CAPÍTULO

1

Solução de Equações Não Lineares Univariadas

1.1 – Introdução

Nessa parte do curso, nosso objetivo é construir aproximações numéricas para a solução de *equações algébricas em uma única variável real*.

Diversos problemas reais são modelados por equações não lineares: modelos financeiros, relacionados a juros e/ou disponibilidade de recursos, modelos de engenharia de diversas sub áreas, modelos matemáticos de otimização univariada, só para citar algumas áreas.

Uma equação não linear é definida como

$$f(x) = 0, \tag{1.1}$$

sendo $f : \mathbb{R} \to \mathbb{R}$ uma função não linear, tipicamente algébrica com polinômios de grau 2 ou superior, ou transcendental.

Uma função é dita ser algébrica se ela pode ser expressa como

$$P_0(x)y^n + P_1(x)y^{n-1} + P_2(x)y^{n-2} + \cdots + P_n(x) = 0, \tag{1.2}$$

sendo $P_n(x)$ um polinômio de grau n em x, definido como

$$P_n(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \cdots + a_nx^n. \tag{1.3}$$

Funções transcendentais são funções que não são algébricas, como funções trigonométricas, exponenciais, logarítmicas ou a combinação delas.

O problema a ser estudado consiste, então, em encontrar todos os valores de x para os quais a equação (1.1) seja satisfeita. Esses valores são chamados *zeros* ou *raízes* dessa equação, e serão referenciados daqui pra frente como x^*. Essas raízes podem ser reais, caso típico de equações formadas por funções transcendentais, ou complexas, mais comumente encontradas para equações formadas por polinômios.

As soluções para o problema de encontrar as raízes de uma equação não linear podem

ajudar a resolver outros problemas relacionados. Por exemplo, o problema de encontrar o ponto de encontro entre duas funções, isto é, encontrar os valores para os quais $f(x) = g(x)$, sendo $f, g : \mathbb{R} \to \mathbb{R}$. Para solucionar esse problema, pode-se "transformá-lo" no problema de solução de equações não lineares simplesmente considerando que

$$f(x) = g(x) \quad \Rightarrow \quad h(x) = f(x) - g(x) = 0. \tag{1.4}$$

Dessa forma, as raízes de $h(x)$ são os pontos de encontro das funções $f(x)$ e $g(x)$.

Exemplo 1.1

Considere duas funções $f(x) = e^{-x} + 0.5$ e $g(x) = \cos(x)$. A Figura 1.1 mostra os gráficos dessas funções e mostra o ponto de intersecção delas, em que $f(x) = g(x)$.

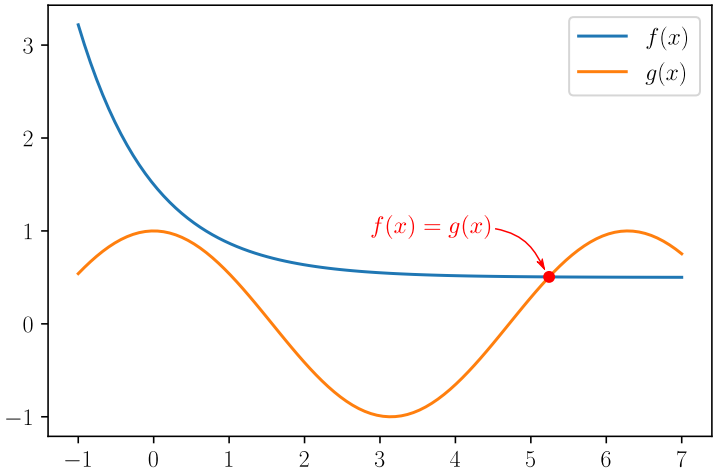


Figura 1.1: Gráficos das funções $f(x) = e^{-x} + 0.5$ (azul) e $g(x) = \cos(x)$ (laranja), destacando o ponto comum entre as duas.

Pode-se definir uma função $h(x) = f(x) - g(x)$, de tal forma que a raiz da equação linear formada por essa função, $h(x) = 0$, será também o ponto exato em que as duas funções se encontram. A Figura 1.2 ajuda a visualizar essa afirmação.

É importante frisar que a raiz x^* é o ponto no eixo das abscissas em que as duas funções se encontram e isso não significa que o valor das funções nesse ponto será igual. De fato, como o gráfico mostra

$$f(x^*) = g(x^*) \neq h(x^*), \tag{1.5}$$

apesar de

$$f(x^*) = g(x^*) \quad \text{e} \quad h(x^*) = 0 \tag{1.6}$$

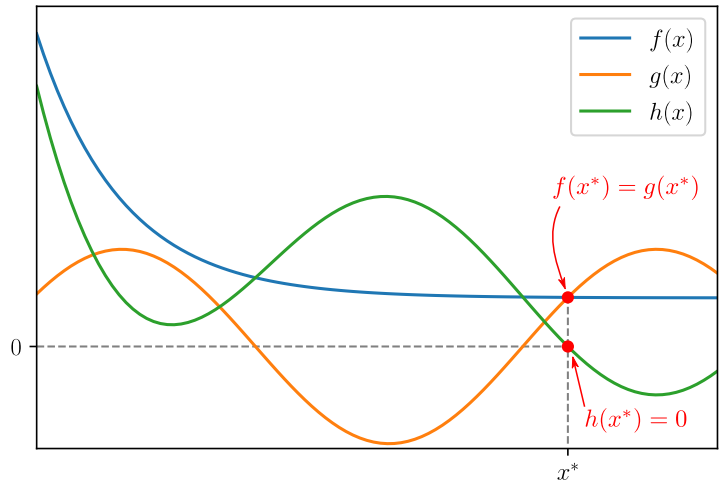


Figura 1.2: Gráficos das funções $f(x)$ (azul), $g(x)$ (laranja), mostradas anteriormente, e $h(x) = f(x) - g(x)$ (verde), destacando o ponto comum entre as duas primeiras e a raiz da equação não linear $h(x) = 0$.

Um segundo tipo de problema que pode utilizar as abordagens para solução de equações não lineares é o de encontrar o máximo ou mínimo local de uma função contínua, com base no Teorema de Fermat, tipicamente visto em disciplina de Cálculo I, que diz que *se uma função f tem um máximo ou mínimo local no ponto c, então* $f'(c) = 0$, sendo $f'(\cdot)$ a primeira derivada da função $f(\cdot)$. Assim, para encontrar o máximo ou mínimo de uma dada função, basta considerar a sua derivada (caso exista) como a função que constitue a equação não linear,

$$f'(x) = 0, \tag{1.7}$$

e os valores de x para os quais a equação (1.7) for satisfeita serão os máximos ou mínimos locais dessa função.

Exemplo 1.2

Seja a função $f(x) = \sin(x) - \cos(3x)$, e sua primeira derivada, $f'(x) = 3\sin(3x) + \cos(x)$, cujos gráficos são mostrados na Figura 1.3.

Como é possível ver nos gráficos, os valores das raízes da equação $f'(x) = 0$ são exatamente os pontos de máximo e mínimo locais para a função $f(x)$, em concordância com o Teorema de Fermat, mencionado logo acima.

Importante também frisar, como no exemplo anterior, que as raízes x_1^* e x_2^* são os pontos no eixo das abscissas em que ocorrem os máximo e mínimo locais da função no intervalo considerado, mas que isso não significa que os valores da função e de sua derivada nesse ponto serão iguais. De fato, como o gráfico mostra

$$f(x_1^*) \neq f'(x_1^*) \quad \text{e} \quad f(x_2^*) \neq f'(x_2^*). \tag{1.8}$$

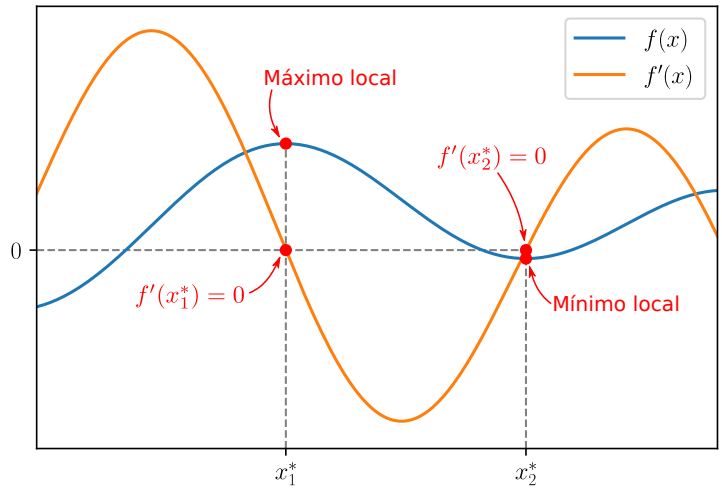


Figura 1.3: Gráficos das funções $f(x) = \sin(x) - \cos(3x)$ (azul) e de sua primeira derivada $f'(x)$ (laranja), destacando os pontos de máximo e mínimo locais de $f(x)$ e as raízes da equação $f'(x) = 0$.

Para resolver esses problemas, é possível utilizar três abordagens:

1. **soluções analíticas diretas –** para o caso em que pode-se encontrar uma expressão analítica para calcular as raízes da equação diretamente. Os leitores com certeza já tiveram contato com esse tipo de abordagem, quando aprenderam como achar as raízes de uma equação de segundo grau, definida como

$$f(x) = ax^2 + bx + c = 0, \tag{1.9}$$

usando a famosa *fórmula de Bhaskara*,

$$x^* = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \tag{1.10}$$

Usando a equação (1.10), pode-se encontrar diretamente todas as raízes da equação (1.9). No entanto, poucos são os casos em que a solução de uma equação não linear pode ser expressa por uma fórmula analítica direta. Sendo sincero, isso é razoavelmente raro... Por esses motivos, outras abordagens são necessárias para a obtenção desses valores.

2. **soluções gráficas –** para o caso em que se tem uma função bem definida na equação não linear, uma possível abordagem é gerar o gráfico dessa função e, por inspeção visual, identificar os pontos nos quais o gráfico toca o eixo das abscissas. Esses pontos serão as raízes ou zeros da equação. O grande problema desse método é sua imprecisão, uma vez que uma busca visual pode acarretar numa aproximação grosseira do valor, a depender de vários fatores como resolução do gráfico, precisão da representação, ou até mesmo a disposição de elementos como grades e valores nos eixos. O exemplo 1.3 ilustra esse tipo de abordagem e as suas dificuldades.

Exemplo 1.3

Considere o gráfico a seguir. É possível verificar que, visualmente, é impossível determinar com precisão a raiz da equação não linear formada pela função do gráfico, mesmo com a representação de valores nos eixos e com elementos como a grade do gráfico. No máximo, é possível estimar que essa raiz se encontra dentro do intervalo entre $[5, 6]$, talvez, $[5, 5.5]$, mas, precisamente, não é possível dar uma estimativa dessa raiz.

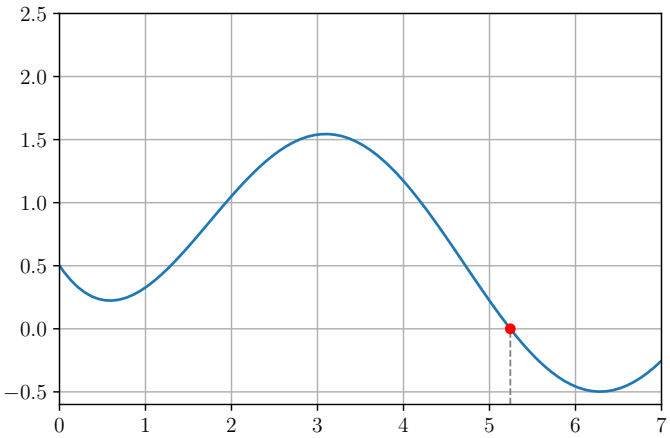


Figura 1.4: Exemplo das dificuldades do uso de inspeção visual para determinação de raízes de equações não lineares.

3. **métodos iterativos –** na impossibilidade de ter um resultado direto, via definição analítica de uma expressão para as raízes da equação, ou de um valor preciso obtido por inspeção visual do gráfico da função que descreve essa equação, a melhor opção para resolver o problema de encontrar a raiz de uma equação não linear é usar métodos iterativos. Nessa abordagem, algoritmos realizam iterações para se aproximar do valor da raiz, até encontrar o valor exato ou uma aproximação cujo erro seja aceitável.

Métodos Iterativos

Os métodos iterativos podem ser divididos em dois grupos: *intervalares* e *abertos*.

Nos métodos intervalares define-se um intervalo que contenha a raiz e, a cada iteração, esse intervalo vai diminuindo até que os limites dele fiquem suficientemente próximos da raiz procurada. Os métodos da *bisseção* e da *falsa posição* são dois dos mais famosos representantes desse grupo.

Nesse ponto, é interessante responder uma pergunta: *como garantir que a raiz da equação está de fato dentro do intervalo desejado?*. A resposta à essa pergunta é dada pelo Teorema do Valor Intermediário (visto com certeza na disciplina de Cálculo I!)

Teorema 1.1. ***Teorema do Valor Intermediário:*** *Seja a função contínua* $f : [a, b] \to \mathbb{R}$*, tal que* $f(a) < f(b)$*. Para qualquer* $k \in (f(a), f(b))$*, existe um* $x^* \in (a, b)$ *tal que* $f(x^*) = k$*.*

Ora, desse teorema, tomando $k = 0$, é possível afirmar que existe $x^* \in (a, b)$ tal que $f(x^*) = 0$, se $f(a) < f(b)$. Mas, como $k \in (f(a), f(b))$, e, por conseguinte, $0 \in (f(a), f(b))$ então, $f(a) < 0 < f(b)$ e, nesse caso, $f(a) \cdot f(b) < 0$.

Em outras palavras, se $f(x)$ é uma função contínua em um dado intervalo $[a, b]$ no qual ela troca de sinal, então ela tem pelo menos uma raíz neste intervalo. Somente haverá troca de sinal quando a relação $f(a) \cdot f(b) < 0$ for válida, isto é, quando a função "cruzar" o eixo das abscissas.

A Figura 1.5 ilustra essa discussão, mostrando a existência de uma raíz na função quando existe troca de sinal dentro do intervalo. Note que as regiões destacadas na curva ilustram, justamente, que a troca de sinal ocorre na passagem por 0.

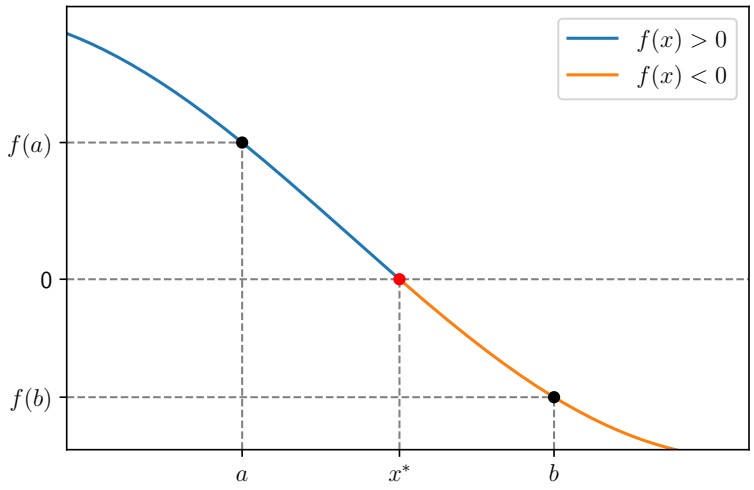


Figura 1.5: Ilustração do Teorema do Valor Intermediário, para o caso em que $k = 0$, mostrando a existência de uma raíz na função quando existe troca de sinal dentro do intervalo.

Exemplo 1.4

Mostre que existe pelo menos uma solução da equação $f(x) = 2 - e^x = 0$ no intervalo $[0, 1]$.

De acordo com o Teorema 1.1, como $f(0) = 1 > 0$ e $f(1) = -0.72 < 0$ e, portanto, $f(0) \cdot f(1) = 1 \cdot (-0.72) = -0.72 < 0$, então existe ao menos uma solução para a equação $f(x) = 2 - e^x = 0$ no intervalo $[0, 1]$.

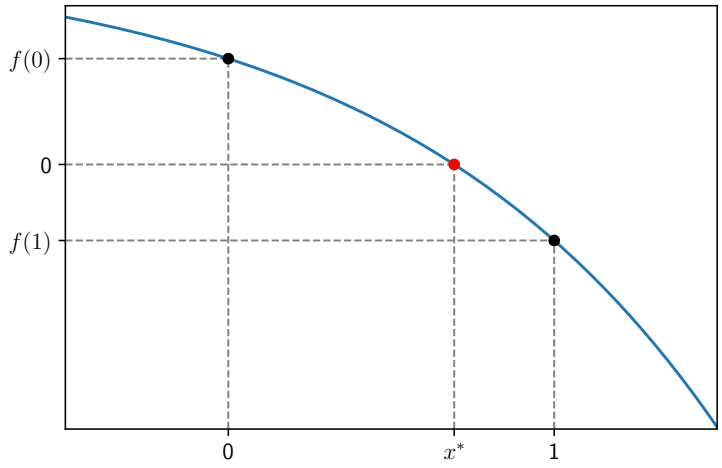


Figura 1.6: Gráfico da função $f(x) = 2 - e^x$, mostrando a existência de uma raíz no intervalo $[0, 1]$.

Na implementação de métodos intervalares para a solução de equações não lineares, é interessante contemplar um teste para verificar a existência de raiz no intervalo, baseado no teorema anterior, para evitar que a equação seja avaliada em um intervalo sem raiz, o que levaria o método utilizado à nunca convergir ou retornar um valor errado da raiz. No entanto, a depender dos valores trabalhados, a produto $f(a) \cdot f(b)$ pode ocasionar erros de *overflow* ou *underflow*, conforme discutido no capítulo anterior.

Uma forma alternativa para essa avaliação é utilizar a função *sinal*, $\text{sgn}(x)$, definida como

$$\text{sgn}(x) = \begin{cases} -1, & \text{se } x < 0, \\ 0, & \text{se } x = 0, \\ 1, & \text{se } x > 0, \end{cases} \tag{1.11}$$

para realizar o teste

$$\text{sgn}(f(a)) \cdot \text{sgn}(f(b)) = -1, \tag{1.12}$$

que, caso seja verdadeiro, indica a presença de raiz no intervalo testado, obtendo o mesmo resultado de $f(a) \cdot f(b) < 0$, mas sem risco de incorrer em erros de *overflow* ou *underflow*, uma vez que, independente do valor de x, a função só retorna os valores mostrados na equação (1.11).

Esse teste pode ser facilmente executado utilizando a função `sign` da biblioteca `Numpy`, cuja documentação é mostrada no link e que implementa a função descrita pela equação (1.11). O exemplo a seguir ilustra o uso dessa função.

Exemplo 1.5

Use a função `sign` da biblioteca `Numpy` para o caso trabalhado no exemplo 1.4.

Antes de tudo, é preciso carregar a biblioteca `Numpy`,

```
In [ ]: import numpy as np
```

e, sem seguida, criar a função a ser avaliada,

```
In [ ]: f = lambda x: 2 - np.exp(x)
```

e definir os limites do intervalo

```
In [ ]: a,b = 0,1
```

para, então, poder fazer o teste

```
In [ ]: np.sign(f(a))*np.sign(f(b))
Out[ ]: -1.0
```

indicando, como no exemplo 1.4, que no intervalo existe ao menos uma raiz para a equação avaliada.

Diferentemente dos métodos intervalares, nos métodos abertos não existe a necessidade de definir um intervalo que contenha a raiz para então realizar sua busca. Esses métodos partem de um ou dois valor inicial de x (que não definem um intervalo de busca) e, então, realizam a verificação se esse valor escolhido é, de fato, a raiz x^*. Caso não seja, o valor atual de x é atualizado por uma expressão, e o novo valor é novamente verificado. Esse processo iterativo se mantém até que o valor de x na iteração atual seja próximo o suficiente de x^*, de acordo com algum critério adotado. Os métodos da *Iteração de Ponto Fixo*, de *Newton-Raphson* e da *Secante* são alguns dos métodos abertos mais comumente utilizados.

Critérios de parada para métodos iterativos

Como visto, métodos iterativos iniciam a partir de um intervalo ou de um valor inicial, e seguem realizando iterações em busca do valor exato da raiz, ou, como acontece na maioria dos casos, da melhor aproximação. No entanto, é preciso definir: *qual a melhor aproximação?*

Para responder isso, é preciso estabelecer uma *tolerância*, ou seja, um limite de erro aceitável, abaixo do qual se considere a aproximação como satisfatória. Assim, três critérios

baseados na tolerância podem ser adotados como critério de paradas para métodos iterativos

$$|x_n - x_{n-1}| < \text{atol}, \tag{1.13a}$$

$$\frac{|x_n - x_{n-1}|}{|x_n|} < \text{rtol}, \tag{1.13b}$$

$$|f(x_n)| < \text{ftol}, \tag{1.13c}$$

sendo `atol` o erro absoluto máximo tolerado, ou *tolerância absoluta*, `rtol`, o erro relativo máximo tolerado, ou *tolerância relativa* e `ftol` o erro absoluto máximo tolerado do valor da função na aproximação. Esses três valores são especificados de acordo com a necessidade de cada problema.

Além desse, muitas implementações também consideram o número de iterações um critério extra para a parada do método, muitas vezes coexistindo com o anterior. Nesses casos em que os dois critérios são considerados, o algoritmo para sua execução sempre que um dos dois critérios for satisfeito.

1.2 – Método da Bisseção

O método da bisseção é um método iterativo intervalar, isto é, que busca a raiz dentro de um intervalo predefinido, e que toma como aproximação da raiz o ponto médio do intervalo considerado. Assim, para a função $f : [a, b] \to \mathbb{R}$, com $f(a)$ e $f(b)$ tendo sinais opostos (o que garante a existência da raiz no intervalo), pode-se definir que a aproximação x_{i+1} da raiz x^*, na i-ésima iteração, é dada por

$$x_{i+1} = \frac{a_i + b_i}{2}, \tag{1.14}$$

sendo a_i e b_i, respectivamente, os limites inferior e superior do intervalo analisado na i-ésima iteração.

Basicamente, o método calcula uma estimativa a partir do ponto médio de um subintervalo do $[a, b]$, dividido ao meio à cada iteração, de acordo com presença ou não da raiz nesses subintervalos. Ao fim de algumas iterações, o subintervalo considerado é pequeno o suficiente para que seu ponto médio seja exatamente a raiz procurada ou uma aproximação considerada satisfatória.

Para visualizar melhor o funcionamento do método, considere um intervalo $[a, b]$ genérico que contenha uma única raiz a ser encontrada. Na primeira iteração do algoritmo, $i = 0$, tem-se

$$a_0 = a \qquad b_0 = b \qquad x_1 = \frac{a_0 + b_0}{2} = \frac{a + b}{2}. \tag{1.15}$$

Essa iteração é ilustrada na Figura 1.7a. Se $f(x_1) \neq 0$, o que indica que x_1 não é

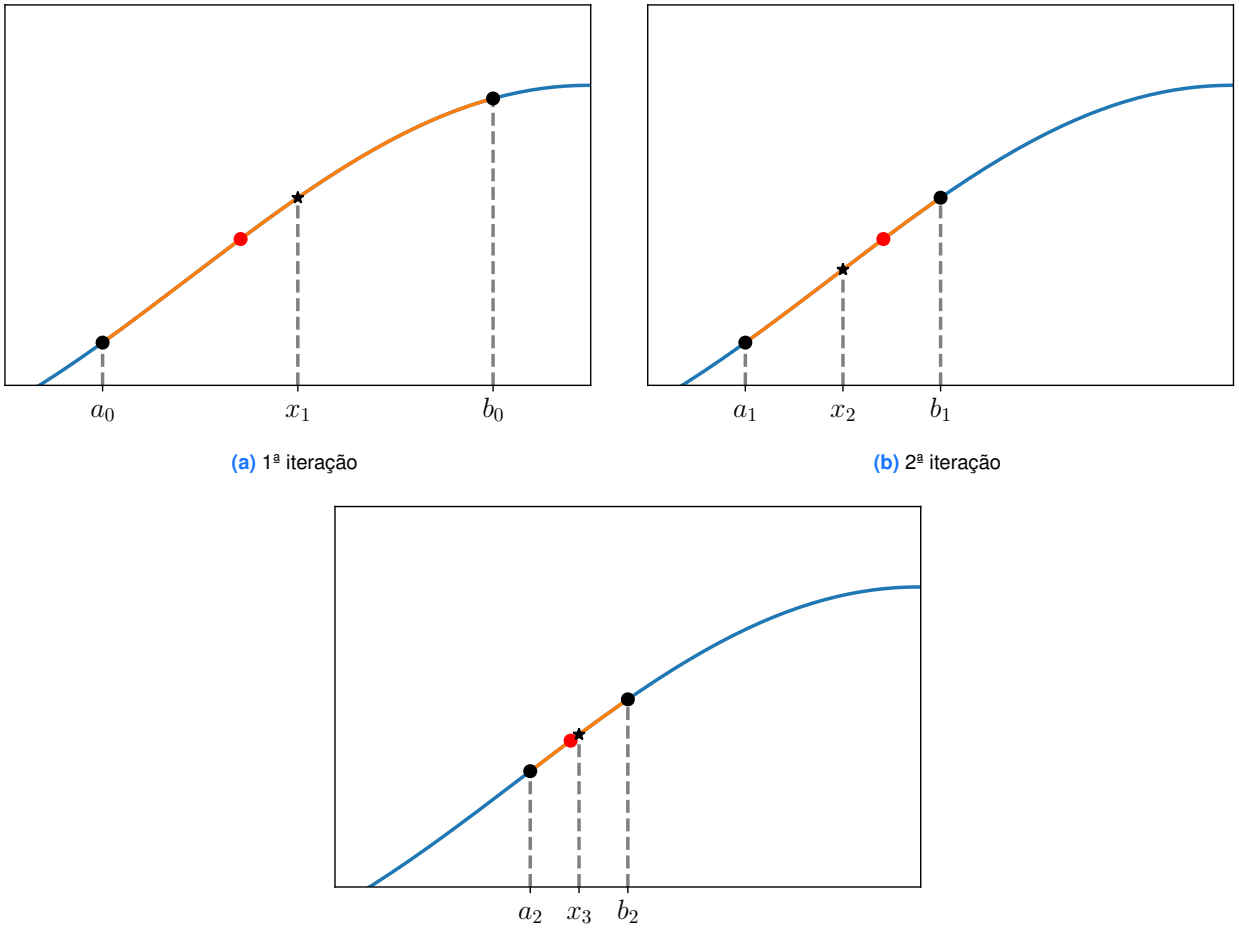
uma raiz, é necessário encontrar um subintervalo de $[a, b]$ para avaliar. O método gera, com a partição no meio, dois subintervalos: $[a, x_1]$ e $[x_1, b]$, e avalia a presença da raiz em cada um deles, usando o teste derivado do Teorema do Valor Intermediário (teorema 1.1):

- se $f(a) \cdot f(x_1) < 0$, o subintervalo $[a, x_1]$ contém a raiz;
- se $f(x_1) \cdot f(b) < 0$, o subintervalo $[x_1, b]$ contém a raiz.

Na consideração de se ter apenas uma raiz no intervalo $[a, b]$, apenas um dos testes acima será verdadeiro e o intervalo referente a isso será, então, o intervalo da próxima iteração. Consideremos, para fins de exemplo, que o intervalo $[a, x_1]$ contenha a raiz ($f(a) \cdot f(x_1) < 0$), como ilustrado na Figura 1.7b. Tem-se então, para a segunda iteração, em que $i = 1$,

$$a_1 = a_0 = a \qquad b_1 = x_1 \qquad x_2 = \frac{a_1 + b_1}{2} = \frac{a + x_1}{2}, \tag{1.16}$$

isto é, o novo subintervalo considerado será $[a, x_1]$, com ponto médio x_2 e um novo particionamento será feito, em que dois novos subintervalos são gerados: $[a_1, x_2] = [a, x_2]$ e $[x_2, b_1] = [x_2, x_1]$. Novamente, o algoritmo testa para saber em qual desses subintervalos a raiz está; pelo apresentado na Figura 1.7b, é possível ver que a raiz está no segundo subintervalo, $[x_2, b_1]$, ou seja, $f(x_2) \cdot f(b_1) < 0$.



(a) 1ª iteração

(b) 2ª iteração

(c) 3ª iteração

Figura 1.7: Visualização de três iterações do método da bisseção.

Dessa forma, na terceira iteração (ver Figura 1.7c),

$$a_2 = x_2 \qquad b_2 = b_1 \qquad x_3 = \frac{a_2 + b_2}{2} = \frac{x_2 + b_1}{2}, \tag{1.17}$$

e os subintervalos gerados são $[a_2, x_3] = [x_2, x_3]$ e $[x_3, b_1] = [x_3, b_1]$.

Percebe-se, pelas ilustrações mostradas na Figura 1.7, que na terceira iteração a aproximação (estrela) está bem próxima do valor real da raiz (círculo vermelho). À medida que o procedimento continua, a diferença entre aproximação e valor real será cada vez menor, até que o valor exato seja alcançado ou a aproximação torne-se boa o suficiente para não haver necessidade de novas iterações do algoritmo.

O método da bisseção pode ser descrito pelo algoritmo 1.1:

Algoritmo 1.1 ***Método da Bisseção***

Entrada: limites do intervalo a e b; função $f(x)$

Passo 1: inicializar $i = 0$, $a_i = a$, $b_i = b$

Passo 2: verificar a existência da raiz testando

se *sign*$(f(a)) \cdot$ *sign*$(f(b)) \neq -1$ **então**
não existe raiz no intervalo e o programa encerra

senão se *sign*$(f(a)) \cdot$ *sign*$(f(b)) = -1$ **então**

enquanto o critério de parada não for satisfeito **faça**

Passo 3: calcular o ponto médio do intervalo $x_{i+1} = \dfrac{a_i + b_i}{2}$

Passo 4: verificar se x_{i+1} é a raiz

se $f(x_{i+1}) = 0$ **então**
a raiz é o valor atual de x_{i+1} e o programa encerra

senão

Passo 5: verificar em que subintervalo está a raiz

se *sign*$(f(a_i)) \cdot$ *sign*$(f(x_{i+1})) = -1$ **então**
o primeiro subintervalo tem a raiz, atualizar o limite superior $b_i = x_{i+1}$

senão
o segundo subintervalo tem a raiz, atualizar o limite inferior $a_i = x_{i+1}$

Passo 6: atualizar $i = i + 1$ e voltar ao Passo 3

retorna Valor exato ou aproximado da raiz x_{i+1}

O método da bisseção é um algoritmo de busca binária, global, que tem como grande vantagem a garantia de solução da equação não linear, e como grande restrição a sua "lentidão" de convergência. Essas características são explicitadas pelo teorema 1.2 a seguir.

Teorema 1.2. *Seja* $[a_0, b_0] = [a, b]$ *o intervalo inicial a ser avaliado, com* $f(a) \cdot f(b) < 0$ *e* $x_n = (a_{n-1} + b_{n-1})/2$ *a raiz aproximada da equação na* n*-ésima iteração.*

Existe, então, uma raiz $x^* \in [a, b]$ *que satisfaz*

$$|x^* - x_n| = \left(\frac{1}{2}\right)^n (b-a), \textit{quando } n \geqslant 1. \tag{1.18}$$

Demonstração. Como, pelo método da bisseção, todo intervalo $[a_{n-1}, b_{n-1}]$ será dividido ao meio pelo valor x_n, pode-se, então, garantir que a raiz $x^* \in [a_{n-1}, x_n]$ ou $x^* \in [x_{n-1}, b_n]$, e que esses dois subintervalos tem a mesma amplitude, isto é, $x_n - a_{n-1} = b_{n-1} - x_n = \frac{1}{2}(b_{n-1} - a_{n-1})$. Dessa forma, têm-se que a distância entre raiz x^* e a aproximação x_n é, no máximo, o tamanho do subintervalo que contém a raiz,

$$|x^* - x_n| \leqslant \frac{1}{2}(b_{n-1} - a_{n-1}). \tag{1.19}$$

Além disso, é também premissa do método que a amplitude de um intervalo em uma dada iteração será sempre metade da amplitude do intervalo na iteração anterior, ou seja,

$$b_n - a_n = \frac{1}{2}(b_{n-1} - a_{n-1}). \tag{1.20}$$

Assim, intuitivamente, tem-se

$$\begin{aligned}
b_1 - a_1 &= \frac{1}{2}(b_0 - a_0) \\
b_2 - a_2 &= \frac{1}{2}(b_1 - a_1) = \left(\frac{1}{2}\right)^2 (b_0 - a_0) \\
b_3 - a_3 &= \frac{1}{2}(b_2 - a_2) = \left(\frac{1}{2}\right)^3 (b_0 - a_0) \\
&\vdots \\
b_n - a_n &= \left(\frac{1}{2}\right)^n (b_0 - a_0) = \left(\frac{1}{2}\right)^n (b-a).
\end{aligned} \tag{1.21}$$

Substituindo as equações (1.20) e (1.21) na equação (1.19), obtém-se, então

$$|x^* - x_n| \leqslant \left(\frac{1}{2}\right)^n (b-a), \tag{1.22}$$

e a prova está concluída. □

Do teorema 1.2, é possível, então, verificar que quando $n \to \infty$, $|x^* - x_n| \to 0$ e, portanto, $x_n \to x^*$, garantindo, dessa forma, que para um número de iterações suficientemente grande, o algoritmo convergirá para a raiz da equação analisada.

O algoritmo 1.1 de fato encontra a raiz da equação descrita pela função $f(x)$, porém apresenta algumas limitações que podem ocasionar um custo computacional relativamente alto. Dessa forma, algumas modificações podem ser inseridas para buscar otimizar seu funcionamento.

A primeira é com relação ao cálculo do ponto médio do intervalo analisado, no passo 3 do algoritmo. O cálculo usando a equação (1.14) pode acarretar um problema de *overflow* caso a e b sejam números relativamente grandes. Uma possível solução para isso é adaptar esse cálculo, de forma que

$$x_{i+1} = a_i + 0.5 \cdot (b_i - a_i), \tag{1.23}$$

o que garante encontrar o mesmo valor, porém de forma menos susceptível ao problema.

A segunda mofificação é relativa a minimização do cálculo de funções. Olhando para o algoritmo 1.1, é possível perceber que o cálculo da função em algum ponto ocorre duas vezes no teste do passo 2 e três vezes nos testes do passo 4, que se repete nas i iterações, de forma que, nessa implementação, são realizados $3i + 2$ vezes o cálculo da função em algum ponto. Se a função avaliada for relativamente complexa de calcular, e/ou a busca pela raiz exigir um número relativamente alto de iterações, essa implementação pode tornar-se computacionalmente limitada, exigindo um tempo de execução elevado para o cálculo da raiz.

Para solucionar isso, algumas alterações podem ser realizadas no algoritmo 1.1:

- no passo 1, inicializar uma variável com o valor da função no ponto a: $fa = f(a_i)$;
- no passo 3, calcular também o valor da função no ponto médio: $fx = f(x_{i+1})$;
- no passo 4, substituir o cálculo da função no primeiro teste pela resultado do cálculo da função no passo 3: $fx = 0$;
- no passo 5, substituir os cálculos da função no teste pelos valores já calculados nas alterações anteriores: $\text{sign}(fa) \cdot \text{sign}(fx) = -1$, e atualizar o valor de fa para o caso em que o segundo subintervalo tem a raiz, fazendo $fa = fx$.

Feitas essas modificações, é possível verificar que só existem cálculos de funções em duas ocasiões: na inicialização de variáveis, passo 1, que ocorre uma única vez, e no passo 3, quando se calcula o valor do ponto médio. Ou seja, só existirão $i + 1$ cálculo de funções, bem menos que os $3i + 2$ cálculos anteriores.

Por fim, é preciso estabelecer um critério de parada para o algoritmo, ainda não estabelecido no algoritmo 1.1. Como discutido na sessão 1.1, dois critérios podem ser usados para estabelecer a parada do algoritmo: um critério baseado no erro máximo admitido, aqui chamado de tolerância e um critério baseado no número máximo de iterações.

Para o primeiro caso, basta considerar uma das expressões apresentadas na equação (1.13a), colocando a tolerância (tol) como um dos parâmetros de entrada do algoritmo. Mais frequentemente os erros absoluto ou relativo são considerados como critério de parada,

sendo, portanto,

$$|x_{i+1} - x_i| \leqslant tol, \tag{1.24a}$$

$$\left|\frac{x_{i+1} - x_i}{x_{i+1}}\right| \leqslant tol, \tag{1.24b}$$

sendo x_i e x_{i+1} o valor da estimativa da raiz, respectivamente, na iteração anterior e na atual. É preciso também notar que o valor de tol muda para cada caso: quando o erro é absoluto, tol é um valor absoluto, e quando o erro é relativo, esse valor varia entre $0 \leqslant tol \leqslant 1$ ou de forma percentual.

Especificamente para o erro absoluto, da definição do método e como discutido no teorema 1.2, a distância entre raiz e aproximação será sempre metade do tamanho do intervalo $[a_n, b_n]$. Dessa forma,

$$|x_{i+1} - x_i| = \frac{b_i - a_i}{2} \leqslant tol \tag{1.25}$$

pode ser adotado como o critério para verificação de erro absoluto.

O segundo critério a ser considerado, o número máximo de iterações, é importante para evitar que o algoritmo caia em um laço infinito ou que processe uma quantidade excessiva de iterações até alcançar o limite tolerável de erro. Esse valor máximo de iterações pode ser escolhido, porém, o método da bisseção possibilita calcular o número de iterações necessárias para se chegar a um determinado valor de erro desejado, antes mesmo de executar o método, como mostra o teorema a seguir.

Teorema 1.3. *Para alcançar a acurácia de*

$$|x^* - x_n| \leqslant tol \tag{1.26}$$

é suficiente fazer n iterações, sendo n dado por

$$n \geqslant \log_2\left(\frac{b-a}{tol}\right). \tag{1.27}$$

Demonstração. Do teorema 1.2,

$$\begin{aligned} |x^* - x_n| &\leqslant tol \\ \left(\frac{1}{2}\right)^n (b-a) &\leqslant tol. \end{aligned} \tag{1.28}$$

Aplicando logaritmo aos dois lados da equação (1.2), tem-se

$$\begin{aligned}
\log\left[\left(\frac{1}{2}\right)^{n}(b-a)\right] &\leqslant \log(tol) \\
\log\left(\frac{1}{2}\right)^{n} + \log(b-a) &\leqslant \log(tol) \\
n\log(2) &\geqslant \log(b-a) - \log(tol) \\
n &\geqslant \frac{\log\left(\frac{b-a}{tol}\right)}{\log(2)} \qquad (1.29) \\
n &\geqslant \log_2\left(\frac{b-a}{tol}\right), \qquad (1.30)
\end{aligned}$$

e a prova está concluída.

□

Dessa forma, o critério do número máximo de iterações pode ser calculado como

$$n = \frac{\log\left(\frac{b-a}{tol}\right)}{\log(2)}. \qquad (1.31)$$