Boletín de Ejercicios:
Resolución Numérica de Ecuaciones No Lineales

(Punto fijo, Bisección, Newton–Raphson y Búsqueda por Fuerza Bruta)

Física II - Grado en Robótica

6 de mayo de 2025

Introducción

A continuación presentamos cuatro métodos clásicos para hallar raíces de ecuaciones no lineales $f(x) = 0$.

1. Iteración de Punto Fijo

Reescribimos la ecuación en la forma $x = g(x)$ y generamos la sucesión $x_{n+1} = g(x_n)$ hasta que $|x_{n+1} - x_n| < \varepsilon$. La convergencia depende de la elección de g y de la semilla x_0. Si $|g'(x\_*)| < 1$ en la raíz, la convergencia es lineal.

```
import math

def fixed_point(g, x0, tol=1e-8, maxiter=50):
    for _ in range(maxiter):
        x1 = g(x0)
        if abs(x1 - x0) < tol:
            return x1
        x0 = x1
    raise RuntimeError('No convergence')
```

2. Método de la Bisección

Requiere un intervalo $[a, b]$ con $f(a)\, f(b) < 0$. Divide sucesivamente el intervalo a la mitad garantizando convergencia (lenta, lineal). No necesita semillas pero sí acotar la solución en un intervalo.

```
def bisection(f, a, b, tol=1e-8):
    fa, fb = f(a), f(b)
    if fa * fb > 0:
        raise ValueError('Sin cambio de signo en el intervalo')
    while (b - a) / 2 > tol:
        c = 0.5 * (a + b)
        if f(c) == 0:
            return c
        if fa * f(c) < 0:
            b, fb = c, f(c)
        else:
```

```
            a, fa = c, f(c)
    return 0.5 * (a + b)
```

3. Newton–Raphson

Utiliza la aproximación lineal $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$, con convergencia cuadrática (muy rápida) cerca de la raíz. Depende críticamente de la semilla x_0 y de la existencia de $f'(x) \neq 0$.

```
def newton(f, df, x0, tol=1e-10, maxiter=25):
    for _ in range(maxiter):
        fx, dfx = f(x0), df(x0)
        if dfx == 0:
            raise ZeroDivisionError('Derivada nula')
        x1 = x0 - fx / dfx
        if abs(x1 - x0) < tol:
            return x1
        x0 = x1
    raise RuntimeError('No converge')
```

4. Búsqueda por Fuerza Bruta

Se evalúa f sobre una malla uniforme y se detectan cambios de signo o mínimos absolutos. No garantiza exactitud alta y es lento, aunque robusto y — con refinamiento progresivo — puede conseguir cualquier tolerancia a cambio de un tiempo de computo mucho mayor que otros métodos.

```
import numpy as np

def brute_force(f, a, b, N=10000):
    xs = np.linspace(a, b, N)
    fs = f(xs)
    idx = np.argmin(np.abs(fs))
    return xs[idx]
```

Dependencia de la semilla. Los métodos de punto fijo y Newton–Raphson pueden diverger o converger a raíces diferentes según la elección de x_0. En los ejercicios se sugiere un rango de semillas; prueba varias y compara.

Problemas Propuestos

Para cada ecuación: 1) plantea un $g(x)$ apropiado y aplica iteración de punto fijo; 2) encuentra un intervalo con cambio de signo y usa bisección; 3) deriva $f'(x)$ y utiliza Newton–Raphson; 4) realiza una búsqueda bruta en un intervalo adecuado. Compara la velocidad de convergencia, el número de iteraciones y la sensibilidad a la semilla.

1. $e^{-x} - x = 0$ (Raíz cercana a $x \approx 0{,}567$)
2. $\ln(x+2) + x^2 - 3 = 0, \quad x > -2$
3. $x \cos x - 1 = 0$

4. $\tanh x - 0{,}5x = 0$
5. $e^{0,1x} - x^3 + x - 1 = 0$
6. $\sqrt{x+3} - \cos x = 0, \quad x \geq -3$

Indicaciones

- Para (1), define $g(x) = e^{-x}$.
- En (2), una opción es $g(x) = \sqrt{3 - \ln(x+2)}$ (revisa convergencia con $g'(x)$).
- Para Newton en (3): $f'(x) = \cos x - x \sin x$.
- Prueba con distintas semillas x_0 en (4) y observa cuándo diverge.
- En (5), intenta dos semillas muy separadas para mostrar convergencia a raíces diferentes.
- Para (6) utiliza un dominio inicial amplio y compara fuerza bruta con los demás métodos.

Recomendación: documenta en una tabla el número de evaluaciones de f y el error final para cada método y ecuación.