mantivessemos esta relação entre t_y e t_x, ao considerar

um erro mais pequeno $\varepsilon = 10^{-16}$, o tempo para calcular y_{1000} seria maior que x_{16} (a menos que $t_y < 0.0016 t_x$).

2. De forma semelhante existe uma grande diferença entre considerar x_n a convergir supralinearmente e y_n linearmente. Cálculos semelhantes levariam ao mesmo tipo de conclusões.
3. Será que existe também uma grande diferença entre várias ordens de convergência supralinear?

Resposta: Não, suponhamos que x_n converge com ordem p e y_n com ordem q, com $p > q$. Obtemos (para coeficientes iguais, para coeficientes diferentes podemos obter a mesma relação no limite),

$$\frac{T_x}{T_y} = \frac{log_p(log(M^{\frac{1}{\varepsilon}}))t_x}{log_q(log(M^{\frac{1}{\varepsilon}}))t_y} = \frac{log(q)t_x}{log(p)t_y}, \qquad (3.1.3.7)$$

ou seja, neste caso obtém-se uma razão constante. Podemos concluir que se $t_x = t_y$, aumentar a ordem do método $q > 1$ para p, apenas se traduz na relação de tempo de cálculo no factor $\frac{log(q)}{log(p)}$. No caso de passarmos de ordem convergência quadrática para cúbica o salto não é significativo e traduz-se um simples decréscimo de $\frac{log(2)}{log(3)}$, ou seja, aproximadamente 63%. Isto significa que a priori pode não compensar executar um algoritmo que nos dê ordem de convergência cúbica ao invés de quadrática se ele demorar o dobro do tempo. Para que houvesse uma grande diferença entre estes valores, o p teria que ser tão grande quanto possível, ou seja, iríamos entrar no caso de convergência exponencial.

4. Comparação entre métodos com a mesma ordem de convergência, mas com coeficientes diferentes.

Já mencionámos que no caso de convergência supralinear, para os exemplos considerados, a diferença esbate-se no limite. No caso de convergência linear, essa diferença assume um caráter de proporcionalidade tal como acontecia anteriormente, no caso supralinear, ao considerar diferentes ordens p.

Com efeito, se considerarmos $M_x = cM_y$ e $r_x \neq r_y$, obtemos

$$\frac{T_x}{T_y} = \frac{log(M_x)log(r_y)t_x}{log(r_x)log(M_y)t_y} \to \frac{log(r_y)t_x}{log(r_x)t_y} \text{ com } M_y \to \infty \qquad (3.1.3.8)$$

ou seja, verifica-se uma relação semelhante à anterior, mas agora para os coeficientes assimptóticos que são $K_{x,\infty} = \frac{1}{r_x}$ e $K_{y,\infty} = \frac{1}{r_y}$.

3.1.4 Composição de um método iterativo

Nos capítulos anteriores, vimos que, utilizando aproximações anteriores para calcular as novas aproximações, um método numérico iterativo constrói uma sequência de aproximações $x_1, x_2, x_3, \ldots$ de uma raiz de f.

Observamos que, em condições específicas, a sequência construída converge para o valor preciso da raiz. Portanto, é possível obter uma aproximação que satisfaça uma precisão previamente determinada por meio de um número finito de repetições do procedimento.

Tipicamente, a aplicação de um método iterativo parte de uma **estimativa inicial** (x_0), da solução a determinar. Por aplicação de um procedimento bem definido, vão sendo gerados os termos de uma sucessão de estimativas x_n que se pretende que convirja para a solução s pretendida. Em cada **iteração** é calculado um termo da sucessão, ou seja, uma nova estimativa, x_k, à custa da estimativa anterior, x_{k-1}, por intermédio de uma regra que caracteriza o método. Este processo iterativo é terminado assim que a estimativa x_k satisfaz um dado **critério de paragem** (por exemplo x_k estar próximo de s ou $f(x_k)$ ser próximo de 0) ou após um número máximo de iterações ou tempo de processamento. Os critérios de paragem mais frequentes, quando se pretende aproximar a raiz s com uma precisão ε, são:

1. Critério do erro absoluto: $|x_k - x_{k-1}| \leq \varepsilon$
2. Critério do erro relativo: $|x_k - x_{k-1}| \leq \varepsilon |x_k|$
3. Critério do valor da função: $|f(x_k)| \leq \varepsilon$

O fluxograma seguinte mostra como funcionam os métodos iterativos no cálculo de raízes.

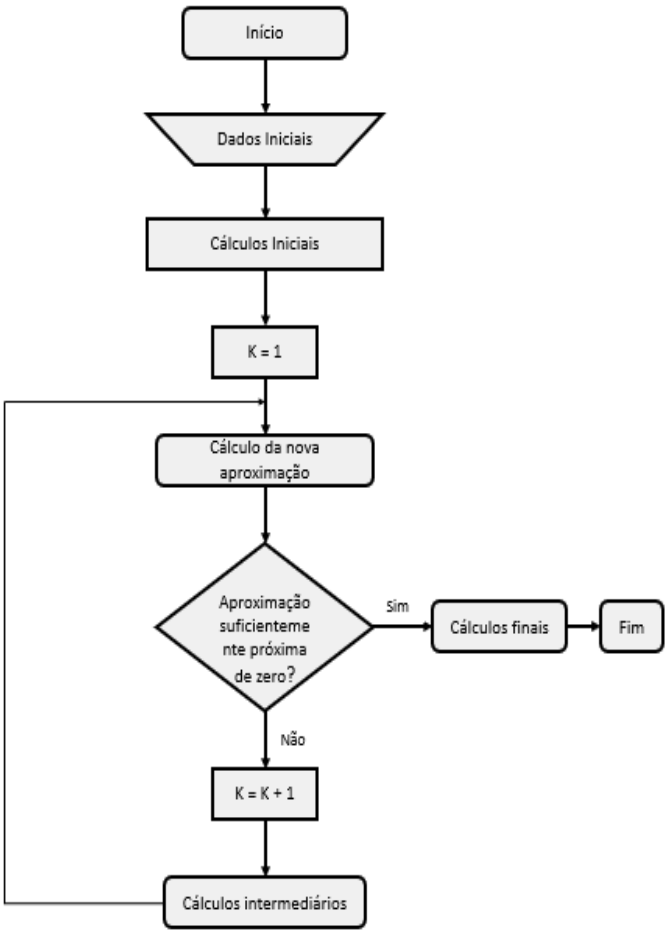


Figura 3.1.4.1: Fluxograma de um método iterativo

Podemos ainda considerar acrescentar ao nosso modelo iterativo uma restrição inicial que seria, por exemplo, verificar se são satisfeitas as hipóteses do **Teorema de Bolzano** no intervalo [a,b] para o qual estamos a usar nos dados iniciais.

3.1.5 Classificação dos métodos iterativos

Métodos de quebra: ainda que os métodos de quebra sejam os mais intuitivos geometricamente, eles convergem mais lentamente. Estas técnicas são chamadas assim porque dividem um intervalo que contém uma raiz da função em intervalos menores que ainda contém a raiz. Podemos usar várias abordagens dependendo do ponto de quebra do intervalo, como

- Método da bissecção
- Método da falsa-posição

Métodos de ponto fixo: Nos métodos de ponto fixo começamos de uma aproximação inicial x_0 e construímos a seqüência x_k na qual cada termo é dado por $x_{k+1} = \zeta(x_k)$, onde ζ é uma função de iteração. Conforme a escolha de ζ (zeta),teremos diferentes métodos de ponto fixo, tais como.

- Método de Newton
- Método do ponto fixo

Métodos de múltiplos pontos: os métodos de múltiplos pontos são uma extensão do método anterior, onde para determinar um ponto x_{k+1} utilizamos vários pontos anteriores: x_k, x_{k-1}, ..., x_{k-p}.

- Método da Secante.

3.2 Método de Bissecção

3.2.1 Descrição do método

O **método da bissecção** é um método bastante simples que se baseia na continuidade da função. Se considerarmos uma função contínua f no intervalo finito $I = [a,\ b]$, tal que $f(a)$ e $f(b)$ têm sinais diferentes, podemos concluir (pelo **Teorema do valor intermédio** ou de **Bolzano**) que existe pelo menos um valor s entre a e b tal que $f(s) = 0$.

O método da bissecção consiste em construir subintervalos $I_k = [a_k,\ b_k] \subset [a,\ b]$ por divisões sucessivas a meio e relativamente aos quais também se verifique que $f(a_k)$ e $f(b_k)$ tenham sinais diferentes. Deste modo vamos confinando uma raiz da função f a intervalos tão pequenos quanto desejarmos.

O algoritmo que concretiza este método é o seguinte. Ponhamos

$$I_0 = [a_0,\ b_0] = I = [a,\ b], \qquad I_k = [a_k,\ b_k], \qquad k = 1,\ 2,\ \ldots \qquad (3.2.1.1)$$

e seja x_{k+1} o ponto médio do intervalo I_k, i.e.,

$$x_{k+1} = \frac{a_k + b_k}{2} \qquad (3.2.1.2)$$

Vamos considerar 3 situações diferentes:

1. $f(x_{k+1}) = 0$, então x_{k+1} é um zero;
2. $f(x_{k+1}) \times f(a_k) < 0$, faça-se

   $$a_{k+1} = a_k \qquad e \qquad b_{k+1} = x_{k+1} \qquad (3.2.1.3)$$

   de modo a que o novo subintervalo $I_{k+1} = [a_{k+1}, b_{k+1}]$ contenha pelo menos um zero de f.

3. $f(x_{k+1}) \times f(b_k) < 0$, faça-se

   $$a_{k+1} = x_{k+1} \qquad e \qquad b_{k+1} = b_k \qquad (3.2.1.4)$$

   de modo a que o novo subintervalo $I_{k+1} = [a_{k+1}, b_{k+1}]$ contenha pelo menos um zero de f.

Pela forma como foi construído, este subintervalo tem um comprimento que é metade do comprimento do subintervalo que o precedeu. Portanto

$$b_{k+1} - a_{k+1} = \frac{b_k - a_k}{2} \qquad (3.2.1.5)$$

A Figura 3.2.1.1 representa esquematicamente este método.

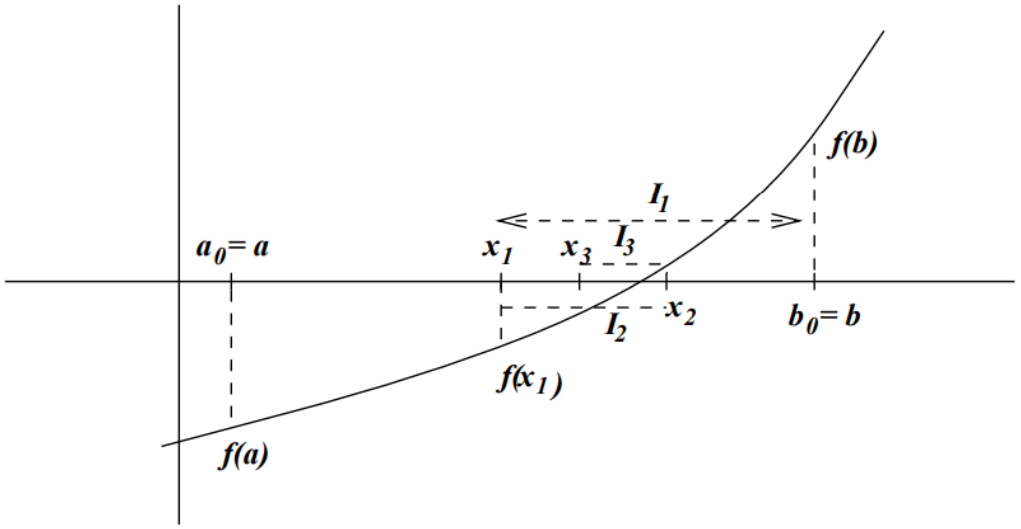


Figura 3.2.1.1: Método da Bissecção

3.2.2 Erro e Convergência

A simplicidade do método da bissecção torna intuitiva a análise da respectiva convergência. Todavia, é útil proceder a uma análise rigorosa que apresentamos no teorema seguinte.

Teorema 3.2.2.1 Seja $f \in C[a,\ b]$ e tal que $f(a)$ e $f(b)$ tenham sinais diferentes. Então, a sucessão x_k determinada pelo método da bissecção converge para uma raiz de f neste intervalo, e o erro satisfaz a relação

$$|e_k| \leq \frac{b-a}{2^k} \qquad (3.2.2.1)$$

Demonstração: Tendo em conta que

$$b_{k+1} - a_{k+1} = \frac{b_k - a_k}{2}, \qquad (3.2.2.2)$$

demonstra-se por indução matemática que

$$b_k - a_k = \frac{b-a}{2^k}. \qquad (3.2.2.3)$$

A sucessão $a_k,\ k = 0,\ 1,\ \ldots$ é crescente e limitada superiormente por b, e a sucessão $b_k,\ k = 0,\ 1,\ \ldots$ é decrescente e limitada inferiormente por a. Então ambas as sucessões convergem e denotemos por $\bar{a}$ e $\bar{b}$ os respectivos limites. Pela forma como o método foi elaborado podemos dizer que

$$a_k < x_{k+1} < b_k \qquad e \qquad \lim_{k\to\infty} |a_k - b_k| = 0 \qquad (3.2.2.4)$$

e, portanto,

$$\bar{a} = \lim_{k\to\infty} a_k = \bar{b} = \lim_{k\to\infty} b_k = \lim_{k\to\infty} x_{k+1} \qquad (3.2.2.5)$$

Por outro lado, atendendo ao modo de construção dos os intervalos,

$$f(a_k) \times f(b_k) < 0, \qquad k = 0,\ 1,\ \ldots \qquad (3.2.2.6)$$

Como f é contínua, temos que

$$\begin{aligned} 0 \geq \lim_{k\to\infty} |f(a_k) \times f(b_k)| = \\ = |\lim_{k\to\infty} f(a_k)| \times |\lim_{k\to\infty} f(b_k)| = \\ = f(\bar{a}) \times f(\bar{b}) = [f(\bar{a})]^2 = [f(\bar{b})]^2 \geq 0 \end{aligned} \qquad (3.2.2.7)$$

Destas desigualdades concluímos que $f(\overline{a}) = f(\overline{b}) = 0$ e que, por conseguinte, $\overline{a} = \overline{b} = s$ é um zero da função f.

O **erro** cometido na iteração k satisfaz

$$|e_k| \leq \frac{|a_k - b_k|}{2} \qquad (3.2.2.8)$$

Tendo em conta que cada subintervalo tem um comprimento que é metade do comprimento do subintervalo que o precedeu deduz-se imediatamente que

$$0 \leq \left|\frac{e_{k+1}}{e_k}\right| \leq \frac{1}{2} \qquad (3.2.2.9)$$

O que mostra a convergência do método da bissecção é linear. Todavia, a análise feita não permite concluir que a constante de erro assimptótico seja $c = \frac{1}{2}$.

3.2.3 Estimativa do número de iterações

Considerando uma precisão ε e um intervalo inicial $[a, b]$ é possível saber quantas iterações devem ser efetuadas pelo método da bissecção até que se obtenha $|a - b| \leq \varepsilon$, usando o algoritmo deste método.

$$b_k - a_k = \frac{b_{k-1} - a_{k-1}}{2} = \frac{b_0 - a_0}{2^k} \qquad (3.2.3.1)$$

Deve-se obter o valor de k tal que $b_k - a_k < \varepsilon$, ou seja,

$$\frac{b - a}{2^k} < \varepsilon \Rightarrow \frac{b - a}{\varepsilon} < 2^k$$

$$log(b - a) - log(\varepsilon) < k \times log(2) \qquad (3.2.3.2)$$

$$\frac{log(b - a) - log(\varepsilon)}{log(2)} < k$$

Portanto, se k satisfaz a relação acima, ao final da iteração k teremos o intervalo $[a_k, b_k]$ que contém a raiz s e cujo comprimento é inferior a ε.

3.2.4 Tempo de cálculo do método

Em cada iteração, a função é calculada apenas uma vez (no ponto x_{k+1}), já que os valores $f(a_k)$ e $f(b_k)$ devem estar guardados, não devem ser recalculados! Portanto, sendo t_f o tempo de cálculo médio da função f, o tempo total para obter x_n será aproximadamente

$$T = nt_f \qquad (3.2.4.1)$$

desprezando o tempo necessário para verificar se tem o mesmo sinal (a multiplicação é suficientemente rápida para poder ser desprezada).

3.2.5 Algoritmo do método

O seguinte código é uma implementação em Python do algoritmo da Bissecção. As variáveis de entrada são:

- f - função
- a - extremo esquerdo do intervalo de inspeção $[a,\ b]$
- b - extremo direito do intervalo de inspeção $[a,\ b]$
- tol - tolerância (tipo de tolerância: convergência da função)
- n - número máximo de iterações

A variável de saída é:

- p - aproximação da raiz de f, isto é, $f(p) \approx 0$.

Tabela 3.2.5.1: Versão simplificada do método da Bissecção em Python

PASSO 1: *importamos a biblioteca "math".* ***PASSO 2:*** *começamos com i = 0.* ***PASSO 3:*** *enquanto não chegarmos ao número máximo de iterações (n):* ***PASSO 4:*** *o programa apenas irá continuar se o* ***Teorema de Bolzano*** *for cumprido.* ***PASSO 5:*** *criamos o ponto médio m.* ***PASSO 6:*** *se o valor da tolerância for superior a |b-a|/2 (se esta condição for quebrada, o programa salta para o* ***PASSO 8****.* ***PASSO 7:*** *devolve o último m obtido (raiz) e o programa termina.* ***PASSO 8:*** *caso contrário:* ***PASSO 9:*** *se f(a)*f(m) for positivo, voltamos ao PASSO 3 mas desta vez no intervalo [m,b]* ***PASSO 10****: caso contrário, voltamos ao* ***PASSO 3*** *mas desta vez no intervalo [a,m]*	*import math* *def* ***Bissecção*** *(a,b,tol,n,f):* *for i in range(n+1):* *assert f(a)*f(b) <= 0* *m = (a+b)/2.0* *if abs(b-a)/2.0 < tol:* *return m* *if f(a)*f(m) > 0:* *a = m* *else:* *b = m*

Outros critérios de paragem podem ser utilizados no **PASSO 6** mas iriam levar a vários problemas que poderiam vir a prejudicar o resultado final deste método. Por exemplo, nós podemos selecionar a tolerância como $\varepsilon > 0$ e geramos $m_1, \ldots, m_n$ até que uma das condições seja cumprida:

$$|m_n - m_{n-1}| < \varepsilon \qquad (3.2.5.1)$$

$$\frac{|m_n - m_{n-1}|}{|m_n|} < \varepsilon, \quad m_n \neq 0 \qquad (3.2.5.2)$$

$$|f(m_n)| < \varepsilon \qquad (3.2.5.3)$$

Infelizmente, podem aparecer certas dificuldades se utilizarmos algum destes critérios. Por exemplo, podem ocorrer sequências $\{m_n\}_{n=0}^{\infty}$ com a propriedade de que as diferenças $m_n - m_{n-1}$ convergem para zero, enquanto a própria sequência diverge. Também é possível que $f(m_n)$ esteja próximo de zero, mesmo quando m_n for significativamente diferente de s. Sem outras informações sobre f ou m, a segunda desigualdade é o melhor critério de paragem a ser aplicado, pois é o que mais se aproxima de testar o erro relativo.

É também importante realçar a importância de definir um limite superior para o número de iterações, pois assim é possível evitar a ocorrência de um ciclo infinito, algo que pode ocorrer quando a sequência diverge (e também quando a codificação do programa é incorreta).

3.2.6 Aplicação

Vamos considerar a equação $x^3 - x - 3 = 0$ para permitir o uso do método da bissecção utilizando a função definida por $f(x) = x^3 - x - 3$. Na Tabela 3.2.6.1 pode observar-se o desenvolvimento do método com a apresentação de 14 iterações, desde o intervalo $[0,\ 2]$. Este é um intervalo aceitável para iniciar o processo, uma vez que são satisfeitas as condições do **Teorema de Bolzano**, ou seja, $f(0) \times f(2) < 0$, garantindo assim a existência de uma solução.

Tabela 3.2.6.1: Aplicação do método da Bissecção em Python

Iteração	a	b	PM(m)	|b-a|/2	Tolerância	f(m)
0	0	2	1	1	0,0001	-3
1	1	2	1,5	0,5	0,0001	-1,125
2	1,5	2	1,75	0,25	0,0001	0,609375
3	1,5	1,75	1,625	0,125	0,0001	-0,3339843
4	1,625	1,75	1,6875	0,0625	0,0001	0,1179199
5	1,625	1,6875	1,65625	0,03125	0,0001	-0,1128845
6	1,65625	1,6875	1,671875	0,015625	0,0001	0,0012931
7	1,65625	1,671875	1,6640625	0,0078125	0,0001	-0,0561003
8	1,6640625	1,671875	1,6679687	0,0039062	0,0001	-0,0274799
9	1,6679687	1,671875	1,6699218	0,0019531	0,0001	-0,0131124
10	1,6699218	1,671875	1,6708984	0,0009765	0,0001	-0,0059144
11	1,6708984	1,671875	1,6713867	0,0004882	0,0001	-0,0023118
12	1,6713867	1,671875	1,6716308	0,0002441	0,0001	-0,0005096

13	1,6716308	1,671875	1,6717529	0,000122	0,0001	0,0003917
14	1,6716308	1,6717529	1,6716918	6,1035156	0,0001	-5,97E-05

Podemos assim concluir que, segundo o método da bissecção, o zero da função f é 1.6716918. A evolução dos valores centrais m_i é representada graficamente na Figura 3.2.6.1. Estes valores parecem estabilizar entre 1.6 e 1.7.

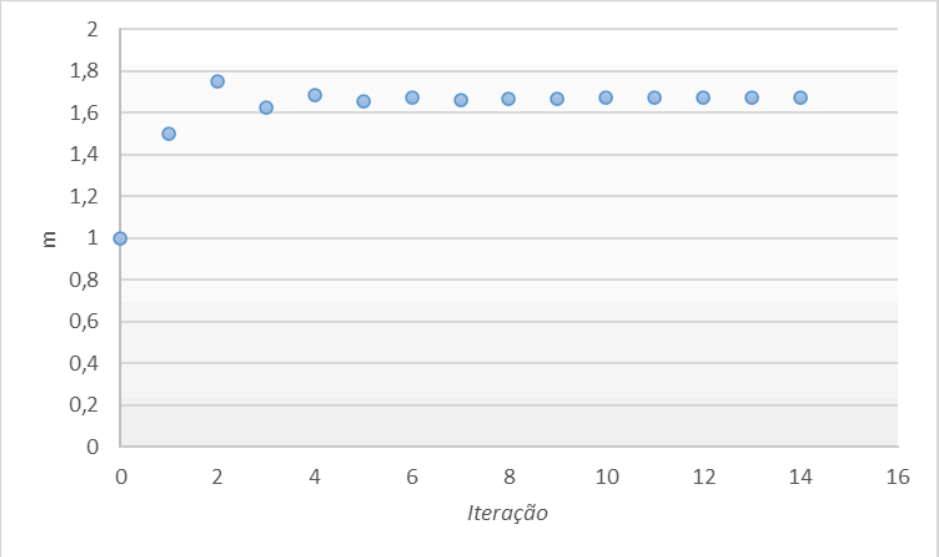


Figura 3.2.6.1: Evolução dos valores centrais (m) para o método da Bissecção

Podemos confirmar a conjectura da estabilização fazendo mais iterações, como mostrado na Tabela 3.2.6.2 e na Figura 3.2.6.2.

Tabela 3.2.6.2: Aplicação do método da Bissecção sem limite em Python

Iteração	a	b	PM(m)	|b-a|/2	Tolerância	f(m)
0	0	2	1	1	0,0001	-1
1	1	2	1,5	0,5	0,0001	0,875
2	1	1,5	1,25	0,25	0,0001	-0,296875
3	1,25	1,5	1,375	0,125	0,0001	0,2246093
4	…	…	…	…	…	…
5	…	…	…	…	…	…
22	1,6716995	1,6717	1,6716997	2,3841857	0,0001	-1,34E-06
23	1,6716997	1,6717	1,6716998	1,1920928	0,0001	-6,03E-07
24	1,6716998	1,6717	1,6716999	5,9604644	0,0001	1,35E-07

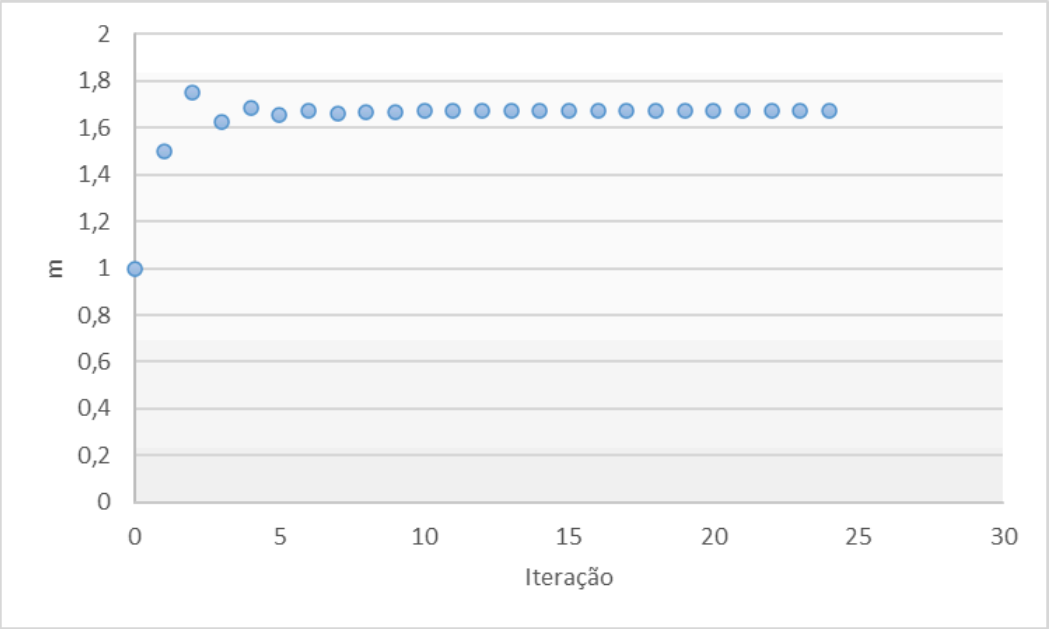


Figura 3.2.6.2: Evolução dos valores centrais (m) para o método da Bissecção sem limite

3.2.7 Considerações finais

Depois de analisarmos o método da bissecção podemos tirar as seguintes conclusões:

1) **As iterações não envolvem cálculos laboriosos**, devido à sua simplicidade este método acaba por ser fácil de utilizar;

2) **O método da bisseção é sempre convergente,** já que este método enquadra a raiz e os intervalos estão sempre contidos no intervalo inicial;

3) **O intervalo estudado é sempre reduzido para metade em cada iteração,** então, pode se garantir sempre o erro na solução da equação;

4) **Se ocorrer um erro de arredondamento,** mesmo que pequeno, no momento em que a máquina avalia o sinal da função no ponto médio, poderemos encontrar um intervalo que efetivamente não contém uma raiz, apesar de o método ser teoricamente seguro;

5) **Pode ser difícil encontrar um intervalo inicial** $[a,\ b]$**,** tal que $f(a) \times f(b) < 0$, em equações com raízes múltiplas ou muito próximas;

6) **A convergência é lenta,** pois se o intervalo inicial for muito comprido e se ε for muito pequeno, o número de iterações (k) tende a ser muito grande;

7) Muitas vezes este método é aplicado no início do processo de procura da solução. **No final deste processo é usado outro método mais rápido**;

8) **Incapaz de encontrar mais do que uma raíz, quando essas existem, dentro do intervalo [a,b]**, sendo que o método irá apenas se concentrar numa delas. Para se resolver este problema

pode-se até considerar um método da bissecção modificado em que se introduz um número natural k tal que o intervalo $[a,\ b]$ é dividido em k sub-espaços, podendo assim analisar a existência de uma solução em cada um deles. Quando não fosse encontrada nenhuma raiz no intervalo este seria rejeitado, este método só iria terminar quando já não houvesse mais intervalos para analisar. Mesmo assim o método não iria garantir a obtenção de todas as soluções de uma determinada equação já que é quase impossível garantir que num dado intervalo não há raízes (a não ser estudando o sinal de f').

3.3 Método da Falsa Posição

3.3.1 Descrição do método

Embora a bissecção seja um método absolutamente confiável para identificar raízes, a sua abordagem do tipo "força bruta" é relativamente ineficiente. Uma alternativa baseada na percepção gráfica é a falsa posição. Uma desvantagem do método da bissecção é que, na divisão do intervalo de $[a_k,\ b_k]$ em metades iguais, não são levados em conta os módulos de $f(a_k)$ e $f(b_k)$. Por exemplo, se $f(a_k)$ estiver muito mais próximo de zero do que $f(b_k)$, será provável que a raiz esteja mais próxima de a_k do que de b_k.

Um método alternativo que explora essa percepção gráfica é ligar $f(a_k)$ e $f(b_k)$ por uma reta. Uma estimativa refinada da raiz é representada pela intersecção dessa reta com o eixo x. O nome, **método da falsa posição**, ou, em latim, ***regula falsi***, vem do facto de usar uma reta em vez de uma curva para produzir uma "falsa posição" da raiz. Ele também é conhecido como técnica de interpolação linear.

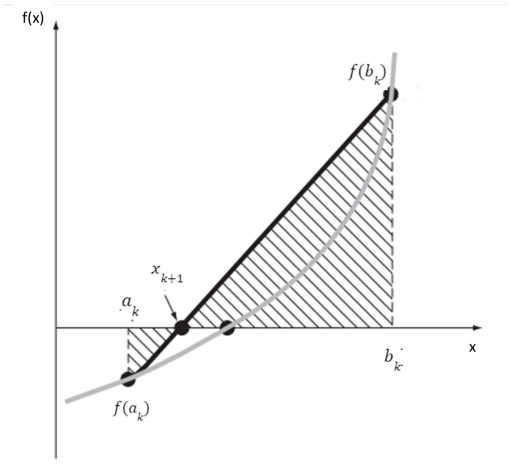


Figura 3.3.1.1: Método da Falsa Posição

Usando triângulos semelhantes (Figura 3.3.1.1), a equação desta secante pode escrever-se nas seguintes formas alternativas

$$y = f(a_k) + f[a_k,\ b_k](x - a_k) \qquad (3.3.1.1)$$

$$y = f(b_k) + f[a_k,\ b_k](x - b_k) \qquad (3.3.1.2)$$

pelo que a sua interseção com eixo dos xx ocorre no ponto x_{k+1} dado por qualquer das seguintes expressões

$$x_{k+1} = a_k - \frac{f(a_k)}{f[a_k, b_k]} \qquad (3.3.1.3)$$

$$x_{k+1} = b_k - \frac{f(b_k)}{f[a_k, b_k]} \qquad (3.3.1.4)$$

$$x_{k+1} = \frac{a_k \times f(b_k) - b_k \times f(a_k)}{f(b_k) - f(a_k)} \qquad (3.3.1.5)$$

Como se adota o mesmo princípio na escolha de subintervalos, $f(a_k)$ e $f(b_k)$ possuem sinais diferentes, e, por conseguinte, o denominador nunca poderá ser nulo. Este facto implica que não são de recear efeitos negativos resultantes dos erros de arredondamento numa subtração de dois números tendencialmente muito pequenos e semelhantes em valor absoluto, ou seja, não é preciso recear o aparecimento de cancelamento subtrativo.

É importante também realçar, apesar de intuitivo, que $x_{k+1} \in \ ]a_k,\ b_k[$. Suponhamos que $f(a_k) < 0$ e que $f(b_k) > 0$. Verificamos que

$$a_k < x_{k+1} = \frac{a_k \times f(b_k) - b_k \times f(a_k)}{f(b_k) - f(a_k)} < b_k \qquad (3.3.1.6)$$

é equivalente a (porque $f(b_k) > f(a_k)$)

$$f(b_k)a_k - f(a_k)a_k < f(b_k)a_k - f(a_k)b_k < f(b_k)b_k - f(a_k)b_k, \qquad (3.3.1.7)$$

ou seja,

$$f(a_k)(b_k - a_k) < 0, \qquad 0 < f(b_k)(b_k - a_k) \qquad (3.3.1.8)$$

Como $b_k > a_k$, então por hipótese, estas condições são verificadas (o mesmo iria acontecer na outra situação possível, $f(a_k) > 0$ e $f(b_k) < 0$)

3.3.2 Erro e Convergência

O teorema seguinte fornece condições suficientes para a convergência do método da falsa posição e uma estimativa de erro

Teorema 3.3.2.1 Seja f uma função convexa ou côncava no intervalo $I = [a,\ b]$ com $f(a)$ e $f(b)$ de sinais diferentes. Então, o método da falsa posição converge linearmente para a raiz de f neste intervalo.

Demonstração: Vamos desenvolver a demonstração para o caso de f ser convexa e crescente, já que para funções côncavas o tratamento é inteiramente análogo. A sucessão x_k determinada pelo algoritmo é não-decrescente e limitada à direita por b. Logo, converge para um limite $\bar{x}$ e, se este valor verificar $f(\bar{x}) = 0$, o método da falsa posição converge para o único zero de f no intervalo $[a, b]$. Vejamos se assim acontece ou não.

Suponhamos que o limite $\bar{x}$ não verifica $f(\bar{x}) = 0$. Então, existem valores de N e $\epsilon > 0$ tais que

$$f(x_n) <- \epsilon \qquad para \qquad n > N \qquad (3.3.2.1)$$

Por outro lado, tira-se que

$$x_{n+1} - b = \frac{f(b)}{f[b, x_n]} = (b - x_n)\frac{f(b)}{f(b)-f(x_n)} \qquad (3.3.2.2)$$

Como $M = f(b) > 0$, então

$$|x_{n+1} - b| \leq \frac{M}{M+\epsilon}|x_n - b| \qquad (3.3.2.3)$$

e, por aplicação repetida desta expressão, deduzimos que

$$|x_{k+n} - b| \leq \left(\frac{M}{M+\epsilon}\right)^k |x_n - b| \qquad (3.3.2.4)$$

pelo que fomos forçados a concluir que

$$\bar{x} = \lim_{k \to \infty} x_k = b \qquad (3.3.2.5)$$

Mas as hipóteses do teorema e a própria construção do algoritmo garantem que se verifica $f(x_k)f(b) \leq 0$ e, por conseguinte,

$$0 \geq \lim_{k \to \infty} f(x_k)f(b) = [f(b)]^2 > 0 \qquad (3.3.2.6)$$

o que é uma contradição. Portanto, a sucessão x_k tem de convergir para o zero da função f no intervalo $[a, b]$.

Passamos agora para o estudo do **erro**, utilizando o teorema de erro da interpolação (**Teorema 2.1.7**), sendo s raiz de f, podemos escrever que

$$f(s) = 0 = f(b) + f[b, x_k](s - b) + f[b, x_k, s](s - b)(s - x_k) \qquad (3.3.2.7)$$

Por outro lado, vimos no início deste capítulo que

$$f(x_{k+1}) = 0 = f(b_k) + f[x_k, b_k](x_{k+1} - b_k) \qquad (3.3.2.8)$$

Subtraindo estas últimas duas equações membro a membro e tendo em atenção que $b = b_k$, facilmente obtemos

$$e_{k+1} = \frac{f[b, x_k, s]}{f[b, x_k]}(b - s)e_k \qquad (3.3.2.9)$$

Passando ao limite,

$$\lim_{k \to \infty} \frac{|e_{k+1}|}{|e_k|} = \lim_{k \to \infty} \left|\frac{f[b, x_k, s]}{f[b, x_k]}(b - s)\right| = \left|\frac{f[b, s, s]}{f[b, s]}(b - s)\right| = c \qquad (3.3.2.10)$$

A constante de erro assimptótico c é, como vamos provar, inferior a um. Desenvolvendo as diferenças divididas, obtemos as expressões

$$f[b, s] = \frac{f(s) - f(b)}{s - b} \qquad (3.3.2.11)$$

$$f[b, s, s] = \frac{f[s, s] - f[b, s]}{s - b} \qquad (3.3.2.12)$$

chegando por fim à conclusão que

$$c = 1 - \frac{f[s, s]}{f[b, s]} = 1 - \frac{f'(s)}{f'(\xi)} \qquad (3.3.2.13)$$

com $\xi \in (s, b)$. Como, por hipótese, f é crescente e convexa, podemos afirmar que f' é crescente e positiva, logo, $0 \neq f'(s) < f'(\xi)$, o que implica imediatamente que $c < 1$. Fica assim concluída a demonstração.

3.3.3 Estimativa do número de iterações

Considerando uma precisão ε e um intervalo inicial $[a, b]$ é possível saber quantas iterações (n) são efetuadas pelo método da falsa posição até que se obtenha $|a - b| \leq \varepsilon$, usando o algoritmo deste método.

$$\frac{log(b - a) - log(2\varepsilon)}{log(2)} < n \qquad (3.3.3.1)$$

3.3.4 Tempo de cálculo do método

Tal como o método da bissecção, em cada iteração a função é apenas calculada uma vez, no ponto x_{k+1}, a partir dos valores $f(a_k)$, $f(b_k)$ guardados. Normalmente, o tempo de cálculo da função é muito maior que o tempo de cálculo das operações elementares (três subtrações, uma divisão e uma multiplicação) que será designado ε. Sendo t_f o tempo de cálculo médio da função f, o tempo de cálculo total, ao fim de n iterações, será aproximadamente

$$T = n(t_f + \varepsilon) \approx nt_f \qquad (3.3.4.1)$$

desprezando o tempo de cálculo para as operações elementares.

3.3.5 Algoritmo do método

O seguinte código é uma implementação em Python do algoritmo da Falsa Posição. As variáveis de entrada são:

- f - função
- a - extremo esquerdo do intervalo de inspeção $[a,\ b]$
- b - extremo direito do intervalo de inspeção $[a,\ b]$
- tol - tolerância (tipo de tolerância: convergência da função)
- n - número máximo de iterações

A variável de saída é:

- p - aproximação da raiz de f, isto é, f(p)≈0.

Tabela 3.3.5.1: Versão simplificada do método da Falsa posição

PASSO 1: *importamos a biblioteca "math".* ***PASSO 2:*** *enquanto não chegarmos ao número máximo de iterações (n):* ***PASSO 3:*** *o programa apenas irá continuar se o* ***Teorema de Bolzano*** *for cumprido.* ***PASSO 4:*** *criamos o ponto x0.* ***PASSO 5:*** *se o valor da tolerância for superior a |f(x0)| (se esta condição for quebrada, o programa salta para o* ***PASSO 7):*** ***PASSO 6:*** *devolve o último x0 obtido (raiz) e o programa termina.* ***PASSO 7:*** *caso contrário:* ***PASSO 8:*** *se f(a)*f(x0) for negativo, voltamos ao* ***PASSO 3*** *mas desta vez no intervalo [a,x0]* ***PASSO 9:*** *caso contrário, voltamos ao* ***PASSO 3*** *mas desta vez no intervalo [x0,b]*	*import math* *def* ***FalsaPosição****(a,b,tol,n,f):* *for i in range(n+1):* *assert f(a)*f(b) <= 0* *x0 = (a*f(b) - b*f(a))/(f(b)-f(a))* *if abs(f(x0)) < tol:* *return x0* *if f(a)*f(x0) < 0:* *b = x0* *else:* *a = x0*

3.3.6 Aplicação

Vamos considerar a equação $x^3 + 2x^2 - 3x - 1 = 0$ para permitir o uso do método da falsa posição utilizando a função definida por $f(x) = x^3 + 2x^2 - 3x - 1$. Pode observar-se o desenvolvimento do método, desde o intervalo $[1,\ 1.3]$, com a apresentação de 3 iterações na Tabela 3.3.6.1. Este é um intervalo aceitável para iniciar o processo, uma vez que este satisfaz as condições do **Teorema de Bolzano**, ou seja, $f(1) \times f(1.3) < 0$, garantindo assim a existência de uma solução.

Tabela 3.3.6.1: Aplicação do método da Falsa Posição em Python

Iteração	a	b	f(a)	f(b)	x0	Tolerância	f(x0)
0	1	1,3	-1,00	0,6769	1,1788	0,0001	-0,11870
1	1,1788	1,3	-0,12	0,6769	1,1969	0,0001	-0,01057
2	1,1969	1,3	-0,01	0,6769	1,1985	0,0001	-0,00092
3	1,1985	1,3	0,00	0,6769	1,1986	0,0001	-0,00008

Podemos assim concluir que, segundo o método da falsa posição, o zero da função f é 1.1986. A evolução dos valores centrais x_0 é representada graficamente na Figura 3.3.6.1.

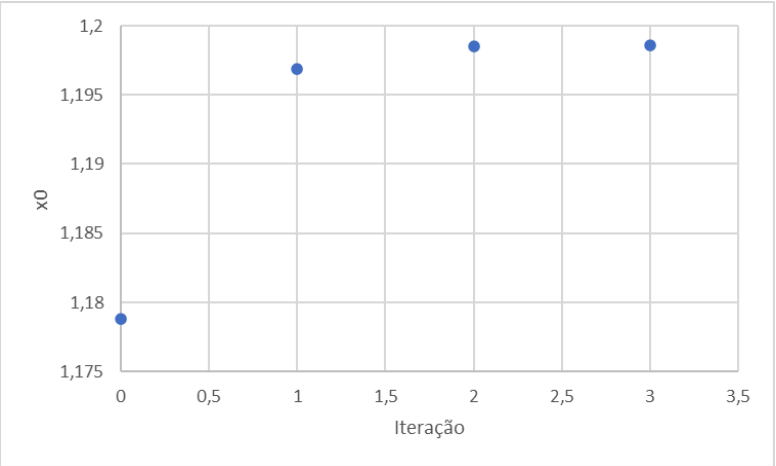


Figura 3.3.6.1: Evolução dos valores centrais para o método da Falsa Posição sem limite

3.3.7 Considerações finais

Depois de analisarmos o método da falsa posição podemos tirar as seguintes conclusões:

1) **Estabilidade e convergência** para a solução procurada;

2) **Desempenho regular e consistente;**

3) **Lentidão do processo de convergência** (que exige o cálculo de f(x) em um elevado número de iterações);

4) **Necessidade de conhecimento prévio** da região na qual se encontra a raiz de interesse (o que nem sempre é possível).

5) **Se f é uma função côncava ou convexa em $[a,\ b]$,** então uma das extremidades permanece fixa no método da falsa posição. (vamos analisar isto com mais detalhe)

Proposição 3.3.7.1 Consideremos f, com $f \in C^2(V_s)$, em que V_s é uma vizinhança da raiz. Se $f''(s) \neq 0$, então, a partir de certa ordem, um dos extremos do intervalo $[a_k,\ b_k]$, definido pelo método da falsa posição, fica fixo.

Demonstração: Seja $a = lim\, a_k$, $b = lim\, b_k$. Quando $b \neq a = s$, isso implica que a sucessão b_k seja constante a partir de certa ordem, por que se não tomaria valores x_k, que convergem para s e teríamos $b = s$. Situação semelhante ocorre quando $a \neq b = s$, sendo neste caso a_k constante a partir de certa ordem.

Resta ver que a hipótese $a = b = s$ não pode ocorrer sob as hipóteses da proposição. Como $f(a)f(b) < 0$ e há apenas uma raiz, só podemos ter $f'(s) = 0$ se $f''(s) = 0$, caso excluído.

Portanto $f'(s) \neq 0$ e então uma vizinhança $V_{\varepsilon'}(s)$ a função f' (que é contínua) tem o mesmo sinal que $f'(s)$. Da mesma forma, uma vizinhança $V_{\varepsilon''}(s)$, na função f'' terá o mesmo sinal que $f''(s)$. Assim considerando $V_s = V_{\varepsilon''}(s)$, o sinal de $\frac{f''}{2f'}$ é fixo em V_s.

Por outro lado, se consideramos que

$$e_{k+1} = -\frac{f''(\eta_k)}{2f'(\xi_k)}(s - a_k)(s - b_k) \qquad (3.3.7.1)$$

o sinal de $e_{k+1} = s - x_{k+1}$ é igual ao sinal de $\frac{f''(\eta_k)}{2f'(\xi_k)}$, porque o valor $(s - a_k)(s - b_k)$ é sempre negativo.

Se, por absurdo, $a = b = s$, temos $a_k,\ b_k \to s$. Notando que $\xi_k, \eta_k \in]a_k,\ b_k[$ temos $\xi_k, \eta_k \in V_z$, para k suficientemente grande. Isto significa que o sinal de e_{k+1} é fixo, ou seja, ou x_{k+1} é sempre maior que s, ou será sempre menor. Isto significa que um dos extremos fica fixo, não podendo convergir para s, o que contradiz $a = b = s$.

A condição de f'' se anular na raiz raramente acontecerá, e assim o método da falsa posição, a partir de certa ordem p, um dos extremos de $[a_k,\ b_k]$ irá ficar imobilizado. Temos assim duas hipóteses:

1) Se $f''(s)f'(s) > 0$, ou que é equivalente, $f''(s)f(b_p) > 0$ (pois o sinal de b_p é igual ao de f'), então as iterações x_{p+1}, x_{p+2}, … vão ficar à esquerda da raiz significando isso que temos $b_p = b_{p+1} = b_{p+2} =$ …

2) Se $f''(s)f'(s) < 0$, ou que é equivalente, $f''(s)f(a_p) > 0$ (pois o sinal de a_p é o oposto de f'), então $a_p = a_{p+1} = a_{p+2} =$ …

Resumindo, se f'' tiver sinal constante num intervalo $[a_p, b_p]$ então o extremo do intervalo para o qual f possui esse sinal, permanece fixo.

3.3.8 Método da falsa posição modificada

Trata-se de uma pequena modificação no método anterior de forma a evitar que um dos extremos se imobilize permanentemente. A ideia principal é dividir o valor da função por 2 no extremo que se imobiliza.

Suponhamos que se imobilizou o extremo a_k, tendo-se $a_{k+1} = a_k$, neste caso consideramos $F_a = \frac{f(a_k)}{2}$ mantendo $F_b = f(b_k)$ e calculamos

$$\hat{x}_{k+1} = b_k - F_b \frac{b_k - a_k}{F_b - F_a}, \qquad (3.3.8.1)$$

o valor $\hat{x}_{k+1}$ ainda pertence ao intervalo $[a_k, b_k]$ mas é menor que o valor x_{k+1} calculando usando a fórmula do método da falsa posição original, que neste caso seria maior que a raiz, ou seja $z < x_{k+1}$. Portanto como $\hat{x}_{k+1} < x_{k+1}$ é provável que $\hat{x}_{k+1} < z$, passando neste caso a ter-se $\hat{x}_{k+1} = a_{k+1}$.

Se isso não acontecer, podemos ainda considerar um $\hat{F}_{a+1} = \frac{F_a}{2}$ que levaria a um $\hat{\hat{x}}_{k+1}$ ainda mais pequeno…, e assim sucessivamente até que obtivéssemos um valor $\hat{\hat{x}}_{k+1} < z$.

Desta forma consegue-se uma maior rapidez de convergência.

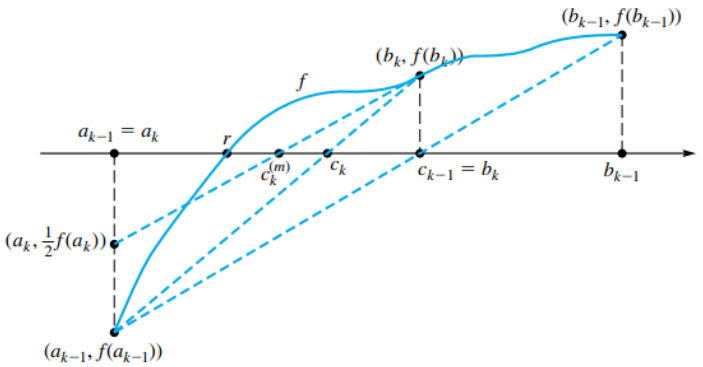


Figura 3.3.8.1: Método da Falsa Posição modificado

Para este método não iremos criar um algoritmo já este não será analisado com tanta profundidade quanto os outros nem será comparado com os restantes.

Não iremos apresentar também estimativas de erro específicas para este método, mas podemos sempre recorrer ao teorema que apresentamos inicialmente e que nos dá estimativas de erro a posteriori elementares, baseadas no teorema do valor médio.

Quanto ao tempo de cálculo, será semelhante ao método da falsa posição, já que a divisão por 2 não é significativa, compensando em termos de eficácia, já que irá evitar que um dos extremos fique constante, acelerando a convergência.

3.4 Método do Ponto Fixo

3.4.1 Descrição do método

Nesta secção, consideremos a determinação das soluções dos problemas de ponto fixo e a relação entre estes e os problemas de determinação de raiz que desejamos resolver.

Os resultados de ponto fixo ocorrem em muitas áreas da matemática e são uma das principais ferramentas utilizadas pelos economistas para fornecer resultados relativos ao equilíbrio. Embora a ideia básica da técnica seja antiga, a terminologia foi empregada pela primeira vez pelo matemático holandês **L.E.J. Breouwer (1881-1966)**, no início do século passado.

Todos os métodos estudados até agora reportavam à equação não linear $f(x) = 0$, mas para este método vamos ter de considerar a seguinte forma

$$x = g(x). \qquad (3.4.1.1)$$

É importante realçar que este modo de escrever a equação não introduz qualquer restrição relativamente ao caso $f(x) = 0$, pois é sempre possível transformar esta equação numa do tipo $x = g(x)$, pondo, por exemplo, $g(x) = x + f(x)$. Esta não é a única escolha possível para a função g, poderíamos também escolher, por exemplo, $g(x) = x - f(x)$, ou outras.

Diferentes escolhas para a função g, representando a mesma equação $f(x) = 0$, dão lugar a métodos do ponto fixo com diferentes níveis de eficiência (ver também a secção 3.4.7).

A equação escrita na forma $x = g(x)$ sugere imediatamente o seguinte esquema iterativo

$$x_{k+1} = g(x_k), \qquad k = 0, 1, \ldots \qquad (3.4.1.2)$$

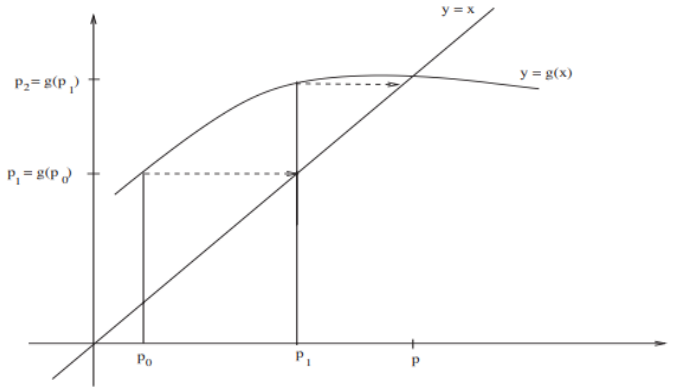


Figura 3.4.1.1: Método do ponto fixo - caso de convergência

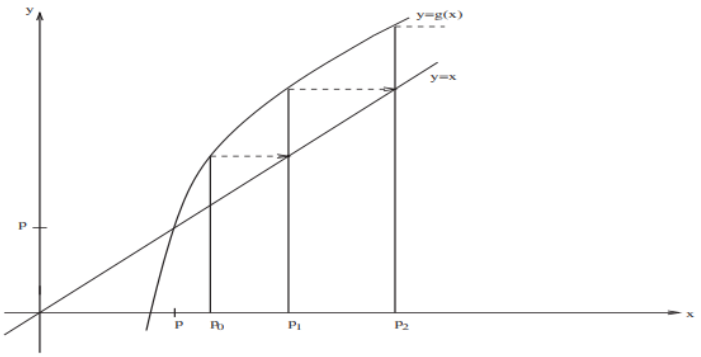


Figura 3.4.1.2: Método do ponto fixo - caso de divergência

A Figura 3.4.1.1 e 3.4.1.2 mostram uma representação geométrica deste processo iterativo, exibindo dois casos distintos (esta diferença de comportamento será explicada mais à frente):

a) um em que se verifica convergência
b) um em que se verifica divergência

3.4.2 Visualização do método iterativo

É importante ter-se uma noção visual deste processo de iteração de uma função φ, e para isso veremos como fazer iterações usando apenas o esboço da função.

A primeira coisa que devemos fazer é desenhar o gráfico da função, e em seguida a bissetriz, que nos auxiliará. Depois escolhemos uma condição inicial x_0, o objetivo é encontrar a posição, na abscissa, de $x_1 = \varphi(x_0)$.

Movendo-nos verticalmente, encontraremos o gráfico de φ, na posição $(x_0, \varphi(x_0))$. Como $\varphi(x_0) = x_1$, este é o ponto (x_0, x_1), ou seja, já encontramos x_1, mas ele é a segunda coordenada do ponto encontrado. O nosso objetivo, no entanto, é encontrar o ponto de coordenadas $(x_1, 0)$.

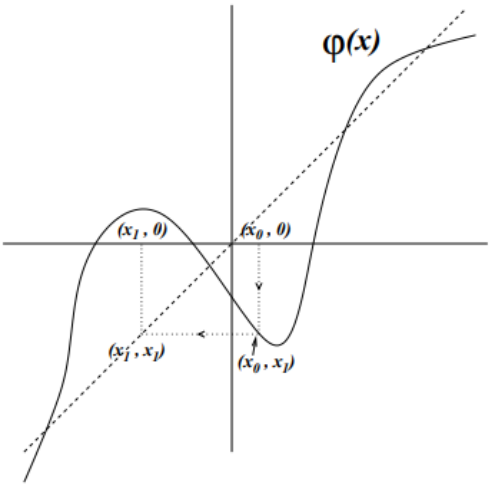


Figura 3.4.2.1: Visualização do método iterativo

Então movemo-nos horizontalmente, isto é, mantendo fixa a segunda coordenada, a partir de (x_0, x_1) até encontrar a bissetriz. Na bissetriz, os valores da primeira e da segunda coordenada são iguais; como a segunda foi mantida sempre igual a x_1, então esse ponto será (x_1, x_1), e com um movimento vertical determinamos x_1 sobre a abscissa.

Para a determinação de x_2 o procedimento é análogo: movimento vertical até encontrar o gráfico, depois movimento horizontal até encontrar a abscissa e finalmente movimento vertical até encontrar a abscissa. E assim por diante!

Observamos que podemos poupar um pouco de trabalho quando fazemos uma série de iterações sucessivas. De x_0 vamos verticalmente até o gráfico (à altura x_1), depois horizontalmente até a abcissa (à posição horizontal x_1).

Depois, de acordo com o que foi descrito acima, iríamos verticalmente até a abcissa (à altura zero) e então verticalmente até o gráfico (à altura $x_2 = \varphi(x_1)$). Ora, a composição de dois movimentos verticais ainda é um movimento vertical, que poderia ser feito de uma vez só. Ou seja, logo após nos movermos horizontalmente até a bissetriz, encontrando (x_1, x_1), podemos nos mover verticalmente até o gráfico, encontrando (x_1, x_2). Em seguida continuamos, indo horizontalmente até a bissetriz, no ponto (x_2, x_2), e depois verticalmente até o gráfico, no ponto (x_2, x_3), e assim por diante.

Coletando as primeiras coordenadas de cada ponto de encontro com o gráfico, teremos a sequência de iterados a partir de x_0. Veja na figura abaixo uma ilustração desse procedimento.

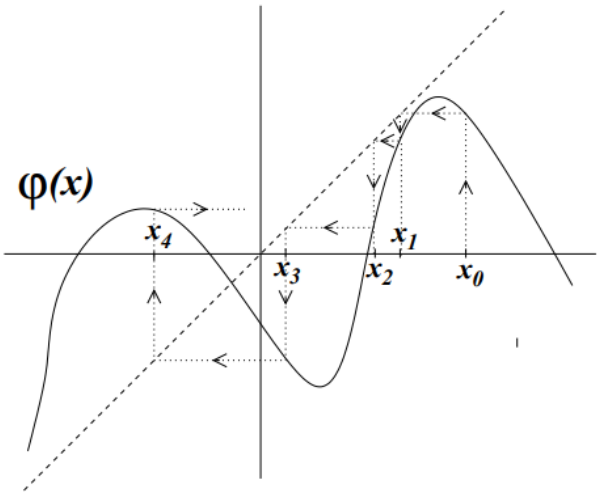


Figura 3.4.2.2: Procedimento do método do ponto fixo

3.4.3 Erro e Convergência

Antes de passarmos ao estudo da convergência deste método é necessário introduzir a seguinte definição.

Definição 3.4.3.1 Uma função g diz-se contractiva no intervalo $I = [a, b]$ se existir uma constante M com $0 \leq M < 1$ tal que

$$|g(x_1) - g(x_2)| \leq M|x_1 - x_2| \qquad (3.4.3.1)$$

para todos os $x_1, x_2 \in I$.

É importante notar que a própria formulação desta definição implica que uma função contractiva é necessariamente contínua. A verificação da contractividade nem sempre é fácil. Um dos casos mais simples e simultaneamente dos mais interessantes na prática é quando $g \in C^1(I)$ e em que a aplicação do teorema do valor médio nos conduz a

$$g(x_1) - g(x_2) = g'(\xi)(x_1 - x_2), \qquad com \qquad \xi \in inter(x_1, x_2) \tag{3.4.3.2}$$

Portanto, se $|g'(\xi)| < 1$ para todo o $\xi \in I$, a função g é contractiva. Nestas condições podemos tomar para M o valor

$$M = max\,|g'(x)|, \qquad x \in I \tag{3.4.3.3}$$

Teorema 3.4.3.1 Se existir um intervalo $I = [a, b]$ tal que $g(I) \subset I$ e relativamente ao qual a função g é contractiva, então esta função possui um único ponto fixo neste intervalo, e a sucessão x_k, acima definida, converge para este ponto fixo qualquer que seja a estimativa inicial $x_0 \in I$.

Demonstração: Vamos começar por demonstrar que as condições enunciadas garantem a existência de ponto fixo. Introduzimos a função auxiliar $h(x) = g(x) - x$, que satisfaz, uma vez que $g(I) \subset I$, as desigualdades óbvias

$$h(a) = g(a) - a \geq 0 \qquad e \qquad h(b) = g(b) - b \leq 0 \tag{3.4.3.4}$$

Como h é contínua em I, pelo teorema do valor intermédio, existe (pelo menos) um valor s tal que $h(s) = g(s) - s = 0$.

Fica deste modo provada a existência de (pelo menos) um ponto fixo de g, pelo que podemos passar à demonstração da sua unicidade. Suponhamos que existem dois pontos fixos, s e s'. Então, devemos ter, por definição, que

$$s = g(s) \qquad e \qquad s' = g(s') \tag{3.4.3.5}$$

Subtraindo membro a membro estas desigualdades, tomando valores absolutos e majorando, vem que

$$|s - s'| = |g(s) - g(s')| \leq M|s - s'| \tag{3.4.3.6}$$

Esta relação implica imediatamente que $(1 - M)|s - s'| \leq 0$, e, portanto, como $0 \leq M < 1$, concluímos que $s = s'$, ou seja, o ponto fixo é único.

Estudemos em seguida a convergência do método iterativo. O erro $e_k = s - x_k$ satisfaz as relações evidentes

$$|e_k| = |s - x_k| = |g(s) - g(x_{k-1})| \leq M|s - x_{k-1}| = M|e_{k-1}| \tag{3.4.3.7}$$

Aplicando esta relação repetidamente, temos que

$$|e_k| \leq M|e_{k-1}| \leq M^2|e_{k-2}| \leq \ \ldots \ \leq M^k|e_0| \qquad (3.4.3.8)$$

e, por conseguinte, uma vez que $M < 1$, $\lim_{k \to \infty} |e_k| = 0$. Concluímos assim que o método do ponto fixo é convergente, independentemente da escolha de $x_0 \in I$.

Se a função $g \in C^1(I)$, a expressão anterior pode escrever-se da seguinte forma

$$|e_k| = |g'(\xi_k)||s - x_k| = |g'(\xi_k)||e_{k-1}| \qquad com \qquad \xi_k \in inter(x_k, s) \qquad (3.4.3.9)$$

e no caso de se verificar convergência,

$$\lim_{k \to \infty} \frac{|e_k|}{|e_{k-1}|} = \lim_{k \to \infty} |g'(\xi_k)| = |g'(s)| \qquad (3.4.3.10)$$

Portanto, se $g'(s) \neq 0$, a convergência do método é linear. Por outro lado, se $g'(s) = 0$, obtemos uma convergência quadrática conforme iremos provar na secção 3.5.2.

Um ponto fixo que possua uma vizinhança na qual g é contractiva e para o qual o processo iterativo é convergente diz-se que é um **ponto de atração** (ou atractor) da função g; caso contrário, diz-se que é um **ponto de repulsão** (ou repulsor).

Um ponto fixo atractivo é como um "buraco negro", ou seja, as iterações que se aproximem demasiado dele são inexoravelmente atraídas.

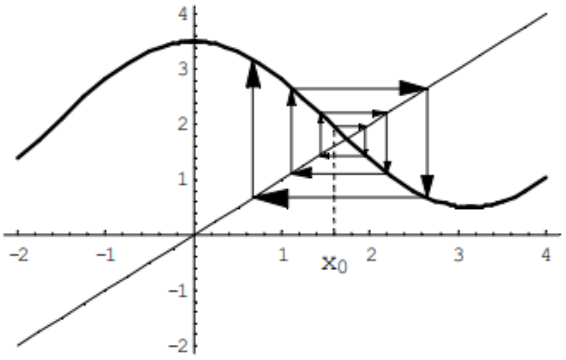


Figura 3.4.3.1: Ponto de repulsão

Teorema 3.4.3.2 (não convergência para um ponto fixo) Seja V_z vizinhança de um ponto fixo z de $g \in C^1(V_z)$, tal que $|g'(z)| > 1$. Neste caso, a sucessão $x_{k+1} = g(x_k)$ não pode convergir para esse ponto fixo z (exceto se, excepcionalmente, $x_m = z$ para algum m).

Demonstração: Supondo por absurdo, que x_k converge para o ponto fixo $z \in V_z$, então:

$$\forall \varepsilon > 0 \ \forall k \in \mathbb{N} \ \exists p: \ k \geq p: \ |z - x_k| < \varepsilon \qquad (3.4.3.11)$$

Como $|g'(z)| > 1$, podemos sempre considerar um $\varepsilon > 0$ suficientemente pequeno tal que $I = [z - \varepsilon, z + \varepsilon] \subset V_z$ em que $|g'(x)| > 1, \ \forall x \in I$. Aplicando o teorema do valor médio

$$|z - x_{k+1}| = |g'(\xi_k)||z - x_k|. \qquad (3.4.3.12)$$

como x_k, $z \in I$ também $\xi_k \in I$, logo $|g'(x)| > 1$ e temos $|z - x_{k+1}| > |z - x_k|$ para $k \geq p$, pois $z \neq x_k$. Isto significa que a sucessão $|z - x_k|$ é crescente e consequentemente não converge para zero, o que provoca contradição.

Teorema 3.4.3.3 (ordem de convergência do método do ponto fixo) Seja g uma função $C^p(V_z)$, com $p \geq 2$, onde V_z é uma vizinhança de z ponto fixo de g.
Se

$$g'(z) = \ldots = g^{(p-1)}(z) = 0, \text{ com } g^{(p)}(z) \neq 0 \qquad (3.4.3.13)$$

então

$$\lim_{m \to \infty} \frac{e_{k+1}}{e_k^p} = -\frac{(-1)^p}{p!} g^{(p)}(z) \qquad (3.4.3.14)$$

ou seja, o ponto fixo tem convergência de ordem p, e o coeficiente assimptótico é

$$K_\infty^p = \frac{|g^{(p)}(z)|}{p!}. \qquad (3.4.3.15)$$

Demonstração: Fazendo o desenvolvimento em série de Taylor, para um $x_k \in I$

$$g(x_k) = g(z) + g'(z)(x_k - z) + \ldots + \frac{g^{(p-1)}(z)}{(p-1)!}(x_k - z)^{p-1} + \frac{g^{(p)}(\xi_k)}{p!}(x_k - z)^p \qquad (3.4.3.16)$$

com $\xi_k \in \,]x_k, z[$. Usando as hipóteses, temos

$$x_{k+1} = g(x_k) = z + \frac{g^{(p)}(\xi_k)}{p!}(x_k - z)^p \qquad (3.4.3.17)$$

ou seja,

$$e_{k+1} = -(-1)^p \frac{g^{(p)}(\xi_k)}{p!}(e_k)^p. \qquad (3.4.3.18)$$

Concluímos, reparando que $\xi_k \to z$, logo $lim \frac{|e_{k+1}|}{|e_k^p|} = \frac{g^{(p)}(z)}{p!} = K_\infty > 0$.

3.4.4 Tempo de cálculo do método

O método do ponto fixo envolve apenas o cálculo de g em cada iterada, pelo que podemos escrever

$$T = nt_g \qquad (3.4.4.1)$$

É claro que, uma vez que podemos escolher diferentes funções g modelando a mesma equação, o tempo de cálculo t_g irá variar com a maior ou menor complexidade de g.

3.4.5 Algoritmo do método

O seguinte código é uma implementação em Python do algoritmo do Ponto Fixo. As variáveis de entrada são:

- g – função cujo ponto fixo pretendemos determinar
- x_0 – valor inicial
- tol - tolerância (tipo de tolerância: convergência da função)
- n - número máximo de iterações

A variável de saída é:

- p - aproximação do ponto fixo de g, isto é, $g(p) \approx p$.

Tabela 3.4.5.1: Versão simplificada do método do Ponto Fixo

PASSO 1: *importamos a biblioteca "math".*
PASSO 2: *começamos com i = 0.*
PASSO 3: *enquanto não chegarmos ao número máximo de iterações (n):*
PASSO 4: se o valor da tolerância for superior a |g(x0) - x0| (se esta condição for quebrada, o programa salta para o **PASSO 6**):
PASSO 5: devolve o último x0 obtido (ponto fixo) e o programa termina.
PASSO 6: calculamos o novo x0, que passa a ser g(x0) e regressamos ao **PASSO 3**.
	import math
	
	def **PontoFixo**(x0,tol,n,g):
	for i in range(n+1):
	if abs(g(x0)-x0) < tol:
	return x0
	else:
	x0 = g(x0)

3.4.6 Aplicação

Vamos agora aplicar o método do ponto fixo à solução da equação $x = cos(x)$ no intervalo $I = [a, b] = [0, \frac{\pi}{2}]$.

Vamos primeiro verificar se as condições suficientes de convergência se verificam. Pondo $g(x) = cos(x)$ e como

$$g(a) = cos(0) = 1 > 0 \qquad e \qquad g(b) = cos(\tfrac{\pi}{2}) = 0 < \tfrac{\pi}{2} \qquad (3.4.6.1)$$

e g é monótona no intervalo I, concluímos que $g(I) \subset I$. Além disso, como $g'(x) = -sin(x)$, temos que $|g'(x)| < 1$ numa vizinhança suficientemente pequena da raiz, vizinhança que neste caso podemos tomar como intervalo aberto $I = (a, b) = (0, \frac{\pi}{2})$. Estão assim reunidas todas as condições de convergência. O cálculo das sucessivas iterações fornece os resultados do quadro seguinte:

Tabela 3.4.6.1: Aplicação do método da Falsa posição

Iteração	x0	g(x0)	Tolerância
0	1	0,5403	1,00E-05
1	0,5403	0,85755	1,00E-05
2	0,85755	0,65428	1,00E-05
3	0,65428	0,79348	1,00E-05
4	0,79348	0,70136	1,00E-05
5	0,70136	0,76395	1,00E-05
6	0,76395	0,7221	1,00E-05
7	0,7221	0,75041	1,00E-05
…	…	…	…
25	0,73907	0,73909	1,00E-05
26	0,73909	0,73907	1,00E-05
27	0,73907	0,73908	1,00E-05
28	0,73908	0,73908	1,00E-05

Podemos assim concluir que, segundo o método do ponto fixo, o zero da função f é 0.73908. Na Figura 3.4.6.1 pode observar-se graficamente a evolução de x_0.

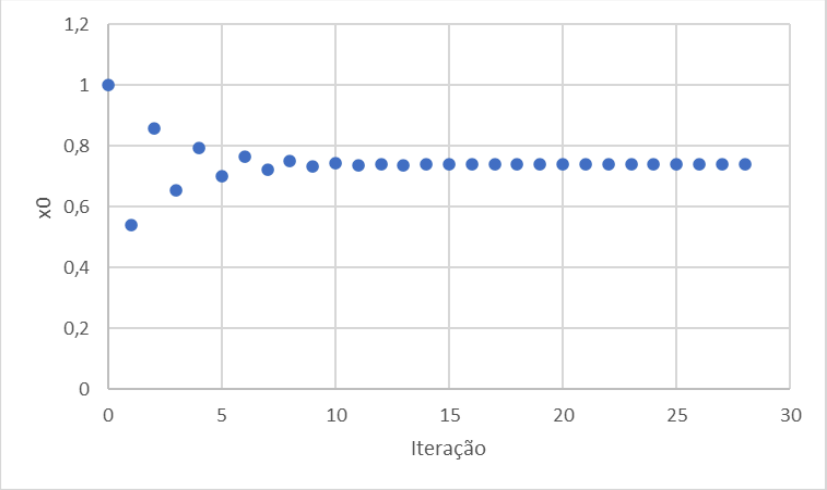


Figura 3.4.6.1: Evolução de estimativas no método do Ponto Fixo

3.4.7 Considerações finais

Depois de analisarmos o método do ponto fixo podemos tirar as seguintes conclusões:

1) **Simples e fácil de entender** – O Método do Ponto Fixo é direto e descomplicado, tornando-o fácil de ser entendido e aplicado por qualquer pessoa.

2) **Requer menos esforço computacional** – É menos exigente em termos de computação, o que significa que necessita de menos recursos e tempo para obter resultados.

3) **Aplicável a problemas complexos** – Este método pode ser aplicado a uma ampla gama de problemas, mesmo aqueles que são complexos e intrincados, tornando-o versátil.

4) **Não há necessidade de informações sobre derivadas** – Não requer conhecimento de derivadas, tornando-o mais acessível e fácil de usar, principalmente nos casos em que o cálculo da derivada é muito laborioso.

5) **A convergência nem sempre é garantida** – O Método do Ponto Fixo pode nem sempre levar a uma solução, o que significa que a convergência nem sempre é garantida.

6) **Pode ser generalizado para a resolução de sistemas** de equações não lineares.

7) **Requer um bom palpite inicial** – Depende muito de um bom palpite inicial, um ponto de partida mal escolhido pode levar à divergência.

8) **Sensível à escolha da função** – A eficácia do método é sensível à função g escolhida, com algumas funções levando a melhores resultados do que outras. Recordamos que a mesma equação pode ser modelada por diferentes funções g.

3.5 Método de Newton

3.5.1 Descrição do método

O método de **Newton** ou de **Newton-Raphson**, como por vezes é chamado, pode ser considerado como um caso particular do método do ponto fixo, definido pela escolha da função

$$g(x) = x - \frac{f(x)}{f'(x)}. \qquad (3.5.1.1)$$

Graficamente, este método pode ser descrito aproximando o gráfico da função f pela sua tangente. A intersecção desta tangente com o eixo dos xx é tomada como o novo valor da aproximação à raiz de f.

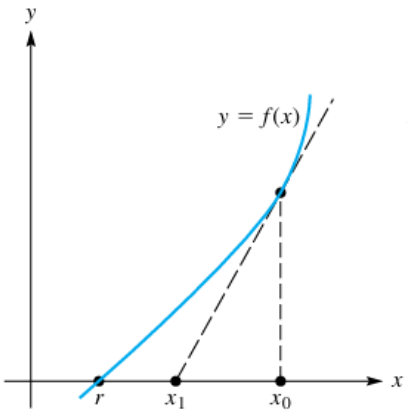


Figura 3.5.1.1: Método de Newton

A equação da tangente à curva $y = f(x)$ que passa pelo ponto $(x_k, f(x_k))$ é

$$y = f(x_k) + f'(x_k)(x - x_k) \qquad (3.5.1.2)$$

e, portanto, a sua intersecção com o eixo xx ocorre na posição

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \qquad (3.5.1.3)$$

Sendo assim, a expressão acima pode ser escrita nesta outra forma

$$x_{k+1} = x_k + h_k \qquad com \qquad h_k = -\frac{f(x_k)}{f'(x_k)} \qquad (3.5.1.4)$$

que é suscetível de seguinte leitura: o novo valor x_{k+1} obtém-se do anterior x_k adicionando a este uma correção h_k a calcular de acordo com a fórmula acima.

3.5.2 Erro e Convergência

A primeira observação que se deve fazer é que o método de Newton nem sempre é convergente. No teorema seguinte apresentamos as condições suficientes para que se garanta a convergência do método.

Teorema 3.5.2.1 Se uma vizinhança (a, b) da raiz s suficientemente pequena se verificar que $f \in C^2[a, b]$ e que para todo ξ nessa vizinhança

$$0 < m_1 \leq |f'(\xi)| \qquad e \qquad |f''(\xi)| \leq M_1 \qquad (3.5.2.1)$$

então o método de Newton converge, e o **erro** que satisfaz a relação

$$|e_{k+1}| \leq M|e_k|^2 \qquad com \qquad M = \frac{M_1}{2m_1} \qquad (3.5.2.2)$$

Demonstração: Desenvolvendo a função f em série de Taylor em torno de x_k, temos que

$$0 = f(s) = f(x_k) + f'(x_k)(s - x_k) + \frac{f''(\xi_k)}{2}(s - x_k)^2, \qquad \xi_k \in inter\,(s,\, x_k) \qquad (3.5.2.3)$$

donde se conclui

$$s = x_k - \frac{f(x_k)}{f'(x_k)} - \frac{f''(\xi_k)}{2f'(x_k)}(s - x_k)^2 \qquad (3.5.2.4)$$

Recordando a expressão do método de Newton, podemos escrever que

$$s - x_{k+1} = e_{k+1} = -\frac{f''(\xi_k)}{2f'(x_k)}(e_k)^2 \qquad (3.5.2.5)$$

Majorando o segundo membro e introduzindo as condições do teorema, chega-se sem dificuldade à conclusão pretendida. Daí, concluimos que a sucessão e_k tem como limite zero e, por conseguinte, o método de Newton converge. Além disso

$$c = \lim_{k \to \infty} \frac{|e_{k+1}|}{|e_k|^2} = \lim_{k \to \infty} \frac{|f''(\xi_k)|}{|2f'(x_k)|} = \frac{|f''(s)|}{|2f'(s)|} \qquad (3.5.2.6)$$

e pelas hipóteses do teorema este valor é finito. Logo, a ordem de convergência do método é $p = 2$, o que completa a demonstração.

Teorema 3.5.2.2 Se tivermos uma função f de classe C^2 tal que $f(s) = 0$ e $f'(s) \neq 0$ então existe uma vizinhança $[a,\, b]$ de s em que as condições do teorema anterior estão satisfeitas. Por outras palavras, se a aproximação inicial estiver suficientemente perto da solução s, o método de Newton converge.

3.5.3 Tempo de cálculo do método

O método de Newton, sendo um método de ponto fixo, também envolve o tempo de cálculo $T = nt_g$. No entanto, como g assume aqui uma forma particular em que é necessário o cálculo da derivada da função f, devemos considerar (desprezando o tempo de subtração e divisão)

$$T = n(t_f + t_{f'}) \qquad (3.5.3.1)$$

Note-se que para funções polinomiais o cálculo da derivada envolve até menor tempo que o tempo para calcular f, mas de modo geral passa-se a situação inversa. Por vezes, é utilizado o subterfúgio de considerar um ponto adicional, próximo da iterada, para aproximar f' por diferenças finitas, e nesse caso podemos considerar $T = 2nt_f$ (este artifício conduz a um método parecido com o método da secante, que iremos estudar mais à frente).

3.5.4 Método de otimização

Querendo encontrar um mínimo para uma função regular ϕ, isso corresponde a encontrar um z tal que $\phi'(z) = 0$, pelo que o método de Newton pode ser aplicado desde que $\phi \in C^2$. Convém apenas referir que neste caso, ao definir

$$x_{k+1} = x_k - \frac{\phi'(x_k)}{\phi''(x_k)}, \qquad (3.5.4.1)$$

o ponto x_{k+1} é o ponto de mínimo para a parábola definida por

$$y = \phi(x_k) + \phi'(x_k)(x - x_k) + \frac{\phi''(x_k)}{2}(x - x_k)^2, \qquad (3.5.4.2)$$

que corresponde a uma aproximação de ϕ pelo seu desenvolvimento em série de Taylor (de segunda ordem).

3.5.5 Método de Newton no caso de zeros múltiplos

Definição 3.5.5.1 Dizemos que uma função f tem um zero z de multiplicidade $p > 1$ se existir uma função h contínua em z tal que $h(z) \neq 0$ e que

$$f(x) = (x - z)^p h(x). \qquad (3.5.5.1)$$

Se $h \in C^p(V_z)$ então significa que

$$f(z) = f'(z) = \ldots = f^{(p-1)}(z) = 0, \qquad f^{(p)}(z) \neq 0 \qquad (3.5.5.2)$$

O facto de, para zeros múltiplos, se ter $f'(z) = 0$ coloca algumas questões relativamente à aplicabilidade do método de Newton. No entanto, podemos ver que mesmo nesse caso o método apresenta convergência local, embora já não seja de ordem quadrática.

Com efeito, calculando,

$$f'(x) = (x - z)^p h'(x) + p(x - z)^{p-1} h(x) \qquad (3.5.5.3)$$

e considerando

$$g(x) = x - \frac{Cf(x)}{f'(x)} = x - \frac{C(x-z)^p h(x)}{(x-z)^p h'(x) + p(x-z)^{p-1} h(x)} = x - \frac{C(x-z)\ h(x)}{(x-z)\ h'(x) + ph(x)} \qquad (3.5.5.4)$$

temos a função iteradora do método de Newton apenas quando $C = 1$.

Calculando a derivada, obtemos facilmente $g'(z) = 1 - \frac{C}{p}$. Ou seja, de acordo com o teorema que vimos, acerca da convergência do método do ponto fixo, quando $p > 1$, podemos ainda assegurar

convergência local para o método de Newton usual ($C = 1$), porque $0 < g'(z) = 1 - \frac{1}{p} < 1$, mas a ordem de convergência será linear e não quadrática.

Uma pequena modificação no método de Newton, usando $C = p$, ou seja,

$$x_{k+1} = x_k - \frac{p \times f(x_k)}{f'(x_k)} \qquad (3.5.5.5)$$

permite voltar a obter a convergência quadrática, pois neste caso $g'(z) = 0$. No entanto, há que salientar que isto pressupõe que saibamos à partida p, a multiplicidade do zero que pretendemos aproximar (esse conhecimento pode ser obtido através de resultados teóricos).

3.5.6 Algoritmo do método

O seguinte código é uma implementação em Python do algoritmo de Newton. As variáveis de entrada são:

- f - função
- df – primeira derivada da função objetivo
- $x0$ – valor inicial
- tol - tolerância (tipo de tolerância: convergência do ponto fixo)
- n - número máximo de iterações

A variável de saída é:

- p - aproximação da raiz de f, isto é, $f(p) \approx 0$.

Tabela 3.5.6.1: Versão simplificada do método de Newton

***PASSO 1**: importamos a biblioteca "math"* ***PASSO 2**: começamos com $i = 0$.* ***PASSO 3**: enquanto não chegarmos ao número máximo de iterações (n):* ***PASSO 4**: o programa apenas irá continuar se a derivada de f no ponto x0 (df(x0)) é diferente de 0.* ***PASSO 5**: calculamos x1.* ***PASSO 6**: se o valor da tolerância for superior a |x1- x0|(critério de paragem, caso isto não se verifique, saltamos para o **PASSO 8**):* ***PASSO 7**: devolve o último x1 obtido (raiz).* ***PASSO 8**: x0 passa a ser x1 e regressamos ao **PASSO 3**.*	*import math* *def **NewtonRaphson** (x0,tol,n,f,df):* *for i in range(n+1):* *assert df(x0) != 0* *x1 = x0-f(x0)/df(x0)* *if abs(x1-x0) < tol:* *return x1* *break* *x0 = x1*

3.5.7 Aplicação

Vamos agora aplicar o método de Newton para determinar uma aproximação, com um erro absoluto inferior a 5×10^{-6}, do (único) zero da função $f(x) = 1 + x + e^{x}$, que se sabe estar no intervalo $[-2, -1]$.

Vamos primeiro verificar se as condições suficientes de convergência se verificam.

- $f'(x) = 1 + e^{x} \rightarrow f'(x) > 0$
- $f''(x) = e^{x} \rightarrow f''(x) > 0$

Logo, o método converge desde que x_0 esteja à direita de zero, garantindo $f(x_0)f''(x_0) > 0$. Então, escolhendo $x_0 = -1$, garante-se a convergência do método.

Estão assim reunidas todas as condições de convergência. O cálculo das sucessivas iterações fornece os resultados do quadro seguinte:

Tabela 3.5.7.1: Aplicação do método de Newton

Iteração	x0	x1	|x1 - x0|	Tolerância	f(x0)
0	1	-0,268941	1,27	0,000001	4,718281
1	-0,268941	-1,116496	0,85	0,000001	1,495246
2	-1,116496	-1,275396	0,16	0,000001	0,210928
3	-1,275396	-1,278463	0,00	0,000001	0,003923
4	-1,278463	-1,278464	1,03	0,000001	1,97E-06
5	-1,278464	-1,278464	1,15	0,000001	6,94E-07

Podemos assim concluir que uma aproximação do zero da função, obtida pelo método de Newton é -1.278464. Na Figura 3.5.7.1 pode observar-se graficamente a evolução de x_1.

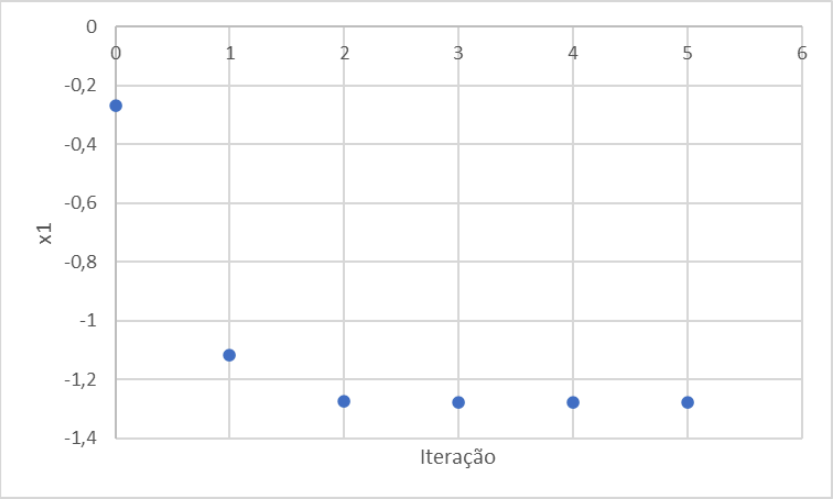


Figura 3.5.7.1: Evolução dos valores da estimativa no método de Newton

3.5.8 Considerações finais

1. Dada uma estimativa inicial x_0, esta fórmula gera uma sucessão x_k que presumivelmente **irá convergir para a raiz da função**.

2. A realização deste método **implica que se tenha de programar, além da função f, a sua derivada f'**.

3. Se f for uma função com expressão analítica muito complicada, o **cálculo da derivada e a respetiva programação serão tarefas trabalhosas.** O próprio código pode ficar mais lento devido ao cálculo desta expressão complicada de f'.

4. Este método **converge quadraticamente**, sendo o método que converge mais rapidamente apresentado até agora.

5. Este método pode ser generalizado para resolver sistemas de equações não-lineares.

3.6 Método da Secante

3.6.1 Descrição do método

Como já tinha sido referido anteriormente, este método é uma modificação do método de Newton onde se evita o cálculo de f'. Vamos recordar a fórmula para se calcular a iteração seguinte no método de Newton.

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \qquad (3.6.1.1)$$

No método da secante, nós substituímos $f'(x_k)$ por uma aproximação. Já que a derivada é definida por

$$f'(x) = \lim_{h \to \infty} \frac{f(x+h) - f(x)}{h} \qquad (3.6.1.2)$$

nós dizemos que para h pequeno mas fixo,

$$f'(x) \approx \frac{f(x+h)-f(x)}{h} \qquad (3.6.1.3)$$

Em particular, se $x = x_k$ e $h = x_{k-1} - x_k$, nós temos

$$f'(x_k) \approx \frac{f(x_{k-1})-f(x_k)}{x_{k-1}-x_k} \qquad (3.6.1.4)$$

que quando inserida na fórmula do método de Newton dá lugar à equação do **método da secante**.

Sendo muito semelhante ao método de Newton, o método da secante substitui o cálculo das derivadas pelo cálculo de uma razão incremental. Geometricamente, corresponde a substituir o papel da reta tangente, no método de Newton, por uma reta secante (de onde vem o nome).

Sendo assim, o **método da secante** consiste em, partindo de duas iterações quaisquer $(x_{k-1},\ f(x_{k-1}))$ e $(x_k,\ f(x_k))$, obter o valor seguinte x_{k+1} como intersecção da secante que passa pelos referidos pontos com o eixo dos xx.

A expressão que permite obter x_{k+1} pode ser escrita da seguinte forma

$$x_{k+1} = \frac{y_k x_{k-1} - y_{k-1} x_k}{y_k - y_{k-1}} \qquad (3.6.1.5)$$

É claro que agora, não havendo certeza de que y_k e y_{k-1} tenham sinais contrários, a expressão acima pode originar cancelamento subtrativo. Uma expressão equivalente à anterior mas menos sensível a erros de arredondamentos é

$$x_{k+1} = x_k - \frac{y_k}{f[x_{k-1}, x_k]} \qquad (3.6.1.6)$$

Esta expressão pode ser escrita noutra forma equivalente mas mais elucidativa

$$x_{k+1} = x_k + h_k \qquad com \qquad h_k = \frac{-y_k}{f[x_{k-1}, x_k]} \qquad (3.6.1.7)$$

que é suscetível de seguinte leitura: o novo valor x_{k+1} obtém-se do anterior x_k adicionando a este uma correção h_k a calcular de acordo com a fórmula acima.

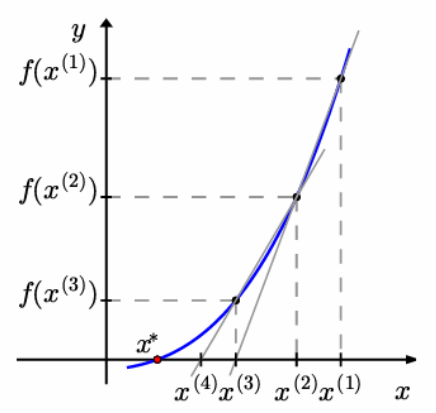


Figura 3.6.1.1: Método da Secante

É importante observar a semelhança entre o método da secante e o método da falsa posição. Por exemplo, as equações para calcular a iteração seguinte são idênticas numa comparação termo a termo. Ambos usam duas estimativas iniciais para calcular uma aproximação do declive da função que é utilizada para projetar para o eixo x e assim obter uma nova estimativa da raiz.

No entanto, há uma diferença significativa entre os dois métodos: no método da falsa posição, as duas estimativas sempre delimitam a raiz, enquanto no outro método, um dos valores iniciais é substituído por uma nova estimativa. Portanto, para todos os propósitos práticos, o método sempre converge porque a raiz é mantida dentro do intervalo.

Em contraste, o método da secante substitui os valores em sequência estrita, com o novo valor x_{k+1} substituindo x_k e x_k substituindo x_{k-1}. Para certos casos, isso pode levar à divergência. Todavia, quando converge, fá-lo com uma ordem de convergência superior a um, como veremos mais adiante.

3.6.2 Erro e Convergência

O teorema seguinte dá-nos condições suficientes que garantem a convergência do método da secante e revela a respetiva ordem de convergência.

Teorema 3.6.2.1 Se todas as iterações estiverem contidas numa vizinhança (a, b) suficientemente pequena da raiz s da função $f \in C^2[a, b]$, então o método da secante é convergente, e o erro satisfaz a relação

$$|e_{k+1}| \leq M|e_k||e_{k-1}| \qquad (3.6.2.1)$$

com

$$M = \frac{M_1}{2m_1}, \qquad 0 < m_1 \leq |f'(\xi)| \qquad e \qquad |f''(\xi)| \leq M_1 \qquad (3.6.2.2)$$

para todo o $\xi \in [a, b]$, e a ordem de convergência é $p = \frac{(1+\sqrt{5})}{2} \approx 1.618$.

Demonstração: Como sabemos, a interpolação linear da função f nos pontos x_{k-1} e x_k fornece a identidade

$$0 = f(s) = f(x_k) + f[x_k, x_{k-1}](s - x_k) + f[x_k, x_{k-1}, x](s - x_k)(s - x_{k-1}) \qquad (3.6.2.3)$$

donde se tira que o zero z é expresso por

$$z = x_k - \frac{f[x_k, x_{k-1}, s]}{f[x_k, x_{k-1}]}(s - x_k)(s - x_{k-1}) \qquad (3.6.2.4)$$

Mas, tendo em conta que $x_{k+1} = x_k - \frac{y_k}{f[x_{k-1}, x_k]}$, podemos simplificar a expressão anterior da seguinte forma

$$s = x_{k+1} - \frac{f[x_k, x_{k-1}, s]}{f[x_k, x_{k-1}]}(s - x_k)(s - x_{k-1}) \qquad (3.6.2.5)$$

do qual resulta que

$$e_{k+1} = -\frac{f[x_k, x_{k-1}, s]}{f[x_k, x_{k-1}]} e_k e_{k-1} \qquad (3.6.2.6)$$

Recordando a relação entre diferenças divididas e derivadas (teorema 2.1.7), podemos pôr que

$$f[x_k, x_{k-1}] = f'(\xi_k) \qquad e \qquad f[x_k, x_{k-1}, s] = \frac{f''(\eta_k)}{2} \qquad (3.6.2.7)$$

com $\xi_k \in inter\,(x_k, x_{k-1})$ e $\eta_k \in inter\,(x_k, x_{k-1}, x)$. Portanto,

$$e_{k+1} = -\frac{f''(\eta_k)}{2f'(\xi_k)} e_k e_{k-1} \qquad (3.6.2.8)$$

Tomando os valores absolutos de ambos os membros desta expressão e majorando de forma óbvia, vem que

$$|e_{k+1}| \leq \frac{M_1}{2m_1}|e_k||e_{k-1}| = M|e_k||e_{k-1}| \qquad (3.6.2.9)$$

Utilizando a notação $u_k = M|e_k|$, a relação anterior transforma-se nesta outra com aspecto ligeiramente mais simples

$$u_{k+1} \leq u_k u_{k-1} \qquad (3.6.2.10)$$

Admitimos que os pontos x_0 e x_1 pertencem a uma vizinhança suficientemente próxima do zero no sentido de que

$$u_0 \leq 1 \qquad e \qquad u_1 \leq \delta \qquad (3.6.2.11)$$

com $\delta < 1$. Então, deduzimos que

$$u_2 \leq \delta, \quad u_3 \leq \delta^2, \quad u_4 \leq \delta^3, \quad \ldots, \quad u_k \leq \delta^{\gamma_k} \qquad (3.6.2.12)$$

em que os exponentes γ_k satisfazem a seguinte relação de recorrência

$$\gamma_{k+1} = \gamma_k + \gamma_{k-1}, \qquad k = 1, 2, \ldots \qquad (3.6.2.13)$$

com $\gamma_0 = 0$ e $\gamma_1 = 1$. As relações anteriores permitem concluir que $\lim_{k \to \infty} u_k = 0$ e que, portanto, o método da secante converge. Os números γ_k gerados são os célebres *números de Fibonacci*, e não é difícil demonstrar a fórmula de Binet.

$$\gamma_k = \frac{[(\frac{1+\sqrt{5}}{2})^k - (\frac{1-\sqrt{5}}{2})^k]}{\sqrt{5}} \qquad (3.6.2.14)$$

Em particular, é verdadeira a igualdade

$$\lim_{k \to \infty} \frac{\gamma_{k+1}}{\gamma_k} = \frac{1+\sqrt{5}}{2} \qquad (3.6.2.15)$$

Então, para k suficientemente grande, podemos escrever as seguintes relações aproximadas

$$u_{k+1} \approx \delta^{\gamma_{k+1}} \approx \delta^{p\gamma_k} \approx (u_k)^p \qquad (3.6.2.16)$$

onde $p = \frac{1+\sqrt{5}}{2}$. É, portanto, plausível que

$$\lim_{k \to \infty} \frac{|u_{k+1}|}{|u_k|^p} \approx 1 \qquad (3.6.2.17)$$

Nestas condições também se deve ter que

$$\lim_{k \to \infty} \frac{|e_{k+1}|}{|e_k|^p} \approx c = M^{p-1} \qquad (3.6.2.18)$$

Este resultado sugere que a ordem do método da secante é $p = \frac{1+\sqrt{5}}{2} \approx 1.618$. A hipótese $|f'(\xi)| > 0$ implica que a raiz s é simples, pelo que este teorema não contempla o caso de zeros múltiplos.

3.6.3 Tempo de cálculo do método

Como o método da secante envolve apenas o cálculo de f uma vez o tempo de cálculo pode ser escrito da seguinte forma

$$T = nt_f \qquad (3.6.3.1)$$

Se compararmos o tempo de cálculo relativamente entre o método da secante e o método de Newton, para obter o mesmo majorante de erro ε, assumimos constantes semelhantes, que

$$T_N = \frac{log\ p\ (t_f + t_{f'})T_s}{t_f\ log\ 2} = 0.6942 \times (1 + \frac{t_{f'}}{t_f})T_s \qquad (3.6.3.2)$$

pelo que podemos concluir que o método de Newton será mais eficaz se o tempo de cálculo da derivada verificar

$$0.6942 \times (1 + \frac{t_{f'}}{t_f}) < 1 \Leftrightarrow \frac{t_{f'}}{t_f} < 0.4405, \qquad (3.6.3.3)$$

ou seja, será conveniente que o cálculo da derivada demore menos que metade do tempo do cálculo de f, o que normalmente não acontece!

3.6.4 Algoritmo do método

O seguinte código é uma implementação em Python do algoritmo da Secante. As variáveis de entrada são:

- f - função objetivo
- x_0 – aproximação inicial 1
- x_1 - aproximação inicial 2
- tol - tolerância (tipo de tolerância: convergência da função)
- n - número máximo de iterações

A variável de saída é:

- p - aproximação da raiz de f, isto é, $f(p) \approx 0$.

Tabela 3.6.4.1: Versão simplificada do método da Secante

PASSO 1**: importamos a biblioteca "math".* ***PASSO 2**: começamos com i = 0.* ***PASSO 3**: enquanto não chegarmos ao número máximo de iterações (n):* ***PASSO 4**: calculamos x2.* ***PASSO 5**: se o valor da tolerância for superior a |f(x2)| (se esta condição for quebrada, o programa salta para o **PASSO 7**):* ***PASSO 6**: devolve o último x2 obtido (raiz) arredondado à casa decimal definida no início (arre) e o programa termina.* ***PASSO 7**: calculamos o novo x0 e x1, que passam a ser, respectivamente, x1 e x2 e regressamos ao **PASSO 3	*import math* *def **Secante**(x0,x1,tol,n,f):* *for i in range(n+1):* *x2 = x1 - (f(x1)*(x1-x0))/(f(x1)-f(x0))* *if abs(f(x2)) < tol:* *return x2* *x0 = x1* *x1 = x2*

3.6.5 Aplicação

Vamos agora aplicar o método da Secante para procurar uma aproximação, com um erro absoluto inferior a 10^{-6}, do zero da função $f(x) = e^{3x+1} + e^{x}$. Salientamos que esta função não tem zeros, pelo que este é um bom exemplo de uma situação que não converge devido à má escolha da função.

Verificamos que algumas das condições suficientes de convergência estão satisfeitas.

- $f'(x) = 3e^{3x+1} + e^{x} \rightarrow f'(x) > 0$
- $f''(x) = 9e^{3x+1} + e^{x} \rightarrow f''(x) > 0$

Contudo, o método não converge: apesar de os valores de $f(x_n)$ convergirem para zero a sucessão x_n não converge.

O cálculo das sucessivas iterações fornece os resultados do quadro seguinte:

Tabela 3.6.5.1: Aplicação do método da Secante

Iteração	x0	x1	x2	f(x2)	Tolerância
0	-2	-1	-2,3934	0,0933	1,00E-06
1	-1	-2,3934	-2,7109	0,0672	1,00E-06
2	-2,3934	-2,7109	-3,5288	0,0294	1,00E-06
3	-2,7109	-3,5288	-4,1640	0,0155	1,00E-06
4	-3,5288	-4,164	-4,8773	0,0076	1,00E-06
5	-4,164	-4,8773	-5,5620	0,0038	1,00E-06

6	-4,8773	-5,562	-6,2583	0,0019	1,00E-06
7	-5,562	-6,2583	-6,9502	0,0009	1,00E-06
8	-6,2583	-6,9502	-7,6438	0,0004	1,00E-06
9	-6,9502	-7,6438	-8,3367	0,0002	1,00E-06
10	-7,6438	-8,3367	-9,0300	0,0001	1,00E-06
11	-8,3367	-9,03	-9,7231	5,99E-05	1,00E-06
12	-9,03	-9,7231	-10,4162	2,99E-05	1,00E-06
13	-9,7231	-10,4162	-11,1094	1,50E-05	1,00E-06
14	-10,4162	-11,1094	-11,8025	7,49E-06	1,00E-06
15	-11,1094	-11,8025	-12,4957	3,74E-06	1,00E-06
16	-11,8025	-12,4957	-13,1888	1,87E-06	1,00E-06
17	-12,4957	-13,1888	-13,8820	9,36E-07	1,00E-06

Na Figura 3.6.5.1 pode observar-se graficamente a evolução de x_2, ou seja, se continuássemos as iterações, x_2 iria continuar a descer.

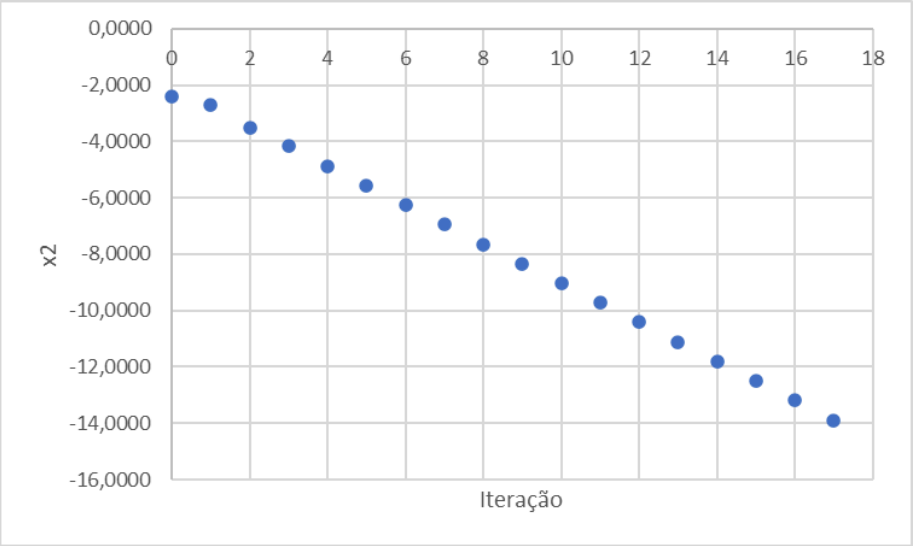


Figura 3.6.5.1: Evolução dos valores da estimativa no método da Secante

3.6.6 Considerações finais

1. Este método **precisa de dois "bons" pontos de partida.**

2. A velocidade convergência de convergência deste método é **quase tão rápida quanto o método de Newton.**

3. Não existe uma necessidade de se calcular a derivada neste método, o que por sua vez leva a uma **diminuição da dificuldade de processamento computacional** mas pode levar ao aumento do erro devido a usar-se uma aproximação da fórmula da derivada para o cálculo da próxima iteração.

4. Durante o processo de iteração deste método, **não se precisa de se satisfazer as condições do teorema de Bolzano-Cauchy** (sendo esta a principal diferença entre o método da secante e o da falsa posição).

Capítulo 4: Comparação de métodos iterativos

A eficiência de um método iterativo é determinada fundamentalmente pelo trabalho computacional envolvido no cálculo de cada iteração e pela rapidez de convergência para a solução.

O trabalho envolvido no cálculo de cada iteração pode ser estimado contando o número de operações aritméticas executadas. Note-se que isto se complica quando estão envolvidos cálculos de valores de funções transcendentes. Em geral, a maior percentagem do esforço computacional no cálculo de cada iteração é devida ao cálculo de valores de uma função e eventualmente das suas derivadas.

1) Na primeira iteração do método da bissecção são calculados dois valores de f. Depois, em cada iteração posterior é calculado um valor de f.

2) Na primeira iteração do método da falsa posição são calculados dois valores de f. Depois, em cada iteração posterior é calculado um valor de f.

3) Em cada iteração do método do ponto fixo é calculado um valor da função g.

4) Em cada iteração do método de Newton são calculados um valor da função f e um valor de f' (sem considerar divisão necessária entre ambas).

5) Na primeira iteração do método da secante são calculados dois valores de f. Depois, em cada iteração posterior é calculado um valor de f.

Vamos assim aplicar as equações listadas na Tabela 4.1 para cada um dos métodos iterativos (estas equações foram escolhidas por representarem uma gama de situações, como a necessidade de se trabalhar com funções logarítmicas, trigonométricas, algébricas e exponenciais).

Tabela 4.1: Comparação dos métodos iterativos

Número	f(x)
1	$x^3 - x - 1$
2	$x\,log(x) - 1$
3	$cos(x) - x$
4	$x^3 - 4x^2 - 10$
5	$x^2 - 5x + 2$
6	$(x + 1)^2 e^{(x^2 - 2)} - 1$

Foram implementados códigos referentes a cada método no Python e, assim, executados para todas as equações a fim de realizar a análise comparativa, restringindo um número de iterações de até 100, e uma tolerância de 10^{-6}. Os resultados são exibidos da Tabela 4.2 a 4.7. No final, uma análise referente

à média desses valores foi realizada para se comparar a efetividade geral dos métodos, e definir qual o melhor.

Tabela 4.2: Aplicação dos métodos estudados na função 1

Método	Dados iniciais	Tempo de execução	Nº de iterações	Raiz
Bissecção	[0, 2]	0,000685453415	20	1,3247175216
Falsa Posição	[0, 2]	0,000633478165	19	1,3247177315
Ponto Fixo	1	0,000462532043	23	1,3247163448
Newton	1	0,000153064728	4	1,3247179572
Secante	[0, 2]	0,000556468964	13	1,3247179475

Tabela 4.3: Aplicação dos métodos estudados na função 2

Método	Dados iniciais	Tempo de execução	Nº de iterações	Raiz
Bissecção	[1, 3]	0,000957727	20	1,7632226943
Falsa Posição	[1, 3]	0,000327826	7	1,7632224657
Ponto Fixo	1	0,000646353	24	1,7632222963
Newton	1	0,000185013	4	1,7632228343
Secante	[1, 3]	0,000144958	4	1,7632228338

Tabela 4.4: Aplicação dos métodos estudados na função 3

Método	Dados iniciais	Tempo de execução	Nº de iterações	Raiz
Bissecção	[-π, π]	0,0006175041	22	0,7390858752
Falsa Posição	[-π, π]	0,0002110004	6	0,7390850204
Ponto Fixo	-π	0,0004210472	34	0,7390845495
Newton	-π	0,0001506805	8	0,7390851332
Secante	[-π, π]	0,0001583099	7	0,7390851331

Tabela 4.5: Aplicação dos métodos estudados na função 4

Método	Dados iniciais	Tempo de execução	Nº de iterações	Raiz
Bissecção	[1, 2]	0,000379562	19	1,324717522
Falsa Posição	[1, 2]	0,000271797	16	1,324717763
Ponto Fixo	1	0,000370741	30	1,365229444
Newton	1	0,0000777245	4	1,324717957
Secante	[1, 2]	0,000108957	5	1,324717965

Tabela 4.6: Aplicação dos métodos estudados na função 5

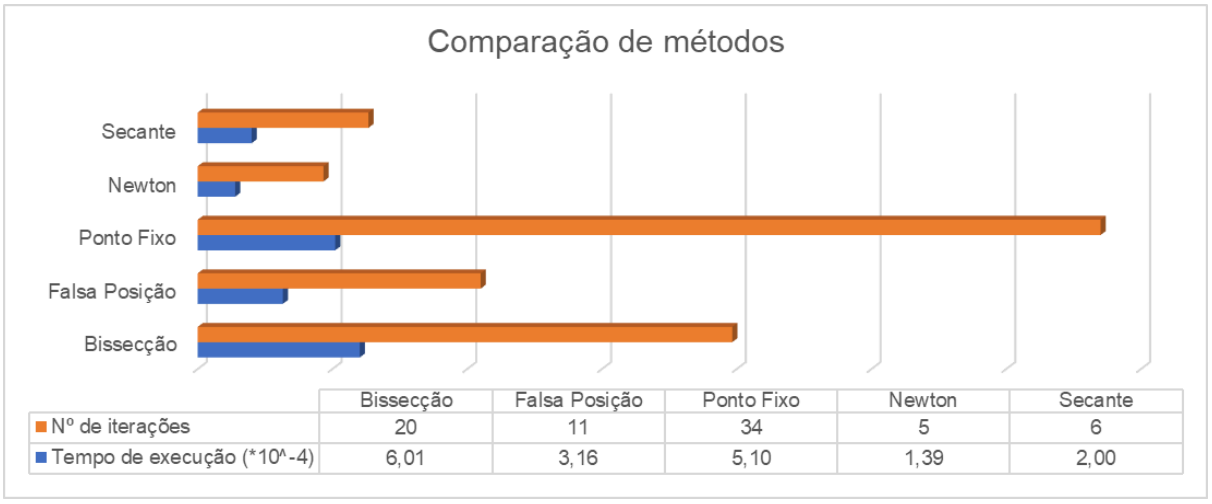
Método	Dados iniciais	Tempo de execução	Nº de iterações	Raiz
Bissecção	[0, 1]	0,000363111	19	0,438446999
Falsa Posição	[0, 1]	0,000127792	6	0,438447236
Ponto Fixo	1	0,00018096	8	0,438448209
Newton	1	0,00009036	4	0,4384471871
Secante	[0, 1]	0,000075817	4	0,438447187

Tabela 4.7: Aplicação dos métodos estudados na função 6

Método	Dados iniciais	Tempo de execução	Nº de iterações	Raiz
Bissecção	[0, 1]	0,000603437	19	0,866873741
Falsa Posição	[0, 1]	0,000322342	9	0,866873368
Ponto Fixo	1	0,000977755	82	-0,4767445*
Newton	1	0,00017548	4	0,8668735434
Secante	[0, 1]	0,000155210	5	0,866873687

*Reparemos que, para a função 6, o método do ponto fixo encontra outra solução fora do intervalo desejado.

Fazendo-se uma média entre todos estes valores, chega-se ao seguinte gráfico, visualizado na Figura 4.1:



	Bissecção	Falsa Posição	Ponto Fixo	Newton	Secante
Nº de iterações	20	11	34	5	6
Tempo de execução (*10^-4)	6,01	3,16	5,10	1,39	2,00

Figura 4.1: Comparação de métodos

Chegamos assim à conclusão que o método mais eficiente é o **método de Newton** já que possui o menor tempo de execução e é o método que consegue chegar ao resultado com o menor número de iterações.

Capítulo 5: Conclusão

Como vimos, resolver equações não lineares não é uma tarefa fácil.

As técnicas analíticas resolvem uma grande quantidade de equações, mas é preciso certa prática com as manipulações algébricas. Especialmente, é necessário ter-se atenção ao domínio das funções envolvidas para determinar corretamente as condições de existência de solução.

Todos os métodos numéricos que foram estudados encontraram as soluções de forma aproximada, porém com bastante precisão. Este trabalho tem como objetivo não só comparar múltiplos métodos iterativos como também escolher o melhor entre eles, tendo em conta as características de cada método como aplicabilidade, vantagens, desvantagens, tempo de execução, condições para convergência e ordem de convergência, permitem-nos escolher o método mais adequado para o problema.

O método da Bissecção foi escolhido por ser bastante intuitivo e de fácil aplicação apesar de apresentar uma desvantagem no facto de ter de tomar um intervalo inicial especial. O método da Falsa posição foi escolhido pela sua estabilidade e convergência, apesar de apresentar a mesma desvantagem que o método da Bissecção. O método do ponto fixo foi escolhido pela sua facilidade em calcular a iteração seguinte apesar de o resultado desse método depender bastante da função $g(x)$ que for utilizada. O método de Newton foi escolhido não só por ser o mais famoso como também por ter a maior ordem de convergência, tendo como desvantagem a necessidade de se calcular a derivada. Por fim, o método da Secante foi escolhido por ser uma modificação do método de Newton em que não é necessário o cálculo da derivada.

Apesar de cada um destes métodos poder ser considerado o mais adequado dependendo da situação, podemos concluir que o **método de Newton** não foi apenas o que mostrou o melhores resultados, com a vantagem de ter um menor esforço computacional, e com o maior número de vantagens, sendo a única desvantagem a necessidade de se calcular a derivada.

Na atualidade, o método de Newton é o método mais utilizado não só na resolução de equações não lineares relacionadas com problemas reais como também na realização de vários projetos científicos.

Um exemplo de uma aplicação real seria utilizar o método de Newton para calcular a taxa de crescimento de uma população tendo em conta a seguinte fórmula que representa o número de indivíduos da população em um tempo t.

$$P(t) = P_0 e^{kt} + \frac{m}{k}(e^{kt} - 1), \text{ sendo } k \text{ a taxa de crescimento} \qquad (5.1)$$

Suponha-se que certa população se desenvolve de acordo com os seguintes dados:

- População inicial: $P_0 = 2000000$
- Número de imigrantes no primeiro ano: $m = 500000$
- População ao final do primeiro ano: $P_1 = 2800000$

Para determinar a taxa de natalidade dessa população, devemos determinar k na equação

$$2800000 = 2000000e^{k} + \frac{500000}{k}(e^{k} - 1) \qquad (5.2)$$

Como não é possível isolar k, o método de Newton terá que ser utilizado para encontrar a solução aproximada de k. Para isso teremos que determinar o zero da função

$$f(k) = 20e^{k} + \frac{5}{k}(e^{k} - 1) - 28. \qquad (5.3)$$

Aplicando o método de Newton com estimativa inicial $k_0 = 0,5$ com um erro inferior a 10^{-6}, obtemos a solução $k = 0,1254369529$. Com essa taxa de crescimento podemos, por exemplo, prever a população ao final do segundo ano supondo que $m = 500000$ nesse ano. Assim, a equação $P(t) = P_0 e^{kt} + \frac{m}{k}(e^{kt} - 1)$ pode ser escrita

$$P(2) = 3,706915250 \times 10^{6}. \qquad (5.4)$$

Capítulo 6: Bibliografia

Alves, Carlos J. S., Fundamentos de Análise Numérica (I) - Teoria e Exercícios Versão de 2001/2002

Araújo, Adérito, Matemática Computacional, Universidade de Coimbra, 2017

Asano, Claudio, Colli, Eduardo, Cálculo numérico - Fundamentos e Aplicações, IME-USP, 2019

Barroso, Leônidas C. et. al., Cálculo Numérico (com Aplicações), 2a edição, Editora Harbra, São Paulo, 1987.

Chapra, Steven C. Métodos numéricos para engenharia, AMGH, 2011

Cheney, Ward, Kincaid, David, Numerical Mathematics and Computing, Sixth Edition, Thomson, 2008

Cunha, Francisco, Castro, Jânio, Cálculo Numérico, Fortaleza, CE, 2010

Diogo, Teresa, Conceitos básicos da Teoria dos Erros, Departamento de Matemática - Instituto Superior Técnico, 1995

Epperson, James F., An Introduction to Numerical Methods and Analysis, Mathematical Reviews - Second edition

Ferreira, J.C. Introdução à Análise Matemática, Fundação Calouste Gulbenkian, 1991

Kaw, Autar, Numerical Methods with Applications, University at South Florida, 2008

Kharab, Abdelwahab, Guenther, Ronald, An introduction to numerical methods : a MATLAB approach, CRC Press, 2018

L. Burden, J. Douglas Faires, Numerical Analysis, Ninth Edition Richard, Brooks/Cole, 2010

Lemos, Carlos e Pina. Heitor, Métodos numéricos - complementos e guia prático. IST Press

Matos, Aníbal, Apontamentos de Análise Numérica, Universidade do Porto, 2005

Namekar, Swapni, Shukla, Mritunjay, Role of Bisection Method, International Journal for Research in Applied Science & Engineering Technology (IJRASET), 2019

Pilling, Sérgio. II – Métodos numéricos para encontrar raízes (zeros) de funções reais - Cálculo Numérico, UNIVAP.

Pina, Heitor. Métodos Numéricos, Editora Mc Graw-Hill-Portugal. 1995

Renan, Elias, Renan, Silva, Jane, Almeida, Dalton, Valadares, Comparação de Métodos Iterativos de Resolução de Equações não Lineares Implementados no Octave, Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco (IFPE), 2016

Ruggiero, Márcia A. G., LOPES, Vera Lúcia R., Cálculo Numérico: Aspectos Teóricos e Computacionais, 2a edição, Makron Books, São Paulo, 1996.

Simões, Pedro, Computação Aplicada I - Equações não-lineares, Universidade de Coimbra, 2005

Sossolote, Junior, O Método de Newton-Raphson e Aplicações, Universidade Federal da Grande Dourados, 2021

Valença, Maria, Métodos Numéricos, Instituto Nacional de Investigação Científica, Braga, 1990

Capítulo 7: Anexos

7.1 Método da Bissecção em Python (versão completa)

```
import math

def format_float(x, num_digits):
    """    formats a floating-point number for printing in tables with
variable number of digits."""
    formatted_string = "{:.{}f}".format(x, num_digits)
    formatted_string = formatted_string.rstrip('0').rstrip('.') if '.' in
formatted_string else formatted_string
    return formatted_string

def Bissecção (a,b,tol,n,f):
    """   Approximate solution of f(x)=0 on interval [a,b] by bisection
method.

    Parameters
    f : function
        The function for which we are trying to approximate a solution
f(x)=0.
    a,b : numbers
        The interval in which to search for a solution.
    n : (positive) integer
        Maximum number of iterations
    tol : (positive) number
    Returns a table with all the iterations until the stopping criteria
is met or we reach the maximum number of iterations (n) ."""
    D_frame = []
    for i in range(n+1):
        assert f(a)*f(b) < 0
        m = (a+b)/2.0
        D_frame.append([i, a, b, m, abs(b-a)/2.0, tol, float(f(m))])
        if abs(b-a)/2.0 < tol:
            print ("Iteração  a  b  PM(m)  |b-a|/2  Tolerância  f(m)")
            for l in D_frame:
                print ( format_float ( l[0], 3 ) + "  " + format_float (
l[1], 9) + "  " + format_float ( l[2], 9) + "  " + format_float ( l[3],
9) + "  " + format_float ( l[4], 9 ) + "  " + format_float ( l[5], 9 ) +
"  " + format_float ( l[6], 9 ) )
            break
        if (f(a)*f(m) > 0):
                a = m
        else:
                b = m
```

7.2 Método do Ponto Fixo em Python (versão completa)

```
import math

def format_float(x, num_digits):
    """    formats a floating-point number for printing in tables with
variable number of digits."""
    formatted_string = "{:.{}f}".format(x, num_digits)
    formatted_string = formatted_string.rstrip('0').rstrip('.') if '.' in
formatted_string else formatted_string
    return formatted_string

def PontoFixo(x0,tol,n,g):
    """   Approximate solution of f(x)=0 on starting from x0 by fixed
point method.

    Parameters

    g : function
        The function for which we are trying to approximate a solution
g(x)=x.
    x0 : number
        The number for which we start our search for a solution.
    n : (positive) integer
        Maximum number of iterations
    tol : (positive) number

    Returns a table with all the iterations until the stopping criteria
is met or we reach the maximum number of iterations (n) ."""
    D_frame = []
    for i in range(n+1):
        D_frame.append([i, x0, g(x0), tol])
        if abs(g(x0)-x0) < tol:
            print ("Iteração      x0       g(x0)      Tolerância")
            for l in D_frame:
                    print ( format_float ( l[0], 3 ) + "   " +
format_float ( l[1], 9 ) + "   " + format_float ( l[2], 9 ) + "   " +
format_float ( l[3], 9 ) )
            break
        else:
            x0 = g(x0)
```

7.3 Método da Falsa Posição em Python (versão completa)

```
import math

def format_float(x, num_digits):
    """    formats a floating-point number for printing in tables with
variable number of digits."""
    formatted_string = "{:.{}f}".format(x, num_digits)
    formatted_string = formatted_string.rstrip('0').rstrip('.') if '.' in
formatted_string else formatted_string
    return formatted_string

def FalsaPosição(a,b,tol,n,f):
    """   Approximate solution of f(x)=0 on interval [a,b] by false
position method.

    Parameters

    f : function
        The function for which we are trying to approximate a solution
f(x)=0.
    a,b : numbers
        The interval in which to search for a solution.
    n : (positive) integer
        Maximum number of iterations
    tol : (positive) number

    Returns a table with all the iterations until the stopping criteria
is met or we reach the maximum number of iterations (n) ."""
    D_frame = []
    for i in range(n+1):
        assert f(a)*f(b) < 0
        x0 = (a*f(b) - b*f(a))/(f(b)-f(a))
        D_frame.append([i, a, b, f(a), f(b), x0, tol, float(f(x0))])
        if abs(f(x0)) < tol:
            print ("Iteração     a          b         f(a)     f(b)
x0       Tolerância     f(x0)")
            for l in D_frame:
                print ( format_float ( l[0], 3 ) + "  " + format_float (
l[1], 9 ) + "  " + format_float ( l[2], 9 ) + "  " + format_float ( l[3],
9 ) + "  " + format_float ( l[4], 9 ) + "  " + format_float ( l[5], 9 ) +
"  " + format_float ( l[6], 9 ) + "  " + format_float ( l[7], 9 ) )
            break
        if f(a)*f(x0) < 0:
            b = x0
        else:
            a = x0
```

7.4 Método de Newton-Raphson em Python (versão completa)

```
import math

def format_float(x, num_digits):
    """    formats a floating-point number for printing in tables with
variable number of digits."""
    formatted_string = "{:.{}f}".format(x, num_digits)
    formatted_string = formatted_string.rstrip('0').rstrip('.') if '.' in
formatted_string else formatted_string
    return formatted_string

def NewtonRaphson (x0,tol,n,f,df):
    """   Approximate solution of f(x)=0 starting from x0 by the
NewtonRaphson method.

    Parameters
    f : function
        The function for which we are trying to approximate a solution
f(x)=0.
    x0 : number
        The number which we start our search for a solution.
    n : (positive) integer
        Maximum number of iterations
    tol : (positive) number
    df : function
        The derivative of f

    Returns a table with all the iterations until the stopping criteria
is met or we reach the maximum number of iterations (n) ."""
    D_frame = []
    for i in range(n):
        assert df(x0) != 0
        x1 = x0-f(x0)/df(x0)
        D_frame.append([i, x0, x1, abs(x1-x0), tol, float(f(x0))])
        if abs(x1-x0) < tol:
            print ("Iteração     x0          x1     |x1 - x0|  Tolerância
f(x0)")
            for l in D_frame:
                print ( format_float ( l[0], 10 ) + "     " +
format_float ( l[1], 9 ) + "  " + format_float ( l[2], 9 ) + "  " +
format_float ( l[3], 9 ) + "  " + format_float ( l[4], 9 ) + "  " +
format_float ( l[5], 9 ) )
            break
        x0=x1
```

7.5 Método da Secante em Python (versão completa)

```
import math

def format_float(x, num_digits):
    """    formats a floating-point number for printing in tables with
variable number of digits."""
    formatted_string = "{:.{}f}".format(x, num_digits)
    formatted_string = formatted_string.rstrip('0').rstrip('.') if '.' in
formatted_string else formatted_string
    return formatted_string

def Secante(x0,x1,tol,n,f):
    """   Approximate solution of f(x)=0 starting from x0 by the Secant
method.

    Parameters

    f : function
        The function for which we are trying to approximate a solution
f(x)=0.
    x0, x1 : numbers
        The interval in which to search for a solution.
    n : (positive) integer
        Maximum number of iterations
    tol : (positive) number

    Returns a table with all the iterations until the stopping criteria
is met or we reach the maximum number of iterations (n) ."""
    D_frame = []
    for i in range(n+1):
        x2 = x1 - (f(x1)*(x1-x0))/(f(x1)-f(x0))
        D_frame.append([i, x0, x1, x2, float(f(x2)), tol])
        if abs(f(x2)) < tol:
            print (" Iteração      x0         x1        x2
f(x2)      Tolerância")
            for l in D_frame:
                print ( format_float ( l[0], 10 ) + "  " + format_float (
l[1], 9 ) + "  " + format_float ( l[2], 9 ) + "  " + format_float ( l[3],
9 ) + "  " + format_float ( l[4], 9 ) + "  " + format_float ( l[5], 9 ) )
            break
        x0 = x1
        x1 = x2
```