4.2. TP 2 : Simulation de Vecteurs Gaussiens

Dans ce TP, nous allons explorer les méthodes pour simuler des vecteurs gaussiens en dimension 2. Voici les questions et leurs solutions :

4.2.1 Question 1 : Simulation d'un vecteur gaussien en dimension 2

Simulez un vecteur gaussien $Z = (Z_0, Z_1)$

```python
import numpy as np
import matplotlib.pyplot as plt

# Generer des variables uniformes
u1 = np.random.uniform(0, 1, 1000)
u2 = np.random.uniform(0, 1, 1000)

# Transformation de Box-Muller pour obtenir deux variables
    normales
z0 = np.sqrt(-2 * np.log(u1)) * np.cos(2 * np.pi * u2)
z1 = np.sqrt(-2 * np.log(u1)) * np.sin(2 * np.pi * u2)

# Affichage des points dans le plan
plt.scatter(z0, z1, alpha=0.6)
plt.title('Exemple d\'un vecteur Gaussien ')
plt.xlabel('z0')
plt.ylabel('z1')
plt.grid(True)
plt.show()
```

4.2.2 Question 2 : Décomposition de Cholesky

Effectuez la décomposition de Cholesky pour une matrice symétrique positive d'ordre 2.

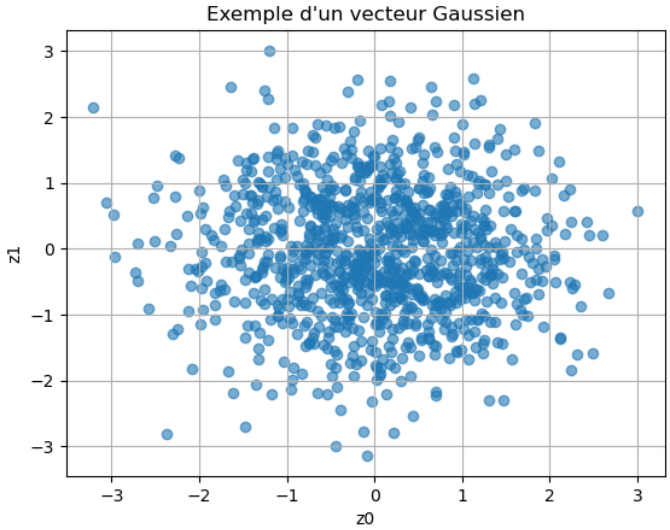

FIGURE 4.7 – Exemple de nuage de point pour un vecteur gaussien standard

```python
# Matrice de covariance
Sigma = np.array([[2, 1], [1, 2]])

# Decomposition de Cholesky
L = np.linalg.cholesky(Sigma)
print(L)
```

Donc la décomposition de Cholesky de la matrice

$$\Sigma = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

est

$$L = \begin{pmatrix} \sqrt{2} & 0 \\ \frac{1}{\sqrt{2}} & \sqrt{\frac{3}{2}} \end{pmatrix}$$

4.2.3 Question 3 : Simulation d'un vecteur gaussien de moyenne nulle et de matrice de covariance Σ

Utilisez les résultats précédents pour simuler un vecteur gaussien de moyenne nulle.

Lemme 4.2-1 *Si X est un vecteur aléatoire suivant une loi gaussienne $\mathcal{N}(0, I)$, et si L est une matrice triangulaire inférieure, alors le vecteur $Y = m + L^T X$ suit une loi gaussienne de moyenne m et de matrice de covariance $\Sigma = LL^T$.*

```
# Generer des vecteurs gaussiens standards
z = np.random.normal(0, 1, (2, 1000))
m=[0,0]
# Transformation par la matrice de Cholesky
x = m+np.dot(L, z)

# Affichage des points
plt.scatter(x[0, :], x[1, :], alpha=0.6)
plt.title('Vecteur Gaussien en dimension 2 (M thode de
    Cholesky)')
plt.xlabel('x0')
plt.ylabel('x1')
plt.grid(True)
plt.show()
```

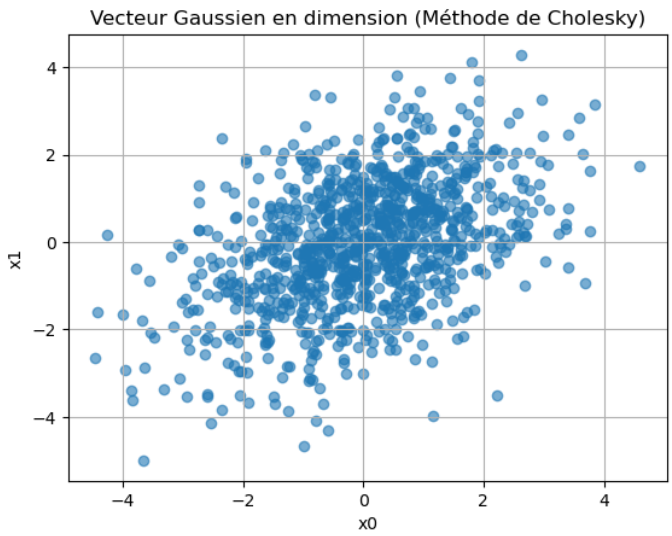

FIGURE 4.8 – Simulation d'un vecteur gaussien en dimension 2 avec la méthode de Cholesky

4.2.4 Question 4 : Simulation d'un vecteur gaussien en dimension 2 avec la méthode directe de Python

Utilisez les fonctions Python pour simuler directement un vecteur gaussien en dimension 2.

```
# Moyenne et matrice de covariance
mu = [0, 0]
Sigma = [[2, 1], [1, 2]]

# Simulation directe
x = np.random.multivariate_normal(mu, Sigma, 1000).T

# Affichage des points
plt.scatter(x[0, :], x[1, :], alpha=0.6)
plt.title('Vecteur Gaussien (M thode Directe)')
plt.xlabel('x0')
plt.ylabel('x1')
plt.grid(True)
plt.show()
```

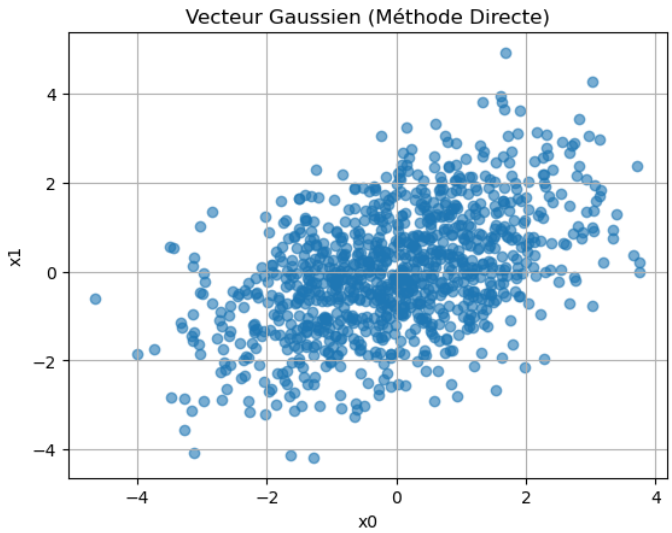

FIGURE 4.9 – Simulation d'un vecteur gaussien avec la méthode directe

4.2.5 Question 5 : Comparaison des méthodes

Comparez les résultats des questions 3 et 4.

Les deux méthodes donnent des résultats très similaires, ce qui valide l'exactitude des simulations.

4.2.6 Question 6 : Étude théorique des vecteurs propres et valeurs propres de la matrice de covariance

Rappel sur les valeurs propres et vecteurs propres de la matrice de covariance d'un vecteur gaussien dans $\mathbb{R}^2$ et leur relation avec les nuages de points.

Les valeurs propres λ_i de la matrice de covariance Σ représentent la variance des données le long des vecteurs propres correspondants v_i. Les vecteurs propres déterminent la direction des axes principaux du nuage de points.

- Si les valeurs propres sont égales, le nuage de points est circulaire. - Si les valeurs propres sont différentes, le nuage de points est ellipsoïdal. - Les vecteurs propres indiquent les directions des axes principaux de l'ellipse.

Nous abordons maintenant les différents cas :

Cas 1 : Nuage de points formant un cercle

```
# Cas 1 : Matrice de covariance avec valeurs propres  gales (cercle)
Sigma_cercle = np.array([[2, 0], [0, 2]])
x_cercle = np.random.multivariate_normal([0, 0], Sigma_cercle, 1000).T

# Calcul des vecteurs propres et valeurs propres
eigvals_cercle, eigvecs_cercle = np.linalg.eig(Sigma_cercle)
print("Valeurs propres (Cercle) : ", eigvals_cercle)
print("Vecteurs propres (Cercle) : ", eigvecs_cercle)

# Affichage des vecteurs propres
origin = np.array([[0, 0], [0, 0]])  # point d'origine
plt.quiver(*origin, eigvecs_cercle[0, :], eigvecs_cercle[1, :], color=['r', 'b'], scale=3)
```

```
plt.scatter(x_cercle[0, :], x_cercle[1, :], alpha=0.6)
plt.title('Nuage de points gaussien - Cercle')
plt.xlabel('x0')
plt.ylabel('x1')
plt.grid(True)
plt.axis('equal')
plt.show()
```

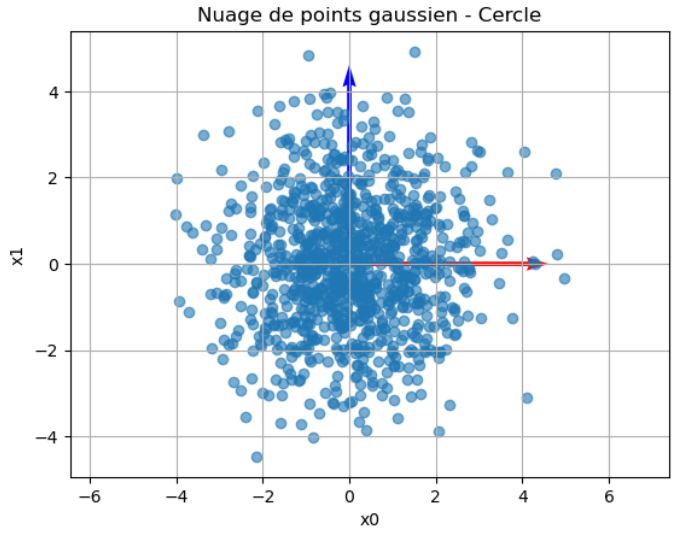

FIGURE 4.10 – Cas 1 : Matrice de covariance avec valeurs propres égales

Interprétation : Dans ce cas, les valeurs propres de la matrice de covariance sont égales. Cela signifie que la variance est la même dans toutes les directions, ce qui donne un nuage de points circulaire. Les vecteurs propres indiquent les axes principaux, mais comme les variances sont égales, ils ne changent pas la forme circulaire du nuage.

Cas 2 : Nuage de points formant une ellipse

```
# Cas 2 : Matrice de covariance avec valeurs propres
    diff rentes (ellipse)
Sigma_ellipse = np.array([[3, 1], [1, 1]])
x_ellipse = np.random.multivariate_normal([0, 0],
    Sigma_ellipse, 1000).T

# Calcul des vecteurs propres et valeurs propres
```

```
eigvals_ellipse, eigvecs_ellipse = np.linalg.eig(Sigma_ellipse)
print("Valeurs propres (Ellipse) : ", eigvals_ellipse)
print("Vecteurs propres (Ellipse) : ", eigvecs_ellipse)

# Affichage des vecteurs propres
origin = np.array([[0, 0], [0, 0]])  # point d'origine
plt.quiver(*origin, eigvecs_ellipse[0, :], eigvecs_ellipse[1,
    :], color=['r', 'b'], scale=3)
plt.scatter(x_ellipse[0, :], x_ellipse[1, :], alpha=0.6)
plt.title('Nuage de points gaussien - Ellipse')
plt.xlabel('x0')
plt.ylabel('x1')
plt.grid(True)
plt.axis('equal')
plt.show()
```

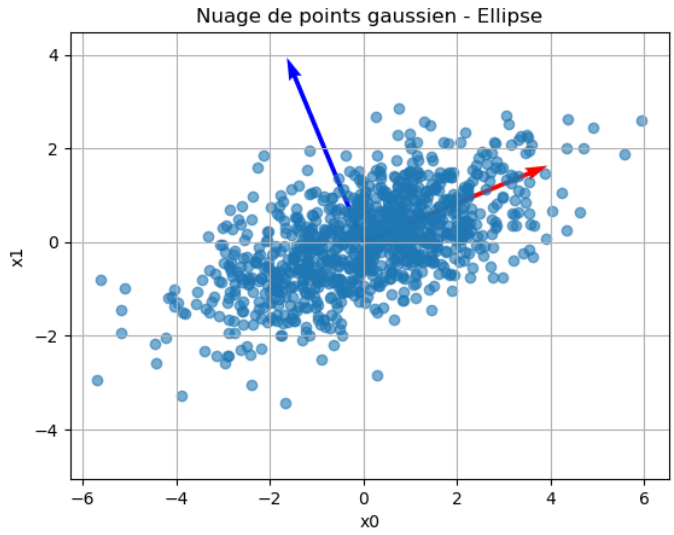

FIGURE 4.11 – Cas 2 : Matrice de covariance avec valeurs propres différentes

Interprétation : Dans ce cas, les valeurs propres sont différentes, ce qui indique des variances différentes le long des axes principaux. Le nuage de points est ellipsoïdal, et les vecteurs propres montrent les directions des axes principaux de l'ellipse.

Cas 3 : Matrice de covariance avec éléments hors-diagonaux négatifs

```
# Cas 3 : Matrice de covariance avec    lments    hors-diagonaux
    n gatifs
Sigma_negatif = np.array([[2.0, -1.5], [-1.5, 2.0]])
x_negatif = np.random.multivariate_normal([0, 0],
    Sigma_negatif, 1000).T

# Calcul des vecteurs propres et valeurs propres
eigvals_negatif, eigvecs_negatif = np.linalg.eig(Sigma_negatif)
print("Valeurs propres (   lments    hors-diagonaux n gatifs) :
    ", eigvals_negatif)
print("Vecteurs propres (   lments    hors-diagonaux n gatifs) :
    ", eigvecs_negatif)

# Affichage des vecteurs propres
origin = np.array([[0, 0], [0, 0]])  # point d'origine
plt.quiver(*origin, eigvecs_negatif[0, :], eigvecs_negatif[1,
    :], color=['r', 'b'], scale=3)
plt.scatter(x_negatif[0, :], x_negatif[1, :], alpha=0.6)
plt.title('Nuage de points gaussien -    lments    hors-diagonaux
    n gatifs')
plt.xlabel('x0')
plt.ylabel('x1')
plt.grid(True)
plt.axis('equal')
plt.show()
```

Interprétation : Dans ce cas, les éléments hors-diagonaux négatifs de la matrice de covariance provoquent une orientation différente de l'ellipse. Les vecteurs propres montrent les directions principales, mais l'ellipse peut sembler "décroissante" en fonction de l'angle formé avec les axes principaux.

Cas 4 : Matrice de covariance singulière

```
# Cas 4 : Matrice de covariance singuli re
Sigma_singuliere = np.array([[2, 2], [2, 2]])
x_singuliere = np.random.multivariate_normal([0, 0],
    Sigma_singuliere, 1000).T
```

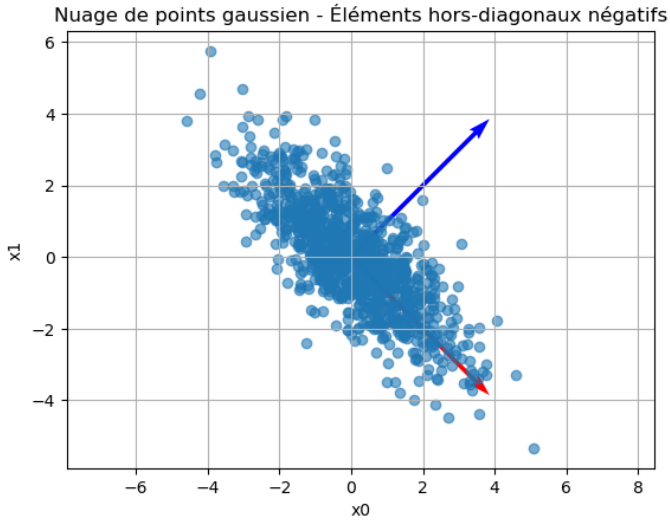

FIGURE 4.12 – Cas 3 : Matrice de covariance avec éléments hors-diagonaux négatifs

```
4
5  # Calcul des vecteurs propres et valeurs propres
6  eigvals_singuliere, eigvecs_singuliere =
       np.linalg.eig(Sigma_singuliere)
7  print("Valeurs propres (Matrice de covariance singuli re) : ",
       eigvals_singuliere)
8  print("Vecteurs propres (Matrice de covariance singuli re) :
       ", eigvecs_singuliere)
9
10 # Affichage des vecteurs propres
11 origin = np.array([[0, 0], [0, 0]])  # point d'origine
12 plt.quiver(*origin, eigvecs_singuliere[0, :],
       eigvecs_singuliere[1, :], color=['r', 'b'], scale=3)
13 plt.scatter(x_singuliere[0, :], x_singuliere[1, :], alpha=0.6)
14 plt.title('Nuage de points gaussien - Matrice de covariance
       singuli re')
15 plt.xlabel('x0')
16 plt.ylabel('x1')
17 plt.grid(True)
18 plt.axis('equal')
19 plt.show()
```

Interprétation : Dans ce cas, la matrice de covariance est singulière, ce qui signifie que l'une des valeurs propres est zéro. Cela provoque un alignement des points le long

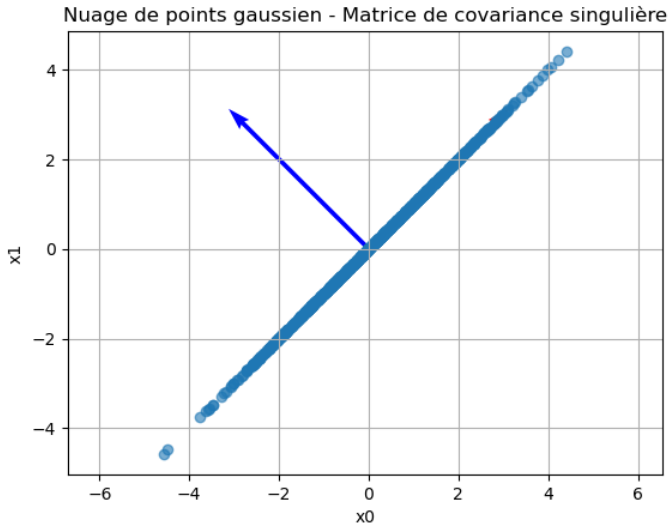

FIGURE 4.13 – Cas 4 : Matrice de covariance singulière

d'une ligne droite, indiquant qu'il n'y a de la variance que dans une seule direction.