Zeros de funções de uma variável

Prof. Luiz T. F. Eleno

Departamento de Engenharia de Materiais
Escola de Engenharia de Lorena
Universidade de São Paulo

2019

Plano de aula

Plano de aula

Zeros de funções

- Os **zeros** ou **raízes** de uma função $f(x)$ são os valores de x para os quais

$$f(x) = 0$$

- Conhecemos maneiras de encontrar **analiticamente** os zeros de apenas algumas funções
- Por exemplo, sabemos resolver equações polinomiais de grau um ou dois:

$$f(x) = a_0 + a_1 x = 0 \qquad \Rightarrow \qquad x = -\frac{a_0}{a_1}$$

$$f(x) = a_0 + a_1 x + a_2 x^2 = 0 \qquad \Rightarrow \qquad x = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_0 a_2}}{2a_2}$$

  - É possível resolver assim também equações polinomiais de graus 3 e 4, mas não mais do que isso
- Sabemos também outros casos, por exemplo

$$f(x) = \mathrm{sen}(\pi x) = 0 \qquad \Rightarrow \qquad x = 0, \pm 1, \pm 2, \ldots$$

- E existem casos bem simples sem solução analítica, por exemplo:

$$f(x) = x - \cos x = 0 \qquad \Rightarrow \qquad x = ?$$

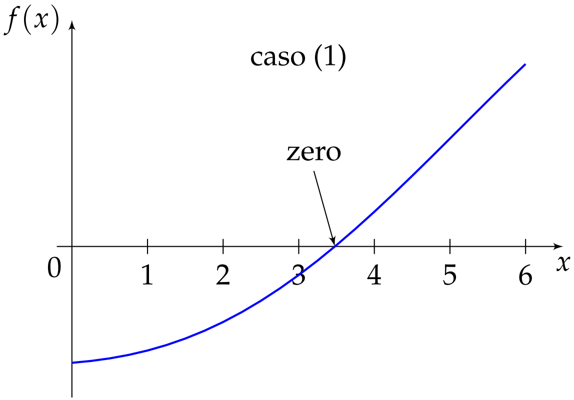
- Nessas situações, precisamos usar **métodos numéricos**

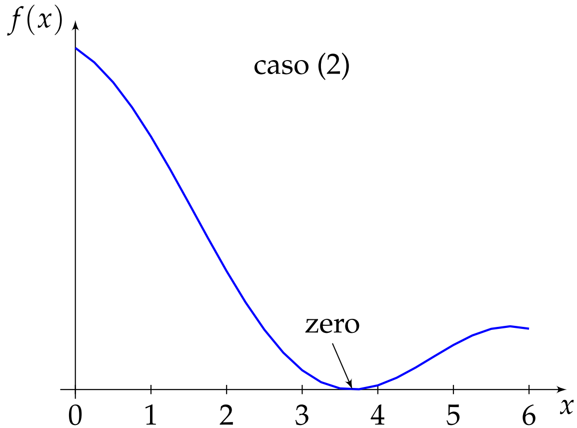
Plano de aula

1. Contextualização
2. Delimitação da raiz
3. Método da bissecção ou dicotomia
4. Método da falsa posição
5. Método de Newton
6. `scipy.optimize`

Delimitação da raiz

- Boa parte dos métodos numéricos requer a **delimitação** do zero da função
  - ou seja, dada uma raiz x, encontrar um intervalo $[a, b]$ tal que $a < x < b$
- Uma maneira é **delimitar graficamente** o zero da função.
  - ou seja, faça o gráfico da função e procure-a visualmente!
  - Encontre então um intervalo $[a, b]$ contendo a raiz





- Nos dois casos acima, o intervalo $[a, b] = [3, 4]$, por exemplo, delimita uma raiz.
- No entanto:
  - no caso (1): $f(a) \cdot f(b) < 0$; é uma **raiz simples** ou de **multiplicidade ímpar**
  - no caso (2): $f(a) \cdot f(b) > 0$; é uma **raiz de multiplicidade par**

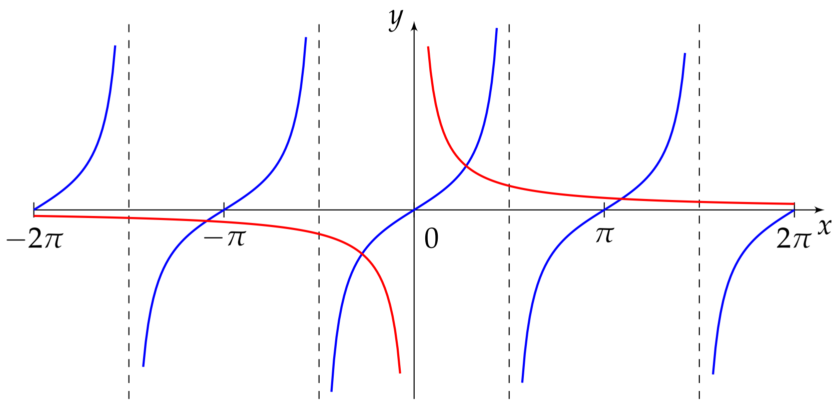
Delimitação da raiz

- Às vezes, é mais fácil encontrar os zeros de uma função $f(x)$ separando-a em duas partes e encontrando as intersecções entre os gráficos dessas partes
- Por exemplo, para achar os zeros de

$$f(x) = x \operatorname{tg} x - 1\,,$$

podemos encontrar as intersecções entre os gráficos de

$$g(x) = \operatorname{tg} x \qquad \text{e} \qquad h(x) = \frac{1}{x}$$

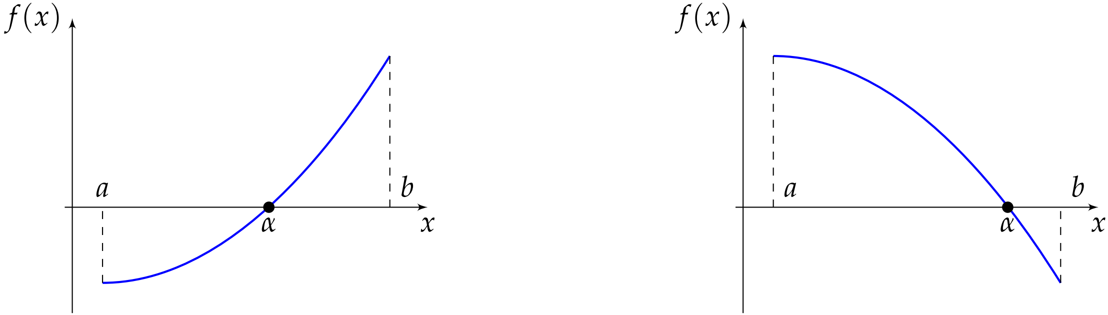


- **Obs.:** Você vai precisar resolver uma equação parecida com essa em Mecânica Quântica!!!

Plano de aula

Raiz simples
ou de multiplicidade ímpar

- Uma propriedade de uma raiz simples (ou de multiplicidade ímpar) α, tal que $f(\alpha) = 0$, delimitada no intervalo $[a, b]$, é observada nas figuras abaixo:



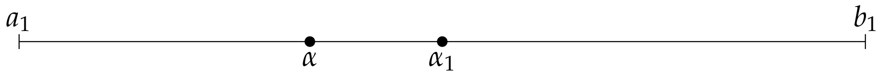
- **Propriedade:** desde que $f(x)$ seja contínua no intervalo $[a, b]$, vale sempre a relação

$$f(a) \cdot f(b) < 0$$

- Para isso, é preciso que α seja o único zero de $f(x)$ no intervalo $[a, b]$
- Essa propriedade não vale se a raiz tem multiplicidade par!
- Podemos usar essa propriedade como base de um método numérico →
- **Prelúdio:** teoria dos jogos (fez parte da Lista 1)

Método da bissecção ou dicotomia

- O método numérico mais simples é o chamado **Método da bissecção** ou **dicotomia**
- Funciona no caso de raízes simples ou de multiplicidade ímpar
- A raiz α tal que $f(\alpha) = 0$ deve ser inicialmente delimitada no intervalo $[a_1, b_1]$
- α deve ser a única raiz no intervalo $[a_1, b_1]$
  - deve acontecer então que $f(a_1) \cdot f(b_1) < 0$



- A primeira tentativa para a raiz é o ponto médio do intervalo:

$$\alpha_1 = \frac{a_1 + b_1}{2}$$

- o erro está delimitado por

$$|\alpha - \alpha_1| \leq \frac{b_1 - a_1}{2}$$

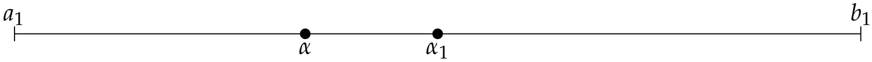
- Veja que, no caso da figura acima, a raiz está agora delimitada no intervalo $[a_1, \alpha_1]$
  - temos portanto que

$$f(a_1) \cdot f(\alpha_1) < 0 \qquad \text{e} \qquad f(\alpha_1) \cdot f(b_1) > 0$$

- Vamos agora fazer mais tentativas — ou seja, iterar! $\rightarrow$

Método da bissecção ou dicotomia

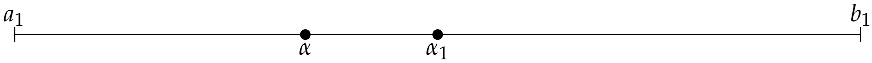
passo 1



$$\alpha_1 = \frac{a_1 + b_1}{2}, \qquad |\alpha - \alpha_1| \leq \frac{b_1 - a_1}{2}$$

Método da bissecção ou dicotomia

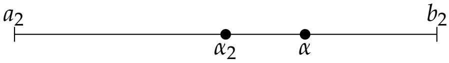
passo 1



$$\alpha_1 = \frac{a_1 + b_1}{2}, \qquad |\alpha - \alpha_1| \leq \frac{b_1 - a_1}{2}$$

$$f(a_1) \cdot f(\alpha_1) < 0 \qquad \Rightarrow \qquad a_2 = a_1, \quad b_2 = \alpha_1$$

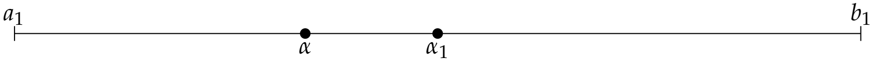
passo 2



$$\alpha_2 = \frac{a_2 + b_2}{2}, \qquad |\alpha - \alpha_2| \leq \frac{b_2 - a_2}{2} = \frac{b_1 - a_1}{2^2}$$

Método da bissecção ou dicotomia

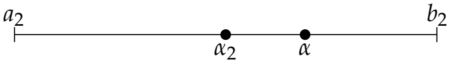
passo 1



$$\alpha_1 = \frac{a_1 + b_1}{2}, \qquad |\alpha - \alpha_1| \leq \frac{b_1 - a_1}{2}$$

$$f(a_1) \cdot f(\alpha_1) < 0 \qquad \Rightarrow \qquad a_2 = a_1, \quad b_2 = \alpha_1$$

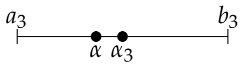
passo 2



$$\alpha_2 = \frac{a_2 + b_2}{2}, \qquad |\alpha - \alpha_2| \leq \frac{b_2 - a_2}{2} = \frac{b_1 - a_1}{2^2}$$

$$f(a_2) \cdot f(\alpha_2) > 0 \qquad \Rightarrow \qquad a_3 = \alpha_2, \quad b_3 = b_2$$

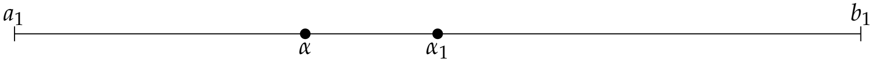
passo 3



$$\alpha_3 = \frac{a_3 + b_3}{2}, \qquad |\alpha - \alpha_3| \leq \frac{b_3 - a_3}{2} = \frac{b_0 - a_0}{2^3}$$

Método da bissecção ou dicotomia

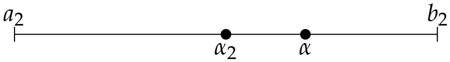
passo 1



$$\alpha_1 = \frac{a_1 + b_1}{2}, \qquad |\alpha - \alpha_1| \leq \frac{b_1 - a_1}{2}$$

$$f(a_1) \cdot f(\alpha_1) < 0 \qquad \Rightarrow \qquad a_2 = a_1, \quad b_2 = \alpha_1$$

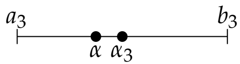
passo 2



$$\alpha_2 = \frac{a_2 + b_2}{2}, \qquad |\alpha - \alpha_2| \leq \frac{b_2 - a_2}{2} = \frac{b_1 - a_1}{2^2}$$

$$f(a_2) \cdot f(\alpha_2) > 0 \qquad \Rightarrow \qquad a_3 = \alpha_2, \quad b_3 = b_2$$

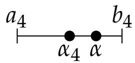
passo 3



$$\alpha_3 = \frac{a_3 + b_3}{2}, \qquad |\alpha - \alpha_3| \leq \frac{b_3 - a_3}{2} = \frac{b_0 - a_0}{2^3}$$

$$f(a_3) \cdot f(\alpha_3) < 0 \qquad \Rightarrow \qquad a_4 = a_3, \quad b_4 = \alpha_3$$

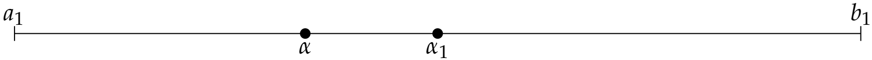
passo 4



$$\alpha_4 = \frac{a_4 + b_4}{2}, \qquad |\alpha - \alpha_4| \leq \frac{b_4 - a_4}{2} = \frac{b_0 - a_0}{2^4}$$

Método da bissecção ou dicotomia

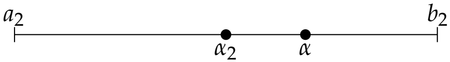
passo 1



$$\alpha_1 = \frac{a_1 + b_1}{2}, \qquad |\alpha - \alpha_1| \leq \frac{b_1 - a_1}{2}$$

$$f(a_1) \cdot f(\alpha_1) < 0 \qquad \Rightarrow \qquad a_2 = a_1, \quad b_2 = \alpha_1$$

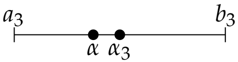
passo 2



$$\alpha_2 = \frac{a_2 + b_2}{2}, \qquad |\alpha - \alpha_2| \leq \frac{b_2 - a_2}{2} = \frac{b_1 - a_1}{2^2}$$

$$f(a_2) \cdot f(\alpha_2) > 0 \qquad \Rightarrow \qquad a_3 = \alpha_2, \quad b_3 = b_2$$

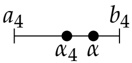
passo 3



$$\alpha_3 = \frac{a_3 + b_3}{2}, \qquad |\alpha - \alpha_3| \leq \frac{b_3 - a_3}{2} = \frac{b_0 - a_0}{2^3}$$

$$f(a_3) \cdot f(\alpha_3) < 0 \qquad \Rightarrow \qquad a_4 = a_3, \quad b_4 = \alpha_3$$

passo 4



$$\alpha_4 = \frac{a_4 + b_4}{2}, \qquad |\alpha - \alpha_4| \leq \frac{b_4 - a_4}{2} = \frac{b_0 - a_0}{2^4}$$

$$f(a_4) \cdot f(\alpha_4) > 0 \qquad \Rightarrow \qquad a_5 = \alpha_4, \quad b_5 = b_4$$

$$\vdots$$

Algoritmo

- Inicialmente, defina o intervalo $[a, b]$ contendo a raiz α tal que $f(\alpha) = 0$
  - certifique-se de que $f(a) \cdot f(b) < 0$ e que só há uma raiz nesse intervalo
- estabeleça uma precisão ϵ (por exemplo, $\epsilon = 10^{-4}$, ou seja, três casas decimais)

Algoritmo para o método da bissecção

1. Calcule
$$\alpha_1 = \frac{a+b}{2} \qquad \text{e} \qquad \delta = \frac{b-a}{2}$$
2. Se $\delta \leq \epsilon$, pare. Se não, continue:
3. Se $f(a) \cdot f(\alpha_1) < 0$, faça $b = \alpha_1$; senão, faça $a = \alpha_1$. Volte para o passo 1.

- Ao final da execução do algoritmo, α_1 será a melhor aproximação para a raiz dentro da precisão pré-estabelecida $|\alpha - \alpha_1| \leq \epsilon$
- Por garantia, você pode implementar uma verificação da condição $f(a) \cdot f(b) < 0$ antes de começar a rodar o algoritmo

Número de iterações

- Das equações dos slides anteriores, sabemos que, em cada iteração do método da bissecção, a raiz está delimitada num intervalo $[a_n, b_n]$ tal que

$$|\alpha - \alpha_n| \leq \frac{b_n - a_n}{2} = \frac{b_0 - a_0}{2^n}$$

- Se precisamos de uma precisão ϵ, é só impor

$$\frac{b_0 - a_0}{2^n} \leq \epsilon$$

ou, isolando n:

$$\boxed{n \geq \log_2\left(\frac{b_0 - a_0}{\epsilon}\right)}$$

que também pode ser escrita como

$$n \geq \log_2 10\, \log_{10}\left(\frac{b_0 - a_0}{\epsilon}\right) \approx 3.322 \log_{10}\left(\frac{b_0 - a_0}{\epsilon}\right)$$

- Por exemplo, se $b_0 - a_0 = 1$ e $\epsilon = 10^{-4} \Rightarrow n \geq 13.287$
  - ou seja, $n = 14$ garante a precisão desejada.

Vantagens e desvantagens

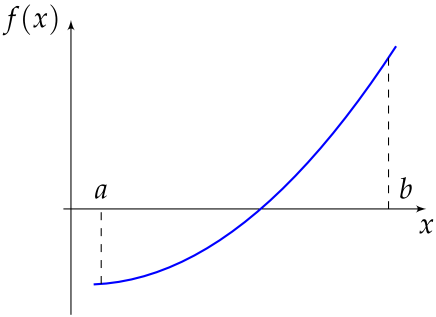
Vantagens e desvantagens do método da bissecção:

- **Vantagens:**
  - sempre conhecemos a precisão e sabemos exatamente o número de iterações para atingir a convergência desejada
  - uma vez conhecido um intervalo delimitando a raiz, é um método de convergência garantida
- **Desvantagem:**
  - é um método lento, exigindo muitas iterações (em comparação com os métodos a seguir)

Plano de aula

Método da falsa posição

- No método da bissecção, a nova aproximação para a raiz, $\alpha_n = (a_n + b_n)/2$, não leva em conta os valores da função nos extremos, ou seja, $f(a_n)$ e $f(b_n)$
- O **Método da falsa posição** ou ***regula falsi*** (latim, "usualmente falso") é uma maneira de tentar fazer isso e de acelerar a convergência para o zero



- A única diferença em relação ao método da bissecção é a maneira como a aproximação para a raiz é calculada em cada iteração:

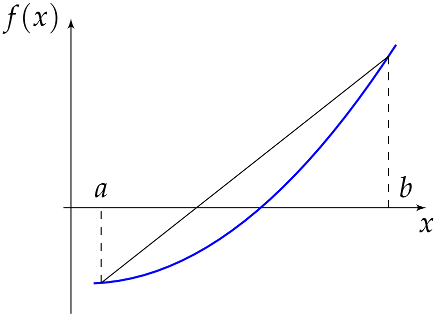
$$\alpha_n = \frac{a_n f(b_n) - b_n f(a_n)}{f(b_n) - f(a_n)}$$

que é o zero da reta que une $(a_n, f(a_n))$ e $(b_n, f(b_n))$

- Como a raiz continua sempre delimitada, a convergência é garantida
- Geralmente (mas nem sempre!) é mais rápido que a bissecção
- No entanto, aumenta o número de operações por iteração
- Além disso, não há como saber previamente o número de iterações para a precisão desejada

Método da falsa posição

- No método da bissecção, a nova aproximação para a raiz, $\alpha_n = (a_n + b_n)/2$, não leva em conta os valores da função nos extremos, ou seja, $f(a_n)$ e $f(b_n)$
- O **Método da falsa posição** ou ***regula falsi*** (latim, "usualmente falso") é uma maneira de tentar fazer isso e de acelerar a convergência para o zero



- A única diferença em relação ao método da bissecção é a maneira como a aproximação para a raiz é calculada em cada iteração:

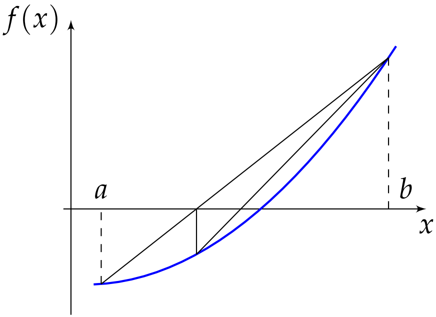
$$\alpha_n = \frac{a_n f(b_n) - b_n f(a_n)}{f(b_n) - f(a_n)}$$

que é o zero da reta que une $(a_n, f(a_n))$ e $(b_n, f(b_n))$

- Como a raiz continua sempre delimitada, a convergência é garantida
- Geralmente (mas nem sempre!) é mais rápido que a bissecção
- No entanto, aumenta o número de operações por iteração
- Além disso, não há como saber previamente o número de iterações para a precisão desejada

Método da falsa posição

- No método da bissecção, a nova aproximação para a raiz, $\alpha_n = (a_n + b_n)/2$, não leva em conta os valores da função nos extremos, ou seja, $f(a_n)$ e $f(b_n)$
- O **Método da falsa posição** ou ***regula falsi*** (latim, "usualmente falso") é uma maneira de tentar fazer isso e de acelerar a convergência para o zero



- A única diferença em relação ao método da bissecção é a maneira como a aproximação para a raiz é calculada em cada iteração:

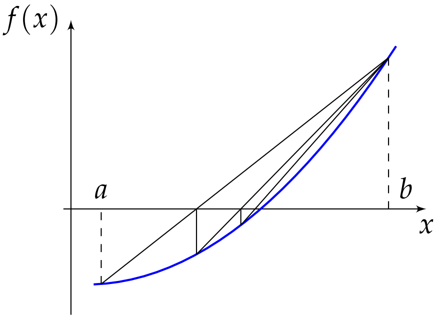
$$\alpha_n = \frac{a_n f(b_n) - b_n f(a_n)}{f(b_n) - f(a_n)}$$

que é o zero da reta que une $(a_n, f(a_n))$ e $(b_n, f(b_n))$

- Como a raiz continua sempre delimitada, a convergência é garantida
- Geralmente (mas nem sempre!) é mais rápido que a bissecção
- No entanto, aumenta o número de operações por iteração
- Além disso, não há como saber previamente o número de iterações para a precisão desejada

Método da falsa posição

- No método da bissecção, a nova aproximação para a raiz, $\alpha_n = (a_n + b_n)/2$, não leva em conta os valores da função nos extremos, ou seja, $f(a_n)$ e $f(b_n)$
- O **Método da falsa posição** ou ***regula falsi*** (latim, "usualmente falso") é uma maneira de tentar fazer isso e de acelerar a convergência para o zero



- A única diferença em relação ao método da bissecção é a maneira como a aproximação para a raiz é calculada em cada iteração:

$$\alpha_n = \frac{a_n f(b_n) - b_n f(a_n)}{f(b_n) - f(a_n)}$$

que é o zero da reta que une $(a_n, f(a_n))$ e $(b_n, f(b_n))$

- Como a raiz continua sempre delimitada, a convergência é garantida
- Geralmente (mas nem sempre!) é mais rápido que a bissecção
- No entanto, aumenta o número de operações por iteração
- Além disso, não há como saber previamente o número de iterações para a precisão desejada

Algoritmo

- Inicialmente, defina o intervalo $[a, b]$ contendo a raiz α tal que $f(\alpha) = 0$
  - certifique-se de que $f(a) \cdot f(b) < 0$ e que só há uma raiz nesse intervalo
- estabeleça uma **tolerância** η
  - que é diferente da precisão ϵ

Algoritmo para o método da falsa posição

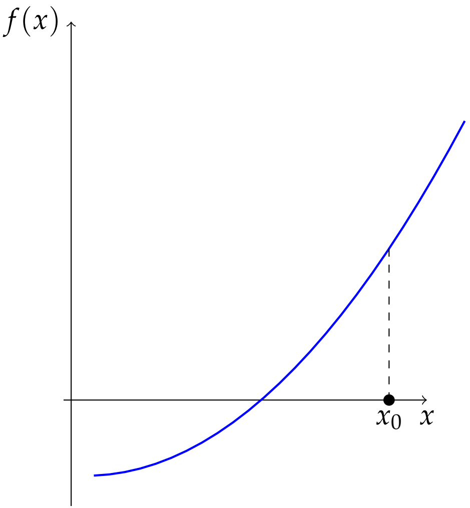
1. Calcule
$$\alpha_0 = \frac{af(b) - bf(a)}{f(b) - f(a)}$$
2. Se $|f(\alpha_0)| \leq \eta$, pare. Se não, continue:
3. Se $f(a) \cdot f(\alpha_0) < 0$, faça $b = \alpha_0$; senão, faça $a = \alpha_0$. Volte para o passo 1.

- Ao final da execução do algoritmo, α_0 será a melhor aproximação para a raiz dentro da tolerância pré-estabelecida $|f(\alpha_0)| \leq \eta$
- Como não sabemos quantas iterações serão necessárias para atingir a convergência, é melhor adicionar um número máximo de iterações ao **critério de parada**

Plano de aula

Método de Newton

- **Método de Newton,** ou de Newton-Raphson:



- A nova aproximação x_n é dada pelo zero da reta tangente a $f(x)$ em $x = x_{n-1}$
- a equação da reta tangente é

$$y = f(x_{n-1}) + f'(x_{n-1})\,(x - x_{n-1})$$

cujo zero x_n é encontrado fazendo $y = 0$:

$$f(x_{n-1}) + f'(x_{n-1})\,(x_n - x_{n-1}) = 0$$

$\Rightarrow$

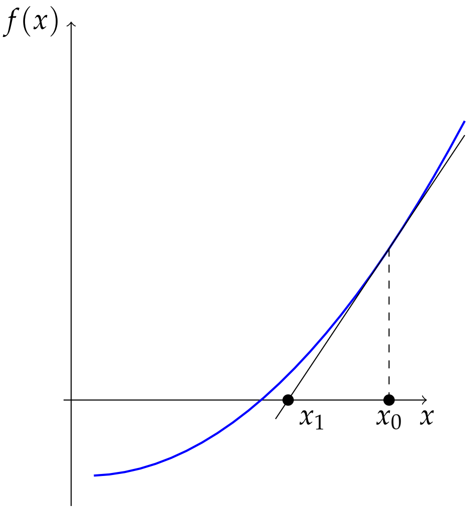
$$\boxed{x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}}$$

- Como ilustra a figura, o método de Newton é extremamente rápido
- Não é necessário delimitar a raiz, basta um "chute" inicial

- Mas a convergência não é garantida!
  - dependendo da função e do chute inicial, o método de Newton pode se afastar da raiz (e possivelmente tender a $\pm\infty$), ou oscilar em torno de algum valor
- Além disso, requer o cálculo da primeira derivada da função, que pode ser custoso ou inacessível computacionalmente

Método de Newton

- **Método de Newton,** ou de Newton-Raphson:



- A nova aproximação x_n é dada pelo zero da reta tangente a $f(x)$ em $x = x_{n-1}$
- a equação da reta tangente é

$$y = f(x_{n-1}) + f'(x_{n-1})\,(x - x_{n-1})$$

cujo zero x_n é encontrado fazendo $y = 0$:

$$f(x_{n-1}) + f'(x_{n-1})\,(x_n - x_{n-1}) = 0$$

$\Rightarrow$

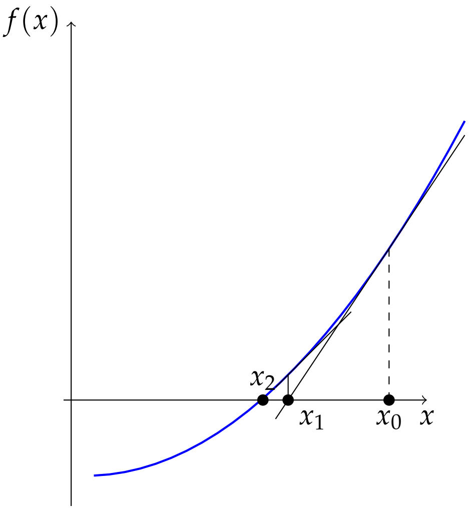
$$\boxed{x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}}$$

- Como ilustra a figura, o método de Newton é extremamente rápido
- Não é necessário delimitar a raiz, basta um "chute" inicial

- Mas a convergência não é garantida!
  - dependendo da função e do chute inicial, o método de Newton pode se afastar da raiz (e possivelmente tender a $\pm\infty$), ou oscilar em torno de algum valor
- Além disso, requer o cálculo da primeira derivada da função, que pode ser custoso ou inacessível computacionalmente

Método de Newton

- **Método de Newton,** ou de Newton-Raphson:



- A nova aproximação x_n é dada pelo zero da reta tangente a $f(x)$ em $x = x_{n-1}$
- a equação da reta tangente é

$$y = f(x_{n-1}) + f'(x_{n-1})\,(x - x_{n-1})$$

cujo zero x_n é encontrado fazendo $y = 0$:

$$f(x_{n-1}) + f'(x_{n-1})\,(x_n - x_{n-1}) = 0$$

$\Rightarrow$

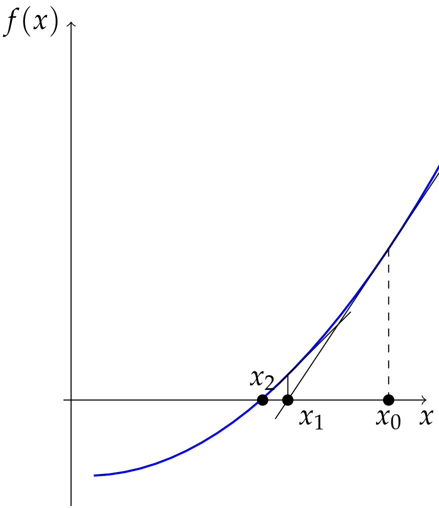
$$\boxed{x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}}$$

- Como ilustra a figura, o método de Newton é extremamente rápido
- Não é necessário delimitar a raiz, basta um “chute” inicial

- Mas a convergência não é garantida!
  - dependendo da função e do chute inicial, o método de Newton pode se afastar da raiz (e possivelmente tender a $\pm\infty$), ou oscilar em torno de algum valor
- Além disso, requer o cálculo da primeira derivada da função, que pode ser custoso ou inacessível computacionalmente

Método de Newton

- **Método de Newton,** ou de Newton-Raphson:



- A nova aproximação x_n é dada pelo zero da reta tangente a $f(x)$ em $x = x_{n-1}$
- a equação da reta tangente é

$$y = f(x_{n-1}) + f'(x_{n-1})\,(x - x_{n-1})$$

cujo zero x_n é encontrado fazendo $y = 0$:

$$f(x_{n-1}) + f'(x_{n-1})\,(x_n - x_{n-1}) = 0$$

$$\Rightarrow \qquad \boxed{x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}}$$

- Como ilustra a figura, o método de Newton é extremamente rápido
- Não é necessário delimitar a raiz, basta um "chute" inicial
  - CHUTE: **C**álculo **H**ipotético **U**niversal **T**eórico **E**xplicativo
- Mas a convergência não é garantida!
  - dependendo da função e do chute inicial, o método de Newton pode se afastar da raiz (e possivelmente tender a $\pm\infty$), ou oscilar em torno de algum valor
- Além disso, requer o cálculo da primeira derivada da função, que pode ser custoso ou inacessível computacionalmente

Algoritmo

- Inicialmente, encontre um valor inicial x_0 próximo à raiz α tal que $f(\alpha) = 0$
- estabeleça uma **tolerância** η

Algoritmo para o método de Newton

1. Calcule
$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$
2. Se $|f(x_1)| \leq \eta$, pare. Se não, faça $x_0 = x_1$ e volte para o passo 1.

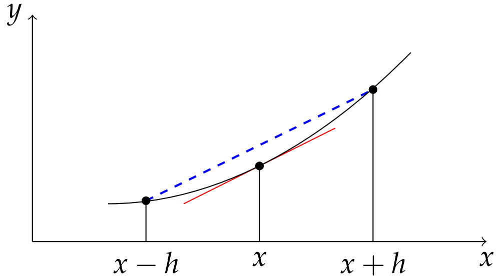
- Ao final da execução do algoritmo, x_1 será a melhor aproximação para a raiz dentro da tolerância pré-estabelecida $|f(x_1)| \leq \eta$
- Mas não há nenhuma garantia de que o método vá convergir para a solução correta!
- Além disso, é melhor estabelecer um critério de parada num número máximo de iterações, para evitar um loop infinito

Método das secantes

- Uma estratégia para evitar o cálculo da derivada é fazer uma aproximação:

$$y'(x) \approx \frac{y(x+h) - y(x-h)}{2h}$$

  para algum valor de h pequeno o suficiente
- geralmente, é suficiente adotar h um pouco maior que a tolerância desejada para a solução
  - por exemplo, se a tolerância é $|f(x_1)| < \eta$, adotar $h \approx 10\eta$ funciona (usualmente)
- Ou seja, estamos efetivamente usando uma *derivação numérica*



- Essa estratégia é conhecida como **Método das secantes**

Plano de aula

Rotinas em `scipy`

- A biblioteca `scipy.optimize` contém diversas rotinas para encontrar o zero de funções
  - inclusive funções de mais de uma variável e sistemas de equações não-lineares
- Documentação completa em `https://docs.scipy.org/doc/scipy/reference/optimize.html`

Exemplo

O código abaixo calcula o zero da função $f(x) = x - \cos x$ começando em $x_0 = 1$, usando a função `newton` da biblioteca `scipy.optimize`:

```
import numpy as np
import scipy.optimize as sp

def f(x):
    return x - np.cos(x)

sol = sp.newton(f, 1.)

print(sol)
```