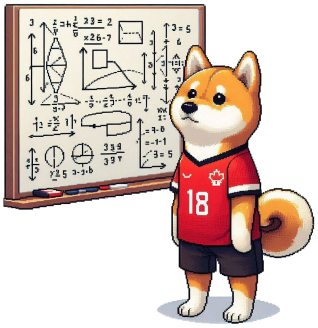
Mathématiques PT. 2025 - 2026

Mathématiques & Informatique - F. Gaunard
http://frederic.gaunard.com
Lycée Voltaire, Paris 11e.



Préparation à l'oral

Math-Info : Math 2
Semaine du 8 Juin

Sujet OB-MATH2-1

Exercice de mathématiques

Soit $f(x, y) = (x - y^2)(y - x^2)$.

1. Tracer la courbe des points où f s'annule et déterminer les zones où f prend des valeurs strictement positives et strictement négatives.
2. Déterminer les points critiques de f appartenant à la droite $y = x$ et déterminer leur nature.
3. Existe-t-il d'autres points critiques?

Exercice d'informatique

1. Écrire une fonction `parcours` d'argument une liste $L = [a_0, a_1, \dots, a_{n-1}]$ qui modifie cette liste en permutant a_k et a_{k+1} si $a_{k+1} < a_k$, pour k variant de 0 à $n-2$, et qui renvoie le nombre de permutations effectuées.
   Par exemple, si `L = [5, 4, 1, 6]`, alors `parcours(L)` renvoie le nombre 2 et la liste `L` devient `[4, 1, 5, 6]`.

2. Que peut-on dire du dernier élément d'une liste à laquelle on a déjà appliqué une fois la fonction `parcours` ?

3. On veut utiliser cette fonction pour trier par ordre croissant une liste selon le principe suivant : pour une liste L donnée en entrée on répète l'application de la fonction parcours jusqu'à ce que `L` devienne invariante par cette fonction.
   Écrire une fonction `tri` qui ordonne une liste donnée en entrée selon ce principe et qui renvoie le nombre d'itérations effectuées.

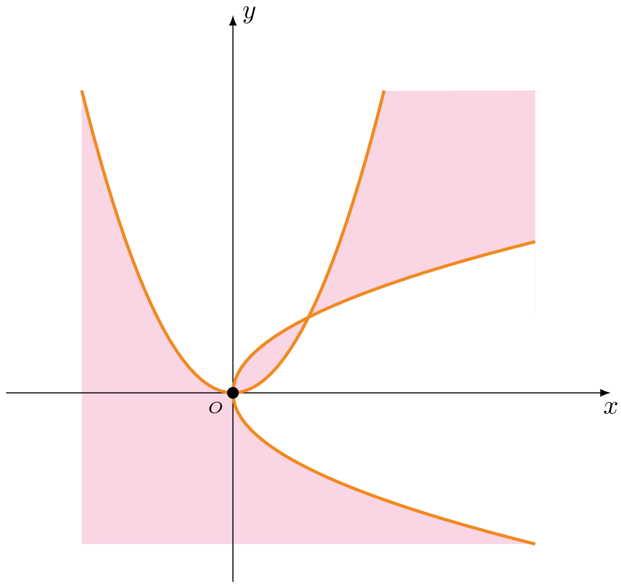
4. Expliquer pourquoi la fonction `tri` donne bien le résultat voulu en un temps fini.
   Évaluer son coût de calcul dans le meilleur des cas et dans le pire des cas.

5. Écrire une fonction `tri1` comme une amélioration de la fonction `tri` en évitant les comparaisons inutiles.

Solution du sujet OB-MATH2-1

Math (Math II 2018)

1. $f(x,y) = 0 \iff (x-y^2)(y-x^2) = 0 \iff x = y^2$ ou $y = x^2$.
L'ensemble recherchée est donc la réunion de deux paraboles, qui se trace aisément. On complète ensuite le dessin avec un peu de réflexion : $y < x^2$ correspond à l'intérieur de la parabole, et $y > x^2$ à son extérieur. Donc les points extérieurs aux deux paraboles (ou intérieurs aux deux) correspondent à $f(x,y) > 0$, les autres points à $f(x,y) < 0$.



2. On a $\dfrac{\partial f}{\partial x} = y - 3x^2 + 2xy^2$ et $\dfrac{\partial f}{\partial y} = x - 3y^2 + 2yx^2$, donc si le point (x_0, x_0) est un point critique, on a deux fois la même équation $2x_0^3 - 3x_0^2 + x_0 = 0$ qui s'écrit $x_0(x_0-1)(2x_0-1) = 0$, on a donc ici trois points critiques $(0,0)$, $(1,1)$ et $(\frac{1}{2}, \frac{1}{2})$.

La hessienne de f en (x,y) est : $H_f(x,y) = \begin{pmatrix} -6x+2y^2 & 1+4xy \\ 1+4xy & -6y+2x^2 \end{pmatrix}$.

Donc $H_f(1,1) = \begin{pmatrix} -4 & 5 \\ 5 & -4 \end{pmatrix}$ dont le déterminant vaut -9 donc les valeurs propres sont de signes opposés et donc c'est un point col.

$H_f(\frac{1}{2}, \frac{1}{2}) = \begin{pmatrix} \dfrac{-5}{2} & 2 \\ 2 & \dfrac{-5}{2} \end{pmatrix}$ dont le déterminant vaut $\dfrac{9}{4}$ donc les valeurs propres sont de même signe et donc c'est un extremum, de plus la trace est négative donc les deux valeurs propres sont négatives, c'est donc un maximum local.

$H_f(0,0) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ dont le déterminant vaut -1 donc les valeurs propres sont de signes opposés et donc c'est un point col.

3. Soit (x_0, y_0) un point stationnaire, alors $\dfrac{\partial f}{\partial x}(x_0, y_0) = 0 = \dfrac{\partial f}{\partial y}(x_0, y_0)$.

Donc $x_0 \dfrac{\partial f}{\partial x}(x_0, y_0) - y_0 \dfrac{\partial f}{\partial y}(x_0, y_0) = 0$ ce qui donne $3x_0^3 - 3y_0^3 = 0 \iff x_0^3 = y_0^3 \iff x_0 = y_0$.
Il n'y a donc pas d'autre point stationnaire que ceux trouvés précédemment.

Info (Math II 2021)

1.
```python
def parcours(L):
    n = len(L)
    nb = 0
    for k in range(n-1):
        if L[k] > L[k+1]:
            L[k], L[k+1] = L[k+1], L[k]
            nb += 1
    return nb
```

2. À la fin d'un parcours, le dernier élément est un élément maximal de L.

3.
```
def tri1(L):
    nb = 1 # Nombre d'itérations égal à un, car il en faut
                # au moins une pour l'initialisation
    while parcours(L) != 0:
        nb += 1
    return nb
```

4. Le nombre de permutations décroit strictement à chaque itération; en effet lors d'une itération un élément maximal est placé en fin de liste et ne subit aucune permutation lors des itérations suivantes.
Si la liste est triée, aucune permutation n'est effectuée et la liste est parcourue une seule fois, ce qui donne une complexité linéaire. Si elle triée dans le sens décroissant, alors : il faut $n-1$ permutations, puis $n-2$ permutation, etc... et la liste est parcourue n^2 fois, ce qui conduit à une complexité quadratique.

5.
```
def tri2(L):
    n = len(L)
    while parcours(L) != 0 and n > 0:
        parcours(L[0:n])
        n -= 1
```

`tri2` améliore `tri1`, puisque à chaque étape elle opère sur une liste de longueur strictement plus petite.