Université de Versailles — Année 2024-2025
`http://dumas.perso.math.cnrs.fr/M1-optim.html` — M1, semestre 1

TD 6: algorithmes d'optimisation sans gradient (partie 2)

Exercice 1

Dans une stratégie d'évolution pour minimiser une fonction f définie sur $\mathbb{R}^n$, on choisit d'utiliser la mutation gaussienne $y = x + h$ où la densité de la variable alétoire $h \in \mathbb{R}^n$ est donnée par :

$$p(h) = \frac{1}{(2\pi)^{\frac{n}{2}} \det(C)^{\frac{1}{2}}} \exp(-\frac{1}{2}{}^t hC^{-1}h)$$

1. Calculer la moyenne de la variable aléatoire h. Dans le cas isotrope ($C = \alpha I_n$) et avec $n = 1$ que vaut la variance de la variable aléatoire h ?
2. Représenter approximativement (à l'ordre 2) les lignes de niveaux d'une fonction f supposée C^2 près d'un minimum noté x^* (on pourra prendre un exemple en dimension 2).
3. De manière générale, quel est le meilleur choix pour la matrice C par rapport à $Hf(x)$ (Hessien de f en x) si x se trouve proche de x^* ? Expliquer votre réponse sur le graphique précédent.

Exercice 2

```
I_j = j ∀ j ∈ {1, ..., λ}
for i = 1 to λ do
    for j = 1 to λ − 1 do
        sample u ∈ U(0, 1) (uniform random number generator)
        if (φ(x_{I_j}) = φ(x_{I_{j+1}}) = 0) or (u < P_f) then
            if f(x_{I_j}) > f(x_{I_{j+1}}) then
                swap(I_j, I_{j+1})
            fi
        else
            if φ(x_{I_j}) > φ(x_{I_{j+1}}) then
                swap(I_j, I_{j+1})
            fi
        fi
    od
    if no swap done break fi
od
```

Fig. 2. Stochastic ranking procedure, $P_f = 0.45$.

La méthode du classement stochastique décrite ci-dessus permet de classer λ individius dans un problème d'optimisation avec contrainte.

Dans cet algorithme, f est la fonction coût à minimiser et Φ la pénalisation et P_f est un paramètre.

1. Que se passe t-il quand $P_f = 0$, respectivement 1 ?
2. Ecrire une fonction Python de classement alétaoire de λ individus pour une fonction f et une pénalisation Φ données.

Exercice 3 (CC3, 2023) Le script suivant a été écrit pour résoudre un problème de minimisation :

```
n = 2; lambda = 50, mu = 5; Ngen = 200; sigma = 0.1
def f(x):
    return 100 * (x[1] - x[0] ** 2) ** 2 + np.sum((x[:-1] - 1) ** 2)
Ap = np.random.rand(mu, n + 1)
for i in range(mu):    Ap[i, n] = f(Ap[i, :n])
Ae = np.zeros((lmbda, n + 1))
BestX = np.zeros((Ngen, n))
for k in range(Ngen):
    tau = 0;  X = np.mean(Ap[:, :n], axis=0); val = f(X)
    for i in range(lambda):
        Ae[i, :n] = X + sigma * np.random.normal(0, 1, size=n)
        Ae[i, n] = --------------------------
        if Ae[i, n] < val:
            tau += 1 / lambda
    sorted_indices = np.argsort(Ae[:, n])
    Ae = Ae[sorted_indices, :]; Ap = Ae[:mu, :];BestX[k, :] = Ap[0, :n]
    sigma = np.exp(1 / 3 * (tau - 1 / 5) / (1 - 1 / 5)) * sigma
print(BestX[-1, :])
```

et affiche le résultat : `[0.98110579 0.96252734]`.

1. Expliquer la méthode utilisée par ce script pour résoudre le problème de minimisation.
2. Que représente les matrices `Ap` et `Ae` ? Décrire le détail d'une ligne de ces matrices.
3. Un morceau de ligne a malencontreusement été effacé et remplacé par des$- - - - - - -$. Reconstituer et expliquer cette ligne.
4. Que représente la variable `sigma` et comment va-t-elle évoluer au cours des itérations ?
5. Que vaut `a` si on tape dans Python : `a = np.argsort(np.array([3.3,4,2,6,1]))` ?
6. Proposer et écrire une autre méthode de sélection.