CC3: optimisation numérique

Exercice 1

On considère la méthode pattern search appliquée avec un ensemble de directions D, engendrant positivement $\mathbb{R}^n$, pour une fonction f supposée C^1 et de gradient Lipschitzien de rapport ν.

1. Montrer que la quantité :
$$cm(D) = \min_{||v||=1} \max_{d\in D} < v, \frac{d}{||d||} >$$
est bien définie et strictement positive.
2. . Soit $x \in \mathbb{R}^n$ et $\alpha > 0$ tel que pour toute direction $d \in D$, on a $f(x) \leq f(x+\alpha d)$ (cas d'échec). Montrer que
$$||\nabla f(x)|| \leq \frac{\nu}{2} cm(D)^{-1} (\max_{d\in D} ||d||)\alpha$$
On pourra commencer par écrire, en le justifiant, que
$$f(x+\alpha d) - f(x) = \int_0^1 < \nabla f(x+t\alpha d), \alpha d > dt$$

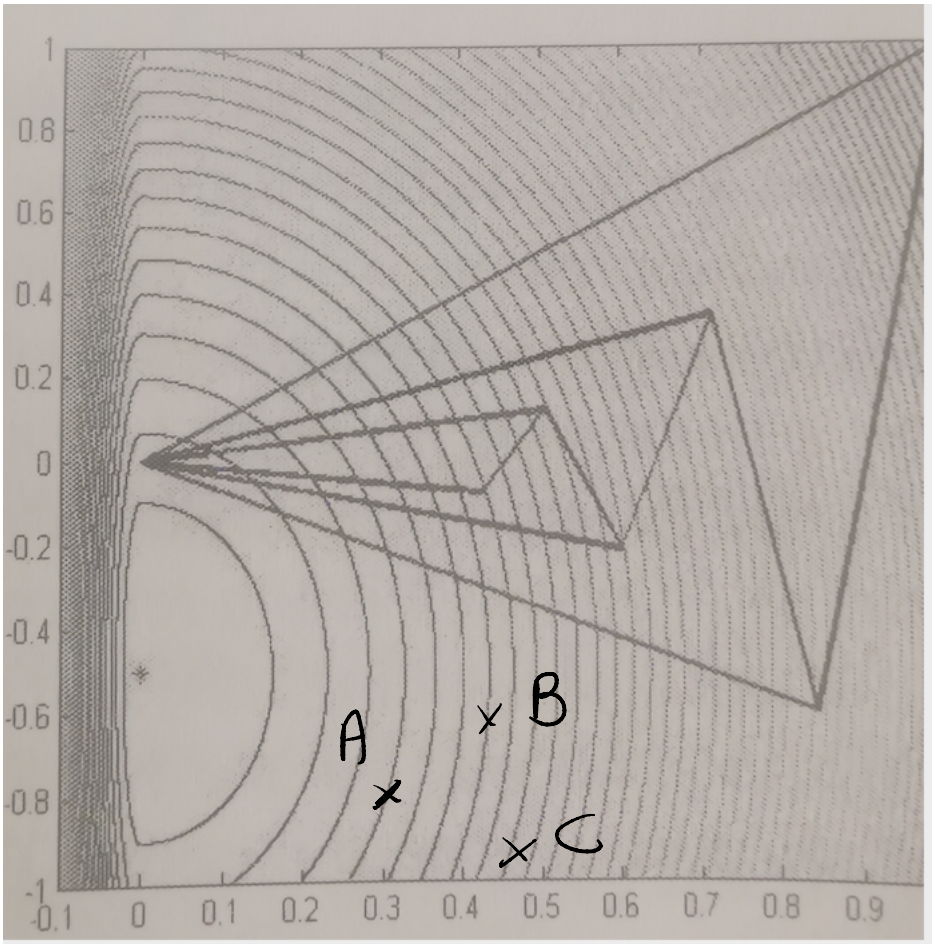
Exercice 2

On considère la fonction $f : \mathbb{R}^2 \to \mathbb{R}$ telle que
$$f(x) = \frac{1}{2} \max(||x-c_1||^2, ||x-c_2||^2)$$
avec $c_1 = (1,-1)$ et $c_2 = -c_1$.

1. Représenter les lignes de niveau de la fonction f sur le carré $[-1,1]^2$.
2. La fonction f est-elle C^0 ? C^1 ? C^2 ?
3. On considère la méthode pattern search appliquée à la fonction f avec les directions
$$D = \{(0,1),(0,-1),(1,0),(-1,0)\}$$
et le point initial $x_0 = (-0.5,-0.5)$. Que se passe t-il dans ce cas pour la méthode au cours des itérations ? On tracera avec précision les lignes de niveau autour de x_0 pour justifier la réponse.

Exercice 3

On a représenté sur la figure ci-dessous les premières itérations de la méthode de Nelder Mead pour une fonction f dont les lignes de niveau sont également représentées.



1. Où se situe le minimum de la fonction représenté ?
2. Décrire ce qu'il s'est passé lors des 4 premières itérations représentées sur la figure.
3. On reprend la méthode avec le simplexe de départ $\{A, B, C\}$. Tracer le résultat de la première itération.

Exercice 4 On considère le script Python suivant pour la méthode pattern Search :

```
def Pattern_search(f,X0,Niter):
    n=len(X0)
    alpha=1
    rho=0.5
    tau=1.5
    D=np.concatenate([np.eye(n),-np.eye(n)])  # set of directions
    N=2*n
    Xk=X0
    for k in range(Niter):
        i=0
```

```
        while(i<=N-1 and ----------------:
            i=i+1
        if (i==N):#echec
            alpha=alpha* -----------------
        else:
            Xk=------------------------------
            alpha=alpha*-----------------
    return Xk
```

Quatre morceaux de lignes ont malencontreusement été effacés et remplacés par des$---$. Reconstituer ces 4 lignes.

Exercice 5 Le script suivant a été écrit pour résoudre le problème de la canette (en remplaçant π par 1) :

```
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 F(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 = 30000
beta = 0.1
alpha_i = 0.1
tau = 0.3
rho=0.05
V0=1000
X0 = np.array([12,22])
print(F(J, GradJ, beta, alpha_i, tau, X0, N))
```

et affiche le résultat :

```
[  9.97606867  20.08601486]
```

1. Expliquer la méthode utilisée par ce script pour résoudre le problème.
2. Le résultat obtenu est-il satisfaisant ? On pourra calculer le résultat exact.