

# CÁLCULO DE RAÍCES DE ECUACIONES DE UNA VARIABLE

BISECCIÓN. PUNTO FIJO. NEWTON-RAPHSON.

---

Manuel Carlevaro

Departamento de Ingeniería Mecánica

Grupo de Materiales Granulares - UTN FRLP

manuel.carlevaro@gmail.com

Cálculo Avanzado • 2025

 · Xe<sub>La</sub>TeX · 

## Problema:

$$f(x) = 0$$

$$f(x) = g(x) \implies h(x) = f(x) - g(x) = 0$$

## Teorema : Valores intermedios.

Sea  $f : [a, b] \rightarrow \mathbb{R}$  una función **continua** en  $[a, b]$  tal que  $f(a) < f(b)$ . Entonces:  $\forall u \in (f(a), f(b))$  existe  $c \in [a, b]$  tal que  $f(c) = u$ .

## Problema:

$$f(x) = 0$$

$$f(x) = g(x) \implies h(x) = f(x) - g(x) = 0$$

### Teorema : Valores intermedios.

Sea  $f : [a, b] \rightarrow \mathbb{R}$  una función **continua** en  $[a, b]$  tal que  $f(a) < f(b)$ . Entonces:  $\forall u \in (f(a), f(b))$  existe  $c \in [a, b]$  tal que  $f(c) = u$ .

## Estrategia general:

- ▶ Mostrar que existe al menos una solución ( $x^*$ )
- ▶ Aislar una raíz:  $D \subset \mathbb{R}$ ,  $x^* \in D$  y
$$f(x) \neq 0 \quad \forall x \in D \setminus \{x^*\}$$
- ▶ Iterar

## Problema:

$$f(x) = 0$$

$$f(x) = g(x) \implies h(x) = f(x) - g(x) = 0$$

### Teorema : Valores intermedios.

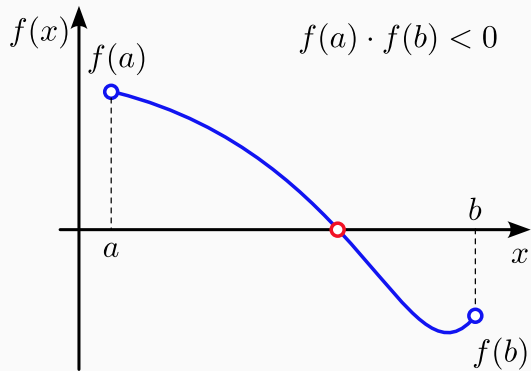
Sea  $f : [a, b] \rightarrow \mathbb{R}$  una función **continua** en  $[a, b]$  tal que  $f(a) < f(b)$ . Entonces:  $\forall u \in (f(a), f(b))$  existe  $c \in [a, b]$  tal que  $f(c) = u$ .

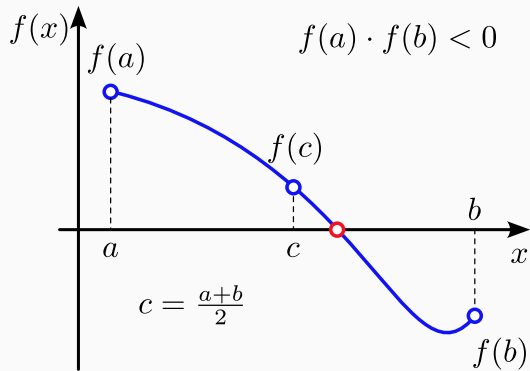
## Estrategia general:

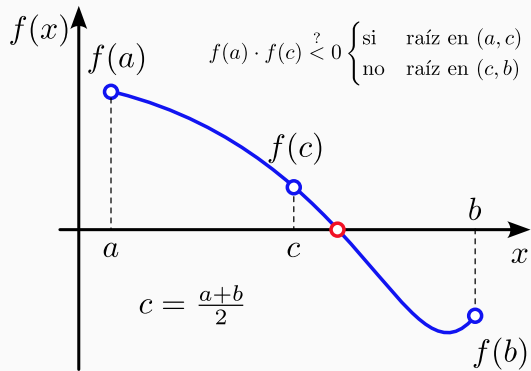
- ▶ Mostrar que existe al menos una solución ( $x^*$ )
- ▶ Aislar una raíz:  $D \subset \mathbb{R}$ ,  $x^* \in D$  y
$$f(x) \neq 0 \quad \forall x \in D \setminus \{x^*\}$$
- ▶ Iterar

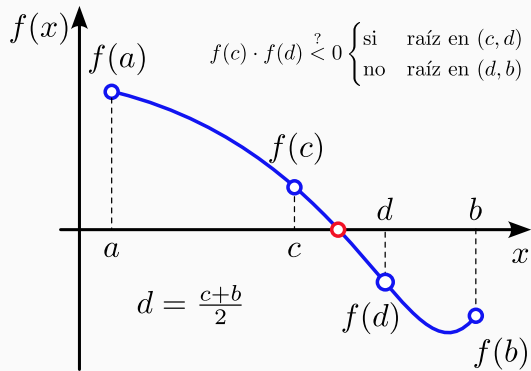
## Métodos:

- ▶ *Bracketing* (cerrados)
  - > Bisección
  - > Posición falsa (*regula falsi*)
  - > ITP
- ▶ Interpolación
  - > Secante
  - > Muller
- ▶ Iterativos (abiertos)
  - > Punto fijo
  - > Newton-Raphson
  - > Secante
  - > Broyden
- ▶ Combinación de cerrados y abiertos
  - > Brent
  - > Ridders

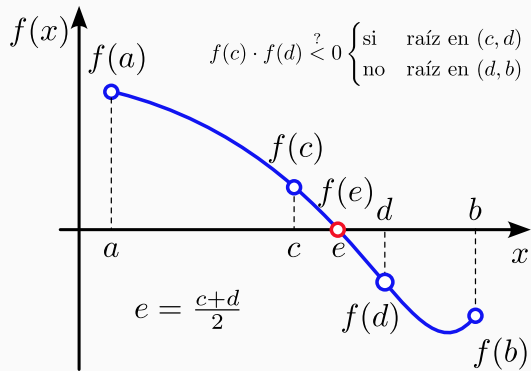


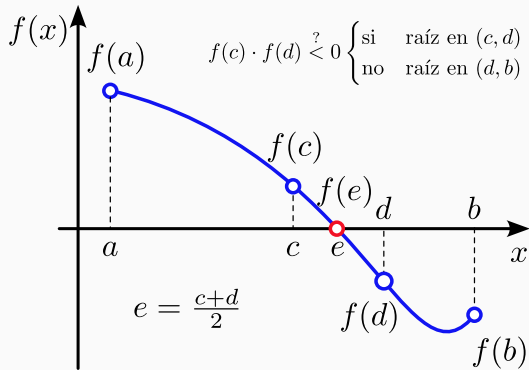












## Teorema : Convergencia y error .

Sea  $[a, b]$  el intervalo inicial, donde  $f(a) \cdot f(b) < 0$ .  
 Defina una raíz aproximada como

$$x_n = \frac{a_{n-1} + b_{n-1}}{2}$$

Entonces existe una raíz  $x^*$  tal que

$$|x^* - x_n| \leq \frac{b - a}{2^n}$$

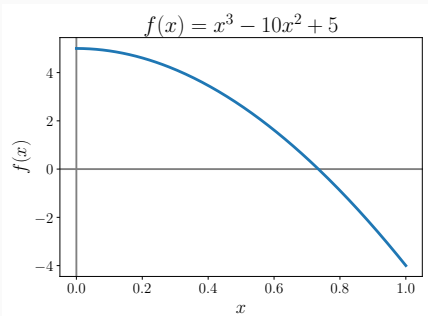
Además, para obtener una precisión de  $|x^* - x_n| \leq \epsilon$   
 es necesario iterar  $n$  veces con

$$n \geq \frac{\log(b - a) - \log \epsilon}{\log 2}$$

## Código

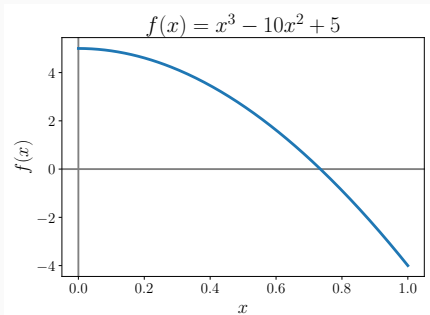
```
1 #!/usr/bin/env python3
2
3 def biseccion(f, x1, x2, switch=0, eps=1.0e-9):
4     """
5     raiz = biseccion(f, x1, x2, switch=0, tol=1.0e-9)
6     Encuentra una raíz de  $f(x) = 0$  por el método de
7     bisección. La raíz debe estar acotada en  $(x1, x2)$ .
8     Haciendo switch = 1 devuelve raiz = None
9     si  $f(x)$  aumenta como resultado de la bisección.
10    """
11    from math import log, ceil
12    f1 = f(x1)
13    if abs(f1) < eps: return x1
14    f2 = f(x2)
15    if abs(f2) < eps: return x2
16    if f1 * f2 > 0:
17        print("Error: la raíz no está acotada.")
18        quit()
19    n = ceil(log(abs(x2 - x1)/eps) / log(2.0))
```

```
20    for i in range(n):
21        x3 = 0.5 * (x1 + x2)
22        f3 = f(x3)
23        if switch and (abs(f3) > abs(f1)) \
24            and (abs(f3) > abs(f2)):
25            return None
26        if abs(f3) < eps: return x3
27        if f2 * f3 < 0.0:
28            x1 = x3
29            f1 = f3
30        else:
31            x2 = x3
32            f2 = f3
33    return (x1 + x2) / 2.0
34
35 def f(x):
36     return x**3 - 10 * x**2 + 5
37
38 raiz = biseccion(f, 0.0, 1.0)
39 print(f"La raíz de  $f(x) = 0$  es {raiz}")
```



```
$ ./biseccion.py
```

```
La raíz de  $f(x) = 0$  es 0.7346035079099238
```



```
$ ./biseccion.py
La raiz de f(x) = 0 es 0.7346035079099238
```

## SciPy: optimize

```
1 #!/usr/bin/env python3
2
3 from scipy import optimize
4
5 def f(x):
6     return x**3 - 10 * x**2 + 5
7
8 raiz = optimize.bisect(f, 0.0, 1.0)
9 print(f"La raiz de f(x) = 0 es {raiz}")
```

```
$ ./scipy-biseccion.py
La raiz de f(x) = 0 es 0.7346035077880515
```

## PUNTO FIJO

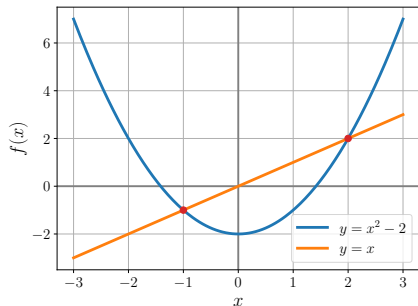
$p$  es un **punto fijo** de  $g$  si  $g(p) = p$ .

Si  $g$  tiene un punto fijo en  $p$ , entonces:

$$f(x) = x - g(x)$$

tiene un cero en  $p$ .

**Ejemplo:**  $g(x) = x^2 - 2 \rightarrow p = \{-1, 2\}$ .



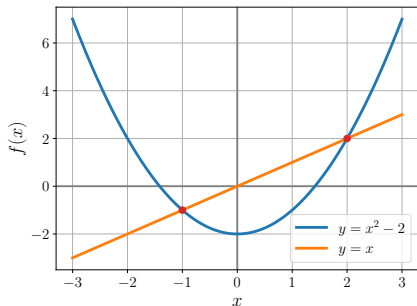
$p$  es un **punto fijo** de  $g$  si  $g(p) = p$ .

Si  $g$  tiene un punto fijo en  $p$ , entonces:

$$f(x) = x - g(x)$$

tiene un cero en  $p$ .

**Ejemplo:**  $g(x) = x^2 - 2 \rightarrow p = \{-1, 2\}$ .



## Teorema : Existencia y unicidad.

- i) Si  $g \in \mathcal{C}[a, b]$  y  $g(x) \in [a, b] \forall x \in [a, b]$ , entonces  $g$  tiene por lo menos un punto fijo en  $[a, b]$ .
- ii) Si, además,  $g'(x)$  existe en  $[a, b]$  y hay una constante positiva  $k < 1$  con

$$|g'(x)| \leq k, \forall x \in [a, b]$$

entonces existe exactamente un punto fijo en  $[a, b]$ .

## Iteración de punto fijo:

- ▶ Aproximación inicial  $p_0$
- ▶  $\{p_n\}_{n=0}^{\infty}$  con  $p_n = g(p_{n-1})$ ,  $n \geq 1$
- ▶ Si la sucesión converge a  $p$  y  $g$  es continua:

$$p = \lim_{n \rightarrow \infty} p_n = \lim_{n \rightarrow \infty} g(p_{n-1}) = g\left(\lim_{n \rightarrow \infty} p_{n-1}\right) = g(p)$$

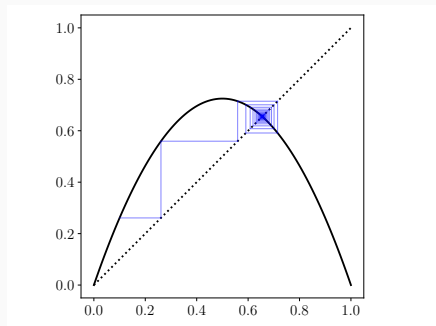
## Código

```
1 #!/usr/bin/env python3
2
3 def punto_fijo(g, x0, max_n=200, eps=1.0e-5):
4     """
5     p = punto_fijo(g, x0, max_n=200, eps=1.0e-5)
6     Itera la relación  $x = g(x_0)$ ,  $x_0 \leftarrow x$ 
7     hasta agotar el número de iteraciones (en ese
8     caso devuelve None), o hasta que se produzca
9     convergencia con la tolerancia eps, en la que
10    devuelve el punto fijo.
11    """
12    for k in range(max_n):
13        x = g(x0)
14        delta_x = x - x0
15        print(f"{k:3d} {x:1.16f} {delta_x:1.16f}")
16        if abs(delta_x / x) < eps:
17            break
18        x0 = x
19    else:
20        x = None
21    return x
```

```
23 def g(x):
24     """
25     Mapeo logístico  $x_{n+1} = r x_n (1 - x_n)$ 
26     """
27     r = 2.9 # Un punto fijo
28     # r = 3.1 # Atractor de dos puntos
29     # r = 4.0 # Región caótica
30     return r * x * (1 - x)
31
32 pfijo = punto_fijo(g, 0.1)
33 print(f"El punto fijo de g(p) = p es {pfijo}")
```

```
$ ./pfijo.py
0 0.2610000000000000 0.1610000000000000
1 0.5593491000000000 0.2983491000000000
2 0.7147852845546510 0.1554361845546509
...
94 0.6551757611798358 0.0000070667498989
95 0.6551694011125498 -0.0000063600672859
El punto fijo de g(p) = p es 0.6551694011125498
```

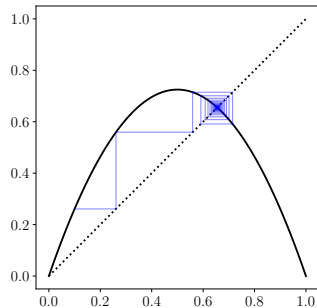




$$r = 2.9$$

$$p = g(p) = \frac{19}{29}$$

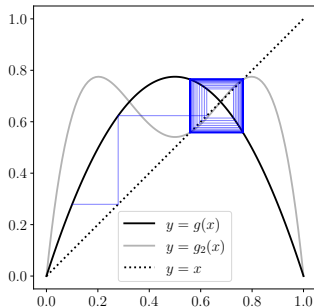
$$g'(p) = -\frac{9}{10}$$



$$r = 2.9$$

$$p = g(p) = \frac{19}{29}$$

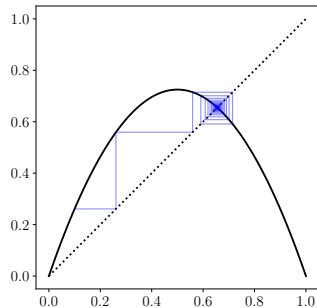
$$g'(p) = -\frac{9}{10}$$



$$r = 3.1$$

$$p = g(p) = \frac{21}{31}$$

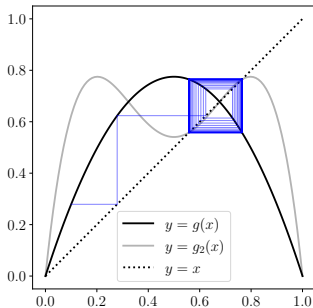
$$g'(p) = -\frac{11}{10}$$



$$r = 2.9$$

$$p = g(p) = \frac{19}{29}$$

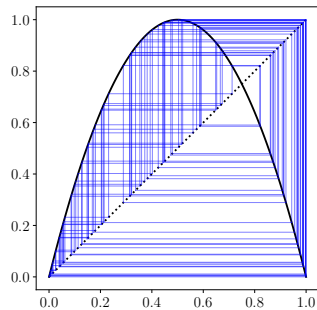
$$g'(p) = -\frac{9}{10}$$



$$r = 3.1$$

$$p = g(p) = \frac{21}{31}$$

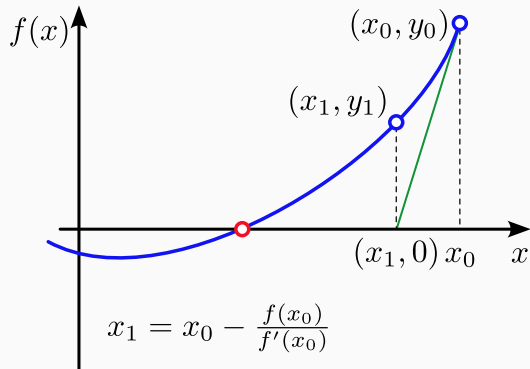
$$g'(p) = -\frac{11}{10}$$



$$r = 4.0$$

$$p = g(p) = \frac{3}{4}$$

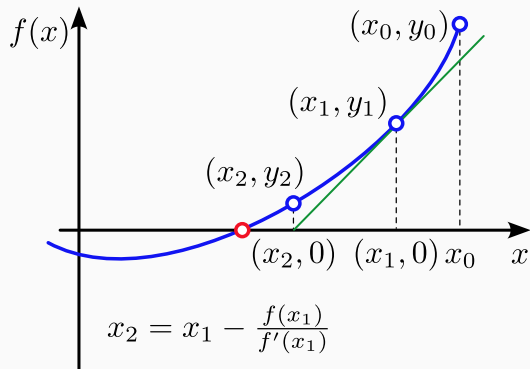
$$g'(p) = -2$$



$$\frac{y - y_0}{x - x_0} = f'(x_0) \Rightarrow y = f(x_0) + f'(x_0)(x - x_0)$$

Haciendo  $y = 0$ :

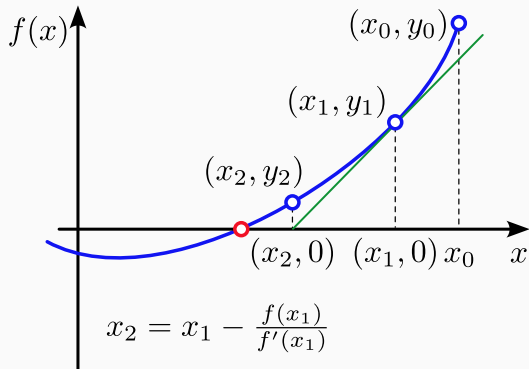
$$x = x_0 - \frac{f(x_0)}{f'(x_0)} \mapsto x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$



$$\frac{y - y_0}{x - x_0} = f'(x_0) \Rightarrow y = f(x_0) + f'(x_0)(x - x_0)$$

Haciendo  $y = 0$ :

$$x = x_0 - \frac{f(x_0)}{f'(x_0)} \mapsto x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$



$$\frac{y - y_0}{x - x_0} = f'(x_0) \Rightarrow y = f(x_0) + f'(x_0)(x - x_0)$$

Haciendo  $y = 0$ :

$$x = x_0 - \frac{f(x_0)}{f'(x_0)} \mapsto x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Serie de Taylor ( $x_n \approx x^*$ ):

$$f(x) = f(x_n) + (x - x_n)f'(x_n) + \frac{1}{2}(x - x_n)^2 f''(\xi)$$

$\xi_n \in (x, x_n)$  Hacemos  $f(x) = 0$ :

$$x = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{1}{2}(x - x_n)^2 \frac{f''(\xi_n)}{f'(x_n)}$$

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

## Código:

```
1 #!/usr/bin/env python3
2
3 from scipy import optimize
4
5 def NR(f, df, x, n=30, eps=1.0e-9):
6     """
7     raiz, iter = NR(f, df, x, n=30, tol=1.0e-9)
8     Encuentra una raíz de  $f(x) = 0$  por el método de
9     Newton-Raphson. Si se alcanza el máximo número
10    de iteraciones  $n$ , devuelve None
11    """
12    for i in range(n):
13        try:
14            dx = -f(x) / df(x)
15        except ZeroDivisionError:
16            print("No hay convergencia.")
17            quit()
18        x += dx
19        if abs(dx) < eps:
20            return x, i
21    print("Límite de iteraciones alcanzado.")
22    return None, i
```

```
24 def f(x):
25     return x**3 - 10 * x**2 + 5
26
27 def df(x):
28     return 3 * x**2 - 20 * x
29
30 raiz, iteraciones = NR(f, df, 1.0)
31 print(f"La raíz de  $f(x) = 0$  es {raiz}.")
32 print(f"Iteraciones: {iteraciones}")
33
34 raiz_2 = optimize.newton(f, 1.0, fprime=df)
35 print(f"La raíz 2 de  $f(x) = 0$  es {raiz_2}.")
36 print(f"Diferencia: {abs(raiz_2 - raiz)}")
```

```
$ ./newton-raphson.py
La raíz de  $f(x) = 0$  es 0.7346035077893033.
Iteraciones: 4
La raíz 2 de  $f(x) = 0$  es 0.7346035077893033.
Diferencia: 0.0
```

**Convergencia:** Serie de Taylor alrededor de  $\alpha$ ,  
( $f(\alpha) = 0$ ):

$$f(\alpha) = f(x_n) + f'(x_n)(\alpha - x_n) + R_1 \quad (1)$$

Forma de Lagrange de  $R_1$ :

$$R_1 = \frac{1}{2!} f''(\xi_n)(\alpha - x_n)^2$$

donde  $\xi_n$  está entre  $\alpha$  y  $x_n$ . Como  $f(\alpha) = 0$ :

$$0 = f(x_n) + f'(x_n)(\alpha - x_n) + \frac{1}{2!} f''(\xi_n)(\alpha - x_n)^2 \quad (2)$$

Dividiendo (2) por  $f'(x_n)$  y acomodando:

$$\frac{f(x_n)}{f'(x_n)} + (\alpha - x_n) = -\frac{f''(\xi_n)}{2f'(x_n)}(\alpha - x_n)^2 \quad (3)$$

$$\text{NR: } x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad (4)$$

resulta:

$$\underbrace{\alpha - x_{n+1}}_{\epsilon_{n+1}} = -\frac{f''(\xi_n)}{2f'(x_n)} \underbrace{(\alpha - x_n)_{\epsilon_n}}_{\epsilon_n}^2$$

$$|\epsilon_{n+1}| = \frac{|f''(x_n)|}{2|f'(x_n)|} \cdot \epsilon_n^2 \quad (5)$$

Convergencia al menos **cuadrática** si:

- ▶  $f'(x) \neq 0, \forall x \in I \equiv [\alpha - |\epsilon_0|, \alpha + |\epsilon_0|]$
- ▶  $f''(x) \in \mathcal{C}, \forall x \in I$
- ▶  $M|\epsilon_0| < 1$  con

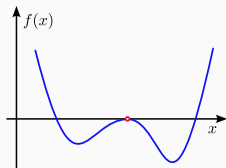
$$M = \frac{1}{2} \left( \sup_{x \in I} |f''(x)| \right) \left( \sup_{x \in I} \frac{1}{|f'(x)|} \right)$$

$|\epsilon_{n+1}| \leq M \cdot \epsilon_n^2$

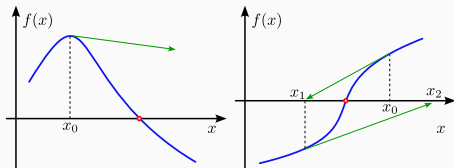


## Problemas:

► Raíz doble:

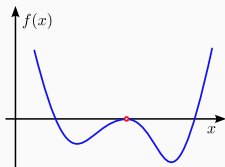


► Divergencia:

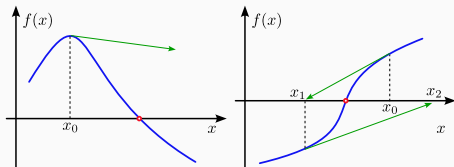


## Problemas:

### ► Raíz doble:



### ► Divergencia:



## Scipy.optimize

```
1 #!/usr/bin/env python3
2
3 from scipy import optimize
4
5 def f(x):
6     return x**3 - 10 * x**2 + 5
7
8 def df(x):
9     return 3 * x**2 - 20 * x
10
11 metodos = ['bisect', 'newton', 'brentq', 'toms748']
12
13 for m in metodos:
14     sol = optimize.root_scalar(f, x0=1.0,
15                               bracket=[0.01, 2], fprime=df, method=m)
16     print(f"Método: {m:7s}, raiz = {sol.root}")
17     print(sol)
```

```
$ ./root_scalar.py
```

```
Método: bisect , raiz = 0.7346035077910437
```

```
Método: newton , raiz = 0.7346035077893033
```

```
Método: brentq , raiz = 0.7346035077893016
```

```
Método: toms748, raiz = 0.7346035077893033
```

- ▶ R.L. Burden, D.J. Faires y A.M. Burden. ***Análisis numérico***. 10.<sup>a</sup> ed. Mexico: Cengage Learning, 2017. Capítulo 2.
- ▶ J.F. Epperson. ***An Introduction to Numerical Methods and Analysis***. 2.<sup>a</sup> ed. Hoboken, United States: John Wiley & Sons, 2013. Capítulo 3.
- ▶ J. Kiusalaas. ***Numerical Methods in Engineering with Python 3***. Cambridge, United Kingdom: Cambridge University Press, 2013. Capítulo 4.
- ▶ E. Kreyszig, H. Kreyszig y E.J. Norminton. ***Advanced Engineering Mathematics***. Hoboken, USA: John Wiley & Sons, Inc, 2011. Capítulo 19.