Contrôle continu 2: optimisation numérique

Exercice 1.

Soit J une fonction strictement convexe et coercive. Soit $m \in \mathbb{N}^*$ et $\phi_i : \mathbb{R}^n \to \mathbb{R}$ des fonctions convexes pour tout $i \in \{1, ...m\}$. On note

$$C = \{x \in \mathbb{R}^n, \quad \phi_i(x) \leq 0, \, 1 \leq i \leq m\}$$

On suppose que C est borné et on note son intérieur :

$$D = \{x \in \mathbb{R}^n, \quad \phi_i(x) < 0, \, 1 \leq i \leq m\}$$

Soit $\epsilon > 0$. Pour tout $x \in D$, on définit la fonction pénalisée :

$$J_\epsilon(x) = J(x) - \epsilon \sum_{i=1}^{m} \frac{1}{\phi_i(x)}$$

1. Montrer que J possède sur C un unique minimum noté x^*
2. Montrer que J_ϵ possède sur D un unique minimum noté x_ϵ.
3. Montrer, quitte à extraire, que x_ϵ tend vers x^* lorsque ϵ tend vers 0.
4. Montrer que si $0 < \epsilon' < \epsilon$, alors $J(x^*) \leq J(x_{\epsilon'}) \leq J(x_\epsilon)$

Exercice 2.

Soit A une matrice de taille m$\times n$ avec $m < n$, de rang maximal m et $b \in \mathbb{R}^m$. Soit $x_0 \in \mathbb{R}^n$. On note :

$$D = \{x \in \mathbb{R}^n, \quad Ax = b\}$$

et on cherche à calculer la valeur $d(x_0, D)$, égale à la distance de x_0 à D.

1. Exprimer le problème précédent sous la forme d'un problème d'optimisation quadratique avec m contraintes égalités.
2. Montrer que le problème précédent possède une unique solution x^* et que $d(x_0, D) = ||x_0 - x^*||$
3. Montrer que le multiplicateur de Lagrange λ^* assoocié à cet optimum est égal à

   $$\lambda^* = -(A * A^T)^{-1}(b - Ax_0)$$

   et que

   $$x^* = x_0 + A^T(A * A^T)^{-1}(b - Ax_0)$$

4. On suppose que $m = 1$ (A est alors une matrice ligne). Montrer que dans ce cas :

   $$d(x_0, D) = \frac{||b - Ax_0||}{||A||}$$

5. Ecrire un algorithme de type Uzawa prenant en entrée x_0, A et b et calculant une valeur approchée de $d(x_0, D)$.

Exercice 3. Soit le code Python suivant consistant à résoudre le problème de la canette (ou un problème équivalent où on a considéré que $\pi = 1$) :

```
import numpy as np
def J(v):
    x = v[0]
    y = v[1]
    p=x**2*y-2*V0
    return x**2+x*y+rho*p**2
def GradJ(v):
    x = v[0]
    y = v[1]
    p=x**2*y-2*V0
    gradJ = np.array([2*x+y+2*rho*(2*x*y)*p, x+2*rho*(x**2)*p])
    return gradJ
def canette(J, GradJ, beta, alpha_init, tau, X0, N):
    x_k = X0
    for k in range(N):
        d_k = -GradJ(x_k)
        alpha = alpha_init
        cd=np.dot(d_k, GradJ(x_k))
        while (not(J(x_k + alpha*d_k) <= J(x_k) + alpha*beta*cd)):
            alpha *= tau
            x_k = x_k+alpha*d_k
    return x_k
n=2
rho=0.05
V0=1000
x0 = np.array([8,20])
N = 30000
beta = 0.1
alpha_i = 0.01
tau = 0.3
x=canette(J, GradJ, beta, alpha_i, tau, x0, N)
```

1. Expliquer l'objectif de la fonction `canette` et quelle méthode est utilisée pour résoudre ce problème.
2. Que représentent la variable `rho` et la variable de sortie `x` ?
3. Retrouver le problème mathématique initial de minimisation sous contrainte à résoudre et montrer que sa solution exacte pour la valeur de V0 proposée est égale à $x = (10, 20)$.
4. Proposer une alternative pour résoudre ce problème avec la méthode de Newton et écrire un script `canette2`.