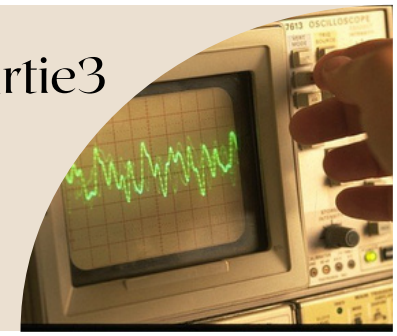
Compte rendu sur le projet TNS : Partie3

Filière : GSTR1

CHEMMAM FATIMA EZZAHRA



Partie 3 : Documentation du Code : Traitement de Signal avec Filtrage Passe-Bas et Analyse FFT

Objectif :

Le code simule un signal périodique composé d'impulsions, applique un filtre passe-bas et analyse le signal avant et après filtrage à l'aide de la transformée de Fourier rapide (FFT).

1. Importation des Bibliothèques :

```
[50]: import numpy as np
      import matplotlib.pyplot as plt
      from scipy.signal import butter, lfilter
```

- numpy : Utilisé pour la manipulation de données numériques et les calculs mathématiques.
- matplotlib.pyplot : Utilisé pour la création de graphiques et la visualisation des signaux.
- scipy.signal : Contient des fonctions pour la conception de filtres et le traitement du signal.

2. Création du Signal Périodique :

```
sampling_frequency = 8000  # Hz (sampling rate)
duration = 0.5  # seconds (duration of the signal)
t = np.linspace(0, duration, int(sampling_frequency * duration), endpoint=False)
```

- sampling_frequency : La fréquence d'échantillonnage est de 8000 Hz, ce qui signifie que le signal est échantillonné 8000 fois par seconde.
- duration : La durée du signal est de 0.5 seconde.
- t : Vecteur temporel généré avec np.linspace pour représenter le temps, de 0 à 0.5 seconde, avec un échantillonnage à 8000 Hz.

3. Rectification du Signal :

```
signal = np.zeros_like(t)
impulse_indices = np.arange(0, len(t), step=200)  # Impulse every 200 samples
signal[impulse_indices] = 1  # Set impulses to amplitude 1
```

- Rectification du signal pour ne conserver que les valeurs positives.

4. Filtrage Passe-Bas :

```python
# Define the low-pass filter function
def low_pass_filter(signal, cutoff_frequency, sampling_rate):
    nyquist = sampling_rate / 2
    normal_cutoff = cutoff_frequency / nyquist
    b, a = butter(4, normal_cutoff, btype='low', analog=False)
    filtered_signal = lfilter(b, a, signal)
    return filtered_signal

# Apply low-pass filtering
cutoff_frequency = 400  # Hz
filtered_signal = low_pass_filter(rectified_signal, cutoff_frequency, sampling_frequency)
```

- La fonction low_pass_filter applique un filtre passe-bas à l'aide de la fonction butter de SciPy.
- Le cutoff_frequency définit la fréquence de coupure du filtre.
- normal_cutoff est la fréquence de coupure normalisée par rapport à la fréquence de Nyquist (moitié de la fréquence d'échantillonnage).
- lfilter applique ce filtre au signal rectifié.

5. Application de la FFT :

```python
# Compute FFT for the original and filtered signals
fft_rectified = np.fft.fft(rectified_signal)
fft_filtered = np.fft.fft(filtered_signal)
frequencies_fft = np.fft.fftfreq(len(fft_rectified), 1 / sampling_frequency)

# Plot the results
plt.figure(figsize=(12, 8))
```

- fft_rectified et fft_filtered contiennent les coefficients de la FFT pour les signaux rectifiés et filtrés respectivement.
- frequencies_fft génère les fréquences correspondantes aux composantes du signal dans le domaine fréquentiel, en utilisant np.fft.fftfreq.

6. Visualisation des Résultats :

```python
# Original rectified signal in the time domain
plt.subplot(3, 1, 1)
plt.plot(t, rectified_signal, label="Rectified Signal", color='magenta')
plt.title("Rectified Signal (Time Domain)")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.legend()

# Frequency domain (rectified signal)
plt.subplot(3, 1, 2)
plt.stem(frequencies_fft[:len(frequencies_fft)//2], np.abs(fft_rectified[:len(fft_rectified)//2]), basefmt=" ", use_line_collection=True, label="FFT of FFT of Rectified Signal")
plt.title("Frequency Spectrum of Rectified Signal")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Amplitude")
plt.legend()

# Frequency domain (filtered signal)
plt.subplot(3, 1, 3)
plt.stem(frequencies_fft[:len(frequencies_fft)//2], np.abs(fft_filtered[:len(fft_filtered)//2]), basefmt=" ", use_line_collection=True, label="FFT of FiT of Filtered Signal")
plt.title("Frequency Spectrum of Filtered Signal")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Amplitude")
plt.legend()

plt.tight_layout()
plt.show()
```

Résultats de l'Exécution du Code :

- À l'issue de l'exécution du code, les résultats obtenus sont les suivants :
- Signal rectifié dans le domaine temporel : Un graphique montre le signal périodique rectifié, où seules les valeurs positives sont conservées.
- Spectre de fréquence du signal rectifié : Un graphique de la transformée de Fourier rapide (FFT) du signal rectifié révèle les fréquences présentes dans le signal avant le filtrage.
- Spectre de fréquence du signal filtré : Après application du filtre passe-bas, un autre graphique de la FFT montre l'atténuation des fréquences supérieures à 400 Hz, illustrant l'effet du filtrage.
- Ces résultats permettent d'observer l'impact du filtrage sur la composition fréquentielle du signal.