```
import autograd as ag
import autograd.numpy as np
import matplotlib.pyplot as plt
```

1 Optimisation sous contrainte

L'objectif

$$f(a,b) \quad = \quad a^3 + 2\,a^2 - 2\,a\,b + b^2 + a\,b^3 - 2\,b + 5$$

admet cinq points stationnaires dont un minimum local de coordonnées

$$(a^*, b^*) \quad \simeq \quad (.2255, .9312)\,. \tag{1}$$

On cherche à minimiser $f(a,b)$ sous la contrainte

$$a^2 + b^2 \quad \leq \quad \frac{1}{2}. \tag{2}$$

1.1 Résolution graphique

Le fichier `graphe.py` permet de visualiser ces informations.

Question 1. Le minimum local (1) satisfait-il la contrainte ? À l'optimum, la contrainte sera-t-elle active ? Si oui, transformer la contrainte (2) en une contrainte d'égalité (3)

$$c(a,b) \quad = \quad 0\,. \tag{3}$$

Question 2. Déterminer graphiquement les points (a,b) qui satisfont la contrainte et où le gradient de la contrainte (3) et celui de l'objectif sont parallèles.

Question 3. Vérifier en écrivant une fonction `nabla_c` pour la contrainte et en affichant quelques vecteurs gradients (en bleu pour l'objectif et en rouge pour la contrainte).

1.2 Calcul du minimum

Question 4. Donner un Lagrangien du problème : minimiser $f(a,b)$ sous la contrainte (3). Quelles sont les variables du Lagrangien ?

Question 5. Écrire une fonction `Lagrangien`, paramétrée par un tableau `u` contenant les variables du Lagrangien (pour pouvoir utiliser `autograd`). Votre fonction pourrait commencer ainsi :

```
def Lagrangien (u) :
    a = u[0]
    ...
```

Question 6. Affecter à deux variables `nabla_Lagrangien` et `H_Lagrangien` des fonctions calculant le gradient et la hessienne du Lagrangien, en utilisant `autograd`.

Question 7. Calculer la solution optimale du problème d'optimisation sous contrainte par une méthode de Newton sans contrôle du pas, appliquée au Lagrangien.

Question 8. Vérifier que la méthode de Newton avec contrôle du pas (tel qu'on l'a vu en cours) ainsi que l'algorithme du gradient ne fontionnent pas (ou pas toujours) lorsqu'on les applique à un Lagrangien. Pourquoi ? Aide : le Lagrangien peut-il prendre des valeurs négatives ? arbitrairement grandes en valeur absolue ?

2 Estimation de paramètres pour la fonction logistique

L'importance de la fonction logistique (4) en dynamique de populations a été mise en évidence pour la première fois par Pierre-François Verhulst en 1838 [2].

$$y(x) \quad = \quad \frac{\kappa}{1+e^{\alpha-\rho\,x}}. \tag{4}$$

En dynamique de populations, la variable indépendante x représenterait le temps, la variable dépendante $y(x)$ la population au temps x et les trois lettres grecques κ, α, ρ des paramètres. La courbe de la fonction logistique a la forme d'une sigmoïde ayant pour asymptotes horizontales les droites $y = 0$ et $y = \kappa$.

On considère les données suivantes, adaptées de [1, Fig. 7.2, page 229]. À une constante additive près, il s'agit de $m = 10$ mesures de la quantité d'eau présente dans des ufs de *Locustana pardalina* en formation, à une température de 35 degrés.

nb. jours	x_i	0	1	2	3	4	5	6	7	8	9
qté eau	y_i	.53	.53	1.53	2.53	12.53	21.53	24.53	28.53	28.53	30.53

(5)

2.1 Estimation par moindres carrés linéaires

Introduisons la fonction logit définie par

$$\text{logit}(p) \quad = \quad \ln\left(\frac{p}{1-p}\right).$$

À partir de (4), on trouve

$$\frac{\kappa}{y} \quad = \quad 1+e^{\alpha-\rho\,x} \quad \text{et donc} \quad \text{logit}\left(\frac{y}{\kappa}\right) \quad = \quad \rho\,x-\alpha\,.$$

Supposons κ connu. Il suffit alors d'appliquer la transformation ci-dessus (logit) sur les ordonnées y_i des points expérimentaux puis d'estimer les deux paramètres α et ρ par la méthode des moindres carrés linéaires. Comment déterminer κ ? Visuellement, il est souvent facile d'estimer l'asymptote horizontale (d'équation $y = \kappa$) vers laquelle tend la sigmoïde. C'est la méthode préconisée dans les ouvrages anciens [1, page 150]. On remarque que l'emploi de cette méthode implique de choisir $\kappa > y_i$ pour $1 \le i \le m$ parce que logit(p) n'est définie que pour $0 < p < 1$. En prenant $\kappa = 30.54$ on trouve :

$$\alpha,\, \rho \quad = \quad 5.163,\, 1.188\,.$$

2.2 Estimation par moindres carrés non linéaires

L'estimation obtenue par la méthode de la section 2.1 peut maintenant servir de point initial pour une méthode de calcul de minimum local. Voir figure 1.

Question 9. Affiner l'estimation obtenue par la méthode de Newton sans et avec contrôle du pas, et par l'algorithme du gradient.

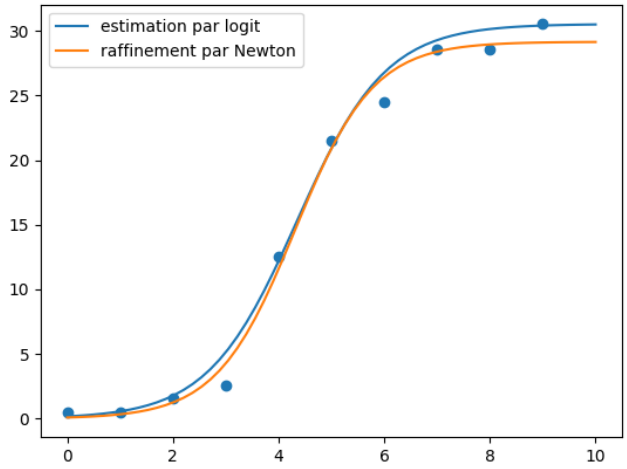

FIGURE 1 – Une première estimation de κ a permis d'estimer les valeurs de ρ et α par moindres carrés linéaires. Cette première estimation a servi de point de départ à un schéma de Newton et/ou un algorithme du gradient pour obtenir la solution optimale.

3 Placement de cercles

Cet exercice-ci montre la puissance d'un logiciel comme `autograd` (voir figure 2). Un cercle est défini par trois nombres : son rayon r et les coordonnées (x_c, y_c) de son centre. On suppose que ces nombres sont stockés dans deux tableaux `u` et `v`. Voici quelques solutions possibles :

$$\mathtt{u}, \mathtt{v} \quad = \quad [r, x_c, y_c], [] \tag{6}$$
$$\mathtt{u}, \mathtt{v} \quad = \quad [x_c, y_c], [r] \tag{7}$$
$$\mathtt{u}, \mathtt{v} \quad = \quad [r], [x_c, y_c] \tag{8}$$

On suppose donnée une fonction `decompose`, paramétrée par `u` et `v` et qui retourne le triplet r, x_c, y_c. La fonction suivante correspond au choix (6) :

```
def decompose (u, v) :
    return u[0], u[1], u[2]
```

Question 10. Écrire une fonction `trace_cercle` paramétrée par `u` et `v` et qui trace le cercle (utiliser `decompose` pour récupérer les valeurs des paramètres, faire varier un angle θ entre $-\pi$ et π, etc.).

Le problème. Dans les fonctions qui suivent, on suppose donnés deux tableaux `Tx` et `Ty` contenant les abscisses et les ordonnées de points expérimentaux. Par exemple,

```
Tx = np.array ([11,12,8,7,3], dtype=np.float64)
Ty = np.array ([4,8,13,12,7], dtype=np.float64)
```

On cherche à placer un cercle au plus près des points. On suppose qu'on dispose d'un premier cercle, pas trop éloigné des points. Par exemple

$$r, x_c, y_c \quad = \quad 4, 8, 7\,.$$

Question 11. Écrire un objectif `f` paramétré par `u`, `v`, `Tx` et `Ty` et qui retourne la somme des carrés des distances entre le cercle et chacun des points expérimentaux (utiliser `decompose` ainsi que `np.arctan2` pour calculer l'angle dans la direction duquel le point courant se trouve par rapport au centre du cercle, etc.)

Question 12. Affecter à des variables `nabla_f` et `H_f` des fonctions retournant le gradient et la hessienne de `f`. Notes :

- les fonctions `nabla_f` et `H_f` reçoivent exactement les mêmes paramètres que `f` ;
- les fonctions `nabla_f` et `H_f` calculent les dérivées partielles de `f` par rapport à son premier paramètre (ici `u`) uniquement.

Question 13. Écrire une fonction `backtrack_line_search` paramétrée par `u`, `v`, `Tx`, `Ty`, un flottant `alpha0` $\in]0, 1]$ un vecteur `h` et le vecteur gradient `g` et qui retourne une valeur α qui permette de contrôler le pas de l'algorithme de Newton et/ou celui du gradient.

Question 14. Déterminer un cercle qui passe au plus près des points en ajustant les valeurs des trois paramètres r, x_c et y_c. Faire le calcul par l'algorithme de Newton et par l'algorithme du gradient.

Question 15. Même question mais seul le rayon r peut être ajusté. Que suffit-il de changer ?

Question 16. Même question mais seules les coordonnées du centre peuvent être ajustées. Que suffit-il de changer ?

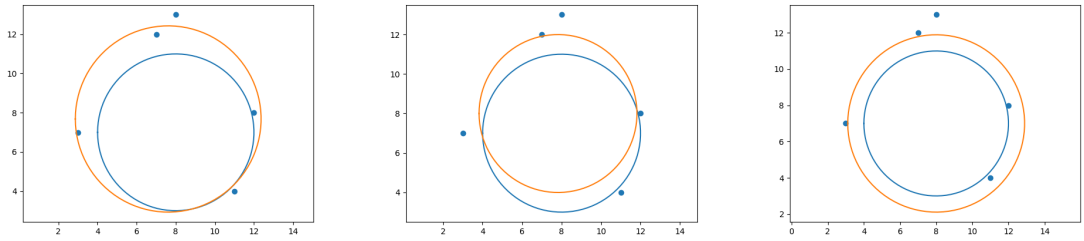

FIGURE 2 – Le cercle initial en bleu et le cercle final en orange. Les trois figures ont été obtenues avec le même code Python. Seules les variables du problème d'optimisation changent : les coordonnées du centre et le rayon (à gauche), les coordonnées du centre (au milieu) et le rayon (à droite).

Références

[1] L. C. Birch and H. G. Andrewartha. *The Distribution and Abundance of Animals*. The University of Chicago Press, 1954.

[2] Pierre-François Verhulst. Notice sur la loi que la population suit dans son accroissement. *Correspondance mathématique et physique*, 10 :113–121, 1838.