# Método de Brent

**Autor:** Wily Calib Caira Huancollo
**Escuela Profesional:** Estadística e Informática
**Docente:** Ing. Fred Torres Cruz
**Universidad Nacional del Altiplano - Puno**

# Definición del método de Brent

El **método de Brent** es un algoritmo numérico híbrido utilizado para encontrar raíces reales de ecuaciones no lineales de la forma:

$$f(x) = 0$$

Este método combina tres técnicas fundamentales:

- **Bisección:** garantiza convergencia, aunque es lenta.
- **Secante:** método rápido, no requiere derivadas.
- **Interpolación cuadrática inversa:** mejora la precisión.

En cada iteración, el método selecciona automáticamente la técnica más adecuada según la estabilidad y el progreso de la convergencia. De esta manera, logra la **seguridad del método de bisección** y la **rapidez del método de la secante**.

## Características principales

- No requiere derivadas de $f(x)$.
- Requiere que $f(a) \cdot f(b) < 0$ para iniciar.
- Tiene convergencia **superlineal** (orden entre 1.6 y 2.0).
- Es el método usado por defecto en librerías como `SciPy`, `MATLAB`, `R` y `Julia`.

# Ejemplo práctico

Resolver la ecuación:

$$f(x) = x^3 - 2x - 5 = 0$$

en el intervalo $[2, 3]$.

### Evaluación inicial

$$f(2) = -1, \quad f(3) = 16$$

Como $f(2) \cdot f(3) < 0$, existe una raíz dentro del intervalo.

### Iteraciones del método

| Iteración | $a$ | $b$ | $f(a)$ | $f(b)$ | Aproximación |
|---|---|---|---|---|---|
| 1 | 2.0 | 3.0 | -1 | 16 | 2.5 |
| 2 | 2.0 | 2.5 | -1 | 5.625 | 2.09 |
| 3 | 2.09 | 2.5 | -0.02 | 5.625 | 2.09455 |

### Resultado final

$$x \approx 2{,}09455 \quad \text{y} \quad f(2{,}09455) \approx 0$$

Por lo tanto, la raíz de la ecuación se encuentra en $x \approx 2{,}09455$.

## Código en Python

```
import math

# Funci n ejemplo
def f(x):
    return x**3 - 2*x - 5

# M todo de Brent (simplificado)
def brent(f, a, b, tol=1e-12, maxiter=100):
    fa, fb = f(a), f(b)
    if fa * fb > 0:
        raise ValueError("f(a) y f(b) deben tener signos opuestos.")
    c, fc = a, fa
    d = e = b - a
    for i in range(maxiter):
        if abs(fc) < abs(fb):
            a, b, c = b, c, b
            fa, fb, fc = fb, fc, fb
        tol_act = 2 * tol * max(1, abs(b))
        m = 0.5 * (c - b)
        if abs(m) <= tol_act or fb == 0:
            return b
        if abs(e) < tol_act or abs(fa) <= abs(fb):
```

```
            d = e = m
        else:
            s = fb / fa
            if a == c:
                p = 2 * m * s
                q = 1 - s
            else:
                q = fa / fc
                r = fb / fc
                p = s * (2 * m * q * (q - r) - (b - a) * (r - 1))
                q = (q - 1) * (r - 1) * (s - 1)
            if p > 0:
                q = -q
            p = abs(p)
            if 2 * p < min(3 * m * q - abs(tol_act * q), abs(e * q)):
                e, d = d, p / q
            else:
                d = e = m
        a, fa = b, fb
        if abs(d) > tol_act:
            b += d
        else:
            b += tol_act if m > 0 else -tol_act
        fb = f(b)
        if (fb > 0 and fc > 0) or (fb < 0 and fc < 0):
            c, fc = a, fa
    return b

# Ejemplo de uso
raiz = brent(f, 2, 3)
print("Ra z aproximada:", raiz)
```

Listing 1: Implementación del método de Brent en Python

# Conclusión

El método de Brent combina lo mejor de los métodos de bisección y secante, logrando una convergencia rápida y segura sin necesidad de derivadas. Gracias a su estabilidad numérica, es uno de los algoritmos más utilizados en **programación científica e ingeniería computacional**.