Optimisation

Seance 1

Objectif :

$\left\{ \begin{array}{l} \text{chercher } x \approx ? / \\ f(x) = 0 \end{array} \right.$

. Methode predefini :

1. bisect.

bisect (f, a, b)

Condition : $f(a) \times f(b) < 0$

2. fsolve

fsolve (f, a)

Condition $a \in$ domaine $f(x)$

resultat array

3. newton

newton (f, a)

// comme fsolve (f, a)

resultat nbr reelle

4. Solve

var ("x")

solve (f(x))

Condition on doit declaré le x comme var

Methode à definir

1. Dichotomique :

$a, b \rightarrow x = \frac{a+b}{2}$

$a = x$ si $f(x) \times f(b) \leq 0$

$b = x$ sinon

condition $|b - a| > \varepsilon$

2. Lagrange

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

$a = x$: $f(x) - f(b) \leq 0$

$b = x$: sinon

Condition $|f(x)| > \varepsilon$

3. Newton

$x_0 \longrightarrow x = x_0 - \frac{f(x_0)}{f'(x_0)}$

simplifier $f'(x_0)$:

$x_0 \rightarrow x = x_0 - \frac{2h\, f(x_0)}{f(x_0 + h) - f(x_0 - h)}$

condition $|f(x_0)| > \varepsilon$

4. La Secante

$$x_i, x_j \rightarrow x = \frac{x_i f(x_j) - x_j f(x_i)}{f(x_j) - f(x_i)}$$

$$(x_i, x_j) = (x, x_i)$$

<u>Condition</u> $|x_j - x_i| > \varepsilon$

Optimisation Systeme

Chapitre 1 : Resolut° approché de $f(x) = 0$

$x \approx ? \;/\; f(x) \approx 0$

Methode predefinie :

1. bisect
2. fsolve
3. newton

$\} \in$ scipy.optimize

4. $\underbrace{\text{solve}}_{\text{fct}} \rightarrow \in \underbrace{\text{sympy}}_{\text{biblio à importer}}$

Importat° des biblio :

→ methode 1 : from scipy.optimize import *

→ methode 2 : from scipy.optimize import bisect, fsolve, newton

1. La fct bisect : bisect(f, a, b)
   → Elle prend en parametres a et b
   → $\boxed{f(a) \times f(b) < 0}$ condition

2. La fct fsolve : fsolve(f, a)
   . a ∈ l'ens du domaine du f.
   . solution dans un array !!

3. newton (f, a)

→ comme la fct "fsolve" juste la solution réelle mais pas tableau.

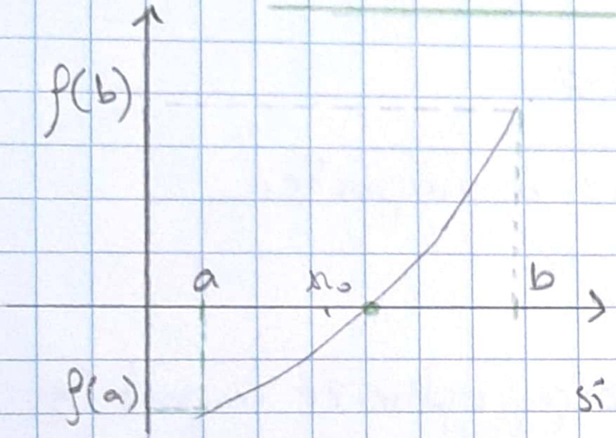
4. from sympy import var, solve

→ var(" x")

→ solve (f(x))

- Les methodes qu'on va definir manuellement:

1. Methode Dichotomique:



$a, b \rightarrow \boxed{x = \frac{a+b}{2}}$

$a = x$ si $f(x) \times f(b) < 0$; $b = x$ sinon

cond tq $|b-a| > \varepsilon$

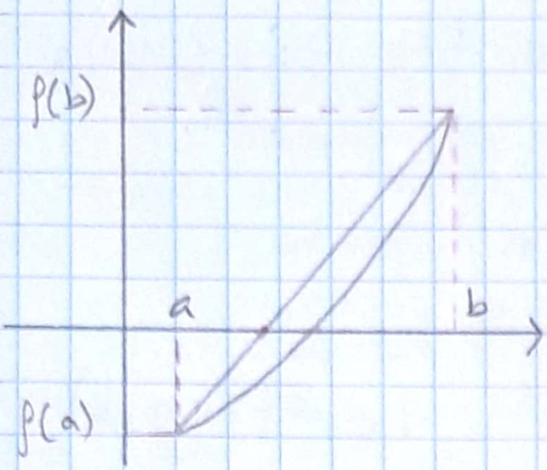
Methode iterative,

```
def dichotomiqueI(f, a, b, eps=10**(-10)):
    while abs(b-a) > eps:
        x = (a+b)/2
        if f(x) * f(b) <= 0:
            a = x
        else:
            b = x
    return (a+b)/2
```

Methode Reccursive :

```
def dichotomiqueR(f, a, b, eps= 10**(-10)):
    if abs(b-a) > eps:
        x = (a+b)/2
        if f(x)*f(b) <= 0:
            return dichotomiqueR(f, x, b)
        else:
            return dichotomiqueR(f, a, x)
    else:
        return (a+b)/2
```

2. Methode de Lagrange :



$a, b \rightarrow x = \dfrac{a\,f(b) - b\,f(a)}{f(b) - f(a)}$

$a = x$ si $f(x) * f(b) <= 0$

$b = x$ sinon

condition tq $|f(x)| > \varepsilon$

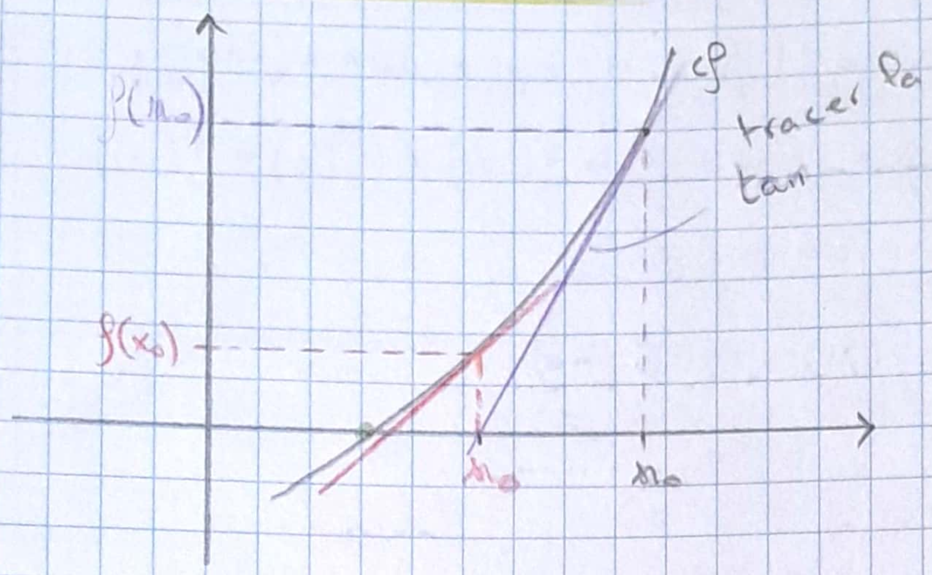
Methode iterative :

```
def lagrangeI (f, a, b, eps = 10 ** (-10)) :
    x = (a * f(b) - b * f(a)) / (f(b) - f(a))
    while abs (f(x)) > eps :
        if f(x) * f(b) <= 0 :
            a = x
        else :
            b = x
        x = (a * f(b) - b * f(a)) / (f(b) - f(a))
    return x
```

Methode reccursive :

```
def lagrangeR(f, a, b, eps = 10**(-10)):
    x = (a * f(b) - b * f(a)) / (f(b) - f(a))
    if abs(f(x)) > ε :
        if f(x) * f(b) <= 0 :
            a = x
        else :
            b = x
        return lagrangeR(f, a, b)
    else :
        return x
```

- Methode de Newton :



- Pour chaque x_0, on trace la tangente qui passe par l'intersect° entre le x_0 et la courbe.
- On nomme l'intersection entre la tangente et l'axe d'abscisse par le nouveau x_0.

→ On refait ces 2 étapes jusqu'à avoir la valeur de x_0 proche de l'intersect° de la courbe avec l'axe d'abscisse.

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Demonstrat° :

Pour trouver le "x", il faut resoudre ce syst.

$$\begin{cases} y = 0 & \text{(intersect° courbe et l'axe "X")} \\ y = f(x_0) + (x - x_0) f'(x_0) & \text{éq de la tan} \end{cases}$$

si on remplace y par 0, on obtient :

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

$$\Rightarrow \quad x - x_0 = -\frac{f(x_0)}{f'(x_0)}$$

$$\Rightarrow \quad \boxed{x = x_0 - \frac{f(x_0)}{f'(x_0)}}$$

→ Comment trouver $f'(x_0)$???

À l'aide du dev. limité :

$$\boxed{f(x) = f(x_0) + (x - x_0) f'(x_0) + \frac{(x-x_0)^2}{2} f''(x_0) + R}$$

R : reste du dev limité

→ De cette formule on doit sortir le terme $f'(x_0)$

⇒ Donc on a besoin de remplacer $x = x_0 + h$
et $x = x_0 - h$ dans la fct $f(x)$.
et faire la soustraction.

$$- \left\{ \begin{array}{l} f(x_0 + h) = f(x_0) + h f'(x_0) + \frac{h^2}{2} f''(x_0) \\ f(x_0 - h) = f(x_0) - h f'(x_0) + \frac{h^2}{2} f''(x_0) \end{array} \right.$$

$$\Rightarrow \quad f'(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h}$$

$$x_0 \longrightarrow x = x_0 - \frac{2h\, f(x_0)}{f(x_0+h) - f(x_0-h)}$$

$$\text{tq} \quad |f(x_0)| > \varepsilon.$$

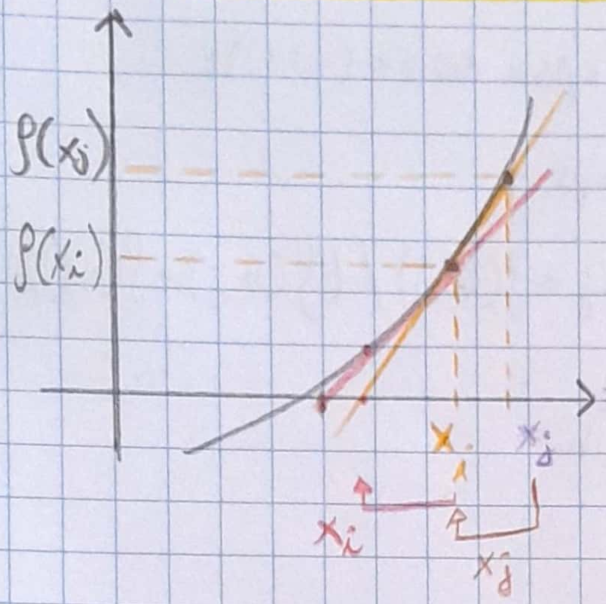
Methode Iterative :

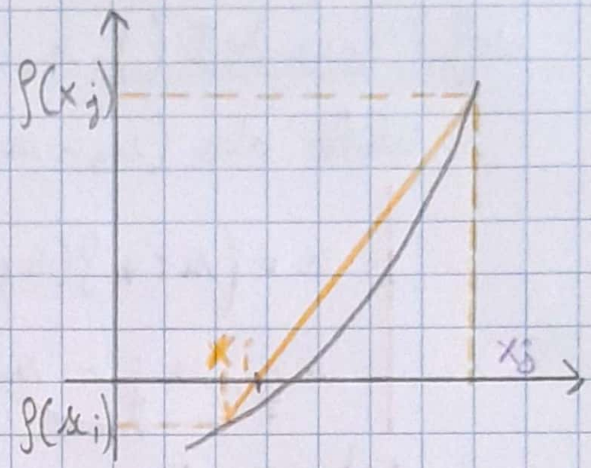
```
def newtonI (f, x0, h = 0.001 , eps = 10 ** (-10)):
    while abs(f(x0)) > eps :
        x = x0 - (f(x0) * 2 * h) /(f(x0 + h) - f(x0 - h))
        x0 = x
    return x0
```

Methode Reccursive :

```
def newtonR (f, x0, h = 0.001, eps = 10 ** (-10)):
    if abs (f(x0)) > eps :
        x = x0 - (f(x0) * 2 * h) /(f(x0 + h) - f(x0 - h))
        return newtonR (f, x)
    else :
        return x0
```

• Méthode de la secante





si x_i et x_j sont proches
⇒ comme meth de
newton

si x_i et x_j sont loins
⇒ comme meth de
lagrange

→ On trace la droite qui passe par les deux points x_i et x_j

→ L'intersection de cette droite avec l'axe des X est le point x qui devient nouveau x_i et le point de l'ancienne x_i devient le nouveau x_j.

$$x_i, x_j \rightarrow x = \frac{x_i f(x_j) - x_j f(x_i)}{f(x_j) - f(x_i)}$$

$$(x_i, x_j) = (x, x_i)$$

$$\text{tq } |x_i - x_j| > \varepsilon$$

Methode Iterative :

```
def secanteI(f, xi, xj, eps=10**(-10)):
    while abs(xj - xi) > eps :
        x = (xi * f(xj) - xj * f(xi)) / (f(xj) - f(xi))
        xi, xj = x , xi
    return xi
```

devient

Methode Reccursive :

```
def secanteR(f, xi, xj, eps = 10**(-10)):
    if abs(xj - xi) > eps
        x = (xi * f(xj) - xj * f(xi))/(f(xj) - f(xi))
        return secanteR(f, x, xi)
    else :
        return xi
```

Seance 2

Objectif

surface

Chercher $S \approx ?$ /

$$S = \int_a^b f(x)\,dx$$

Calcul d'integrale $\Leftrightarrow$ fct primitive

. Methode predefini

1. quad

quad(f, a, b)

resultat (integral, precision)

2. trapz

x = linspace (a, b, n)

y = f(x)

trapz (y, x)

3. simps

simps (y, x)

4. integrate

var("x")

integrate (fct(x))

ou

integrate(fct, (x, a, b))

Methodes à definir

1. Methode des rectangles

$a, b, n \longrightarrow \boxed{h = \frac{b-a}{n}}$

$\boxed{x_i = a + ih}$

$$S = \int_a^b f(x)\,dx = \sum_{i=0}^{n-1} S_i$$

$$\boxed{S = h \sum f\left(x_i + \frac{h}{2}\right)}$$

. Si on minore le S :

$\Leftrightarrow h \to 0$

$\Rightarrow \boxed{S = h f(x_i)}$

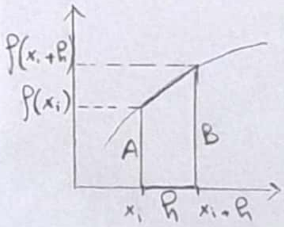
. Si on majore le S :

$\Rightarrow \boxed{S = h f(x_i + h)}$

. Si on prend la moitié

$\Rightarrow \boxed{S = h f\left(x_i + \frac{h}{2}\right)}$

2. Methode des Trapezes



$$S = \frac{h}{2}(A + B)$$

$$\boxed{S = \frac{h}{2} \sum_i \left(f(x_i) + f(x_i + h)\right)}$$

3. Methode des Simpsons

$$S = \frac{h}{6} \sum \left(f(x_i) + 4 f\left(x_i + \frac{h}{2}\right) + f(x_i + h)\right)$$