Rapport sur Exercice 5 TD 5

Plan de travail :

1- Créez un tableau de données de dimensions (100, 3) où chaque colonne représente une Variable aléatoire et Calculer la matrice de covariance.

2- Appliquez la transformation de Fourier sur un tableau de données sinusoïdales et affichez le spectre de Fréquences.

3- Simulez 1000 lancers de deux dés et affichez l'histogramme des sommes obtenues.

Réaliser par : ABOULOUARD Hamza **Encadrer par : Mr Mounsif**

Filière : IOTR

1- Créez un tableau de données de dimensions (100, 3) où chaque colonne représente une Variable aléatoire et Calculer la matrice de covariance.

Rappel Mathématique sur la Matrice de Covariance pour Trois Variables

La matrice de covariance est une généralisation de la covariance pour plusieurs variables. Pour trois variables aléatoires **X**, **Y**, et **Z**, la matrice de covariance est un tableau 3×33 \times 33×3 qui résume les covariances entre chaque paire de variables.

Définition de la Matrice de Covariance

La matrice de covariance pour trois variables **X**, **Y**, et **Z** est définie comme suit :

$$\text{Cov} = \begin{pmatrix} \text{Cov}(X,X) & \text{Cov}(X,Y) & \text{Cov}(X,Z) \\ \text{Cov}(Y,X) & \text{Cov}(Y,Y) & \text{Cov}(Y,Z) \\ \text{Cov}(Z,X) & \text{Cov}(Z,Y) & \text{Cov}(Z,Z) \end{pmatrix}$$

- Les éléments diagonaux **Cov (X, X), Cov (Y, Y)**, **Cov (Z, Z)** représentent la variance de chaque variable.
- Les éléments non diagonaux **Cov (X, Y)**, **Cov (X, Y)**, **Cov (X, Z)** etc. représentent la covariance entre les variables.
- Calcul de la Covariance :

Pour calculer la covariance entre deux variables **X** et **Y**, vous pouvez utiliser la formule suivante :

$$\text{Cov}(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})$$

Où :

- **N :** est le nombre d'observations.
- **Xi et Yi** sont les valeurs des variables X et Y.
- **X¯\bar{X}X¯ et Y¯\bar{Y}Y¯ sont les moyennes des variables X et Y.**

Code :

exercice 5 td5.py > ...

```python
import numpy as np
donnes = np.random.rand(100, 3)
cov_matrice = np.cov(donnes, rowvar=False)
print("Tableau de données :")
print(donnes)
print("\nMatrice de covariance :")
print(cov_matrice)
```

Explication de code :

- **Importation de la Bibliothèque :**

Import numpy as np : Cette ligne importe la bibliothèque **NumPy**, qui est une bibliothèque essentielle pour le calcul numérique en Python. Elle fournit des fonctionnalités pour travailler avec des tableaux et des matrices.

- **Génération de Données Aléatoires :**

donnes = np. Random. Rand (100, 3) : Cette ligne génère un tableau de dimensions 100×3 rempli de valeurs aléatoires uniformément distribuées entre 0 et 1. Ici, chaque colonne représente une variable, et chaque ligne représente une observation.

- **Calcul de la Matrice de Covariance :**

Cov_matrice = np.cov(data, rowvar=False) : Cette ligne calcule la matrice de covariance pour les données générées. L'argument **rowvar=False** indique que chaque colonne du tableau data représente une variable et chaque ligne une observation. La matrice de covariance résultante montre comment les variables varient ensemble.

- **Affichage des Résultats :**

- **Print ("Tableau de données :")** et **print (donnes) :** Ces lignes affichent le tableau de données générées.
- **Print ("\nMatrice de covariance :")** et **print (cov_matrice) :** Ces lignes affichent la matrice de covariance calculée.

```
Tableau de données :
[[0.72677053 0.76011249 0.73351331]
 [0.22284626 0.09857747 0.5613386 ]
 [0.09013298 0.56577944 0.65681408]
 [0.54569746 0.20302322 0.37030501]
 [0.22299089 0.06571539 0.64339676]
 [0.86920261 0.78703039 0.09971412]
 [0.64530556 0.82850779 0.23013697]
 [0.09616798 0.13458752 0.79424198]
 [0.12244471 0.07041925 0.08779498]
 [0.36316631 0.32875858 0.68347013]
 [0.04031404 0.91443951 0.76406481]
 [0.41463407 0.54309384 0.55318102]
 [0.22705807 0.43061572 0.78520598]
 [0.95644042 0.86614241 0.04210487]
 [0.29718304 0.00359544 0.97761865]
 [0.7149606  0.49715056 0.17490407]
 [0.85359343 0.42722055 0.03178775]
 [0.49308119 0.16634574 0.1474509 ]
 [0.13123323 0.56594392 0.75025482]
 [0.31028623 0.29912084 0.2627502 ]
 [0.19775485 0.22184318 0.60465054]
 [0.41640396 0.34909191 0.47441737]
 [0.43432645 0.0950059  0.44925555]
 [0.18159448 0.93713079 0.91948981]
 [0.16308709 0.52984764 0.93027953]
 [0.2854457  0.01257888 0.70587664]
 [0.27694526 0.04275811 0.41990343]
 [0.23459149 0.56739754 0.2376809 ]
 [0.10495138 0.42117865 0.26321135]
 [0.14217735 0.99610081 0.05343744]
 [0.99587382 0.60682404 0.2535962 ]
 [0.48892716 0.75891459 0.52830451]
 [0.34046567 0.35664549 0.40898157]
 [0.23477785 0.50455459 0.70034378]
 [0.80338516 0.73108136 0.16991681]
 [0.07236548 0.83124559 0.74397474]]

Matrice de covariance :
[[ 0.08297914  0.00054121 -0.01378185]
 [ 0.00054121  0.09221843 -0.00121726]
 [-0.01378185 -0.00121726  0.06904851]]
```

4- Appliquez la transformation de Fourier sur un tableau de données sinusoïdales et affichez le spectre de

Fréquences.

Application de la Transformation de Fourier sur des Données Sinusoïdales

Rappel :

La transformation de Fourier est un outil mathématique puissant qui permet d'analyser les composants fréquentiels d'un signal. Dans cette section, nous appliquerons cette transformation à un tableau de données sinusoïdales afin de visualiser son spectre de fréquences.

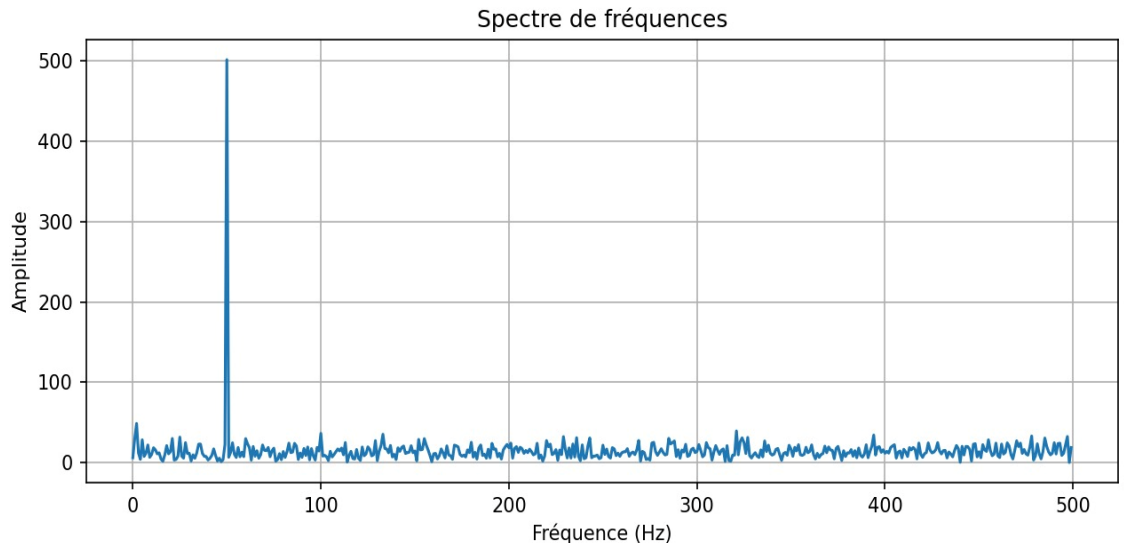
- **Génération du Signal :** Le signal est constitué de deux sinusoïdes de différentes fréquences.
- **Application de la Transformée de Fourier :** Nous utilisons la fonction de transformation de Fourier pour obtenir le spectre de fréquences.
- **Visualisation du Spectre :** Le spectre est affiché pour analyser les amplitudes des différentes fréquences présentes dans le signal.

Code

Voici le code utilisé pour réaliser cette analyse :

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

# Paramètres de la sinusoïde
fs = 1000  # Fréquence d'échantillonnage en Hz
t = np.arange(0, 1, 1/fs)  # Temps de 0 à 1 seconde
f1 = 50  # Fréquence de la première sinusoïde en Hz
f2 = 120  # Fréquence de la deuxième sinusoïde en Hz

# Génération des données sinusoïdales
signal = 0.5 * np.sin(2 * np.pi * f1 * t) + 0.5 * np.sin(2 * np.pi * f2 * t)

# Application de la Transformée de Fourier
freqs = np.fft.fftfreq(len(signal), 1/fs)
fft_values = np.fft.fft(signal)

# Affichage du spectre de fréquences
plt.figure(figsize=(12, 6))
plt.plot(freqs[:len(freqs)//2], np.abs(fft_values)[:len(fft_values)//2])
plt.title('Spectre de Fréquences')
plt.xlabel('Fréquence (Hz)')
plt.ylabel('Amplitude')
plt.grid()
plt.xlim(0, 200)  # Limiter l'axe des X pour voir les fréquences pertinentes
plt.show()
```

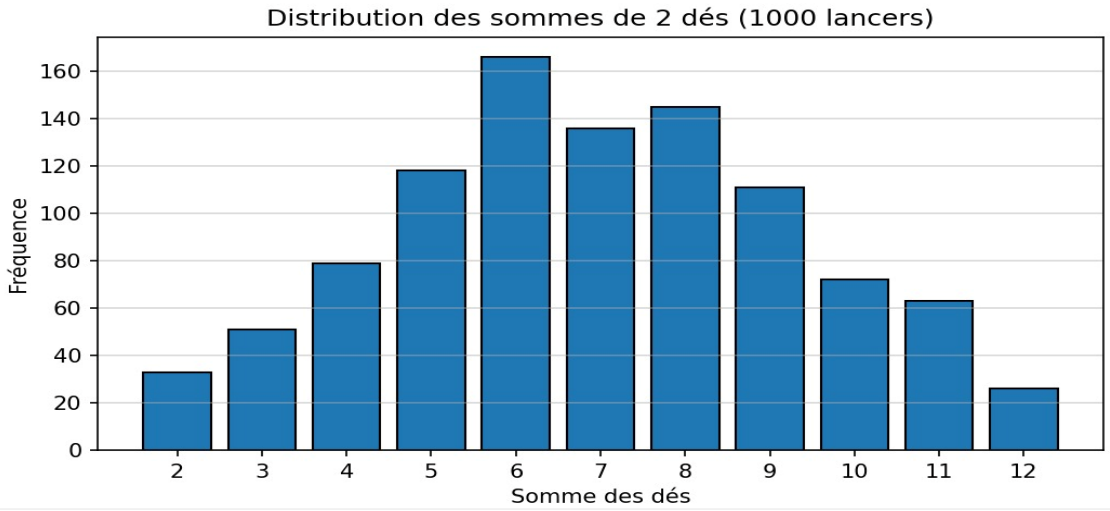
D’après la compilation de code voilà les résultats afficher :



Résultats

Le graphique ci-dessus montre le spectre de fréquences du signal. On peut observer des pics d'amplitude aux fréquences de 50 Hz et 120 Hz, confirmant la présence des sinusoïdes dans le signal.

5- Simulez 1000 lancers de deux dés et affichez l'histogramme des sommes obtenues.



FIN