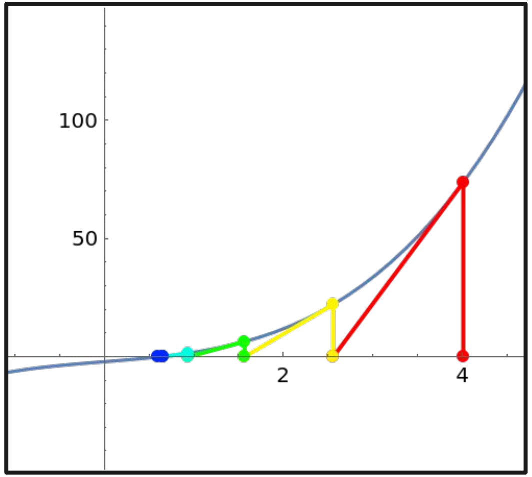
Clase 1:
Búsqueda de raíces

Búsqueda de raíces



- Motivación en el marco de la materia
- Algoritmos (root-finding algorithms)
  - Bisección
  - Newton-Raphson
  - Secante
- Comparación
- Uso de scipy
- Bibliografía

Motivación en el marco de la materia

- Sistemas dinámicos, autónomos, unidimensionales, regidos por ODE

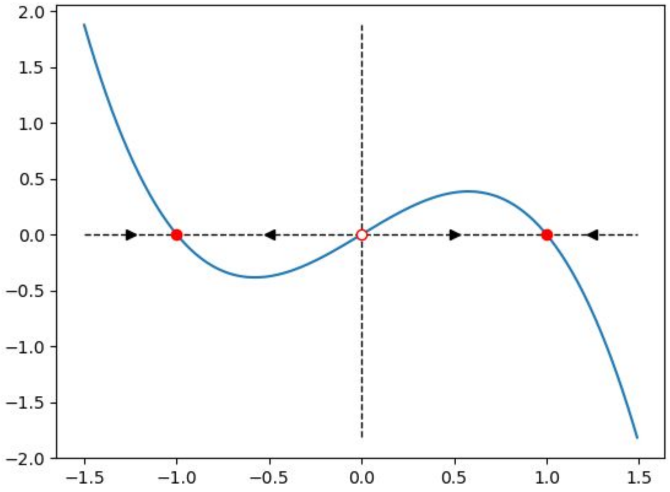
  $\dot{x} = dx/dt = f(x) \longrightarrow$ campo vector

- Encontrar puntos fijos

  $\dot{x} = f(x) = 0 \longrightarrow$ raíces de f(x)

  - Analíticamente
  - Gráficamente
  - Métodos numéricos

Laboratorio numérico

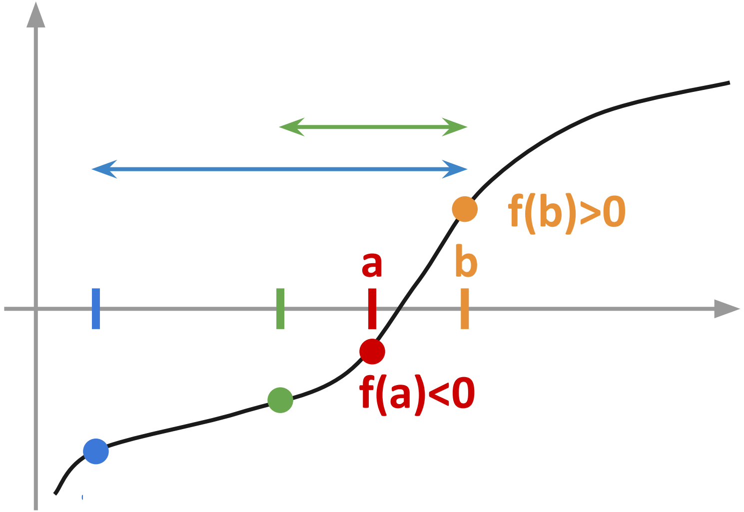


Método de bisección

Sea la función f(x) continua en el intervalo [a, b] tal que f(a).f(b) < 0, entonces, por teorema de valor intermedio, existe al menos una raíz de f(x) en [a, b]

Defino c como el punto medio entre a y b

$$c = (a + b)/2$$



Tomo c como el a o b para que se cumpla f(a).f(b) < 0, y así sucesivamente

Varios criterios de corte, por ejemplo,

$|b - a| < \epsilon \longrightarrow$ tolerancia

Error, error relativo, f(c), iteraciones,…

Método de bisección

- Iterativo
- Condición inicial: intervalo [a, b]
- La función f(x) tiene que ser continua y cumplir que $f(a).f(b) < 0$

\+ Siempre converge si se cumplen las hipótesis

\- Convergencia lenta

\+ Útil cuando no tenemos una buena estimación inicial

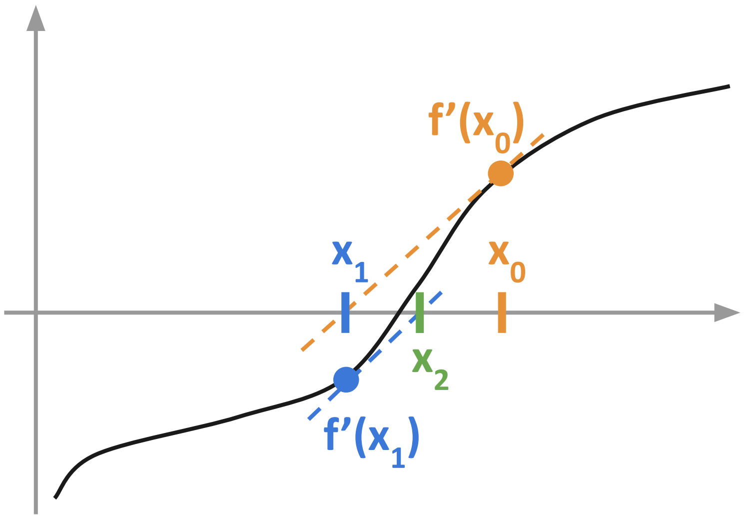
Método de Newton-Raphson

Sea la función f(x) continua y diferenciable en x_0, si x_0 es una buena estimación de la raíz, puedo hacer el desarrollo de Taylor a primer orden

$$y = f'(x_0)(x - x_0) + f(x_0)$$

Puedo definir una relación recursiva con la cual la tangente (potencialmente) me acercará a la raíz real, hasta un corte

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



Problemas de convergencia?

Método de Newton-Raphson

- Iterativo
- Condición inicial: un punto x_0
- La función f(x) y su derivada tienen que ser continuas

- No siempre converge, puede fallar

\+ Convergencia muy rápida, cuando está cerca de la raíz

- Necesito conocer la derivada y diverge si la estimación inicial es mala

Método de la secante

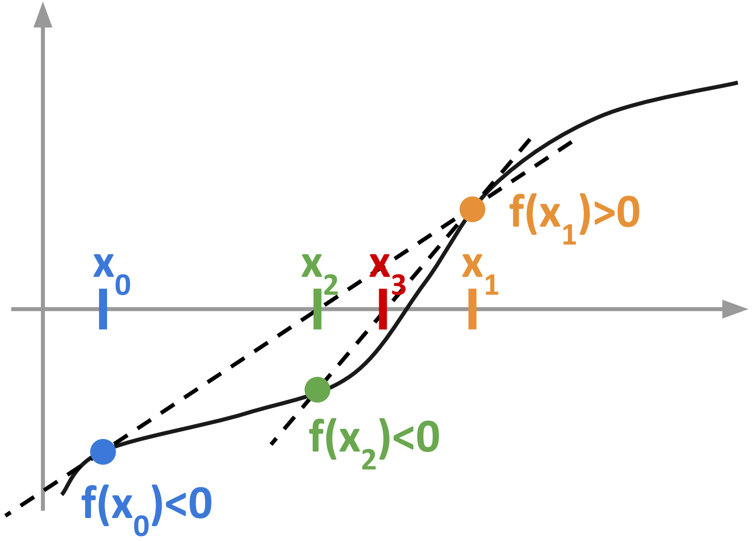
Sea la función f(x) continua en el intervalo [a, b] tal que f(a).f(b) < 0, entonces, por teorema de valor intermedio, existe al menos una raíz de f(x) en [a, b]

Uso la secante entre a y b para obtener una mejor estimación de la raíz

$$y = \frac{f(b) - f(a)}{b - a}(x - a) + f(a)$$

$$x = a - f(a)\frac{b - a}{f(b) - f(a)}$$

Lo hago sucesivamente con valores de x_i y x_j como a y b, hasta un corte



Método de la secante

- Iterativo
- Condición inicial: intervalo [a, b]
- La función f(x) tiene que ser continua y cumplir que $f(a).f(b) < 0$
- Es una mezcla entre el método de bisección y el método de Newton-Raphson, porque en lugar de aproximarse por el valor medio usa la secante que es una buena estimación de la tangente

\+ Siempre converge si se cumplen las hipótesis

\- Puede ser menos eficiente que Newton-Raphson

\+ No requiere de la derivada, útil cuando calcularla es costoso

Comparación de métodos

Método	Descripción	Ventajas	Desventajas
Bisección	Divide el intervalo por la mitad	Sencillo, rápido, y garantiza convergencia	Convergencia lenta (lineal)
Newton-Raphson	Utiliza función y derivada	Convergencia rápida (cuadrática)	Puede no converger
Secante	No requiere cálculo de derivadas pero es similar a Newton-Raphson	No requiere derivada, más rápido que la bisección	Convergencia no tan rápida (superlineal)

Otros métodos

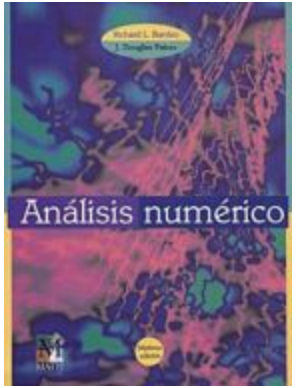
- Método de regula falsi
- Método de punto fijo
- Interpolación cuadrática inversa (IQI)
- Método de Brent
- …
- Combinaciones de métodos
- Aleatorización o automatización de estimaciones iniciales (varias raíces)

Funciones integradas en paquetes de Python

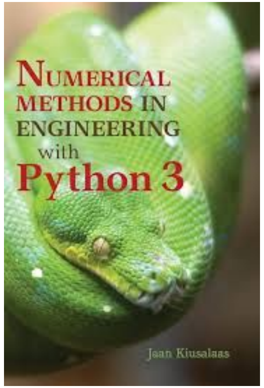
Scipy tiene varias funciones para estos métodos y otros

- Bisección
  - scipy.optimize.bisect(f, a, b, xtol, maxiter)
- Newton-Raphson o secante (si fprime=None)
  - scipy.optimize.newton(func, x0, fprime, tol, maxiter)
- Otros
  - scipy.optimize.fixed_point (punto fijo)
  - scipy.optimize.brentq (Brent, similar a secante)
  - scipy.optimize.fsolve; scipy.optimize.root (problema multivariado)

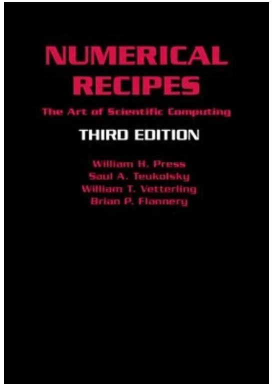
Bibliografía recomendada



Burden & Faires 2010



Kiusalaas 2013



Press et al 2007

Google

stack overflow

towards data science